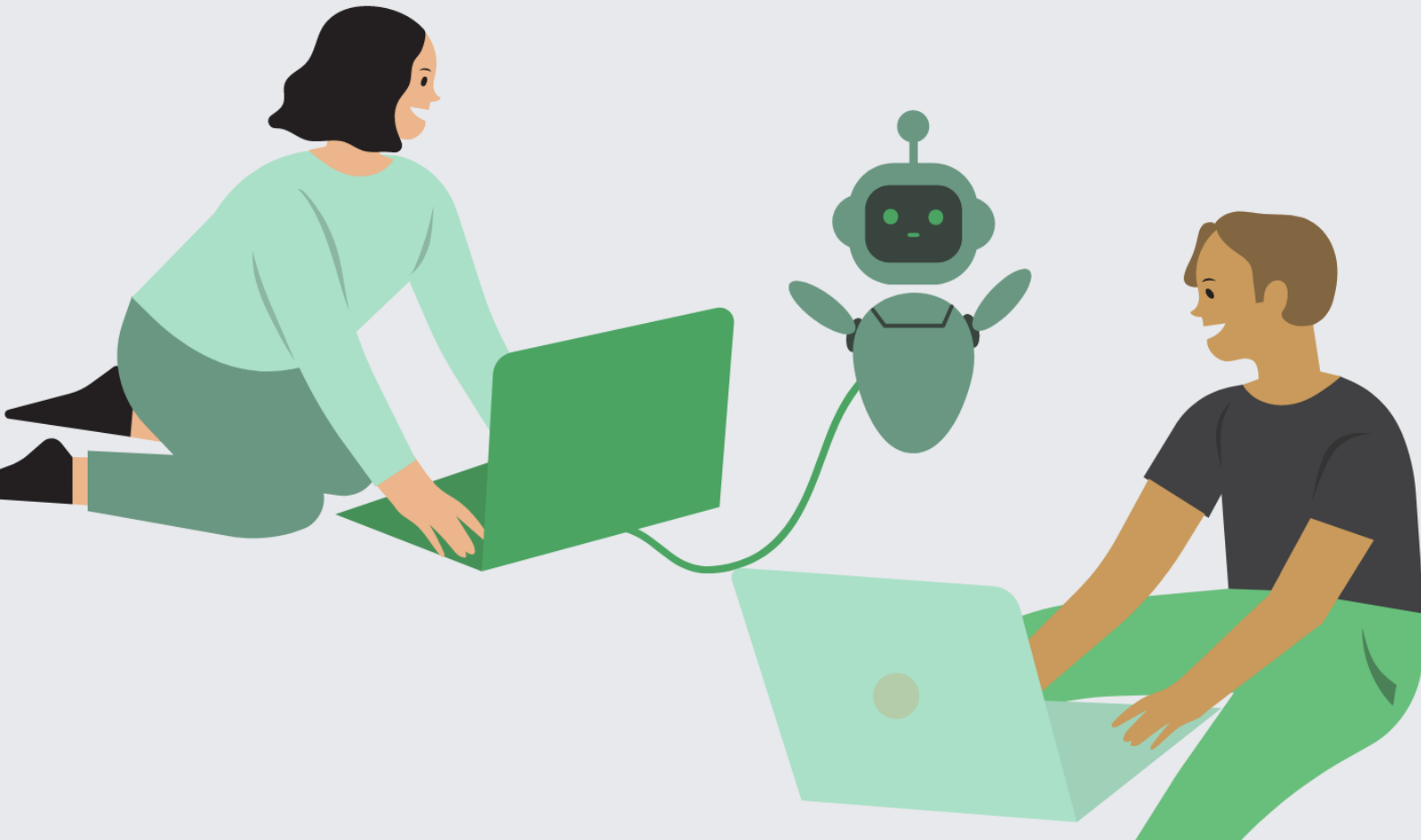
Με τη συγχρηματοδότηση
της Ευρωπαϊκής Ένωσης

Η Ενότητα του CodER στον Προγραμματισμό & Μικροελεγκτές: 20 ώρες μαθήματος



Περιεχόμενα

Εισαγωγή 4
1. Το πλαίσιο του έργου CodER 5
1.1. Εξετάζοντας την ευρύτερη εικόνα: Η παιχνιδοκεντρική μάθηση και ο ρόλος της στην εκπαίδευση 5
1.2. Ο στόχος της παιχνιδοκεντρικής μάθησης αναφορικά με την απόκτηση γνώσεων στην κωδικοποίηση και τους μικροελεγκτές 7
1.3. Το κοινό και η ομάδα-στόχος των δραστηριοτήτων του έργου 10
ΜΕΡΟΣ Α: Ενότητα του έργου CodER στον Προγραμματισμό (10 ώρες) 13
1. Εισαγωγή στον προγραμματισμό 14
1.1. Τι είναι προγραμματισμός 14
1.2. Γιατί να μάθουμε προγραμματισμό; Ποια είναι τα οφέλη; 14
1.3. Η διαδικασία ανάπτυξης προγραμμάτων και οι αλγόριθμοι 15
1.4. Γλώσσες προγραμματισμού – Οι πιο απαιτητικές γλώσσες προγραμματισμού 18
2. Εγκατάσταση της Python 19
2.1. Τι είναι η Python; 19
2.2. Τα πιο κοινόχρηστα Περιβάλλοντα Ολοκληρωμένης Ανάπτυξης της Python (IDE) ανά κλάδο 21
2.3. Η εγκατάσταση ενός Ολοκληρωμένου Περιβάλλοντος Ανάπτυξης της Python (IDEs) 21
2.4. Προγράμματα Python που τρέχουν από τη γραμμή εντολών 27
3. Βασικές έννοιες στον προγραμματισμό με Python 28
3.1. Βασικές έννοιες: Τύποι δεδομένων (Βασικοί και Σύνθετοι) 28
3.2. Μεταβλητές, εκφράσεις και δηλώσεις 29
3.3. Λίστες, λεξικά και πλειάδες 35
3.3.1. Λίστες 35
3.3.2. Πλειάδες 41
3.3.3. Λεξικά 46
3.4. Συνθήκες και βρόχοι 50
3.4.1. Εκφράσεις Λογικό Ή, ΚΑΙ, ΟΧΙ (and, or, not) 50
3.4.2. Λογικοί τελεστές 53
3.4.3. Δηλώσεις nested if 53
3.4.4. Βρόχοι 54
3.5. Κατανόηση της ροής της εκτέλεσης μέσω συναρτήσεων 57
3.6. Συναρμολογώντας το παζλ - Πώς να δημιουργήσετε ένα πρόγραμμα 62
3.7. Πώς να προσαρμόσετε ένα πρόγραμμα ώστε να ανταποκρίνεται στις ανάγκες σας 66

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

3.8. Συντακτικά λάθη, Σφάλματα Χρόνου Εκτέλεσης και Σημασιολογικά λάθη – Διαχείριση σφαλμάτων στην Python 67
3.9. Πρακτική 69
ΜΕΡΟΣ Β: Ενότητα του έργου CodER στους Μικροελεγκτές (10 ώρες) 71
1. Εισαγωγή στους μικροελεγκτές 72
1.1. Τι είναι ένας μικροελεγκτής 72
1.2. Τι είναι το Arduino και οι διάφοροι τύποι του 74
1.3. Βασικές έννοιες: Είσοδος, έξοδος, αναλογική, ψηφιακή 79
2. Βασικές έννοιες στο προγραμματισμό με Arduino 82
2.1. Ρύθμιση του IDE Arduino και βασικές εντολές 82
2.2. Ο Προγραμματισμός στο Arduino και η μεταφόρτωση προγραμμάτων στην πλατφόρμα 85
2.3. Αναβόσβημα ενός Λαμπτήρα Led με Arduino 88
3. Εφαρμογές 92
3.1. Τι είναι η ρομποτική; 92
3.2 Τύποι ρομπότ 93
3.3 Οδήγηση κινητήρα συνεχούς ρεύματος (DC motor) με ασπίδα κινητήρα (shield motor) 98
3.4. Η κατασκευή μιας μηχανής paintbot με τη χρήση κινητήρα DC και Arduino 102
3.5 Η κατασκευή ενός διαδραστικού παιχνιδιού από χαρτί με έναν σερβοκινητήρα 105
3.6. Ηλεκτρική κλειδαριά ασφαλείας με τηλεχειριστήριο 112
3.7. Μέτρηση θερμοκρασίας, υγρασίας, φωτός και χρώματος 118
3.7.1. Η χρήση ενός αισθητήρα με το Arduino 119
3.7.2. Κατασκευή ενός θέρεμιν με Arduino και έναν αισθητήρα φωτός 120
3.7.3. Ρομπότ διαλογής αντικειμένων με βάση το χρώμα 122
3.7.4. Εισαγωγή στον αισθητήρα τύπου DHT11 124
3.7.5. Η κατασκευή ενός έξυπνου ανεμιστήρα με ψύξη 126

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Εισαγωγή

Στον σύγχρονο κόσμο που ζούμε, υπάρχει μια σαφώς αυξανόμενη έμφαση στην απόκτηση ψηφιακών δεξιοτήτων. Το υφιστάμενο χάσμα μεταξύ προσφοράς και ζήτησης στην αγορά εργασίας παρουσιάζει νέες προκλήσεις τόσο για τους εργοδότες όσο και για τα άτομα που αναζητούν εργασία. Το εν λόγω χάσμα επηρεάζει πρωτίστως τους νέους, καθώς η νεανική ανεργία είναι ένα από τα σημαντικότερα ζητήματα που αντιμετωπίζει η Ευρώπη. Η σύμπραξη του έργου CodER αντιπροσωπεύει μια προσπάθεια γεφύρωσης του προαναφερθέντος χάσματος μέσω της εφαρμογής της καινοτόμου μεθόδου των Δωματίων Απόδρασης για την παροχή γνώσεων σχετικά με τον προγραμματισμό και τους μικροελεγκτές σε εργαζόμενους και οργανισμούς στον τομέα της νεολαίας. Στόχος των Δωματίων Απόδρασης είναι η εκπαίδευση των νέων και η αύξηση του ενδιαφέροντος και της ενασχόλησής τους σε επαγγέλματα που στρέφονται προς την τεχνολογία. Το έργο CodER συγχρηματοδοτείται από το πρόγραμμα Erasmus+ και οι Ευρωπαίοι εταίροι που συμμετέχουν στο έργο είναι: Digijeunes (Γαλλία), Challedu (Ελλάδα), Citizens in Power – CIP (Κύπρος), RITE (Κύπρος), AKMI (Ελλάδα) και Kalimera (Κροατία).

Μέσω του έργου αυτού, οι εργαζόμενοι και οι οργανισμοί στον τομέα της νεολαίας θα εφοδιαστούν με νέες δεξιότητες και καινοτόμες μεθόδους στην προσπάθεια προσέλκυσης νεαρών ατόμων και ιδιαίτερα νέων γυναικών σε επαγγέλματα που στρέφονται προς την τεχνολογία. Η Ενότητα του CodER αντιπροσωπεύει το πρώτο βήμα προς την υλοποίηση των στόχων που έθεσε το έργο για το ευρύ κοινό. Η ενότητα αυτή πραγματεύεται τις βασικές έννοιες του προγραμματισμού και των μικροελεγκτών μέσω έμπρακτων παραδειγμάτων, ούτως ώστε οι εκπαιδευόμενοι να είναι σε θέση να συσχετίζουν το περιεχόμενο με παραδείγματα από την καθημερινή ζωή. Η ενότητα προσπαθεί να εμφυσήσει στους εκπαιδευόμενους τη λογική πίσω από τον προγραμματισμό και τους μικροελεγκτές και να καλλιεργήσει, μεταξύ άλλων, την κριτική τους σκέψη, τη δημιουργικότητά τους και τις δεξιότητές τους στην επίλυση προβλημάτων.

Αρχικά, θα παρουσιάσουμε το πλαίσιο και την προσέγγιση του έργου CodER, ούτως ώστε οι αναγνώστες να κατανοήσουν το σκεπτικό της προσέγγισης που ακολουθήσαμε σε σχέση με την παιχνιδοκεντρική μάθηση. Το πρώτο μέρος της ενότητας καταπιάνεται με την κατανόηση του τρόπου χρήσης και των πλεονεκτημάτων του προγραμματισμού, την εγκατάσταση του Ολοκληρωμένου Περιβάλλοντος Ανάπτυξης (IDE) της Python και την εφαρμογή των γνώσεων γύρω από αυτήν μέσω της δημιουργίας ενός μικρού προγράμματος. Το δεύτερο μέρος της ενότητας πραγματοποιεί μια εισαγωγή στους μικροελεγκτές και τον τρόπο χρήσης τους, την εγκατάσταση της πλατφόρμας του Arduino και πώς αυτό μπορεί να χρησιμοποιηθεί για την εκτέλεση μικρών εργασιών με μικροελεγκτές.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

1. Το πλαίσιο του έργου CodER

1.1. Εξετάζοντας την ευρύτερη εικόνα: Η παιχνιδοκεντρική μάθηση και ο ρόλος της στην εκπαίδευση

Την τελευταία δεκαετία, υπήρξε μια μεταστροφή από τις παραδοσιακές μεθόδους διδασκαλίας με επίκεντρο τον εκπαιδευτικό σ' αυτό που ονομάζουμε μαθητοκεντρική μάθηση στην εκπαίδευση. Ενώ στις παραδοσιακές μεθόδους διδασκαλίας «οι μαθητές γίνονται παθητικοί αποδέκτες της γνώσης που μεταδίδει ο εκπαιδευτικός» (McGuire & Gubbins, 2010, σελ. 1), στην μαθητοκεντρική μάθηση, αντιθέτως, οι μαθητές αποκτούν ενεργό ρόλο στην πρόσληψη της γνώσης και είναι σε θέση να αξιολογούν τη μαθησιακή τους εμπειρία (...) εφόσον ο εκπαιδευτικός δεν είναι πλέον «προφήτης», αλλά καθοδηγητής στη διαδικασία της μάθησης» (McGuire & Gubbins, 2010, σ.1). Οι σύγχρονες προσεγγίσεις στη μάθηση βασίζονται κυρίως στην τεχνολογία και την εφαρμογή διαδραστικών μέσων, που στόχο έχουν να δώσουν περισσότερα κίνητρα στους μαθητών, ούτως ώστε να έχουν ένα πιο ενεργό ρόλο κατά τη μαθησιακή διαδικασία.

Ένας λόγος για την υιοθέτηση αυτών των νέων προσεγγίσεων στην εκπαίδευση που βασίζονται στην τεχνολογία και ειδικά στην εκπαίδευση STEM, είναι ότι ένας σημαντικός αριθμός θέσεων εργασίας επικεντρώνεται πλέον σε δραστηριότητες και εργασίες που απαιτούν τη χρήση του διαδικτύου και των ψηφιακών τεχνολογιών (Daniel Calderón-Gómez et al. 2020), ενώ στο εγγύς μέλλον ακόμη περισσότερες θέσεις εργασίας θα πρέπει να υιοθετήσουν αυτό το πρότυπο. Αυτό σημαίνει ότι οι νέες θέσεις εργασίας θα απαιτούν την απόκτηση τουλάχιστον ορισμένων βασικών ψηφιακών δεξιοτήτων, ενώ οι πιο ευνοϊκές και υψηλά αμειβόμενες θέσεις εργασίας θα απαιτούν ακόμη πιο προηγμένες γνώσεις και δεξιότητες των ψηφιακών τεχνολογιών (Karpinski et al., 2021). Ένας άλλος παράγοντας που έπαιξε ρόλο στη μεταστροφή προς την μαθητοκεντρική εκπαίδευση, εκτός από τις ραγδαίες τεχνολογικές εξελίξεις, είναι το φαινόμενο της έλλειψης κινήτρων, των χαμηλών επιδόσεων και ποσοστών συμμετοχής των μαθητών σε ποικίλα εκπαιδευτικά περιβάλλοντα (όπως αναφέρεται στο Levels et al., 2022 Callanan et al., 2009). Στην περίπτωση των νέων ατόμων ηλικίας 15 έως 29 ετών, το φαινόμενο αυτό είναι ιδιαίτερα ανησυχητικό σύμφωνα με τα διαθέσιμα στατιστικά στοιχεία στην Ευρώπη, όπου το 13,7% των νέων δε είναι ενεργό ούτε στον τομέα της εκπαίδευσης αλλά ούτε στον επαγγελματικό τομέα ή σε κάποιο πρόγραμμα επαγγελματικής κατάρτισης (ΕΕΑΚ: εκτός εκπαίδευσης, απασχόλησης, ή επαγγελματικής κατάρτισης) (Eurofound, 2022). Ως εκ τούτου, οι υπεύθυνοι χάραξης πολιτικής και οι οργανώσεις νεολαίας βρέθηκαν αντιμέτωποι με νέες προκλήσεις στην προσπάθειά να κινητοποιήσουν και να επανεντάξουν τους νέους στους τομείς της εκπαίδευσης, της απασχόλησης ή/και της επαγγελματικής κατάρτισης.

Σε διάφορα επίπεδα εκπαίδευσης, η μάθηση που βασίζεται στο παιχνίδι άρχισε να κερδίζει ολοένα και μεγαλύτερη δημοτικότητα τόσο στο πλαίσιο της τυπικής όσο και της μη τυπικής και άτυπης εκπαίδευσης. Μέσα στο πλαίσιο αυτό, προέκυψαν οι ορισμοί της

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

«παιχνιδοποίησης» και της «μάθησης βασισμένης στο παιχνίδι». Αν και οι δύο όροι χρησιμοποιούνται συχνά εναλλακτικά, παρουσιάζουν κάποιες μικρές διαφορές μεταξύ τους. Με τον όρο «παιχνιδοποίηση», συνήθως αναφερόμαστε στη «χρήση στοιχείων σχεδίασης παιχνιδιών σε περιβάλλοντα εκτός παιχνιδιού» (όπως αναφέρεται στους Dichev & Dicheva, 2017, σ. 2; Deterding, et al., 2011; Hamari et al., 2014 Werbach, 2014). Στο πλαίσιο της εκπαίδευσης, η παιχνιδοποίηση αναφέρεται στην εφαρμογή «στοιχείων σχεδίασης ενός παιχνιδιού και εμπειρίας ενός παιχνιδιού στο πλαίσιο ενός μαθησιακού στόχου» (Dichev & Dicheva, 2017, σ.2). Η παιχνιδοποίηση χρησιμοποιεί στοιχεία όπως τη δυναμική, τους μηχανισμούς και το σκεπτικό ενός παιχνιδιού (Dichev & Dicheva, 2017). Από την άλλη πλευρά, η μάθηση με βάση το παιχνίδι (αγγλικό:GBL) χρησιμοποιεί τις τεχνικές που εφαρμόζουν οι σχεδιαστές παιχνιδιών για να δημιουργήσουν μια ευχάριστη και καθηλωτική εμπειρία για τον χρήστη, με μοναδικό στόχο το σχεδιασμό ενός εικονικού περιβάλλοντος μάθησης στο οποίο ο χρήστης συμμετέχει σε εκπαιδευτικές δραστηριότητες. Αυτός ο τύπος μάθησης μπορεί να πραγματοποιηθεί τόσο δια ζώσης όσο και διαδικτυακά. Η μάθηση με βάση το παιχνίδι θεωρείται ότι έχει σαφώς καθορισμένα μαθησιακά αποτελέσματα τα οποία επιτυγχάνονται μέσω του περιεχομένου του παιχνιδιού (Sanchez, 2019). Αυτός ο τύπος μάθησης είναι γνωστός επίσης με τους όρους «σοβαρά παιχνίδια» (serious games), «ψηφιακή μάθηση» ή «εκπαιδευτικά παιχνίδια» (Sanchez, 2019).

Υπάρχει ένα ευρύ σύνολο διαθέσιμων ερευνών που μελέτησαν τα αποτελέσματα της παιχνιδοκεντρικής μάθησης σε άτομα διαφορετικών ηλικιών, από μαθητές της πρωτοβάθμιας και δευτεροβάθμιας εκπαίδευσης μέχρι φοιτητές της τριτοβάθμιας εκπαίδευσης και, σε κάποιες περιπτώσεις, σε ενήλικες (π.χ. Dichev & Dicheva, 2017; Ninaus et al., 2017; Sanchez et al., 2020). Η κύρια αιτία της μεταστροφής αυτής προς την παιχνιδοκεντρική μάθηση έγκειται στη δυνατότητα που αυτή παρέχει σχετικά με την παροχή κινήτρων στους μαθητές νεαρότερων κυρίως ηλικιών, να αποκτήσουν πιο ενεργό ρόλο συμμετοχής στην εκπαίδευσή τους. Όπως αναφέρεται από τους Dichev & Dicheva (2017), τέτοιες προσεγγίσεις στη μάθηση προϋποθέτουν «την εμπειρία εμβύθισης του ατόμου κατά τρόπο παρόμοιο με εκείνο που συμβαίνει όταν παίζουμε ένα παιχνίδι» (Dichev & Dicheva, 2017, σ.2). Δεδομένου ότι τα βιντεοπαιχνίδια έχουν σχεδιαστεί κυρίως για ψυχαγωγικούς σκοπούς, «μπορούν να φέρουν τον χρήστη σε καταστάσεις επιθυμητών εμπειριών, παρακινώντας τον να παραμείνει απορροφημένος σε μια δραστηριότητα» (Dichev & Dicheva, 2017, σ. 12). Οι περισσότερες μελέτες αναδεικνύουν ως πιο σημαντικό παράγοντα στην ενασχόληση και την επιτυχή ολοκλήρωση δραστηριοτήτων που απαιτούν χρόνο και προσπάθεια, την παροχή κινήτρων στους νέους. Μέσω της παιχνιδοκεντρικής μάθησης, η έννοια του χρόνου και της προσπάθειας μετατρέπεται σε ένα εκπαιδευτικό και ψυχαγωγικό περιβάλλον που επιτρέπει στους χρήστες να αξιοποιήσουν ταυτόχρονα τις γνώσεις, τις δεξιότητες και τις στάσεις τους.

Η απλή πρακτική της τοποθέτησης μιας δραστηριότητας ή μιας εργασίας στο πλαίσιο ενός παιχνιδιού αρκεί για να επιφέρει θετικά αποτελέσματα στην ψυχολογία ενός ατόμου, όπως

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

αυτά της αφοσίωσης και της απόλαυσης (Dichev & Dicheva, 2017). Η άποψη αυτή φαίνεται να βασίζεται στην υπόθεση ότι οι άνθρωποι έχουν εγγενώς την προδιάθεση, η οποία πηγάζει από μια ψυχολογική τους ανάγκη, να ικανοποιούν «το αίσθημα της αυτοπεποίθησης για τις ικανότητές τους ή το αίσθημα της αυτοαποτελεσματικότητας» (Ninaus et al., 2017, σ. 16), ένα στοιχείο που ενυπάρχει στην επίλυση ενός προβλήματος ή στο πλαίσιο της ολοκλήρωσης του στόχου ενός παιχνιδιού. Η δυνατότητα των παιχνιδιών να κεντρίζουν την προσοχή των παικτών και να τους απορροφούν όσο τους εμπλέκουν παράλληλα σε δοκιμασίες επίλυσης προβλημάτων, είναι σύμφυτη με τη βασική ιδέα πίσω από τον σχεδιασμό παιχνιδιών, που δεν είναι άλλη από την καθήλωση των παικτών σε μια κατάσταση κατά την οποία καλούνται να ολοκληρώσουν τους στόχους ενός σεναρίου ή μιας αποστολής. Σύμφωνα με τον Tulloch (2014), συχνά παραγνωρίζεται η παιδαγωγική διάσταση που προδιαθέτει το παιχνίδι και αυτή αφορά τη «διαδικασία κατά την οποία ο χρήστης μαθαίνει πώς να παίζει το παιχνίδι» (όπως αναφέρεται στους Dichev & Dicheva, 2017, σ. 23). Επομένως, η παιχνιδοκεντρική μάθηση επιδιώκει να προσομοιώσει την άκρως αυτή καθηλωτική εμπειρία που έχει ένα χρήστης όταν παίζει ένα παιχνίδι, στο πλαίσιο όμως ενός εκπαιδευτικού σκοπού.

1.2. Ο στόχος της παιχνιδοκεντρικής μάθησης αναφορικά με την απόκτηση γνώσεων στην κωδικοποίηση και τους μικροελεγκτές

Από τότε που ξεκίνησε η πανδημία, η έμφαση που δόθηκε στην ενσωμάτωση της τεχνολογίας σε διάφορους τομείς έγινε ακόμη πιο έντονη. Το όραμα της Ευρωπαϊκής Επιτροπής που παρουσιάστηκε το 2021 περιλαμβάνει τέσσερα σημεία ζωτικής σημασίας στο πλαίσιο της επίτευξη του στόχου για τον ψηφιακό μετασχηματισμό της Ευρώπης μέχρι το 2030. Ένα από αυτά τα τέσσερα σημεία ζωτικής σημασίας που έθεσε η Ευρωπαϊκή Επιτροπή είναι η εδραίωση των ψηφιακών ικανοτήτων ανάμεσα στους πολίτες της, η οποία θα συμβάλει στην οικοδόμηση μιας ανθεκτικής ευρωπαϊκής οικονομίας και κοινωνίας. Ο κύριος στόχος που έθεσε η Ευρωπαϊκή Επιτροπή σε σχέση με τον ψηφιακό μετασχηματισμό της Ευρώπης μέχρι το 2030, είναι το 80% των πολιτών της να αποκτήσει βασικές ψηφιακές δεξιότητες (Ευρωπαϊκή Επιτροπή, 2021α).

Ωστόσο, το μεγαλύτερο εμπόδιο στην υλοποίηση του ευρωπαϊκού ψηφιακού μετασχηματισμού μέχρι το 2030 είναι η δυσαναλογία που προκύπτει μεταξύ της ανεπάρκειας ψηφιακών δεξιοτήτων σε σχέση με την υψηλή τους ζήτηση στην αγορά εργασίας. Το ποσοστό των εργοδοτών που αντιμετωπίζουν προβλήματα έλλειψης ψηφιακά καταρτισμένου υπαλληλικού προσωπικού φτάνει το 70% (Ευρωπαϊκή Επιτροπή, 2021b). Αυτό διευρύνει το χάσμα των ψηφιακών δεξιοτήτων που υπάρχει στην αγορά εργασίας, όπου ακόμη και τα επαγγέλματα χαμηλής ειδίκευσης έως ημιειδίκευσης απαιτούν ένα επίπεδο ψηφιακών δεξιοτήτων (Karpinski et al., 2021). Όπως αναφέρεται στον Δείκτη της Ψηφιακής Οικονομίας και Κοινωνίας (DESI), το 56% των Ευρωπαίων πολιτών διαθέτει βασικές ψηφιακές δεξιότητες, αλλά μόνο το 31% διαθέτει ψηφιακές δεξιότητες πάνω από το βασικό επίπεδο (Ευρωπαϊκή Επιτροπή, 2021β). Έτσι, η σημασία των δεξιοτήτων

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

επίλυσης προβλημάτων και της υπολογιστικής σκέψης βρίσκεται ψηλά στην ατζέντα της Ευρωπαϊκής Επιτροπής, καθώς μόνο το 13% των νέων και το 6% των ενηλίκων ατόμων διαθέτουν τέτοιες δεξιότητες στην ΕΕ (Eurostat, 2020).

Το Ευρωπαϊκό Πλαίσιο Ψηφιακών Ικανοτήτων για τους Πολίτες (DigComp),ορίζει την ψηφιακή ικανότητα ως συνδυασμό 21 ικανοτήτων που ομαδοποιούνται σε πέντε βασικούς τομείς: 1) Παιδεία πληροφοριών και δεδομένων, 2) Επικοινωνία και συνεργασία, 3) Δημιουργία ψηφιακού περιεχομένου, 4) Ασφάλεια και 5) Επίλυση προβλημάτων (Centeno et al., 2019). Υποστηρίζεται ότι οι νέες τεχνολογικές απαιτήσεις των επαγγελμάτων θα δημιουργήσουν ένα νέο διαχωρισμό στην αγορά εργασίας, που θα περιλαμβάνει τρεις διαφορετικές κατηγορίες εργαζομένων με βάση τις ψηφιακές τους ικανότητες: «το ψηφιακό πρεκαριάτο» (αποτελούμενο από εργαζομένους σε αντίξοες οικονομικές συνθήκες)· «η παραδοσιακή ψηφιακή εργασία» (αποτελούμενη κυρίως από εργαζομένους που ασχολούνται με παραγωγικές ψηφιακές εργασίες) και την «καινοτόμο τάξη» (αποτελούμενη από εργαζομένους που ασχολούνται με παραγωγικές και επικοινωνιακές ψηφιακές εργασίες)» (Calderón-Gómez et al., 2020, σ. 7-8). Επιπλέον, έχει προταθεί ότι το «ψηφιακό πρεκαριάτο» θα αποτελείται κυρίως από νέους και γυναίκες (Calderón-Gómez et al., 2020, σελ. 9). Υπό το πρίσμα αυτό, οι νέοι που ήδη κινδυνεύουν με κοινωνικό και επαγγελματικό αποκλεισμό λόγω της αδράνειας ή της αδυναμίας τους να βρουν εργασία ή να συνεχίσουν περαιτέρω την εκπαίδευση και την επαγγελματική τους κατάρτιση βρίσκονται στο σταυροδρόμι ενός διττού αποκλεισμού.

Ο Corneliussen (2021) τονίζει επίσης ότι η έλλειψη γυναικείων προτύπων σε επαγγέλματα ΤΠΕ (Τεχνολογία Πληροφοριών και Επικοινωνίας) τα οποία συχνά μονοπωλούνται από άνδρες, αποτελεί έναν αποθαρρυντικό παράγοντα για τα νεαρά κορίτσια και τις γυναίκες που θέλουν να επιδιώξουν την επαγγελματική τους σταδιοδρομία στον τομέα STEM. Σε γενικές γραμμές, οι ευνοϊκότερες και πιο ακριβοπληρωμένες θέσεις εργασίας πληρούνται ως επί το πλείστων από άνδρες παρά από γυναίκες (Calderón-Gómez et al., 2020). Όπως αναφέρθηκε και πιο πάνω, οι γυναίκες μαζί με τους νέους είναι οι κυρίαρχες ομάδες του «ψηφιακού πρεκαριάτου». Ταυτόχρονα, οι γυναίκες «υποεκπροσωπούνται στην καινοτόμο τάξη, ενώ υπερεκπροσωπούνται και στην παραδοσιακή ψηφιακή εργασία» (Calderón-Gómez et al., 2020, σ. 8). Μια μελέτη καταδεικνύει ότι οι άνδρες αφιερώνουν περισσότερο χρόνο στο δδιαδίκτυο για δραστηριότητες που σχετίζονται με την εργασία απ' ό,τι οι γυναίκες, ενώ μάλιστα μεταξύ «όσων είναι κάτοχοι πανεπιστημιακού τίτλου, το 77,5% των ανδρών σε σύγκριση με το 70,0% των γυναικών χρησιμοποιούν το διαδίκτυο στην εργασία». (Calderón-Gómez et al., 2020, σελ.18-20). Επιπλέον, οι γυναίκες τείνουν να χρησιμοποιούν τις ψηφιακές τεχνολογίες περισσότερο για επικοινωνία και για χρήση κινητού τηλεφώνου, ενώ οι άνδρες τείνουν να χρησιμοποιούν τις ψηφιακές τεχνολογίες περισσότερο για παραγωγικούς σκοπούς, όπως για γραφειακές εργασίες και για πολλαπλές λειτουργίες και χρήσεις (Calderón-Gómez et al. 2020). Γενικά, σε σύγκριση με τους άνδρες, οι γυναίκες είναι λιγότερο πιθανό να χρησιμοποιούν το διαδίκτυο ή/και τις ψηφιακές τεχνολογίες για πιο σύνθετες χρήσεις.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Ως εκ τούτου, οι γνώσεις και οι δεξιότητες στον προγραμματισμό και τους μικροελεγκτές έχουν μεγάλη σημασία για τις τρέχουσες ανάγκες της αγοράς εργασίας. Η παιχνιδοκεντρική μάθηση στον κλάδο του προγραμματισμού και των μικροελεγκτών μας επιτρέπει να φτάσουμε στην ομάδα-στόχο μας. Σύμφωνα με τα οφέλη που συζητήθηκαν προηγουμένως, η παιχνιδοκεντρική μάθηση λειτουργεί ως μέσο μεταλαμπάδευσης γνώσεων και δεξιοτήτων μέσω μιας ευχάριστης και ουσιαστικής μαθησιακής εμπειρίας. Πιο συγκεκριμένα, στα μαθήματα που έχουν εφαρμοστεί τα δωμάτια απόδρασης ως εργαλείο εκπαίδευσης αποδείχτηκε ότι αυτά έχουν τη δυνατότητα να διεγείρουν θετικά συναισθήματα, να αυξάνουν το επίπεδο συμμετοχής και να δημιουργούν μια, στο σύνολό της, θετική μαθησιακή εμπειρία για τους μαθητές (Llerena-Izquierdo & Sherry, 2022, Sánchez-Martín et al. ., 2020). Τα ενθαρρυντικά αποτελέσματα τέτοιων μελετών είναι ελπιδοφόρα για το έργο CodER, στόχος του οποίου είναι να γεφυρώσει το χάσμα που υπάρχει μεταξύ των ψηφιακών δεξιοτήτων και των αναγκών της αγοράς εργασίας, και να προσφέρει νέες ευκαιρίες σε νέους που κινδυνεύουν με επαγγελματικό και κοινωνικό αποκλεισμό.

Μια άλλη πτυχή που σχετίζεται άμεσα με τη δημιουργία ψηφιακών δωματίων απόδρασης είναι ο περιορισμός των δωματίων απόδρασης που πραγματοποιούνται δια ζώσης. Όπως επισημαίνουν οι Fotaris and Mastoras (2019, p.3) «δεν είναι εφικτό ή νόμιμο να κλειδώνεις ένα υποσύνολο μιας τάξης σε ένα δωμάτιο και να περιμένεις τους παίκτες μέχρι να βγουν έξω έχοντας λύσει τους γρίφους (…), επίσης πολλά δωμάτια απόδρασης που έχουν σχεδιαστεί στο πλαίσιο της τάξης έχουν πλέον απλοποιηθεί και μετατραπεί σε μια μάλλον ομαδική δραστηριότητα κατά την οποία οι παίκτες συγκεντρώνονται γύρω από ένα τραπέζι και προσπαθούν να λύσουν τους γρίφους μέσω μιας σειράς κλειδωμένων κουτιών». Από την άλλη πλευρά όμως, η λύση με τα ψηφιακά δωμάτια απόδρασης στερείται τον καθηλωτικό χαρακτήρα που διέπουν συνήθως τα δωμάτια απόδρασης που πραγματοποιούνται σε φυσικούς χώρους. Υπάρχει επίσης μια σειρά από άλλες προκλήσεις που πρέπει να ξεπεραστούν με την εφαρμογή εκπαιδευτικών δωματίων απόδρασης, που αφορούν κυρίως τη διάθεση σημαντικού χρόνου για την προετοιμασία και το στήσιμο των δωματίων απόδρασης, των χρονικών περιορισμών που πολλές φορές δεν επιτρέπουν τις δοκιμαστικές εφαρμογές τους παιχνιδιού και που μπορούν να οδηγήσουν σε ακόμη περισσότερα προβλήματα, όπως π.χ. η κακή σχεδίαση και η ο μη ισορροπημένος βαθμός δυσκολίας των γρίφων και, τέλος, οι μεγάλες σε μέγεθος ομάδες που περιπλέκουν τον τρόπο με τον οποίο οι μαθητές θα συμμετέχουν στο δωμάτιο απόδρασης ή επηρεάζουν ακόμη και το βαθμό συμμετοχής τους σ’ αυτό (Fotaris & Mastoras, 2019). Από αυτή την άποψη, η ψηφιακή γεννήτρια δημιουργίας δωματίων απόδρασης που θα αποτελέσει και το τελικό παραδοτέου του έργου μας, έχει στόχο να επιλύσει τους περισσότερους από τους προαναφερθέντες περιορισμούς και να επιτρέψει στους επαγγελματίες στον τομέα της νεολαίας να σχεδιάζουν με μεγαλύτερη ευελιξία ψηφιακά δωμάτια απόδρασης, σύμφωνα με τις εκάστοτε ανάγκες των εκπαιδευομένων για τους οποίους προορίζονται.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Επιπλέον, οι Llerena-Izquierdo & Sherry (2022) υπογραμμίζουν ότι η έλλειψη παροχής οδηγών σχεδίασης και εμπειρίας από τους χρήστες είναι δύο περιοριστικοί παράγοντες όσον αφορά την ευρεία προσαρμογή τέτοιων εργαλείων σε εκπαιδευτικά πλαίσια τυπικής, μη τυπικής και άτυπης μάθησης. Αυτή είναι μια άλλη σημαντική πτυχή που προσπαθούμε να επιλύσουμε μέσω της δημιουργίας αυτής της ενότητας ως πρώτου βήματος προς την κατεύθυνση αυτή, ενώ με τα άλλα αναμενόμενα παραδοτέα του έργου προσπαθούμε να εφοδιάσουμε τους επαγγελματίες στον τομέα της νεολαίας με τις απαραίτητες γνώσεις και δεξιότητες για να είναι σε θέση να μεταδώσουν αυτή την νέα γνώση τους νέους.

1.3. Το κοινό και η ομάδα-στόχος των δραστηριοτήτων του έργου

Το κοινό στο οποίο απευθύνεται το έργο CodER είναι μέλη τοπικών, εθνικών και ευρωπαϊκών οργανώσεων νεολαίας. Αυτό περιλαμβάνει όλους τους τύπους επαγγελματιών που εργάζονται σε μια οργάνωση νεολαίας και που ασχολούνται με μειονεκτούσες ομάδες, όπως τα άτομα που εμπίπτουν στην κατηγορία ΕΕΑΚ, δηλαδή αυτούς που εγκαταλείπουν πρόωρα το σχολείο, τους μετανάστες, τους πρόσφυγες, τους αιτούντες άσυλο και πρώην κρατούμενους. Το έργο μας απευθύνεται επίσης σε οργανώσεις νεολαίας που επικεντρώνονται στην παροχή μη τυπικής εκπαίδευσης σε θέματα που αφορούν την τεχνολογία και την καινοτομία, σε οργανώσεις που επιδιώκουν να αυξήσουν την απασχολισημότητα των νέων ή που ασχολούνται με την εκπαίδευση STEM, καθώς επίσης και για επαγγελματίες στον ίδιο τομέα που ασχολούνται με τον προγραμματισμό, τους μικροελεγκτές και τα εκπαιδευτικά δωμάτια απόδρασης. Επιπλέον, η ομάδα-στόχος του έργου CodER συμπεριλαμβάνει επαγγελματίες ηλεκτρονικών υπολογιστών, εκπροσώπους νεοφυών επιχειρήσεων στον τομέα των ΤΠΕ, επιστήμονες, ερευνητές, πανεπιστημιακό προσωπικό, διαμορφωτές κοινής γνώμης και άλλους φορείς, γενικότερα, που μπορούν να ασκήσουν επιρροή όπως ενώσεις προγραμματιστών, κέντρα καινοτομίας, εκπρόσωποι κρατικών φορέων, μικρού και μεσαίου μεγέθους επιχειρήσεις, κοινωνικούς και δημόσιους φορείς, επαγγελματίες μάρκετινγκ και δημοτικούς υπαλλήλους.

Μέσα από αυτό το κοινό, στοχεύουμε να προσεγγίσουμε την ομάδα-στόχο μας, που είναι οι νέοι, ούτως ώστε οι οργανισμοί νεολαίας να τους μεταδώσουν τη γνώση που απέκτησαν μέσα από το έργο μας. Η ομάδα-στόχος συμπεριλαμβάνει μεταξύ άλλων, νέους που ανήκουν σε εκτοπισμένους πληθυσμούς (μετανάστες, αιτούντες άσυλο, πρόσφυγες, μειονοτικούς πληθυσμούς), νέους που διατρέχουν γενικά κίνδυνο κοινωνικοοικονομικού αποκλεισμού (που εγκαταλείπουν πρόωρα το σχολείο, ΕΕΑΚ κ.λπ.) και νέους με μαθησιακές δυσκολίες. Όπως αναφέρθηκε και προηγουμένως, ο λόγος για τον οποίο ένας μεγάλος αριθμός νέων σήμερα παλεύει με την ανεργία αφορά την έλλειψη δεξιοτήτων στον προγραμματισμό, τις οποίες αναζητούν σήμερα οι εργοδότες από τους υπαλλήλους τους. Ως εκ τούτου, το έργο CodER έχει σχεδιαστεί με τέτοιο τρόπο, ώστε να διδάξει τις βασικές έννοιες γύρω από τον προγραμματισμό και τους μικροελεγκτές σε οργανώσεις νεολαίας, τις οποίες και θα προμηθέψει με ένα μεθοδολογικό και ένα παιδαγωγικό οδηγό αναφορικά με το πώς να χρησιμοποιούν τα δωμάτια απόδρασης στο πλαίσιο της

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

παιχνιδοκεντρικής εκπαίδευσης των νέων, δημιουργώντας μια σειρά από διαφορετικά σενάρια που μπορούν να αξιοποιηθούν ως πρότυπα ή να προσαρμοστούν ανάλογα, καθώς και μια ψηφιακή γεννήτρια δωματίων απόδρασης.

Αναφορές:

Calderón-Gómez, D., Casas-Mas, B., Urraco-Solanilla, M., & Revilla, J. C. (2020). The labour digital divide: digital dimensions of labour market segmentation. *Work Organisation, Labour & Globalisation, 14*(2), 7-30.

Centeno, C., Vuorikari, R., Punie, Y., OÂ, W., Kluzer, S., Vitorica, A., ... & Bartolome, J. (2019). *Developing digital competence for employability: Engaging and supporting stakeholders with the use of DigComp* (No. JRC118711). Joint Research Centre.

Corneliussen, H. G. (2021). Women empowering themselves to fit into ICT. In *Technology and Women's Empowerment*. Taylor & Francis

Dichev, C. & Dicheva, D. (2017). Gamifying education: what is known, what is believed and what remains uncertain: a critical review. *International Journal of Educational Technology in Higher Education* 14**,** 9. https://doi.org/10.1186/s41239-017-0042-5

DigitalEurope (n.d.). Key indicators for a stronger digital Europe. https://www.digitaleurope.org/key-indicators-for-a-stronger-digital-europe/

European Commission (2021a). *Europe's Digital Decade: digital targets for 2030.* https://ec.europa.eu/info/strategy/priorities-2019-2024/europe-fit-digital-age/europes-digital-decade-digital-targets-2030_en

European Commission (2021b). *Digital skills and jobs*. https://digital-strategy.ec.europa.eu/en/policies/digital-skills-and-jobs

Eurofound (2022). NEETs. https://www.eurofound.europa.eu/topic/neets

Eurostat (2020). *Being young in Europe today – digital world*. https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Being_young_in_Europe_today_-_digital_world&oldid=528990#Information_and_communications_technology_skills

Fotaris, P., & Mastoras, T. (2019). Escape rooms for learning: A systematic review. In *Proceedings of the European Conference on Games Based Learning*, 235-243.

Karpinski, Z., Biagi, F., & Di Pietro, G. (2021). Computational Thinking, Socioeconomic Gaps, and Policy Implications. IEA Compass: Briefs in Education. 12. *International Association for the Evaluation of Educational Achievement*

Levels, M., Brzinsky-Fay, C., Holmes, C., Jongbloed, J., & Taki, H. (2022). The dynamics of marginalized youth: *Not in education, employment, or training around the world.* Taylor & Francis.

Liu, Z.-Y., Shaikh, Z. A., & Gazizova, F. (2020). Using the Concept of Game-Based Learning in Education. *International Journal of Emerging Technologies in Learning (iJET), 15* (14), 53–64. https://doi.org/10.3991/ijet.v15i14.14675

Llerena-Izquierdo, J., & Sherry, L. L. (2022). Combining Escape Rooms and Google Forms to Reinforce Python Programming Learning. In Á. Rocha, P. C. López-López & J. P.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Salgado-Guerrero (Eds.), *Communication, Smart Technologies and Innovation for Society* (pp. 107-116). Springer, Singapore.

Mcguire, D. & Gubbins, C. (2010). The Slow Death of Formal Learning: A Polemic. Human Resource Development Review. 9. 249-265. 10.1177/1534484310371444.

Ninaus, M., Moeller, K., McMullen, J., & Kiili, K. (2017). Acceptance of Game-Based Learning and Intrinsic Motivation as Predictors for Learning Success and Flow Experience. *International Journal of Serious Games*, *4*(3), 15–30.

Sanchez, E. (2019) Game-Based Learning. In: Tatnall A. (eds) Encyclopedia of Education and Information Technologies. Springer, Cham. https://doi.org/10.1007/978-3-319-60013-0_39-1

Sánchez-Martín, J., Corrales-Serrano, M., Luque-Sendra, A., & Zamora-Polo, F. (2020). Exit for success. Gamifying science and technology for university students using escape-room. A preliminary approach. *Heliyon*, *6*(7). https://doi.org/10.1016/j.heliyon.2020.e04340

Vasilescu, M. D., Serban, A. C., Dimian, G. C., Aceleanu, M. I., & Picatoste, X. (2020). Digital divide, skills and perceptions on digitalisation in the European Union—Towards a smart labour market. *PLoS ONE, 15*(4), 1–39. https://doi.org/10.1371/journal.pone.0232032

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

ΜΕΡΟΣ Α: Ενότητα του έργου CodER στον Προγραμματισμό (10 ώρες)

Περιγραφή:

Στην μέρος Α της ενότητας, παρέχεται μια ολοκληρωμένη εισαγωγή στον προγραμματισμό συμπεριλαμβανομένης της χρήσης και την εφαρμογής του μέσω απτών παραδειγμάτων. Στη πρώτη υποενότητα, θα γίνει μια ανάλυση στα οφέλη του προγραμματισμού και της εφαρμογής του στην αγορά εργασίας, καθώς και της διαδικασίας που ακολουθείται από τον αλγόριθμο και της ανάπτυξη ενός προγράμματος. Στη δεύτερη υποενότητα, θα γίνει μια ανάλυση της γλώσσας προγραμματισμού της Python, της χρήσης ενός Ολοκληρωμένου Περιβάλλοντος Ανάπτυξης (IDE) και του τρόπου εγκατάστασής του. Η τελευταία υποενότητα πραγματεύεται στους διάφορους τύπους δεδομένων, τη χρήση και την εφαρμογή τους μέσω παραδειγμάτων, καθώς και την ανάπτυξη μικρών προγραμμάτων.

Φόρτος εργασίας:

10 ώρες

Μαθησιακά αποτελέσματα:

Με το πέρας αυτού του μαθήματος, οι εκπαιδευόμενοι θα είναι σε θέση:

- ⇒ Να αντιληφθούν τη σημασία και τη χρήση του προγραμματισμού
- ⇒ Να κατανοήσουν τη ροή εκτέλεσης σε προγράμματα
- ⇒ Να χρησιμοποιούν βασική σύνταξη για πρόσβαση, τροποποίηση και διαγραφή τύπων δεδομένων στην Python
- ⇒ Να χρησιμοποιούν την γλώσσα προγραμματισμού Python για να δημιουργήσουν μικρά προγράμματα

Απαιτούμενο υλικό και πόροι:

- ⇒ Υπολογιστής ή φορητός υπολογιστής
- ⇒ Πρόσβαση στο διαδίκτυο
- ⇒ Spyder
- ⇒ Κωδικός Visual Studio

Πρακτικά:

Η υποενότητα αυτή έχει σχεδιαστεί για να καλύπτει 10 ώρες μαθήματος. Κάθε υποενότητα έχει μια καθορισμένη διάρκεια διεκπεραίωσης, ωστόσο, οι εκπαιδευόμενοι ή οι εκπαιδευτικοί/ εκπαιδευτές είναι ελεύθεροι να αποφασίσουν πόσο χρόνο θα αφιερώσουν για κάθε επί μέρους θέμα, αναλόγως των προγενέστερών τους γνώσεων και της προηγούμενής τους ενασχόλησης με παρόμοια θέματα. Το περιεχόμενο βασίζεται στις

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

αρχές της προοδευτικής μαθησιακής ανάπτυξης και είναι διαβαθμισμένο με τρόπο ώστε να παρέχει βασικές γνώσεις και δεξιότητες σε αρχάριους.

Υποενότητες:

1. Εισαγωγή στον Προγραμματισμό
2. Εγκατάσταση Python
3. Βασικά στοιχεία προγραμματισμού στην Python

1. Εισαγωγή στον προγραμματισμό

⇒ **Αριθμός συμμετεχόντων:** 1-10 άτομα ανά συντονιστή
⇒ **Διάρκεια:** 0,5-0,75 ώρες
⇒ **Μέθοδοι διδασκαλίας:** Διάλεξη, παρουσίαση
⇒ **Απαιτούμενα υλικά:** Παρουσίαση

1.1. Τι είναι προγραμματισμός

Ο Προγραμματισμός είναι η διαδικασία δημιουργίας ενός προγράμματος που στοχεύει στην ολοκλήρωση μιας συγκεκριμένης εργασίας μέσω ενός υπολογιστή.

Η δημιουργία προγραμμάτων βασίζεται σε μια διαδικασία που πρέπει να εκτελεστεί βήμα προς βήμα. Η διαδικασία αυτή αναφέρεται συχνά και ως δημιουργία μιας «συνταγής», η οποία χρησιμοποιείται για να επικοινωνήσει με τον υπολογιστή μέσω ενός προκαθορισμένου συνόλου συντακτικών κανόνων (σύνταξη) και συναρτήσεων.

Μπορεί αυτό, εκ πρώτης όψεως, να σας ακούγεται περίπλοκο, αλλά στην πραγματικότητα δεν είναι. Η μάθηση μιας γλώσσας προγραμματισμού έχει πολλές ομοιότητες με την εκμάθηση μιας φυσικής γλώσσας από την αρχή, κατά την οποία ο μαθητευόμενος καλείται να κατανοήσει τη σύνταξη και το λεξιλόγιό της για να μπορέσει να την χρησιμοποιήσει σωστά. Μια εξίσου σημαντική πτυχή στην μάθηση μιας γλώσσας προγραμματισμού είναι εξάσκηση και η τριβή με αυτή, όπως συμβαίνει και με την εκμάθηση μιας φυσικής γλώσσας.

Φυσική γλώσσα (Αγγλικά)	Γλώσσα προγραμματισμού
John reads every day	print("Hello, World")

Πίνακας 1 - Σύγκριση μεταξύ μιας φυσικής γλώσσας και μιας γλώσσας προγραμματισμού

1.2. Γιατί να μάθουμε προγραμματισμό; Ποια είναι τα οφέλη;

Ίσως αναρωτιέστε ποιος είναι ο σκοπός της εκμάθησης μιας γλώσσας προγραμματισμού και τα οφέλη που αυτή συνεπάγεται.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Σύμφωνα με τον Steve Jobs: «Όλοι σε μια χώρα θα πρέπει να μάθουν να προγραμματίζουν έναν υπολογιστή γιατί αυτό θα τους μάθει να σκέφτονται». Σύμφωνα λοιπόν με το απόσπασμα αυτό, ένα από τα κύρια οφέλη του προγραμματισμού είναι ότι μπορεί να αναπτύξει την υπολογιστική μας σκέψη. Η υπολογιστική σκέψη αναφέρεται «στην ικανότητα ενός ατόμου να προσδιορίζει, να δοκιμάζει και να εφαρμόζει πιθανές αλγοριθμικές λύσεις σε ένα συγκεκριμένο πρόβλημα ή σε ανάλογα προβλήματα που μπορεί να προκύψουν σε ένα νέο πλαίσιο ή κατάσταση» (Karpinski et al., 2021, σ. 3). Η υπολογιστική σκέψη μπορεί επίσης να συσχετισθεί άμεσα με τις δεξιότητες επίλυσης προβλημάτων, οι οποίες ανήκουν στις θεμελιώδεις εκείνες δεξιότητες που χρειαζόμαστε επιτακτικά να αποκτήσουμε, για να μπορέσουμε να προσαρμοστούμε στις αλλαγές και τις απαιτήσεις της σύγχρονης αγοράς εργασίας (όπως αναφέρεται στους Karpinski et al., 2021, Czaja and Urbaniec, 2019· Rakowska and Cichorzewska, 2016· Slavinskis et al., 2015). Οι δεξιότητες επίλυσης προβλημάτων χαίρουν μεγάλης εκτίμησης μεταξύ των εργοδοτών και θεωρούνται μία από τις πιο σημαντικές δεξιότητες του 21ου αιώνα.

Δεν μαθαίνουμε, κατ' ανάγκη, να χρησιμοποιούμε μια γλώσσα προγραμματισμού επειδή θέλουμε να γίνουμε προγραμματιστές, αλλά γιατί ο προγραμματισμός μπορεί να είναι επωφελής για οποιουδήποτε είδους καριέρα, καθώς μπορεί να ανοίξει νέες δρόμους και ευκαιρίες επαγγελματικής σταδιοδρομίας.

Η ευελιξία που παρέχει ο προγραμματισμός σε σχέση με τις μελλοντικές θέσεις εργασίας, οι οποίες θα είναι πιο προσανατολισμένες στον προγραμματισμό είναι ένα άλλο σημαντικό πλεονέκτημα της εκμάθησης μιας γλώσσας προγραμματισμού. Επιπλέον, η κατοχή βασικών γνώσεων προγραμματισμού θα θεωρείται απαραίτητη σε όλα σχεδόν τα μελλοντικά επαγγέλματα. Το γεγονός αυτό καθιστά πολύ χρήσιμες τις δεξιότητες προγραμματισμού για την κοινωνία και την αγορά εργασίας του αύριο.

Κάθε άτομο που αισθάνεται στάσιμο στην καριέρα του μπορεί να μάθει μια γλώσσα προγραμματισμού για να αυξήσει το εισόδημά του και να βελτιώσει τις προοπτικές της επαγγελματικής του σταδιοδρομίας, καθώς και για να αναβαθμίσει τις δεξιότητές του στην επίλυση προβλημάτων. Ως εκ τούτου, ο προγραμματισμός είναι μια δεξιότητα που βρίσκει ευρεία εφαρμογή και που συμπεριλαμβάνει τόσο τεχνικές όσο και ήπιες δεξιότητες (soft skills), ενώ μπορεί να τις αποκτήσει οποιοσδήποτε αρκεί να θέλει να δοκιμάσει.

1.3. Η διαδικασία ανάπτυξης προγραμμάτων και οι αλγόριθμοι

Εάν εξακολουθείτε να διαβάζετε την υποενότητα αυτή, τότε επιτρέψτε μας να σας εξηγήσουμε τι είναι ένας αλγόριθμος και πώς αναπτύσσεται ένα πρόγραμμα.

Οι αλγόριθμοι αποτελούν αναπόσπαστο μέρος των προγραμμάτων, καθώς αντιπροσωπεύουν τα βήματα που πρέπει να ακολουθηθούν για την ολοκλήρωση της εργασίας που καθορίζεται από τον κώδικα. Ένας καλός αλγόριθμος πρέπει να περιλαμβάνει τα εξής: τα βήματα ή τις δραστηριότητες που απαιτούνται για την

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

ολοκλήρωση της εργασίας, τη σωστή σειρά αυτών των δραστηριοτήτων και τον τερματισμό τους. Ένα παράδειγμα που χρησιμοποιείται συχνά για την κατανόηση της διαδικασίας ενός αλγορίθμου είναι μια συνταγή μαγειρικής. Μπορούμε να παρομοιάσουμε τη διαδικασία ενός αλγόριθμου με μια συνταγή κέικ, όπως φαίνεται στην πιο κάτω εικόνα.

Εικόνα 1 - Οπτική αναπαράσταση αλγορίθμου

Όπως μπορείτε να δείτε η σειρά κάθε δραστηριότητας ή βήματος βασίζεται σε μια λογική βάση. Για παράδειγμα, δεν μπορείτε να τοποθετήσετε το ταψί στο φούρνο χωρίς να βάλετε πρώτα τα υλικά σ' αυτό, εάν προφανώς θέλετε να έχετε ένα κέικ. Το ίδιο ισχύει και στην περίπτωση των αλγόριθμων, όπου κάθε βήμα ή δραστηριότητα εκτελείται γιατί έχει έναν σκοπό να εξυπηρετήσει. Έτσι λοιπόν, για να λειτουργήσει ένας αλγόριθμους όπως τον θέλουμε, θα πρέπει επίσης να ακολουθήσει μια σωστή σειρά. Η διαδικασία που μόλις περιγράψαμε ονομάζεται επίσης *ψευδοκώδικας* (*pseudocode*), όρος που χρησιμοποιείται για να εξηγήσει τη διαδικασία που ακολουθείται από έναν αλγόριθμο.

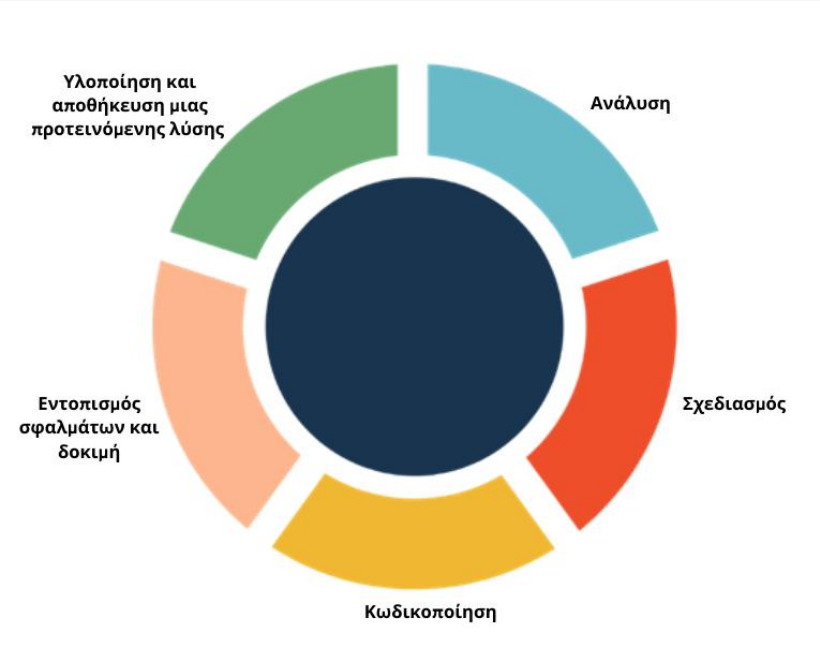
Μια άλλη σημαντική πτυχή είναι το μοτίβο που ακολουθείται για να σχεδιαστεί ένας συγκεκριμένος αλγόριθμος, ο οποίος συνήθως περιλαμβάνει το Μοτίβο Εισόδου (Input Pattern), Επεξεργασίας (Process) και Εξόδου (Output Pattern), γνωστά με την αγγλική συντομογραφία IPO. Ο αλγόριθμος ξεκινά με την ανάγνωση δεδομένων, τα οποία στη συνέχεια επεξεργάζεται για να εμφανίσει στο τέλος ένα αποτέλεσμα. Για παράδειγμα, ας υποθέσουμε ότι θέλετε να υπολογίσετε τους βαθμούς των μαθημάτων των μαθητών σας: πρώτα θα προσθέσετε τις βαθμολογίες τους από κάθε διαγώνισμα (Μοτίβο Εισόδου), στη συνέχεια θα υπολογίσετε τη μέση βαθμολογία (Επεξεργασία) και, τέλος, θα έχετε τη μέση βαθμολογία τους (Μοτίβο Εξόδου). Αυτό είναι ένα πολύ χρήσιμο μοτίβο που πρέπει να λάβετε υπόψη όταν δημιουργείτε έναν αλγόριθμο.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Εικόνα 2 – Μοτίβο IPO

Εφόσον έχουμε μάθει τι είναι ένας αλγόριθμος, καθώς και τη διαδικασία που απαιτείται για τη δημιουργία ενός δυνατού αλγόριθμου, μπορούμε τώρα να προχωρήσουμε με την ανάπτυξη ενός προγράμματος για να κατανοήσουμε καλύτερα πώς αυτό λειτουργεί. Η ανάπτυξη ενός προγράμματος διέρχεται από έναν κύκλο που περιλαμβάνει ανάλυση, σχεδιασμό, κωδικοποίηση, εντοπισμό σφαλμάτων και δοκιμή, καθώς και υλοποίηση και αποθήκευση της προτεινόμενης λύσης.



Εικόνα 3 – Κύκλος Ανάπτυξης Προγράμματος

Στο πρώτο βήμα, θα πρέπει να προσπαθήσουμε να κατανοήσουμε, να εντοπίσουμε και να αναλύσουμε το πρόβλημα που απαιτεί υπολογιστική λύση. Το δεύτερο βήμα αφορά το σχεδιασμό του προγράμματος, το οποίο περιλαμβάνει τη δημιουργία ενός οπτικού διαγράμματος της διαδικασίας που θα ακολουθήσει το πρόγραμμα. Αυτό είναι ιδιαίτερα χρήσιμο, καθώς θα σας βοηθήσει να κατακερματίσετε το πρόβλημα σε μικρότερα μέρη. Το επόμενο βήμα αφορά την κωδικοποίηση του προγράμματος με βάση το οπτικό διάγραμμα που δημιουργήθηκε στο προηγούμενο βήμα. Ο χρήστης τώρα θα πρέπει να εκτελέσει τον

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

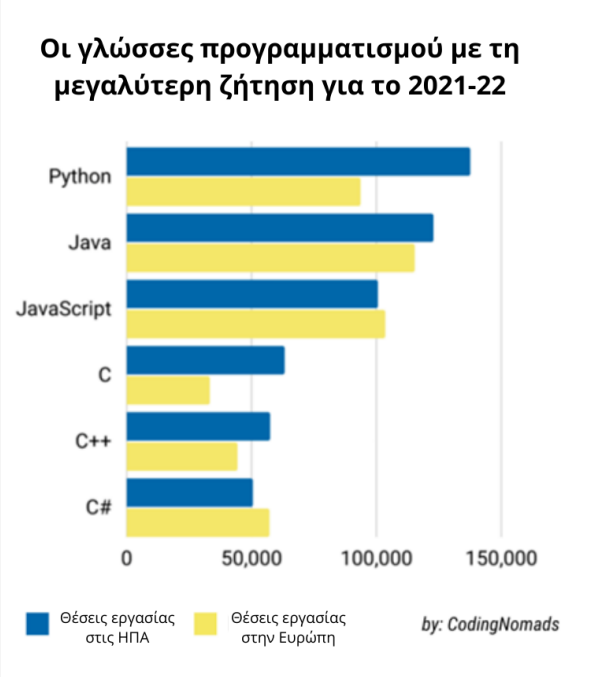
Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

κώδικα στον υπολογιστή για να εντοπίσει τυχόν σφάλματα. Στο τέταρτο βήμα, ο χρήστης θα πρέπει να πραγματοποιήσει αποσφαλμάτωση (debugging) του προγράμματος προκειμένου να αποφευχθούν πιθανά σφάλματα. Το πέμπτο βήμα απαιτεί από τον χρήστη να εκτελέσει το πρόγραμμα και να βεβαιωθεί ότι δεν υπάρχουν συντακτικά ή λογικά σφάλματα. Το έκτο βήμα που είναι και το τελευταίο αφορά την τεκμηρίωση και τη συντήρηση του προγράμματος, που απαιτεί μια εξήγηση της λογικής βάσης του προγράμματος και των διαδικασιών του.

Εάν υπάρχουν κάποια σημεία που δεν έχετε κατανοήσει από αυτή τη διαδικασία, μην ανησυχείτε. Μόλις ξεκινήσουμε την εξάσκηση, το τοπίο θα αρχίσει να ξεκαθαρίζει. Στην υποενότητα που ακολουθεί, θα επικεντρωθούμε στα τρία πρώτα βήματα. Ως εκ τούτου, θα προσπαθήσουμε να εξηγήσουμε τη διαδικασία ανάλυσης ενός προβλήματος και την ανάπτυξη ενός προγράμματος που μπορεί να χρησιμεύσει ως μια πιθανή λύση.

1.4. Γλώσσες προγραμματισμού – Οι πιο απαιτητικές γλώσσες προγραμματισμού

Σύμφωνα με μια ανάλυση χιλιάδων θέσεων εργασίας στις ΗΠΑ και την Ευρώπη, η Python έχει καταταχθεί ως η πιο περιζήτητη γλώσσα προγραμματισμού για το 2022, ακολουθούμενη από την Java και την JavaScript. Ενώ τα αποτελέσματα κατέδειξαν ότι η ζήτηση για τις γλώσσες προγραμματισμού C, C++ και C# δεν ήταν τόσο υψηλή, εξακολουθούσε εντούτοις να υπάρχει κάποια ζήτηση για αυτές σε συγκεκριμένα επαγγέλματα. Ενώ η Python είναι διαθέσιμη εδώ και δεκαετίες, υπολογίζεται ότι η ζήτησή της το 2022 θα συνεχίσει να αυξάνεται χάρη στην εκθετική αύξηση της χρήσης της στις ακμάζουσες βιομηχανίες της επιστήμης δεδομένων, της μηχανικής μάθησης και της τεχνητής νοημοσύνης (AI).



Εικόνα 4 - Οι γλώσσες προγραμματισμού με τη μεγαλύτερη ζήτηση
Πηγή: https://www.siliconrepublic.com/careers/python-most-in-demand-coding-language-2022

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

2. Εγκατάσταση της Python

⇒ **Αριθμός συμμετεχόντων: 1-10 ανά συντονιστή**
⇒ **Διάρκεια: 0,75-1 ώρες**
⇒ **Μέθοδοι διδασκαλίας:** Παρουσίαση, Καθοδηγούμενη διδασκαλία, Βιωματική μάθηση
⇒ **Απαιτούμενα υλικά:** Παρουσίαση, Υπολογιστές (1 ανά συμμετέχοντα) και σταθερή σύνδεση στο διαδίκτυο

Πριν εξηγήσουμε πώς να εγκαταστήσετε ένα Ολοκληρωμένο Περιβάλλον Ανάπτυξης της Python (IDE) και πώς να εκτελέσετε την Python από τη γραμμή εντολών, θα εξηγήσουμε πρώτα εν συντομία τι είναι και γιατί είναι η πιο κοινόχρηστη γλώσσα προγραμματισμού στην αγορά εργασίας.

2.1. Τι είναι η Python;

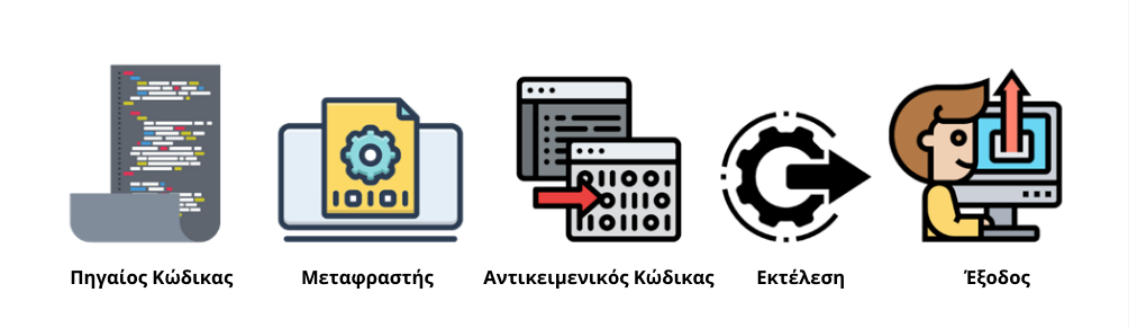
Η Python είναι μια γλώσσα προγραμματισμού υψηλού επιπέδου που επιτρέπει μεγαλύτερη ευελιξία και αναγνωσιμότητα κατά την κωδικοποίηση, ενώ μπορεί εύκολα να προσαρμοστεί σε διαφορετικά είδη συσκευών χωρίς καμία ή λίγες πιθανές τροποποιήσεις. Άλλες γλώσσες υψηλού επιπέδου περιλαμβάνουν την C, C++ και την Java.

Η διαφορά μεταξύ των γλωσσών προγραμματισμού υψηλού και χαμηλού επιπέδου είναι ότι οι υπολογιστές εκτελούν προγράμματα που είναι γραμμένα σε γλώσσες χαμηλού επιπέδου. Οι γλώσσες χαμηλού επιπέδου προγραμματισμού χρησιμοποιούν δυαδικό κώδικα (0,1) για την εκτέλεση ενός προγράμματος. Ωστόσο, οι γλώσσες χαμηλού επιπέδου προγραμματισμού δεν μπορούν εύκολα να ερμηνευτούν από τον άνθρωπο, εκτός και αν είναι έμπειρος. Επομένως, τα προγράμματα που είναι γραμμένα σε γλώσσες υψηλού επιπέδου πρέπει να μετατραπούν ή να μεταφραστούν σε δυαδική μορφή για να μπορεί να καταλάβει ο υπολογιστής τι πρέπει να κάνει. Αυτό απαιτεί κάποιο επιπρόσθετο χρόνο επεξεργασίας, αλλά αυτό αποτελεί ένα σχετικά ασήμαντο μειονέκτημα των γλωσσών υψηλού επιπέδου προγραμματισμού.

Γενικά, υπάρχουν δύο τρόποι με τους οποίους ένα πρόγραμμα μεταφράζεται σε δυαδική μορφή πριν εκτελεστεί. Ο πρώτος τρόπος περιλαμβάνει τη χρήση ενός μεταφραστή ή μεταγλωττιστή, που είναι ένα πρόγραμμα υπολογιστή που διαβάζει κώδικα γραμμένο σε μια γλώσσα προγραμματισμού (την πηγαία γλώσσα) και τον μεταφράζει σε ισοδύναμο κώδικα σε μια άλλη γλώσσα προγραμματισμού (τη γλώσσα στόχο) πριν τον εκτελέσει. Όπως φαίνεται στην πιο κάτω εικόνα, η διαδικασία ξεκινά από τον πηγαίο κώδικα (για μια γλώσσα προγραμματισμού υψηλού επιπέδου), προχωρά στον αντικειμενικό κώδικα ή στον εκτελεστή (που μεταφράζει τη γλώσσα προγραμματισμού υψηλού επιπέδου σε μια γλώσσα χαμηλότερου επιπέδου, δηλαδή σε μια γλώσσα μηχανής η οποία έχει δυαδική μορφή) και επιστρέφει για να μεταγλωττίσει το πρόγραμμα που θα εκτελεστεί ως έξοδος, χωρίς περαιτέρω μετάφραση.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης



Εικόνα 5 – Μεταφραστής ή Μεταγλωττιστής (Προσαρμογή από Downey et al., 2002, σελ.30)

Ο δεύτερος τρόπος χρησιμοποιεί μόνο έναν διερμηνευτή ή διερμηνέα ως μεσολαβητή μεταξύ του πηγαίου κώδικα και της εξόδου. Ο διερμηνευτής ουσιαστικά εκτελεί ή ερμηνεύει εντολές σε κάποια γλώσσα προγραμματισμού. Αυτή είναι μια συνεχής διαδικασία εμπρός και πίσω μέχρι να ολοκληρωθεί το πρόγραμμα, όπως μπορείτε να δείτε στην πιο κάτω εικόνα.

Εικόνα 6 – Διερμηνέας ή Διερμηνευτής (Προσαρμογή από Downey et al., 2002, σελ.30)

Η Python χρησιμοποιεί τον διερμηνευτή για να εκτελέσει τα προγράμματά της και έτσι, θεωρείται ως διερμηνευμένη γλώσσα. Αυτό μπορεί να γίνει είτε από την γραμμή εντολών (command line) είτε με τα εργαλεία IDLE (Python GUI) και Module Docs. Με τον πρώτο τρόπο, απλά γράφετε τον κώδικά σας και ο διερμηνευτής εκτυπώνει στην οθόνη το τελικό αποτέλεσμα. Με τα εργαλεία IDLE και Module Docs, ανοίγει ο διερμηνευτής της Python ο οποίος επεξεργάζεται ένα πρόγραμμα της python υπό μορφή αρχείου που τελειώνει σε .py και εκτελεί τα περιεχόμενά του. Θα πρέπει να σημειωθεί ότι τα περισσότερα προγράμματα της Python τελειώνουν σε **.py** , ωστόσο, αυτό εξαρτάται από το Ολοκληρωμένο Περιβάλλον Ανάπτυξης (IDE) που χρησιμοποιείτε.

Στην παρούσα υποενότητα, όλα τα παραδείγματα που χρησιμοποιούμε εφαρμόζουν τη μέθοδο της γραμμής εντολών, η οποία επιτρέπει την άμεση εκτέλεση των προγραμμάτων και την καλύτερη κατανόηση της διαδικασίας. Μόλις ολοκληρώσετε ένα πρόγραμμα, μπορείτε να το αποθηκεύσετε για να μπορείτε να το εκτελέσετε ως σενάριο αργότερα.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

2.2. Τα πιο κοινόχρηστα Περιβάλλοντα Ολοκληρωμένης Ανάπτυξης της Python (IDE) ανά κλάδο

Στην προηγούμενη ενότητα, αναφέραμε τα Περιβάλλοντα Ολοκληρωμένης Ανάπτυξης (IDEs). Ας εξηγήσουμε λοιπόν εν συντομία τι είναι ένα IDE. Ένα IDE είναι ουσιαστικά μια εφαρμογή λογισμικού που προσφέρει διάφορες διευκολύνσεις στους προγραμματιστές υπολογιστών στην ανάπτυξη του λογισμικού τους, όπως ένα πρόγραμμα επεξεργασίας πηγαίου κώδικα, εργαλεία αυτοματισμού κατασκευής (build automation tools) και λειτουργίες debugging. Αν αυτή τη στιγμή δεν βγάζετε αρκετό νόημα απ' αυτά που διαβάζετε, μην ανησυχείτε. Μόλις ξεκινήσουμε την εξάσκηση, θα κατανοήσετε καλύτερα όλα όσα έχουμε αναλύσει μέχρι τώρα.

Υπάρχουν διαφορετικά είδη Ολοκληρωμένων Περιβαλλόντων Ανάπτυξης Python (IDE) από τα οποία μπορεί να επιλέξει κανείς με βάση τις ανάγκες του. Μερικοί από τους παράγοντες που λαμβάνονται υπόψη όταν μια εταιρεία ή ένας οργανισμός επιλέγει ένα IDE είναι: 1) ο σκοπός χρήσης του IDE, όπως π.χ. για τον σχεδιασμό και την ανάπτυξη ιστοτόπων, για την επιστήμη δεδομένων, τη γλώσσα προγραμματισμού σεναρίων ή τη διασφάλιση της ποιότητας 2) το λειτουργικό σύστημα του υπολογιστή του χρήστη (π.χ. Linux/macOS, Windows ή μικτό OS) 3) η ποιότητα του υλισμικού του υπολογιστή του χρήστη και 4) το γνωστικό επίπεδο του προγραμματιστή (π.χ. αρχάριος, μεσαίος ή προχωρημένος).

Εικόνα 7 - Τα πιο κοινόχρηστα IDE ανά κλάδο
(Προσαρμογή από το: https://www.geeksforgeeks.org/top-10-python-ide-and-code-editors-in-2020/)

2.3. Η εγκατάσταση ενός Ολοκληρωμένου Περιβάλλοντος Ανάπτυξης της Python (IDEs)

Στην υποενότητα αυτή, μπορείτε να χρησιμοποιήσετε είτε το Spyder (https://www.anaconda.com/products/individual) είτε το Visual Studio Code (https://code.visualstudio.com/), καθώς αυτοί είναι οι πιο κοινόχρηστοι και εύχρηστοι

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

επεξεργαστές κειμένου ανάμεσα σε αρχάριους που επιθυμούν να εξοικειωθούν με την γλώσσα προγραμματισμού της Python.

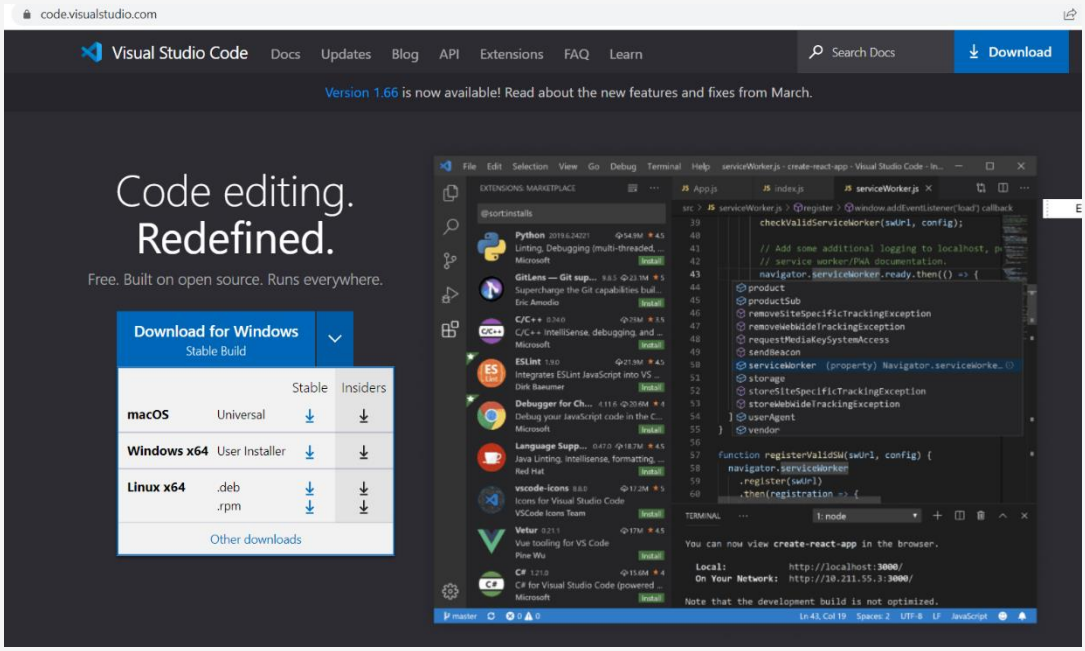
Ωστόσο, εάν δεν είστε έτοιμοι να δεσμευτείτε με ένα συγκεκριμένο IDE, έχετε επίσης την επιλογή να χρησιμοποιήσετε άλλους επεξεργαστές κειμένου στο Διαδίκτυο όπως είναι οι ακόλουθοι:

- Python.org
- Programiz
- Tutorialspoint

Συνιστούμε να χρησιμοποιήσετε ένα IDE, καθώς αυτό θα σας βοηθήσει να παρακολουθείτε πιο εύκολα τα αρχεία σας και την πρόοδό σας στον προγραμματισμό.

2.3.1 Εγκατάσταση του Visual Studio Code

1. Μεταβείτε στον ιστότοπο: https://code.visualstudio.com/
2. Μόλις γίνει η φόρτωση της σελίδας, θα εμφανιστεί η σήμανση «Λήψη». Αυτό θα επιλέξει αυτόματα το λειτουργικό σύστημα του υπολογιστή σας. Ωστόσο, μπορείτε να κάνετε κλικ στο βελάκι δίπλα από τη λήψη για να επιλέξετε το λειτουργικό σύστημα που υποστηρίζει ο υπολογιστής σας. Φροντίστε να κατεβάσετε το stable built!

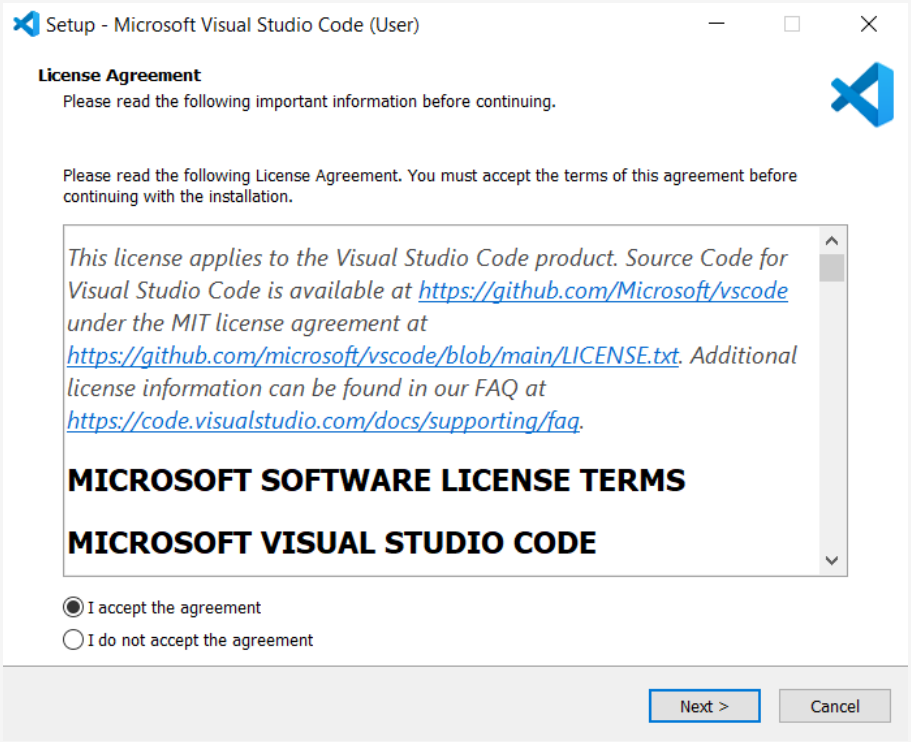


Εικόνα 8 – Λήψη αρχείου του Visual Studio Code

3. Όταν ολοκληρωθεί η λήψη, ανοίξτε το αρχείο και ακολουθήστε τις οδηγίες εγκατάστασης.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

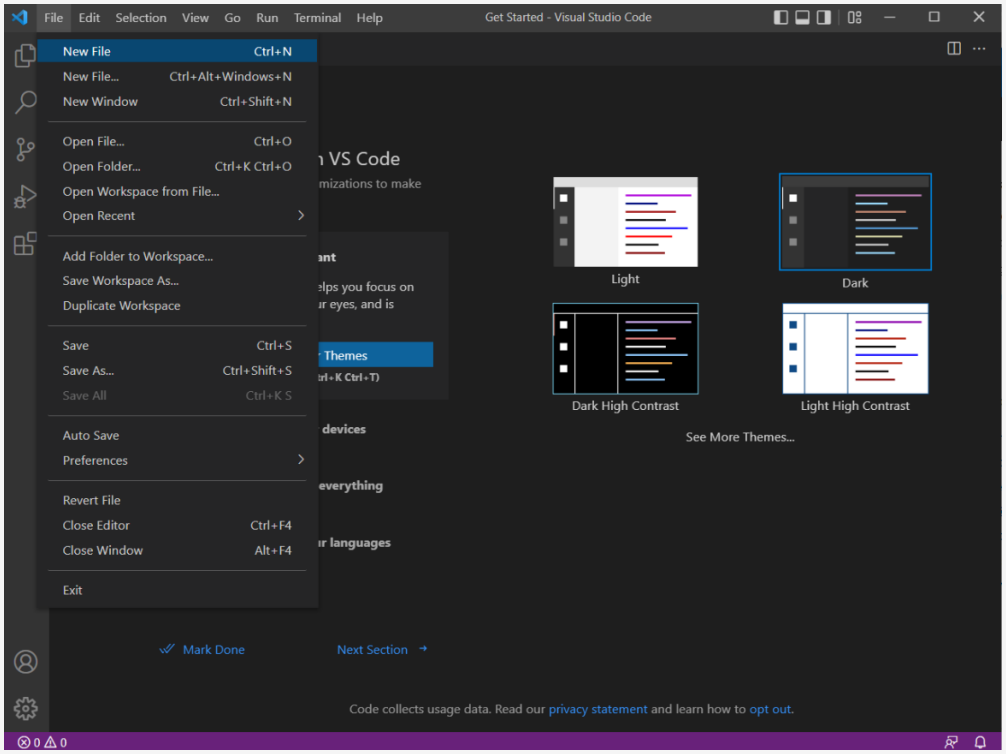
Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης



Εικόνα 9 – Εγκατάσταση του IDE

4. Μόλις ολοκληρωθεί η εγκατάσταση του προγράμματος, μπορείτε να το ανοίξετε ή θα ξεκινήσει αυτόματα.

5. Επιλέξτε File → New

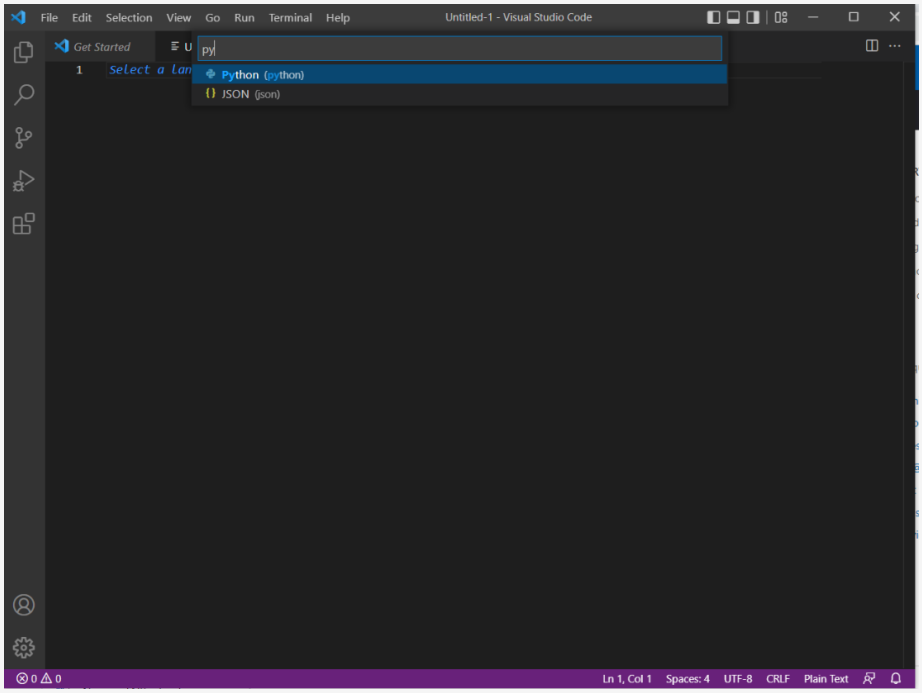


Εικόνα 10 – Δημιουργία νέου αρχείου στο Visual Studio Code

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

6. Κάντε κλικ στις επιλογές Select a Language και επιλέξτε την Python



Εικόνα 11 – Επιλογή της γλώσσας προγραμματισμού

7. Μόλις επιλέξετε την Python, θα ανακατευθυνθείτε στο αντίστοιχο μέρος για να εγκαταστήσετε την Python.

Εικόνα 12 – Εγκατάσταση της Python από Extensions

8. Μόλις εγκαταστήσετε την Python, θα είστε έτοιμοι να ξεκινήσετε με την κωδικοποίηση!

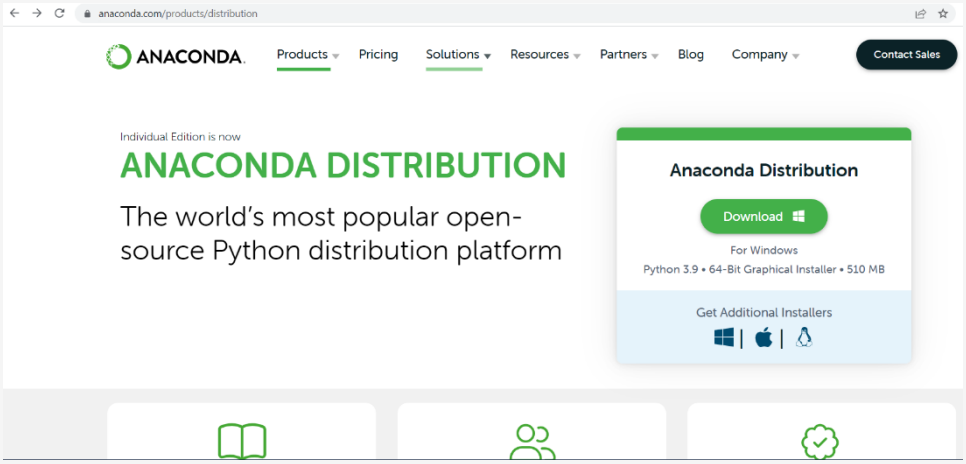
Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Για περισσότερες πληροφορίες, μπορείτε επίσης να συμβουλευτείτε τον ιστότοπο του Visual Studio Code σχετικά με το πώς να ξεκινήσετε:
https://code.visualstudio.com/docs/introvideos/basics

2.3.2. Εγκατάσταση του Spyder από το Anaconda

1. Μεταβείτε στον ιστότοπο: https://www.anaconda.com/products/individual
2. Μόλις γίνει η φόρτωση της σελίδας, θα δείτε τη σήμανση Λήψη. Ανάλογα με το λειτουργικό σύστημα του υπολογιστή σας, επιλέξτε την αντίστοιχη έκδοση. Συνιστούμε το πρόγραμμα εγκατάστασης γραφικών 64-bit.



Εικόνα 13 – Η λήψη του Anaconda Distribution

   a. Για περισσότερες πληροφορίες σχετικά με τα βήματα που απαιτούνται για τα Windows, ακολουθήστε αυτόν τον σύνδεσμο: https://docs.anaconda.com/anaconda/install/windows/
   b. Για τους χρήστες macOS, ακολουθήστε αυτόν τον σύνδεσμο: https://docs.anaconda.com/anaconda/install/mac-os/
   c. Για τους χρήστες Linux, ακολουθήστε αυτόν τον σύνδεσμο: https://docs.anaconda.com/anaconda/install/linux/

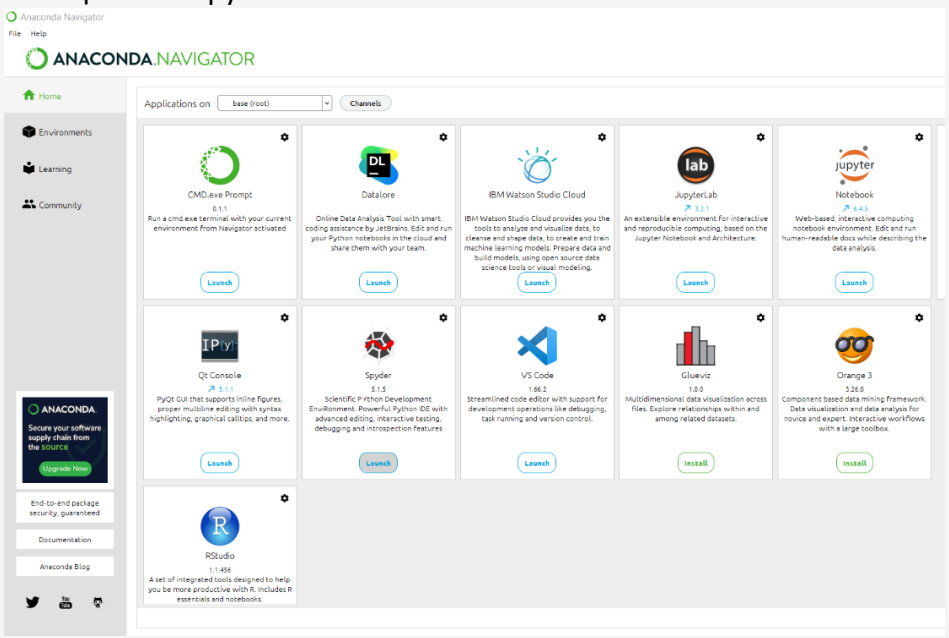
3. Όταν ολοκληρωθεί η λήψη, ανοίξτε το αρχείο και ακολουθήστε τις οδηγίες εγκατάστασης.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Εικόνα 14 – Εγκατάσταση του Anaconda

4. Μόλις ολοκληρωθεί η εγκατάσταση του προγράμματος, μπορείτε να ανοίξετε το Anaconda Navigator.
5. Μόλις γίνει η φόρτωση του Navigator, επιλέξτε την εφαρμογή Spyder και κάντε κλικ στο Launch. Εάν δεν είναι εγκατεστημένο, κάντε κλικ στη σήμανση Install για να εγκαταστήσετε το Spyder.

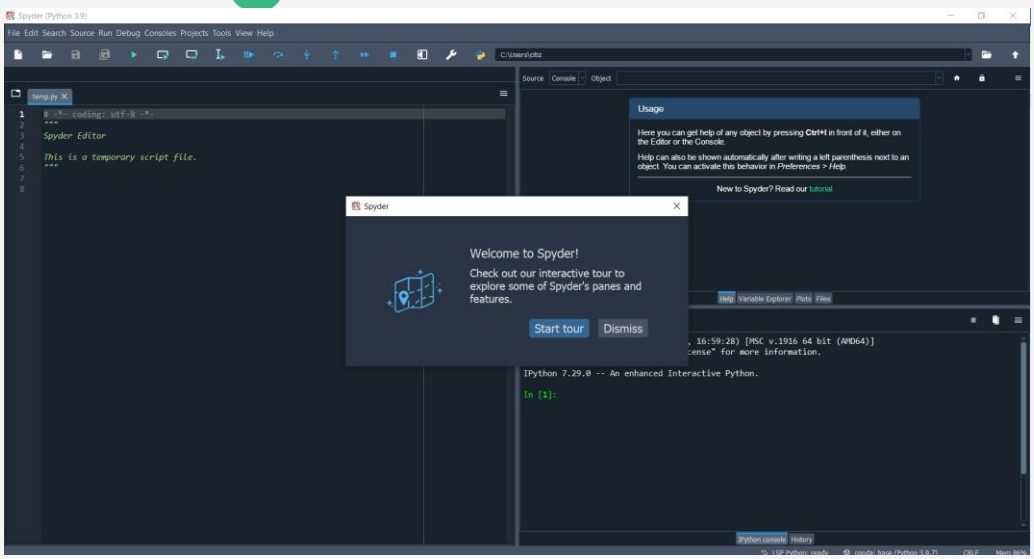


Εικόνα 15 – Anaconda Navigator

6. Όταν ανοίξετε το Spyder, μπορείτε να επιλέξετε να κάνετε μια περιήγηση ή να απορρίψετε αυτή την επιλογή.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης



Εικόνα 15 – Άνοιγμα του Spyder για πρώτη φορά

7. Τώρα μπορείτε να ξεκινήσετε με την κωδικοποίηση!

Για περισσότερες πληροφορίες, μπορείτε επίσης να συμβουλευτείτε το Anaconda starter: https://docs.anaconda.com/anaconda/user-guide/getting-started/

Είμαστε σίγουροι ότι θα καταφέρετε να ολοκληρώσετε αυτό το βήμα εύκολα!

2.4. Προγράμματα Python που τρέχουν από τη γραμμή εντολών

Τώρα είναι ώρα να «τρέξουμε» το πρώτο μας πρόγραμμα. Ίσως αναρωτιέστε αν είναι τόσο απλό. Η απάντηση είναι: Ναι, είναι!

Απλώς πληκτρολογήστε τα εξής:

```
print("Hello, World!")
```

Τώρα, μπορείτε να το «τρέξετε». Θα πρέπει να εκτυπωθεί στην οθόνη το κείμενο: **Hello, World!**

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

3. Βασικές έννοιες στον προγραμματισμό με Python

⇒ **Αριθμός συμμετεχόντων:** 1-10 ανά συντονιστή
⇒ **Διάρκεια:** 8.00 ώρες
⇒ **Μέθοδοι διδασκαλίας:** Παρουσίαση, Καθοδηγούμενη διδασκαλία, Βιωματική μάθηση
⇒ **Απαιτούμενα υλικά:** Παρουσίαση, Υπολογιστές (1 ανά συμμετέχοντα) και σταθερή σύνδεση στο διαδίκτυο

Αφού τρέξατε με επιτυχία το πρώτο σας πρόγραμμα, θα συνεχίσουμε να μαθαίνουμε περισσότερα για τις βασικές λειτουργίες της Python και τη λογική πίσω από αυτές.

3.1. Βασικές έννοιες: Τύποι δεδομένων (Βασικοί και Σύνθετοι)

Οι τύποι δεδομένων είναι μια σημαντική έννοια στον κόσμο του προγραμματισμού, καθώς καθορίζουν τον τρόπο επεξεργασίας των δεδομένων. Οι ενσωματωμένοι τύποι δεδομένων στην Python εμπίπτουν στις ακόλουθες κατηγορίες:

Γενικές ομάδες τύπων δεδομένων	Συγκεκριμένοι τύποι δεδομένων που χρησιμοποιούνται στην Python	Παραδείγματα
Κείμενο	Κλάση str	"Hello, World!" (μια σειρά από συμβολοσειρές, στις οποίες πρέπει πάντοτε να χρησιμοποιούνται εισαγωγικά)
Αριθμοί	ακέραιοι, αριθμοί κινητής υποδιαστολής (float), μιγαδικοί αριθμοί (complex)	3 (ακέραιος αριθμός -int), 3.2 (αριθμοί κινητής υποδιαστολής – περιλαμβάνει δεκαδικούς αριθμούς), 2j+3 (περιλαμβάνει χαρακτήρες και αριθμούς)
Ακολουθίες	Λίστα (list), πλειάδα (tuple), εύρος (range)	["apple", "cherry", 1, 1] (λίστα, ακολουθία στοιχείων οποιουδήποτε τύπου), () (πλειάδα, ταξινομημένη συλλογή), εύρος(5) (επαναλαμβάνει την ακολουθία αριθμών, έχει σημείο έναρξης και λήξης (5))
Χάρτης (mapping)	Κλάση dict	{"brand": "Ford"} (λεξικό, μη ταξινομημένη συλλογή ζευγών κλειδιών-τιμών (key-value pairs))
Σύνολο (set)	σύνολο, frozenset	{"μήλο", "κεράσι", 1} (σύνολο, μη ταξινομημένη συλλογή χωρίς

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

		διπλότυπα στοιχεία οποιουδήποτε τύπου διαχωρισμένα με κόμματα)
Μπούλειοι τύποι	Λογικές δυαδικές τιμές	Οι τιμές True (αληθές) και False (ψευδές)
Δυαδικοί τύποι	byte, bytearray, memoryview	χρησιμοποιείται για την επεξεργασία δυαδικών δεδομένων

Πίνακας 2 – Οι διάφοροι τύποι δεδομένων της Python με παραδείγματα (αναπροσαρμογή από το : https://www.w3schools.com/python/python_datatypes.asp)

Αυτοί είναι όλοι οι ενσωματωμένοι τύποι δεδομένων που υπάρχουν στην Python. Θα τους δούμε όλους τους τύπους πιο αναλυτικά στη συνέχεια εκτός από τους τελευταίους, τους δυαδικούς τύπους δεδομένων, αφού οι άλλοι είναι πιο χρήσιμοι για αρχάριους.

3.2. Μεταβλητές, εκφράσεις και δηλώσεις

Μεταβλητές

Οι μεταβλητές ουσιαστικά περιέχουν δεδομένα τα οποία εκχωρούμε σε αυτές με τη μορφή ονομάτων, για να κάνουμε τον κώδικά μας πιο ευανάγνωστο και να μπορούμε να χρησιμοποιήσουμε μια συγκεκριμένη μεταβλητή περισσότερες από μία φορές.

Γράφουμε το όνομα για κάθε μεταβλητή χρησιμοποιώντας το σύμβολο ίσων (=) και εκχωρούμε την τιμή που θέλουμε να αποθηκεύσουμε στη δεξιά του πλευρά.

Παράδειγμα:

```
x = 5
print(x)
```

Εδώ, η μεταβλητή x ισούται με 5 και η εντολή print(x) εκτυπώνει την τιμή που περιέχεται στο x (δηλαδή, 5).

***Λάβετε υπόψη ότι η Python διαθέτει διάκριση πεζών-κεφαλαίων, επομένως τα x και X δεν θεωρούνται ως η ίδια μεταβλητή.**

Θυμηθείτε τους τύπους δεδομένων που εξετάσαμε προηγουμένως, μπορείτε να γράψετε την ακόλουθη εντολή για να δείτε τον τύπο δεδομένων της μεταβλητής x που μόλις δημιουργήσαμε:

```
print(type(x))
```

Η οθόνη θα εμφανίσει: **int**

Ας υποθέσουμε ότι θέλετε να δημιουργήσετε 3 μεταβλητές όπου η μία περιέχει έναν ακέραιο αριθμό, η άλλη έναν αριθμό κινητής υποδιαστολής (float) και μια άλλη, μια συμβολοσειρά. Θα γράφατε τα εξής:

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

```
age = 35
name = "John"
price = 12.99
```

Τώρα, αν προσπαθήσουμε να το «τρέξουμε», η οθόνη δεν θα δείξει τίποτα. Ίσως να σκέφτεστε γιατί συμβαίνει αυτό, εφόσον δεν έχουμε καλέσει τη μεταβλητή, αλλά της εκχωρήσαμε μόνο μια τιμή. Για να δούμε τι περιέχει η μεταβλητή μας, πρέπει να χρησιμοποιήσουμε την εντολή **print (όνομα μεταβλητής)**. Διαφορετικά, αυτό δεν θα συμβεί.

Παράδειγμα:

```
print(age)
print(name)
print(price)
```

Ή μπορείτε να τα εκτυπώσετε όλα μαζί γράφοντας τα ως εξής:

```
print(age,name,price)
```

Μπορείτε επίσης να δώσετε οποιοδήποτε όνομα στη μεταβλητή και να μην περιοριστείτε μόνο σε μια λέξη.

Ωστόσο, δεν πρέπει να χρησιμοποιείτε πολύ μεγάλα ονόματα μεταβλητών γιατί αυτό δεν είναι βολικό ούτε για εσάς ούτε για κάποιον άλλο που θα διαβάσει τον κώδικά σας.

Προσπαθήστε να διατηρήσετε τα ονόματα των μεταβλητών σύντομα και κατανοητά. Εάν θέλετε το όνομα της μεταβλητής να αποτελείται από δύο ξεχωριστές λέξεις, απλώς χρησιμοποιήστε την κάτω παύλα στο ενδιάμεσο αυτών, π.χ. my_age.

***Λάβετε υπόψη ότι η Python δεν σας επιτρέπει να γράφετε τα ονόματα των μεταβλητών ξεκινώντας με αριθμούς ή να περιλαμβάνετε στα ονόματά τους ειδικά σύμβολα όπως @, #, $, κ.λπ.**

Στην Python υπάρχουν επίσης κάποιες δεσμευμένες λέξεις-κλειδιά για συγκεκριμένες συναρτήσεις, οι οποίες δεν μπορούν να χρησιμοποιηθούν ως ονόματα μεταβλητών. Αυτές οι λέξεις-κλειδιά είναι:

and	def	exec	if	not	return
assert	del	finally	import	or	try
break	elif	for	in	pass	while
class	else	from	Is	print	yield
continue	except	global	lambda	raise	

Πίνακας 3 - Δεσμευμένες λέξεις-κλειδιά στην Python (Προσαρμογή από Downey et al., 2002, σελ. 15)

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Εντολές

Μέχρι στιγμής έχουμε δει δύο εντολές που χρησιμοποιούνται στην Python: την εντολή εκτύπωσης και ανάθεσης μεταβλητής. Οι εντολές που δίνονται στον διερμηνευτή της Python που πρόκειται να εκτελεστεί ονομάζονται statements.

Όπως έχουμε ήδη αναφέρει, οι εντολές, όπως είναι η ανάθεση μεταβλητής, δεν παράγουν κάποιο αποτέλεσμα. Ωστόσο, η εντολή εκτύπωσης παράγει ένα αποτέλεσμα, το οποίο είναι η τιμή της μεταβλητής.

Μια ακολουθία εντολών θεωρείται ένα σενάριο που είτε παράγει αποτελέσματα είτε όχι, ανάλογα με τον τύπο εντολών που χρησιμοποιούνται. Τα αποτελέσματα εμφανίζονται με βάση τη σειρά που δόθηκαν οι εντολές.

Για παράδειγμα, εάν πληκτρολογήσετε τις ακόλουθες εντολές:

```
print(name)
height_cm = 1.75
print(age)
```

Θα λάβατε ως έξοδο:

John

35

Η μεταβλητή “height” αποθηκεύεται μόνο ως μεταβλητή αλλά δεν παράγει κανένα αποτέλεσμα.

Εκφράσεις

Υπάρχουν διάφορα είδη εκφράσεων που χρησιμοποιούνται στην Python. Οι εκφράσεις περιλαμβάνουν έναν συνδυασμό διαφορετικών μεταβλητών, τελεστών (operators) και τελεστέων (operands) που παράγουν μια άλλη τιμή ως έξοδο.

Για παράδειγμα, ας προσπαθήσουμε να προσθέσουμε δύο αριθμούς χωρίς να τους εκχωρήσουμε ονόματα μεταβλητών ή να χρησιμοποιήσουμε την εντολή εκτύπωσης:

```
2+2
```

Έξοδος:

Στο παράδειγμα αυτό, μπορούμε να δούμε ότι η Python δε θα μας δώσει μια έξοδο. Επομένως, σε μια έκφραση είναι απαραίτητο να συμπεριλαμβάνετε μεταβλητές, τελεστές και τιμές για να έχετε μια έξοδο.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Εάν θέλετε να έχετε ένα αποτέλεσμα από την έκφραση 2+2, τότε πρέπει είτε να συμπεριλάβετε την εντολή εκτύπωσης είτε να εκχωρήσετε σ' αυτήν ένα όνομα μεταβλητής και να την εκτυπώσετε.

Παράδειγμα:

```
print(2+2)
addition = 2+2
print(addition)
```

Στο παράδειγμα της έκφρασης 2+2, ζητάμε από την Python να αξιολογήσει την έκφραση. Ωστόσο, αν γράφαμε απλά το 2 και το 2 σε ξεχωριστές γραμμές χωρίς τον τελεστή (+), τότε δεν θα είχαμε κάποιο αποτέλεσμα. Ακόμα κι αν η εντολή θεωρείται νόμιμη, δεν παράγει κάποιο αποτέλεσμα.

Εξετάστε το ακόλουθο σενάριο:

```
56
4.5
"John"
print(3+2)
```

Εδώ, η μόνη έξοδος που θα λάβατε είναι το 5 σύμφωνα με την τελευταία γραμμή του 3+2. Τι μπορούμε να προσθέσουμε στο σενάριο για να εμφανιστούν όλες οι εκφράσεις στην οθόνη ως έξοδος;

Συμβουλή: το έχουμε συναντήσει πολλές φορές μέχρι τώρα

Τελεστές

Ήρθε η ώρα να εξηγήσουμε μερικούς από τους τελεστές που χρησιμοποιούνται στην Python, όπως το σύμβολο της πρόσθεσης (+) που χρησιμοποιήσαμε νωρίτερα. Οι τελεστές είναι τα ειδικά σύμβολα που χρησιμοποιούνται για την εκτέλεση μαθηματικών πράξεων, τα οποία είναι τα ακόλουθα:

Συν/ Πρόσθεση(3+20)	Μείον/Αφαίρεση (x-6)	Επί/Πολλαπλασιασμός (3*5)	Δια/Διαίρεση (10/5)	Δύναμη (4**2)

Μπορείτε επίσης να χρησιμοποιήσετε ονόματα μεταβλητών που περιέχουν ακέραιους αριθμούς ή αριθμούς κινητής υποδιαστολής, αντί για απλούς ακέραιους στην εκτέλεση μαθηματικών πράξεων.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Παράδειγμα:

```
course_grade1 = 15
course_grade2 = 18
coursesum = course_grade1+course_grade2
print(coursesum)
```

Έξοδος: 33

Για παράδειγμα, για να βρούμε τη μέση βαθμολογία των βαθμών 2 μαθημάτων, πρέπει να τους προσθέσουμε και στη συνέχεια να τους διαιρέσουμε με το 2. Πώς πιστεύετε ότι μπορεί να γίνει αυτό; Σκεφτείτε το για ένα λεπτό.

Απάντηση:

```
courseavg = (course_grade1+course_grade2)/2
print(courseavg)
```

Έξοδος: 16,5

Αυτό το παράδειγμα φανερώνει επίσης ότι η Python αναγνωρίζει τη σειρά υπολογισμού με την οποία πρέπει να εκτελεστούν οι μαθηματικές πράξεις, η οποία υπαγορεύει ότι η πράξη εντός της παρένθεσης έχει υψηλότερη προτεραιότητα από την πράξη της διαίρεσης που βρίσκεται έξω από την παρένθεση.

Προτεραιότητα τελεστών στην Python (από την υψηλότερη στη χαμηλότερη) 1) παρένθεση, 2) εκθετική αύξηση, 3) πολλαπλασιασμός/διαίρεση και 4) πρόσθεση/αφαίρεση.

Πρέπει να σημειωθεί ότι οι τελεστές συνήθως συσχετίζονται από τα αριστερά προς τα δεξιά, δηλαδή οι τελεστές με την ίδια προτεραιότητα υπολογίζονται από τα αριστερά προς τα δεξιά. Για παράδειγμα, εξετάστε την ακόλουθη έκφραση: 5*30/60. Αυτή ισούται με 2,5, που φανερώνει ότι πρώτα γίνεται ο πολλαπλασιασμός (=150) και μετά η διαίρεση, η οποία μας δίνει και το τελικό αποτέλεσμα.

Εξάσκηση:

https://www.w3schools.com/python/exercise.asp?filename=exercise_variables1

Οι τελεστές που μπορούν να εφαρμοστούν σε συμβολοσειρές

Ακόμα κι αν μια συμβολοσειρά θεωρείται ένας αριθμός (όπως π.χ. το '15'), γενικά δεν μπορείτε να εκτελέσετε τις προαναφερθείσες μαθηματικές πράξεις στις συμβολοσειρές.

Ωστόσο, οι τελεστές πρόσθεσης (+) και πολλαπλασιασμού (*) έχουν διαφορετικό αποτέλεσμα όταν εφαρμόζονται σε συμβολοσειρές. Εάν χρησιμοποιείτε πρόσθεση μεταξύ δύο μεταβλητών που περιέχουν συμβολοσειρές, τότε αυτές συνενώνονται. Αυτό σημαίνει ότι θα συνενωθούν και θα μετατραπούν αυτόματα ως ακολουθία χαρακτήρων.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Παράδειγμα:

```
first_name = 'Emma'
last_name = ' Smith'
print(first_name + last_name)
```

Έξοδος: Emma Smith

*Σημειώστε ότι εάν τα εισαγωγικά και οι συμβολοσειρές έχουν κενό μεταξύ τους, τότε θα έχετε το αποτέλεσμα "EmmaSmith" αντί για "Emma Smith". Επομένως, τα κενά αποτελούν μέρος των συμβολοσειρών.

Όταν εφαρμόζεται σε συμβολοσειρές, τότε ο τελεστής πολλαπλασιασμού (*) εκτελεί επανάληψη.

Παράδειγμα:

```
print("First"*3)
```

Έξοδος: FirstFirstFirst

Για να λειτουργήσει αυτό, πρέπει να χρησιμοποιήσετε έναν ακέραιο αριθμό για να καθορίσετε πόσες φορές θέλετε να επαναλαμβάνεται μια συμβολοσειρά.

Σχόλια (Comments)

Καθώς τα προγράμματα αποκτούν όλο και περισσότερο όγκο δεδομένων, γίνεται όλο και πιο δύσκολο να εντοπιστούν και να κατανοηθούν οι αλγόριθμοι που χρησιμοποιούνται για την παραγωγή ενός τελικού αποτελέσματος. Έτσι, τα σχόλια χρησιμοποιούνται συνήθως ως σημειώσεις από τον προγραμματιστή, τα οποία τον βοηθούν να εξηγήσει το σκεπτικό και τις διαδικασίες που ακολούθησε για τη δημιουργία ενός προγράμματος σε μια φυσική γλώσσα. Θεωρείστε λοιπόν ότι τα σχόλια λειτουργούν σαν προσθήκη σημειώσεων όπου και όποτε χρειάζεται, ούτως ώστε ο κώδικας να γίνεται πιο ευανάγνωστος στον προγραμματιστή. Απλά προσθέστε το σύμβολο hashtag (#) ακολουθούμενο από μικρά σχόλια.

Παράδειγμα:

```
#calculating the average grade of students based on the course grade
courseavg = (course_grade1+course_grade2)/2
```

Μπορείτε επίσης να προσθέσετε ένα σχόλιο στην ίδια γραμμή με τον κώδικα:

```
print(first_name + last_name) #concatenating two strings
```

Ό,τι γράψετε μετά το σύμβολο hashtag (#) αγνοείται και δεν εκτελείται ως κώδικας από το πρόγραμμα.

Μια άλλη επιλογή που μπορεί να χρησιμοποιηθεί για μεγαλύτερα μέρη κειμένου είναι η προσθήκη 3 εισαγωγικών (“””) πριν και 3 (“””) μετά την ολοκλήρωση του κειμένου.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Παράδειγμα:

```
"""
I can add as much text as I like to explain a function
"""
```

Η χρήση σχολίων είναι μια χρήσιμη πρακτική για κάθε επαγγελματία που ασχολείται με τον προγραμματισμό.

Εξάσκηση:
https://www.w3schools.com/python/exercise.asp?filename=exercise_comments1

3.3. Λίστες, λεξικά και πλειάδες

Στην υποενότητα αυτή, θα δούμε πώς να χρησιμοποιούμε λίστες, λεξικά και πλειάδες στην Python. Όπως είδαμε νωρίτερα στην υποενότητα με τους τύπους δεδομένων, οι λίστες και οι πλειάδες εμπίπτουν στην ομάδα τύπου δεδομένων των ακολουθιών και τα λεξικά στην ομάδα τύπου δεδομένων Mapping. Όλοι αυτοί οι τύποι δεδομένων έχουν διαφορετικά χαρακτηριστικά και ποιότητες και μας επιτρέπουν να επεξεργαστούμε τα δεδομένα με διαφορετικούς τρόπους.

3.3.1. Λίστες

Οι λίστες ξεκινούν με τη χρήση αγκύλων [] και περιέχουν μια σειρά από ταξινομημένα στοιχεία σε μία μεταβλητή.

Παράδειγμα:

```
adj = ["smart", "beautiful", "stunning"]
```

Τα στοιχεία που περιλαμβάνονται σε μια λίστα είναι ταξινομημένα, μεταβλητά (δηλαδή, μπορούν να αλλάξουν) και επιτρέπουν διπλότυπες τιμές.

Κάθε στοιχείο σε μια λίστα προσδιορίζεται από ένα ευρετήριο. Η δεικτοδότηση στην Python ξεκινά από το μηδέν [0] και αυξάνεται κατά ένα κάθε φορά. Έτσι, το πρώτο στοιχείο στη λίστα έχει ευρετήριο [0] και το δεύτερο [1] και ούτω καθεξής. Να σημειωθεί ότι η σειρά τους δεν μπορεί να αλλάξει. Όταν προστεθούν νέα στοιχεία, θα τοποθετηθούν στο τέλος της λίστας.

Επίσης, οι λίστες είναι μεταβλητές, πράγμα που σημαίνει ότι τα στοιχεία μπορούν να τροποποιηθούν, να αφαιρεθούν ή να προστεθούν μετά τη δημιουργία της λίστας. Δεδομένου ότι επιτρέπουν διπλότυπες τιμές, μπορείτε να βρείτε μια τιμή περισσότερες από μία φορές σε μια λίστα.

Παράδειγμα:

```
fruits = ["apple", "banana", "apple", "grapefruit"]
```

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Οι λίστες μπορούν επίσης να περιέχουν έναν συνδυασμό ακεραίων αριθμών, συμβολοσειρών ή δυαδικών τιμών:

```
employee1 = ["John Smith", 35, "male", 25000, True]
```

Εάν θέλετε να προσδιορίσετε τον αριθμό των στοιχείων που περιέχονται σε μια λίστα, μπορείτε να χρησιμοποιήσετε τη συνάρτηση **len()** :

```
print(len(employee1))
```

*** Σημειώστε ότι ο αριθμός των παρενθέσεων πρέπει να ταιριάζει από την αρχή μέχρι το τέλος για να εκτελεστεί ο κώδικας.**

Εάν παρατηρήσετε τον κώδικα στο παράδειγμα που μόλις είδαμε, τότε θα διαπιστώσετε ότι έχουμε μία παρένθεση στη αρχή της συνάρτησης len() και μία εντός της λίστας, και γι' αυτό βλέπετε δύο παρενθέσεις κλεισίματος στο τέλος του κώδικα.

Μπορείτε επίσης να χρησιμοποιήσετε τη συνάρτηση **list()** για να ξεκινήσετε μια λίστα:

```
new_list = list(("tomato", "cucumber", "peas", "peppers"))
print(new_list)
```

*** Σημειώστε ότι στην περίπτωση που χρησιμοποιήσετε τη συνάρτηση list() θα πρέπει να προσθέσετε διπλές παρενθέσεις για να δημιουργήσετε μια λίστα.**

Πρόσβαση σε στοιχεία λιστών

Εάν θέλετε να αποκτήσετε πρόσβαση σε ένα στοιχείο σε μια λίστα, πρέπει να χρησιμοποιήσετε τον αριθμό του ευρετηρίου του (index number).

Ας χρησιμοποιήσουμε στο ακόλουθο παράδειγμα τη λίστα employee1 που δημιουργήσαμε νωρίτερα για να λάβουμε την ηλικία του υπαλλήλου, η οποία είναι το δεύτερο στοιχείο στη λίστα:

```
print(employee1[1])
```

*** Θυμάστε γιατί χρησιμοποιούμε το 1 αντί το 2 ως αριθμό ευρετηρίου για το δεύτερο στοιχείο της λίστας; Αυτό συμβαίνει γιατί η δεικτοδότηση στην JavaScript ξεκινά πάντοτε από το 0.**

Εάν έχετε μια μεγάλη λίστα στοιχείων και θέλετε να λάβετε το τελευταίο στοιχείο από τη λίστα, μπορείτε να χρησιμοποιήσετε αρνητική ευρετηρίαση. Αυτό σημαίνει ότι η Python θα χρησιμοποιήσει το -1 για να αποκτήσει πρόσβαση στο τελευταίο στοιχείο που βρέθηκε στη λίστα και το -2 για να προσπελάσει το τελευταίο δεύτερο στοιχείο, και ούτω καθεξής.

Παράδειγμα:

```
print(employee1[-1])
```

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Μπορείτε επίσης να καθορίσετε ένα εύρος (range) ευρετηρίων που υποδεικνύει από ποιον αριθμό αυτό θα ξεκινήσει και σε ποιον αριθμό θα καταλήξει.

Παράδειγμα:

```
print(employee1[2:4])
```

Στο παράδειγμα πιο πάνω, το εύρος θα ξεκινήσει από το τρίτο στοιχείο της λίστας και θα τερματίσει στο τελευταίο, καθώς έχουμε μόνο 5 στοιχεία στη λίστα. Να θυμάστε πάντα ότι η αρίθμηση ξεκινά από το 0.

Εάν αποφασίσετε ότι δεν θέλετε να καθορίσετε τον αριθμό ευρετηρίου έναρξης, το εύρος θα ξεκινήσει από το πρώτο στοιχείο που βρίσκεται στη λίστα:

```
print(employee1[:4])
```

Μπορείτε επίσης να καθορίσετε μόνο τον αριθμό ευρετηρίου έναρξης χωρίς αριθμό τελικού ευρετηρίου:

```
print(employee1[1:])
```

Μπορείτε επίσης να χρησιμοποιήσετε την αρνητική ευρετηρίαση στην οποία αναφερθήκαμε προηγουμένως, για να αποκτήσετε πρόσβαση σε στοιχεία στο τέλος μιας λίστας:

```
print(employee1[-4:-1])
```

Αυτό θα μας δώσει το τέταρτο στοιχείο από το τέλος μιας λίστας μέχρι το τελευταίο στοιχείο που βρέθηκε σε αυτή.

Μπορείτε επίσης να ελέγξετε εάν ένα στοιχείο υπάρχει σε μια λίστα. Από τη λίστα επιθέτων που δημιουργήσαμε νωρίτερα, ας πούμε ότι θέλουμε να ελέγξουμε αν το επίθετο “beautiful” συμπεριλαμβάνεται σε αυτή:

```
adj = ["smart", "beautiful", "stunning"]
if "beautiful" in adj:
    print("Yes, beautiful is included in the list")
```

Προσθήκη στοιχείων σε λίστες

Εάν θέλετε να προσθέσετε νέα στοιχεία στο τέλος της λίστας, μπορείτε να χρησιμοποιήσετε τη μέθοδο **append()**. Στην περίπτωση που θέλουμε να προσθέσουμε ένα νέο επίθετο στη λίστα επιθέτων, μπορούμε να κάνουμε το εξής:

```
adj.append("innovative")
print(adj)
```

Έχετε επίσης την επιλογή να προσθέσετε νέα στοιχεία καθορίζοντας τον αριθμό ευρετηρίου. Μπορείτε να το κάνετε αυτό χρησιμοποιώντας τη μέθοδο **insert()**:

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

```
adj.insert(1, "innovative")
print(adj)
```

Όπως μπορείτε να δείτε στο πιο πάνω παράδειγμα, πρώτα θα πρέπει να καθορίσετε τον αριθμό ευρετηρίου και μετά το επίθετο που θέλετε να προσθέσετε χωρίζοντάς το με κόμμα.

Μια άλλη μέθοδος που μπορεί να χρησιμοποιηθεί για την προσθήκη νέων στοιχείων από μια λίστα στην τρέχουσα λίστα είναι η **extension().** Ας πούμε ότι έχουμε μια λίστα λέξεων στην οποία θέλουμε να προσθέσουμε επίθετα:

```
words = ["I", "am", "good", "you", "are", "nice"]
words.extend(adj)
print(words)
```

Τροποποίηση λιστών

Εάν θέλετε να τροποποιήσετε ένα συγκεκριμένο στοιχείο που βρίσκεται σε μια λίστα, πρέπει να ανατρέξετε στον αριθμό ευρετηρίου του. Για παράδειγμα, θα χρησιμοποιήσουμε τη λίστα employee1 για να τροποποιήσουμε την ηλικία του υπαλλήλου:

```
employee1 = ["John Smith", 35, "male", 25000, True]
employee1[1] = 36
print(employee1)
```

Μπορείτε επίσης να αλλάξετε πολλά στοιχεία σε μια λίστα εντός ενός καθορισμένου εύρους. Μπορείτε να το κάνετε αυτό ορίζοντας τα νέα στοιχεία που θα προστεθούν και ανατρέχοντας στους αριθμούς ευρετηρίου που θέλετε να τροποποιηθούν.

Για παράδειγμα, πρόκειται να τροποποιήσουμε τα τρία πρώτα στοιχεία της λίστας employee1, επειδή ένας συγκεκριμένος υπάλληλος δεν εργάζεται πλέον σε μια εταιρεία:

```
employee1[0:2] = ["Ella Smith", 30, "female"]
print(employee1)
```

Αφαίρεση στοιχείων από λίστες

Εάν έχετε αποφασίσει να αφαιρέσετε ένα καθορισμένο στοιχείο από μια λίστα, μπορείτε να χρησιμοποιήσετε τη μέθοδο **remove()**. Ας υποθέσουμε ότι θέλουμε να αφαιρέσουμε τη λογική δυαδική τιμή True από τη λίστα employee1:

```
employee1 = ["Ella Smith", 30, "female", 25000, True]
employee1.remove(True)
print(employee1)
```

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Μπορείτε επίσης να αφαιρέσετε ένα στοιχείο καθορίζοντας τον αριθμό ευρετηρίου του με τη μέθοδο **pop()** :

```
employee1 = ["Ella Smith", 30, "female", 25000, True]
employee1.pop(4)
print(employee1)
```

Εάν δεν καθορίσετε τον αριθμό ευρετηρίου, τότε το τελευταίο στοιχείο που βρέθηκε στη λίστα θα αφαιρεθεί:

```
employee1.pop()
print(employee1)
```

Εναλλακτικά, μπορείτε να χρησιμοποιήσετε την εντολή **del** για να καταργήσετε ένα στοιχείο με καθορισμένο ευρετήριο:

```
del employee1[1]
```

Επίσης, μπορείτε να χρησιμοποιήσετε την εντολή **del** για να διαγράψετε ολόκληρη τη λίστα:

```
del employee1
```

Ωστόσο, μπορεί να θέλετε να αδειάσετε τα περιεχόμενα μια λίστας. Στην περίπτωση αυτή, μπορείτε να χρησιμοποιήσετε τη μέθοδο **clear()**:

```
employee1 = ["Ella Smith", 30, "female", 25000, True]
employee1.clear()
print(employee1)
```

Αντιγραφή λιστών

Υπάρχουν δύο τρόποι με τους οποίους μπορείτε να αντιγράψετε μια λίστα με ένα νέο όνομα μεταβλητής. Ο πρώτος τρόπος είναι να χρησιμοποιήσετε τη μέθοδο **copy()**:

```
employee1 = ["Ella Smith", 30, "female", 25000, True]
employee2 = employee1.copy()
print(employee2)
```

Ο δεύτερος τρόπος με τον οποίο μπορεί να γίνει αντιγραφή μιας λίστας είναι χρησιμοποιώντας τη μέθοδο **list()**:

```
employee2 = list(employee1)
print(employee2)
```

Ταξινόμηση λιστών (sort)

Εάν θέλετε να ταξινομήσετε μια λίστα με αλφαριθμητική σειρά, φθίνουσα ή αύξουσα, μπορείτε να χρησιμοποιήσετε τη μέθοδο **sort()**.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

***Σημειώστε ότι η μέθοδος sort() θα ταξινομήσει τα στοιχεία με αύξουσα σειρά από προεπιλογή, εάν δεν οριστεί διαφορετικά.**

Ας ταξινομήσουμε με αλφαβητική σειρά την πιο κάτω λίστα λέξεων, όπως φαίνεται στο παράδειγμα:

```
adj = ["smart", "beautiful", "stunning"]
adj.sort()
print(adj)
```

Ένα άλλο παράδειγμα που περιλαμβάνει μια αριθμητική λίστα:

```
monthly_income= [5000, 1000, 2500, 1300, 1200, 1500, 2300]
monthly_income.sort()
print(monthly_income)
```

Αυτά τα δύο παραδείγματα θα παρατίθενται σε αύξουσα σειρά, δηλ. από τον μικρότερο αριθμό στον μεγαλύτερο ή από το a στο z σύμφωνα με τους εκάστοτε τύπους δεδομένων.

Εάν θέλετε να ταξινομήσετε μια λίστα με φθίνουσα σειρά, τότε αυτό μπορεί να καθοριστεί χρησιμοποιώντας το όρισμα **reverse = True** στην παρένθεση. Θυμάστε όταν αναφέραμε τους μπούλειους τύπους (Boolean types) ως τύπους δεδομένων; Στην περίπτωση αυτή λοιπόν, το όρισμα **reverse = True** χρησιμοποιείται για να επιτρέψει την αντιστροφή αλφαριθμητικού, δηλώνοντας την τιμή του ορίσματος reverse ίση με True (αληθές), αφού από προεπιλογή είναι False (ψευδές).

Παράδειγμα 1:

```
adj = ["smart", "beautiful", "stunning"]
adj.sort(reverse=True)
print(adj)
```

Παράδειγμα 2:

```
monthly_income= [5000, 1000, 2500, 1300, 1200, 1500, 2300]
monthly_income.sort()
print(monthly_income)
```

Η ταξινόμηση εφαρμόζεται από προεπιλογή με διάκριση πεζών-κεφαλαίων, πράγμα που σημαίνει ότι όλα τα κεφαλαία γράμματα ταξινομούνται πριν από τα πεζά.

Στο παράδειγμα πιο κάτω, μπορείτε να λάβετε μερικά εκπληκτικά αποτελέσματα εξαιτίας αυτής της δυνατότητας:

```
words = ["I", "am", "good", "you", "are", "nice"]
words.sort()
print(words)
```

Για να επιλυθεί αυτό το πρόβλημα, μπορείτε να χρησιμοποιήσετε τη μέθοδο sort χωρίς διάκριση πεζών-κεφαλαίων με το όρισμα **key = str.lower**. Εδώ, χρησιμοποιείται ο τύπος δεδομένων της συμβολοσειράς για να μετατρέψει τα γράμματα σε πεζά.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Παράδειγμα:

```
words = ["I", "am", "good", "you", "are", "nice"]
words.sort(key=str.lower)
print(words)
```

Επιπλέον, μπορείτε να χρησιμοποιήσετε τη μέθοδο **reverse()** για να διατηρήσετε τη σειρά μιας λίστας ανεξάρτητα από την αλφαβητική της σειρά:

```
words = ["I", "am", "good", "you", "are", "nice"]
words.reverse()
print(words)
```

Συνένωση λιστών στην Python

Θυμάστε όταν είδαμε τους τελεστές που χρησιμοποιούνται στην Python και πιο συγκεκριμένα στην περίπτωση των συμβολοσειρών; Μία από τις μεθόδους που χρησιμοποιούνται για την συνένωση λιστών στην Python είναι το **σύμβολο + (συν)**:

```
adj = ["smart", "beautiful", "stunning"]
words = ["I", "am", "good", "you", "are", "nice"]
all_words = adj + words
print(all_words)
```

Εδώ, δημιουργήσαμε μια νέα λίστα συνενώνοντας τις δύο ήδη υπάρχουσες λίστες επιθέτων και λέξεων. Ανεξάρτητα από τους τύπους δεδομένων που βρίσκονται σε κάθε λίστα, μπορείτε να τους συνενώσετε με τον τελεστή +.

Εάν θέλετε να προσθέσετε στοιχεία από τη μια λίστα στην άλλη, απλώς χρησιμοποιήστε τη μέθοδο **extend()**:

```
words.extend(adj)
print(words)
```

Στο παράδειγμα αυτό, η λίστα των λέξεων ενώνεται με τη λίστα των επιθέτων. Επομένως, η λίστα των λέξεων θα περιέχει τώρα και τις δύο λίστες.

Εξάσκηση: https://www.w3schools.com/python/exercise.asp?filename=exercise_lists1

3.3.2. Πλειάδες

Οι πλειάδες περιέχουν επίσης μια ακολουθία στοιχείων σε μία μεταβλητή όπως οι λίστες. Οι πλειάδες όπως και οι λίστες, μπορούν να ταξινομηθούν. Ωστόσο, μια βασική διαφορά μεταξύ των λιστών και των πλειάδων είναι ότι οι πλειάδες είναι αμετάβλητες (δηλαδή δεν μπορούν να τροποποιηθούν). Οι πλειάδες τοποθετούνται επίσης μέσα σε ζεύγη αγκύλων().

Παράδειγμα:

```
coordinates = (38.09, -77.06)
```

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Στην πραγματικότητα, οι πλειάδες χρησιμοποιούνται συνήθως όπως ένα λεξικό της Python, χωρίς όμως λέξεις-κλειδιά για την αποθήκευση δεδομένων, καθώς μπορούν να επαναληφθούν ταχύτερα ενώ τα στοιχεία τους δεν μπορούν να τροποποιηθούν. Οι πλειάδες επιτρέπουν επίσης διπλές τιμές.

Εάν θέλετε να δείτε πόσα στοιχεία υπάρχουν σε μια πλειάδα, μπορείτε να χρησιμοποιήσετε τη συνάρτηση **len()**:

```
print(len(coordinates))
```

Για να δημιουργήσετε μια πλειάδα που περιέχει μόνο ένα στοιχείο, θα πρέπει να προσθέσετε κόμμα μετά από αυτό. Διαφορετικά, η Python δεν θα το αναγνωρίσει ως πλειάδα. Θα χρησιμοποιήσουμε τη συνάρτηση **type** για να προσδιορίσουμε τη διαφορά μεταξύ της χρήσης του κόμματος.

Παράδειγμα:

```
name1 = ("John",)
print(type(name1))

name2 = ("John") #not a tuple
print(type(name2))
```

Κατά τρόπο παρόμοιο όπως και οι λίστες, οι πλειάδες μπορούν να περιέχουν αντικείμενα οποιουδήποτε τύπου δεδομένων είτε σε ξεχωριστές πλειάδες είτε ως συνδυασμός αυτών, όπως φαίνεται στο πιο κάτω παράδειγμα:

```
personal_info1 = ("John Kent", 34, True, "male")
print(personal_info1)
```

Ένας εναλλακτικός τρόπος για να δημιουργήσετε μια πλειάδα είναι να χρησιμοποιήσετε τον κατασκευαστή **tuple()** :

```
coordinates = tuple((38.09, -77.06))
print(coordinates)
```

*** Σημειώστε ότι όταν χρησιμοποιείτε τον κατασκευαστή tuple(), πρέπει να προσθέσετε διπλές αγκύλες (()).**

Πρόσβαση σε πλειάδες

Μπορούμε να έχουμε πρόσβαση στα αντικείμενα μέσα σε μια πλειάδα μέσω του τελεστή ευρετηρίασης κατά τρόπο παρόμοιο με τις λίστες. Αυτό σημαίνει ότι κάθε στοιχείο σε μια πλειάδα αντιστοιχεί με έναν αριθμό. Η ευρετηρίαση βασίζεται στη σειρά των στοιχείων σε μια πλειάδα η οποία ακολουθεί τη δεικτοδότηση της Python, ξεκινώντας δηλαδή από το 0 για το πρώτο στοιχείο και ανεβαίνοντας κατά 1 κάθε φορά για τα υπόλοιπα στοιχεία της πλειάδας.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Για να αποκτήσετε πρόσβαση σε ένα συγκεκριμένο στοιχείο, πρέπει να καθορίσετε τη θέση του στοιχείου μέσα σε ένα ζευγάρι αγκύλων σύμφωνα με τον αριθμό ευρετηρίου του:

```
personal_info1 = ("John Kent", 34, True, "male")
print(personal_info1)
```

***Θυμηθείτε ότι ο αριθμός ευρετηρίου 1 στην πλειάδα αντιστοιχεί στο δεύτερο στοιχείο, δηλαδή το 34.**

Μπορείτε επίσης να χρησιμοποιήσετε αρνητική ευρετηρίαση για πρόσβαση στο τελευταίο στοιχείο μιας πλειάδας [-1] ή στο τελευταίο δεύτερο στοιχείο [-2] και ούτω καθεξής.

```
personal_info1 = ("John Kent", 34, True, "male")
print(personal_info1[-1])
```

Αυτό θα εμφανίσει το στοιχείο "male" στην πλειάδα.

Εάν θέλετε, μπορείτε να καθορίσετε ένα εύρος ευρετηρίων (range) που υποδεικνύει από ποιον αριθμό αυτό θα ξεκινήσει και σε ποιον αριθμό θα καταλήξει.

Παράδειγμα:

```
personal_info1 = ("John Kent", 34, True, "male")
print(personal_info1[1:3])
```

Στο πιο πάνω παράδειγμα, η πλειάδα θα ξεκινήσει από το τρίτο στοιχείο της λίστας και θα τελειώσει στο τελευταίο, αφού έχουμε μόνο 5 στοιχεία στη λίστα. Να θυμάστε πάντα ότι η δεικτοδότηση στην Python ξεκινά πάντοτε από το 0.

Εάν αποφασίσετε ότι δεν θέλετε να καθορίσετε τον αριθμό ευρετηρίου έναρξης, το εύρος θα ξεκινήσει από το πρώτο στοιχείο που βρίσκεται στην πλειάδα:

```
print(personal_info1[:3])
```

Μπορείτε επίσης να καθορίσετε μόνο τον αριθμό ευρετηρίου έναρξης χωρίς αριθμό τελικού ευρετηρίου:

```
print(personal_info1[1:])
```

Θυμάστε όταν χρησιμοποιήσαμε την αρνητική ευρετηρίαση, μπορείτε λοιπόν να την χρησιμοποιήσετε και εδώ για να αποκτήσετε πρόσβαση σε στοιχεία σε μια λίστα από το τέλος της:

```
print(personal_info1[-3:-1])
```

Αυτό θα μας δώσει το τέταρτο στοιχείο από το τέλος μέχρι το τελευταίο στην πλειάδα.

Μπορείτε επίσης να ελέγξετε εάν ένα στοιχείο υπάρχει σε μια πλειάδα κατά τρόπο παρόμοιο με εκείνο που είδαμε με τις λίστες. Ας πούμε ότι θέλουμε να ελέγξουμε αν το επίθετο “beautiful” περιλαμβάνεται στην πλειάδα adj:

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

```
adj = ("smart", "beautiful", "stunning")
if "beautiful" in adj:
    print("Yes, beutiful is included in the adj tuple")
```

Προσθήκη, τροποποίηση και αφαίρεση στοιχείων από πλειάδες

Τεχνικά μιλώντας, δεν μπορείτε να προσθέσετε ή να αφαιρέσετε στοιχεία από πλειάδες καθώς αυτές είναι αμετάβλητες (δηλαδή, δεν μπορούν να τροποποιηθούν). Ωστόσο, υπάρχει μια λύση. Απλώς χρειάζεται να μετατρέψετε την πλειάδα σε λίστα, να την τροποποιήσετε και στη συνέχεια να τη μετατρέψετε ξανά σε πλειάδα.

Ας δούμε ένα παράδειγμα για να το καταλάβουμε λίγο καλύτερα:

```
personal_info1 = ("John Kent", 34, True, "male")
#here we create a new variable that converts the tuple into a list
modify_pinfo1 = list(personal_info1)
modify_pinfo1[1] = 35 #modify/change the item that we want
personal_info1  = tuple(modify_pinfo1)
print(personal_info1)
```

Εάν θέλετε να προσθέσετε στοιχεία σε μια πλειάδα, υπάρχουν δύο τρόποι με τους οποίους μπορείτε να εκτελέσετε αυτή την ενέργεια. Ο πρώτος τρόπος ακολουθεί την ίδια λογική με το παράδειγμα που είδαμε στην περίπτωση που θέλαμε να τροποποιήσουμε ένα στοιχείο σε μια πλειάδα.

Παράδειγμα:

```
personal_info1 = ("John Kent", 34, True, "male")
modify_pinfo1 = list(personal_info1) #convert tuple into list in a new variable
modify_pinfo1.append("1 Charming Street") #adding new item
personal_info1 = tuple(modify_pinfo1) #convert it back
```

Ο δεύτερος τρόπος για να προσθέσετε ένα νέο στοιχείο είναι προσθέτοντας μια πλειάδα σε μια άλλη πλειάδα. Μπορείτε να δημιουργήσετε μια νέα πλειάδα που περιέχει το στοιχείο που θέλετε να προσθέσετε στην υπάρχουσα πλειάδα και απλά να την προσθέσετε:

```
personal_info1 = ("John Kent", 34, True, "male")
modify_pinfo1 = ("1 Charming Street",) #creating new tuple, do not forget the comma
personal_info1 += modify_pinfo1 #adding new tuple into the existing one
print(personal_info1)
```

Στο παράδειγμα πιο κάτω, θα δούμε πώς μπορούμε να αφαιρέσουμε αντικείμενα από πλειάδες. Και εδώ ισχύει η ίδια λογική με τα προηγούμενα παραδείγματα που είδαμε αναφορικά με την προσθήκη ή την τροποποίηση στοιχείων σε πλειάδες.

Παράδειγμα:

```
personal_info1 = ("John Kent", 34, True, "male")
modify_pinfo1 = list(personal_info1) #convert tuple into list in a new variable
modify_pinfo1.remove("male") #delete item male
personal_info1 = tuple(modify_pinfo1) #convert back
print(personal_info1)
```

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Στο παρακάτω παράδειγμα, θα δούμε πώς μπορείτε να διαγράψετε μια πλειάδα χρησιμοποιώντας την εντολή del:

```
personal_info1 = ("John Kent", 34, True, "male")
del personal_info1 #it is now deleted
print(personal_info1) # this will raise an error since we have deleted it
```

Όταν δημιουργούμε μια πλειάδα, αναθέτουμε ουσιαστικά τιμές ή αντικείμενα ως περιεχόμενό της. Αυτή η διαδικασία ονομάζεται επίσης «packing» μιας πλειάδας:

```
personal_info1 = ("John Kent", 34, True, "male")
```

Στην Python, έχετε επίσης τη δυνατότητα να εξαγάγετε τις τιμές σε μεταβλητές και αυτή η διαδικασία ονομάζεται «unpacking»:

```
personal_info1 = ("John Kent", 34, True, "male")
# assigned separate variable name to each tuple item
(name, age, employed, sex) = personal_info1
# printing each new item variable from tuple
print(name)
print(age)
print(employed)
print(sex)
```

*** Σημειώστε ότι τα ονόματα των μεταβλητών πρέπει να ταιριάζουν με τα στοιχεία που περιέχονται στην πλειάδα, διαφορετικά μπορείτε να χρησιμοποιήσετε έναν αστερίσκο (*) για να συλλέξετε τα υπόλοιπα ως μια λίστα.**

Ας υποθέσουμε ότι θέλετε να εξαγάγετε μόνο δύο μεταβλητές από την πλειάδα personal_info1, τότε μπορείτε λοιπόν να χρησιμοποιήσετε τον αστερίσκο:

```
personal_info1 = ("John Kent", 34, True, "male")
(name, *demographics) = personal_info1
# all items after the first will be contained in the demographics variable
print(name)
print(demographics)
```

Εάν προσθέσετε τον αστερίσκο σε ένα άλλο όνομα μεταβλητής αντί του τελευταίου, τότε οι τιμές θα εκχωρηθούν στη μεταβλητή με τον αστερίσκο έως ότου οι τιμές που απέμειναν ταιριάζουν με τις εναπομείναντες μεταβλητές:

```
personal_info1 = ("John Kent", 34, True, "male")
(name, *age_job, sex ) = personal_info1
print(name)
print(age_job)
print(sex)
```

Συνένωση πλειάδων

Εάν θέλετε να συνδέσετε δύο πλειάδες, μπορείτε να χρησιμοποιήσετε τον **τελεστή πρόσθεσης (+)** :

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

```
personal_info1 = ("John Kent", 34, True, "male")
personal_info2 = ("Kylie Smith", 40, True, "female")
total_pinfo = personal_info1 + personal_info2
print(total_pinfo)
```

Έχετε επίσης την επιλογή να πολλαπλασιάσετε το περιεχόμενο μιας πλειάδας για πολλές φορές χρησιμοποιώντας τον **τελεστή πολλαπλασιασμού (*)** :

```
personal_info1 = ("John Kent", 34, True, "male")
general = personal_info1 * 2
print(general)
```

Εξάσκηση: https://www.w3schools.com/python/exercise.asp?filename=exercise_tuples1

3.3.3. Λεξικά

Τα λεξικά εμπίπτουν στην ομάδα τύπου δεδομένων Mapping, καθώς μπορούν να αποθηκεύουν δεδομένα σε ζεύγη κλειδιών και τιμών. Περιέχουν μια ακολουθία στοιχείων ζεύγους που είναι ταξινομημένα και μεταβλητά (δηλαδή μπορούν να τροποποιηθούν) ενώ δεν επιτρέπουν διπλές τιμές. Τα λεξικά περικλείονται σε ένα ζεύγος αγκίστρων {}.

*** Σημειώστε ότι τα λεξικά στην Python 3.6 και σε προηγούμενες εκδόσεις της τα λεξικά δεν ήταν ταξινομημένα, αλλά με την νεότερη έκδοση Python 3.7. αυτά έγιναν ταξινομημένα.**

Ένα παράδειγμα της χρήσης λεξικού στην Python είναι η μετάφραση των αγγλικών σε μια άλλη γλώσσα. Ας υποθέσουμε ότι θέλουμε να μεταφράσουμε από τα αγγλικά στα ισπανικά:

```
eng2esp = {
        "hello": "hola",
        "good morning": "buenas dias",
        "good afternoon": "buenas tardes",
        "how are you": "como estas"
        }
print(eng2esp)
```

Μια άλλη επιλογή είναι να δημιουργήσετε ένα άδειο λεξικό και να προσθέσετε τα στοιχεία ένα προς ένα:

```
eng2esp = {}
eng2esp ["hello"] = "hola"
eng2esp["good morning"] = "buenas dias" # and so on
```

Όπως αναφέραμε, τα λεξικά δεν επιτρέπουν διπλές τιμές για τα κλειδιά τους, όπως π.χ. το «hello» στο λεξικό eng2esp που δημιουργήσαμε παραπάνω.

Εάν θέλετε να μάθετε πόσα στοιχεία υπάρχουν σε ένα λεξικό, μπορείτε να χρησιμοποιήσετε τη συνάρτηση **len()** με τον ίδιο τρόπο που την εφαρμόσαμε προηγουμένως στις λίστες και τις πλειάδες:

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

```
print(len(eng2esp))
```

Κατά τρόπο παρόμοιο με τις πλειάδες και τις λίστες λοιπόν, τα λεξικά μπορούν να περιέχουν οποιονδήποτε τύπο δεδομένων:

```
sofas_inv = { "sofas": 5,
              "colours": [ "red", "blue", "grey"],
              "year": [2016, 2022, 2018],
              "sofabed": False}
```

Όπως μπορείτε να δείτε στο πιο πάνω παράδειγμα, έχουμε τύπους δεδομένων όπως ακέραιους αριθμούς, συμβολοσειρές, μπούλειους τύπους και λίστες.

Για να προσδιορίσετε τον τύπο δεδομένων της μεταβλητής «sofabed», πρέπει να χρησιμοποιήσετε τη συνάρτηση **type()** :

```
print(type(sofas_inv))
```

Πρόσβαση σε στοιχεία λεξικών

Εάν θέλετε να αναζητήσετε ποια είναι η αντίστοιχη λέξη του "hello" στα Ισπανικά, τότε μπορείτε να χρησιμοποιήσετε τα εξής:

```
eng2esp = {
        "hello": "hola",
        "good morning": "buenas dias",
        "good afternoon": "buenas tardes",
        "how are you": "como estas"
        }
print(eng2esp["hello"])
```

Αυτό θα εκτυπώσει το αποτέλεσμα: "hola".

Όταν αναζητάτε στοιχεία σε ένα λεξικό, πρέπει να περικλείετε το όνομα του κλειδιού σε ένα ζεύγος τετράγωνων αγκύλων:

```
# here we created a variable to store the value "hola"
esp_hello = eng2esp["hello"]
```

Στο παράδειγμα πιο πάνω, το "hello" είναι το κλειδί για την τιμή του "hola".

Μια εναλλακτική μέθοδος είναι να χρησιμοποιήσετε τη μέθοδο **get()** με την οποία θα έχετε ακριβώς το ίδιο αποτέλεσμα:

```
esp_hello = eng2esp.get("hello")
```

Εάν θέλετε να αναζητήσετε όλα τα κλειδιά της λίστας που περιέχει ένα λεξικό, μπορείτε να χρησιμοποιήσετε τη μέθοδο **keys()**:

```
all_keys = eng2esp.keys()
print(all_keys)
```

Οποιεσδήποτε αλλαγές γίνουν στο λεξικό eng2esp θα αντικατοπτρίζονται στη λίστα all_keys που δημιουργήσαμε:

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

```
eng2esp["bye"] = "adios"
print(all_keys)  #after the addition
```

Επίσης, έχετε την επιλογή να λάβετε όλες τις τιμές που περιέχονται σε ένα λεξικό με τη μέθοδο **values()**:

```
all_values = eng2esp.values()
print(all_values)
```

Η ίδια λογική ισχύει και για τις τιμές ως προς τις αλλαγές που έγιναν. Εάν προσθέσουμε μια νέα τιμή, θα προστεθεί στη λίστα των τιμών που δημιουργήσαμε.

Μια άλλη επιλογή που έχετε είναι να λάβετε όλα τα ζεύγη κλειδιών/τιμών χρησιμοποιώντας τη μέθοδο **items()** :

```
all_pairs = eng2esp.items()
print(all_pairs)
```

Και πάλι, τυχόν αλλαγές που έγιναν στο λεξικό είτε στα κλειδιά είτε στις τιμές του λεξικού θα αντικατοπτρίζονται στη λίστα all_pairs που δημιουργήσαμε.

Εάν δεν είστε βέβαιοι αν ένα συγκεκριμένο κλειδί υπάρχει ήδη σε ένα λεξικό, μπορείτε να το αναζητήσετε με βάση τον πιο κάτω τρόπο:

```
if "hello" in eng2esp:
    print( "Yes, 'hello' is one of the keys of the eng2esp dictionary")
```

Προσθήκη, τροποποίηση ή αφαίρεση στοιχείων από λεξικά

Για να προσθέσετε στοιχεία σε ένα λεξικό, υπάρχουν δύο τρόποι με τους οποίους μπορείτε να πραγματοποιήσετε αυτή την ενέργεια. Σύμφωνα με τον πρώτο τρόπο, μπορείτε να προσθέσετε ένα νέο κλειδί μέσω του τελεστή ευρετηρίασης και την αντίστοιχη τιμή που πρέπει να προστεθεί:

```
sofas_inv = { "sofas": 5,
              "colours": [ "red", "blue", "grey"],
              "year": [2016, 2022, 2018],
              "sofabed": False}
sofas_inv["material"] = "leather"
print(sofas_inv)
```

Ο δεύτερος τρόπος είναι να χρησιμοποιήσετε τη μέθοδο **update()**, όπου πρέπει να καθορίσετε το λεξικό ή το επαναλαμβανόμενο ζεύγος κλειδιών/τιμών:

```
sofas_inv.update({"material" : "leather"})
print(sofas_inv)
```

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

***Σημειώστε ότι θα πρέπει να συμπεριλάβετε ένα ζεύγος αγκίστρων εντός των αγκύλων για να καταστεί δυνατή η υπόδειξη των ζευγών κλειδιών/τιμών σε ένα λεξικό.**

Εάν θέλετε να τροποποιήσετε τα στοιχεία σε ένα λεξικό, τότε μπορείτε να χρησιμοποιήσετε τις ίδιες προαναφερθείσες μεθόδους και πάλι. Σύμφωνα με την πρώτη μέθοδο θα πρέπει να αναφερθείτε στο όνομα του κλειδιού για να αλλάξετε μια τιμή. Για παράδειγμα, εάν πουλήθηκε ένας καναπές, μπορείτε να αλλάξετε την ποσότητα των καναπέδων στο κατάστημα:

```
sofas_inv = { "sofas": 5,
              "colours": [ "red", "blue", "grey"],
              "year": [2016, 2022, 2018],
              "sofabed": False}
sofas_inv["sofas"] = 4
```

Η δεύτερη επιλογή που έχετε είναι να χρησιμοποιήσετε τη μέθοδο **update()**:

```
sofas_inv.update({"sofas": 4})
```

Υπάρχουν 3 διαφορετικοί τρόποι με τους οποίους μπορείτε για να αφαιρέσετε στοιχεία από ένα λεξικό. Ο πρώτος τρόπος είναι με τη μέθοδο **pop()**:

```
sofas_inv = { "sofas": 5,
              "colours": [ "red", "blue", "grey"],
              "year": [2016, 2022, 2018],
              "sofabed": False}
sofas_inv.pop("sofabed")
print(sofas_inv)
```

Ο δεύτερος τρόπος είναι να χρησιμοποιήσετε την εντολή **del**:

```
del sofas_inv["sofabed"]
print(sofas_inv)
```

Ο τρίτος διαθέσιμος τρόπος, ο οποίος ισχύει μόνο στην έκδοση 3.7. της Python, αφαιρεί το τελευταίο στοιχείο που προστέθηκε. Ωστόσο, σε παλαιότερες εκδόσεις της Python, με τον τρόπο αυτό θα αφαιρεθεί ένα τυχαίο στοιχείο.

Παράδειγμα:

```
sofas_inv.popitem()
print(sofas_inv)
```

Εάν θέλετε να διαγράψετε εντελώς ένα λεξικό, μπορείτε να χρησιμοποιήσετε την εντολή **del**:

```
del sofas_inv
print(sofas_inv) #Python will raise error since the dictionary no longer exists
```

Μια άλλη επιλογή που έχετε είναι απλώς να αδειάσετε το λεξικό χρησιμοποιώντας τη μέθοδο **clear()**:

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

```
sofas_inv.clear()
print(sofas_inv)  # you will have an empty dictionary
```

Αντιγραφή λεξικών

Θυμάστε όταν στην περίπτωση της αντιγραφής των λιστών είπαμε ότι δεν είναι εφικτό να αντιγράψουμε ένα αντικείμενο σε άλλο χρησιμοποιώντας το σύμβολο ίσων (=), γιατί έτσι οι αλλαγές της μιας λίστας θα μεταφέρονταν στην άλλη; Όπως και οι λίστες λοιπόν, τα λεξικά δεν πρέπει να αντιγράφονται με αυτόν τον τρόπο. Υπάρχουν δύο τρόποι με τους οποίους μπορείτε να δημιουργήσετε ένα αντίγραφο ενός λεξικού.

Ο πρώτος τρόπος είναι να χρησιμοποιήσετε τη μέθοδο **copy()**:

```
new_inv = sofas_inv.copy()
print(new_inv)
```

Ο δεύτερος τρόπος είναι να χρησιμοποιήσετε τη μέθοδο **dict()**:

```
new_inv = dict(sofas_inv)
```

3.4. Συνθήκες και βρόχοι

Στην υποενότητα αυτή, θα εξηγήσουμε τη λογική πίσω από τη χρήση των εντολών ελέγχου ροής οι οποίες γίνονται υπό προϋποθέσεις ή βρόχους. Οι εντολές ελέγχου ροής αποτελούν βασικές και χρήσιμες γνώσεις στις γλώσσες προγραμματισμού γενικότερα.

- ⇒ Οι εντολές **if, elif, else** .
- ⇒ Οι εντολές βρόχου **for** και **while**

Πριν προχωρήσουμε στην αναλυτικότερη παρουσίαση των εντολών ελέγχου ροής, θα πρέπει πρώτα να μάθουμε για τον τελεστή == και τις εκφράσεις Λογικό Ή, ΚΑΙ, ΟΧΙ (and, or, not).

3.4.1. Εκφράσεις Λογικό Ή, ΚΑΙ, ΟΧΙ (and, or, not)

Οι εντολές ελέγχου ροής χρησιμοποιούνται για να ελεγχθεί αν μια έκφραση ή μια δήλωση είναι αληθής ή ψευδής. Υπάρχουν δύο τρόποι με τους οποίους μπορείτε να γράψετε τις εντολές ελέγχου ροής. Ο πρώτος τρόπος χρησιμοποιεί τον τελεστή == ο οποίους συγκρίνει δύο τιμές για να επιστρέψει την τιμή True (Αληθές) ή False (Ψευδές):

```
print(8 == 8)
```

Έξοδος:

True

```
print(9 == 8)
```

Έξοδος:

False

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Και στις δύο πιο πάνω δηλώσεις, ο τελεστής == αξιολογεί αν οι δύο ακέραιοι αριθμοί είναι ίσοι (δηλ. έχουν την ίδια τιμή). Όπως μπορείτε να δείτε, στην πρώτη δήλωση η έξοδος είναι True ενώ στην δεύτερη δήλωση η έξοδος είναι False.

Ο τελεστής == είναι ένας από τους πολλούς τελεστές σύγκρισης που χρησιμοποιούνται στην Python. Οι υπόλοιποι τελεστές σύγκρισης παρατίθενται πιο κάτω:

x > y	Το x είναι μεγαλύτερο από το y
x < y	Το x είναι μικρότερο από το y
x >= y	Το x είναι μεγαλύτερο ή ίσο του y
x <= y	Το x είναι μικρότερο ή ίσο του y
x != y	Το x δεν είναι ίσο με το y

Πιθανότατα να έχετε συναντήσει αυτούς τους τελεστές στο παρελθόν, ωστόσο, ορισμένα από αυτά τα σύμβολα έχουν διαφορετικές λειτουργίες στην Python.

Για παράδειγμα, αν θέλατε να δείτε αν μια τιμή είναι ίδια με μια άλλη, δεν θα χρησιμοποιούσατε το σύμβολο ίσων (=) επειδή αυτό θα εκχωρούσε την τιμή στη μεταβλητή που βρίσκεται στην αριστερή πλευρά.

Για να συγκρίνετε δύο τιμές, πρέπει να χρησιμοποιήσετε **διπλά σύμβολα ίσων (==).**

Επίσης, αν θέλετε να ελέγξετε αν μια τιμή είναι **μεγαλύτερη ή ίση (>=)** από/με μια άλλη τιμή, τότε θα πρέπει να χρησιμοποιήσετε τον τελεστή **μεγαλύτερο από (>)** ή τον τελεστή **μικρότερο από (<)** αντίστοιχα, αντί να χρησιμοποιήσετε το **σύμβολο ίσων (=).**

Αυτές είναι μερικές σημαντικές διακρίσεις που θα πρέπει να θυμάστε όταν προγραμματίζετε στην Python για να αποφεύγετε τυχόν ανεπιθύμητα λάθη ή εκπλήξεις.

Ας δούμε μερικά παραδείγματα των τελεστών σύγκρισης με τη λέξη-κλειδί **if**:

```
x = 55
y = 500
if y > x:
    print("y is greater than x")
```

Στο παράδειγμα αυτό, συγκρίνουμε τις μεταβλητές x και y για να δούμε αν το y είναι μεγαλύτερο από το x. Δεδομένου ότι γνωρίζουμε ήδη ότι το y είναι μεγαλύτερο από το x, επιλέγουμε να το εκτυπώσουμε ως έξοδο.

*** Η εσοχή κώδικα (indentation) που εμφανίζεται στη δεύτερη γραμμή (δηλαδή, το κενό διάστημα που βρίσκεται στην αρχή της γραμμής) είναι σημαντική καθώς αυτή καθορίζει το επίπεδο εσοχής της λογικής γραμμής, και αυτό με τη σειρά του προσδιορίζει την ομαδοποίηση των εντολών. Αυτό σημαίνει ότι οι εντολές που πάνε μαζί πρέπει και**

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

οφείλουν να έχουν το ίδιο επίπεδο εσοχής. Κάθε τέτοια ομάδα εντολών καλείται πλοκάδα (block). Η εσοχή κώδικα είναι ένα σημαντικό μέρος της σύνταξης στην Python, η οποία θα σας επιτρέψει αν αποφύγετε πολλά τυχόν σφάλματα.

Μια άλλη λέξη-κλειδί που χρησιμοποιείται στις εκφράσεις υπό όρους είναι το **elif**, το οποίο χρησιμοποιείται ως εναλλακτική λύση εάν η πρώτη συνθήκη δεν είναι αληθής. Ας δούμε ένα παράδειγμα τελεστών σύγκρισης με τη λέξη-κλειδί **elif**:

```
x = 55
y = 55
if y < x:
    print("y is greater than x")
elif x == y:
        print("x is equal to y")
```

Στο παράδειγμα αυτό, έχουμε την πρώτη συνθήκη που ελέγχει αν το y είναι μεγαλύτερο από το x. Δεδομένου ότι δεν είναι, το πρόγραμμα μετακινείται στη δεύτερη συνθήκη που δηλώνεται με τον όρο elif **(ο οποίος συνδυάζει δύο συσχετιζόμενες εντολές if else-if else)** και ο οποίος ελέγχει αν το x είναι ίσο με το y. Εφόσον η δεύτερη συνθήκη είναι αληθής, εκτυπώνεται η αντίστοιχη δήλωση στην κονσόλα.

Η επόμενη λέξη-κλειδί που χρησιμοποιείται στις εντολές ελέγχου ροής είναι το **else**, που επιτρέπει την επεξεργασία ενός άλλου συνόλου εντολών εάν οι προηγούμενες συνθήκες δεν είναι αληθείς.

Ας δούμε ένα παράδειγμα με τον όρο **else** για να καταλάβουμε καλύτερα πώς λειτουργεί:

```
x = 500
y = 55
if y > x:
    print("y is greater than x")
elif x == y:
    print("x is equal to y")
else:
    print("x is greater than y")

if x > y: print("x is greater than y")
```

Στο παράδειγμα αυτό, μπορείτε να δείτε ότι η συνθήκη με τον όρο **if** δεν είναι αληθής, το ίδιο και η δεύτερη συνθήκη ενώ η τελευταία είναι αληθής και επομένως εκτυπώνεται η αντίστοιχη δήλωση στην κονσόλα.

Επίσης, εάν έχετε μόνο μια σύντομη δήλωση για εκτέλεση, μπορείτε να την γράψετε μόνο σε μία γραμμή:

```
if x > y: print("x is greater than y")
```

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Μπορείτε επίσης να κάνετε το ίδιο για τον συνδυασμό δύο συσχετιζόμενων εντολών **if** και **else**

```
print("x is greater than why") if x > y else print("y is greater than x")
```

Μια άλλη επιλογή είναι να συμπεριλάβετε πολλαπλές δηλώσεις με τον όρο **else** σε μία γραμμή :

```
print("X") if x > y else print("=") if x == y else print("Y")
```

3.4.2. Λογικοί τελεστές

Η Python περιλαμβάνει 3 λογικούς τελεστές, όπως είχαμε αναφέρει στην αρχή της υποενότητας, το Λογικό **Ή, ΚΑΙ, ΟΧΙ** (**if, and**, **not**) Οι τελεστές αυτοί χρησιμοποιούνται κατά τρόπο παρόμοιο με εκείνο στις φυσικές γλώσσες, δηλαδή για τον συνδυασμό μιας ή περισσότερων προϋποθέσεων.

Ας δούμε πώς μπορούν να χρησιμοποιηθούν στις εντολές υπό όρους στην Python:

```
x = 500
y = 55
z = 800
if x > y and z > x:
    print("Both conditions are true")
```

Ας δούμε ένα παράδειγμα του τελεστή **Ή (or)** που συνδυάζει εντολές υπό όρους:

```
x = 500
y = 55
z = 800
if x > y and z > x:
    print("One of the two conditions is true")
```

Το επόμενο παράδειγμα επικεντρώνεται στον τελεστή Λογικό **ΟΧΙ** (**not**):

```
x = 500
y = 55
if not y == x:
    print("x and y are not equal")
```

3.4.3. Δηλώσεις nested if

Οι δηλώσεις που περιέχουν μια δήλωση με τον όρο **if** μέσα σε μια άλλη δήλωση **if** αναφέρονται ως **nested if δηλώσεις**. Ας δούμε το ακόλουθο παράδειγμα για να το καταλάβουμε καλύτερα:

```
y = 50
if y > 20:
    print("y is above 20, ")
    if y > 30:
        print("and above 30 ")
    else:
        print("but not above 30")
```

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Όπως και με τις συναρτήσεις, οι δηλώσεις με τον όρο **if** δεν πρέπει να είναι άδειες. Ωστόσο, εάν για κάποιο συγκεκριμένο λόγο έχετε μια δήλωση με το όρο **if** χωρίς κάποιο περιεχόμενο, χρησιμοποιήστε την δήλωση **pass** για να αποφύγετε τυχόν σφάλματα στην Python:

```
x = 500
y = 55
if x > y:
    pass
```

Εξάσκηση: https://www.w3schools.com/python/exercise.asp?filename=exercise_ifelse1

3.4.4. Βρόχοι

Η Python διαθέτει δύο κύριες εντολές για τη δημιουργία εντολών βρόχου: το **while** και το **for**.

Η εντολή βρόχου **while** χρησιμοποιείται όταν θέλετε να επαναλαμβάνεται μια εντολή σε μια ακολουθία αντικειμένων, δηλ. να εκτελείται σε κάθε αντικείμενο μιας ακολουθίας, εφόσον η συνθήκη είναι αληθής. Για παράδειγμα, ας δημιουργήσουμε μια εντολή βρόχου **while** που εκτυπώνει το i υπό την προϋπόθεση ότι αυτό (το i) είναι μικρότερο από το 10:

```
i = 1
while i < 10:
    print(i)
    i += 1
```

Σύμφωνα με το παράδειγμα αυτό, δίνουμε εντολή στην Python να **εκτυπώνει και να αυξάνει κατά 1 το i κάθε φορά που ο βρόχος επαναλαμβάνεται και όσο το i είναι μικρότερο από το 10.**

***Σημειώστε ότι εάν δεν προσαυξάνετε το i, η εντολή βρόχου while θα συνεχίσει να εκτελείται χωρίς διακοπή και θα καταλήξετε σε μια άπειρη επανάληψη του βρόχου.**

Μπορείτε επίσης να χρησιμοποιήσετε την εντολή **break** για να υποδείξετε στην Python να διακόψει την επανάληψη των εντολών σε μια τρέχουσα πλοκάδα βρόχου έστω και αν η συνθήκη είναι αληθής:

```
i = 1
while i < 10:
    print(i)
    i += 1
```

Μια άλλη διαθέσιμη εντολή βρόχου είναι το **continue,** την οποία μπορείτε να χρησιμοποιήσετε για να υποδείξετε στην Python να παραλείψει τις υπόλοιπες εντολές στην τρέχουσα πλοκάδα βρόχου και να συνεχίσει με την επόμενη επανάληψη του βρόχου.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

```
i = 1
while i < 10:
    i += 1
    if i == 5:
        continue
    print(i)
```

Η δήλωση **else** που είδαμε στις υπό όρους δηλώσεις μπορεί επίσης να χρησιμοποιηθεί σε εντολές βρόχου:

```
i = 1
while i < 10:
    print(i)
    i += 1
else:
    print("i is no longer less than 10")
```

Εξάσκηση:

https://www.w3schools.com/python/exercise.asp?filename=exercise_while_loops1

Οι εντολές βρόχου που χρησιμοποιούν τη λέξη-κλειδί **for** επαναλαμβάνουν μια ακολουθία στοιχείων σε λίστες, πλειάδες, λεξικά, σύνολα ή συμβολοσειρές.

Ας δούμε ένα παράδειγμα:

```
nouns = ["table", "book", "chair", "house"]
for noun in nouns: # for each noun in the nouns list
    print(noun)
```

Στον συγκεκριμένο τύπο βρόχου, δεν χρειάζεται να καθορίσετε εκ των προτέρων τον δείκτη της μεταβλητής. Ένας δείκτης μιας μεταβλητής στην προκειμένη περίπτωση είναι ο ενικός αριθμός των ουσιαστικών ως η τιμή που πρέπει να ληφθεί για το καθένα (δηλαδή για κάθε αντικείμενο στη λίστα ουσιαστικών).

Ο εντολή βρόχου **for** μπορεί επίσης να επαναλαμβάνεται πάνω από μια συμβολοσειρά ως ακολουθίες χαρακτήρων:

```
for x in "book": # for each character in the word book
    print(x)
```

Η εντολή **break** που είδαμε νωρίτερα μπορεί επίσης να συνδυαστεί με μια εντολή βρόχου **for**:

```
nouns = ["table", "book", "chair", "house"]
for noun in nouns: # for each noun in the nouns list
    print(noun)
    if noun == "chair":
        break
```

Στο παράδειγμα αυτό, όταν ο βρόχος συναντά το ουσιαστικό "chair" θα σταματήσει να επαναλαμβάνεται πριν περάσει από όλα τα στοιχεία στη λίστα.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Εάν είχατε τοποθετήσει την εντολή **break** πριν από την εντολή εκτύπωσης, τι νομίζετε ότι θα συνέβαινε; Δοκιμάστε το και μόνοι σας.

```
for noun in nouns: # for each noun in the nouns list
    if noun == "chair":
        break
    print(noun)
```

Μια άλλη εντολή βρόχου που είδαμε νωρίτερα είναι το **continue**, η οποία διακόπτει την τρέχουσα επανάληψη για να προχωρήσει στην επόμενη:

```
for noun in nouns: # for each noun in the nouns list
    if noun == "chair":
        continue
    print(noun)
```

Η συνάρτηση range() είναι αρκετά χρήσιμη όταν θέλετε να καθορίσετε πόσες φορές θέλετε να συμβεί μια επανάληψη. Το προεπιλεγμένο σημείο εκκίνησης της συνάρτησης range είναι το 0, το οποίο αυξάνεται κατά 1 σε κάθε επανάληψη και τελειώνει σε έναν προκαθορισμένο αριθμό:

```
for x in range(10):
    print(x)
```

*** Σημειώστε ότι επειδή το range ξεκινά από το 0, θα σταματήσει στον αριθμό 9. Αν θέλουμε να φτάσει στο 10, τότε θα πρέπει να χρησιμοποιήσουμε το range(11).**

Έχετε την επιλογή να καθορίσετε το αρχικό και το τελικό range:

```
for x in range(2, 10):
    print(x)
```

Εδώ, το range ξεκινά από το 2 και τελειώνει στο 9, αφού το 10 δεν συμπεριλαμβάνεται σε αυτό.

Μια άλλη επιλογή που έχετε είναι να ρυθμίσετε το κατά πόσο θέλετε να αυξάνεται σε κάθε επανάληψη:

```
for x in range(2, 40, 2):
    print(x)
```

Στο παράδειγμα αυτό, ορίσαμε το range να ξεκινά από το 2 μέχρι το 40 και να αυξάνεται κατά 2 κάθε φορά. Ως εκ τούτου, ο αριθμός θα σταματήσει στο 38, δεδομένου ότι δεν μπορεί να αυξηθεί κατά 2 από το 38 και μετά.

Στις εντολές βρόχου, μπορείτε επίσης να χρησιμοποιήσετε την λέξη-κλειδί *else* για να καθορίσετε τί θέλετε να συμβαίνει όταν ο βρόχος τερματιστεί.

```
for x in range(10):
    print(x)
else:
    print("Done!")
```

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Ας προσπαθήσουμε να χρησιμοποιήσουμε την δήλωση με τους όρους **if** και *else* σε μια εντολή βρόχου **for**:

```
for x in range(10):
    if x == 8: break
    print(x)
else:
    print("Done!")
```

Πιστεύετε ότι η δήλωση else θα εκτελεστεί; Ναι ή όχι και γιατί; Δοκιμάστε το για να μάθετε!

Τώρα, ας θυμηθούμε τις δηλώσεις **nested if else**, τις οποίες μπορείτε να χρησιμοποιήσετε για nested βρόχους:

```
words = ["I", "am", "you", "are", "working"]
nouns = ["book", "house", "person", "love"]
for word in words:              # outer loop
    for noun in nouns:             # inner loop
        print(word, noun)
```

Όπως μπορείτε να δείτε, ο εσωτερικός βρόχος θα εκτελεστεί μία φορά κατά τη διάρκεια κάθε επανάληψης του εξωτερικού βρόχου.

Κατά τρόπο παρόμοιο με τις συναρτήσεις και τις δηλώσεις με τον όρο **if**, έτσι λοιπόν και οι δηλώσεις με τον όρο **for** δεν πρέπει να είναι άδειες. Ωστόσο, εάν για κάποιο συγκεκριμένο λόγο έχετε ένα βρόχο **for** χωρίς περιεχόμενο, μπορείτε να χρησιμοποιήσετε τη δήλωση **pass** για να αποφύγετε τυχόν σφάλματα από την Python.

```
for word in words:
    pass
```

Εξάσκηση:
https://www.w3schools.com/python/exercise.asp?filename=exercise_while_loops1

3.5. Κατανόηση της ροής της εκτέλεσης μέσω συναρτήσεων

Μέχρι στιγμής, έχουμε δει έναν αριθμό ενσωματωμένων συναρτήσεων που χρησιμοποιούνται στην Python όπως το print(), type(), dict(), len(). Κάθε συνάρτηση έχει ένα συγκεκριμένο σκοπό και χρειάζεται να καθορίσουμε ποια μεταβλητή θέλουμε να εκτελέσει.

Παράδειγμα:

```
x = 5
print(x)
```

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Στη συνάρτηση print() θα πρέπει να καθορίσουμε την μεταβλητή x αν θέλουμε να την εκτυπώσουμε. Η μεταβλητή ή οι μεταβλητές που περιέχονται στην παρένθεση ονομάζονται **ορίσματα (arguments) ή παράμετροι (parameters)**.

Αυτοί οι δύο όροι χρησιμοποιούνται συνήθως εναλλακτικά, αλλά οι **παράμετροι** είναι οι μεταβλητές που ορίζονται μέσα στην παρένθεση μιας συνάρτησης, π.χ. print(x), ενώ τα **ορίσματα** θεωρούνται οι τιμές που αποστέλλονται στη συνάρτηση όταν την καλούμε.

Παρόλο που οι ενσωματωμένες συναρτήσεις είναι αρκετά χρήσιμες, δεν μπορούν επιλύσουν όλα τα προβλήματα που τυχόν να παρουσιαστούν στην Python. Ωστόσο, έχουμε την επιλογή να δημιουργήσουμε νέες συναρτήσεις για την επίλυση συγκεκριμένων προβλημάτων, που αποτελεί μια από τις πιο σημαντικές δυνατότητες που παρέχει η Python.

Μια συνάρτηση είναι μια ονομαζόμενη ακολουθία δηλώσεων η οποία πραγματοποιεί ένα συγκεκριμένο υπολογισμό. Η σύνταξη που χρησιμοποιείται για τη δημιουργία μιας συνάρτησης είναι η ακόλουθη:

```
>> def όνομα(ορίσματα):
        δηλώσεις
```

Οι συναρτήσεις ορίζονται χρησιμοποιώντας τη λέξη-κλειδί **def**, που ουσιαστικά ορίζει το όνομα της συνάρτησης μαζί με το/α όρισμα/τα όταν αυτά καλούνται, με σκοπό την επίτευξη ενός επιθυμητού αποτελέσματος. Οι ενσωματωμένες συναρτήσεις έχουν ήδη οριστεί και έτσι, δεν μπορούμε να δούμε τις δηλώσεις που περιέχουν, αλλά μόνο το παραγόμενο αποτέλεσμα ή την έξοδό τους.

Όταν δημιουργείτε τη δική σας συνάρτηση, είστε ελεύθεροι να χρησιμοποιήσετε οποιαδήποτε ονόματα θέλετε εκτός από τις λέξεις-κλειδιά της Python που αναφέραμε σε προηγούμενες υποενότητες (Υποενότητα 3.2.). Τα ορίσματα που χρησιμοποιούνται στην παρένθεση μιας συνάρτησης παρέχουν τις πληροφορίες που απαιτούνται για να λειτουργήσει μια συνάρτηση.

Ας δούμε ένα παράδειγμα για να κατανοήσουμε τη διαδικασία που ακολουθείται από μια συνάρτηση:

```
def my_function():
    print("Hello, World!")
```

Εδώ δημιουργήσαμε μια συνάρτηση χωρίς κάποια απαραίτητα ορίσματα, η οποία θα εκτυπώσει το κείμενο που καθορίζεται στη δεύτερη γραμμή του κώδικα.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Θυμάστε όταν μιλήσαμε για την εσοχή κώδικα στην προηγούμενη υποενότητα; Η ίδια λογική ισχύει και εδώ, που στην προκειμένη περίπτωση υποδεικνύει ότι ο κώδικας θα εκτελεστεί σε ένα τμήμα μιας συνάρτησης.

Επομένως, θα γράψουμε τη δήλωση print("Hello, World!") εκτός της συνάρτησης, όπως κάναμε και σε προηγούμενα παραδείγματα, θεωρώντας την ως μια **καθολική** δήλωση. Ωστόσο, τώρα που η δήλωση είναι γραμμένη εντός της συνάρτησης, τότε αυτή θεωρείται τοπική.

Για να καλέσουμε μια συνάρτηση, απλά χρησιμοποιούμε το όνομά της ακολουθούμενο από μια παρένθεση:

```
my_function()
```

Η έξοδος αυτής της συνάρτησης θα είναι:
Hello, World!

Ας δούμε ένα άλλο παράδειγμα στο οποίο έχουμε ένα άλλο όρισμα στην παρένθεση μιας συνάρτησης:

```
def my_function(country):
    print("I am from " + country)
```

Όταν καλούμε τη συνάρτηση, τότε το όνομα της χώρας περνιέται στην κλήση της συνάρτησης και κατόπιν εκτυπώνεται, ακολουθώντας δηλαδή τις εντολές που δώσαμε. Το όρισμα "country" είναι η τιμή η οποία περνιέται από το πρόγραμμα στην κλήση της συνάρτησης:

```
my_function("France")
my_function("Greece")
my_function("Germany")
```

Έξοδος:
Είμαι από τη Γαλλία.
Είμαι από την Ελλάδα
Είμαι από τη Γερμανία.

Μπορείτε επίσης να καθορίσετε τον τύπο δεδομένων του ορίσματος χρησιμοποιώντας την ακόλουθη σύνταξη: argument:datatype

Παράδειγμα:

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

```
def my_function(country:str):
    print("I am from " + country)
```

Όταν καλείται αυτή η συνάρτηση, η Python αναμένει ότι το όρισμα "country" θα είναι υπό τη μορφή μιας συμβολοσειράς. Διαφορετικά, θα προκύψει σφάλμα.

```
my_function("France")
my_function(France) #will raise error
```

Επίσης, μπορείτε να έχετε περισσότερα από 1 ορίσματα. Ας πούμε ότι θέλουμε να προσδιορίσουμε την πόλη και τη χώρα:

```
def my_function(city, country):
    print("I am from " + city + ", "+ country)
```

Σημειώστε ότι συμπεριλάβαμε εισαγωγικά με κόμμα για να διαχωρίσουμε την πόλη και τη χώρα μεταξύ τους, ώστε να μην εκτυπωθούν χωρίς κενό διάστημα μεταξύ τους. Δεδομένου ότι έχουμε καθορίσει τα 2 ορίσματα, θα πρέπει να καλέσουμε τον ίδιο ακριβώς αριθμό ορισμάτων που καθορίσαμε στην παρένθεση, ούτε περισσότερο ούτε λιγότερο, εάν θέλουμε να λειτουργήσει η συνάρτηση. Διαφορετικά, θα προκύψει σφάλμα.

```
my_function("Paris", "France")
```

Έξοδος:
Είμαι από το Παρίσι, Γαλλία

```
my_function("Paris") # this will raise an error in Python
```

Μια άλλη επιλογή που έχετε αν δεν γνωρίζετε τον αριθμό των ορισμάτων που πρέπει να συμπεριληφθούν σε μια συνάρτηση, είναι να προσθέσετε απλά έναν αστερίσκο (*) πριν από το όνομα του ορίσματος στον ορισμό της συνάρτησης.

```
def my_function(*countries):
    for country in countries:
        print("I have visited ", country)
```

Με αυτόν τον τρόπο, η συνάρτηση θα λάβει μια πλειάδα ορισμάτων και τα στοιχεία θα μπορέσουν να προσπελαστούν αναλόγως.

*** Αυτοί οι τύποι άγνωστου αριθμού ορισμάτων ονομάζονται arbitrary arguments και συχνά αναφέρονται ως *args στην τεκμηρίωση της Python.**

Εδώ προσδιορίσαμε 3 χώρες που θα εκτυπωθούν σε ξεχωριστές προτάσεις και σε μία πρόταση:

```
my_function("Ireland", "France", "Belgium")
my_function("Ireland, France, Belgium")
```

Έξοδος:
Έχω επισκεφθεί την Ιρλανδία

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Έχω επισκεφθεί τη Γαλλία
Έχω επισκεφτεί το Βέλγιο
Έχω επισκεφθεί την Ιρλανδία, τη Γαλλία, το Βέλγιο

Μια άλλη επιλογή που έχετε στη διάθεσή σας είναι να στείλετε ορίσματα ως ζεύγη κλειδιού = τιμής. Στην επιλογή αυτή, η σειρά δεν έχει σημασία.

Παράδειγμα:

```
def my_function(first, last):
    print("My name is " + first + last )

my_function(first = "Jane ", last = "Kent")
```

Μία ακόμη διαθέσιμη επιλογή είναι η χρήση **δύο αστερίσκων (**)** πριν από το όνομα του ορίσματος στον ορισμό της συνάρτησης, εάν δεν γνωρίζετε πόσα ορίσματα με λέξεις-κλειδιά θα πρέπει να περαστούν στην κλήση της συνάρτησης. Με αυτόν τον τρόπο, η συνάρτηση θα λάβει τα ορίσματα σε μορφή λεξικού και θα έχει πρόσβαση σε αυτά αναλόγως:

```
def total_words(**words):
    print(words, type(words))

total_words(paper = 6, I = 10, nice = 9)
```

Όπως μπορείτε να δείτε, μπορείτε να προσθέσετε νέα ορίσματα όταν καλείτε τη συνάρτηση. Σημειώστε ότι αυτοί οι τύποι ορισμάτων ονομάζονται **arbitrary arguments λέξεων-κλειδιών** και συχνά αναφέρονται ως ****kwargs** στην τεκμηρίωση της Python.

Επιπρόσθετα, μία ακόμη επιλογή που έχετε είναι να ορίσετε μια προεπιλεγμένη τιμή παραμέτρου. Αυτό σημαίνει ότι όταν καλούμε μια συνάρτηση χωρίς όρισμα, τότε αυτή θα χρησιμοποιήσει την προεπιλεγμένη τιμή που θέσαμε:

```
def my_function(country= "United Kingdom"):
            print("I am from" + country)

my_function("France")
my_function("Malta")
my_function()
```

Ένα όρισμα μπορεί να επιτρέψει οποιονδήποτε τύπο δεδομένων ως όρισμα (π.χ. συμβολοσειρά, λίστα, λεξικό κ.λπ.).

Οι συναρτήσεις μπορούν επίσης να επιστρέψουν τιμές χρησιμοποιώντας τη εντολή **return**:

```
def calc(x):
    return 3 * x

print(calc(5))
print(calc(8))
print(calc(9))
```

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Δεδομένου ότι δεν έχουμε ορίσει τη συνάρτηση εκτύπωσης στη συνάρτηση που δημιουργήσαμε, πρέπει ωστόσο να τη χρησιμοποιήσουμε όταν καλούμε τη συνάρτηση για να έχουμε ένα αποτέλεσμα.

Γενικά, οι συναρτήσεις δεν πρέπει να είναι άδειες. Ωστόσο, εάν για κάποιο συγκεκριμένο λόγο έχετε έναν ορισμό συνάρτησης χωρίς περιεχόμενο, θα πρέπει να χρησιμοποιήσετε τη δήλωση pass για να αποφύγετε τη λήψη τυχόν σφαλμάτων από την Python:

```
def my_function():
    pass
```

Εξάσκηση: https://www.w3schools.com/python/exercise.asp?filename=exercise_ifelse1

3.6. Συναρμολογώντας το παζλ - Πώς να δημιουργήσετε ένα πρόγραμμα

Μέχρι στιγμής, έχουμε μάθει πολλές διαφορετικές δηλώσεις, λέξεις-κλειδιά, μεθόδους και συναρτήσεις οι οποίες χρησιμοποιούνται στην Python. Τώρα μπορεί να αναρωτιέστε πώς μπορούμε πραγματικά να δημιουργήσουμε ένα απλό πρόγραμμα. Στην υποενότητα αυτή, θα προσπαθήσουμε να εξηγήσουμε βήμα προς βήμα τη διαδικασία δημιουργίας ενός απλού προγράμματος, ώστε να είστε σε θέση να κατανοείτε πώς να εφαρμόζετε στην πράξη ό,τι έχετε μάθει.

Θέλουμε να δημιουργήσουμε ένα πρόγραμμα που:

1. διαβάζει ένα αρχείο κειμένου,
2. δημιουργεί ένα λεξικό που περιέχει όλες τις λέξεις του αρχείου κειμένου και τη συχνότητά με την οποία εμφανίζονται,
3. επιτρέπει στους χρήστες να γράφουν μια λέξη και να τους εμφανίζεται η συχνότητά της,
4. που τους ενημερώνει ότι δεν υπάρχει μια συγκεκριμένη λέξη μέσω εμφάνισης ενός μηνύματος

Αυτό μπορεί να σας φαίνεται ακατανίκητο στην αρχή, αλλά μην ανησυχείτε! Θα εξηγήσουμε βήμα προς βήμα τη διαδικασία.

Εδώ, θα μάθουμε πώς να δημιουργούμε αναδρομικές συναρτήσεις (recursive functions), δηλαδή συναρτήσεις που θα καλούν η μια την άλλη για την εκτέλεση του προγράμματός μας.

Πρώτα, θα πρέπει να καλέσουμε ένα άρθρωμα (module), το οποίο ονομάζεται string. Το ενσωματωμένο άρθρωμα (built-in module) string περιέχει διάφορες συναρτήσεις που σας επιτρέπουν να επεξεργαστείτε συμβολοσειρές στην Python, και τις οποίες θα χρησιμοποιήσουμε αργότερα για να αφαιρέσουμε όλα τα σημεία στίξης.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Υπάρχουν πολλά διαθέσιμα αρθρώματα στην Python που εξυπηρετούν διάφορους σκοπούς και μπορείτε να τα καλέσετε μέσω της εντολής import χρησιμοποιώντας λέξεις-κλειδιά:

```
import string
```

Η πρώτη συνάρτηση θα διαβάσει το αρχείο κειμένου και θα δημιουργήσει ένα λεξικό:

```
def process_text(filename):
    dictionary = dict()
    fin = open(filename, 'r')
    for line in fin:
        process_line(line, dictionary)
    return dictionary
```

Προσπαθούμε πάντα να δίνουμε ονόματα στις συναρτήσεις που μπορούμε να καταλάβουμε εύκολα, ενώ θα πρέπει να καθορίσουμε το όνομα του αρχείου (filename) του ορίσματος όταν θα το καλέσουμε. Η επόμενη γραμμή δημιουργεί ένα άδειο λεξικό. Η **τοπική μεταβλητή fin** καλεί την συνάρτηση **open** για να ανοίξει και να διαβάσει το αρχείο.Η **εντολή βρόχου for** στην επόμενη γραμμή καλεί την επόμενη συνάρτηση η οποία θα επεξεργαστεί κάθε γραμμή, θα δημιουργήσει το λεξικό μας και θα το επιστρέψει.

Μπορεί να ακούγεται λίγο περίπλοκο, αλλά στην πραγματικότητα είναι αρκετά απλό. Δημιουργήσαμε ουσιαστικά μια συνάρτηση που θα δημιουργήσει ένα άδειο λεξικό, που θα διαβάσει το όνομα αρχείου (filename) και θα επεξεργαστεί κάθε γραμμή του αρχείου για να δημιουργήσει ένα λεξικό.

Τώρα, γιατί χρειάζεται να επεξεργαστούμε κάθε γραμμή;

Επειδή η Python διαθέτει διάκριση πεζών-κεφαλαίων, καθώς και σημείων στίξης δίπλα στις λέξεις, ακόμα και αν η λέξη είναι η ίδια όπως "Love" και "love" ή "love" και "love".

Δεδομένου ότι θέλουμε να μετρήσουμε τη συχνότητα με την οποία εμφανίζεται κάθε λέξη στο αρχείο, πρέπει να βεβαιωθούμε ότι όλες οι λέξεις είναι γραμμένες με πεζά γράμματα και ότι όλα τα σημεία στίξης (δηλ. κόμματα, τελείες, θαυμαστικά κ.λπ.) αφαιρέθηκαν, ούτως ώστε το πρόγραμμα να κάνει έναν σωστό υπολογισμό.

Ας γράψουμε μια συνάρτηση που επεξεργάζεται κάθε γραμμή του αρχείου μας:

```
def process_line(line,dictionary):
    line = line.replace('-', '')

    for word in line.split():
        word = word.strip(string.punctuation + string.whitespace)
        word = word.lower()
        dictionary[word] = dictionary.get(word,0) +1
```

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Και πάλι, εδώ χρησιμοποιούμε την εντολή βρόχου **for** η οποία θα επαναλαμβάνεται σε κάθε γραμμή του κειμένου.

⇒ Πρώτα, θα πρέπει να αντικαταστήσουμε τις παύλες με κενά διαστήματα επειδή δεν μπορούν να αφαιρεθούν με τη **συνάρτηση** string.punctuation.

⇒ Μόλις αντικαταστήσουμε τις παύλες, ξεκινάμε να χωρίζουμε τη γραμμή σε ξεχωριστές λέξεις και να αφαιρούμε από κάθε λέξη τα σημεία στίξης και τα κενά διαστήματα, και στη συνέχεια μετατρέπουμε τη λέξη σε πεζά γράμματα.

⇒ Αφού έχουμε ολοκληρώσει την επεξεργασία που έχουμε μόλις περιγράψει, τότε η λέξη προστίθεται στο λεξικό και **εάν η λέξη υπάρχει, τότε προσθέτετε 1, εάν όχι τότε προσθέτετε το 0.**

Μέχρι στιγμής, έχουμε δημιουργήσει δύο συναρτήσεις που διαβάζουν, επεξεργάζονται το κείμενο και δημιουργούν ένα λεξικό με τις συχνότητες εμφάνισης των λέξεων.

Στο σημείο αυτό, μπορείτε να αναθέσετε στη μεταβλητή dictionary τη συνάρτηση που είναι υπεύθυνη για την επεξεργασία του κειμένου, όπου θα πρέπει να καθορίσετε το όνομα του αρχείου κειμένου (δηλ. 'the_veldt.txt'). Όλες οι συναρτήσεις συνδέονται με βάση τη μεταβλητή dictionary η οποία θα περιέχει όλες τις λέξεις που βρίσκονται στο συγκεκριμένο κείμενο μαζί με τις συχνότητες εμφάνισής τους.

```
dictionary = process_text('the_veldt.txt')
```

Μέχρι τώρα ολοκληρώσαμε τις 2 από τις 4 συνολικά ενέργειες που θέλουμε να εκτελέσει το πρόγραμμα, ενώ μας απομένουν ακόμη 2 ενέργειες.

Τώρα, θέλουμε οι χρήστες να είναι σε θέση να γράψουν μια λέξη και αν η λέξη υπάρχει στο κείμενο, τότε να εμφανίζεται η συχνότητά της, διαφορετικά να ενημερώνει το χρήστη ότι η λέξη δεν υπάρχει στο λεξικό.

Ας γράψουμε μια συνάρτηση που θα συγκρίνει τις λέξεις που εισάγονται από εμάς τους χρήστης με εκείνες που βρίσκονται στο λεξικό που δημιουργήσαμε:

```
def findwords(dictionary):
    for key,value in dictionary.items():
        if key == find_word:
            return(value)
    return("This word was not found in the text, please look for another word ")
```

Αυτή η συνάρτηση χρησιμοποιείται για να περάσει το κλειδί και την τιμή κάθε στοιχείου στο λεξικό μέσω μιας εντολής βρόχου for. Αν το κλειδί (λέξη) είναι το ίδιο με τη λέξη που εισήγαγε ο χρήστης, τότε επιστρέφει την τιμή (δηλαδή, τη συχνότητα της συγκεκριμένης

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

λέξης). Διαφορετικά, επιστρέφει μια δήλωση ότι η λέξη δεν βρέθηκε και προτρέπει τον χρήστη να αναζητήσει μια άλλη.
Η τέταρτη ενέργεια που θέλουμε να εκτελείται αφορά την επανάληψη του προγράμματος,

```
while True:
    find_word = input("Please enter word to find its frequency, or type 'q' to quit: ").lower()
    if find_word != 'q':
        print(findwords(dictionary))
        continue
    else:
        break
```

μέχρι ο χρήστης να ορίσει πότε αυτή θα τερματιστεί. Για την εκτέλεση αυτής της ενέργειας, θα πρέπει να χρησιμοποιήσουμε μια εντολή βρόχου **while** που δεν βρίσκεται μέσα σε συνάρτηση.

Απλά χρησιμοποιούμε μια εντολή βρόχου **while με μια υπό όρους δήλωση True (αληθές)** εάν θέλουμε το πρόγραμμα να συνεχίσει τη λειτουργία του. Να είστε προσεκτικοί όταν χρησιμοποιείτε μια εντολή βρόχου με μια υπό όρους δήλωση **True** γιατί μπορεί να καταλήξετε σε μια άπειρη επανάληψη του βρόχου. Εάν η υπό όρους δήλωση είναι αληθής, τότε θα συνεχίσει να εκτελείται το πρόγραμμα.

Αναθέτουμε μια μεταβλητή στην εισαγωγή στοιχείων από τον χρήστη, προκειμένου να μπορούμε να τα συγκρίνουμε με τα κλειδιά του λεξικού (λέξεις) και όποια λέξη εισάγεται από τον χρήστη να μετατρέπεται σε πεζά γράμματα όταν το πρόγραμμα τη διαβάζει. Ο λόγος γι' αυτό είναι ότι ο χρήστης μπορεί να γράψει μια λέξη που είναι όλη γραμμένη σε κεφαλαία γράμματα, με αποτέλεσμα το πρόγραμμα να μην επιφέρει κάποιο αποτέλεσμα. Αυτό συμβαίνει γιατί η Python, όπως έχουμε αναφέρει και προηγουμένως, δε διαθέτει διάκριση πεζών-κεφαλαίων με αποτέλεσμα να μην μπορεί να αναγνωρίσει ότι δύο λέξεις είναι οι ίδιες αν η μια είναι γραμμένη σε κεφαλαία και η άλλη σε πεζά γράμματα.

Η επόμενη γραμμή δίνει εντολή ότι, όταν η λέξη που εισάγεται από τον χρήστη δεν είναι ίση με το γράμμα ‘q’, τότε να εκτυπώνονται τα αποτελέσματα της συνάρτησης που ανακτούν τη συχνότητα της λέξης και μετά να επαναλαμβάνουν το ίδιο με τις επόμενες αναζητήσεις στο κείμενο. Διαφορετικά, εάν η λέξη που εισάγεται από τον χρήστη είναι ίση με το γράμμα ‘**q**’, τότε το πρόγραμμα να τερματίζει τη λειτουργία του (δηλαδή, να δίνεται η εντολή break του βρόχου).

Υπάρχουν πολλοί τρόποι με τους οποίους θα μπορούσε κανείς να γράψει αυτό το μικρό πρόγραμμα. Στην υποενότητα αυτή, έχουμε προσπαθήσει να εφαρμόσει πολλές από τις γνώσεις που αποκτήσαμε μέχρι τώρα για τον προγραμματισμό, μαθαίνοντας επίσης πώς να εισάγουμε αναδρομικές συναρτήσεις (recursive functions) και πώς να κατακερματίζουμε ένα πρόγραμμα σε μικρότερα τμήματα κώδικα για την καλύτερη διαχείρισή του.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

*** Σημειώστε ότι όταν γράφετε έναν κώδικα, είναι καλύτερα να εκτελείτε κάθε μικρό του τμήμα ξεχωριστά, για να είστε σε θέση να εντοπίζετε πιθανά σφάλματα πριν συνεχίσετε με το επόμενό του τμήμα.**

Αφού έχετε ολοκληρώσει το πρόγραμμά σας, μπορεί να χρειαστεί να επιστρέψετε σε κάποια σημεία που χρειάζονται περεταίρω επεξεργασία ή να αφαιρέσετε ορισμένα αρχικά τμήματα του κώδικα που χρησιμοποιήσατε για δοκιμή ή ακόμη και να συνενώσετε πολλαπλές δηλώσεις, ούτως ώστε το πρόγραμμα σας να είναι πιο ευανάγνωστο.

3.7. Πώς να προσαρμόσετε ένα πρόγραμμα ώστε να ανταποκρίνεται στις ανάγκες σας

Έχουμε δει πώς μπορείτε να δημιουργήσετε ένα μικρό πρόγραμμα στην Python, ωστόσο, μερικές φορές μπορεί να συναντήσετε προγράμματα που έχουν ήδη γραφτεί και μπορεί να χρειαστεί να κάνετε προσαρμογές στον κώδικα για να βελτιώσετε ή να αλλάξετε το τελικό αποτέλεσμα.

Ανάλογα με το ποιος έγραψε τον κώδικα και το πόσο μεγάλος είναι, μπορεί η διαδικασία της προσαρμογής να σας φανεί δύσκολη και αφόρητη στην αρχή.

Μπορείτε να ακολουθήσετε την ίδια λογική που εφαρμόσαμε για την δημιουργία του προηγούμενου προγράμματος, όπου για να ξεκαθαρίσει το τοπίο, δουλέψαμε αποσπασματικά εξετάζοντας τμηματικά τον κώδικα, ούτως ώστε να κατανοήσουμε καλύτερα σε τι εξυπηρετεί η κάθε συνάρτηση.

Επίσης, είναι σημαντικό να γράφετε σχόλια όταν εργάζεστε πάνω σε έναν κώδικα, καθώς τα σχόλια μπορεί να φανούν ιδιαίτερα χρήσιμα για τους άλλους που διαβάζουν τον κώδικά σας αλλά και για εσάς επίσης.

Τι μπορείτε λοιπόν να κάνετε σε περίπτωση που δεν υπάρχουν σχόλια σε ένα πρόγραμμα;

Πρώτα απ’ όλα, πρέπει να «τρέξετε» το πρόγραμμα για να δείτε τα αποτελέσματα που θα επιφέρει και να παρατηρήσετε τις βιβλιοθήκες και τα αρθρώματα που εισάγονται. Στη συνέχεια, θα πρέπει να αναζητήσετε το σημείο εκκίνησης του κώδικα και να προσπαθήσετε να κατανοήσετε τη ροή της εκτέλεσης. Μια άλλη χρήσιμη συμβουλή θα ήταν να εξετάσουμε τις μεταβλητές που έχουν περάσει στο πρόγραμμα και να μάθουμε σε τι εξυπηρετεί η καθεμία στο πρόγραμμα. Αφού έχετε κατανοήσει επαρκώς πώς λειτουργεί το πρόγραμμα, μπορείτε να ξεκινήσετε με την επεξεργασία.

*** Να είστε προσεκτικοί όταν προσπαθείτε να επεξεργαστείτε τη συνάρτηση κάποιου άλλου, επειδή μπορεί να την διαλύσετε**

Μια καλή πρακτική κατά την προσαρμογή ενός προγράμματος δεν είναι η χρήση ενός συγκεκριμένου αριθμού μεταβλητών αλλά η χρήση των **παραμέτρων *args** και ****kwargs,** για να κάνετε τον κώδικά σας πιο ευέλικτο και ευπροσάρμοστο όταν κάποιος άλλος

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

χρήστης πραγματοποιεί προσαρμογή. Έχουμε ήδη εξετάσει τη χρήση των δύο προαναφερθέντων μεταβλητών νωρίτερα στην υποενότητα.

Υπενθυμίζουμε ότι οι παράμετροι ***args** σας επιτρέπουν να περάσετε έναν ποικίλο αριθμό ορισμάτων θέσης ενώ οι παράμετροι ****kwargs** σας επιτρέπουν να περάσετε ένα ποικίλο αριθμό λέξεων-κλειδιών. Το σημαντικό και στις δύο παραμέτρους είναι η χρήση των αστερίσκων (* ή * *), ενώ μπορείτε να δώσετε ό,τι όνομα θέλετε μετά από αυτά.

Για περισσότερες συμβουλές και παραδείγματα, συμβουλευτείτε τις ακόλουθες ιστοσελίδες:

- freeCodeCamp - https://www.freecodecamp.org/news/args-and-kwargs-in-python/
- DZone - https://dzone.com/articles/adding-functionality-to-legacy-code
- Codeacademy - https://www.codecademy.com/resources/blog/how-to-work-with-code-written-by-someone-else/

3.8. Συντακτικά λάθη, Σφάλματα Χρόνου Εκτέλεσης και Σημασιολογικά λάθη – Διαχείριση σφαλμάτων στην Python

Υπάρχουν τρεις διαφορετικοί σημαντικοί τύποι σφαλμάτων που μπορούν να εμφανιστούν όταν γράφετε ένα πρόγραμμα και είναι χρήσιμο να ξέρετε τις διαφορές μεταξύ τους και πώς να τις εντοπίζετε έγκαιρα:

1. **Τα συντακτικά λάθη** προκύπτουν όταν η Python μεταφράζει τον πηγαίο κώδικα σε δυαδική μορφή και υποδεικνύει ότι κάποιο μέρος της σύνταξης είναι λάθος, π.χ. η εσοχή κώδικα, η παράλειψη της άνω και κάτω τελείας στο τέλος μιας δήλωσης def ή η παράλειψη μιας παρένθεσης. Όπως και στις φυσικές γλώσσες, όταν χρησιμοποιούμε λανθασμένη σύνταξη στην Python, λαμβάνουμε ένα σφάλμα που λέει: μη έγκυρη σύνταξη.

   Τα συντακτικά λάθη μπορούν εύκολα να εντοπιστούν. Πιο κάτω παρατίθενται μερικά:

   - Η χρήση των λέξεων-κλειδιών στην Python για ένα όνομα μεταβλητής
   - Απουσία μιας άνω και κάτω τελείας (:) στο τέλος των εντολών βρόχου **for, while** και στις δηλώσεις **def**
   - Εσοχή κώδικα εντός των βρόχων και των υπό όρους δηλώσεων
   - Η χρήση διπλών και απλών εισαγωγικών στις συμβολοσειρές
   - Βεβαιωθείτε ότι δεν έχετε παραλείψει κανένα εισαγωγικό κατά την εγγραφή συμβολοσειρών
   - Η χρήση του απλού ή διπλού συμβόλου ίσων (=/ ==) σε μια υπό όρους δήλωση
   - Η παράλειψη της ολοκλήρωσης των παρενθέσεων σε οποιαδήποτε γραμμή του κώδικά σας, δηλαδή,),], }

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Σε περίπτωση που δεν μπορείτε να εντοπίσετε ένα συντακτικό λάθος, μπορείτε να δημιουργήσετε ένα νέο αρχείο και να προσθέσετε τον κώδικά σας γραμμή προς γραμμή από την αρχή.

2. **Τα σφάλματα χρόνου εκτέλεσης** συμβαίνουν όταν ισχύουν τα ακόλουθα: 1) το πρόγραμμα δεν εκτελεί κάποια λειτουργία, 2) το πρόγραμμα εισέρχεται σε έναν άπειρο βρόχο ή μια αναδρομική συνάρτηση (recursive function), 3) όταν λαμβάνετε ένα σφάλμα/εξαίρεσης (exception) ή 4) όταν έχετε προσθέσει πάρα πολλές εντολές εκτύπωσης.

   Ας δούμε μερικούς τρόπους με τους οποίους μπορείτε να ξεπεράσετε ή να εντοπίσετε την αιτία των προαναφερθέντων σφαλμάτων στην Python:

   - Όταν το πρόγραμμα δεν εκτελεί κάποια λειτουργία, βεβαιωθείτε ότι το έχετε καλέσει για εκτέλεση στη διαδραστική κονσόλα
   - Εάν εισάγετε έναν άπειρο βρόχο, σταματήστε το πρόγραμμα και προσθέστε εντολές *εκτύπωσης* εκεί που νομίζετε ότι μπορεί να υφίσταται το πρόβλημα και προσπαθήστε να χρησιμοποιήσετε hashtags (#) μπροστά από τα μέρη του κώδικά σας για να δείτε τι θα συμβεί
   - Τα σφάλματα εξαίρεσης εμπίπτουν σε 5 κύριες κατηγορίες:
     i. **Το NameError** εμφανίζεται όταν προσπαθείτε να χρησιμοποιήσετε μια μεταβλητή που δεν υπάρχει, δηλαδή τοπικές μεταβλητές
     ii. Το **TypeError** μπορεί να εμφανιστεί για πολλούς λόγους, όπως από την λανθασμένη χρήση μιας τιμής, την αναντιστοιχία μεταξύ αντικειμένων σε μορφή συμβολοσειράς και από τη λανθασμένη χρήση μιας τιμής, την αναντιστοιχία μεταξύ στοιχείων σε μορφήσυμβολοσειράς και αντικειμένων που έχουν τροποποιηθεί ή από έναν λανθασμένο αριθμό ορισμάτων.
     iii. Το **KeyError** εμφανίζεται όταν προσπαθείτε να αποκτήσετε πρόσβαση σε ένα αντικείμενο ή σε ένα λεξικό χρησιμοποιώντας ένα κλειδί που δεν υπάρχει.
     iv. Το **AttributeError** μπορεί να εμφανιστεί όταν προσπαθείτε να αποκτήσετε πρόσβαση σε μια ιδιότητα που δεν υπάρχει.
     v. Το **IndexError** εμφανίζεται όταν υπάρχει αναντιστοιχία μεταξύ του αριθμού ευρετηρίου μιας λίστας, συμβολοσειράς ή πλειάδας και του μήκους της.

3. Τα **σημασιολογικά λάθη** είναι συνήθως εκείνα τα που πιο δύσκολα μπορείτε να εντοπίσετε, δεδομένου ότι πρόγραμμα «τρέχει» αλλά δεν επιφέρει τα επιθυμητό αποτέλεσμα. Μπορεί να σκεφτήκατε ότι η σειρά με την οποία τοποθετήσατε τις δηλώσεις σας έχει κάποιο νόημα, ωστόσο, η Python μπορεί να συμπεριφερθεί με μη αναμενόμενους τρόπους.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Όταν συναντάτε σημασιολογικά σφάλματα, θα ήταν χρήσιμο να πραγματοποιήσετε τα εξής:

- Κατακερματίστε τον κώδικα σε μικρότερα μέρη για να καταλάβετε πώς συμπεριφέρεται το πρόγραμμα σε κάθε βήμα
- Εξετάστε τις ιδέες ή τις σημειώσεις σας σχετικά με το σκεπτικό που ακολουθήσατε πίσω από κάθε γραμμή κώδικα
- Χρησιμοποιήστε εντολές εκτύπωσης για να δείτε τι θα εκτελέσει το πρόγραμμα
- Αν έχετε μια μεγάλη έκφραση (expression), χρησιμοποιήστε προσωρινές μεταβλητές για να ελέγξετε τους τύπους των μεταβλητών
- Αναθέστε μια μεταβλητή στην έκφραση πριν από μια δήλωση return
- Εάν δεν μπορείτε να εντοπίσετε το σφάλμα, κάντε ένα σύντομο διάλειμμα ή ζητήστε βοήθεια

3.9. Πρακτική

Μία από τις πιο σημαντικές πτυχές στον προγραμματισμό είναι η πρακτική άσκηση. Όσο περισσότερη εξάσκηση κάνετε, τόσο καλύτερη θα είναι η πρόοδος σας. Ας ελπίσουμε ότι μέχρι το τέλος αυτής της υποενότητας, θα είστε σε θέση να κατανοήσετε εξίσου καλά, όπως μέχρι και τώρα, τη λογική πίσω από την σύνταξη ενός κώδικα, τους διαφορετικούς τύπους δεδομένων και τη χρήση τους, καθώς και πώς να δημιουργείτε μικρά προγράμματα χρησιμοποιώντας συναρτήσεις.

Σας ενθαρρύνουμε να εξασκηθείτε όσο το δυνατόν περισσότερο προκειμένου να καταφέρετε να δημιουργήσετε τα δικά σας προγράμματα και να βελτιώσετε τις προοπτικές της επαγγελματικής σας σταδιοδρομίας και επιτυχίας.

Είστε έτοιμοι για εξάσκηση; Καλή διασκέδαση όσο προγραμματίζετε!

Πιο κάτω, μπορείτε να βρείτε μια λίστα με ιστοσελίδες τις οποίες μπορείτε να συμβουλευτείτε αν θέλετε να εξασκήσετε περισσότερο τις δεξιότητές σας στον προγραμματισμό:

⇒ **W3Schools** - https://www.w3schools.com/python/python_exercises.asp
⇒ **GeeksforGeeks** - https://www.geeksforgeeks.org/python-exercises-practice-questions-and-solutions/
⇒ **PYnative** - https://pynative.com/python-exercises-with-solutions/

Αναφορές:

Downey, A. Elkner, J. & Meyers, C. (2008). How to Think Like a Computer Scientist: *Learning with Python*. Green Tea Press: Wellesley, Massachusetts.

GeeksforGeeks (2021). *Top 10 Python IDE and Code Editors in 2020.* https://www.geeksforgeeks.org/top-10-python-ide-and-code-editors-in-2020/

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Karpinski, Z., Biagi, F., & Di Pietro, G. (2021). Computational Thinking, Socioeconomic Gaps, and Policy Implications. IEA Compass: Briefs in Education. 12. *International Association for the Evaluation of Educational Achievement*.

O' Dea, B. (2021). *Python named most in-demand coding language for 2022*. https://www.siliconrepublic.com/careers/python-most-in-demand-coding-language-2022

Programiz. *How to Get Started with Python?* https://www.programiz.com/python-programming/first-program

Real Python. *Python args and kwargs: Demystified*. https://realpython.com/python-kwargs-and-args/

W3Schools. *Python Tutorial*. https://www.w3schools.com/python/

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

ΜΕΡΟΣ Β: Ενότητα του έργου CodER στους Μικροελεγκτές (10 ώρες)

Περιγραφή:

Στο δεύτερο μέρος, θα κάνουμε μια εισαγωγή στους μικροελεγκτές, τη χρήση και την εφαρμογή τους μέσω πρακτικών παραδειγμάτων. Η πρώτη υποενότητα αναλύει τα δομικά στοιχεία ενός μικροελεγκτή και τους διαφορετικούς τύπους μικροελεγκτών που υπάρχουν, με στόχο την εξοικείωση των εκπαιδευομένων με τις βασικές έννοιες γύρω από τον μικροελεγκτή. Στη συνέχεια, η υποενότητα θα ασχοληθεί με τη χρήση του λογισμικού Arduino και πώς αυτό μπορεί να συνδεθεί με έναν μικροελεγκτή Arduino σε μια βήμα προς βήμα προσέγγιση.

Φόρτος εργασίας:

10 ώρες

Μαθησιακά αποτελέσματα:

Με το πέρας αυτού του μαθήματος, θα είστε σε θέση:

- ⇒ Να αναγνωρίζετε τι είναι ένας μικροελεγκτής και να προσδιορίζετε τους διάφορους τύπους μικροελεγκτών που υπάρχουν
- ⇒ Να διακρίνετε τις διαφορές μεταξύ μιας αναλογικής και ψηφιακής εισόδου/εξόδου (I/0)
- ⇒ Να χρησιμοποιείτε βασική σύνταξη ενός Ολοκληρωμένου Περιβάλλοντος Ανάπτυξης Arduino (IDE)
- ⇒ Να εκτελείτε διαφορετικές λειτουργίες σε ένα Ολοκληρωμένο Περιβάλλον Ανάπτυξης Arduino και μικροελεγκτών

Απαιτούμενα υλικά και πόροι:

- ⇒ Υπολογιστής ή φορητός υπολογιστής
- ⇒ Πρόσβαση στο διαδίκτυο
- ⇒ Arduino Uno
- ⇒ Arduino IDE

Πρακτικά:

Αυτό το μέρος της ενότητας έχει σχεδιαστεί για να καλύπτει 10 ώρες μαθήματος. Κάθε υποενότητα έχει μια καθορισμένη διάρκεια διεκπεραίωσης, ωστόσο, οι εκπαιδευόμενοι ή οι εκπαιδευτικοί/ εκπαιδευτές είναι ελεύθεροι να αποφασίσουν πόσο χρόνο θα αφιερώσουν για κάθε επιμέρους θέμα της ενότητας, αναλόγως των προγενέστερών τους γνώσεων και της προηγούμενής τους ενασχόλησης με παρόμοια θέματα. Το περιεχόμενο βασίζεται στις αρχές της προοδευτικής μαθησιακής ανάπτυξης των εκπαιδευομένων και είναι διαβαθμισμένο με τρόπο ώστε να παρέχει βασικές γνώσεις και δεξιότητες σε αρχάριους.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Υποενότητες

1. Εισαγωγή στους μικροελεγκτές
2. Βασικές έννοιες στο προγραμματισμό με τον μικροελεγκτή Arduino
3. Εφαρμογές του Arduino

1. Εισαγωγή στους μικροελεγκτές

⇒ **Αριθμός συμμετεχόντων:** 1-10 ανά συντονιστή
⇒ **Διάρκεια:** 1,5 ώρα
⇒ **Μέθοδοι διδασκαλίας:** Παρουσίαση, Καθοδηγούμενη διδασκαλία, Βιωματική εκπαίδευση
⇒ **Απαιτούμενα υλικά:** Παρουσίαση, Πλακέτες Arduino και σταθερή σύνδεση στο διαδίκτυο

1.1. Τι είναι ένας μικροελεγκτής

Πριν εξηγήσουμε τι είναι ένας μικροελεγκτής, ας εξετάσουμε τις ακόλουθες ερωτήσεις:

- Είχατε ποτέ περιέργεια να μάθετε πώς δουλεύουν οι μικροσυσκευές; Ποια είναι η λογική πίσω από τις μικροσυσκευές;
- Θέλατε ποτέ να μάθετε πώς λειτουργούν τα συστήματα που ελέγχουν τους ανελκυστήρες ή τα ηλεκτρονικά παιχνίδια;
- Ή ακόμη θέλατε ποτέ να δημιουργήσετε το δικό σας ρομπότ ή ηλεκτρονικά σήματα για ένα μοντέλο σιδηροδρόμου;
- Έχετε αναρωτηθεί ποτέ πώς συλλέγονται και αναλύονται τα μετεωρολογικά δεδομένα;

Είστε περίεργοι να μάθετε; Λοιπόν, ας ξεκινήσουμε τότε!

Οι μικροελεγκτές μπορούν να διευκολύνουν την κατανόησή μας αναφορικά με τις προαναφερθέντες ηλεκτρονικές διαδικασίες μέσω πρακτικών δραστηριοτήτων.

Ένας μικροελεγκτής είναι ένα συμπαγές προγραμματιζόμενο ολοκληρωμένο κύκλωμα, υπεύθυνο για την εκτέλεση μιας συγκεκριμένης λειτουργίας σε μια συσκευή. Ένας μικροελεγκτής έχει τη δυνατότητα να ερμηνεύει τα δεδομένα που λαμβάνει από τα περιφερειακά του τα οποία διασυνδέονται με τις μονάδες εισόδου και εξόδου (I/O) που διαθέτει, μέσω ενός κεντρικού επεξεργαστή.

Οι μικροελεγκτές μπορούν να ενσωματωθούν σε ένα ευρύ φάσμα συστημάτων και συσκευών. Ορισμένες εφαρμογές των μικροελεγκτών μπορούν να βρεθούν σε κάμερες, ελεγκτές κινητήρα, κλειδαριές πόρτας, συναγερμούς πυρκαγιάς ή καπνού ή αισθητήρες θερμοκρασίας, φωτός και χρώματος.

Μπορείτε να σκεφτείτε, για παράδειγμα, ένα αυτοκίνητο το οποίο μπορεί να διαθέτει πολλούς μικροελεγκτές, κάθε ένας από τους οποίους είναι υπεύθυνος για ένα επιμέρους

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

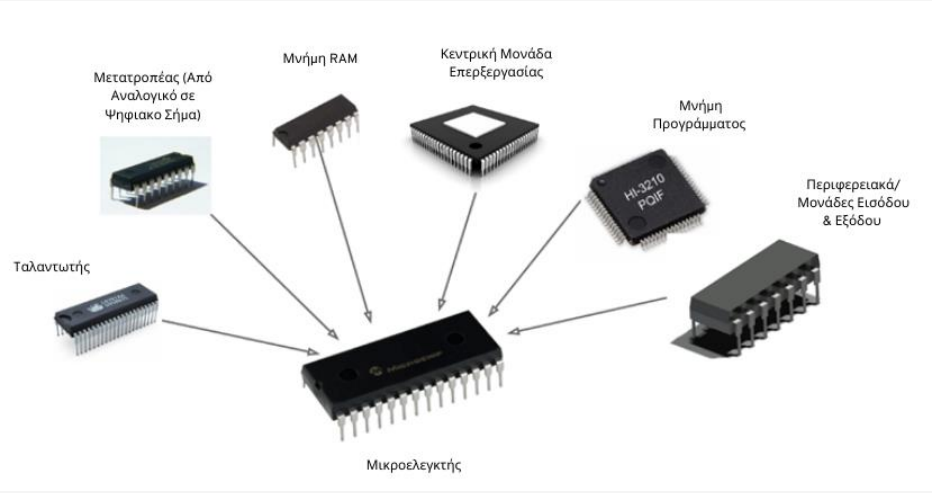
σύστημα ελέγχου του οχήματος, όπως π.χ. ένας μικροελεγκτής για τους αισθητήρες φωτός, ένας άλλος για τα συστήματα αντιεμπλοκής κατά την πέδηση, ένας για τα ηλεκτροκίνητα παράθυρα ή ένας άλλος για τα χειρόφρενα και ούτω καθεξής. Ένα όχημα μπορεί να διαθέτει μέχρι και 50 μικροελεγκτές, ο καθένας από τους οποίους είναι υπεύθυνος για μια επιμέρους λειτουργία, και οι οποίοι είτε επικοινωνούν μεταξύ τους είτε μέσω άλλων πιο σύνθετων συστημάτων για την εκτέλεση συγκεκριμένων ενεργειών.

Ένας μικροελεγκτής είναι ουσιαστικά ένα μικρό τσιπ, το οποίο αποτελείται από τα ακόλουθα στοιχεία:

1. Η Κεντρική Μονάδα Επεξεργασίας ΚΜΕ (αγγλικά: CPU): Ένας επεξεργαστής ή μια κεντρική μονάδα επεξεργασίας (CPU) χρησιμοποιείται για την επεξεργασία και την εκτέλεση διαφόρων εντολών που προσδιορίζουν τη λειτουργία ενός μικροελεγκτή. Ο επεξεργαστής διαβάζει και εκτελεί βασικές αριθμητικές και λογικές πράξεις, καθώς και πράξεις εισόδου/εξόδου. Είναι επίσης υπεύθυνος για τη μεταφορά δεδομένων, έτσι ώστε να επικοινωνεί τις εντολές σε άλλα εξαρτήματα μέσα σε ένα μεγαλύτερο ηλεκτρομηχανικό σύστημα.
2. Η Κεντρική Μνήμη: Η κύρια λειτουργία της μνήμης ενός μικροελεγκτή είναι η αποθήκευση δεδομένων προκειμένου να σταλούν στην κεντρική μονάδα επεξεργασίας και να ανταποκριθούν σε προκαθορισμένες λειτουργίες μέσω προγραμματισμού. Ένας μικροελεγκτής έχει δύο κύριους τύπους μνήμης:
    a. Η μνήμη του προγράμματος που είναι υπεύθυνη για την αποθήκευση μακροπρόθεσμων πληροφοριών σχετικά με τις εντολές που εκτελεί ο επεξεργαστής. Η μνήμη του προγράμματος είναι μη πτητική, που σημαίνει ότι διατηρεί τις πληροφορίες με την πάροδο του χρόνου χωρίς να χρειάζεται πηγή ενέργειας για να λειτουργήσει.
    b. Η μνήμη δεδομένων ή η μνήμη τυχαίας προσπέλασης (RAM) η οποία είναι υπεύθυνη για την αποθήκευση προσωρινών δεδομένων κατά τη διάρκεια της εκτέλεσης του προγράμματος. Αυτός ο τύπος μνήμης διατηρεί τα δεδομένα για όσο διάστημα η συσκευή είναι συνδεδεμένη σε πηγή τροφοδοσίας.
3. Περιφερειακά εισόδου/εξόδου (Input/Output): Τα περιφερειακά εισόδου/εξόδου είναι υπεύθυνα για την επικοινωνία των μικροελεγκτών με άλλα εξαρτήματα. Οι θύρες εισόδου λαμβάνουν πληροφορίες και τις στέλνουν στον επεξεργαστή για περαιτέρω επεξεργασία με τη μορφή δυαδικών δεδομένων. Με βάση τις εντολές του επεξεργαστή, οι θύρες εξόδου εκτελούν τις απαραίτητες εργασίες.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης



Εικόνα 16 – Τα συστατικά στοιχεία ενός μικροελεγκτή.

Πηγή: https://www.circuitbasics.com/introduction-to-microcontrolleres/

Κυκλοφορούν πολλοί τύποι μικροελεγκτών στην αγορά. Οι πιο δημοφιλείς τύποι είναι το Arduino, το Scratch, το Microbit και το Raspberry PI.

Πιο κάτω, μπορείτε να βρείτε τις επίσημες ιστοσελίδες τους για να τους ελέγξετε στον ελεύθερο σας χρόνο:

- Arduino: https://www.arduino.cc/en/soware
- Scratch: https://scratch.mit.edu/
- Microbit: https://microbit.org/
- Raspberry PI: https://www.raspberrypi.org/
-

Στην υποενότητα αυτή, θα επικεντρωθούμε στους μικροελεγκτές Arduino.

1.2. Τι είναι το Arduino και οι διάφοροι τύποι του

Τι είναι το Arduino

Το Arduino είναι μια πλατφόρμα ανοικτού κώδικα που χρησιμοποιείται για την κατασκευή ηλεκτρομηχανικών έργων. Το Arduino αποτελείται τόσο από μια φυσική προγραμματιζόμενη πλακέτα κυκλώματος (που συχνά αναφέρεται και ως μικροελεγκτής) όσο και από ένα λογισμικό ή ένα Ολοκληρωμένο Περιβάλλον Ανάπτυξης (IDE/Integrated Development Environment) που συνδέεται σε έναν υπολογιστή και το οποίο χρησιμοποιείται για να γράψει και να μεταφορτώσει έναν κώδικα ή πρόγραμμα στην φυσική πλακέτα του κυκλώματος, η οποία είναι μια μεγάλη πλατφόρμα που επιτρέπει την κατασκευή πρωτότυπων συσκευών ή συστημάτων (Green Steam Incubator, 2019).

Το μυστικό πίσω από την κατανόηση του τρόπου λειτουργίας ενός μικροελεγκτή Arduino κρύβεται στη πλακέτα του. Το Arduino σχεδιάστηκε από τους Massimo Banzi και τον David Cuartielles το 2005. Το Arduino αποτελεί μια πιο εναλλακτική λύση σε σχέση με τους

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

άλλους διαθέσιμους τύπους μικροελεγκτών, οι οποίοι τυγχάνουν να είναι και ακριβότεροι. Το Arduino επιτρέπει στους χρήστες να δημιουργήσουν διαδραστικά έργα όπως π.χ. συστήματα GPS, αισθητήρες φωτός, τηλεκατευθυνόμενα ρομπότ και ηλεκτρονικά παιχνίδια.

Τα κύρια εξαρτήματα του Arduino είναι:

1. Το λογισμικό: Το Ολοκληρωμένο Περιβάλλον Ανάπτυξης του Arduino είναι υπεύθυνο για την εγγραφή και τη φόρτωση προγραμμάτων για να επικοινωνήσουν με ένα υλισμικό. Οι λειτουργίες της πλακέτας του Arduino ελέγχονται με βάση τις εντολές που αποστέλλονται στον μικροελεγκτή μέσω του Arduino IDE.
2. Υλισμικό: Οι μητρικές πλακέτες του Arduino κυκλοφορούν στην αγορά σε διάφορους τύπους που διαθέτουν ενσωματωμένο μικροελεγκτή με μονάδες εισόδου και εξόδου, ψηφιακές και αναλογικές, και μπορούν να συνδεθούν σε διαφόρων ειδών στοιχεία όπως αισθητήρες για τη συλλογή δεδομένων ή την εκτέλεση λειτουργιών. Μερικές από τις λειτουργίες που μπορούν να εκτελέσουν αφορούν την ενεργοποίηση ενός κινητήρα, την ενεργοποίηση ή απενεργοποίηση ενός λαμπτήρα LED ή το κλείδωμα/ ξεκλείδωμα μιας πόρτας.
3. Γλώσσα προγραμματισμού: Η γλώσσα προγραμματισμού που χρησιμοποιείται στο Arduino είναι μια απλοποιημένη έκδοση της C++.

Οι διάφοροι τύποι του Arduino

Υπάρχουν διάφορες διαθέσιμες μητρικές πλακέτες Arduino που διακρίνονται με βάση τον τύπο μικροελεγκτή που χρησιμοποιούν. Οι διαφορές μεταξύ τους διακρίνονται στα ακόλουθα χαρακτηριστικά:

- αριθμός μονάδων εισόδου και εξόδου (π.χ. αισθητήρες, λαμπτήρες LED, οι σημάνσεις στην πλακέτα)
- ταχύτητα
- τάση λειτουργίας σε βολτ
- συντελεστής μορφής (form factor) κ.λπ.

Ορισμένες μητρικές πλακέτες έχουν σχεδιαστεί αποκλειστικά για να είναι ενσωματωμένες και δεν προσφέρουν διεπαφή προγραμματισμού εφαρμογών (αγγλ. API, από το Application Programming Interface). Κάποιοι τύποι μητρικών πλακετών μπορούν να τροφοδοτηθούν απευθείας με τάση 3,7V, ενώ άλλοι χρειάζονται τάση τουλάχιστον 5V για να λειτουργήσουν. Όποιες και αν είναι οι διαφορές μεταξύ των χαρακτηριστικών τους, μπορούν να προγραμματιστούν μέσω του IDE του Arduino.

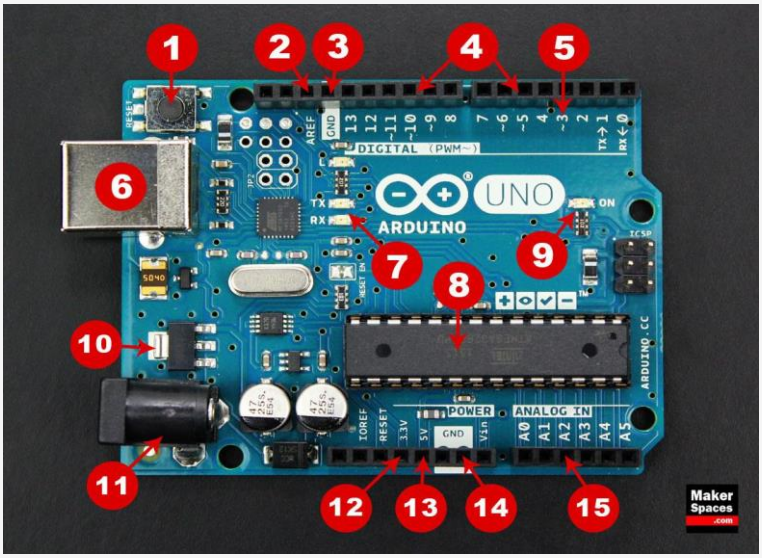
Στην πιο κάτω εικόνα, μπορείτε να δείτε μερικά παραδείγματα από τους διάφορους τύπους μητρικών πλακετών του Arduino.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Εικόνα 17 – Διαφορετικοί τύποι πλακετών Arduino.
Πηγή: https://techatronic.com/types-of-arduino-boards-arduino-uno-mega-mini-specification/

Μια από τις πιο δημοφιλείς πλακέτες Arduino είναι το Arduino Uno. Παρόλο που δεν ήταν η πρώτη πλακέτα που κυκλοφόρησε στην αγορά, παραμένει μία από τις πιο προσφιλείς και ευρέως χρησιμοποιούμενες πλακέτες ανάμεσα στο αγοραστικό κοινό.



Εικόνα 18 – Arduino UNO.
Πηγή: https://www.makerspaces.com/arduino-uno-tutorial-beginners/

1. Σήμανση RESET – Επανεκκινεί οποιονδήποτε κώδικα φορτώθηκε στην πλακέτα Arduino
2. Η ακίδα (pin) AREF – Ακίδα Αναλογικής Εισόδου η οποία ρυθμίζει την τάση αναφοράς της εξωτερικής τροφοδοσίας.
3. Ακίδα γείωσης (GND) – Κλείνει το ηλεκτρικό κύκλωμα και παρέχει μια κοινή τάση αναφοράς

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

4. Ψηφιακή είσοδος/έξοδος – Οι ακίδες 0-13 μπορούν να χρησιμοποιηθούν για ψηφιακή είσοδο ή έξοδο
5. Σύστημα Διαμόρφωσης Πλάτους Παλμού (αγγλ. PWM, Pulse Width Modulation) – Οι ακίδες που επισημαίνονται με το σύμβολο (~) μπορούν να προσομοιώσουν μια αναλογική έξοδο
6. Σύνδεση USB – Τροφοδοτεί το Arduino για τη φόρτωση σκίτσου (sketch), δηλαδή ενός προγράμματος ή κώδικα
7. TX/RX – Σημάνσεις που ανάβουν μέσω δύο Λαμπτήρων LED όταν το Arduino μεταδίδει ή λαμβάνει δεδομένα μέσω USB
8. ATmega Μικροελεγκτής – Αποθηκεύει τα προγράμματα (ο «εγκέφαλος» του Arduino)
9. Σήμανση POWER – Λαμπτήρας LED που ανάβει όταν η πλακέτα είναι συνδεδεμένη με μια πηγή τροφοδοσίας
10. Σταθεροποιητής τάσης – Ελέγχει το ποσό της τάσης που μεταδίδεται στην μητρική πλακέτα του Arduino
11. DC Power Barrel Jack – Φορτίζει το Arduino μέσω τροφοδοτικού
12. Ακίδα ισχύος 3.3V – Τροφοδοτεί με 3.3 βολτ το κύκλωμα
13. Ακίδα ισχύος 5V – Τροφοδοτεί με 5 βολτ το κύκλωμα
14. Ακίδες γείωσης – Κλείνουν το ηλεκτρικό κύκλωμα και παρέχουν μια κοινή τάση αναφοράς
15. Αναλογικές ακίδες – Διαβάζει τα σήματα από τον αναλογικό αισθητήρα και τα μετατρέπει σε ψηφιακά

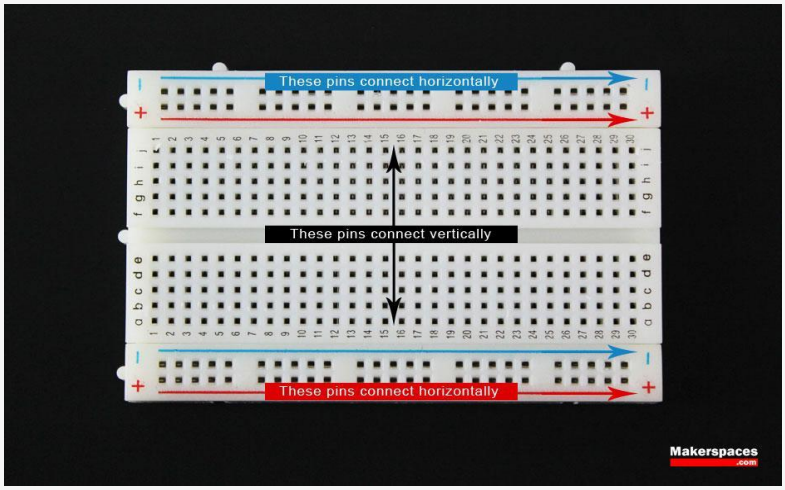
Η μητρικές πλακέτες Arduino Uno πρέπει να συνδεθούν με μια πηγή τροφοδοσίας για να λειτουργήσουν. Υπάρχουν διάφοροι τρόποι σύνδεσης της μητρικής πλακέτας σε μια πηγή τροφοδοσίας, όπως π.χ. απευθείας μέσω USB ενός υπολογιστή, ενώ στην περίπτωση της χρήσης του Arduino με κινητό τηλέφωνο μπορείτε να χρησιμοποιήσετε μια μπαταρία των 9V. Ο τελευταίος τρόπος απαιτεί παροχή ρεύματος 9V AC για να λειτουργήσει.

Σχήμα 19 – Πηγές τροφοδοσίας του Αρδουίνου.
Πηγή: https://www.makerspaces.com/arduino-uno-tutorial-beginners/

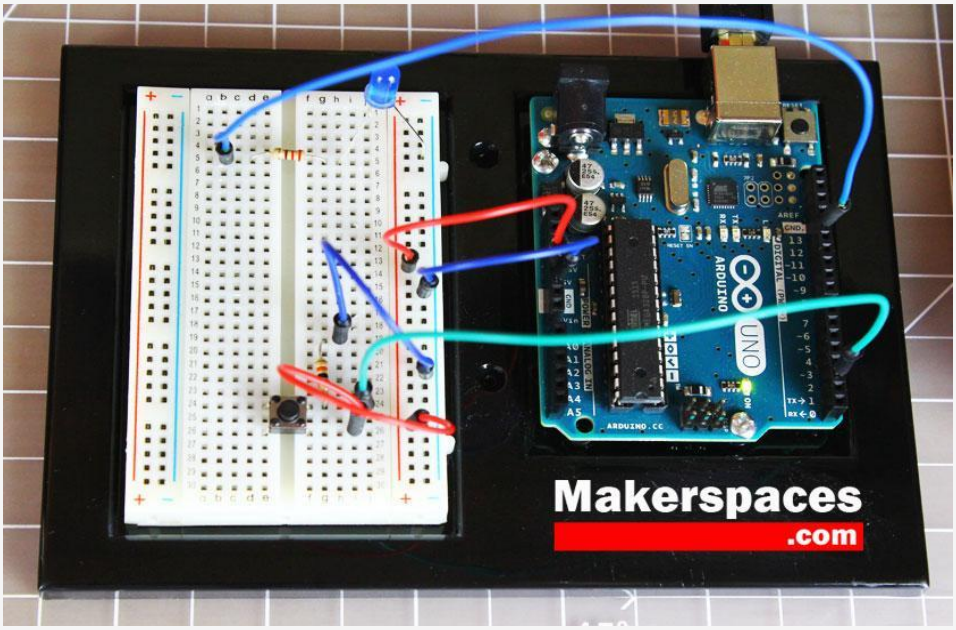
Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Ένα άλλο σημαντικό στοιχείο στην μητρική πλακέτα του Arduino είναι το breadboard, που ονομάζεται επίσης *Solderless Breadboard*. Χρησιμοποιείται κατά τη φάση της προτυποποίησης (breadboarding) για την αξιολόγηση της λειτουργικότητας του κυκλώματος και επιτρέπει την προσωρινή δημιουργία και τον πειραματισμό με διαφορετικούς σχεδιασμούς κυκλωμάτων. Τα σημεία πρόσδεσης (οπές) του πλαστικού περιβλήματος περιέχουν μεταλλικά κλιπ που συνδέονται μεταξύ τους με λωρίδες αγώγιμου υλικού. Ωστόσο, το breadboard δεν τροφοδοτείται από μόνο του και πρέπει να συνδεθεί με την πλακέτα Arduino μέσω καλωδίων jumper. Ο τύπος καλωδίων jumper χρησιμοποιείται επίσης για τη διαμόρφωση του κυκλώματος μέσω σύνδεσης με αντιστάσεις, διακόπτες και άλλων εξαρτημάτων μαζί.



Εικόνα 20 – Το Breadboard.
Πηγή: https://www.makerspaces.com/arduino-uno-tutorial-beginners/



Εικόνα 21 – Ολοκληρωμένο κύκλωμα ενός Arduino.
Πηγή: https://www.makerspaces.com/arduino-uno-tutorial-beginners/

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

1.3. Βασικές έννοιες: Είσοδος, έξοδος, αναλογική, ψηφιακή

Ψηφιακή Είσοδος και Έξοδος(I/O) στη γλώσσα του προγραμματισμού

Οι ψηφιακές είσοδοι και έξοδοι αντιπροσωπεύουν τις αλληλεπιδράσεις μεταξύ ενός ρομπότ/μηχανής με τον πραγματικό κόσμο. Με πιο απλά λόγια, η ψηφιακή είσοδος αναφέρεται στα δεδομένα εκείνα που λαμβάνει ένα ρομπότ/ μηχανή ενώ η ψηφιακή έξοδος αναφέρεται στα αποτελέσματα που προκύπτουν.

Οι ψηφιακές είσοδοι συνήθως μεταδίδονται από αισθητήρες, όπως διακόπτες, ποτενσιόμετρα ή βιντεοκάμερες, ενώ οι ψηφιακές έξοδοι αναφέρονται σε άμεσες ενέργειες που ενεργοποιούνται από κινητήρες, όπως το αναβόσβημα των λαμπτήρων LED ή την απενεργοποίηση ενός συναγερμού.

Εξετάστε το ακόλουθο παράδειγμα: Τα πλήκτρα σε ένα πληκτρολόγιο αντιπροσωπεύουν τις ψηφιακές εισόδους. Κατά την πληκτρολόγηση, ο υπολογιστής λαμβάνει πληροφορίες τις οποίες επεξεργάζεται. Ωστόσο, αυτή η διαδικασία δεν είναι ορατή σε εμάς. Ο υπολογιστής ή μια μηχανή λαμβάνει τα δεδομένα εισόδου (δηλαδή, ό,τι πληκτρολογούμε) και τα εξάγει ως ψηφιακή έξοδο στην οθόνη.

Στο πλαίσιο του Arduino: Ένα πρόγραμμα μπορεί να λάβει την πίεση ενός πλήκτρου ως ψηφιακή είσοδο και, στη συνέχεια, ως ψηφιακή έξοδο να ανάψει ένα λαμπτήρα LED εάν είναι ενεργοποιημένος. Η επεξεργασία των δεδομένων που γίνεται κατά την πίεση ενός πλήκτρου μπορεί να μην είναι ορατή, αλλά ο λαμπτήρας LED που αναβοσβήνει αποτελεί απόδειξη της ψηφιακής εξόδου.

Υπάρχουν δύο τύποι εισόδου/ εξόδου: Αναλογική και Ψηφιακή I/Os.

Ποια είναι η διαφορά μεταξύ μιας αναλογικής και ψηφιακής εξόδου I/Os

Υπάρχουν δύο τρόποι με τους οποίους μπορείτε να διακρίνετε ένα αναλογικό σήμα από ένα ψηφιακό σήμα:

1. Από τον τύπο του αισθητήρα
2. Μέσω της μεθόδου επεξεργασίας (δηλαδή του χρόνου και της ανάλυσης)

1. Οι Αναλογικοί Αισθητήρες σε αντίθεση με τους Ψηφιακούς Αισθητήρες: Ένας λαμπτήρας LED μπορεί να είναι είτε σε κατάσταση εντός η εκτός λειτουργίας (ON/OFF) είτε να έχει μεταβλητή ρύθμιση έντασης, όπως ON με χαμηλή ένταση ή ON με υψηλή ένταση.
    ⇒ Εάν ο αισθητήρας είναι ένας διακόπτης τοποθετημένος σε έναν λαμπτήρα LED, τότε θα ελέγξει αν αυτός (ο λαμπτήρας LED) είναι εντός ή εκτός λειτουργίας και δεν θα ανιχνεύσει τίποτα άλλο. Το παράδειγμα που μόλις εξετάσαμε αποτελεί μια **ψηφιακή είσοδο.**

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

⇒ Εάν ο αισθητήρας είναι ένας φωτοαντίστασης τύπου LDR (light dependent resistor) θα ανιχνεύσει το φως και θα το μετατρέψει σε μια αναλογική τιμή που κυμαίνεται μεταξύ του 0-255. Μπορούμε να ανιχνεύσουμε αν ένας λαμπτήρας LED βρίσκεται στο 0% (off) ή στο 100% (πλήρης φωτεινότητα) και επίσης μπορούμε να διαβάσουμε πολλές άλλες ενδιάμεσες τιμές (π.χ. 20, 23%, 86%). Αυτό αποτελεί ένα παράδειγμα **αναλογικής εισόδου.**

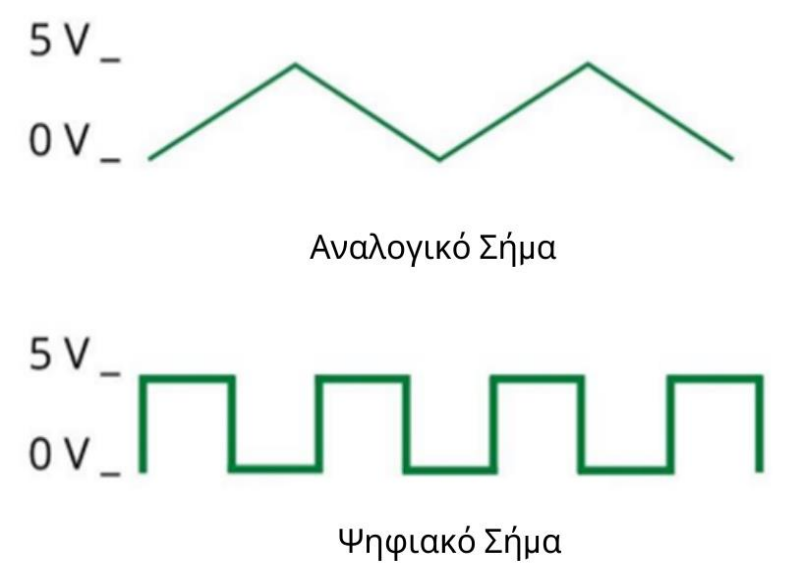
2. Η Αναλογική Επεξεργασία σε αντίθεση με την Ψηφιακή Επεξεργασία: Για να προσδιορίσετε αν έχετε μια Αναλογική ή Ψηφιακή Επεξεργασία, απλά ελέγξτε με βάση τον:
   ⇒ Χρόνο: μια Αναλογική Επεξεργασία στο Arduino επεξεργάζεται συνεχώς τα δεδομένα και κάθε φορά που τροποποιείται η είσοδος, η έξοδος αλλάζει αμέσως ανάλογα με την είσοδο. Η ενημέρωση στο Arduino γίνεται άμεσα. Στην Ψηφιακή Επεξεργασία, υπάρχει μικρή καθυστέρηση στην ενημέρωση του Arduino μέχρι το σύστημα να καταγράψει την αλλαγή. Η καθυστέρηση αυτή καθιερώνεται μέσω της μεθόδου της «συχνότητας δειγματοληψίας» και ελέγχεται με ρολόι.
   ⇒ Ανάλυση: Η αναλογική επεξεργασία διενεργεί άπειρες αναλύσεις επειδή δεν σταματά ποτέ την επεξεργασία δεδομένων και γι' αυτό τον λόγο μπορεί να δώσει πολλές διαφορετικές τιμές. Αντίθετα, η ψηφιακή ανάλυση λειτουργεί με βάση το δυαδικό σύστημα (0 και 1), πράγμα που σημαίνει ότι επεξεργάζεται μόνο δύο τιμές και δύο τύπους σημάτων (αναλογικό και ψηφιακό).

Παραδείγματα με Λαμπτήρα LED:

- Ενεργοποίηση/ Απενεργοποίηση με αναλογική επεξεργασία: όταν ενεργοποιηθεί το δυαδικό σύστημα τιμών, το κύκλωμα θα αλλάξει αμέσως την ένταση του Λαμπτήρα LED αναλόγως.
- Ενεργοποίηση/ Απενεργοποίηση με ψηφιακή επεξεργασία: με μια καθορισμένη συχνότητα δειγματοληψίας, κάθε 5 δευτερόλεπτα, το σύστημα θα διαβάζει τον διακόπτη και θα υποδεικνύει εάν το φως είναι ενεργοποιημένο ή απενεργοποιημένο, ανεξάρτητα από το πόσο έντονο ή φωτεινό είναι. Εάν αλλάξετε το δυαδικό σύστημα μέσα σε αυτά τα 5 δευτερόλεπτα, τότε το φως του λαμπτήρα LED δεν θα αλλάξει.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης



Εικόνα 22 – Αναλογικό Σήμα σε αντίθεση με το Ψηφιακού Σήμα

Περίληψη

Εκ των πραγμάτων, τα αναλογικά σήματα είναι εκείνα που χρησιμοποιούνται συνηθέστερα. Ωστόσο, τα ψηφιακά σήματα και η ψηφιακή επεξεργασία στον τομέα της ρομποτικής είναι μια καλύτερη επιλογή. Οι λόγοι γι' αυτό είναι: 1) η ψηφιακή επεξεργασία είναι φθηνότερη και παρέχει μεγαλύτερη ευελιξία σε σύγκριση με την αναλογική επεξεργασία 2) τα προγράμματα λειτουργούν σε δυαδική μορφή κατά τρόπο παρόμοιο με τα ψηφιακά σήματα και 3) τα αναλογικά σήματα είναι πιο πιθανό να επηρεαστούν από τον θόρυβο που προκαλεί το ηλεκτρονικό κύκλωμα.

Εξαιτίας των λόγων που αναπτύξαμε πιο πάνω, τα αναλογικά σήματα συχνά μετατρέπονται σε ψηφιακά σήματα. Πρέπει να σημειωθεί ότι όλα τα αναλογικά σήματα, τόσο της εισόδου ή της εξόδου, μπορούν να μετατραπούν σε ψηφιακά σήματα, αλλά όχι το αντίστροφο.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

2. Βασικές έννοιες στο προγραμματισμό με Arduino

⇒ **Αριθμός συμμετεχόντων:** 1-10 ανά συντονιστή
⇒ **Διάρκεια:** 3 ώρες
⇒ **Μέθοδοι διδασκαλίας:** Παρουσίαση, Καθοδηγούμενη διδασκαλία, Βιωματική εκπαίδευση
⇒ **Απαιτούμενα υλικά:** Παρουσίαση, Arduino IDE και Μητρικές Πλακέτες, υπολογιστής (1 ανά συμμετέχοντα) και σταθερή σύνδεση στο διαδίκτυο

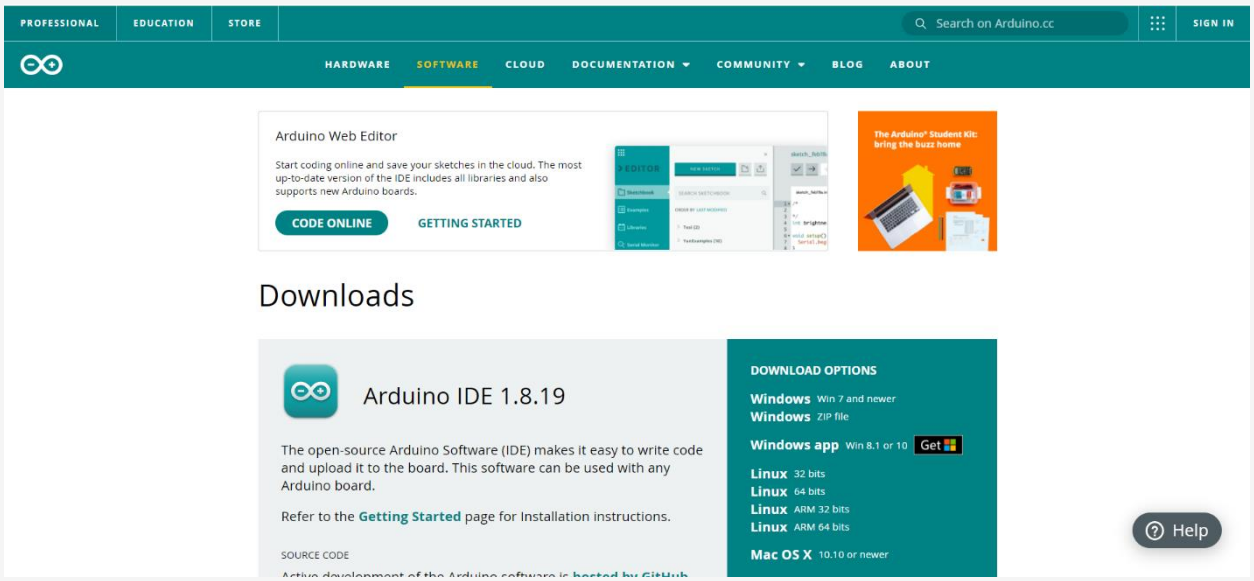
2.1. Ρύθμιση του IDE Arduino και βασικές εντολές

Στην υποενότητα αυτή, θα μάθουμε πώς να κάνουμε λήψη και πώς να «τρέχουμε» το IDE του Arduino για πρώτη φορά. Θα εξετάσουμε επίσης ορισμένες βασικές λειτουργίες του Arduino IDE προκειμένου να εξοικειωθούμε με το πρόγραμμα.

Πώς να κάνετε λήψη του Arduino IDE:

Παρέχονται οι ακόλουθες οδηγίες για τη λήψη και τη ρύθμιση του Arduino IDE:

1. Μεταβείτε στη διεύθυνση: https://www.arduino.cc/
2. Κάντε κλικ στο Software. Θα ανοίξει αυτόματα τη σελίδα που φαίνεται παρακάτω στην Εικόνα 23.



Εικόνα 23 – Λήψη του Arduino IDE

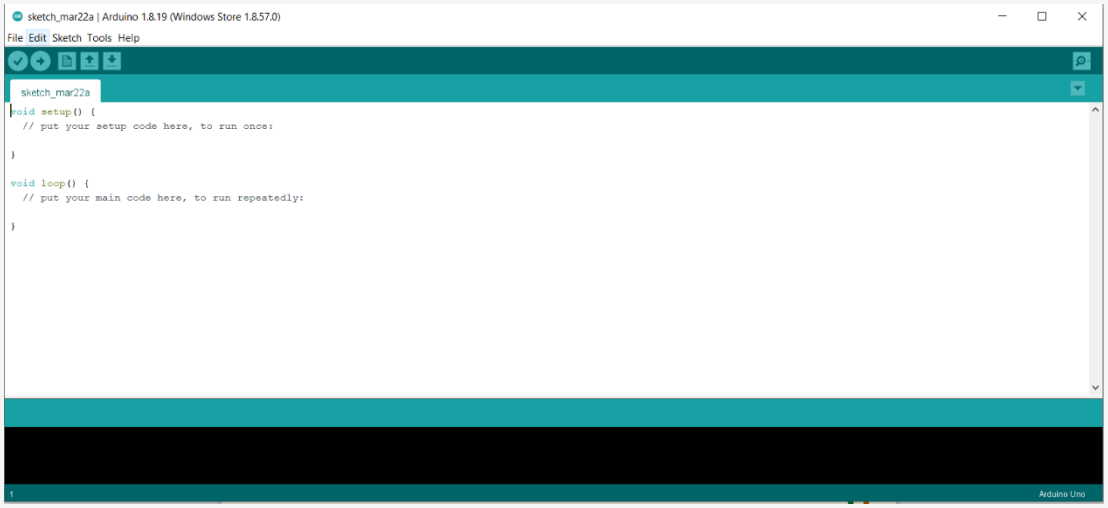
3. Κάντε κλικ στην επιλογή λήψης που ταιριάζει με το λειτουργικό σύστημα του υπολογιστή σας από το "Download Options" (βλ. Εικόνα 23).Εάν έχετε αμφιβολίες, κάντε κλικ στο "Getting Started" για να διαβάσετε περισσότερες πληροφορίες σχετικά με την εγκατάσταση του λογισμικού στον υπολογιστή σας.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Το Arduino IDE χρησιμοποιείται για τη δημιουργία, το άνοιγμα και την αλλαγή σκίτσων. Η μητρική πλακέτα ορίζεται από τα σκίτσα που γράφουμε στον υπολογιστή μέσω του περιβάλλοντος Arduino IDE. Μπορείτε να επιλέξετε μεταξύ των σημάνσεων που εμφανίζονται στο πάνω μέρος του IDE ή επιλέγοντας από το μενού.

Μόλις ολοκληρωθεί η λήψη, θα ανοίξει αυτόματα η σελίδα που φαίνεται στην πιο κάτω εικόνα:



Εικόνα 24 – Άνοιγμα του Arduino IDE για πρώτη φορά

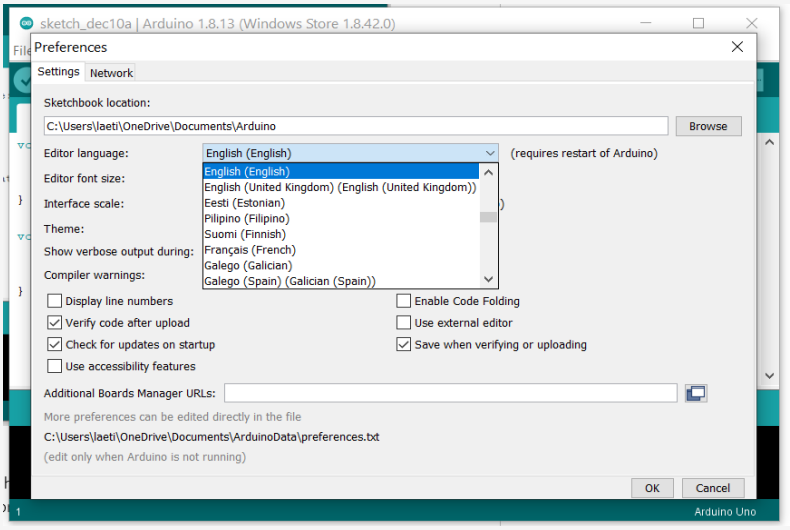
Πώς να αλλάξετε τη γλώσσα:

1. Κάντε κλικ στο *FILE* και επιλέξτε *PREFERENCES*.

2. Δίπλα από το *Editor Language*, εμφανίζεται ένα dropdown μενού που περιέχει τις γλώσσες που υποστηρίζει το Arduino.

3. Επιλέξτε τη γλώσσα της προτίμησής σας από το μενού.

4. Για να χρησιμοποιήσετε την επιλεγμένη γλώσσα, επανεκκινήστε το λογισμικό.



Εικόνα 25 – Αλλαγή ρυθμίσεων γλώσσας στο Arduino IDE

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

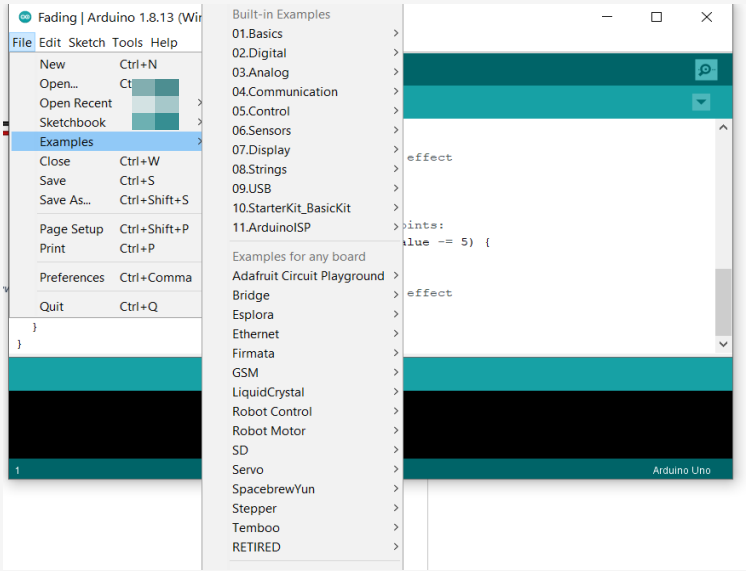
Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Πώς να δημιουργήσετε ένα νέο πρότζεκτ:

File -> New

Είναι επίσης δυνατό να χρησιμοποιήσετε παραδείγματα από έτοιμα πρότζεκτ για έμπνευση ή μπορείτε επίσης να τα αναπαράγετε:

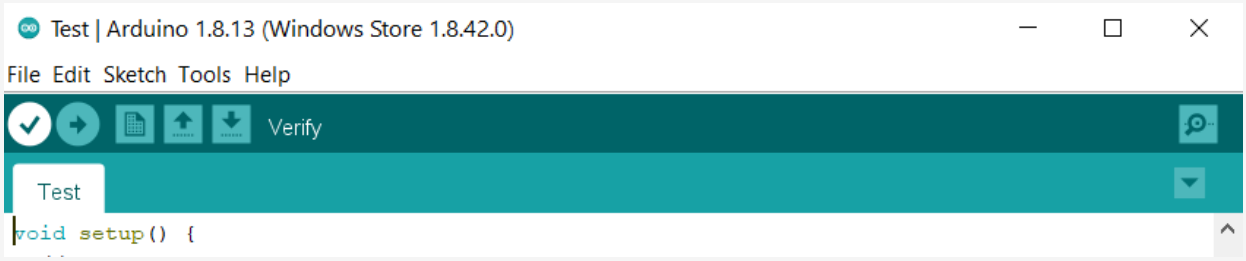
File -> Examples



Εικόνα 26 – Δημιουργία ενός νέου πρότζεκτ στο Arduino IDE

Πώς να επαληθεύσετε ένα πρότζεκτ:

Κάντε κλικ στα αριστερά κάτω από την Καρτέλα *FILE*. Αυτή θα γίνει πορτοκαλί όσο συλλέγει πληροφορίες. Περιμένετε μέχρι να επανέλθει στο αρχικό της χρώμα.



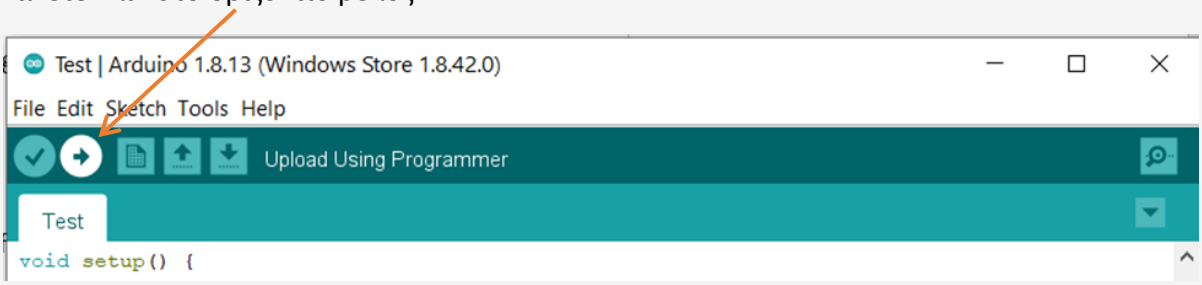
Σχήμα 27 – Επαλήθευση ενός πρότζεκτ στο Arduino IDE

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Πώς να μεταφορτώσετε ένα πρόγραμμα:

Η πλακέτα του Arduino θα πρέπει να συνδεθεί στον υπολογιστή σας με ένα καλώδιο USB για να μεταφορτώσετε το πρόγραμμα. Για να μεταφορτώσετε το πρόγραμμα, θα πρέπει να κάνετε κλικ στο οριζόντιο βέλος:



Σχήμα 28 – Μεταφόρτωση προγράμματος στο Arduino IDE

Όταν το πρόγραμμα μεταφορτωθεί, το βέλος θα επανέλθει στο αρχικό του χρώμα.

Εάν έχετε συνδέσει και εγκαταστήσει την πλατφόρμα του Arduino σύμφωνα με τις ρυθμίσεις που θέλετε, θα είστε σε θέση να παρακολουθήσετε το πρόγραμμά σας σε δράση.

2.2. Ο Προγραμματισμός στο Arduino και η μεταφόρτωση προγραμμάτων στην πλατφόρμα

Πώς να προγραμματίσετε στο Arduino:

Αφού έχετε πρώτα κατανοήσει το υλισμικό της μητρικής πλακέτας τύπου Arduino UNO και πώς γίνεται η λήψη του λογισμικού Arduino, τώρα μπορείτε να προχωρήσετε με τον προγραμματισμό.

Τα προγράμματα Arduino μπορούν να διακριθούν σε τρία κύρια μέρη:

1. Δομή
2. Τιμές (μεταβλητές και σταθερές) και
3. Συναρτήσεις

Στην υποενότητα αυτή, θα μάθουμε πώς να προγραμματίζουμε το Arduino βήμα προς βήμα, καθώς και πώς μπορούμε να γράψουμε ένα πρόγραμμα χωρίς κανένα συντακτικό ή μεταφραστικό λάθος.

Το Arduino – Η Δομή του Προγράμματος

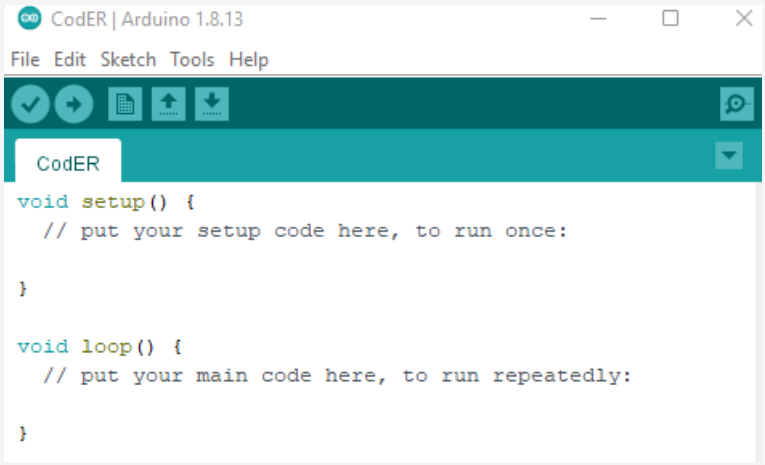
Ας ξεκινήσουμε με τη Δομή. Η δομή του λογισμικού του Arduino αποτελείται από δύο κύριες συναρτήσεις:

Συνάρτηση Setup( )

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Συνάρτηση Loop( )



Εικόνα 29 - Arduino – Η Δομή του Προγράμματος

Η συνάρτηση setup() καλείται μια φορά, όταν το σκίτσο (sketch) ξεκινά ή όποτε κάνει επαναφορά (reset) η πλατφόρμα Arduino. Σε αυτήν γίνονται οι αρχικοποιήσεις των μεταβλητών, η ρύθμιση της κατάστασης των ακίδων (pins) και η προετοιμασία των βιβλιοθηκών κ.λπ. Η συνάρτηση setup() εκτελείτε μόνο μια φορά μετά από κάθε ενεργοποίηση ή επαναφορά (reset) της πλακέτας του Arduino. Μετά τη δημιουργία μιας συνάρτησης setup(), η οποία αρχικοποιεί και ορίζει τις αρχικές τιμές, καλείται ξανά και ξανά η συνάρτηση loop(), η οποία κάνει ακριβώς αυτό που υποδηλώνει και το όνομά της, δηλαδή δημιουργεί βρόχους επανάληψης, επιτρέποντας έτσι στο πρόγραμμα να ανταποκριθεί σε εξωτερικά ερεθίσματα. Θα πρέπει να χρησιμοποιείτε αυτή την συνάρτηση για να έχετε έναν ενεργό έλεγχο της πλακέτας του Arduino.

Τύποι δεδομένων

Ένα τυπικό περιβάλλον Arduino είναι παρόμοιο με την γλώσσα προγραμματισμού C++, η οποία υποστηρίζει τη χρήση βιβλιοθηκών και built-in assumptions σχετικά με το περιβάλλον-στόχο, για την απλοποίηση της διαδικασίας της κωδικοποίησης. Πιο κάτω παρατίθεται μια λίστα μερικών τύπων δεδομένων που χρησιμοποιούνται συνήθως στο περιβάλλον του Arduino:

- Μπούλειοι Τύποι (boolean) που είναι μια λογική δυαδική τιμή (8 bit): true (αληθές)/false (ψευδές)
- byte (8 bit): μη προσημασμένος χαρκτήρας 8 ψηφίων από το 0-255
- char (8 bit): προσημασμένος χαρακτήρας 8 ψηφίων από το -128 έως 127.
- word (16 bit): μη προσημασμένος ακέραιος αριθμός 16 ψηφίων από το 0-65535
- int (16 bit): προσημασμένος ακέραιος αριθμός 16 ψηφίων από το -32768 έως 32767.
- long (32 bit) - προσημασμένος ακέραιος αριθμός 32 ψηφίων από το 0-4,294,967,295.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Arduino – Μεταβλητές

Οι μεταβλητές στη γλώσσα προγραμματισμού C, τις οποίες χρησιμοποιεί το Arduino, έχουν μια ιδιότητα που ονομάζεται scope και αυτή αφορά την εμβέλεια των μεταβλητών. Η εμβέλεια των μεταβλητών καθορίζει τη θέση στην οποία μια μεταβλητή μπορεί να χρησιμοποιηθεί στον κώδικα Arduino. Με άλλα λόγια, το scope είναι μια περιοχή του προγράμματος το οποίο αποτελείται από τρία μέρη, στα οποία δηλώνονται οι μεταβλητές. Αυτά τα μέρη αποτελούν:

- Μέσα στο κυρίως σώμα μιας συνάρτησης ή ενός μπλοκ κώδικα (block), στο οποίο ορίζονται οι τοπικές μεταβλητές.
- Στον ορισμό των παραμέτρων των μεταβλητών μιας συνάρτησης, οι οποίες ονομάζονται ορίσματα.
- Έξω από όλες τις συναρτήσεις, στις οποίες ορίζονται οι καθολικές μεταβλητές.

Παράδειγμα μεταβλητών:

Στο παράδειγμα αυτό, χρησιμοποιούμε τη συνάρτηση 'countUp,' για τη δημιουργία ενός ακέραιου αριθμού, που αρχικά ορίστηκε ως ο αριθμός μηδέν. Η μεταβλητή ανεβαίνει κατά ένα σε κάθε βρόχο.

```
int countUp = 0;             //creates a variable integer called 'countUp'

void setup() {
  Serial.begin(9600);        // use the serial port to print the number
}

void loop() {
  countUp++;                 //Adds 1 to the countUp int on every loop
  Serial.println(countUp);   // prints out the current state of countUp
  delay(1000);
}
```

Εικόνα 30 – Παράδειγμα μεταβλητών στο Arduino.
Πηγή: https://www.arduino.cc/reference/en/language/variables/data-types/int/

Ένας τελεστής είναι ένα σύμβολο που λέει στον μεταφραστή του Arduino να εκτελέσει συγκεκριμένες μαθηματικές ή λογικές συναρτήσεις. Η γλώσσα προγραμματισμού C είναι γεμάτη από ενσωματωμένους τελεστές και παρέχει τους ακόλουθους τύπους τελεστών:

- Αριθμητικοί τελεστές
- Τελεστές Σύγκρισης (ή Σχεσιακοί Τελεστές)
- Λογικοί Τελεστές
- Bitwise Τελεστές (Λογικοί Τελεστές bit)
- Τελεστές Αντικατάστασης και ο Τελεστής κόμμα (Compound Τελεστές)

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

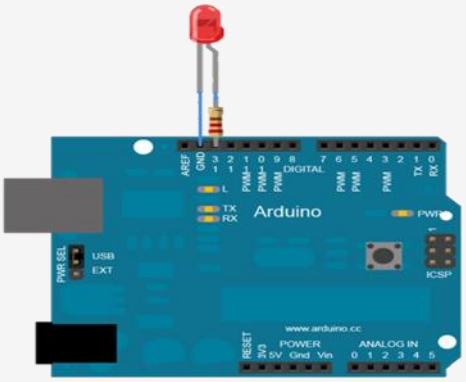
Περισσότερες πληφορίες για τους τύπους τελεστών που χρησιμοποιούνται στη γλώσσα προγραμματισμού C, μπορείτε να βρείτε στη διεύθυνση πιο κάτω:
https://www.tutorialspoint.com/arduino/arduino_operators.htm

2.3. Αναβόσβημα ενός Λαμπτήρα Led με Arduino

Στην υποενότητα αυτή, δημιουργούμε και μεταφορτώνουμε ένα απλό σκίτσο που θα κάνει έναν λαμπτήρα LED να αναβοσβήνει επανειλημμένα, ενεργοποιώντας και απενεργοποιώντας τον για διαστήματα του ενός δευτερολέπτου.

Βήματα:

- Συνδέστε το Arduino στον υπολογιστή με το καλώδιο USB.
- Άνοίξτε το IDE
- Επιλέξτε τη θύρα *Tools4Serial*. Βεβαιωθείτε ότι η θύρα USB είναι επιλεγμένη και ότι η πλακέτα Arduino είναι σωστά συνδεδεμένη.
- Συνδέστε έναν λαμπτήρα LED στην ψηφιακή ακίδα (pin) 13 του Arduino, όπως φαίνεται στην πιο κάτω εικόνα. Μια ψηφιακή ακίδα μπορεί είτε να ανιχνεύσει ένα ηλεκτρικό σήμα είτε να δημιουργήσει μία εντολή. Στο μικρή αυτή εφαρμογή, θα προσπαθήσουμε να παράγουμε ένα ηλεκτρικό σήμα που θα ανάψει τον λαμπτήρα LED.



Εικόνα 31 – Παράδειγμα από το αναβόσβημα ενός Λαμπτήρα LED μέσω μιας Ψηφιακής Ακίδας στο Arduino
Πηγή: https://www.instructables.com/How-to-Blink-LED-Using-Arduino/

- Εισαγάγετε τα ακόλουθα στο σκίτσο σας μεταξύ των αγκίστρων { and }, κάτω από τη συνάρτηση void setup():
  pinMode(13, OUTPUT); // ορίζει την ψηφιακή ακίδα (pin) 13 ως έξοδο

Ο αριθμός 13 στην καταχώρηση αντιπροσωπεύει την ψηφιακή ακίδα στην οποία απευθύνεστε. Ουσιαστικά, ορίζετε την συγκεκριμένη ακίδα ως έξοδο, η οποία και θα παράξει ένα ηλεκτρικό σήμα. Στην περίπτωση που θα θέλατε να ανιχνεύσετε ένα εισερχόμενο ηλεκτρικό σήμα, θα χρησιμοποιούσατε αντ' αυτού μια ψηφιακή ακίδα

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

ΕΙΣΟΔΟΥ. Σημειώστε ότι η συνάρτηση pinMode() τελειώνει με ερωτηματικό (;). Κάθε συνάρτηση σε ένα σκίτσο του Arduino τελειώνει με άνω τελεία.

- Αποθηκεύστε ξανά το σκίτσο σας για να βεβειωθείτε ότι δεν χάσατε καμία από τις εργασίες σας, επιλέγοντας File > Save As.
- Εισαγάγετε ένα σύντομο όνομα για το σκίτσο σας και, στη συνέχεια, κάντε κλικ στο OK.

Θυμηθείτε ότι ο στόχος μας είναι να κάνουμε τον λαμπτήρα LED να αναβοσβήνει επανειλημμένα. Για τον σκοπό λοιπόν αυτό, θα πρέπει να δημιουργήσουμε μια συνάρτηση με βρόχους για να ρυθμίσουμε το Arduino να εκτελέσει μια εντολή επανειλημμένα, μέχρι αυτό να απενεργοποιηθεί ή μέχρι κάποιος να πιέσει τη σήμανση RESET για επαναφορά της πλατφόρμας.

- Εισαγάγετε τον κώδικα που εμφανίζεται με έντονους χαρακτήρες μετά το Void Setup() στην ακόλουθη καταχώρηση για να δημιουργήσετε μια συνάρτηση άδειου βρόχου (empty loop).
- Ολοκληρώστε την έκφραση προσθέτοντας τις παρενθέσεις (}) και, στη συνέχεια, αποθηκεύστε ξανά το σκίτσο σας.

```
void setup() {
{
pinMode(13, OUTPUT); // ορίζει την ψηφιακή ακίδα pin 13 ως έξοδο
}
void loop() {
{
```

// τοποθετήστε τον κώδικα του κύριου βρόχου εδώ:

```
}
```

- Στη συνέχεια, εισάγετε τις σχετικές συναρτήσεις στο void loop() για να τις εκτελέσει το Arduino.
- Εισαγάγετε τα ακόλουθα μεταξύ των παρενθέσεων της συνάρτησης βρόχου και, στη συνέχεια, κάντε κλικ στην επιλογή *Verify* για να βεβαιωθείτε ότι έχετε εισάγει τα πάντα σωστά:

```
digitalWrite(13, HIGH); // ενεργοποεί την ψηφιακή ακίδα pin 13
delay(1000); // κάνει παύση για ένα δευτερόλεπτο
digitalWrite(13, LOW); // απενεργοποιεί την ψηφιακή ακίδα pin 13
delay(1000); // κάνει παύση για ένα δευτερόλεπτο
```

Η **συνάρτηση digitalWrite()** ελέγχει την τάση τροφοδοσίας και την παράγει ως έξοδο από την ψηφιακή ακίδα 13 με το άναμμα του λαμπτήρα LED. Όταν ορίζετε τη δεύτερη παράμετρο της συνάρτησης σε **HIGH**, τότε παράγεται ως έξοδος μια «υψηλή» ψηφιακή

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

τάση. Στη συνέχεια, ηλεκτρικό ρεύμα θα εισρεύσει από την ακίδα, προκαλώντας έτσι τον Λαμπτήρα LED να ανάψει.

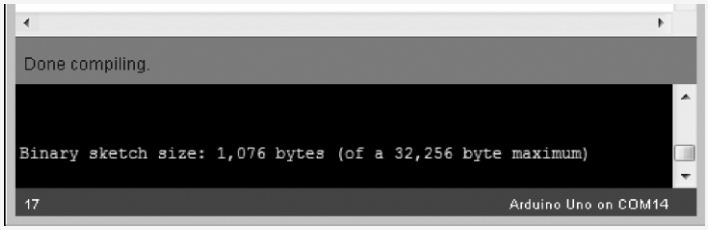
Όσο ο λαμπτήρας LED είναι ενεργοποιημένος, τότε το φως διακόπτεται για 1 δευτερόλεπτο με τη συνάρτηση delay(1000). Η συνάρτηση **delay()** ρυθμίζει το σκίτσο να μην κάνει τίποτα για ένα χρονικό διάστημα - που στην προκειμένη περίπτωση αναλογεί με 1000 χιλιοστά του δευτερολέπτου, δηλαδή 1 δευτερόλεπτο.

- Στη συνέχεια, απενεργοποιούμε την τάση στον λαμπτήρα LED με το DigitalWrite(13, LOW).
- Τέλος, κάνουμε ξανά μια παύση για 1 δευτερόλεπτο όσο η λαμπτήρας LED είναι απενεργοποιημένος, με καθυστέρηση 1 δευτερολέπτου μέσω της συνάρτησης delay(1000).

Το ολοκληρωμένο σκίτσο πρέπει να έχει την εξής μορφή:

```
void setup() {
{
pinMode(13, OUTPUT); // ορίζει την ψηφιακή ακίδα pin 13 ως έξοδο
}
void loop() {
{
digitalWrite(13, HIGH); // ενεργοποεί την ψηφιακή ακίδα pin 13
delay(1000); // κάνει παύση για ένα δευτερόλεπτο
digitalWrite(13, LOW); // απενεργοποιεί την ψηφιακή ακίδα pin 13
delay(1000); // κάνει παύση για ένα δευτερόλεπτο
}
```

- Αποθηκεύστε το σκίτσο σας.
- Κάνετε επαλήθευση του σκίτσου σας για να βεβαιωθείτε ότι ο κώδικας έχει γραφτεί σωστά, για να μπορεί να τον διαβάζει και να τον εκτελεί το Arduino.
- Μόλις επαληθευτεί το σκίτσο, θα εμφανιστεί μια σημείωση στο παράθυρο μηνυμάτων, όπως φαίνεται πιο κάτω:



Εικόνα 32 – Επαλήθευση σκίτσου στο παράδειγμα με τον Λαμπτήρα Led που αναβοσβήνει

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

- Βεβαιωθείτε ότι η πλακέτα του Arduino είναι συνδεδεμένη και κάντε κλικ στη σήμανση Μεταφόρτωση (upload) στο IDE. Το IDE μπορεί να επαληθεύσει ξανά το σκίτσο σας και να το μεταφορτώσει στην πλακέτα του Arduino.

Οι λαμπτήρες LED TX/RX της πλακέτας θα πρέπει να αναβοσβήνουν κατά τη διάρκεια αυτής της διαδικασίας, υποδεικνύοντας έτσι ότι το πρόγραμμά σας λειτουργεί σωστά.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

3. Εφαρμογές

⇒ **Αριθμός συμμετεχόντων:** 1-10 ανά συντονιστή
⇒ **Διάρκεια:** 5.5 ώρες
⇒ **Μέθοδοι διδασκαλίας:** Παρουσίαση, Καθοδηγούμενη διδασκαλία, Βιωματική εκπαίδευση
⇒ **Απαιτούμενα υλικά:** Παρουσίαση, Arduino IDE και Πλακέτες, υπολογιστής (1 ανά συμμετέχοντα), σταθερή σύνδεση στο διαδίκτυο και σχετικά υλικά ανάλογα με την εφαρμογή Arduino που θέλετε να δημιουργήσετε (π.χ. paintbot)

3.1. Τι είναι η ρομποτική;

Η ρομποτική είναι ένας τομέας όπου η επιστήμη, η μηχανική και η τεχνολογία διασταυρώνονται. Στόχος της ρομποτικής είναι η παραγωγή μηχανών, που ονομάζονται ρομπότ, ικανών να υποκαθιστούν, να αντιγράφουν ή να υποβοηθούν ανθρώπινες ενέργειες. Αρχικά, τα ρομπότ κατασκευάστηκαν για να διαχειρίζονται μονότονες εργασίες, ειδικά στον τομέα της αυτοματοποίησης της παραγωγικής διαδικασίας στην βιομηχανία. Από τότε όμως που δημιουργήθηκαν, η χρήση τους έχει διευρυνθεί σε ποικίλους τομείς πέρα από τις αρχικές τους λειτουργίες. Στις μέρες μας, υπάρχουν διάφοροι τύποι ρομπότ οι οποίοι χρησιμοποιούνται για την εκτέλεση ενός ευρύ φάσματος εργασιών για διάφορους σκοπούς, όπως π.χ. για πυρόσβεση, οικιακή καθαριότητα και ιατρικό εξοπλισμό σε περιπτώσεις όπου οι γιατροί πρέπει να πραγματοποιήσουν εξαιρετικά περίπλοκες χειρουργικές επεμβάσεις. Το επίπεδο αυτονομίας ενός ρομπότ μπορεί να διαφέρει εάν αυτό προορίζεται για την εκτέλεση εργασιών που υποκαθιστούν την ανθρώπινη δραστηριότητα και στα οποία ο άνθρωπος έχει τον πλήρη έλεγχο, ενώ υπάρχουν επίσης ρομπότ που είναι πλήρως αυτόνομα και τα οποία εκτελούν εργασίες χωρίς την ανάγκη ανθρώπινης εμπλοκής.

Αν και ο κλάδος της ρομποτικής εξελίσσεται διαρκώς, μπορούμε ωστόσο να διακρίνουμε κάποια κοινά χαρακτηριστικά στους υφιστάμενους τύπους ρομπότ:

1. «Η κατασκευή ρομπότ απαιτεί ένα είδος μηχανολογικού σχεδιασμού. Εκείνο που βοηθά ένα ρομπότ να ολοκληρώσει τις εργασίες στο περιβάλλον για το οποίο έχει σχεδιαστεί είναι ο έλεγχος της λειτουργίας των διάφορων μηχανικών εξαρτημάτων που το απαρτίζουν» (Built In, 2022, §3).
2. «Τα ρομπότ χρειάζονται αυτόν το μηχανικό εξοπλισμό για να μπορούν να ελέγχουν και να τροφοδοτούν ένα λειτουργικό σύστημα» (Built In, 2022,§3).
3. «Τα ρομπότ στο σύνολό τους περιέχουν τουλάχιστον κάποιο επίπεδο προγραμματισμού ηλεκτρονικών υπολογιστών. Χωρίς προγραμματισμό ένα ρομπότ δε θα μπορούσε να εκτελέσει εντολές και οδηγίες, και έτσι θα ήταν ακόμα ένα άλλο απλό μηχάνημα. Η εισαγωγή ενός προγράμματος σε ένα ρομπότ του δίνει τη δυνατότητα να γνωρίζει πότε και πώς να εκτελέσει μια εργασία» (Built In, 2022, §3).

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

3.2 Τύποι ρομπότ

Στο σύγχρονο κόσμο, υπάρχουν διάφοροι τύποι ρομπότ οι οποίοι μπορούν να βρουν εφαρμογές σε διάφορους κλάδους. Καθώς η τεχνολογία εξελίσσεται, τα ρομπότ γίνονται ολοένα και πιο σημαντικά καθώς μπορούν να διαδραματίσουν ένα καθοριστικό ρόλο στην καθημερινή μας ζωή. Υπάρχουν πολλοί και διάφοροι τύποι ρομπότ, από μικροσκοπικά ρομπότ μέχρι τεράστιου μεγέθους ρομπότ που χρησιμοποιούνται στον κλάδο της βιομηχανίας, ρομπότ που αναλαμβάνουν το ρόλο θυρωρού ή ακόμη και ρομπότ για ψυχαγωγικούς σκοπούς.
Με βάση τον τύπο και τις εφαρμογές τους, τα ρομπότ μπορούν να κατηγοριοποιηθούν με πολλούς διαφορετικούς τρόπους. Κάθε ρομπότ έχει τα δικά του μοναδικά χαρακτηριστικά και μπορεί να ποικίλει σημαντικά ως προς το μέγεθος, το σχήμα και τις δυνατότητες που διαθέτει. Ωστόσο, πολλά ρομπότ εμφανίζουν ένα σύνολο κοινών χαρακτηριστικών μεταξύ τους. Πιο κάτω μπορείτε να δείτε τις 13 κατηγορίας στις οποίες μπορούμε να ταξινομήσουμε τα ρομπότ με βάση τον τύπο τους και τις εφαρμογές τους.

Βιομηχανικά ρομπότ

Στην ρομποτική βιομηχανία, μπορούμε να βρούμε κυρίως έξι τύπους ρομπότ: τα αρθρωτά ρομπότ, τα καρτεσιανά ή γραμμικά ρομπότ, τα ρομπότ SCARA, τα κυλινδρικά ρομπότ, τα παράλληλα ρομπότ ή τα ρομπότ δέλτα και, τέλος, τα πολικά ή σφαιρικά ρομπότ.

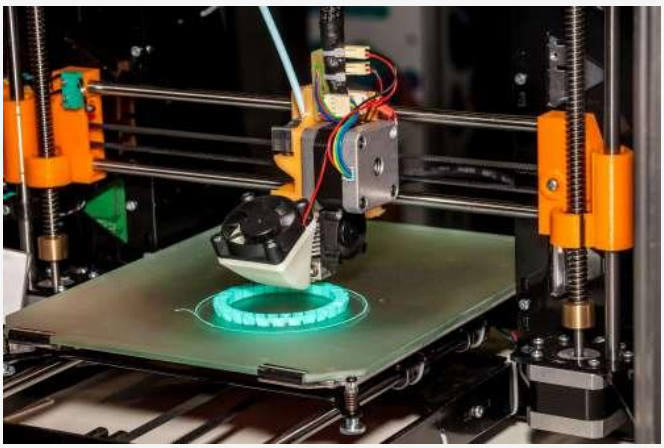
- Το αρθρωτό ρομπότ: Μοιάζει με ανθρώπινο βραχίονα στη διάταξη των ηλεκτρομηχανικών του εξαρτημάτων. Ο ρομποτικός βραχίονας συνδέεται με στροφικές αρθρώσεις, οι οποίες ονομάζονται και «άξονες». Ο αριθμός των στροφικών αρθρώσεων με τους οποίους συνδέεται ο βραχίονας κυμαίνονται από τους δύο μέχρι τους δέκα, επιτρέποντας έτσι αρκετούς βαθμούς ελευθερίας (Analytics Insight, 2021).

Εικόνα 33 – Αρθρωτό ρομπότ
Πηγή: https://diy-robotics.com/article/articulated-robots/

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

- Καρτεσιανά ρομπότ: Γνωστά επίσης και ως γραμμικά ρομπότ ή gantry ρομπότ, τα καρτεσιανά ρομπότ διαθέτουν τρεις γραμμικές αρθρώσεις που χρησιμοποιούν το σύστημα των καρτεσιανών συντεταγμένων (X, Y και Z). Μπορεί επίσης να συνδέονται με έναν καρπό, ο οποίος επιτρέπει την περιστρεφόμενή τους κίνηση. Οι τρεις πρισματικές αρθρώσεις παρέχουν μια γραμμική κίνηση κατά μήκος του άξονα (Analytics Insight, 2021).



Εικόνα 34 – Καρτεσιανό ρομπότ
Πηγή: https://diy-robotics.com/article/what-you-should-know-about-cartesian-robot/

- Ρομπότ SCARA (Selective Compliance Articulated Robot or Arm): Χρησιμοποιούνται συχνά για σκοπούς συναρμολόγησης σε όλο τον κόσμο, λόγω της εύκολής τους χρήσης και εγκατάστασης (Analytics Insight, 2021).

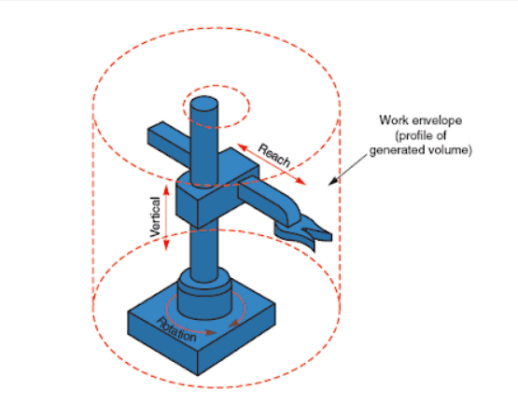
Εικόνα 35 – Βραχίονας ρομπότ SCARA
Πηγή: https://diy-robotics.com/article/articulated-robots/

- Κυλινδρικά ρομπότ: Αυτά τα ρομπότ χρησιμοποιούνται γενικά για σκοπούς συναρμολόγησης, επικάλυψης, χειρισμού μηχανών, συγκόλλησης κατά σημεία και χώνευσης με έγχυση. Διαθέτουν τουλάχιστον μια περιστροφική άρθρωση στη βάση τους και μια πρισματική άρθρωση για την ένωση των συνδέσμων μεταξύ τους. Το κυλινδρικό ρομπότ έχει περιστροφική άρθρωση για περιστροφή και πρισματικό σύνδεσμο για γωνιακή κίνηση γύρω από τον άξονα άρθρωσης. Η περιστροφική

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

άρθρωση κινείται σε περιστροφική κίνηση γύρω από τον άξονα. Αντίθετα, η πρισματική άρθρωση κινείται σε γραμμική κίνηση (Analytics Insight, 2021).



Εικόνα 36 – Κυλινδρικό Ρομπότ
Πηγή: https://learnmech.com/cylindrical-robot-diagram-construction-applications/

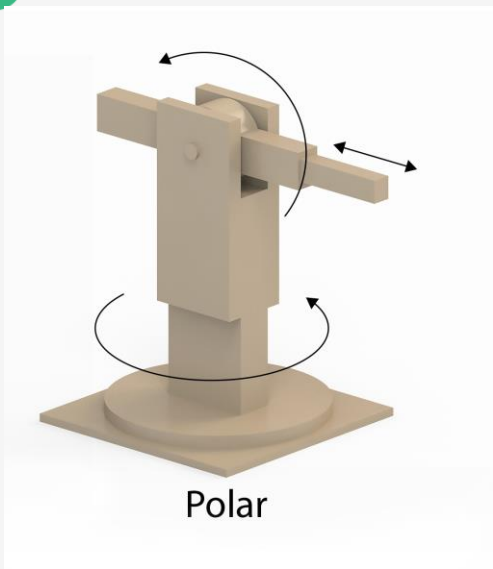
- Ρομπότ δέλτα ή παράλληλα ρομπότ: Γνωστά και ως «ρομπότ αράχνες» διαθέτουν τρεις κινητήρες στη βάση τους για την ενεργοποίηση των βραχιόνων ελέγχου οι οποίοι είναι υπεύθυνοι για την τοποθέτηση των αρθρώσεων. Ο βασικός τύπος ρομπότ δέλτα διαθέτει 3 βραχίονες (άξονες περιστροφής) όμως υπάρχουν και ρομπότ του ίδιου τύπου που διαθέτουν από 4 μέχρι και 6 βραχίονες (Analytics Insight, 2021).

Εικόνα 37 – Ρομπότ δέλτα
Πηγή: https://diy-robotics.com/article/top-six-types-industrial-robots-2020/

- Πολικά ρομπότ: Γνωστά και ως σφαιρικά ρομπότ, έχουν ως βάση το κέντρο μιας σφαίρας και ο βραχίονάς τους εκτείνεται προς όλες τις συντεταγμένες της σφαίρας, περιστρεφόμενος γύρω από ένα ή περισσότερους άξονες. (Analytics Insight, 2021).

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης



Εικόνα 38 – Πολικά ρομπότ
Πηγή: https://www.howtorobot.com/expert-insight/industrial-robot-types-and-their-different-uses

Εκτός από τα βιομηχανικά ρομπότ, υπάρχουν απεριόριστα είδη ρομπότ ανά τομέα, όπως στην εκπαίδευση, την ασφάλεια, τη διαστημική επιστήμη, την ιατρική κτλ. Εδώ μπορείτε να βρείτε μερικά παραδείγματα αυτών των ρομπότ, αλλά να θυμάστε ότι μπορείτε να βρείτε πολλά περισσότερα.

Ρομπότ για οικιακή χρήση

Αναφερόμενα και ώς ρομπότ για καταναλωτές, είναι διαθέσιμα στην αγορά και μπορούν να χρησιμοποιηθούν για ψυχαγωγικούς σκοπούς ή για την εκτέλεση εργασιών ή αγγαρειών. Τα εν λόγω ρομπότ χρησιμοποιούνται για την εκτέλεση οικιακών εργασιών και περιλαμβάνουν εξοπλισμό για τον καθαρισμό πισίνας, ηλεκτρικές σκούπες, εξοπλισμό για τον καθαρισμό υδρορροών, ρομποτικούς βοηθούς που λειτουργούν με τεχνητή νοημοσύνη, καθώς και μια διευρυνόμενη κατηγορία από ρομποτικά παιχνίδια και ρομποτικά κιτ (kit). Αυτή η κατηγορία ρομπότ, γνωστή και ως κοινωνική ρομποτική, μπορεί να περιλαμβάνει ρομπότ για ψυχαγωγικούς σκοπούς που έχουν σχεδιαστεί για να διεγείρουν τα ανθρώπινα συναισθήματα.

Στρατιωτικά ρομπότ και ρομπότ διάσωσης

Τα ρομπότ χρησιμοποιούνται επίσης στον στρατό ή για την αντιμετώπιση καταστάσεων εκτάκτου ανάγκης, όπου μια κατάσταση μπορεί να είναι πολύ επικίνδυνη για τον άνθρωπο. Τα μη επανδρωμένα τηλεκατευθυνόμενα αεροσκάφη (drones), οι ανιχνευτές ναρκών και οι συσκευές ανίχνευσης ευρύτερα, βρίσκουν κοινή χρήση στο πεδίο της μάχης αλλά απαιτούν άμεσο ανθρώπινο έλεγχο. Τα τηλεκατευθυνόμενα οχήματα αποτρέπουν την απώλεια ανθρώπινων ζωών που θα συνέβαινε εάν οι στρατιώτες βρίσκονταν στο πεδίο της μάχης αντί πίσω από την ασφάλεια που παρέχουν τα τηλεκατευθυνόμενα οχήματα (Stanford Edu, 2022). Ωστόσο, η χρήση της ρομποτικής και της τεχνητής νοημοσύνης σε στρατιωτικές

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

επιχειρήσεις είναι αρκετά αμφιλεγόμενη αφού εγείρει προβληματισμούς και ανησυχίες σε θέματα ηθικής. Επιπλέον, αυτοί οι τύποι ρομπότ είναι σε θέση να αντέχουν σε ακραίες καιρικές συνθήκες και να διασχίζουν δύσκολα πεδία. Αυτά τα ρομπότ εκτελούν επικίνδυνες εργασίες, όπως π.χ. η αναζήτηση επιζώντων μετά από μια κατάσταση έκτακτης ανάγκης και η παροχή βοήθειας σε άλλες περιπτώσεις ζωτικής σημασίας στις περιοχές πρόκλησης των καταστροφών (Analytics Insight, 2021).

Ρομποτική Ιατρική

Η ανάπτυξη της ρομποτικής είχε ένα τεράστιο αντίκτυπο στον τομέα της ιατρικής. Η ρομποτική εφαρμόστηκε για πρώτη φορά στην ιατρική πριν από 35 περίπου χρόνια, με στόχο την εισαγωγή ενός ανιχνευτή στον εγκέφαλο για τη λήψη δείγματος βιοψίας. «Σήμερα, η ρομποτική χρησιμοποιείται ευρέως στη χειρουργική, που είναι γνωστή και ως ρομποτική χειρουργική, στην οποία εφαρμόζονται οι σύγχρονες τεχνολογίες των ρομπότ, των υπολογιστών και των λογισμικών για τον χειρισμό χειρουργικών εργαλείων με μεγάλη ακρίβεια, μέσω μιας ή περισσότερων μικρών τομών, για διάφορες χειρουργικές επεμβάσεις. Μια τρισδιάστατη μεγεθυμένη εικόνα υψηλής ανάλυσης στην κονσόλα του χειρουργού επιτρέπει χειρουργικές επεμβάσεις με υψηλή ακρίβεια και έλεγχο» (NCBI, 2019). Μια από τις πιο διάσημες εφαρμογές της ρομποτικής στην ιατρική είναι εκείνη που κατασκευάστηκε από τον da Vinci και η οποία εγκρίθηκε από τον FDA το 2000. Υπολογίζεται ότι μέχρι σήμαρα έχει χρησιμοποιηθεί για την εκτέλεση πάνω από 6 εκατομμυρίων χειρουργικών επεμβάσεων παγκοσμίως (NCBI, 2019).

Υπηρεσιακά Ρομπότ

«Ο Διεθνής Οργανισμός Τυποποίησης ορίζει το τύπο του «υπηρεσιακού ρομπότ» ως ένα ρομπότ «που εκτελεί χρήσιμες για τον άνθρωπο εργασίες ή ως εξοπλιστικές μηχανές, εξαιρουμένων των εφαρμογών για την αυτοματοποίηση της βιομηχανικής παραγωγής» (IFR, 2022). Οι πιο συνηθισμένοι τύποι υπηρεσιακών ρομπότ είναι τα ρομπότ που αναλαμβάνουν τα καθήκοντα ενός θυρωρού ή ενός υπαλλήλου υποδοχής ή εξυπηρέτησης πελατών. Η Guardforce Hong Kong, για παράδειγμα, έχει εγκαινιάσει ένα ρομπότ θυρωρού που έχει τη δυνατότητα να αυτοματοποιεί την εγγραφή των επισκεπτών, να ελέγχει τα σημεία πρόσβασης μέσω ενός συστήματος αναγνώρισης προσώπου, καθώς επίσης και τη δυνατότητα να προσφέρει πληροφορίες πολυμέσων, καθιστώντας το ιδανικό για εμπορικά κτίρια, εκδηλώσεις και εκθέσεις. Η ίδια εταιρεία κατασκεύασε ένα ρομπότ που εκτελεί καθήκοντα υποδοχής και εξυπηρέτησης πελατών σε εμπορικά κέντρα, το οποίο προσφέρει τη δυνατότητα παροχής οδηγών αγοράς, χειραψίας και συνομιλίας με τους επισκέπτες, καθώς και τη δυνατότητα παροχής κουπονιών (Guardforce, 2022).

Τα συνεργατικά ρομπότ (cobot)

Τα συνεργατικά ρομπότ ή τα «cobots» είναι άνθρωποι που συνεργάζονται με μηχανές σε ένα κοινό περιβάλλον για την εκτέλεση των καθηκόντων τους. Μερικά παραδείγματα αυτής της

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

κατηγορίας είναι ο ρομποτικός βραχίονας Sawyer ο οποίος βοηθά τους εργαζομένους σε ένα θερμοκήπιο να συλλέγουν τα φυτά. Ένα από τα ρομπότ που σχεδίασε η Mitsubishi παρέχει τη δυνατότητα παροχής καφέ στο περίπτερο Café X στο Χονγκ Κονγκ.

Drones (μη επανδρωμένα τηλεκατευθυνόμενα αεροσκάφη)

Τα drones είναι στην πραγματικότητα αεροσκάφη χωρίς ανθρώπινο πλοηγό κατά την πτήση. Τα μη επανδρωμένα αεροσκάφη διατίθενται σε διάφορα μεγέθη και μπορεί να έχουν διαφορετικό βαθμό αυτονομίας μεταξύ τους. Αυτός ο τύπος ρομπότ έχει εισβάλει σε ένα ευρύ φάσμα τομέων παγκοσμίως, συμβάλλοντας στην εκτέλεση πολλών διαφορετικών εργασιών όπως π.χ. για τον καθαρισμό της ατμόσφαιρας, την αεροφωτογράφιση και τις ταχυμεταφορές.

Ανθρωποειδή ρομπότ

Αυτό είναι πιθανώς το είδος του ρομπότ που οι περισσότεροι άνθρωποι σκέφτονται πρώτα όταν φέρνουν στη σκέψη τους ένα ρομπότ. Τα ρομπότ αυτού του τύπου μοιάζουν με άνθρωπο και μπορούν επίσης να μιμηθούν την ανθρώπινη συμπεριφορά. Συνήθως εκτελούν ανθρώπινες δραστηριότητες (όπως τρέξιμο, άλματα και μεταφορά αντικειμένων) και μερικές φορές έχουν σχεδιαστεί για να μοιάζουν με ανθρώπους, διαθέτοντας ακόμη ανθρώπινα πρόσωπα και εκφράσεις (Built In, 2022).

Τα ρομπότ για εκπαιδευτικούς σκοπούς

Η ρομποτική στην εκπαίδευση είναι ένας αναδυόμενος κλάδος ο οποίος έχει σχεδιαστεί για να εισάγει τους μαθητές στον κόσμο της ρομποτικής και στον προγραμματισμό μέσω διαδραστικών τρόπων από πολύ μικρή ακόμα ηλικία. Τα εισαγωγικά μαθήματα ρομποτικής στην εκπαίδευση παρέχουν στους μαθητές όλα όσα χρειάζονται για να κατασκευάσουν και να προγραμματίσουν εύκολα ένα ρομπότ ικανό να εκτελέσει διάφορες εργασίες, ενώ τα μαθήματα και το επίπεδο δυσκολίας τους προσαρμόζονται πάντα στην ηλικία των εκάστοτε μαθητών (Iberdrola, 2022).

3.3 Οδήγηση κινητήρα συνεχούς ρεύματος (DC motor) με ασπίδα κινητήρα (shield motor)

Συνήθως, τα ρομπότ και οι αυτοματοποιημένες συσκευές περιέχουν κινούμενα μέρη. Η κίνηση ενεργοποιείται από κινητήρες που είναι προγραμματισμένοι να λειτουργούν με συγκεκριμένο τρόπο. Στην υποενότητα αυτή, θα εξερευνήσουμε πώς μπορούμε να προγραμματίσουμε έναν τυπικό κινητήρα συνεχούς ρεύματος, δηλαδή έναν κινητήρα που μπορεί να περιστρέφεται δεξιόστροφα και αριστερόστροφα. Οι κινητήρες συνεχούς ρεύματος χρησιμοποιούνται ευρέως σε παιχνίδια, καθώς και σε ηλεκτρικές συσκευές, όπως το μίξερ και άλλες μικροσυσκευές κουζίνας.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

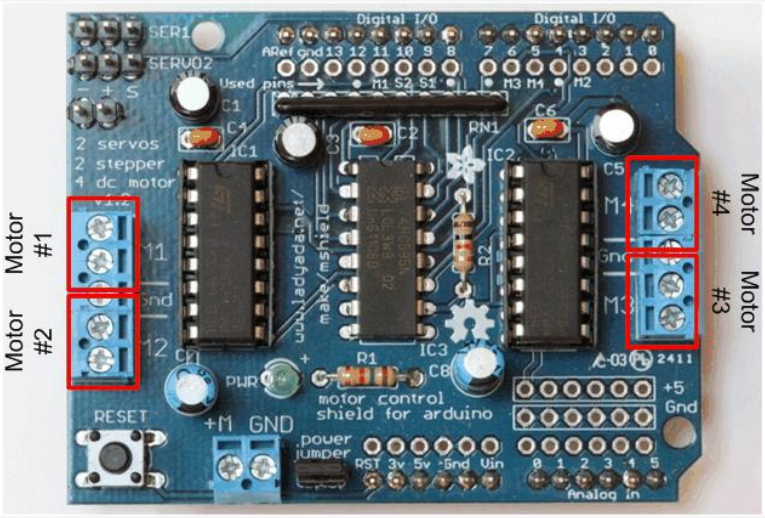
Η λειτουργία ενός κινητήρα συνεχούς ρεύματος καθίσταται δυνατή μέσω μιας προγραμματιζόμενης πλακέτας (π.χ. Arduino), ενώ απαιτεί ένα μικρό αριθμό εξαρτημάτων. Ωστόσο, για λόγους απλοποίησης, είναι γενικά αποδεκτό ότι ένας κινητήρας συνεχούς ρεύματος λειτουργεί καλύτερα όταν ενώνεται με μια ασπίδα κινητήρα.

Υπάρχουν δύο κύριες μέθοδοι για να συνδέσετε έναν κινητήρα συνεχούς ρεύματος σε μια πλακέτα Arduino. Το πρώτο αφορά τη χρήση ενός Mosfet και μιας διόδου, ενώ το δεύτερο βασίζεται σε ένα ηλεκτρονικό εξάρτημα που ονομάζεται ασπίδα κινητήρα.

Μια ασπίδα κινητήρα μας επιτρέπει να παρακάμψουμε έναν αριθμό περίπλοκων καλωδιώσεων που είναι απαραίτητες για τη λειτουργία ενός ή περισσότερων κινητήρων συνεχούς ρεύματος.

Υπάρχουν πολλοί τύποι ασπίδα κινητήρα. Για την ακόλουθη δραστηριότητα, θα χρειαστείτε την ασπίδα κινητήρα τύπου Adafruit V1. Προσέξτε να επιλέξετε την έκδοση 1 και όχι την έκδοση 2 του Adafruit, καθώς ο προγραμματισμός του ενός δεν είναι συμβατός με τον άλλο.

Μια ασπίδα κινητήρα τύπου Adafruit V1 περιέχει δύο τσιπ L293D. Για το λόγο αυτό, μπορεί να οδηγήσει μέχρι και 4 κινητήρες συνεχούς ρεύματος ταυτόχρονα.

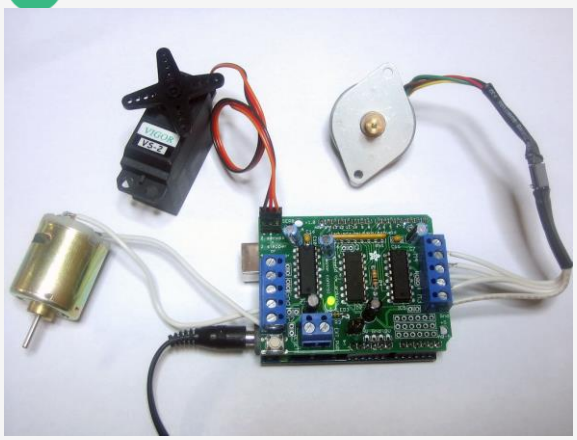


Εικόνα 39 – Ασπίδα κινητήρα τύπου Adafruit V1
Πηγή: https://learn.adafruit.com/adafruit-motor-shield/af-dcmotor-class

Εκτός από τους κινητήρες συνεχούς ρεύματος, είναι δυνατό να οδηγήσετε ηλεκτροκινητήρες SERVO (σερβοκινητήρες) και κινητήρες κλιμακωτής περιστροφικής κίνησης με ασπίδα κινητήρα.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης



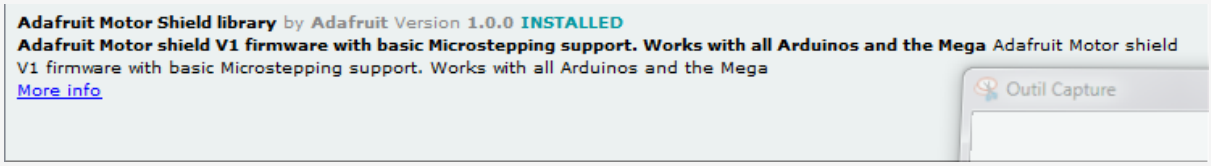
Εικόνα 39 – Ασπίδα κινητήρα τύπου Adafruit V1

Πηγή: https://learn.adafruit.com/adafruit-motor-shield/af-dcmotor-class

Μπορείτε να ενώσετε έναν κινητήρα συνεχούς ρεύματος στην ασπίδα κινητήρα χρησιμοποιώντας καλώδια τύπου M1, M2, M3 ή M4.

Αφού ολοκληρώσετε τη συναρμολόγηση του συστήματος, μπορείτε να ξεκινήσετε με τον προγραμματισμό.
Για τον προγραμματισμό του κινητήρα συνεχούς ρεύματος θα πρέπει πρώτα να εγκαταστήσουμε τη βιβλιοθήκη της ασπίδας κινητήρα Adafruit που είναι συνδεδεμένη με το κινητήρα DC. Για να βρείτε την βιβλιοθήκη του Adafruit, θα πρέπει να την αναζητήσετε την επιλογή “AF Motor” από την λίστα βιβλιοθηκών.

Κάντε κλικ στο μενού "Sketch" και, στη συνέχεια, επιλέξτε «Include Library > Manage Libraries».



Στο στάδιο αυτό, μπορείτε να ξεκινήσετε με τον προγραμματισμό της ασπίδας κινητήρα. Ένας απλός κώδικας που μπορεί να χρησιμοποιηθεί για την περιστροφή του κινητήρα DC είναι ο εξής:

```
#include <AFMotor.h>

AF_DCMotor motor1(1); // Ορίζουμε έναν κινητήρα συνδεδεμένο στο M1 στην ασπίδα κινητήρα

void setup()
{
```

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

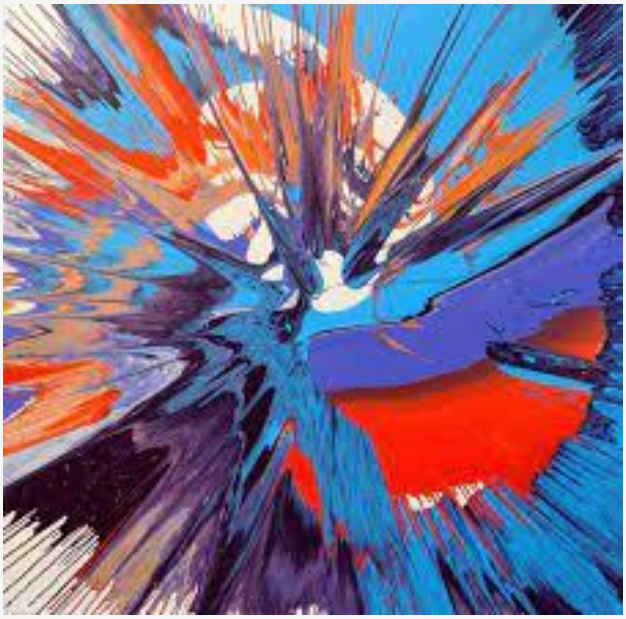
```
motor1.setSpeed(100); // Ορίζουμε την ταχύτητα με την οποία θέλουμε ο κινητήρας να περιστρέφεται

}
void loop()

{
motor1.run(BACKWARD); // ο κινητήρας θα περιστρέφεται είτε δεξιόστροφα είτε αριστερόστροφα, ανάλογα με τον τρόπο που τον συνδέσατε με την ασπίδα. Για να τον κάνετε να περιστρέφεται στην αντίθετη κατεύθυνση θα πρέπει να αντικαταστήσετε το BACKWARD με το FORWARD.

delay(10000); // ο κινητήρας θα περιστρέφεται για 10 δευτερόλεπτα

motor1.run(RELEASE); // ο κινητήρας θα σταματήσει να περιστρέφεται για 1 δευτερόλεπτο και έπειτα  θα επανεκκινήσει την περιστροφή του, ξεκινώντας και πάλι από την κορυφή της συνάρτησης του βρόχου

delay(1000);
 }
```

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

3.4. Η κατασκευή μιας μηχανής paintbot με τη χρήση κινητήρα DC και Arduino

Ο προγραμματισμός ενός κινητήρα DC μπορεί να εφαρμοστεί στο πλαίσιο μιας καλλιτεχνικής δημιουργίας προσδίδοντας διαδραστικότητα και δημιουργικότητα στην όλη διαδικασία. Η μηχανή paintbot παρέχει τη δυνατότητα συγχώνευσης των ηλεκτρονικών με τις τέχνες, χρησιμοποιώντας ως κύριο μέσο το χρώμα για να προσομοιώσει μια φωτογραφία σε μια ζωγραφιά.



Εικόνα 40 – Έργο τέχνης με τη χρήση μιας μηχανής paintbot
Πηγή: https://girlsinstem.eu/

Η PaintBot αποτελείται από μια περιστρεφόμενη πλάκα, η οποία μπορεί να περιστρέφεται σε διάφορες ταχύτητες. Στην περιστρεφόμενη πλάκα, μπορείτε να τοποθετήσετε ένα φύλλο χαρτί ή ένα μικρό καμβά. Αφού ένας χρήστης της paintbot τοποθετήσει ένα φύλλο χαρτί ή έναν καμβά στην περιστρεφόμενη πλάκα και θέσει τη μηχανή σε λειτουργία, αρχίζει να ρίχνει προσεκτικά σταγόνες χρώματος καθώς το φύλλο ή ο καμβάς περιστρέφεται, δημιουργώντας έτσι ένα μοναδικό έργο τέχνης που φιλοτεχνήθηκε τόσο με τη βοήθεια της PaintBot όσο και με την επιδεξιότητα του χρήστη.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Εικόνα 41 – Τα συστατικά ενός paintbot
Πηγή: https://girlsinstem.eu/

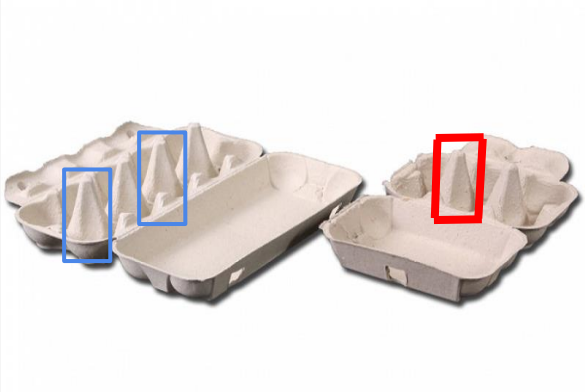
Η δημιουργία του πλαισίου-κουτιού

A. Προμηθευτείτε ένα κόκκινο και ένα μαύρο κομμάτι σύρμα, τουλάχιστον 30 cm το καθένα, και συνδέστε τα με τον κινητήρα. (Φροντίστε να μην κόψετε τα κομμάτια πολύ μικρά, καθώς αυτό ενδέχεται να περιπλέξει τη σύνδεση του κινητήρα στο PaintBot με το υπόλοιπο ηλεκτρονικό κύκλωμα. Όσο μεγαλύτερο είναι το κουτί σας, τόσο μακρύτερα θα πρέπει να είναι και τα καλώδια σας)
B. Μπορείτε να χρησιμοποιήσετε μια χαρτοθήκη αυγών ως βάση στήριξης του κινητήρα
    1. Αφαιρέστε το κάλυμμα του χαρτοκιβωτίου αυγών εφόσον δε θα χρειαστεί για την κατασκευή.
    2. Θα χρειαστείτε μόνο μια από τις κορυφές της χαρτοθήκης αυγών και τις 4 θήκες που το περιβάλλουν. Θα κόψετε δηλαδή μεταξύ των σημείων της 3ης/4ης θήκης και της παρακείμενης κορυφής.
    3. Στη συνέχεια, θα κόψετε την άκρη της κορυφής. Είναι προτιμότερο να αρχίσετε να κόβετε σε μικρά κομμάτια την χαρτοθήκη παρά σε μεγάλα, για να μπορέσετε να της δώσετε το επιθυμητό τελικό σχήμα στη συνέχεια.
    4. Τοποθετήστε τον κινητήρα από κάτω προς τα πάνω με τρόπο ώστε ο άξονάς του να βλέπει προς τα πάνω και τα καλώδια προς τα κάτω. Ο κινητήρας θα πρέπει να εφαρμόζει σφιχτά στην τρύπα που μόλις ανοίξατε. Εάν δεν εφαρμόζει σφιχτά, προσπαθήστε να τον στερεώσετε προσεκτικά μέσα στην τρύπα ή να κόψετε λίγο περισσότερο την κορυφή εάν δεν εφαρμόζει. Είναι σημαντικό ο κινητήρας να εφαρμόζει γερά στην τρύπα ούτως ώστε η όλη κατασκευή να είναι σταθερή, καθώς αυτή θα σχηματίζει τη βάση όπου θα περιστρέφεται η πλακέτα. Για να προσδώσετε ακόμα περισσότερη σταθερότητα και υποστήριξη στην κατασκευή, μπορείτε να χρησιμοποιήσετε ταινία ή μπορείτε να προσθέσετε οδοντογλυφίδες κάτω από τον κινητήρα και μέσα από την χαρτοθήκη.

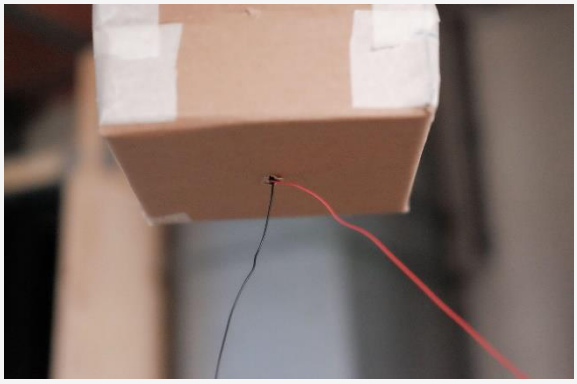
Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

C. Ανοίξτε μια μικρή τρύπα στη μέση της χαρτοθήκης. Τοποθετήστε τα καλώδια του κινητήρα μέσα από την τρύπα με τρόπο ώστε αυτά να εξέχουν από τα μέσα προς τα έξω, στη βάση της χαρτοθήκης. Τοποθετήστε την κορυφή της θήκης πάνω από τα καλώδια.
D. Βεβαιωθείτε ότι η κορυφή της αυγοθήκης είναι τοποθετημένη στη μέση του κουτιού και στερεώστε την σταθερά στο κουτί. Το βήμα αυτό είναι πολύ σημαντικό να γίνει σωστά καθώς η κορυφή θα πρέπει να αντέξει τη δύναμη του περιστρεφόμενου κινητήρα.
E. Τοποθετήστε ένα χάρτινο κομμάτι σε σχήμα δίσκου (στρογγυλό ή οποιοδήποτε άλλο σχήμα) πάνω από τον κόκκινο δίσκο σύνδεσης (τυπωμένο σε 3D μορφή) του κινητήρα. Βεβαιωθείτε ότι είναι μικρότερο από το κουτί, ώστε να μπορεί να περιστρέφεται ελεύθερα.



Εικόνα 42 – 2ο βήμα στην κατασκευή του paintbot



Εικόνα 43 – 3ο βήμα του paintbot

Εικόνα 44 – 4ο βήμα του paintbot

Η ένωση των ηλεκτρονικών εξαρτημάτων

Μετά τη δημιουργία του κουτιού, μπορείτε απλά να συνδέσετε τον κινητήρα DC που είναι συνδεδεμένος με την ασπίδα και να τον προγραμματίσετε με τέτοιο τρόπο ώστε να ρυθμίσετε την ταχύτητα και την κατεύθυνση με την οποία θα περιστρέφεται, προκειμένου να δημιουργήσετε το δικό σας έργο τέχνης!

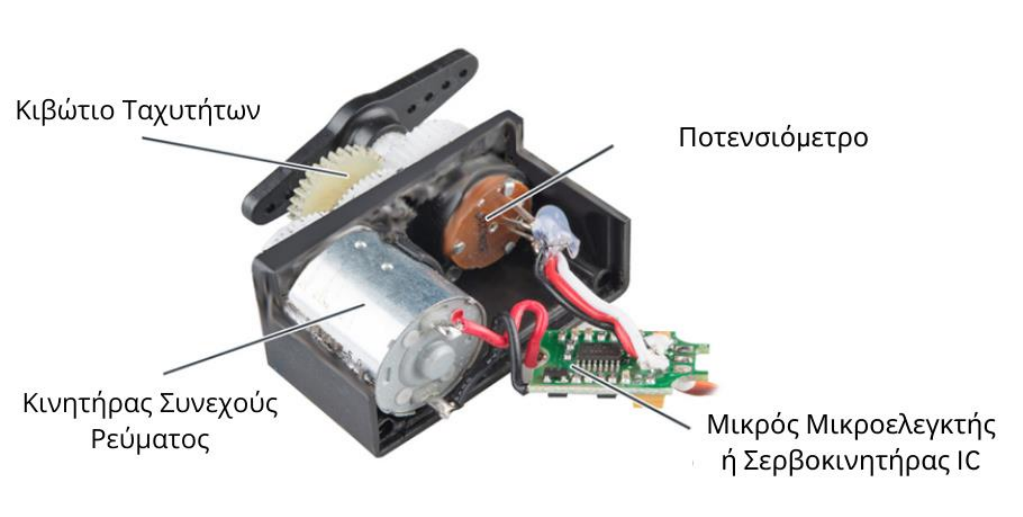
Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

3.5 Η κατασκευή ενός διαδραστικού παιχνιδιού από χαρτί με έναν σερβοκινητήρα

Ένας σερβοκινητήρας είναι ένας τύπος ηλεκτροκινητήρα που σε συνδυασμό με το πρόγραμμα οδήγησης που διαθέτει, συνθέτει μαζί ένα κλειστό κύκλωμα ανάδρασης.

Ένας τυπικός σερβοκινητήρας διαθέτει σχεδόν πάντα τρία βασικά εξαρτήματα: έναν κινητήρα συνεχούς ρεύματος, ένα κύκλωμα ελέγχου και ένα ποτενσιόμετρο ή ένα παρόμοιο μηχανισμό ανάδρασης. Ο κινητήρας συνεχούς ρεύματος συνδέεται με ένα κιβώτιο ταχυτήτων και έναν άξονα περιστροφής ως έξοδο, ο οποίος έχει τη δυνατότητα να αυξάνει την ταχύτητα και τη ροπή του κινητήρα. Ο κινητήρας συνεχούς ρεύματος κινεί τον άξονα περιστροφής. Κάθε σερβοκινητήρας περιλαμβάνει μία παλμογεννήτρια (Resolver/Encoder), η οποία μετατρέπει τη μηχανική κίνηση (δηλ. τις στροφές του άξονα) σε ψηφιακούς παλμούς. Αυτή τη μετατροπή ερμηνεύει στη συνέχεια ο ρυθμιστής στροφών – Inverter και σε συνδυασμό με το κατάλληλο πρόγραμμα οδήγησης που διαθέτει, συνθέτει μαζί με τον σερβοκινητήρα ένα κλειστό κύκλωμα ανάδρασης, το οποίο διέπει σε κάθε χρονική στιγμή τη θέση, τη ροπή και την ταχύτητα.



Εικόνα 45 - Σερβοκινητήρας
Πηγή: https://www.sparkfun.com/servos

Ένας σερβοκινητήρας μπορεί να έχει αμέτρητες εφαρμογές. Στην υποενότητα αυτή, θα σας υποδείξουμε πώς να χρησιμοποιείτε ένα σερβοκινητήρα για να δημιουργήσετε ένα διαδραστικό παιχνίδι από χαρτί, το κεφάλι του οποίου μπορείτε να στρίψετε αριστερά η δεξία με τη βοήθεια ενός ποτενσιόμετρου.

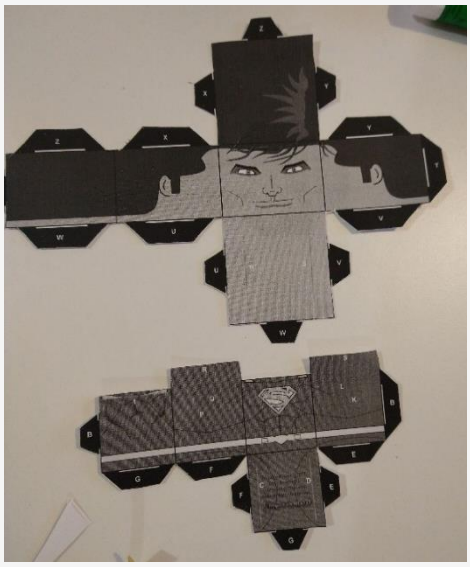
Για το ηλεκτρομηχανικό έργο αυτό δε θα χρειαστεί να γίνει κάποιος προγραμματισμός, ωστόσο, είναι δυνατό να επιτευχθούν τα ίδια ακριβώς αποτελέσματα εάν συνδέσετε έναν σερβοκινητήρα με μια πλακέτα Arduino.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

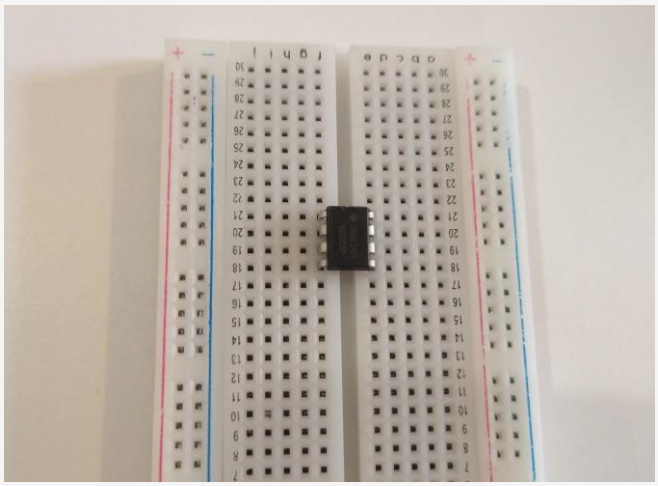
Βήμα 1: Κατασκευή ενός παιχνιδιού από χαρτί

Ξεκινούμε με την δημιουργία του χάρτινου μέρους του παιχνιδιού. Πρώτα απ' όλα, θα πρέπει να επιλέξετε ένα πλαίσιο το οποίο, στη συνέχεια, θα πρέπει να κόψετε και, στο τέλος, να συναρμολογήσετε τα διάφορα κομμάτια σύμφωνα με τις οδηγίες που δίνονται πιο κάτω, για να φτιάξετε το παιχνίδι.
Μπορείτε να διαλέξετε διάφορα είδη πλαισίων τα οποία είναι διαθέσιμα στο Cubeecraft.



Βήμα 2: Κατασκευή του ηλεκτρονικού κυκλώματος

Ξεκινάμε τοποθετώντας το τσιπ NE555 στη μέση του breadboard, ακριβώς όπως φαίνεται στην πιο κάτω εικόνα.

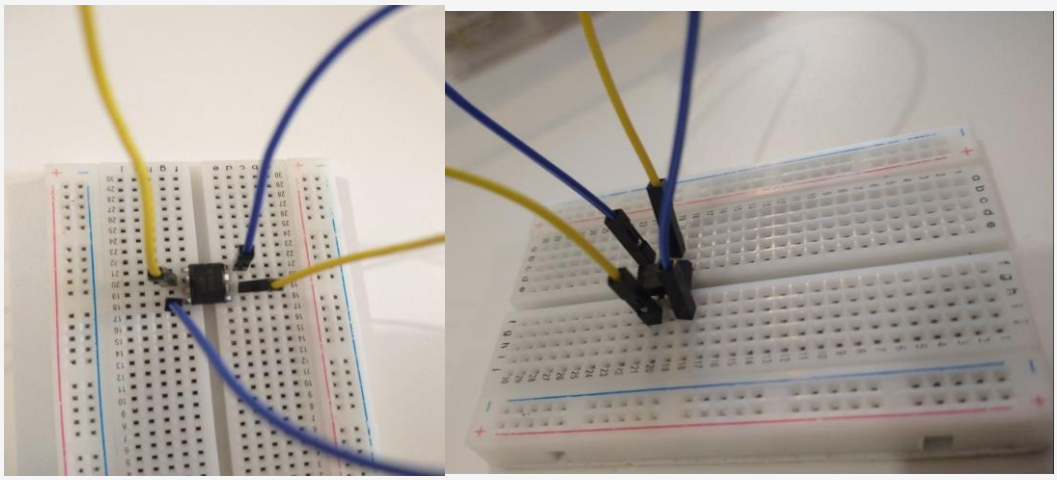


Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Εικόνα 46 – Χρονόμετρο- Διευθέτηση ακίδων
Πηγή: http://www.learningaboutelectronics.com/Articles/555-timer-pinout.php

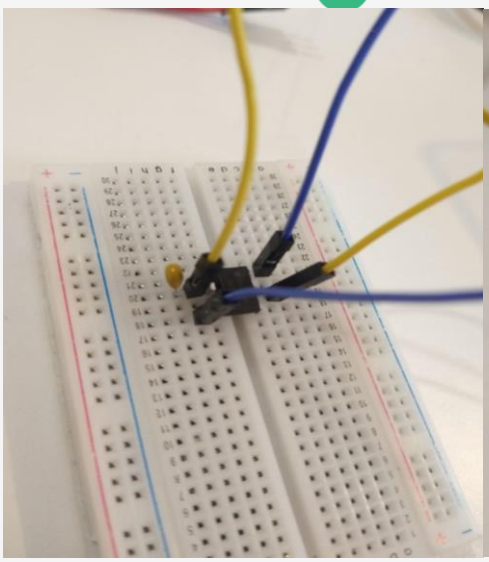
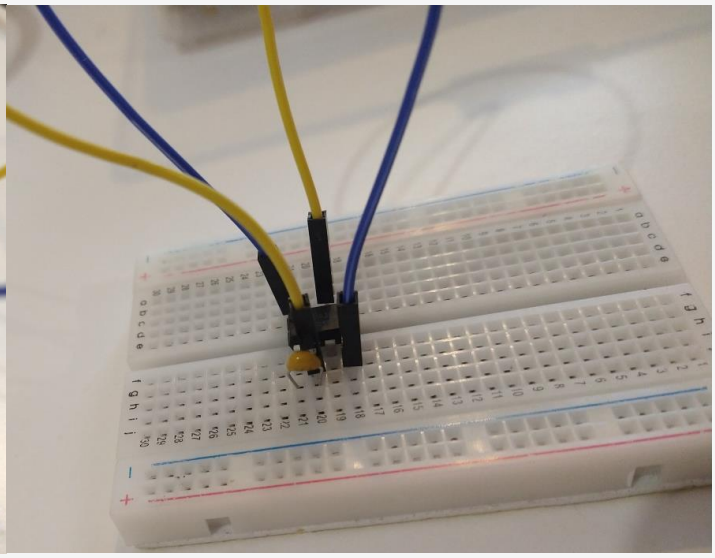
Στη συνέχεια, προσθέτουμε δύο καλώδια τύπου jumper, ένα που συνδέει το πόδι 4 και το πόδι 8 του τσιπ NE555 και ένα άλλο που συνδέει τα πόδια 2 και 6.



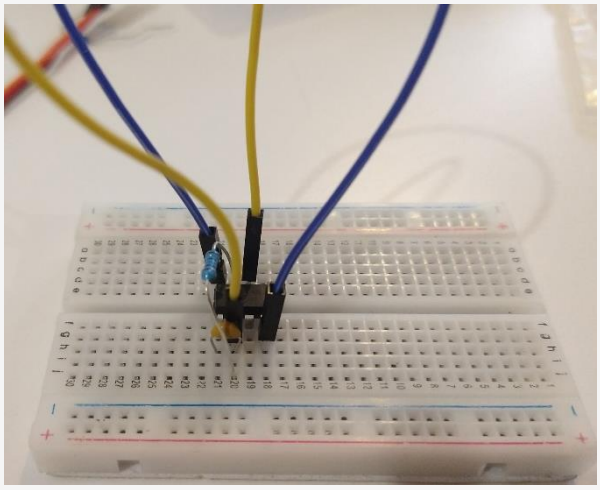
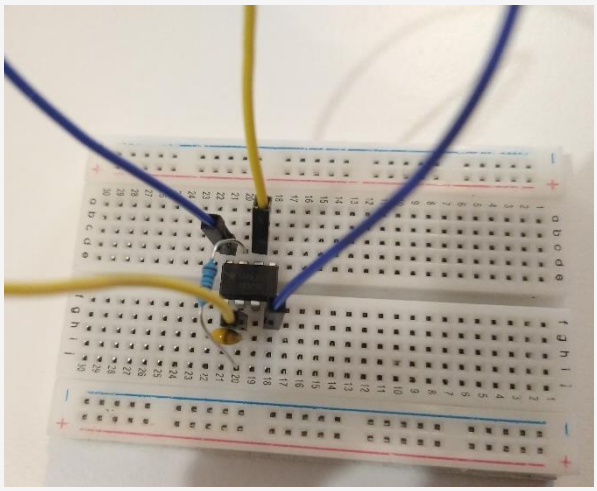
Στη συνέχεια, προσθέτουμε έναν πυκνωτή που συνδέει τα πόδια 1 και 2.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

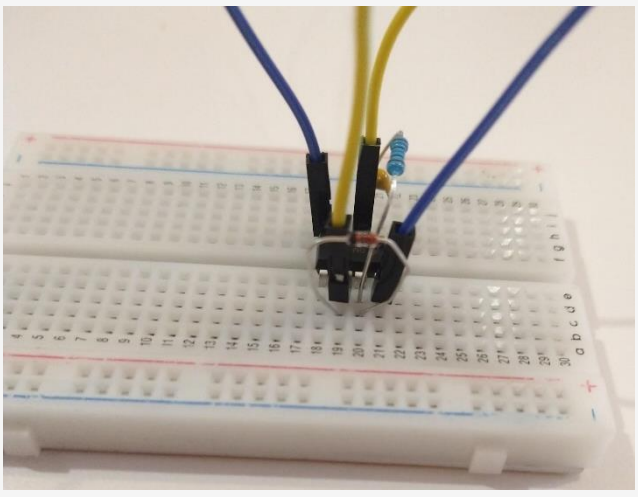
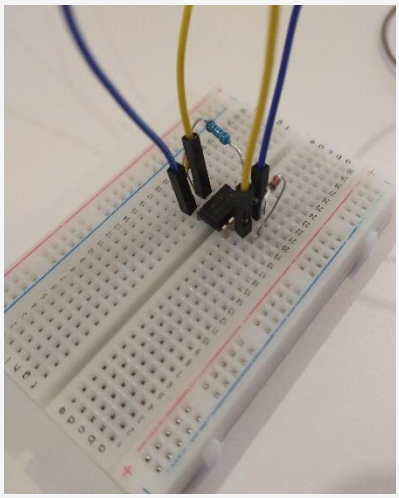
Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Προσθέτουμε μια αντίσταση τερματισμού 220k ohm που συνδέει τα πόδια 2 και 7.

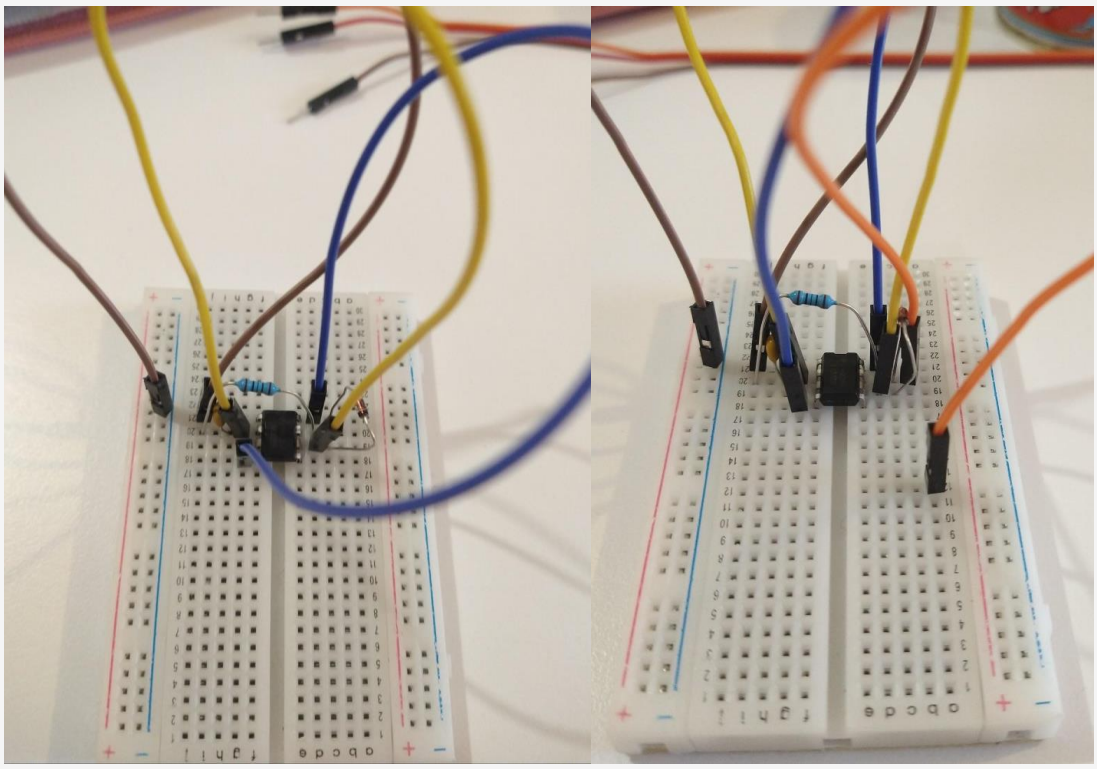
Στη συνέχεια, προσθέτουμε μια δίοδο Zener που συνδέει τα πόδια 6 και 7 του τσιπ NE555, όπως φαίνεται στις εικόνες. Βεβαιωθείτε ότι τοποθετήσατε τη δίοδο προς τη σωστή κατεύθυνση, δηλαδή με τη μαύρη λωρίδα στο πόδι 6.

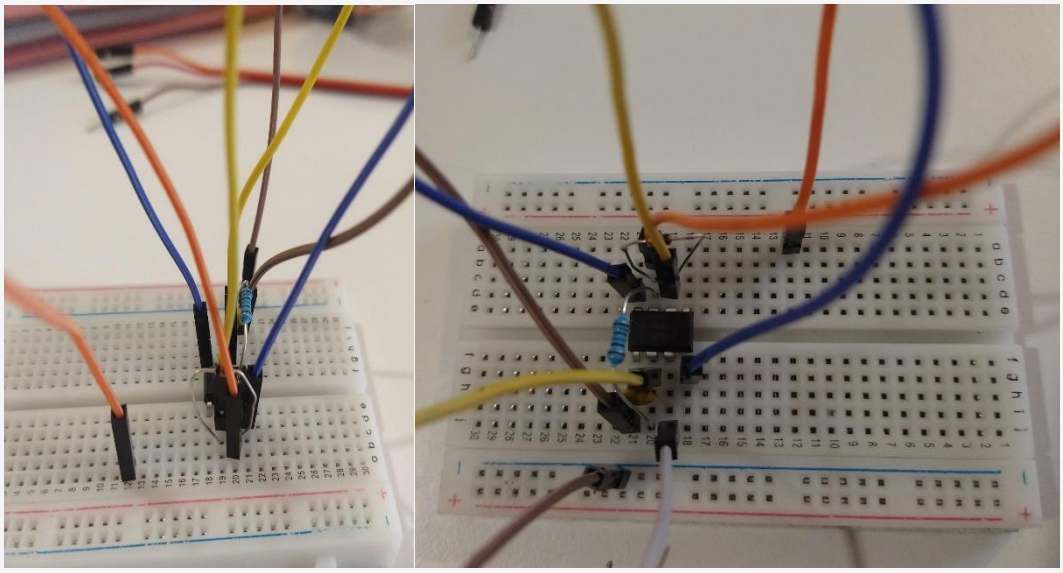



Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Προσθέτουμε ένα καλώδιο τύπου jumper το οποίο περνάει από το πόδι 1 στον αρνητικό πόλο και, στη συνέχεια, προσθέτουμε ένα καλώδιο το οποίο συνδέει το πόδι 7 με μια άλλη γραμμή πιο πέρα στο breadboard. Έπειτα συνδέουμε ένα νέο καλώδιο από τη μια πλευρά στο πόδι 3 του τσιπ NE555 (προς το παρόν δεν μας απασχολεί η ένωση της άλλης άκρης του καλωδίου).



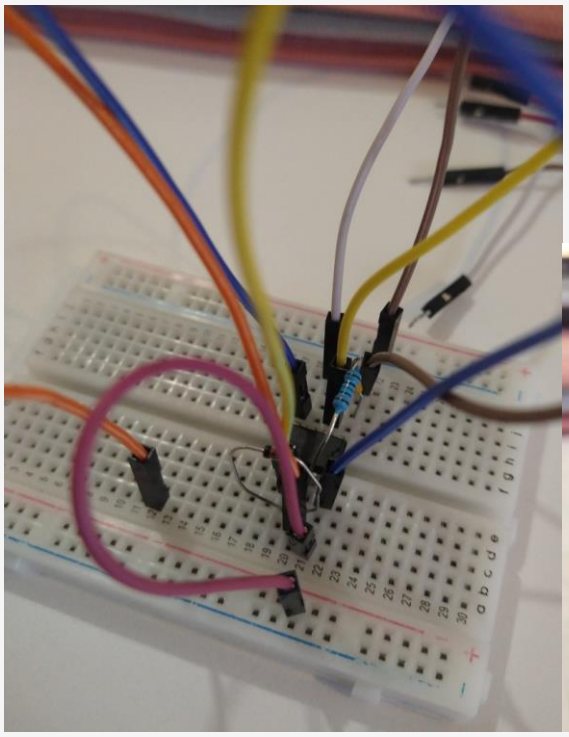


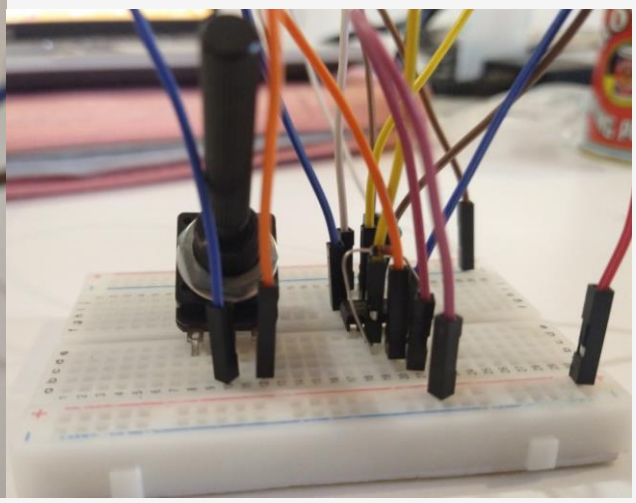
Προσθέτουμε ένα καλώδιο για να συνδέσουμε το πόδι 1 με τον θετικό πόλο. Στη συνέχεια, τοποθετούμε το ποτενσιόμετρο όπως φαίνεται στην πιο κάτω εικόνα, δηλαδή με το ένα

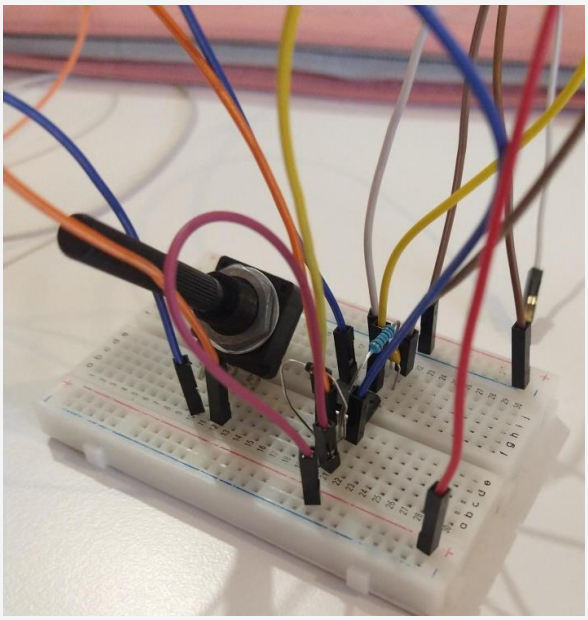
Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

ποδί στην ίδια γραμμή του καλωδίου (πορτοκαλί) από την οποία ξεκινά το πόδι του τσιπ NE555. Προσθέτουμε στη γραμμή όπου βρίσκεται το μεσαίο πόδι του ποτενσιόμετρου ένα άλλο καλώδιο που θα ενωθεί με τον θετικό πόλο.



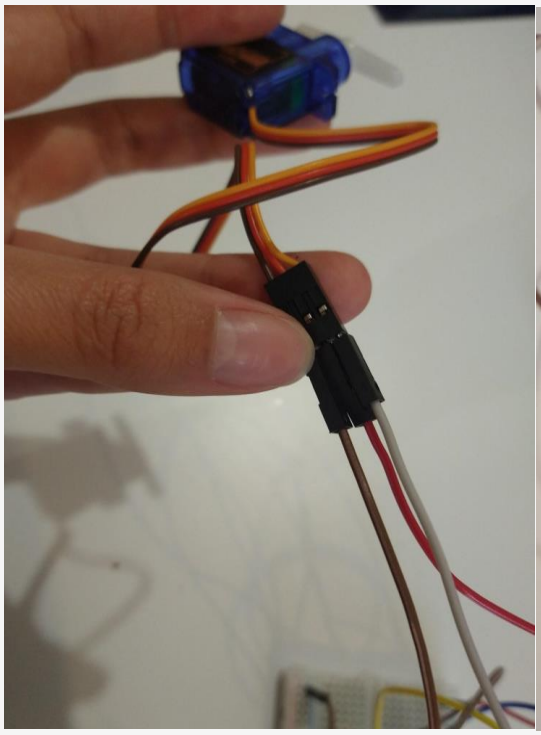


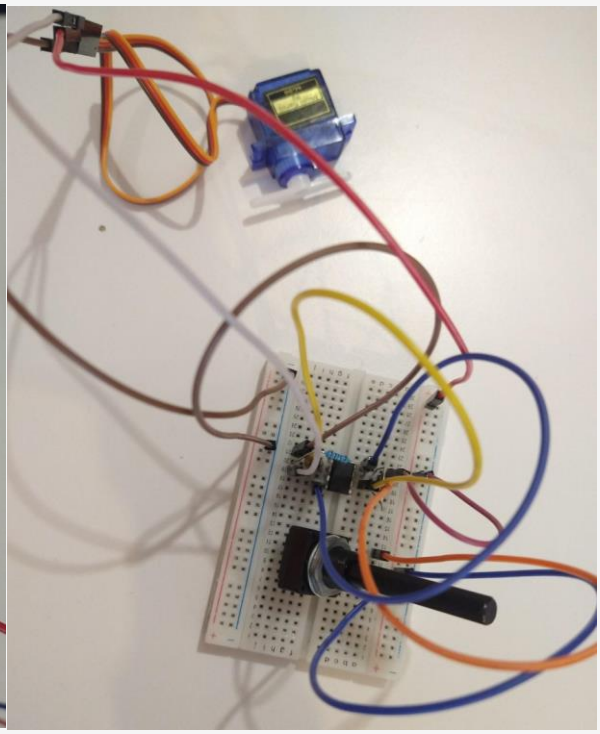


Παίρνουμε τώρα το (λευκό) καλώδιο που ήταν συνδεδεμένο στο πόδι 3 του τσιπ NE555 από το ένα άκρο και το συνδέουμε με το πορτοκαλί καλώδιο του σερβοκινητήρα. Επίσης, συνδέουμε ένα καλώδιο στον θετικό πόλο του breadboard από τη μία άκρη στην άλλη άκρη του κόκκινου καλωδίου του σερβοκινητήρα. Κάνουμε το ίδιο με το καλώδιο που βγαίνει από τον αρνητικό πόλο του breadboard και συνδέεται με το καφέ καλώδιο του σερβοκινητήρα (Δείτε τις εικόνες πιο κάτω).

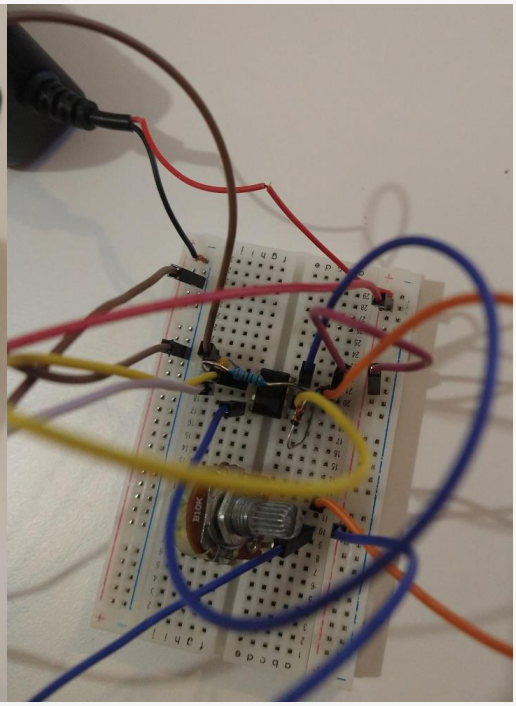
Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης





Το τελευταίο βήμα είναι να πάρετε έναν προσαρμογέα 5V, να κόψετε την άκρη του και να αφαιρέσετε το πλαστικό του για να χρησιμοποιήσετε το κόκκινο και το μαύρο καλώδιο που διαθέτει. Γυμνώνουμε λίγο τις δύο άκρες που θα ενώσουμε με τους θετικούς και αρνητικούς πόλους αντίστοιχα.



Αυτό ήταν, το κύκλωμά μας έχει ολοκληρωθεί!

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Για να ολοκληρώσετε το ηλεκτρομηχανικό σας έργο, συνδέστε τον σερβοκινητήρα στην κεφαλή του παιχνιδιού και κρύψτε το ηλεκτρονικό κύκλωμα μέσα στο χάρτινο κουτί. Φρόντιστε να ανοίξετε μια τρύπα για το ποτενσιόμετρο.

3.6. Ηλεκτρική κλειδαριά ασφαλείας με τηλεχειριστήριο

Η αξιοποίηση των υπέρυθρων στην τεχνολογία ασύρματης επικοινωνίας αποτελεί μια κοινή, προσιτή και εύκολη πρακτική. Το υπέρυθρο φως είναι κατά πολύ παρόμοιο με το ορατό φως, εκτός του ότι έχει ελαφρώς μεγαλύτερο μήκος κύματος. Αυτό σημαίνει ότι το υπέρυθρο φως δεν μπορεί να γίνει αισθητό στο ανθρώπινο μάτι, γεγονός που το καθιστά ιδανικό για την ασύρματη επικοινωνία.

Για να κατανοήσετε τον τρόπο με τον οποίο αξιοποιεί η τεχνολογία ασύρματης επικοινωνίας τις υπέρυθρες, σχεδιάσαμε μια δραστηριότητα στην επόμενη υποενότητα η οποία περιλαμβάνει την κατασκευή μιας ασύρματης κλειδαριάς ασφαλείας με τηλεχειριστήριο που είναι συνδεδεμένη σε μια πλακέτα Arduino.

Η κατασκευή μιας ηλεκτρικής κλειδαριάς ασφαλείας για πόρτα με τηλεχειριστήριο, που λειτουργεί με τεχνολογία υπέρυθρων.

Σε αυτό το ηλεκτρομηχανικό έργο, θα αξιοποιήσουμε τις πληροφορίες που συγκεντρώθηκαν προηγούμενως αναφορικά με την τεχνολογία υπέρυθρων για να κατασκευάσουμε μια ηλεκτρική κλειδαριά ασφαλείας για πόρτα που λειτουργεί με τηλεχειριστήριο.

Τα εξαρτήματα που απαιτούνται για την ολοκλήρωση του έργου είναι ένα τηλεχειριστήριο υπέρυθρων, ένας δέκτης υπέρυθρων και ένας σερβοκινητήρας.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Βήμα 1: Εγκατάσταση βιβλιοθήκης

Μεταβείτε σε αυτή την ιστοσελίδα και κατεβάστε τη βιβλιοθήκη. Στη συνέχεια, θα πρέπει να εγκαταστήσετε τη βιβλιοθήκη στην πλατφόρμα του Arduino IDE. Μεταβείτε στο Arduino IDE → Sketches→ Include a library → IRremote.

Βήμα 2: Λύσεις σχετικά με τον τρόπο επικοινωνίας μεταξύ της κλειδαριάς και του τηλεχειριστηρίου υπέρυθρων

Για να κατανοήσουμε τον τρόπο με τον οποίο ερμηνεύει το Arduino τα ηλεκτρικά σήματα που εκπέμπονται από το τηλεχειριστήριο σας, πρέπει πρώτα να μεταφορτώσετε τον ακόλουθο κώδικα στην πλατφόρμα.

```
/* Αναζήτηση των κωδίκων για την επικοινωνία μεταξύ κλειδαριάς και τηλεχειριστηρίου
Περισσότερες πληροφορίες: https://www.makerguides.com */

#include <IRremote.h<IRremote.h> // εισάγει την βιλιοθήκη IRremote στο σκίτσο μας

#define RECEIVER_PIN 13 // ορίζει την ακίδα που θα είναι ο δέκτης υπέρυθρων
IRrecv receiver(RECEIVER_PIN); // δημιουργεί ένα αντικείμενο της κλάσης IRrecv το οποίο θα είναι ο δέκτης υπερύθρων
decode_results results; // δημιουργεί ένα αντικείμενο αποτελέσματος της κλάσης decode_results

void setup() {
Serial.begin(9600); // εκκινεί τη σειριακή επικοινωνία με ρυθμό μετάδοσης συμβόλων 9600 bauds
receiver.enableIRIn();// ενεργοποιεί τον δέκτη
receiver.blink13(true); // ενεργοποιεί το αναβόσβημα  του ενσωματωμένου λαμπτήρα LED όταν λαμβάνεται ένα υπέρυθρο σήμα
}

void loop() {
if (receiver.decode(&results)) { // αποκωδικοποιεί το σήμα που λήφθηκε και το αποθηκεύει στα αποτελέσματα
Serial.println(results.value, HEX); // εκτυπώνει τις τιμές στη σειριακή οθόνη
receiver.resume(); // επαναφέρει τον δέκτη για τον επόμενο κώδικα
  }
}
```

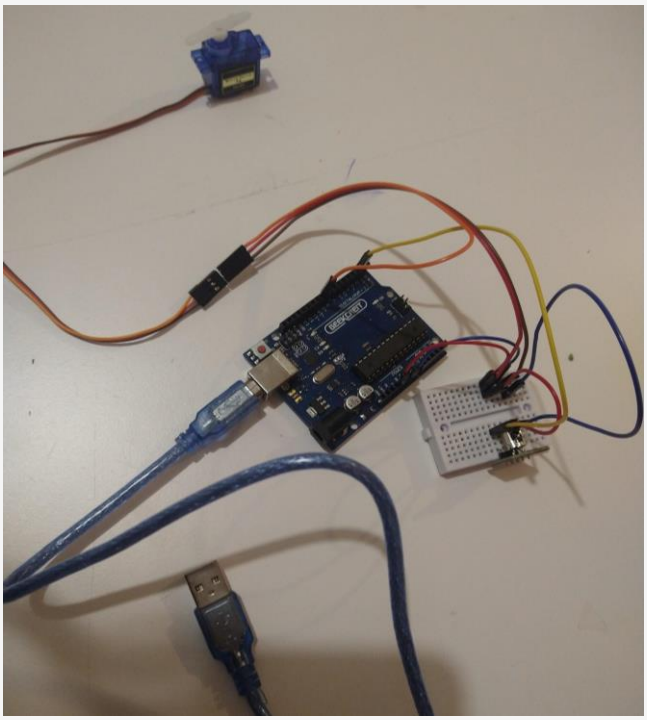
Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Στη συνέχεια, μπορείτε να ανοίξετε τη σειριακή οθόνη Arduino για να δείτε το κλειδί που θα εμφανιστεί κάνοντας κλικ στα κουμπιά του τηλεχειριστηρίου. Κάθε κουμπί του τηλεχειριστηρίου σας αντιστοιχεί με ένα διαφορετικό κλειδί. Σημειώστε τις διαφορετικές ενδείξεις κλειδιών επειδή θα χρειαστείτε αυτές τις πληροφορίες αργότερα.

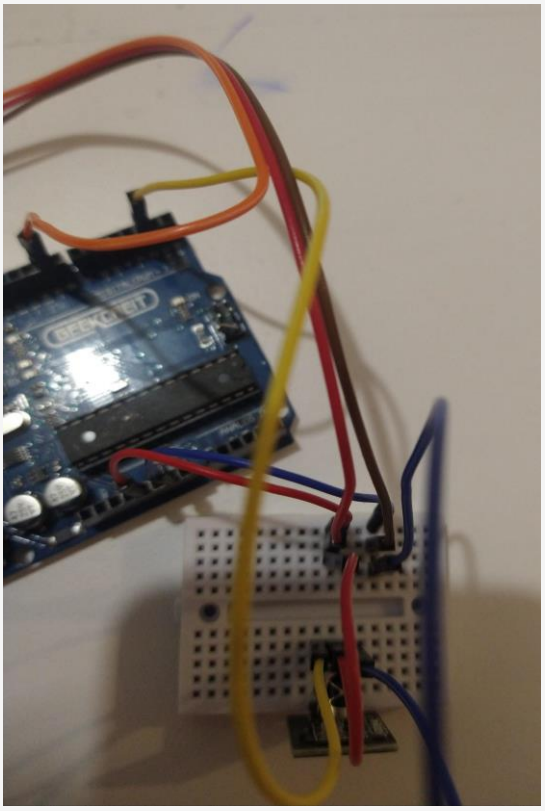
Βήμα 3: Καλωδίωση όλων των εξαρτημάτων

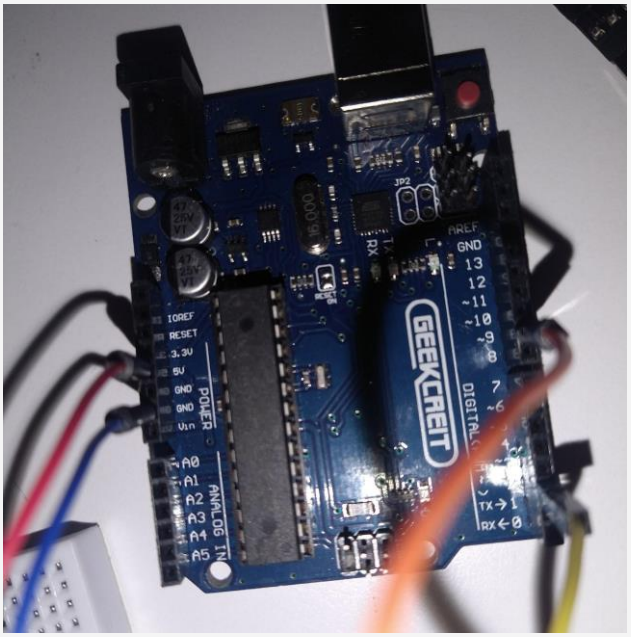
Συνδέστε όλα τα εξαρτήματα μεταξύ τους όπως απεικονίζονται στις πιο κάτω εικόνες. Να είστε προσεκτικοί με τη λήψη υπέρυθρων: η θετική μονάδα εξόδου συνδέεται στο 5V της πλακέτας Arduino, η αρνητική μονάδα εξόδου στο έδαφος ενώ η μονάδα εξόδου για τα σήματα στην ψηφιακή ακίδα PIN 2 (συμβουλευτείτε τον κώδικα πιο κάτω).
Σημειώστε ότι η μονάδα εξόδου των σημάτων του σερβοκινητήρα θα πρέπει να είναι συνδεδεμένη με την ψηφιακή ακίδα 9 της πλακέτας του Arduino.

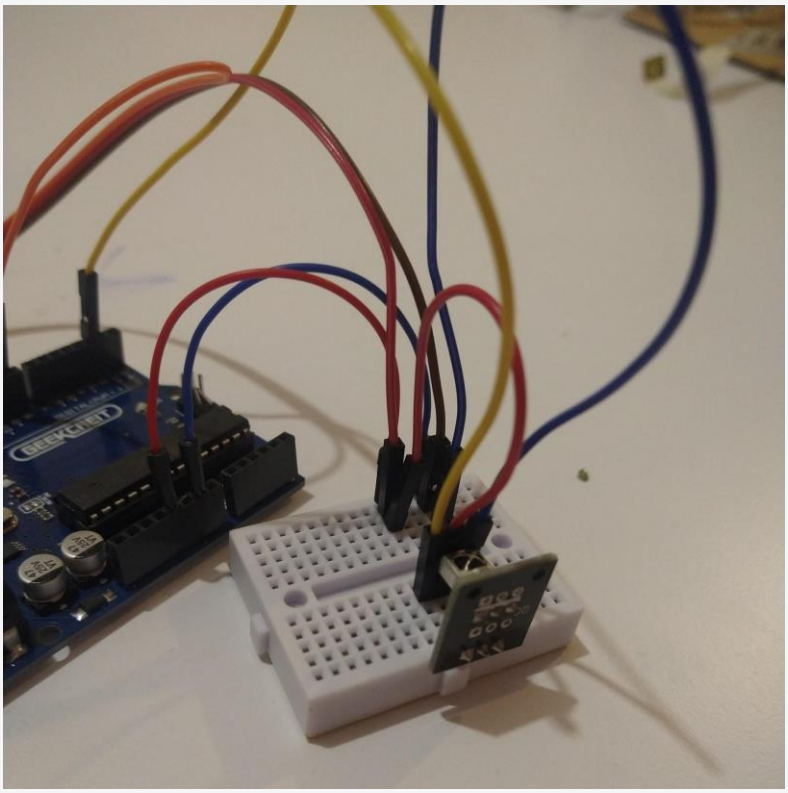


Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης







Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Βήμα 4: Εισαγωγή κώδικα

Πρέπει να μεταφορτώσουμε τον ακόλουθο κώδικα στην πλακέτα του Arduino:

```
#include<IRremote.h>      // δημιουργεί αντίγραφο της βιβλιοθήκης IRremote στις βιβλιοθήκες του Arduino
#include <Servo.h>
# set up 0xFF906F   // σήμανση για την δεξιόστροφη περιστροφή
# set down 0xFFE01F //σήμανση για την αριστερόστροφη περιστροφή

int RECV_PIN = 2;    // η ακίδα-δέκτης των υπέρυθρων σημάτων
Servo servo;
int Val;          //γωνία περιστροφής
bool cwRotation, ccwRotation;//οι καταστάσεις περιστροφής

IRrecv irrecv(RECV_PIN);

decode_results results;

void setup()
{
 Serial.begin(9600);
 irrecv.enableIRIn(); // Εκκίνηση του δέκτη
 sERVO.attach(9);    //η ακίδα του σερβοκινητήρα
}

void loop()
{
if (irrecv.decode(&results)) {
Serial.println(results.value, HEX);
irrecv.resume(); // Λήψη της επόμενης τιμής

   if (results.value == up)
   {
cwRotation = !cwRotation;       //αντιστροφή της τιμής περιστροφής
ccwRotation = false;        //χωρίς περιστροφή προς αυτή την κατεύθυνση
   }

if (results.value == down)
   {
ccwRotation = !ccwRotation;    //αντιστροφή της τιμής περιστροφής
cwRotation = false;           //χωρίς περιστροφή προς αυτή την κατεύθυνση
```

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

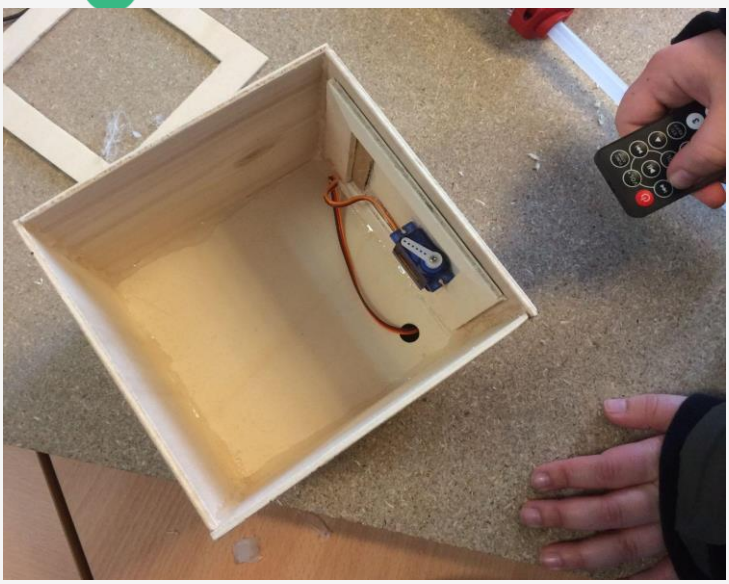
```
  }
 }
if (cwRotation && (val != 175))  {
val++;                    //για το κουμπί δεξιόστροφης κίνησης
 }
 if (ccwRotation && (val != 0))  {
val--;                    //για το κουμπί της αριστερόστροφης κίνησης
 }
 servo.write(val);
 delay(20);        //Ταχύτητα
}
```

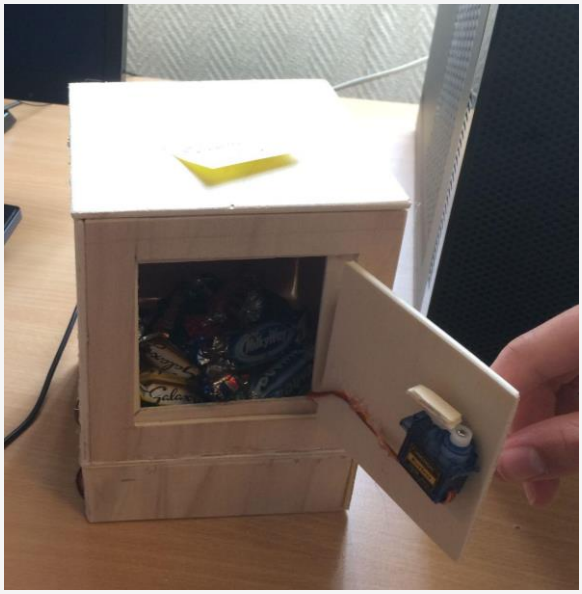
Βήμα 5: Κατασκευή της κλειδαριάς πόρτας

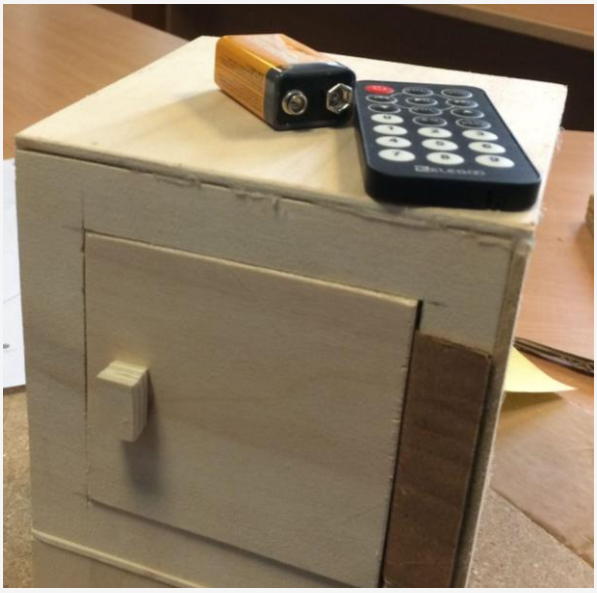
Στο στάδιο αυτό, θα πρέπει να έχετε ήδη κάνει δοκιμή του σκίτσου σας. Εάν ο σερβοκινητήρας ανταποκρίνεται στις εντολές που αποστέλλονται από το τηλεχειριστήριο, τότε ήρθε η ώρα να αποφασίσετε τι είδους κλειδαριά πόρτας θέλετε να χρησιμοποιήσετε. Μπορείτε να είστε δημιουργικοί με την επιλογή του είδους της κλειδαριάς που θέλετε να κατασκευάσετε. Προτείνουμε μια λύση που περιλαμβάνει την προσάρτηση του σερβοκινητήρα απευθείας σε μια πόρτα, όπως φαίνεται στην πιο κάτω εικόνα.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης







3.7. Μέτρηση θερμοκρασίας, υγρασίας, φωτός και χρώματος

Στα προηγούμενα έργα, χρησιμοποιήσαμε κινητήρες για να επιτύχουμε διαφορετικά αποτελέσματα, από την απλή οδήγηση ενός κινητήρα δεξιόστροφα ή αριστερόστροφα μέχρι την κατασκευή πιο εξελιγμένων συσκευών, όπως το διαδραστικό παιχνίδι από χαρτί ή την κατασκευή μιας ασύρματης κλειδαριάς με τηλεχειριστήριο υπέρυθρων.

Τώρα, οι κινητήρες δρουν σαν ενεργοποιητές που σημαίνει ότι προορίζονται να εκτελέσουν μια συγκεκριμένη ενέργεια. Μαζί με τους ενεργοποιητές, θα μπορούσε κανείς να επιλέξει να χρησιμοποιήσει μερικούς αισθητήρες. Είχαμε την ευκαιρία να εξερευνήσουμε μερικούς τύπους αισθητήρες στο προηγούμενο κεφάλαιο. Ο πρώτος τύπος αισθητήρα που είδαμε ήταν το ποτενσιόμετρο, το δεύτερο ο υπέρυθρος αισθητήρας που λαμβάνει τα σήματα υπό μορφή υπέρυθρων και τα μεταφράζει σε κώδικα υπολογιστή στην πλατφόρμα του Arduino.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Αυτή η υποενότητα καταπιάνεται με τους αισθητήρες. Στόχος της υποενότητας είναι να εξοικειωθείτε με αυτόν τον τύπο ηλεκτρικών εξαρτημάτων για να μπορέσετε να τους εφαρμόσετε στα ηλεκτρομηχανικά σας έργα και να τα εξελίξετε.

Πιο συγκεκριμένα, θα αναφερθούμε στους αισθητήρες που χρησιμοποιούνται για την μέτρηση της θερμοκρασίας, του φωτός, της υγρασίας και του χρώματος.

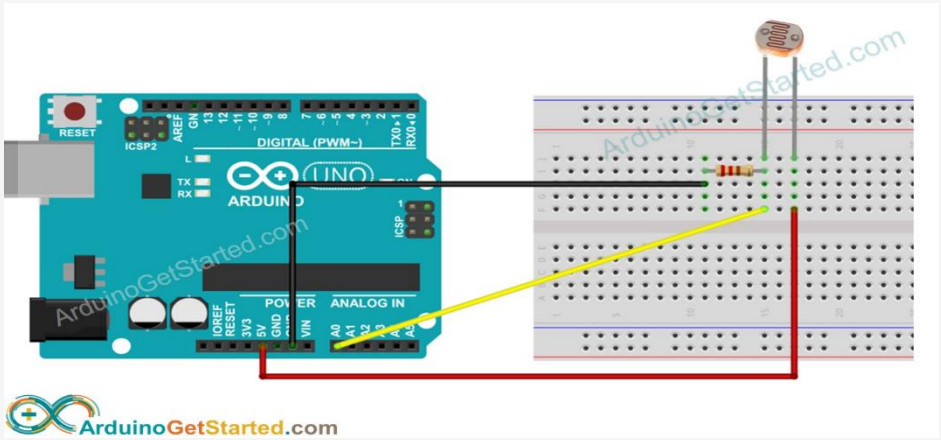
3.7.1. Η χρήση ενός αισθητήρα με το Arduino

Ο αισθητήρας φωτός είναι ένας τύπος φωτοαντιστάτη, ο οποίος είναι γνωστός επίσης και ως φωτοεξαρτώμενος αντιστάτης. Χρησιμοποιείται για την ανίχνευση φωτός, καθώς επίσης και για τη μέτρηση του επιπέδου φωτεινότητας ενός συγκεκριμένου περιβάλλοντος.

Ένας αισθητήρας φωτός διαθέτει δύο ακίδες και επειδή λειτουργεί όπως ένας τυπικός αντιστάτης δεν υπάρχει ανάγκη διάκρισης μεταξύ τους.

Όσο υψηλότερη είναι η ένταση του φωτός, τόσο μικρότερη είναι η αντίσταση που καταγράφεται από τον αισθητήρα φωτός. Ως εκ τούτου, με τη μέτρηση της φωτοαντίστασης από τον αισθητήρα φωτός μπορούμε να γνωρίζουμε πόσο φωτεινό είναι ένα περιβάλλον.

Πιο κάτω μπορείτε να δείτε πώς γίνεται η σύνδεση ενός φωτοαντιστάτη με την πλακέτα του Arduino.



Εικόνα 47 – Η σύνδεση φωτοαντιστάτη στην πλακέτα του Arduino
Πηγή: https://arduinogetstarted.com/tutorials/arduino-light-sensor

Πιο κάτω παρατίθεται ένας απλός κώδικας που μπορείτε να μεταφορτώσετε στην πλατφόρμα του Arudino, ο οποίος εμφανίζει τις τιμές που καταγράφονται από τον αισθητήρα φωτός στη σειριακή οθόνη του Arduino IDE.

```
void setup() {
  // Αρχικοποιεί τη σειριακή επικοινωνία στα 9600 bit ανά δευτερόλεπτο:
Serial.begin(9600);
```

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

```
}

void loop() {

 // διαβάζει την είσοδο στην αναλογική ακίδα A0 (τιμές bits 0-1023)

int analogValue = analogRead(A0);

Serial.print("Analog reading: ");

Serial.print(analogValue);   // αναλογική είσοδος

delay(500);

}
```

3.7.2. Κατασκευή ενός θέρεμιν με Arduino και έναν αισθητήρα φωτός

Το θέρεμιν είναι ένα ηλεκτρονικό μουσικό όργανο το μπορεί να παιχτεί από απόσταση, δηλ. «στον αέρα», χωρίς άμεση επαφή με τον ερμηνευτή. Ο ήχος παράγεται με την κίνηση των χεριών του ερμηνευτή ανάμεσα στις κεραίες και το ηλεκτρικό δυναμικό του σώματος του οργάνου, δημιουργώντας έτσι αλληλεπίδραση με ένα μαγνητικό πεδίο. Οι μεταβολές αυτές του ηλεκτρομαγνητικού πεδίου αναγνωρίζονται, ενισχύονται και αναπαράγονται από τις κεραίες ως ήχος .

Στο ηλεκτρομηχανικό έργο αυτό, θα επιχειρήσουμε να κατασκευάσουμε ένα παρόμοιο ηλεκτρονικό όργανο το οποίο θα έχει τη δυνατότητα να αλλάζει την τονικότητα των φθόγγων όταν ο χρήστης κουνάει τα χέρια του μπροστά από αυτό.

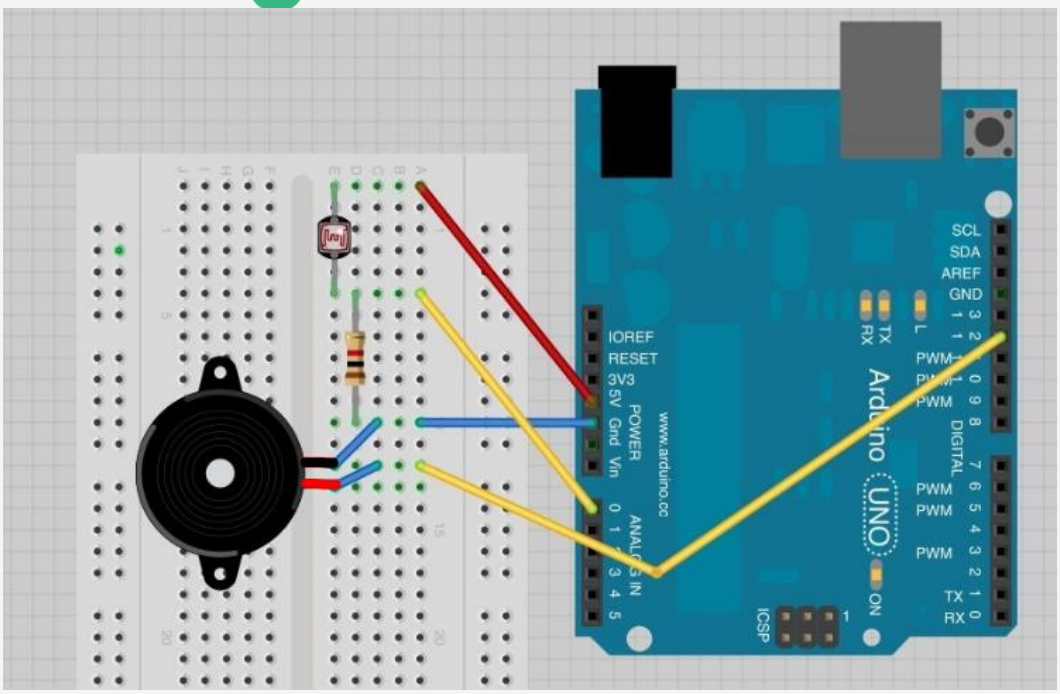
Θα χρειαστούμε ένα πιεζοφωνικό βομβητή (piezzo buzzer), έναν αισθητήρα φωτός και έναν αντιστάτη 1k ohm.

Βήμα 1: Καλωδίωση

Ολοκληρώστε την καλωδίωση σύμφωνα με την πιο κάτω εικόνα

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης



Εικόνα 48 – Η καλωδίωση ενός θερεμίν με Arduino

Πηγή: https://learn.adafruit.com/adafruit-arduino-lesson-10 making-sounds/pseudo-theramin

Βήμα 2: Κώδικας

```
int speakerPin = 12;

int photocellPin = 0;


void setup() {

{

}


void loop() {

{

int reading = analogRead(photocellPin);

int pitch = 200 + reading / 4;
```

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

```
tone(speakerPin, pitch);

}
```

Το σκίτσο που θα μεταφορτώσουμε είναι στην πραγματικότητα πολύ απλό. Αρχικά, θα χρειαστεί να μετρήσουμε την τιμή της έντασης του φωτός χρησιμοποιώντας την αναλογική ακίδα A0. Η τιμή μέτρησης θα κυμαίνεται περίπου από το 0 μέχρι το 700.

Επειδή οι τιμές που παίρνουμε από την αναλογική έξοδο είναι συνεχόμενες, θα πρέπει να ορίσουμε τα 200 Hz ως τη χαμηλότερη συχνότητα ένδειξης και να διαιρούμε την τιμή που λαμβάνουμε δια 4, για να μας δίνει ένα εύρος τιμών μεταξύ των 200 Hz και 370 Hz.

Εάν θέλετε, μπορείτε να διαιρέσετε τις τιμές που διαβάζει ο αισθητήρας φωτός με διαφορετικό τρόπο για να έχετε διαφορετικά αποτελέσματα. Για παράδειγμα, αντικαταστήστε τον διαιρέτη 4 με τον διαιρέτη 2 για να δείτε τι θα προκύψει.

3.7.3. Ρομπότ διαλογής αντικειμένων με βάση το χρώμα

Όπως υποδηλώνει και το όνομά τους, τα εν λόγω ρομπότ χρησιμοποιούνται για την διαλογή και το διαχωρισμό των αντικειμένων με βάση το χρώμα τους.

Προφανώς, αυτό μπορεί να επιτευχθεί μέσω της διάκρισης και της διαλογής των αντικειμένων με βάση το χρώμα τους, ωστόσο, όταν έχουμε έναν μεγάλο όγκο αντικειμένων, η διαδικασία αυτή μπορεί να καταντήσει επίπονη και αρκετά μονότονη. Για τον σκοπό λοιπόν αυτό, τα αυτόματα μηχανήματα διαλογής που επιτρέπουν το διαχωρισμό των αντικειμένων με βάση το χρώμα τους μπορούν να φανούν ιδιαίτερα χρήσιμα σε τέτοιου είδους εργασίες.

Ο συγκεκριμένος τύπος ρομποτικών μηχανημάτων διαθέτει αισθητήρες χρώματος για να ανιχνεύουν και να ξεχωρίζουν οπτικά το χρώμα οποιουδήποτε αντικειμένου. Αφού ανιχνεύσουν το χρώμα ενός αντικειμένου, ενεργοποιείται ένας κινητήρας ή ένα σύστημα κινητήρων το οποίο αρπάζει το αντικείμενο και το τοποθετεί στην κατάλληλη θέση υποδοχής. Τα μηχανήματα διαλογής που επιτρέπουν το διαχωρισμό των αντικειμένων με βάση το χρώμα χρησιμοποιούνται ευρέως σε τομείς όπου η ταυτοποίηση, η διάκριση και η ταξινόμηση των χρωμάτων είναι σημαντική. Ορισμένοι από τους τομείς στους οποίους εφαρμόζονται αυτά τα μηχανήματα διαλογής είναι ο γεωργικός τομέας (όπου χρησιμοποιούνται για την διαλογή σιτηρών με βάση το χρώμα), τη βιομηχανία τροφίμων, τη διαμαντοβιομηχανία και τη μεταλλευτική βιομηχανία, την ανακύκλωση κ.λπ.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Βιομηχανικά μηχανήματα διαλογής αντικειμένων με βάση το χρώμα

Ας ρίξουμε μια ματιά σε μερικά βιομηχανικά μηχανήματα διαλογής αντικειμένων με βάση το χρώμα.

Εικόνα 49 - Βιομηχανικά μηχανήματα διαλογής αντικειμένων με βάση το χρώμα.

Πηγή: http://hugeacademy.com

Υπάρχουν δύο τύποι διαλογέων που βασίζονται σε αισθητήρες χρώματος: ο τύπος αγωγού ή αλεξίπτωτου και ο τύπος ζώνης

Ο διαλογέας τύπου ζώνης τύπου σπάζει σε ένα μικρότερο ποσοστό το υλικό (σημαντικό για τους ξηρούς καρπούς) και τα προϊόντα παραμένουν σχετικά στατικά κατά τη διάρκεια της οριζόντιας μεταφοράς τους πάνω στη ζώνη. Αντίθετα, με τον διαλογέα τύπου αγωγού, το υλικό περνάει τον αγωγό σε ελεύθερη πτώση λόγω της βαρύτητας, κάτι που προκαλεί σύγκρουση, τριβή και μεγάλες κατακόρυφες κινήσεις, αυξάνοντας έτσι την αναλογία του σπασμένου υλικού. Η διάταξη του διαλογέα τύπου ζώνης καθιστά τη μεταφορά των προϊόντων ομαλή και σταθερή χωρίς να προκαλείται αναπήδηση του υλικού.

Οι διαλογείς τύπου αγωγού χρησιμοποιούνται στην βιομηχανία των τροφίμων, καθώς το κόστος επένδυσης και λειτουργίας του είναι χαμηλότερο και επειδή οι δυνατότητες που παρέχει είναι υψηλότερες, καθώς είναι δυνατή η σάρωση των προϊόντων και από τις δύο πλευρές, κάτι που είναι ιδιαίτερα σημαντικό στην απολεπύρωση των σπόρων. Ο διαλογείς τύπου αγωγού είναι συνήθως εφαρμόσιμοι σε συγκεκριμένα προϊόντα, καθώς οι αγωγοί τους σχεδιάζονται συνήθως με ειδικούς διαύλους σύμφωνα με το είδος, το μέγεθος και το σχήμα του εκάστοτε υλικού. Για παράδειγμα, οι διαλογείς τύπου αγωγού μπορούν να επιτρέψουν το χειρισμό υλικών σωματιδίων έως και 5mm, όπως για το ρύζι, τα σιτηρά και τους πλαστικούς κόκκους. Οι διαλογείς τύποι αγωγού που είναι επίπεδοι είναι κατάλληλοι για το χειρισμό υλικών σωματιδίων πλαστικής ύλης, όπως για τον τύπο πλαστικού PET ή τον τύπο πλαστικού που χρησιμοποιείται για τις συσκευασίες του γάλατος.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

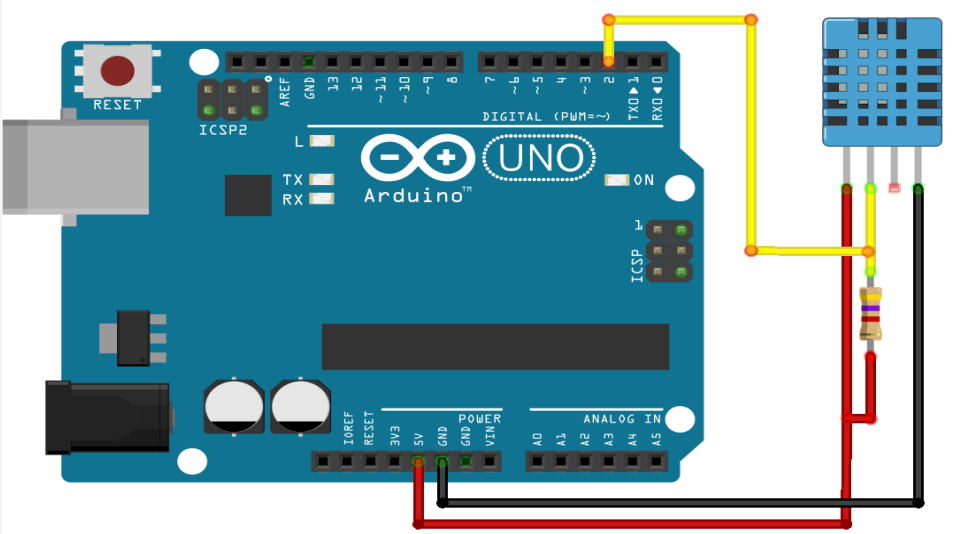
3.7.4. Εισαγωγή στον αισθητήρα τύπου DHT11

Υπάρχει ένας αισθητήρας ο οποίος έχει ένα μεγάλο πεδίο εφαρμογής, συμπεριλαμβανομένης της δυνατότητας μέτρησης της θερμοκρασίας!

Ο αισθητήρας τύπου DHT11 είναι ένας αισθητήρας θερμοκρασίας και υγρασίας, που σημαίνει ότι μπορεί να μετρήσει και τις δύο παραμέτρους. Στο ηλεκτρομηχανικό έργο αυτό, θα μάθουμε πώς γίνεται μια σύνδεση ενός αισθητήρα DHT11 σε μια πλακέτα Arduino και πώς να προγραμματίζουμε την πλατφόρμα του τελευταίου για να λαμβάνουμε τις τιμές θερμοκρασίας που καταγράφονται από τον αισθητήρα θερμοκρασίας σε πραγματικό χρόνο.

Βήμα 1: Καλωδίωση

Συνδέστε όλα τα εξαρτήματα σύμφωνα με την παρακάτω εικόνα:



Εικόνα 50 - Αισθητήρας τύπου DHT11 στο Arduino UNO
Πηγή: https://randomnerdtutorials.com/complete-guide-for-dht11dht22-humidity-and-temperature-sensor-with-arduino/

Είναι δυνατόν να αγνοήσετε την αντίσταση τερματισμού που φαίνεται στο διάγραμμα, η οποία είναι μια αντίσταση των 4,7 k ohm.

Βήμα 2: Εγκατάσταση βιβλιοθηκών

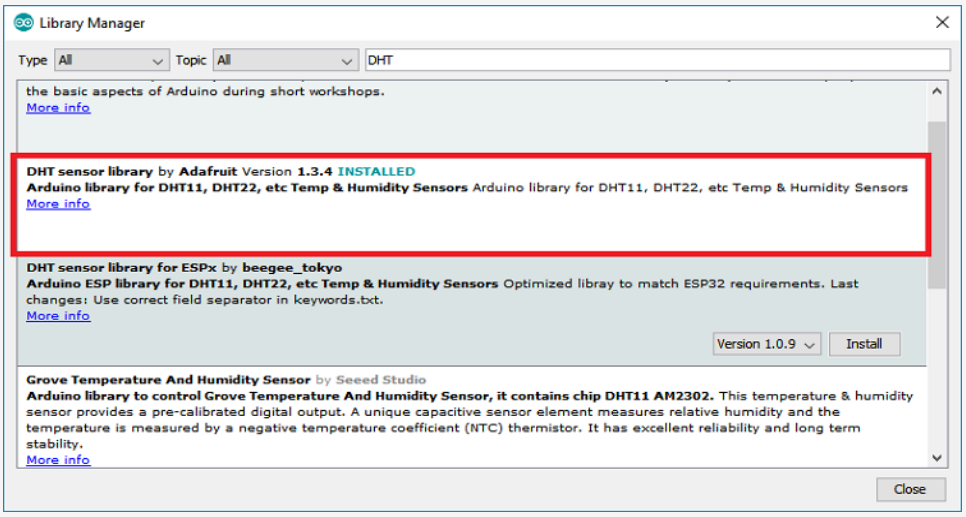
Για να μπορούμε να διαβάζουμε τις τιμές που θα λαμβάνουμε από τον αισθητήρα DHT, θα πρέπει να χρησιμοποιήσουμε τη βιβλιοθήκη DHT από το Adafruit. Για να χρησιμοποιήσετε αυτή τη βιβλιοθήκη πρέπει επίσης να εγκαταστήσετε τη βιβλιοθήκη του Adafruit Unified Sensor. Ακολουθήστε τα επόμενα βήματα για να εγκαταστήσετε αυτές τις βιβλιοθήκες.

Ανοίξτε το IDE Arduino και επιλέξτε Sketch > Include Library > Manage Libraries. Ο Διαχειριστής Βιβλιοθήκης (Library Manager) θα πρέπει λογικά να ανοίξει.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Αναζητήστε το "DHT" στο πλαίσιο αναζήτησης (search box) και εγκαταστήστε τη βιβλιοθήκη DHT από το Adafruit.



Εικόνα 51 – Εγκατάσταση βιβλιοθήκης DHT από το Adafruit

Πηγή: https://randomnerdtutorials.com/complete-guide-for-dht11dht22-humidity-and-temperature-sensor-with-arduino/

Βήμα 3: Κώδικας

```
#include "DHT.h"

#define DHTPIN 2    // ορίζει την ακίδα με την οποία θα είμαστε συνδεδεμένοι

#define DHTTYPE DHT11   // DHT 11

// Αρχικοποιεί τον αισθητήρα DHT με την κανονική συχνότητα 16mhz του Arduino

DHT dht(DHTPIN, DHTTYPE),

void setup() {

Serial.begin(9600);

Serial.println("DHTxx test!");


dht.begin();

}

void loop() {

// Αναμένει για λίγα δευτερόλεπτα μεταξύ των μετρήσεων.

  delay(2000);
```

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

```
// Η ανάγνωση των τιμών θερμοκρασίας και υγρασίας διαρκεί περίπου 250 χιλιοστά του δευτερολέπτου!

// Οι ενδείξεις του αισθητήρα μπορεί επίσης να είναι μέχρι και 2 δευτερόλεπτα πρωθύστερες (πρόκειται για έναν πολύ αργό αισθητήρα)


// Διαβάζει τη θερμοκρασία σε βαθμούς Κελσίου

float t = dht.readTemperature();


Serial.print("Temperature: ");

 Serial.print(t);

Serial.print(" *C ");

}
```

3.7.5. Η κατασκευή ενός έξυπνου ανεμιστήρα με ψύξη

Στο ηλεκτρομηχανικό έργο αυτό, θα εφαρμόσετε τις γνώσεις σας αναφορικά με τους αισθητήρες και τους ενεργοποιητές για να κατασκευάσετε μια χρήσιμη και μοναδική στο είδος της μικροσυσκευή: έναν έξυπνο ανεμιστήρα με ψύξη. Για την υλοποίηση της κατασκευής θα χρειαστείτε έναν κινητήρα συνεχούς ρεύματος DC με ασπίδα κινητήρα L293D (τον οποίο έχουμε συναντήσει σε προηγούμενες ενότητες), καθώς και έναν αισθητήρα τύπου DHT11 για την καταγραφή της θερμοκρασίας ενός δωματίου.

Η τελική κατασκευή θα είναι ένας ανεμιστήρας με ψύξη ο οποίος θα λειτουργεί όταν επιτυγχάνεται μια οριακή θερμοκρασία, την οποία θα έχετε την ευκαιρία να προγραμματίσετε όπως οι ίδιοι επιθυμείτε.

Βήμα 1: Τρισδιάστατη εκτύπωση του ανεμιστήρα με ψύξη

Βρήκαμε αυτό το σχέδιο από το thingiverse. Είναι ένας μίνι ανεμιστήρας με ψύξη που χρησιμοποιεί έναν κινητήρα τύπου DC. Είναι επίσης δυνατό να εκτυπώσετε σε τρισδιάστατη μορφή μια θήκη μπαταρίας για να την χρησιμοποιήσετε ως κάλυμμα μιας μπαταρίας των 9v. Ωστόσο, αυτό δεν είναι απαραίτητο καθώς η βασική χρήση της μπαταρίας των 9V είναι η τροφοδότηση της ασπίδας κινητήρα.

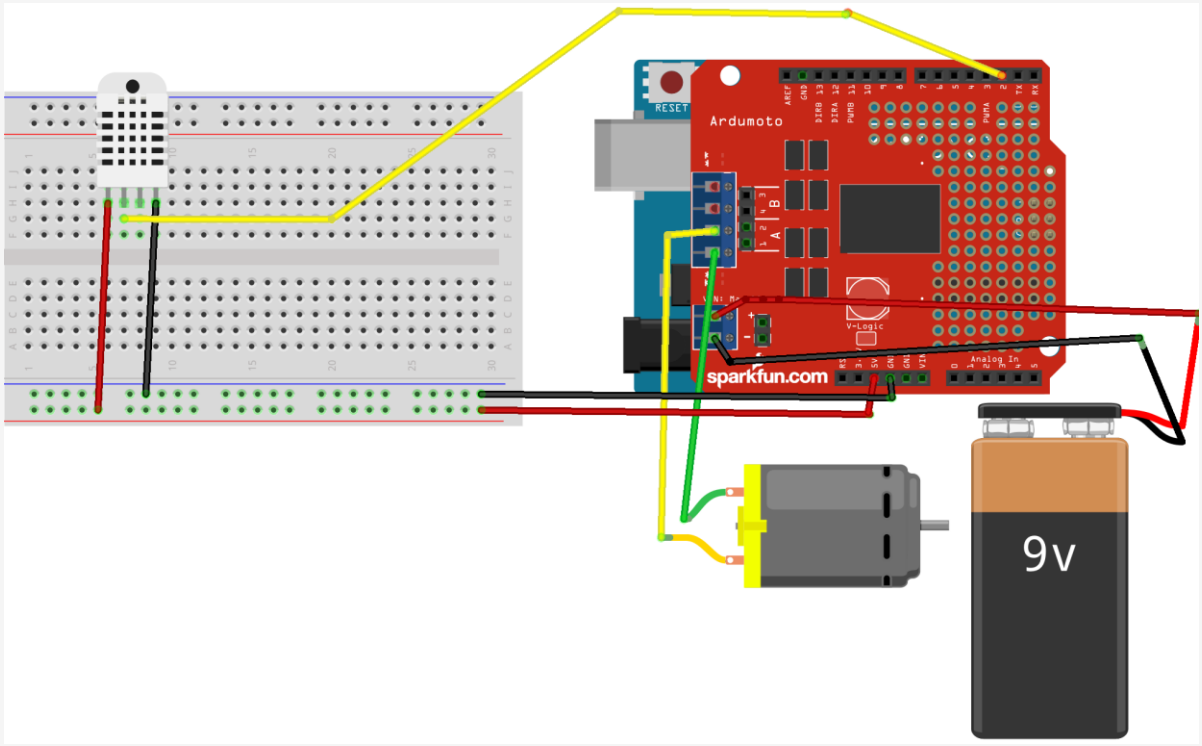
Δοκιμάστε να εκτυπώσετε τον ανεμιστήρα και το υποστηρικτικό υλικό. Εάν δεν έχετε πρόσβαση σε τρισδιάστατο εκτυπωτή, είναι φυσικά δυνατό να δημιουργήσετε αυτά τα δύο στοιχεία από χαρτόνι ή κόντρα πλακέ ή οποιοδήποτε άλλο σταθερό υλικό.

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Βήμα 2: Καλωδίωση των εξαρτημάτων

Συνδέστε όλα τα εξαρτήματα σύμφωνα με την παρακάτω εικόνα:



Εικόνα 52 – Καλωδίωση του έξυπνου ανεμιστήρα με ψύξη
Πηγή: Digijeunes

Βήμα 3: Κώδικας

Σύμφωνα με το πρόγραμμα πιο κάτω, δίνουμε οδηγίες στην πλακέτα του Arduino να ενεργοποιήσει τον κινητήρα DC αν η θερμοκρασία που καταγράφεται από τον αισθητήρα DHT11 είναι ίση ή μεγαλύτερη από τους 24°. Διαφορετικά, ο κινητήρας συνεχούς ρεύματος δεν θα περιστρέφεται.

```
#include "DHT.h"

#include "AFMotor.h"

#define DHTPIN 2    // ορίζει την ακίδα με την οποία θα είμαστε συνδεδεμένοι


AF_DCMotor motor1(1); // Ορίζει τον κινητήρα που θα είναι συνδεδεμένος με το M1 στην ασπίδα κινητήρα


#define DHTTYPE DHT11  // DHT 11
```

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

```
// Αρχικοποιεί τον αισθητήρα DHT με την κανονική συχνότητα 16mhz του Arduino
DHT dht(DHTPIN, DHTTYPE),

void setup() {
 Serial.begin(9600);

 motor1.setSpeed(100); // Ορίζει την ταχύτητα με την οποία θέλουμε ο κινητήρας να περιστρέφεται

dht.begin();
}

void loop() {
// Αναμένει για λίγα δευτερόλεπτα μεταξύ των μετρήσεων.
 delay(2000);

// Η ανάγνωση των τιμών θερμοκρασίας και υγρασίας διαρκεί περίπου 250 χιλιοστά του δευτερολέπτου!
// Οι ενδείξεις του αισθητήρα μπορεί επίσης να είναι μέχρι και 2 δευτερόλεπτα πρωθύστερες (πρόκειται για έναν πολύ αργό αισθητήρα)

// Διαβάζει τη θερμοκρασία σε βαθμούς Κελσίου
float t = dht.readTemperature();

Serial.print("Temperature: ");
 Serial.print(t);
Serial.print(" *C ");
```

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

```
 if (t >= 24)
 {
  motor1.run(BACKWARD);
 }
 else
 {
  motor1.run(RELEASE);
 }
}
```

Αναφορές

Autodesk, Inc. (2020). *What You'll Learn*. https://www.instructables.com/Tools-andMaterials-for-Arduino/

Boxall, J. (2013). *Arduino workshop: A Hands-On introduction with 65 projects*. No Starch Press.

Circuit Basics (n.d.). *Introduction to Microcontrollers*. https://www.circuitbasics.com/introduction-to-microcontrolleres/

Green Steam Incubator. (2019). Module on Microcontrollers: 30 hours lessons. https://steam-incubator.org/wp-content/uploads/2021/11/IO3.2-GSI-Module-on-Microcontrollers.pdf

Makerspaces.com (2022). *Arduino For Beginners*. https://www.makerspaces.com/arduino-uno-tutorial-beginners/

Techatronic (2022). *Types of Arduino Boards.* https://techatronic.com/types-of-arduino-boards-arduino-uno-mega-mini-specification/

Learn Adafruit (2022). *Adafruit motor shield*. https://learn.adafruit.com/adafruit-motor-shield/af-dcmotor-class

Sparkfun (2022). *Servos.* https://www.sparkfun.com/servos

Learning about electronics (2022). *555 timer pinout.* http://www.learningaboutelectronics.com/Articles/555-timer-pinout.php

Girls in STEM (2022). EU funded project. https://girlsinstem.eu/

Arduino Get Started (2022). *Arduino light sensor*. https://arduinogetstarted.com/tutorials/arduino-light-sensor

Adafruit (2012). P*seudo theremin*. https://learn.adafruit.com/adafruit-arduino-lesson-10-making-sounds/pseudo-theramin

Huge academy (2022). http://hugeacademy.com/

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Random nerd tutorials (2022). *Complete guide for DHT11 humidity and temperature sensor with Arduino.* https://randomnerdtutorials.com/complete-guide-for-dht11dht22-humidity-and-temperature-sensor-with-arduino/

DIY robotics (2020). *Articulated robots.* https://diy-robotics.com/article/articulated-robots/

DIY robotics (2020). *What you should know about cartesian robot.* https://diy-robotics.com/article/what-you-should-know-about-cartesian-robot/

DIY robotics (2020). *Scara robots.* https://diy-robotics.com/article/scara-robots/

DIY robotics (2020). *Articulated robots.* https://diy-robotics.com/article/top-six-types-industrial-robots-2020/

Learn mech (2022). *Cylindrical robot diagram construction applications.* https://learnmech.com/cylindrical-robot-diagram-construction-applications/

How to robot (2022). *Industrial robot types and their different uses.* https://www.howtorobot.com/expert-insight/industrial-robot-types-and-their-different-uses

Robot Worx (2022). *What are the main types of robots.* https://www.robots.com/faq/what-are-the-main-types-of-robots

Built In (2022). *Robotics.* https://builtin.com/robotics

Analytics Insight (2021). *Common types of robots.* https://www.analyticsinsight.net/common-types-of-robots-are-there-any-new-ones-you-havent-heard-yet/

Stanford Edu (2022). *Army robots* https://cs.stanford.edu/people/eroberts/cs201/projects/2010-11/ComputersMakingDecisions/army-robots/index.html

NCBI (2019). *Articles.* https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6625162/

Guarforce (2022). https://www.guardforce.com.hk/

Iberdrola (2022). *Educational robots.* https://www.iberdrola.com/innovation/educational-robots

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβρη του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν. (Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696)

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Με τη συγχρηματοδότηση της Ευρωπαϊκής Ένωσης

Το Έργο #CodER είναι συγχρηματοδοτούμενο από το πρόγραμμα ERASMUS+ της Ευρωπαϊκής Ένωσης και βρίσκεται σε εφαρμογή από το Δεκέμβριο του 2021 μέχρι το Νοέμβριο του 2023. Η δημοσίευση αυτή αντικατοπτρίζει τις απόψεις των συντακτών της και η Ευρωπαϊκή Επιτροπή δε φέρει καμία ευθύνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν.

Αριθμός Έργου: 2021-1-FR02-KA220-YOU-000028696

DIGIJEUNES

challedu
inclusion | games | education

RITE
RESEARCH INSTITUTE FOR TECHNOLOGICAL EVOLUTION