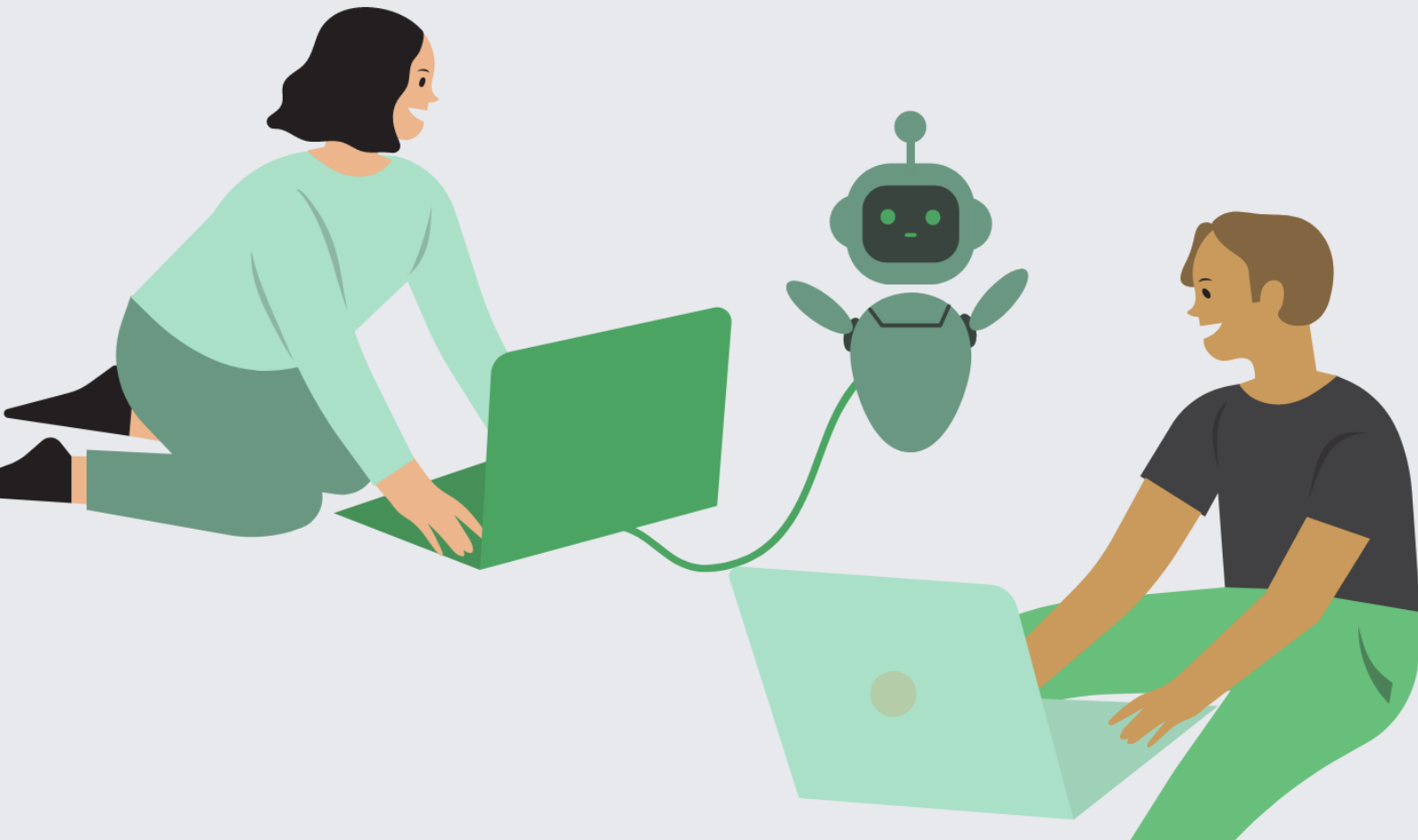
Cofinancé par
l'Union européenne

Le Module CodER sur le Codage & les Microcontrôleurs : 20 heures de cours



Contents

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Introduction

Dans le monde moderne, on met de plus en plus l'accent sur l'acquisition de compétences numériques. Cependant, l'inadéquation actuelle de l'offre et de la demande sur le marché du travail présente de nouveaux défis pour les employeurs et les demandeurs d'emploi. Cette inadéquation affecte principalement les jeunes car le chômage chez les jeunes est l'un des problèmes les plus importants auxquels l'Europe est confrontée. Le projet CodER représente une tentative de combler ce fossé en matérialisant les connaissances sur le codage et les microcontrôleurs grâce à la méthode innovante des Escape Rooms (ER) pour les animateurs et les organisations de jeunesse afin d'éduquer les jeunes et d'accroître leur intérêt et leur engagement envers les métiers à vocation technologique. Les partenaires européens qui participent à ce projet, financé par le programme Erasmus+, sont : Digijeunes (France), Challedu (Grèce), Citizens in Power - C.I.P. (Chypre), RITE (Chypre), AKMI (Grèce) et Kalimera (Croatie).

Grâce à ce projet, les animateurs et les organisations de jeunesse seront dotés de nouvelles compétences et de méthodes innovantes pour attirer les jeunes, et en particulier les jeunes femmes, vers des parcours à vocation technologique. Le module CodER représente la première étape vers la réalisation des objectifs de ce projet pour le grand public. Ce module fournit les bases de la programmation et des microcontrôleurs basées sur des exemples réels pour rendre son contenu pertinent pour un usage quotidien. L'idée est d'inculquer la logique derrière la programmation et les microcontrôleurs pour promouvoir la culture de la pensée critique, de la créativité et des compétences en résolution de problèmes, entre autres compétences.

Le contexte et l'approche du projet CodER sont présentés dans un premier temps pour permettre aux lecteurs de comprendre la logique de l'approche gamifiée/game-learning adoptée. La première partie du module est consacrée à la compréhension de l'utilisation et des avantages de la programmation, à la configuration des environnements de développement intégrés Python (IDE) et à la découverte des bases du langage de programmation Python pour créer un petit programme. La deuxième partie du module est consacrée à l'introduction sur les microcontrôleurs et à leur utilisation, à la configuration du logiciel Arduino et à l'apprentissage de son utilisation pour exécuter de petites tâches sur des microcontrôleurs.

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

1. Contexte du projet CodER

1.1. Vue d'ensemble : la gamification et son rôle dans l'éducation

Au cours de la dernière décennie, il y a eu un passage des méthodes d'apprentissage traditionnelles centrées sur l'enseignant à ce que l'on appelle parfois des approches de l'éducation centrées sur l'apprenant. Alors que les approches traditionnelles de l'éducation impliquent « le transfert passif des connaissances de l'enseignant à l'étudiant » (McGuire et Gubbins, 2010, p.1), dans les approches centrées sur l'apprenant, les étudiants sont censés prendre conscience et évaluer leur propre expérience (…) où l'enseignant n'est plus un oracle, mais un guide qui participe à l'apprentissage » (McGuire & Gubbins, 2010, p.1). Cette approche contemporaine des méthodes d'apprentissage est généralement davantage basée sur la technologie et implique l'utilisation de stratégies interactives qui visent à motiver les élèves à s'engager et à participer activement à leur processus d'apprentissage.

L'une des raisons de l'adoption de ces approches technologiques de l'éducation, et en particulier de l'enseignement des STEM, est qu'un nombre important d'emplois sont désormais centrés sur des activités et des tâches qui nécessitent l'utilisation d'Internet et des technologies numériques (Daniel Calderón-Gómez et al. 2020), et dans un avenir proche, encore plus d'emplois devront adopter cette structure. Cela signifie que ces emplois nécessiteront l'acquisition d'au moins certaines compétences numériques de base, alors que des emplois plus favorables et mieux rémunérés nécessiteront une connaissance encore plus avancée des technologies numériques (Karpinski et al., 2021). Un autre facteur qui a joué un rôle dans l'évolution vers des approches de l'éducation centrées sur l'apprenant, à l'exception des progrès technologiques rapides, est la corrélation trouvée entre le désengagement, les faibles performances et les faibles taux de participation dans divers contextes éducatifs (tel que cité dans Levels et al., 2022 ; Callanan et al., 2009). Dans le cas des jeunes, ayant entre 15 et 29 ans, cela est encore mis en évidence par les statistiques disponibles en Europe où 13,7 % n'étaient ni à l'école, ni en emploi, ni en formation (NEET) (Eurofound, 2022). Par conséquent, les décideurs politiques et les organisations de jeunesse se sont trouvés confrontés à de nouveaux défis pour réintéresser et réintégrer les jeunes dans l'éducation, l'emploi et/ou la formation.

À différents niveaux d'enseignement, la tendance à aborder l'apprentissage par le biais d'éléments basés sur le jeu dans des contextes d'éducation formelle, non formelle et informelle, de la petite enfance à l'enseignement supérieur et au-delà, est devenue de plus en plus populaire. Dans ce contexte, les définitions de la « gamification » et de « l'apprentissage basé sur le jeu » ont émergé. Bien que les deux termes soient souvent utilisés de manière interchangeable, ils présentent quelques légères différences. Par le terme "gamification", nous nous référons généralement à "l'utilisation d'éléments de

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

conception de jeux dans des contextes non ludiques" (comme cité dans Dichev & Dicheva, 2017, p. 2 ; Deterding, et al., 2011 ; Hamari et al., 2014 ; Werbach, 2014). Dans le contexte de l'éducation, la gamification fait référence à l'utilisation "d'éléments de conception de jeux et d'expériences ludiques dans la conception de processus d'apprentissage" (Dichev & Dicheva, 2017, p.2). Les éléments de conception de jeu consistent en la dynamique, la mécanique et les composants d'un jeu (Dichev & Dicheva, 2017). D'autre part, l'apprentissage basé sur le jeu (GBL) utilise les techniques déployées par les concepteurs de jeux pour créer une expérience de jeu divertissante et immersive pour l'utilisateur, dans le seul but de concevoir un environnement d'apprentissage virtuel qui engage ses utilisateurs dans des activités éducatives. Ce type d'apprentissage peut se produire à la fois physiquement et numériquement. L'apprentissage basé sur le jeu est considéré comme ayant des résultats d'apprentissage clairement définis obtenus grâce au contenu du jeu (Sanchez, 2019). Il est aussi communément appelé jeu sérieux, apprentissage numérique ou jeux éducatifs (Sanchez, 2019).

Il existe une pléthore d'études disponibles qui explorent les effets de l'apprentissage basé sur le jeu chez des individus d'âges différents, des écoliers aux étudiants de l'enseignement supérieur et moins fréquemment chez les adultes (par exemple, Dichev & Dicheva, 2017 ; Ninaus et al., 2017 ; Sanchez et al., 2020). La base de ce mouvement vers l'apprentissage basé sur le jeu se trouve dans son potentiel à motiver les apprenants, en particulier dans la tranche d'âge des jeunes, à s'engager activement et à participer à leur propre éducation. Comme décrit par Dichev & Dicheva (2017), de telles approches d'apprentissage appellent à « l'immersion d'une manière similaire à ce qui se passe dans les jeux » (Dichev & Dicheva, 2017, p.2). Étant donné que les jeux vidéo sont principalement conçus pour le divertissement, "ils peuvent produire des états d'expérience souhaitables et motiver les utilisateurs à rester engagés dans une activité" (Dichev & Dicheva, 2017, p.12). La motivation s'avère être l'un des facteurs les plus importants pour que les apprenants s'engagent et réalisent avec succès toute activité qui nécessite du temps et des efforts. Grâce à l'apprentissage basé sur le jeu, le concept de temps et d'effort est transformé en un environnement éducatif et divertissant qui permet simultanément aux utilisateurs de développer leurs connaissances, leurs compétences et leurs aptitudes.

Le simple fait d'encadrer une activité ou une tâche comme un jeu suffit à provoquer des effets psychologiques positifs tels que l'engagement ou le plaisir (Dichev & Dicheva, 2017). Ce point de vue semble être basé sur l'hypothèse selon laquelle la motivation intrinsèque découle du besoin psychologique fondamental de satisfaire ce "sentiment de compétence ou d'auto-efficacité" (Ninaus et al., 2017, p.16) présent dans la résolution d'un problème ou dans notre contexte, remplissant un objectif de jeu. La capacité des jeux à attirer l'attention des joueurs dans un état d'absorption tout en les engageant dans des tâches de résolution de problèmes découle de leur nature axée sur les objectifs qui place le joueur dans un état d'absorption en essayant d'atteindre les objectifs du jeu. Il existe une dimension pédagogique inhérente aux jeux qui est généralement ignorée, comme le suggère Tulloch

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

(2014), à savoir « le processus des jeux apprenant aux joueurs comment jouer » (comme cité dans Dichev & Dicheva, 2017, p.23). Par conséquent, les méthodes d'apprentissage basées sur le jeu cherchent à imiter la façon dont un jeu attire les joueurs dans un état d'absorption et d'engagement complets afin de créer une expérience tout aussi immersive à des fins éducatives.

1.2. Objectif des méthodes d'éducation par le jeu au codage et aux microcontrôleurs

Depuis le début de la pandémie, l'accent mis sur l'intégration de la technologie dans différentes industries est devenu encore plus important. La vision de la Commission européenne présentée en 2021 englobe quatre points essentiels pour permettre la transformation numérique de l'Europe d'ici 2030. L'un de ces points critiques est la compétence numérique nécessaire pour assurer le développement d'une économie et d'une société européennes résilientes. L'objectif à atteindre d'ici 2030 est que 80 % de la population ait des compétences numériques de base (Commission européenne, 2021a).

Cependant, le plus grand obstacle à la réalisation d'une transformation numérique européenne est le manque de compétences numériques par rapport à leur forte demande. Le pourcentage d'employeurs confrontés à une pénurie d'employés compétents en matière numérique atteint 70 % (Commission européenne, 2021b). Cela élargit le déficit de compétences numériques qui existe sur le marché du travail, où même les professions peu qualifiées à semi-qualifiées nécessitent un niveau de compétence numérique (Karpinski et al., 2021). Comme indiqué dans l'indice de l'économie et de la société numériques (DESI), 56 % des Européens possèdent des compétences numériques de base, mais seulement 31 % possèdent des compétences numériques supérieures au niveau de base (Commission européenne, 2021b). Ainsi, l'importance des compétences en résolution de problèmes et de la pensée informatique est une priorité puisque seuls 13 % des jeunes et 6 % des adultes possèdent de telles compétences dans l'UE (Eurostat, 2020).

Les compétences numériques requises sont traduites dans le "Digital Competence Framework for Citizens" (DigComp) développé par la Commission européenne, qui intègre une variété de compétences : 1) Connaissance de l'information et des données, 2) Communication et collaboration, 3) Création de contenu numérique, 4) Sécurité et 5) Résolution de problèmes (Centeno et al., 2019). Certains considèrent que les nouvelles exigences technologiques des métiers créeront une nouvelle division au sein du marché du travail, représentée par trois catégories de travailleurs du numérique : "le 'précariat numérique' (avec une situation économique dégradée) ; le 'travail numérique traditionnel' (impliquant principalement dans les tâches numériques productives) ; et la « classe innovante » (réalisation de tâches numériques productives et communicatives) » (Calderón-Gómez et al., 2020, p. 7-8). En outre, il a été suggéré que le «précariat numérique» sera principalement composé de jeunes et de femmes (Calderón-Gómez et al., 2020, p.9). Dans cette veine, les jeunes qui sont déjà menacés d'exclusion dans la société et

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

sur le marché du travail en raison de leur inactivité ou de leur incapacité à trouver un emploi ou à poursuivre leurs études et leur formation se trouvent au carrefour de la double exclusion.

Corneliussen (2021) souligne également que le manque de modèles féminins dans ces domaines et le récit dominant des professionnels masculins des TIC découragent activement les jeunes filles et les femmes d'une voie orientée STEM. Dans l'ensemble, les postes les plus favorables sont généralement plus occupés par des hommes que par des femmes (Calderón-Gómez et al., 2020). En même temps, les femmes sont « sous-représentées dans la classe innovante, tout en étant également surreprésentées dans le travail analogique traditionnel » (Calderón-Gómez et al., 2020, p. 8). Une étude montre qu'il est plus fréquent que les hommes passent plus de temps sur Internet pour des activités liées au travail que les femmes, et entre "ceux qui détiennent un diplôme universitaire, 77,5% des hommes contre 70,0% des femmes utilisent Internet au travail » (Calderón-Gómez et al., 2020, p.18-20). En outre, les femmes ont tendance à utiliser davantage les technologies numériques pour la communication et l'utilisation mobile, tandis que les hommes ont tendance à utiliser davantage les technologies numériques à des fins de production et de bureau, et pour de multiples fonctions et utilisations (Calderón-Gómez et al. 2020). En général, par rapport aux hommes, les femmes sont moins susceptibles d'utiliser Internet et/ou les technologies numériques pour des usages plus complexes.

Par conséquent, le codage et les microcontrôleurs sont très pertinents pour les besoins actuels du marché du travail. La combinaison de l'apprentissage basé sur le jeu dans le domaine du codage et des microcontrôleurs nous permet d'atteindre notre groupe cible. Sur la base des avantages évoqués dans la section précédente, l'apprentissage basé sur le jeu fonctionne comme un moyen de transférer des connaissances et des compétences grâce à une expérience d'apprentissage divertissante et significative. Plus précisément, les études qui ont utilisé des salles d'évasion comme outil d'apprentissage ont démontré leur capacité à stimuler les émotions positives, à augmenter le niveau d'engagement et à générer une expérience d'apprentissage globalement positive pour les apprenants (Llerena-Izquierdo & Sherry, 2022 ; Sánchez-Martín et al ., 2020). Les résultats encourageants de ces études ouvrent au projet CodER la possibilité de combler les lacunes en matière de compétences numériques et d'offrir de nouvelles opportunités aux jeunes menacés d'exclusion.

Un autre point très pertinent pour nos efforts de création de salles d'évasion numériques est la limitation des salles d'évasion physiques. Comme le soulignent Fotaris et Mastoras (2019, p.3), "il n'est pas possible ni légal d'enfermer un sous-ensemble d'une classe dans une pièce et d'attendre qu'ils trouvent leur chemin (…), de nombreuses salles d'évasion conçues pour la salle de classe ont été réduites à une activité de table de groupe impliquant une série de boîtes verrouillées ». Mais cette solution n'a pas le caractère immersif typique des salles d'évasion. Il existe également d'autres défis qui doivent être résolus lors de la mise en œuvre de salles d'évasion à des fins éducatives, notamment l'engagement de temps en

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

raison du processus à forte intensité de main-d'œuvre requis pour créer des salles d'évasion, des contraintes de temps qui limitent les tests de jeu, ce qui peut entraîner encore plus de problèmes tels qu'une mauvaise conception et une difficulté déséquilibrée, et enfin, des groupes de grande taille qui compliquent la manière dont les élèves vont participer à la salle d'évasion ou leur engagement (Fotaris & Mastoras, 2019). De ce point de vue, le générateur de salles d'évasion numériques qui sera créé à l'issue de ce projet résout la majorité de ces contraintes et offre aux éducateurs une plus grande flexibilité pour concevoir de tels espaces en fonction des besoins de leurs apprenants.

De plus, Llerena-Izquierdo & Sherry (2022) soulignent que le manque de guides de conception et d'expérience des utilisateurs sont deux facteurs limitants pour une large adaptation de tels outils dans des contextes éducatifs formels, non formels et informels. C'est un autre point important que nous essayons d'aborder avec la création de ce module dans un premier temps, et avec les autres résultats attendus du projet qui doteront les animateurs de jeunesse des connaissances et compétences nécessaires pour inculquer de nouvelles connaissances aux jeunes.

1.3. Le public et le groupe cible de ces activités

Le public auquel le projet CodER s'adresse est composé de membres d'organisations de jeunesse locales, nationales et européennes. Cela inclut tous les types de professionnels travaillant pour une organisation de jeunesse, s'occupant de groupes défavorisés tels que les NEET, les jeunes en décrochage scolaire, les migrants, les réfugiés et les demandeurs d'asile, ainsi que les anciens détenus. En outre, les organisations de jeunesse qui se concentrent sur l'offre d'une éducation non formelle liée à la technologie et à l'innovation, les organisations de jeunesse et/ou les centres de jeunesse qui cherchent à accroître l'employabilité des jeunes et les organisations de jeunesse qui s'occupent de l'éducation STEM font également partie du public cible avec des animateurs de jeunesse compétents dans la programmation, les microcontrôleurs, les jeux d'évasion éducatifs. En outre, le public cible de CodER s'étend aux experts en informatique, aux représentants des start-ups des TIC, aux scientifiques, aux chercheurs, au personnel universitaire, aux influenceurs et autres tels que les associations de codage, les centres d'innovation, les représentants des organismes publics, les petites et moyennes entreprises, les organismes sociaux et sectoriels, les experts en commercialisation de produits et les employés municipaux locaux.

A travers ce public, nous visons à atteindre notre groupe cible, qui est les jeunes, pour leur inculquer les connaissances acquises par les organisations de jeunesse. Le groupe cible comprend les jeunes qui appartiennent aux populations déplacées (migrants, demandeurs d'asile, réfugiés, populations minoritaires), les jeunes qui sont généralement à risque d'exclusion socio-économique (décrochage scolaire, NEET, etc.) et les jeunes ayant des troubles d'apprentissage. Comme il a été mentionné précédemment, un grand nombre de jeunes sont menacés par le chômage en raison d'un manque de compétences en codage que

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

les employeurs recherchent chez leurs employés. Par conséquent, ce projet est conçu de manière à enseigner les bases de la programmation et des microcontrôleurs aux organisations de jeunesse, leur fournir un guide méthodologique et pédagogique sur la façon d'utiliser les salles d'évasion pour éduquer les jeunes à travers une expérience ludique et divertissante, créer une série de différents scénarios à utiliser comme modèles ou à ajuster, et créer un générateur de salle d'évasion numérique.

Références

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.

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

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. 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

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

PARTIE A : Le module de codage CodER (10 heures)

Description:

Dans cette partie, une introduction complète à la programmation est fournie avec son utilisation et son application basée sur des exemples concrets. Les avantages de la programmation et son application au marché du travail sont d'abord expliqués, ainsi que le processus suivi par un développement d'algorithmes et de programmes. Ensuite, la deuxième sous-section explique ce qu'est Python, l'utilisation d'un environnement de développement intégré (IDE) et comment le configurer. Le reste du module est consacré aux différents types de données, à leur usage et application à travers des exemples, ainsi qu'au développement de programmes courts.

Charge de travail:

10 heures

Résultats d'apprentissage :

À la fin de ce cours, les apprenants seront capables de :

- ⇒ Reconnaître la valeur et l'utilisation de la programmation
- ⇒ Comprendre le flux d'exécution dans les programmes
- ⇒ Utiliser la syntaxe de base pour accéder, modifier et supprimer différents types de données sur Python
- ⇒ Utiliser Python pour créer de petits programmes

Matériel et ressources nécessaires :

- ⇒ Ordinateur fixe ou portable
- ⇒ Accès à internet
- ⇒ Spyder
- ⇒ Visual Studio Code

Aspects pratiques:

Cette partie du module est conçue pour couvrir 10 heures d'apprentissage. Chaque section du module a un temps désigné, mais l'apprenant ou l'éducateur/formateur est libre de décider du montant horaire à prévoir pour chaque sous-thème en fonction de ses connaissances antérieures et de son engagement dans des sujets similaires. Le contenu est basé sur un développement progressif et est utilisé pour fournir des connaissances et des compétences de base aux débutants.

Sous-sections :

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

1. Introduction à la programmation
2. Configuration avec Python
3. Les bases de la programmation sur Python

1. Introduction à la programmation

⇒ **Nombre de participants :** 1 à 10 par animateur
⇒ **Durée :** 0,5-0,75 h
⇒ **Méthodes d'enseignement :** Cours magistral, présentation
⇒ **Matériel requis :** Présentation

1.1. Qu'est-ce que la programmation?

La programmation est le processus de création d'un programme dédié à l'exécution d'une tâche spécifique via un ordinateur.

Les programmes comprennent une procédure étape par étape, généralement décrite comme la création d'une recette, qui est utilisée pour communiquer avec l'ordinateur en utilisant une série de syntaxes et de fonctions prédéfinies.

Vous pourriez penser que cela semble trop compliqué, mais ce n'est pas le cas. C'est comme apprendre une nouvelle langue naturelle à partir de zéro, vous devez comprendre sa syntaxe et son vocabulaire pour pouvoir l'utiliser correctement. Un aspect tout aussi important est la pratique de vos compétences pour vous améliorer dans n'importe quelle langue naturelle ou langage de programmation que vous apprenez.

Langue naturelle (Français)	Language de programmation
Jean lit tous les jours	print(“Hello, World”)

Tableau 1 - Comparaison entre langue naturelle et langage de programmation

1.2. Pourquoi apprendre la programmation ? Quels en sont les bénéfices?

Vous vous demandez peut-être quel est le but de l'apprentissage d'un langage de programmation et les avantages que cela implique.

Selon Steve Jobs : "Tout le monde dans ce pays devrait apprendre à programmer un ordinateur car cela vous apprend à penser". Sur la base de cette citation, l'un des principaux avantages de la programmation est qu'elle développe la pensée computationnelle. La pensée computationnelle « fait référence à la capacité d'une personne à identifier, tester et mettre en œuvre des solutions algorithmiques possibles au problème en question et à des problèmes analogues qui pourraient survenir dans un nouveau contexte ou une nouvelle situation » (Karpinski et al., 2021, p. 3). Cela concerne également directement les compétences en résolution de problèmes, qui sont considérées comme des compétences

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

souhaitables et fondamentales pour pouvoir s'adapter aux changements et aux exigences du marché du travail actuel (comme cité dans Karpinski et al., 2021 ; Czaja et Urbaniec, 2019 ; Rakowska et Cichorzewska, 2016 ; Slavinskis et al., 2015). Les compétences en résolution de problèmes sont très appréciées par les employeurs et sont considérées comme l'une des compétences non techniques les plus importantes du 21e siècle.

La capacité à utiliser un langage de programmation ne signifie pas que tout le monde doit devenir programmeur, mais que la programmation peut être bénéfique pour n'importe quel métier et peut ouvrir de nouvelles voies de carrière.

La flexibilité que permet la programmation est un autre de ses atouts majeurs puisque de nouvelles opportunités d'emploi vont émerger, plus orientées vers le codage. En plus de cela, des connaissances de base en programmation seront nécessaires pour tout travail futur. Cela augmente sa valeur en tant que compétence fondamentale pour la société et le marché du travail de demain.

Toute personne qui stagne dans sa carrière peut apprendre un langage de programmation pour augmenter ses revenus et ses perspectives de carrière, ainsi que pour approfondir ses compétences en résolution de problèmes. Par conséquent, la programmation est une compétence globale qui nécessite à la fois des compétences techniques et générales et peut être apprise par quiconque est prêt à essayer.

1.3. Le processus de développement du programme et des algorithmes

Si vous lisez encore ce module, laissez-nous vous expliquer ce qu'est un algorithme et comment un programme est développé.

Les algorithmes font partie intégrante des programmes car ils représentent les étapes nécessaires pour accomplir la tâche spécifiée par le code. Un bon algorithme doit inclure les éléments suivants : les étapes ou les activités nécessaires pour accomplir la tâche, l'ordre correct de ces activités et leur fin. Un exemple fréquent utilisé pour comprendre le processus d'un algorithme est une recette de cuisine. La recette suivante de « Bake a Cake » peut être traduite en algorithme :

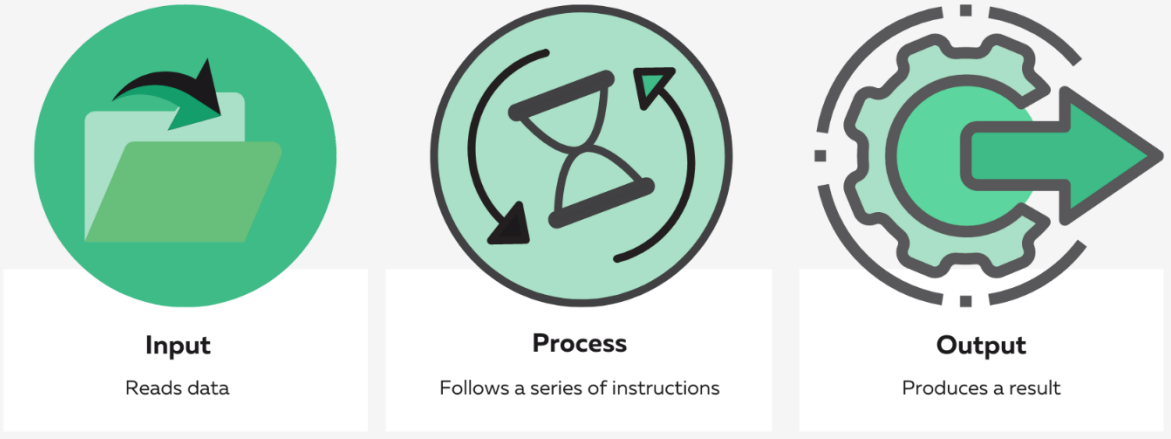
Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Image 1 - Représentation visuelle d'un algorithme

Comme vous le voyez, l'ordre de chaque activité ou étape est basé sur un ordre logique. Par exemple, vous ne pouvez pas placer la casserole dans le four sans d'abord mettre les ingrédients dans la casserole ; pas si vous voulez avoir un résultat fructueux. Il en va de même pour les algorithmes, chaque étape ou activité a un but à servir, et elles doivent avoir un ordre approprié pour que l'algorithme fonctionne comme nous le voulons. La description ci-dessus est également appelée pseudo code et elle est utilisée pour expliquer le processus suivi par un algorithme.

Un autre aspect important est le modèle suivi pour concevoir un algorithme spécifique, qui implique généralement le modèle d'entrée, de processus et de sortie (IPO). L'algorithme commence par lire les données, passe à la manipulation des données et se termine par l'affichage d'un résultat. Par exemple, supposons que vous souhaitiez calculer les notes de vos étudiants : vous devez d'abord parcourir leurs notes à chaque examen (entrée), puis calculer leur note moyenne (processus) et enfin, afficher leur note moyenne (sortie). C'est un modèle très utile à considérer lors de la création d'un algorithme.



Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Image 2 - Modèle d'introduction en bourse

Puisque nous avons appris ce qu'est un algorithme et le processus requis pour créer un bon algorithme, nous pouvons maintenant comprendre un peu mieux comment développer un programme. Le développement du programme passe par un cycle qui comprend l'analyse, la conception, le codage, le débogage et les tests, ainsi que la mise en œuvre et la maintenance de la solution proposée.

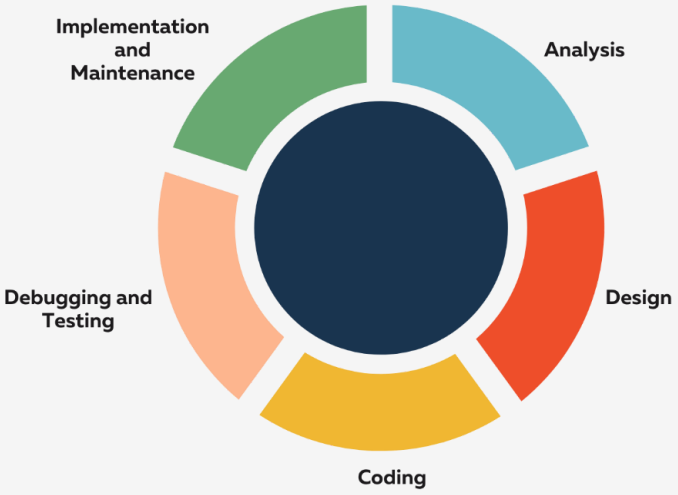


Image 3 – Cycle de développement du programme

Dans la première étape, nous essayons de comprendre, d'identifier et d'analyser le problème qui nécessite une solution informatique. La deuxième étape consiste à concevoir le programme, ce qui implique de créer un diagramme visuel du processus que le programme suivra. Ceci est particulièrement utile pour vous aider à diviser le problème en parties plus petites. L'étape suivante consiste à coder le programme en fonction du diagramme visuel que nous avons créé à l'étape précédente. L'utilisateur de l'ordinateur exécutera le code pour repérer tout problème éventuel. Dans la quatrième étape, l'utilisateur doit commencer à déboguer le programme afin d'éviter les erreurs potentielles. La cinquième étape demande à l'utilisateur d'exécuter le programme et de s'assurer qu'il n'y a pas d'erreurs syntaxiques ou logiques. La dernière et sixième étape tourne autour de la documentation et de la maintenance du programme, ce qui nécessite une explication de la logique du programme et de ses processus.

S'il y a des choses que vous n'avez pas comprises dans ce processus, ne vous inquiétez pas. Une fois que nous commencerons à pratiquer, cela deviendra plus clair. Dans ce module, nous nous concentrerons sur les trois premières étapes. Par conséquent, nous essaierons d'expliquer le processus d'analyse d'un problème et de développement d'un programme pouvant servir de solution potentielle.

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

1.4. Langages de programmation - Les langages de programmation les plus demandés

Selon une analyse de milliers d'offres d'emploi aux États-Unis et en Europe, Python a été classé comme le langage informatique le plus demandé en 2022, suivi de Java et JavaScript. Bien que les résultats suggèrent que la demande pour C, C++ et C# n'était pas aussi forte, elle était toujours présente pour certaines professions. Bien que Python existe depuis des décennies, sa demande en 2022 continuera de croître grâce à son utilisation exponentielle dans les industries florissantes de la science des données, de l'apprentissage automatique et de l'IA.

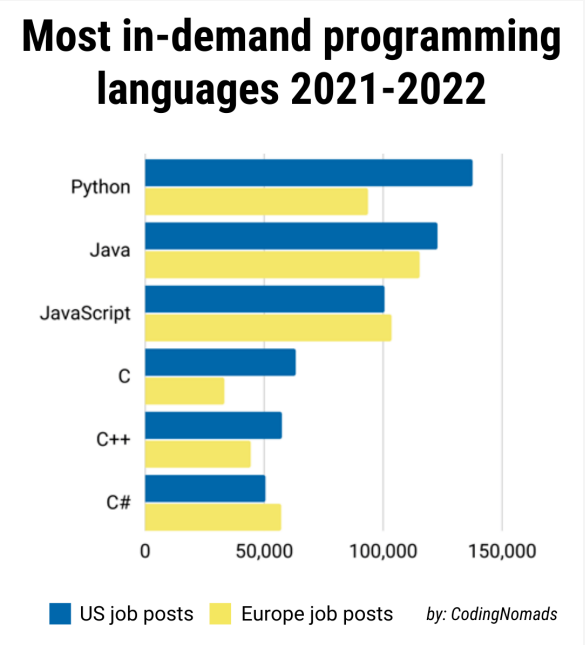


Image 4 - Langages de programmation les plus demandés
Source: https://www.siliconrepublic.com/careers/python-most-in-demand-coding-language-2022

2. Obtenir la configuration avec Python

- ⇒ **Nombre de participants :** 1 à 10 par animateur
- ⇒ **Durée :** 0,75-1h
- ⇒ **Méthodes d'enseignement:** Présentation, Instruction guidée, Apprentissage expérientiel
- ⇒ **Matériel requis :** présentation, ordinateurs (1 par participant) et connexion Internet stable

Avant d'expliquer comment configurer un environnement de développement intégré (IDE) Python et comment exécuter Python dans une ligne de commande, nous expliquerons brièvement ce qu'est Python et pourquoi c'est le langage de programmation le plus utilisé sur le marché du travail.

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

2.1. Qu'est-ce que Python ?

Python est un langage de programmation de haut niveau qui permet une plus grande flexibilité et lisibilité lors du codage et qui peut être facilement adapté à différents types d'ordinateurs avec peu ou pas de modifications potentielles. Les autres langages de haut niveau incluent C, C++ et Java.

La différence entre les langages de programmation de haut niveau et de bas niveau est que les ordinateurs exécutent des programmes écrits dans des langages de bas niveau. Les langages de bas niveau utilisent le codage binaire (0,1) pour exécuter un programme. Cependant, ce n'est pas facile à interpréter pour un humain à moins qu'il n'ait l'expertise. Par conséquent, les programmes écrits dans des langages de haut niveau doivent être transformés ou traduits sous une forme binaire pour que l'ordinateur comprenne ce qu'il faut faire. Cela nécessite un temps de traitement supplémentaire, mais c'est un inconvénient relativement mineur des langages de haut niveau.

Généralement, il existe deux manières de traduire un programme sous forme binaire pour l'exécuter. La première consiste à utiliser un compilateur où le programme est lu et traduit avant son exécution. Comme illustré dans l'image ci-dessous, le processus part du code source (pour un programme de haut niveau) passe au code objet ou exécuteur (qui traduit le programme sous forme binaire) et revient pour compiler le programme à exécuter en sortie sans autre traduction.

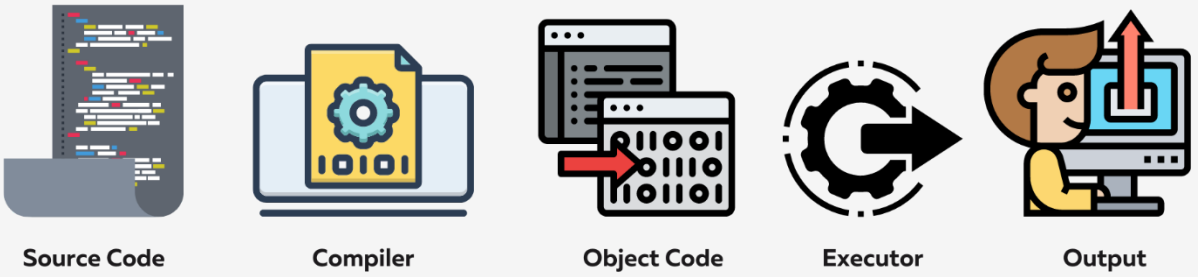


Image 5 - Langage du compilateur (Adapté de Downey et al., 2002, p.30)

La deuxième méthode utilise uniquement un interpréte comme médiateur entre le code source et la sortie. L'interprète lit essentiellement le programme et suit ses instructions ligne par ligne. Il s'agit d'un processus continu de va-et-vient jusqu'à ce que le programme soit terminé, comme vous pouvez le voir sur l'image ci-dessous.

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Image 6 - Langage de l'interprète (Adapté de Downey et al., 2002, p.30)

Python utilise l'interpréteur pour exécuter ses programmes et est donc considéré comme un langage interprété. Cela peut être fait en mode ligne de commande ou en mode script. En mode ligne de commande, vous écrivez votre code et l'interpréteur imprime le résultat final. D'autre part, le mode script utilise un programme qui contient un fichier se terminant par .py et utilise l'interpréteur pour exécuter le contenu du fichier. Il convient de noter que la plupart des programmes Python se terminent par .py, cependant, cela dépend de l'environnement de développement intégré (IDE) que vous utilisez.

Dans ce module, tous les exemples utilisent le mode ligne de commande pour exécuter des programmes instantanément et mieux comprendre le processus. Une fois que vous avez terminé un programme, vous pouvez l'enregistrer pour pouvoir l'exécuter ultérieurement en tant que script.

2.2. Les environnements de développement intégrés (IDE) Python les plus utilisés par secteur

Dans la section précédente, nous avons mentionné les environnements de développement intégrés (IDE), expliquons brièvement ce qu'est un IDE. Un IDE est essentiellement une application logicielle qui contient diverses fonctionnalités permettant aux développeurs informatiques de développer leurs logiciels, tels qu'un éditeur de code source, des outils d'automatisation de construction ou un débogueur. Si tout cela ne semble pas avoir beaucoup de sens en ce moment, ne vous inquiétez pas. Une fois que nous commencerons à pratiquer, vous aurez une meilleure compréhension de tout ce que nous avons expliqué jusqu'à présent.

Selon l'industrie, différents IDE Python sont préférés. Certains des facteurs pris en compte lorsqu'une entreprise ou une organisation choisit un IDE sont : l'utilisation prévue de l'IDE, telle que le développement Web, la science des données, les scripts ou l'assurance qualité ; le système d'exploitation qu'il utilise (c'est-à-dire Linux/macOS, Windows ou système d'exploitation mixte) ; si le matériel est bon ou mauvais ; et le niveau de compétences de la personne qui codera (par exemple, débutant, intermédiaire ou avancé).

Image 7 - Most used IDEs per industry

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

(Adapté de https://www.geeksforgeeks.org/top-10-python-ide-and-code-editors-in-2020/)

2.3. Configuration d'un environnement de développement intégré Python (IDE)

Dans ce module, vous pouvez utiliser Spyder (https://www.anaconda.com/products/individual) ou Visual Studio Code (https://code.visualstudio.com/) car ils sont généralement utilisés par les débutants pour se familiariser avec Python dans un environnement convivial.

Cependant, si vous n'êtes pas prêt à vous engager dans un IDE, vous avez également la possibilité d'utiliser des éditeurs de texte en ligne tels que les suivants :

- Python.org
- Programiz
- Tutorialspoint

Nous vous recommandons d'utiliser un IDE, car il est plus facile de suivre vos fichiers et votre progression lorsque vous commencez à coder.

2.3.1 Installation du code Visual Studio

1. Allez sur ce site: https://code.visualstudio.com/
2. Une fois la page chargée, vous verrez le bouton Télécharger. Il choisit automatiquement votre système d'exploitation ; cependant, vous pouvez cliquer sur la flèche à côté et la modifier en fonction de vos besoins. Assurez-vous de télécharger le “stable build”!

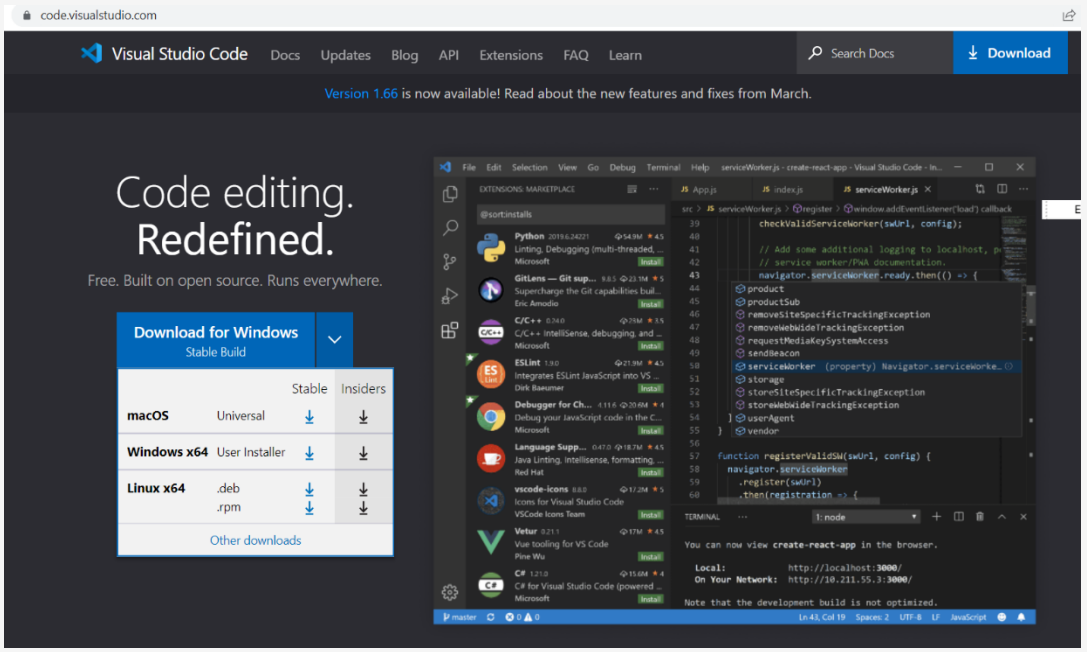


Image 8 – Téléchargement du code Visual Studio

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

3. Lorsque le téléchargement est terminé, ouvrez le fichier et suivez les instructions d'installation.

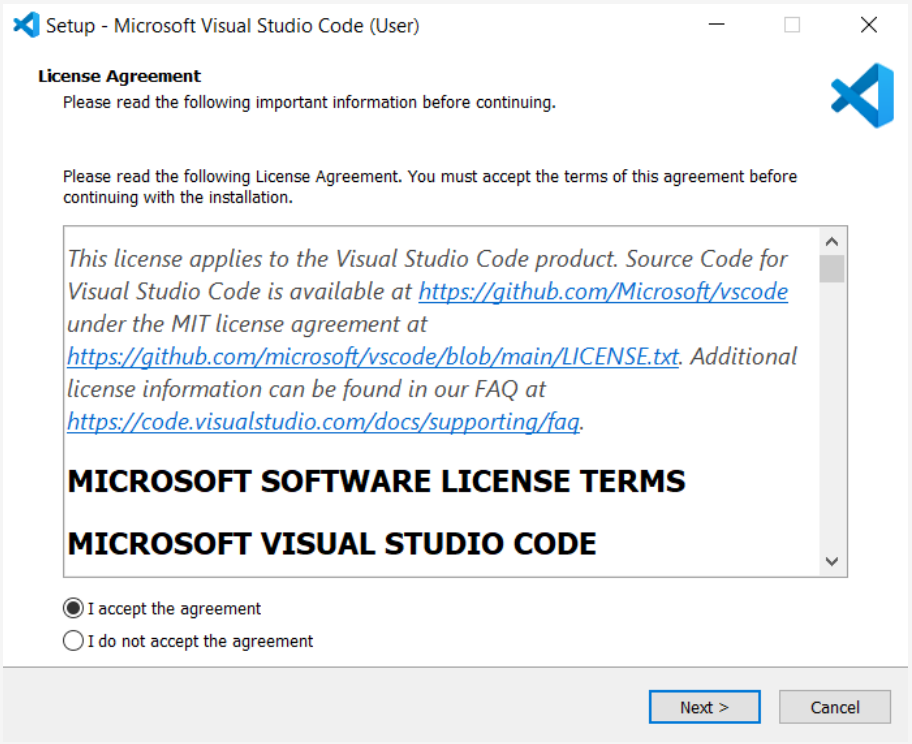
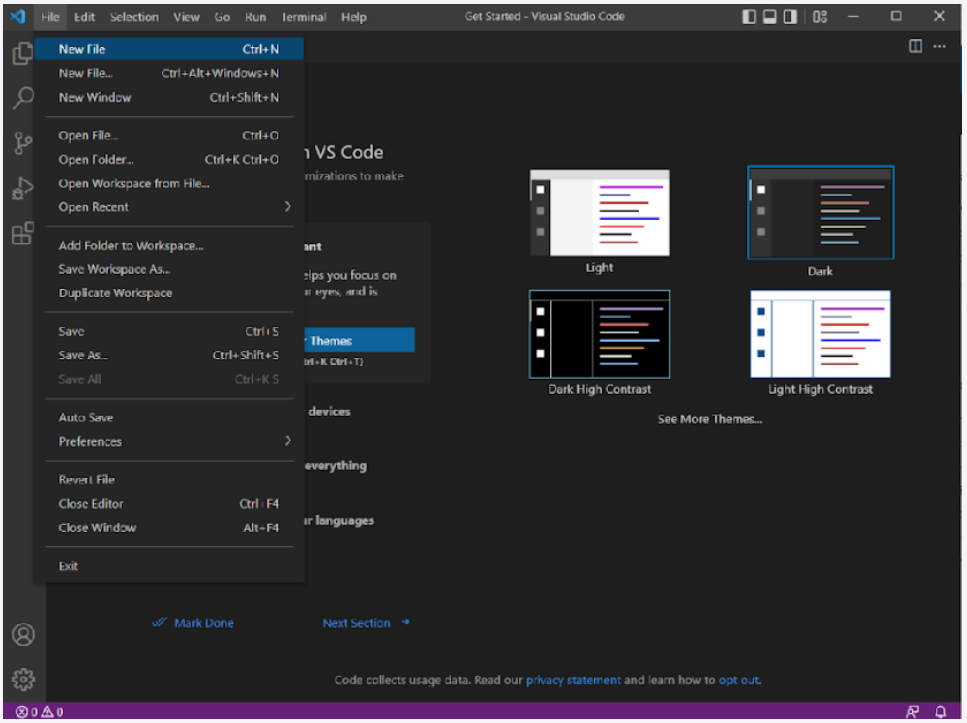


Image 9 – Installation de l'IDE

4. Une fois l'installation du programme terminée, vous pouvez l'ouvrir ou il se lancera automatiquement.

5. Sélectionnez Fichier Nouveau



Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Image 10 – Création d'un nouveau fichier dans Visual Studio Co

6. Cliquez sur sélectionner une langue et choisissez Python

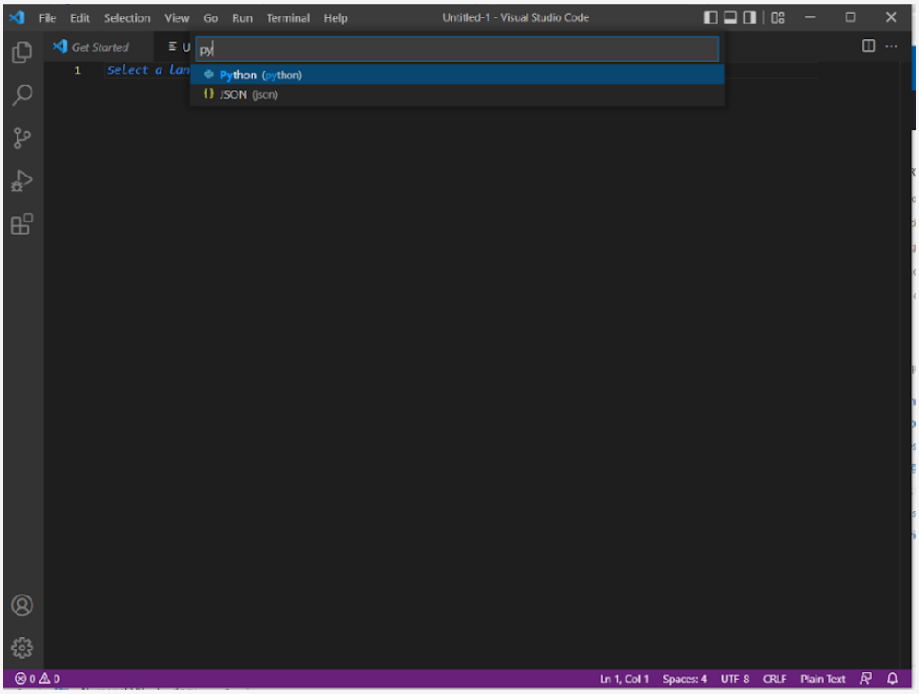


Image 11 – Sélection du langage de programmation

7. Une fois que vous avez sélectionné Python, vous serez redirigé pour installer Python.

Image 12 – Installation de Python à partir d'extensions

8. Une fois que vous avez installé Python, vous êtes prêt à commencer à coder !

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Pour plus d'informations sur la façon de commencer, vous pouvez également consulter leur site Web : https://code.visualstudio.com/docs/introvideos/basics

2.3.2. Installation de Spyder depuis Anaconda

1. Allez sur ce site: https://www.anaconda.com/products/individual
2. Une fois la page chargée, vous verrez le bouton Télécharger. En fonction de votre système d'exploitation, choisissez la version correspondante. Nous recommandons le programme d'installation graphique 64 bits.

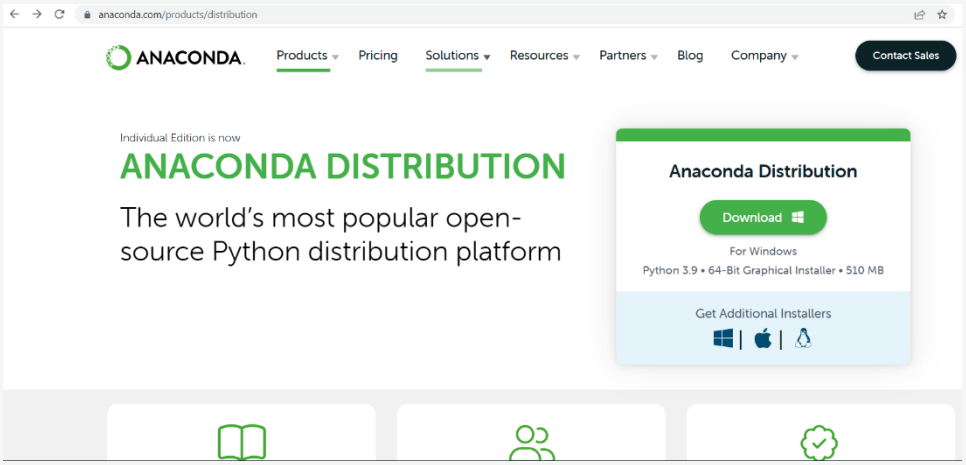


Image 13 – Téléchargement de la distribution Anaconda

   a. Pour plus d'informations sur les étapes requises pour Windows, suivez ce lien: https://docs.anaconda.com/anaconda/install/windows/
   b. Pour les utilisateurs de macOS, suivez ce lien : https://docs.anaconda.com/anaconda/install/mac-os/
   c. Pour les utilisateurs de Linux, suivez ce lien : https://docs.anaconda.com/anaconda/install/linux/

3. Une fois le téléchargement terminé, ouvrez le fichier et suivez les instructions de configuration.

Image 14 – Installation d'Anaconda

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

4. Une fois l'installation du programme terminée, vous pouvez ouvrir le navigateur Anaconda.
5. Une fois le navigateur chargé, sélectionnez l'application Spyder et cliquez sur Lancer. S'il n'est pas installé, cliquez sur le bouton Installer pour installer Spyder.

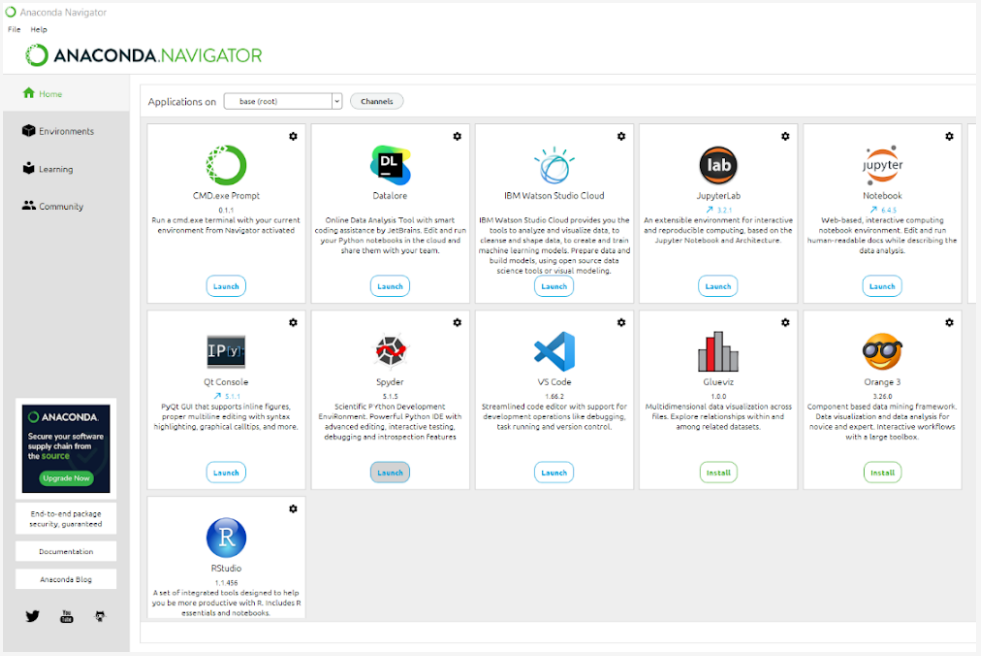


Image 15 – Navigateur Anaconda

6. Lorsque vous ouvrez Spyder, vous pouvez choisir de faire une visite guidée ou de l'ignorer.

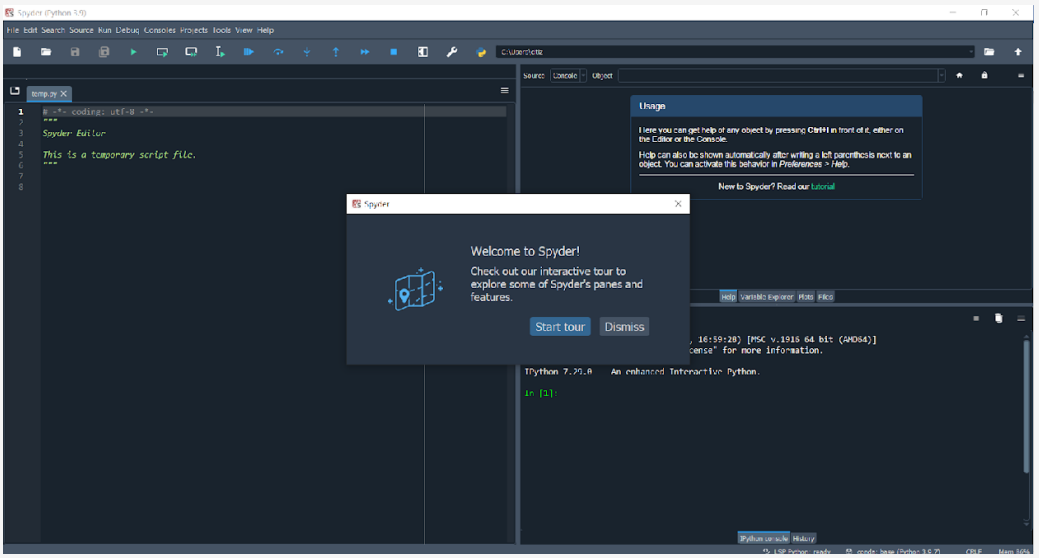


Image 16 – Ouvrir Spyder pour la première fois

7. Désormais vous pouvez commencer à coder !

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Pour plus d'informations, vous pouvez également consulter le starter Anaconda : https://docs.anaconda.com/anaconda/user-guide/getting-started/

Nous sommes sûrs que vous avez pu compléter cette étape facilement !

2.4. Exécuter Python en ligne de commande

Il est temps maintenant d'exécuter notre premier programme. Vous vous demandez peut-être si c'est si simple. La réponse est : oui, ça l'est !

Tapez simplement ce qui suit :

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

Maintenant, vous pouvez l'exécuter. Il devrait afficher sur la console : Hello, World !

3. Les bases de la programmation sur Python

- ⇒ **Nombre de participants :** 1 à 10 par animateur
- ⇒ **Durée:** 8.00h
- ⇒ **Méthodes d'enseignement:** Présentation, Instruction guidée, Apprentissage expérientiel
- ⇒ **Matériel requis :** présentation, ordinateurs (1 par participant) et connexion Internet stable

Après avoir exécuté avec succès votre premier programme, nous continuerons à en apprendre davantage sur les fonctions de base de Python et la logique qui les sous-tend.

3.1. Les bases : types de données (de base et complexes)

Les types de données sont un concept important dans le monde de la programmation car ils déterminent comment les données peuvent être manipulées. Les types de données intégrés dans Python appartiennent aux catégories suivantes :

Groupes généraux de types de données	Types de données spécifiques utilisés sur Python	Exemples
Texte	str	“Hello, World!” (une série de chaînes, utilisez toujours des guillemets)
Numérique	int, float, complex	3 (integer-entier), 3.2 (float – inclut les décimales), 2j+3 (comprend à la fois des caractères et des chiffres)

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Séquence	list, tuple, range	[“pomme”, “cerise”, 1, 1] (liste; séquence d'éléments de n'importe quel type), () (tuple; collection ordonnée), range(5) (itère sur une séquence de nombres qui a un point de départ et une fin(5))
Mappage	dict	{"marque": "Ford"} (dictionnaire; collection non ordonnée de paires clé-valeur)
Set	set, frozenset	{“pomme”, “cerise”, 1} (collection non ordonnée d'éléments sans doublon de tout type séparés par des virgules)
Booléen	bool	Vrai ou faux
Binary	bytes, bytearray, memoryview	utilisé pour manipuler des données binaires

Tableau 2 – Les différents types de données de Python avec des exemples (adapté de https://www.w3schools.com/python/python_datatypes.asp)

Ce sont tous les types de données intégrées qui existent dans Python. Nous passerons en revue tous ces éléments sauf le dernier groupe, binaire, car les autres sont les plus utiles pour les débutants à comprendre.

3.2. Variables, expressions et instructions

Variables

Les *variables* contiennent essentiellement des données que nous leur attribuons sous forme de noms pour rendre notre code plus lisible et pouvoir utiliser une variable particulière plus d'une fois.

Nous écrivons le nom que nous voulons attribuer à notre variable, utilisons le signe égal (=) et mettons la valeur que nous voulons stocker à droite du signe égal.

Exemple:

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

Ici, la variable x est égale à 5 et print(x) imprime la valeur contenue dans x (c'est-à-dire 5).

***Gardez à l'esprit que Python est sensible aux majuscules et minuscules, donc x et X ne seront pas considérées comme la même variable.**

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Rappelez-vous des types de données que nous venons de voir, vous pouvez écrire ce qui suit pour voir le type de données de la variable x que nous venons de créer :

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

La console affichera : **int**

Supposons que vous souhaitiez créer 3 variables dont l'une contient un nombre entier, une autre un nombre décimal et une autre une chaîne. Vous écririez ceci :

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

Si nous essayons de l'exécuter, la console n'affichera rien. Vous vous demandez peut-être pourquoi, c'est simplement parce que nous n'avons pas appelé la variable, mais nous lui avons seulement attribué une valeur. Pour voir ce que contient notre variable, nous devons utiliser print (nom de la variable) pour demander à Python de l'afficher. Sinon, ce ne sera pas le cas.

Exemple:

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

Ou vous pouvez les afficher tous ensemble comme ceci :

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

Nous pouvons donner n'importe quel nom à notre variable, et nous ne sommes pas limités à un seul mot.

Cependant, vous ne devez pas utiliser des noms de variables trop longs car cela n'est pratique ni pour vous ni pour le lecteur.

Essayez de garder les noms des variables courts et compréhensibles. Si vous souhaitez que votre nom de variable soit composé de deux mots distincts, utilisez simplement un tiret bas entre les deux, c'est-à-dire *mon_âge*.

*** Gardez à l'esprit que Python n'autorise pas les noms de variables à commencer par des chiffres ou à inclure des caractères spéciaux tels que @, #, $, etc.**

Il existe également d'autres mots clés utilisés en Python qui ne peuvent pas être utilisés comme noms de variables, car ils sont réservés à des fonctions spécifiques. Ces mots-clés sont :

and	def	exec	if	not	return
assert	del	finally	import	or	try

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

break	elif	for	in	pass	while
class	else	from	Is	print	yield
continue	except	global	lambda	raise	

***Tableau 3** - Mots clés réservés en Python (Adapté de Downey et al., 2002, p. 15)*

Instructions

Nous avons vu deux instructions utilisées en Python jusqu'à présent : print et affectation de variable. Les commandes données à l'interprète Python pour être exécutées sont appelées des instructions.

Comme nous l'avons déjà mentionné, les instructions, telles que l'affectation de variables, ne produisent pas de résultat. Cependant, l'instruction print produit un résultat, qui est la valeur de la variable.

Une séquence d'instructions est considérée comme un script qui produit ou non des résultats, selon le type d'instructions utilisées. Les résultats sont affichés en fonction de l'ordre des instructions.

Par exemple, si vous saisissez les instructions suivantes :

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

Vous obtiendrez en sortie :

John

35

La variable de hauteur est uniquement enregistrée en tant que variable mais ne produit aucun résultat.

Expressions

Il existe différents types d'expressions utilisées en Python. Les expressions impliquent une combinaison de différentes variables, opérateurs et valeurs (opérandes) qui produisent une autre valeur en sortie.

Par exemple, essayons d'additionner deux nombres sans leur attribuer de noms de variables ou utilisons l'instruction *print* :

```
2+2
```

Production:

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Ici, nous pouvons voir que Python ne produira pas de sortie. Il est donc nécessaire d'avoir des variables, des opérateurs et des valeurs pour avoir une expression.

Si vous souhaitez obtenir un résultat de 2 + 2, vous devez soit inclure l'instruction d'affichage, soit attribuer un nom de variable et l'imprimer.

Exemple:

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

Dans l'exemple de 2+2, nous demandons à Python d'évaluer l'expression. Mais si nous écrivions simplement 2 et 2 sur des lignes séparées sans l'opérateur (+), aucun résultat ne serait produit. Même si la déclaration est considérée comme légale, elle ne produit pas de sortie.

Considérez le script suivant :

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

Ici, la seule sortie que vous obtiendriez est *5* de la dernière ligne de 3+2. Que pouvons-nous ajouter au script afin d'afficher toutes les expressions sur la console en sortie ?

Indice : nous l'avons déjà utilisé plusieurs fois maintenant

Operateurs

Il est temps d'expliquer certains des opérateurs utilisés en Python, comme le signe d'addition (+) que nous avons utilisé plus tôt. Les opérateurs sont les symboles spéciaux qui représentent les calculs mathématiques, qui sont les suivants :

addition (3+20) **soustraction (x-6)** **multiplication (3*5)** **division (10/5)** **exponentiation (4**2)**

Vous pouvez également utiliser des noms de variables contenant des nombres entiers ou décimaux, au lieu d'entiers simples pour effectuer des opérations.

Exemple:

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

Sortie : 33

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Pour trouver la note moyenne basée sur les notes des 2 cours, nous devons les additionner puis les diviser par 2. Comment pensez-vous que cela peut être fait ? Prenez une minute pour y réfléchir.

Réponse:

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

Sortie: 16.5

Cet exemple montre également que Python reconnaît l'ordre des opérations. La parenthèse a la priorité la plus élevée au lieu de la division située à l'extérieur de la parenthèse.

En Python, la priorité est la suivante : 1) parenthèse, 2) exponentiation, 3) multiplication/division et 4) addition/soustraction.

A noter que lorsque les opérateurs ont la même priorité, ils sont évalués de gauche à droite. Par exemple, considérons l'opération suivante : 5*30/60. Cela équivaut à 2,5, ce qui montre que la multiplication a lieu en premier (=150) puis la division qui donne le résultat final.

Entraine toi:

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

Opérateurs qui fonctionnent sur des chaînes

Même si une chaîne est considérée comme un nombre (c'est-à-dire '15'), vous ne pouvez généralement pas effectuer les opérations mentionnées ci-dessus sur les chaînes.

Cependant, les opérateurs d'addition (+) et de multiplication (*) ont un effet différent lorsqu'ils sont appliqués sur des chaînes. Si vous utilisez l'addition entre deux variables contenant des chaînes, elles seront concaténées. Cela signifie qu'elles seront jointes en une séquence de chaînes.

Exemple:

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

Sortie: Emma Smith

***Gardez à l'esprit que si les guillemets et les chaînes ont un espace entre eux, vous finirez par "EmmaSmith" au lieu de "Emma Smith". Par conséquent, les espaces font également partie des chaînes.**

Lorsqu'il est utilisé sur des chaînes, l'opérateur de multiplication (*) effectue une répétition.

Exemple:

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

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Sortie: FirstFirstFirst

Pour que cela fonctionne, vous devez utiliser un nombre entier pour spécifier le nombre de répétitions de la chaîne.

Commentaires

Au fur et à mesure que les programmes se développent, il devient plus difficile de retrouver et de donner un sens aux algorithmes utilisés pour produire un résultat final. Ainsi, les commentaires sont utiles pour expliquer la logique et le processus du programme en langage naturel. Pensez aux commentaires comme l'ajout de notes partout où cela est nécessaire pour rendre votre code plus lisible, il vous suffit d'ajouter un symbole hashtag (#) pour les petits commentaires.

Exemple:

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

Vous pouvez également ajouter un commentaire sur la même ligne du code :

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

Tout ce que vous écrivez après le symbole hashtag (#) est ignoré et n'est pas exécuté en tant que code par le programme.

Une autre option qui peut être utilisée pour de plus grandes parties de texte consiste à ajouter 3 guillemets (""") avant et 3 (""") après la fin du texte.

Exemple:

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

L'utilisation de commentaires est une pratique utile pour tout professionnel qui travaille avec le codage.

Entraine toi:

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

3.3. Listes, dictionnaires et tuples

Dans cette section, nous verrons comment utiliser des listes, des dictionnaires et des tuples en Python. Comme nous l'avons vu précédemment dans la section sur les types de données, la liste et les tuples appartiennent au groupe de séquences et les dictionnaires au groupe de mappage. Tous ces types de données ont des caractéristiques et des qualités différentes qui permettent la manipulation des données de différentes manières.

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

3.3.1. Listes

Les listes sont initiées avec l'utilisation de crochets [ ] et contiennent une séquence d'éléments ordonnés dans une variable.

Exemple:

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

Les éléments inclus dans une liste sont ordonnés, modifiables et autorisent les valeurs doubles.

Chaque élément d'une liste est identifié par un index. L'index commence à partir de [0] et augmente de un à chaque fois ; ainsi, le premier élément de la liste a un index [0] et le second [1] et ainsi de suite. Il est à noter que leur ordre ne peut pas être modifié. Lorsque de nouveaux éléments sont ajoutés, ils seront placés à la fin de la liste.

De plus, les listes sont modifiables, ce qui signifie que les éléments peuvent être modifiés, supprimés ou ajoutés après la création de la liste. Puisqu'ils autorisent les valeurs doubles, vous pouvez trouver une valeur plusieurs fois dans une liste.

Exemple:

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

Les listes peuvent également contenir une combinaison d'entiers, de chaînes ou de valeurs booléennes :

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

Si vous voulez déterminer le nombre d'éléments contenus dans une liste, vous pouvez utiliser la fonction len() :

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

*** Gardez à l'esprit que le nombre de parenthèses doit correspondre du début à la fin pour que le code soit exécuté.**

Si vous regardez le code dans cet exemple, nous en avons une devant la fonction len() et une dans la liste, c'est pourquoi vous voyez deux parenthèses fermantes à la fin du code.

Vous pouvez également utiliser la fonction list() pour lancer une liste :

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

*** Notez qu'ici vous devez ajouter une double parenthèse lorsque vous utilisez la fonction list() pour créer une liste.**

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Accéder aux éléments des listes

Si vous souhaitez accéder à un élément d'une liste, vous devez utiliser son numéro d'index.

Utilisons la liste employee1 que nous avons créée précédemment pour obtenir l'âge de l'employé (deuxième élément de la liste) dans l'exemple suivant :

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

*** Vous rappelez-vous pourquoi nous utilisons 1 au lieu de 2 comme numéro d'index pour le deuxième élément de la liste ? C'est parce que l'index commence à partir de 0 dans les listes.**

Si vous avez une longue liste d'éléments et que vous souhaitez obtenir le dernier élément de la liste, vous pouvez utiliser l'indexation négative. Cela signifie qu'on utilisera -1 pour accéder au dernier élément trouvé dans la liste, -2 pour obtenir l'avant-dernier élément, et ainsi de suite.

Exemple:

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

Vous pouvez également spécifier une plage d'index qui indique à partir de quel numéro d'index commencer et à quel numéro d'index terminer la plage.

Exemple:

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

Dans cet exemple, il commencera à partir du troisième élément de la liste et se terminera au dernier puisque nous n'avons que 5 éléments sur la liste. Rappelez-vous toujours que la numérotation commence à partir de 0.

Si vous décidez que vous ne souhaitez pas spécifier le numéro d'index de début, la plage commencera à partir du premier élément trouvé dans la liste :

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

Vous pouvez également spécifier uniquement le numéro d'index de début sans numéro d'index de fin :

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

N'oubliez pas que lorsque nous avons introduit l'indexation négative, vous pouvez également l'utiliser pour accéder aux éléments d'une liste à partir de la fin de la liste :

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

Cela nous donnera le quatrième élément de la fin jusqu'au dernier trouvé dans la liste.

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Vous pouvez également vérifier si un élément existe dans la liste. À partir de la liste d'adjectifs que nous avons créée précédemment, disons que nous voulons vérifier si *beautiful* est inclus dans la liste :

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

Ajouter des éléments aux listes

Si vous souhaitez ajouter de nouveaux éléments à la fin de la liste, vous pouvez utiliser la méthode append(). Disons que nous voulons ajouter un nouvel adjectif à la liste des adjectifs, nous ferons ce qui suit :

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

Vous avez également la possibilité d'ajouter de nouveaux éléments en spécifiant le numéro d'index. Vous pouvez le faire en utilisant la méthode insert() :

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

Comme vous pouvez le voir ici, vous spécifiez d'abord le numéro d'index, puis l'élément que vous souhaitez ajouter séparé par une virgule.

Une autre méthode qui peut être utilisée lors de l'ajout d'éléments d'une autre liste à la liste actuelle est extend(). Disons que nous avons une liste générale de mots et que nous voulons ajouter les adjectifs dans la liste générale de mots que nous avons :

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

Modifier les listes

Si vous souhaitez modifier un élément spécifique présent dans une liste, vous devez vous référer à son numéro d'index. A titre d'exemple, nous allons utiliser la liste employee1 pour modifier l'âge de l'employé :

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

Vous pouvez également modifier plusieurs éléments d'une liste dans une plage spécifiée. Vous pouvez le faire en définissant les nouveaux éléments à ajouter et en vous référant aux numéros d'index que vous souhaitez modifier.

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Par exemple, nous allons modifier les trois premiers éléments de la liste employee1, car cet employé ne travaille plus dans cette entreprise :

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

Supprimer des éléments des listes

Si vous avez décidé de supprimer un élément spécifique d'une liste, vous pouvez utiliser la méthode remove(). Disons que nous voulons supprimer la valeur booléenne de la liste employee1 :

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

Vous pouvez également supprimer un élément en spécifiant son numéro d'index avec la méthode pop() :

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

Si vous ne spécifiez pas de numéro d'index, le dernier élément trouvé dans la liste sera supprimé :

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

Comme alternative, vous pouvez utiliser le mot-clé *del* pour supprimer un élément avec un index spécifié :

```
del employee1[1]
```

Vous pouvez également utiliser *del* pour supprimer toute la liste :

```
del employee1
```

Cependant, vous souhaiterez peut-être vider le contenu de la liste. Dans ce cas, vous pouvez utiliser la méthode **clear()** :

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

Copier des listes

Il existe deux manières de copier une liste avec un nouveau nom de variable. La première consiste à utiliser la méthode copy() :

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

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

La deuxième façon dont cela peut fonctionner est en utilisant la méthode list() :

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

Trier les listes

Si vous souhaitez trier une liste par ordre alphanumérique, décroissant ou croissant, vous pouvez utiliser la méthode sort().

***Notez que la méthode sort() listera les éléments dans l'ordre croissant par défaut, sauf indication contraire.**

Utilisons la liste de mots pour la trier par ordre alphabétique dans cet exemple :

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

Un autre exemple qui inclut une liste numérique :

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

Ces deux exemples seront répertoriés par ordre croissant, c'est-à-dire du plus petit au plus grand ou de a à z selon les types de données.

Si vous souhaitez trier une liste par ordre décroissant, vous devez le spécifier entre parenthèses en utilisant le mot-clé reverse = True. Rappelez-vous quand nous avons mentionné les booléens comme types de données. Dans ce cas, ils sont utilisés pour activer l'option inverse en indiquant qu'elle est vraie puisque la valeur par défaut est fausse.

Exemple 1:

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

Exemple 2:

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

Le tri est par défaut sensible aux majuscules et aux minuscules, ce qui signifie que toutes les lettres majuscules sont triées avant les lettres minuscules.

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Dans cet exemple, vous pourriez obtenir des résultats surprenants grâce à ceci :

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

Pour résoudre ce problème, vous pouvez utiliser la fonction de tri insensible aux majuscules, qui est key = str.lower. Ici, il utilise la chaîne de type de données pour les rendre minuscules.

Exemple:

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

De plus, vous pouvez utiliser la méthode reverse() pour inverser l'ordre d'une liste quel que soit son ordre alphabétique :

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

Joindre des listes

Rappelez-vous quand nous avons vu les opérateurs arithmétiques utilisés et leur effet sur les chaînes. L'une des méthodes pour joindre des listes consiste à utiliser le signe d'addition (+):

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

Ici, nous avons créé une nouvelle liste en combinant les deux listes d'adjectifs et de mots déjà existantes. Quels que soient les types de données trouvés dans l'une ou l'autre des listes, vous pouvez les combiner avec l'opérateur +.

Si vous souhaitez ajouter des éléments d'une liste à une autre, utilisez simplement la méthode extend() :

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

Dans cet exemple, la liste de mots est jointe à la liste d'adjectifs. Par conséquent, la liste de mots contiendra désormais les deux listes.

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

3.3.2. Tuples

Les tuples contiennent également une séquence d'éléments dans une variable comme des listes. Semblables aux listes, les tuples sont également ordonnés. Cependant, une différence

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

clé entre les listes et les tuples est que les tuples sont immuables (c'est-à-dire inchangeables). Les tuples sont entre parenthèses ().

Exemple:

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

Dans la vie réelle, les tuples sont couramment utilisés comme dictionnaire sans clés pour stocker les données car ils sont plus rapides à parcourir et leurs éléments ne peuvent pas être modifiés. Les tuples autorisent également les valeurs doubles.

Si vous voulez voir combien d'éléments il y a dans un tuple, vous pouvez utiliser la fonction len() :

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

Lors de la création d'un tuple qui ne contient qu'un seul élément, vous devez ajouter une virgule après l'élément. Sinon, Python ne le reconnaîtra pas comme tuple. Nous utiliserons la fonction de type pour déterminer la différence entre l'utilisation de la virgule.

Exemple:

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

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

Similaires aux listes, les tuples peuvent contenir des éléments de n'importe quel type de données, soit dans des tuples séparés, soit sous forme de combinaison comme dans l'exemple ci-dessous :

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

Une autre façon de créer un tuple consiste à utiliser le constructeur tuple() :

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

*** Notez que lorsque vous utilisez le constructeur de tuple, vous devez ajouter des parenthèses doubles.**

Accéder aux tuples

Les éléments des tuples utilisent l'indexation, ce qui signifie que chaque élément du tuple correspond à un nombre. L'indexation est basée sur l'ordre des éléments et elle part de 0 pour le premier élément et augmente de 1 à chaque fois.

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Pour accéder à un élément spécifique, vous devez utiliser des crochets et spécifier le numéro d'index :

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

***Rappelez-vous que le numéro d'index 1 dans le tuple correspond au deuxième élément, c'est-à-dire 34.**

Vous pouvez également utiliser l'indexation négative pour accéder au dernier élément d'un tuple [-1] ou à l'avant-dernier élément [-2] et ainsi de suite.

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

Cela affichera l'élément "mâle" dans le tuple.

Si vous le souhaitez, vous pouvez spécifier une plage d'index qui indique à partir de quel numéro d'index commencer et à quel numéro d'index terminer la plage.

Exemple:

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

Dans cet exemple, il commencera à partir du deuxième élément de la liste et se terminera au dernier, puisque nous n'avons que 4 éléments sur la liste. Rappelez-vous toujours que la numérotation commence à partir de 0.

Si vous décidez que vous ne voulez pas spécifier le numéro d'index de début, la plage commencera à partir du premier élément trouvé dans le tuple :

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

Vous pouvez également spécifier uniquement le numéro d'index de début sans numéro d'index de fin :

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

N'oubliez pas que lorsque vous introduisez l'indexation négative, vous pouvez également l'utiliser pour accéder aux éléments d'une liste à partir de la fin de la liste :

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

Cela nous donnera le quatrième élément de la fin jusqu'au dernier du tuple.

Vous pouvez également vérifier si un élément existe dans un tuple, similaire à ce que nous avons vu dans les listes. Disons que nous voulons vérifier si *beautiful* est inclus dans le tuple adj :

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

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

Ajouter, modifier et supprimer des éléments des tuples

Techniquement, vous ne pouvez pas ajouter ou supprimer des éléments des tuples car ils sont immuables (c'est-à-dire non modifiables). Cependant, il existe une solution de contournement. Il vous suffit de convertir le tuple en liste, de le modifier puis de le reconvertir en tuple.

Voyons un exemple pour mieux comprendre cela :

```
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)
```

Si vous souhaitez ajouter des éléments dans un tuple, vous pouvez utiliser deux méthodes. Le premier suit la même logique que l'exemple que nous avons vu pour modifier un élément présent dans un tuple.

Exemple:

```
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
```

La deuxième façon d'ajouter un nouvel élément consiste à ajouter un tuple à un autre tuple. Vous pouvez créer un nouveau tuple contenant l'élément que vous souhaitez ajouter au tuple existant et simplement l'ajouter :

```
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)
```

Ici, nous verrons comment nous pouvons supprimer des éléments des tuples. On applique la même logique des exemples précédents d'ajout ou de modification d'éléments dans les tuples.

Exemple:

```
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)
```

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Dans l'exemple suivant, nous verrons comment vous pouvez supprimer le tuple en utilisant le mot-clé *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
```

Lors de la création d'un tuple, nous attribuons des valeurs ou des éléments à contenir. Ce processus est également appelé « packing » un tuple :

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

En Python, vous avez également la possibilité d'extraire les valeurs dans des variables, ce qui s'appelle « 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)
```

*** Notez que les noms de variables doivent correspondre aux éléments contenus dans le tuple, sinon vous pouvez utiliser un astérisque (*) pour collecter le reste sous forme de liste.**

Disons que vous voulez extraire seulement deux variables du tuple personal_info1, vous pouvez utiliser l'astérisque :

```
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)
```

Si vous ajoutez l'astérisque à un autre nom de variable au lieu du dernier, alors les valeurs seront affectées à la variable avec l'astérisque jusqu'à ce que les valeurs restantes correspondent aux variables restantes :

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

Joindre des tuples

Si vous souhaitez joindre deux tuples, vous pouvez utiliser l'**opérateur d'addition (+)**:

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

opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Vous avez également la possibilité de multiplier le contenu d'un tuple plusieurs fois en utilisant **l'opérateur de multiplication (*)** :

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

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

3.3.3. Dictionnaires

Les dictionnaires font partie du groupe de mappage des types de données car ils peuvent stocker des données dans des paires clé:valeur. Ils contiennent une séquence d'éléments pairs qui sont ordonnés, modifiables et n'autorisent pas les valeurs doubles. Les dictionnaires sont marqués par des accolades {}.

*** Notez que les dictionnaires de Python 3.6 et des versions antérieures n'étaient pas ordonnés, mais à partir de Python 3.7. ils sont devenus ordonnés.**

Un exemple de dictionnaire peut être pour traduire l'anglais dans une autre langue. Disons que nous voulons traduire de l'anglais vers l'espagnol :

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

Une autre option consiste à créer un dictionnaire vide et à ajouter des éléments un par un :

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

Comme nous l'avons mentionné, les dictionnaires n'autorisent pas les valeurs doubles pour leurs clés, c'est-à-dire "hello", dans le dictionnaire eng2esp que nous avons créé ci-dessus.

Si vous voulez savoir combien d'éléments se trouvent dans un dictionnaire, vous pouvez utiliser la fonction len() comme nous l'avons vu pour les listes et les tuples :

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

Encore une fois, à l'instar des tuples et des listes, les dictionnaires peuvent contenir n'importe quel type de données :

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

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Comme vous pouvez le voir ici, nous avons des types de données *int*, *strings*, *bool* et *list*.

Pour déterminer le type de données de la variable “canapés”, utilisez *type()* :

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

Accéder aux éléments des dictionnaires

Si vous voulez rechercher ce qu'est "hello" en espagnol, vous pouvez utiliser ce qui suit :

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

Cela affichera: "hola"

Lorsque vous recherchez des éléments dans un dictionnaire, utilisez leur nom de clé entre crochets :

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

Ici, "hello" est la clé de la valeur "hola".

Une méthode alternative consiste à utiliser la méthode *get()* pour avoir le résultat équivalent:

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

Si vous voulez voir une liste de toutes les clés que contient un dictionnaire, vous pouvez utiliser la méthode *keys()* :

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

Toute modification apportée au dictionnaire eng2esp sera répercutée sur la liste all_keys que nous avons créée :

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

Also, you have the option of getting all the values contained in a dictionary with the *values()* method:

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

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l’Union européenne

La même logique s'applique pour les valeurs en termes de changements. Si nous ajoutons une nouvelle valeur, elle sera ajoutée à la liste de valeurs que nous avons créée.

Une autre option que vous avez est d'obtenir toutes les paires clé:valeur en utilisant la méthode *items()* :

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

Encore une fois, toute modification apportée au dictionnaire, que ce soit dans ses clés ou ses valeurs, sera reflétée dans la liste *all_pairs* que nous avons créée.

Si vous ne savez pas si une clé particulière existe dans un dictionnaire, vous pouvez vérifier :

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

Ajouter, modifier ou supprimer des éléments des dictionnaires

Pour ajouter des éléments dans un dictionnaire, vous pouvez utiliser deux méthodes. Le premier utilise une nouvelle clé d'index et sa valeur correspondante à ajouter :

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

La deuxième façon consiste à utiliser la méthode *update()*, où vous devez spécifier le dictionnaire, ou la paire itérable clé:valeur :

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

***Notez que les accolades après la parenthèse sont nécessaires pour indiquer les paires clé:valeur du dictionnaire.**

Si vous souhaitez modifier des éléments dans un dictionnaire, les mêmes options que celles décrites ci-dessus sont disponibles. La première option consiste à se référer au nom de la clé pour modifier une valeur spécifique. Par exemple, si un canapé a été vendu, vous pouvez modifier la quantité de canapés en magasin :

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

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

La deuxième option fait référence à la méthode ***update()*** :

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

Il existe 3 façons différentes de supprimer des éléments des dictionnaires. La première est la méthode ***pop()*** :

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

La deuxième méthode consiste à utiliser le mot-clé ***del*** :

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

La troisième méthode disponible supprime le dernier élément ajouté dans Python 3.7. Cependant, dans les versions antérieures, un élément aléatoire sera supprimé à la place.

Exemple:

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

Si vous souhaitez supprimer complètement un dictionnaire, vous pouvez utiliser le mot-clé ***del*** :

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

Une autre option que vous avez est de simplement vider le dictionnaire en utilisant la méthode ***clear()*** :

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

Copier des dictionnaires

Rappelez-vous quand nous avons parlé de copier des listes et que nous avons dit que vous ne pouvez pas copier un élément sur un autre en utilisant le signe égal (=) car les modifications d'une liste seront effectuées sur l'autre. Comme pour les listes, les dictionnaires ne doivent pas être copiés de cette façon. Il existe deux façons de faire une copie d'un dictionnaire.

La première consiste à utiliser la méthode ***copy()*** :

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

La deuxième façon est d'utiliser la fonction ***dict()*** :

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

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

3.4. Conditionnels et boucles

Dans cette section, nous expliquerons la logique et l'utilisation derrière les instructions et les boucles conditionnelles, qui sont considérées comme des connaissances essentielles et utiles dans les langages de programmation en général.

⇒ Les instructions conditionnelles utilisent les instructions **if**, **elif**, **else**.
⇒ **For** et **while** sont utilisés pour les boucles.

Avant d'expliquer en détail les conditions et les boucles, nous allons apprendre quelques opérateurs utiles couramment utilisés dans ces instructions : les expressions booléennes (==) et les opérateurs logiques (et, ou, non).

3.4.1. Expressions booléennes

Une expression booléenne est utilisée pour déterminer si une expression ou une instruction est vraie ou fausse. Il existe deux manières d'écrire des expressions booléennes, la première utilise l'opérateur == pour comparer deux valeurs et renvoyer une valeur booléenne :

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

Output (sortie):

Vrai

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

Output (sortie):

Faux

Dans les deux instructions, l'opérateur == évaluait si les deux entiers étaient identiques (c'est-à-dire avaient la même valeur). Comme vous pouvez le voir, la première instruction a donné Vrai et la deuxième a donné Faux.

Le == est l'un des nombreux opérateurs de comparaison utilisés en Python. Les autres opérateurs de comparaison sont présentés ci-dessous :

x > y	x est supérieur à y
x < y	x est inférieur à y
x >= y	x est supérieur ou égal à y
x <= y	x est inférieur ou égal à y
x != y	x n'est pas égal à y

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Vous avez probablement déjà rencontré ces opérateurs, cependant, certains de ces symboles ont des fonctions différentes en Python.

Par exemple, si vous voulez voir si une valeur est identique à une autre, vous n'utiliserez pas un seul signe égal (=) car cela affecterait la valeur à la variable qui se trouve sur le côté gauche.

Pour comparer des valeurs dans des expressions booléennes, vous devez utiliser le **double signe égal (==)**.

De plus, si vous voulez vérifier si une valeur est supérieure ou égale (>=) à une autre valeur, vous devez mettre le **signe supérieur (>)** ou **inférieur (<)** en premier, puis le **signe égal (=)**.

Ce sont des distinctions importantes en Python dont vous devez vous souvenir pour éviter les erreurs ou les mauvaises surprises.

Voyons quelques exemples de ces opérateurs de comparaison avec le mot clé ***if*** :

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

Dans cet exemple, nous comparons les variables x et y pour voir si y est supérieur à x. Puisque nous savons déjà que y est supérieur à x, nous choisissons de l'imprimer en sortie.

*** L'indentation qui se produit dans la deuxième ligne (c'est-à-dire l'espace blanc trouvé au début de la ligne) est utilisée pour indiquer à Python que cette instruction fait partie de ce bloc de code particulier. L'indentation est une partie importante de la syntaxe Python pour éviter de générer des erreurs.**

Un autre mot-clé utilisé dans les expressions conditionnelles est ***elif***, qui est utilisé comme alternative si la première condition n'est pas vraie. Voyons un exemple d'opérateurs de comparaison avec le mot-clé ***elif*** :

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

Dans cet exemple, nous avons la première condition qui vérifie si y est supérieur à x. Comme ce n'est pas le cas, le programme passe à la deuxième condition avec elif (une combinaison de *else* et *if* (sinon)) qui vérifie si x est égal à y, ce qui est vrai et affiche l'instruction correspondante.

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

L'autre mot-clé utilisé dans les conditions c'est *else*, qui est la dernière option disponible si les conditions précédentes ne sont pas vraies.

Voyons un exemple de *else* pour comprendre comment cela fonctionne :

```
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")
```

Ici, vous pouvez voir que la condition ***if*** n'est pas vraie, puis elle passe à la deuxième condition qui n'est pas vraie non plus et se termine par la dernière option disponible dans else, qui est vraie et affiche l'instruction correspondante.

De plus, si vous n'avez qu'une courte instruction à exécuter, vous pouvez l'écrire sur une seule ligne :

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

Vous pouvez également faire la même chose pour une combinaison d'instructions *if* et *else* :

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

Une autre option consiste à inclure plusieurs instructions else sur une seule ligne :

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

3.4.2. Opérateurs logiques

Python comprend 3 opérateurs logiques, comme nous l'avons mentionné au début de cette section, *and*, *or*, et *not*. Ces opérateurs ont des utilisations analogues dans les langages naturels qui combinent une ou plusieurs instructions conditionnelles.

Voyons comment ***and*** peut être utilisé avec des instructions conditionnelles :

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

Voyons un exemple de l'opérateur ***or*** qui combine des instructions conditionnelles :

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

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Notre exemple suivant est axé sur l'opérateur ***not*** :

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

3.4.3. Instructions *if* imbriquées

Les instructions qui contiennent une instruction ***if*** à l'intérieur d'une ***autre instruction if*** sont appelées instructions ***if*** imbriquées. Examinons l'exemple suivant pour comprendre cela :

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

Similaires aux fonctions, les instructions ***if*** ne doivent pas être vides. Cependant, si pour une raison particulière vous avez une instruction ***if*** sans aucun contenu, utilisez l'instruction ***pass*** pour éviter d'obtenir des erreurs de Python :

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

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

3.4.4. Boucles

Python a deux commandes principales pour créer des boucles : ***while*** et ***for***.

La boucle ***while*** est utilisée lorsque vous souhaitez continuer à exécuter tant que la condition est vraie. Par exemple, créons une boucle ***while*** qui affiche *i* tant qu'il est inférieur à 10 :

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

Dans cet exemple, nous disons à Python que **tant que i est inférieur à 10, continuer à afficher i et incrémenter de 1 à chaque fois que la boucle se répète.**

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

*** Gardez à l'esprit que si vous n'incrémentez pas i, la boucle while continuera à s'exécuter sans s'arrêter et vous vous retrouverez dans une boucle infinie.**

Vous pouvez également utiliser l'instruction ***break*** si nous voulons arrêter la boucle même si la condition est vraie :

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

Une autre instruction disponible dans les boucles est ***continue*** qui arrêtera l'itération en cours et passera à la suivante :

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

L'instruction ***else*** que nous avons vue dans les instructions conditionnelles peut également être utilisée dans les boucles :

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

Entraine toi:

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

Les boucles qui utilisent le mot-clé ***for*** itèrent sur une séquence d'éléments, qui peuvent être contenus dans des listes, des tuples, des dictionnaires, des ensembles ou des chaînes.

Voyons un exemple de ceci:

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

Dans ce type de boucle, vous n'avez pas besoin de spécifier la variable d'index au préalable. Une variable d'index est, dans ce cas, la version singulière des noms comme valeur à récupérer (c'est-à-dire chaque élément de la liste des noms).

La boucle ***for*** peut également itérer sur une chaîne sous forme de séquences de caractères :

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

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

L'instruction ***break*** que nous avons vue précédemment peut également être combinée avec une boucle ***for*** :

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

Dans cet exemple, lorsque la boucle rencontre le nom "chaise", elle arrête l'itération avant de parcourir tous les éléments de la liste.

Si vous aviez mis l'instruction ***break*** avant la partie affichage, que pensez-vous qu'il se passerait ?

Essayez-le par vous-même :

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

Une autre instruction que nous avons vue précédemment est l'instruction ***continue***, qui interrompt l'itération en cours pour passer à la suivante :

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

La fonction ***range*** (gamme) est très utile lorsque vous souhaitez spécifier le nombre de fois que vous souhaitez que l'itération se produise. Le point de départ par défaut de la fonction ***range*** est 0, qui augmente de 1 à chaque itération et se termine à un nombre spécifié :

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

*** Gardez à l'esprit que parce que ça commence à 0, elle s'arrêtera au nombre 9. Si nous voulons réellement qu'elle aille jusqu'à 10, nous utiliserons range(11).**

Vous avez la possibilité de spécifier la gamme de début et de fin ::

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

Ici, la plage commence à 2 et se termine à 9, puisque 10 n'est pas inclus dans la plage.

Une autre option que vous avez est d'ajuster de combien le nombre augmente à chaque itération :

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

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Dans cet exemple, nous avons programmé la gamme pour commencer de 2 à 40 et augmenter de 2 à chaque fois. Par conséquent, le nombre s'arrêtera à 38, car il ne peut pas augmenter de 2 à partir de 38.

Dans la boucle ***for***, vous pouvez également utiliser le mot-clé ***else*** pour spécifier ce qui se passe lorsque la boucle se termine :

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

Essayons d'utiliser ***if*** et ***else*** dans une boucle ***for*** :

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

Pensez-vous que l'instruction ***else*** sera exécutée? Pourquoi? Essayez-le pour le savoir !

Rappelez-vous quand nous avons parlé d'instructions ***if*** **imbriquées**. Vous pouvez également avoir des boucles imbriquées :

```
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)
```

Comme vous pouvez le voir, la boucle interne sera exécutée une fois à chaque itération de la boucle externe.
Semblables aux fonctions et aux instructions ***if***, les instructions ***for*** ne doivent pas être vides. Cependant, si pour une raison particulière vous avez une boucle ***for*** sans aucun contenu, utilisez l'instruction ***pass*** pour éviter d'obtenir des erreurs de Python :

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

Entraine toi:
https://www.w3schools.com/python/exercise.asp?filename=exercise_for_loops1

3.5. Comprendre le flux d'exécution à travers les fonctions

Jusqu'à présent, nous avons vu un certain nombre de fonctions intégrées utilisées dans Python telles que ***print()***, ***type()***, ***dict()***, ***len()***. Chaque fonction a un objectif spécifique et nécessite que nous spécifions quelle variable nous voulons qu'elle exécute.

Exemple:

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

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

Dans la fonction d'affichage, nous devons spécifier entre parenthèses la variable x, qui est celle que nous voulons afficher. Quelle que soit la variable ou les variables contenues entre parenthèses, elles sont appelées ***arguments*** ou ***paramètres***.

Ces deux termes sont généralement utilisés de manière interchangeable, mais les **paramètres** sont les variables répertoriées entre parenthèses d'une fonction, par exemple, *print(x)*, tandis que les **arguments** sont considérés comme les valeurs envoyées à la fonction lorsque nous l'appelons.

Même si les fonctions intégrées de Python sont très utiles, elles ne peuvent pas toujours résoudre tous les problèmes que nous voulons qu'elles résolvent. Par conséquent, nous avons la possibilité de créer de nouvelles fonctions pour résoudre des problèmes spécifiques, ce qui est considéré comme l'un des aspects les plus bénéfiques d'un langage à usage général.

Une fonction contient une séquence d'instructions qui produisent une opération souhaitée. La syntaxe utilisée pour une fonction est la suivante :

```
>> def name(arguments):
        statements
```

Def signifie définition, il s'agit essentiellement de définir le nom de la fonction avec son ou ses arguments pour produire un résultat souhaité lorsqu'elle est utilisée. Les fonctions intégrées ont déjà été définies et, par conséquent, nous ne pouvons pas voir les instructions qu'elles contiennent, uniquement le résultat produit ou la sortie.

Lors de la création de votre propre fonction, vous êtes libre d'utiliser les noms de votre choix, à l'exception des mots-clés Python que nous avons mentionnés dans les sections précédentes (Section 3.2.). Les arguments utilisés entre parenthèses de la fonction fournissent les informations nécessaires au fonctionnement de la fonction.

Voyons un exemple pour comprendre le processus suivi par une fonction :

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

Ici, nous avons créé une fonction sans aucun argument requis qui affiichera le texte spécifié dans la deuxième ligne du code.

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Vous vous souvenez quand nous avons parlé d'indentation dans la section précédente ? La même logique s'applique ici où l'indentation indique que le code à exécuter est **local** à la fonction.
Par conséquent, écrire l'instruction print("Hello, World!") en dehors de la fonction comme nous l'avons fait dans les exemples précédents serait considéré comme une instruction **globale**. Cependant, maintenant que c'est écrit dans une fonction, elle est considérée comme **locale**.

Pour appeler la fonction, on utilise simplement son nom suivi de la parenthèse :

```
my_function()
```

La sortie de cette fonction sera :

Hello, World!

Voyons un autre exemple avec un argument spécifié à l'intérieur de la parenthèse de la fonction :

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

Lorsque la fonction est appelée, le nom du pays est passé dans la fonction pour être affiché comme nous l'avons demandé. L'argument *country* peut être spécifié lorsque nous appliquons la fonction :

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

Sortie:
I am from France (Je viens de France)
I am from Greece (Je viens de Grèce)
I am from Germany (Je viens d'Allemagne)

Vous pouvez également spécifier le type de données de l'argument à l'aide de la syntaxe suivante : argument:datatype

Exemple:

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

Lorsqu'on applique la fonction, Python s'attend à ce que le pays soit sous la forme d'une chaîne. Si ce n'est pas le cas, cela générera une erreur.

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

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

De plus, vous pouvez avoir plus d'arguments. Disons que nous voulions préciser la ville et le pays :

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

Notez que nous avons inclus des guillemets avec une virgule pour séparer la ville et le pays l'un de l'autre afin de ne pas avoir les deux mots affichés sans espace. Puisque nous avons spécifié 2 arguments, nous devons appeler exactement le nombre d'arguments spécifiés entre parenthèses, ni plus ni moins, pour que la fonction fonctionne. Sinon, cela générera une erreur.

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

Sortie:
I am from Paris, France (Je viens de Paris, France)

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

Une autre option que vous avez si vous ne connaissez pas le nombre d'arguments à inclure dans une fonction, ajoutez simplement un astérisque (*) avant le nom de l'argument dans la définition de la fonction.

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

De cette façon, la fonction recevra un tuple d'arguments et les éléments seront accessibles en conséquence.

*** Ces types d'arguments à nombre inconnu sont appelés arguments arbitraires et sont souvent appelés *args dans la documentation Python.**

Ici, nous avons spécifié 3 pays à imprimer dans des phrases séparées et dans une seule phrase :

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

Sortie:
I have visited Ireland (J'ai visité l'Irlande)
I have visited France (J'ai visité la France)
I have visited Belgium (J'ai visité la Belgique)
I have visited Ireland, France, Belgium (J'ai visité l'Irlande, la France, la Belgique)

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Une autre option à votre disposition consiste à envoyer des arguments sous forme de paires clé = valeur. De cette façon, l'ordre n'a pas d'importance.

Exemple:

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

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

Une autre option disponible est l'utilisation de deux astérisques (**) avant le nom de l'argument dans la définition de la fonction si vous ne savez pas combien d'arguments de mots-clés devront être passés. De cette manière, la fonction recevra les arguments sous forme de dictionnaire et y accédera en conséquence :

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

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

Comme vous pouvez le voir, vous pouvez ajouter de nouveaux arguments lorsque vous appliquez la fonction. Gardez à l'esprit que ces types d'arguments sont appelés **arguments de mots clés arbitraires** et sont souvent appelés ****kwargs** dans la documentation Python.

De plus, une autre option que vous avez est de définir une valeur de paramètre par défaut. Cela signifie que si nous appliquons la fonction sans argument, elle utilisera la valeur par défaut que nous avons définie :

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

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

Un argument peut autoriser n'importe quel type de données comme argument (par exemple, chaîne, liste, dictionnaire, etc.).
Les fonctions peuvent également renvoyer des valeurs à l'aide de l'instruction return :

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

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

Puisque nous n'avons pas spécifié la fonction d'affichage dans notre fonction créée, nous devons l'utiliser lorsque nous appliquons la fonction afin d'obtenir le résultat produit.

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Généralement, les fonctions ne doivent pas être vides. Cependant, si pour une raison particulière vous avez une définition de fonction sans contenu, utilisez l'instruction *pass* pour éviter d'obtenir des erreurs de Python :

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

Entraine toi:
https://www.w3schools.com/python/exercise.asp?filename=exercise_functions1

3.6. Assembler les pièces - Comment construire un programme

Jusqu'à présent, nous avons appris de nombreuses instructions, mots-clés, méthodes et fonctions différents utilisés en Python. Maintenant, vous vous demandez peut-être comment pouvons-nous réellement construire un programme simple ? Nous allons passer par le processus de construction d'un petit programme étape par étape, afin que vous puissiez comprendre comment utiliser ce que nous avons appris.

Nous voulons construire un programme qui :

1. lit un fichier texte,
2. crée un dictionnaire qui contient tous les mots du fichier texte et leur fréquence,
3. permet aux utilisateurs d'écrire un mot et d'afficher sa fréquence,
4. fournit un message informatif si un mot n'existe pas.

Cela peut sembler écrasant au premier abord, mais ne vous inquiétez pas ! Nous allons le parcourir étape par étape.

Ici, nous allons apprendre à créer des fonctions récursives, ce qui signifie que les fonctions s'appliqueront mutuellement pour exécuter notre programme.

Tout d'abord, nous devons appliquer un module, qui s'appelle ***string***. Le module de chaîne intégré contient plusieurs fonctions qui vous permettent de manipuler des chaînes en Python, que nous utiliserons pour supprimer tous les ensembles de ponctuation ultérieurement.

Il existe de nombreux modules disponibles en Python à des fins diverses que vous pouvez appliquer en utilisant simplement le mot-clé ***import*** :

```
import string
```

La première fonction lira le fichier texte et créera un dictionnaire :

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

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Nous essayons toujours de donner un nom de fonction que nous pouvons comprendre, et l'argument du nom de fichier sera spécifié lorsque nous l'appliquons. La ligne suivante crée un dictionnaire vide. La ***variable locale fin*** appelle la fonction ***open*** pour aller ouvrir le fichier et le lire. La boucle ***for*** dans la ligne suivante appelle la fonction suivante qui traitera chaque ligne, créera notre dictionnaire et le renverra.

Cela peut sembler un peu technique, mais c'est assez simple. Nous avons essentiellement créé une fonction qui créera un dictionnaire vide, lira le nom du fichier et traitera chaque ligne du fichier pour créer un dictionnaire.

Maintenant, pourquoi voulons-nous traiter chaque ligne ?

Parce que Python fera la différence entre les majuscules et les minuscules et tiendra compte des points de ponctuation à côté des mots, même si le mot est le même, comme "Amour" et "amour" ou "amour" et "amour".

Puisque nous voulons compter la fréquence de chaque mot dans le fichier, nous devons nous assurer que tous les mots sont en minuscules et que tous les points de ponctuation (c'est-à-dire les virgules, les points, les points d'exclamation, etc.) sont supprimés pour que le programme compte correctement.

Écrivons une fonction qui traite chaque ligne de notre fichier :

```
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
```

Encore une fois, nous utilisons une boucle ***for*** pour parcourir chaque ligne du texte.

- ⇒ Tout d'abord, nous remplaçons les tirets par des espaces car ils ne peuvent pas être supprimés avec la fonction ***string.punctuation***.
- ⇒ Une fois que nous avons remplacé les tirets, nous commençons par diviser la ligne en mots séparés et prenons chaque mot pour supprimer les ponctuations, les espaces, puis le mettons en minuscules.
- ⇒ Après tout le traitement, le mot est ajouté au dictionnaire et **si le mot existe, vous ajoutez 1, sinon 0.**

Jusqu'à présent, nous avons créé deux fonctions qui lisent, éditent le texte et créent un dictionnaire des fréquences des mots.

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

À ce stade, vous affectez au dictionnaire de variables la fonction qui traite le texte, où vous précisez le nom du fichier texte (c'est-à-dire "*the_veldt.txt*"). Toutes les fonctions sont connectées en fonction du dictionnaire de variables qui contiendra tous les mots trouvés dans un texte donné avec leurs fréquences.

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

Maintenant, nous en avons terminé avec 2 des 4 actions que nous voulons que le programme fasse, il ne nous en reste plus que 2.

Maintenant, nous voulons que les utilisateurs puissent écrire un mot et si le mot existe dans le texte afficher sa fréquence, sinon informer l'utilisateur que le mot n'existe pas.

Écrivons une fonction qui comparera l'entrée de notre utilisateur avec le dictionnaire que nous avons créé :

```
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 ")
```

Cette fonction est utilisée pour accéder à la clé et à la valeur de chaque élément du dictionnaire via une boucle ***for***. Si la clé (mot) est la même que le mot écrit par l'utilisateur, alors il renvoie la valeur (c'est-à-dire la fréquence d'un mot donné). Sinon, il renvoie une déclaration indiquant que le mot n'a pas été trouvé et invite l'utilisateur à en rechercher un autre.

```
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
```

Maintenant, pour la dernière partie du code, nous voulons que le programme se répète jusqu'à ce que l'utilisateur décide de quitter. Pour cette partie, nous utiliserons une boucle ***while*** qui n'est pas dans une fonction.
Nous utilisons simplement la boucle ***while True*** pour continuer le programme. Soyez prudent lorsque vous utilisez une boucle ***while True*** car vous pourriez vous retrouver dans une boucle infinie. Alors que ***True*** signifie “tant que cela est vrai, continuez à exécuter le programme”.

Nous attribuons une variable à l'entrée utilisateur afin de pouvoir la comparer avec les clés du dictionnaire (mots) et quelles que soient les entrées utilisateur, elles seront converties en minuscules lorsque le programme les lira. La raison en est qu'un utilisateur peut écrire un mot en majuscules et que le programme ne produira pas de résultat, car Python ne

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

reconnaît pas qu'il s'agit du même mot, car nous l'avons mentionné. Python est sensible aux majuscules/ minuscules.

La ligne suivante indique que si l'entrée n'est pas égale à '*q*', afficher les résultats de la fonction qui récupère la fréquence et passer à l'itération suivante. Sinon, si l'entrée de l'utilisateur est *q*, il arrête le programme (c'est-à-dire rompt la boucle).

Selon la personne qui écrit le code, ce petit programme aurait pu être écrit de différentes manières. Nous avons essayé d'incorporer autant de choses que nous avons apprises ici, ainsi que d'introduire des fonctions récursives et comment diviser votre programme en petits morceaux de code gérables.

*** Gardez à l'esprit que lorsque vous écrivez du code, vous devez essayer d'exécuter chaque petite partie de votre code pour détecter les erreurs potentielles avant de passer à la partie suivante.**

Une fois que vous avez terminé votre programme, vous devrez peut-être revenir en arrière et modifier pour supprimer certaines parties expérimentales/initiales du code ou consolider plusieurs instructions pour rendre le programme plus compact et plus facile à lire.

3.7. Comment adapter un programme à vos besoins

Nous avons vu comment créer un petit programme en Python, cependant, vous pouvez parfois rencontrer des programmes déjà écrits et vous devrez peut-être apporter des modifications au code pour améliorer ou modifier le résultat final.

Selon qui a écrit le code et sa durée, cela peut être une procédure difficile et frustrante.

Vous pouvez suivre la même logique que lors de la construction d'un programme, où vous prenez de petites parties du code pour réaliser ce que fait chaque fonction, si ce n'est pas clair.

En outre, il est important d'écrire des commentaires lorsque vous travaillez avec du code, car les commentaires peuvent être extrêmement utiles pour les autres personnes lisant votre code et pour vous également.

Alors, que faites-vous lorsqu'il n'y a pas de commentaires sur un programme ?

Tout d'abord, exécutez le programme pour voir quelle est sa sortie et notez les bibliothèques ou les modules qui sont importés. Ensuite, vous devez rechercher le point de départ du code et essayer de comprendre le flux d'exécution. Un autre conseil utile serait de regarder les variables que le programme contient et leur utilisation tout au long du programme. Après avoir bien compris le fonctionnement du programme, vous pouvez commencer à éditer.

*** Soyez prudent lorsque vous essayez de modifier la fonction de quelqu'un d'autre, car vous pourriez finir par la casser**

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Une bonne pratique lors de l'ajustement d'un programme est de ne pas utiliser un nombre spécifique de variables, mais plutôt d'utiliser ***args** et ****kwargs** pour rendre votre code plus flexible et adaptable lorsqu'un autre utilisateur pourrait avoir besoin de l'ajuster. Nous avons vu l'utilisation de ces deux variables plus tôt dans le module.

Pour rappel, ***args** permet de passer un nombre variable d'arguments positionnels et ****kwargs** permet de passer un nombre variable de mots clés. L'importance des deux est l'utilisation des astérisques (* ou **), les noms peuvent être ce que vous voulez qu'ils soient.

Pour plus de conseils et d'exemples, consultez les sites Web suivants :

- 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. Erreurs syntaxiques, d'exécution et sémantiques - Gestion des erreurs en Python

Il existe trois types d'erreurs différents très importants qui peuvent survenir lors de l'écriture d'un programme et il est utile de connaître leurs différences et de les repérer rapidement :

1. Des **erreurs de syntaxe** se produisent lorsque Python traduit le code source sous forme binaire et indique qu'une partie de la syntaxe est erronée, par exemple, l'indentation, l'absence de deux-points à la fin d'une instruction *def* ou une parenthèse manquante. Semblable aux langages naturels, lorsque nous utilisons une syntaxe incorrecte en Python, nous obtenons une erreur disant : *syntaxe invalide*.

   Les erreurs de syntaxe sont plus faciles à repérer, voici quelques éléments à surveiller:

   - Utilisation d'un mot-clé Python pour un nom de variable
   - Deux-points (:) manquants à la fin des instructions ***for***, ***while***, ***if*** et ***def***
   - Indentation dans les boucles et les conditions
   - Guillemets correspondants pour les chaînes, vérifiez s'ils sont doubles ou simples
   - Vérifiez que vous n'avez pas oublié de guillemets lors de l'écriture des chaînes
   - Utiliser un signe égal au lieu de deux dans une instruction conditionnelle
   - Un crochet non fermé dans toutes les lignes de votre code, c'est-à-dire, ***)***, ***]***, ***}***.

Si vous ne trouvez pas l'erreur de syntaxe, créez un nouveau fichier et ajoutez votre code ligne par ligne depuis le début.

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

2. Des **erreurs d'exécution** se produisent lorsque les conditions suivantes s'appliquent : 1) le programme ne fait rien, 2) le programme entre dans une boucle ou une récursivité infinie, 3) vous obtenez une erreur d'exception ou 4) vous avez ajouté trop d'instructions d'affichage.

   Voyons quelques façons de surmonter ou de repérer la racine de ces instances :

   - Lorsque votre programme ne fait rien, assurez-vous que vous lui avez demandé de s'exécuter dans la console interactive.
   - Si vous entrez dans une boucle infinie, arrêtez le programme et ajoutez des instructions d'affichage là où vous pensez que le problème pourrait être et essayez d'utiliser des hashtags (#) devant des morceaux de votre code pour voir ce qui se passe.
   - Les erreurs d'exception se répartissent en 5 catégories principales:
     i. **NameError** quand vous essayez d'utiliser une variable qui n'existe pas, c'est-à-dire des variables locales;
     ii. **TypeError** peut se produire pour plusieurs raisons, à savoir l'utilisation incorrecte d'une valeur, une incompatibilité entre les éléments au format chaîne et les éléments convertis ou un nombre incorrect d'arguments;
     iii. **KeyError** peut se produire lors de la tentative d'accès à un élément d'un dictionnaire à l'aide d'une clé qui n'existe pas;
     iv. **AttributeError** peut se produire lors d'une tentative d'accès à un attribut qui n'existe pas;
     v. **IndexError** quand il y a incompatibilité entre le numéro d'index d'une liste, d'une chaîne ou d'un tuple et sa longueur.

3. Les **erreurs sémantiques** sont généralement les erreurs les plus difficiles à comprendre car le programme est en cours d'exécution mais ne produit pas le résultat souhaité. Vous avez peut-être pensé que l'ordre dans lequel vous avez placé vos instructions a du sens, cependant, Python peut se comporter de manière inattendue.

   Lorsque vous rencontrez des erreurs sémantiques, il est utile de procéder comme suit :

   - Décomposez le code en parties plus petites pour comprendre comment le programme se comporte à chaque étape.
   - Passez en revue vos pensées ou vos notes sur la logique derrière chaque ligne de code.
   - Utilisez des instructions d'affichage pour voir ce que fait le programme.

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

- Si vous avez une longue expression, utilisez des variables temporaires pour vérifier les types de variables.
- Attribuez une variable à votre expression avant une instruction de retour.
- Si vous ne pouvez pas comprendre l'erreur, faites une courte pause ou demandez de l'aide.

3.9. Entraînement

L'un des aspects les plus importants du codage est la pratique. Plus vous pratiquerez, plus facile ça deviendra. Espérons qu'à la fin de ce module, vous ayez une solide compréhension de la logique derrière l'écriture de code, des différents types de données et de leur utilisation, ainsi que de la manière de créer de petits programmes à l'aide de fonctions.

Nous vous encourageons à pratiquer autant que possible afin d'utiliser le codage pour créer vos propres projets individuels et améliorer vos perspectives de carrière et votre réussite.

Prêt(e) à commencer à pratiquer ? Bon codage !

Ici, vous pouvez trouver une liste de différents sites Web que vous pouvez utiliser pour mettre en pratique vos compétences de codage :

⇒ **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/

Références:

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/

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/

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

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

PARTIE B: Le module de microcontrôleurs CodER (10 hours)

Description:

Cette partie fournit une introduction aux microcontrôleurs, leur utilisation et leur application basée sur des exemples concrets. La première sous-section explique les composants d'un microcontrôleur et ses différents types afin de familiariser les apprenants avec ce concept. Ensuite, il passe à l'utilisation du logiciel Arduino et à la façon dont il se connecte aux microcontrôleurs Arduino dans une approche étape par étape.

Charge de travail:

10 heures

Résultats d'apprentissage:

À la fin de ce cours, les apprenants seront capables de:

- ⇒ Reconnaître ce qu'est un microcontrôleur et être capable d'identifier les différents types de microcontrôleurs.
- ⇒ Différencier les entrées/sorties analogiques et numériques (I/0).
- ⇒ Utiliser la syntaxe de base de l'IDE Arduino.
- ⇒ Exécuter différents exemples d'IDE Arduino et de microcontrôleurs.

Matériel et ressources nécessaires:

- ⇒ Ordinateur fixe ou portable
- ⇒ Accès Internet
- ⇒ Arduino Uno
- ⇒ EDI Arduino

Aspects pratiques:

Cette partie du module est conçue pour couvrir 10 heures d'apprentissage. Chaque section du module a un temps désigné, mais l'apprenant ou l'éducateur/formateur est libre de décider du montant à dépenser pour chaque sous-thème en fonction de ses connaissances antérieures et de son engagement dans des sujets similaires. Le contenu est basé sur un

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

développement progressif et est utilisé pour fournir des connaissances et des compétences de base aux débutants.

Sous-sections:

1. Introduction aux microcontrôleurs
2. Les bases de la programmation avec Arduino
3. Applications d'Arduino

1. Introduction aux Microcontrôleurs

⇒ **Nombre de participants :** 1 à 10 par animateur

⇒ **Durée :** 1h30

⇒ **Méthodes d'enseignement:** Présentation, Instruction guidée, Apprentissage expérientiel

⇒ **Matériel requis :** présentation, cartes Arduino et connexion Internet stable

1.1. Qu'est-ce qu'un microcontrôleur

Avant d'expliquer ce qu'est un microcontrôleur, examinons les questions suivantes :

- Avez-vous déjà été curieux de savoir comment faire fonctionner les gadgets ? Quelle est la logique derrière eux ?
- Avez-vous déjà voulu savoir comment faire fonctionner les systèmes qui contrôlent les ascenseurs ou les jouets électroniques ?
- Ou même créer votre propre robot ou signaux électroniques pour un chemin de fer miniature ?
- Vous êtes-vous déjà demandé comment les données météorologiques sont optimisées et analysées ?

Êtes vous curieux? Eh bien, commençons alors!

Les microcontrôleurs peuvent faciliter une meilleure compréhension de ces processus électroniques grâce à des activités pratiques.

Un microcontrôleur est un circuit intégré compact chargé d'exécuter une fonction spécifique dans un appareil. Il interprète les données qu'il reçoit de ses périphériques d'entrée et de sortie (E/S) via son processeur central.

Les microcontrôleurs peuvent se trouver dans une variété de systèmes et d'appareils. Certaines applications des microcontrôleurs peuvent être trouvées dans les caméras, les

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

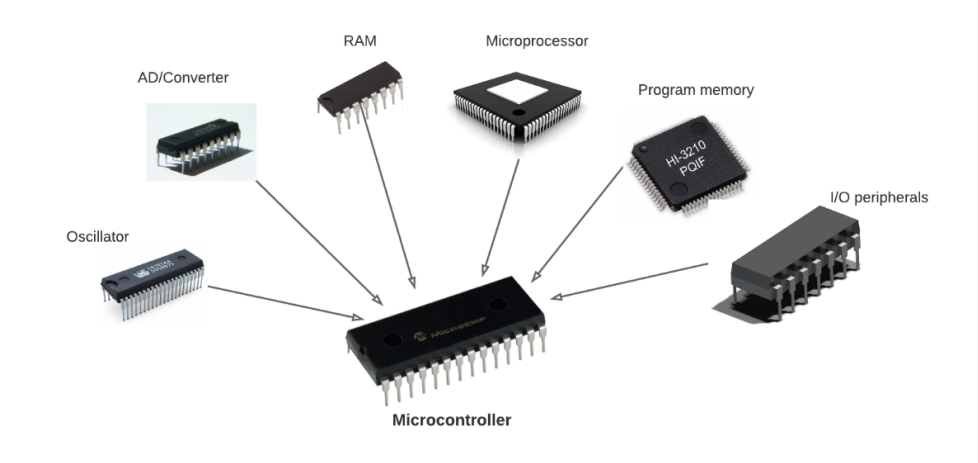
Cofinancé par l'Union européenne

commandes de moteur, les serrures de porte, les avertisseurs d'incendie ou de fumée, ou les capteurs de température, de lumière et de couleur.

Prenons l'exemple d'une voiture avec de nombreux microcontrôleurs pour contrôler des systèmes individuels tels que des capteurs de lumière, des systèmes de freinage antiblocage, des vitres électriques ou des commandes de freinage. Un véhicule peut avoir jusqu'à 50 microcontrôleurs responsables d'opérations spécifiques qui communiquent entre elles ou avec d'autres systèmes plus complexes pour effectuer des actions appropriées.

Un microcontrôleur est essentiellement une petite puce composée des éléments suivants :

1. Le processeur (CPU) : Un processeur ou unité centrale de traitement (CPU) est utilisé pour traiter et exécuter diverses instructions qui dirigent la fonction d'un microcontrôleur. Il lit et exécute des opérations arithmétiques, logiques et d'E/S de base. Il est également responsable du transfert de données pour communiquer des commandes à d'autres composants au sein d'un système plus vaste.
2. Mémoire : La fonction principale de la mémoire d'un microcontrôleur est de stocker des données afin de les envoyer au processeur et de répondre à sa fonction de programmation prédéterminée. Un microcontrôleur possède deux principaux types de mémoire :
    a. La mémoire programme est celle qui stocke des informations à long terme sur les opérations que le microcontrôleur a été programmé pour exécuter. Ce type de mémoire n'a pas besoin d'une source d'alimentation pour fonctionner et stocke les informations pendant une longue période.
    b. La mémoire de données ou Random Access Memory (RAM) est utilisée pour stocker des données temporaires pendant l'exécution du programme. Ce type de mémoire conserve les données tant que l'appareil est connecté à une source d'alimentation.
3. Périphériques d'entrée/sortie (E/S) : les périphériques d'E/S sont responsables de la communication des microcontrôleurs avec d'autres composants. Les ports d'entrée reçoivent des informations et les envoient pour un traitement ultérieur au processeur sous forme de données binaires. Sur la base des commandes du processeur, les ports de sortie exécutent les tâches nécessaires.



Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Image 17 - Éléments d'un microcontrôleur.

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

Il existe de nombreux exemples de microcontrôleurs disponibles sur le marché. Arduino, Scratch, Microbit et Raspberry PI sont parmi les plus populaires.

Voici leurs sites Web officiels pour que vous puissiez les consulter à votre rythme :

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

Dans ce module, nous nous concentrerons sur les microcontrôleurs Arduino.

1.2. Qu'est-ce que Arduino et ses différents types

Qu'est-ce que Arduino

Arduino est une plate-forme open source utilisée pour la construction de projets électroniques. Arduino se compose à la fois d'une carte de circuit physique programmable (souvent appelée microcontrôleur) et d'un logiciel ou IDE (environnement de développement intégré) qui s'exécute sur un ordinateur, utilisé pour écrire et télécharger du code informatique sur la carte de circuit physique, qui est une formidable plateforme de prototypage de projets et de créations (Green Steam Incubator, 2019).

Les cartes Arduino détiennent la clé pour comprendre les processus électroniques grâce à des activités pratiques. Ce système a été créé par Massimo Banzi et David Cuartielles en 2005. Il offre une alternative aux microcontrôleurs autrement coûteux et vous permet de construire des projets interactifs tels que des systèmes de suivi GPS, des capteurs de lumière, des robots télécommandés et des jeux électroniques.

The main components of Arduino are:

1. **Logiciel** : L'IDE Arduino est responsable de l'écriture des programmes utilisés pour communiquer avec votre hardware. Les fonctions de la carte sont contrôlées en fonction des instructions envoyées au microcontrôleur via Arduino IDE.
2. **Hardware** : Les cartes Arduino sont de différents types et peuvent lire les signaux d'entrée analogiques ou numériques de différents capteurs pour produire une sortie. Certains exemples de sorties incluent l'activation d'un moteur, l'activation/la désactivation d'une LED ou le verrouillage/déverrouillage d'une porte.

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

3. **Langage de programmation** : Le langage de programmation utilisé dans Arduino est une version simplifiée de C++.

Types d'Arduino

Il existe plusieurs types de cartes Arduino disponibles en fonction du microcontrôleur utilisé. Les différences sont principalement présentées dans les fonctionnalités suivantes :

- nombre d'entrées et de sorties (c.-à-d. capteurs, LEDs, boutons sur une carte)
- la vitesse
- tension de fonctionnement
- facteur de forme, etc.

Certaines cartes sont uniquement conçues pour être embarquées et ne proposent pas d'interface de programmation. D'autres peuvent être directement alimentées par une batterie de 3,7 V, tandis que d'autres ont besoin d'au moins 5 V pour fonctionner. Quelles que soient leurs différences dans ces fonctionnalités, elles peuvent toujours être programmées via l'IDE Arduino.

Quelques exemples des différents types de cartes Arduino peuvent être vus dans l'image ci-dessous.

Image 18 – Différents types de cartes Arduino.
Source: https://techatronic.com/types-of-arduino-boards-arduino-uno-mega-mini-specification/

Parmi les cartes Arduino les plus populaires se trouve l'Arduino Uno. Même s'il ne s'agissait pas de la première carte à sortir sur le marché, elle reste l'une des cartes les plus activement et largement documentées.

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

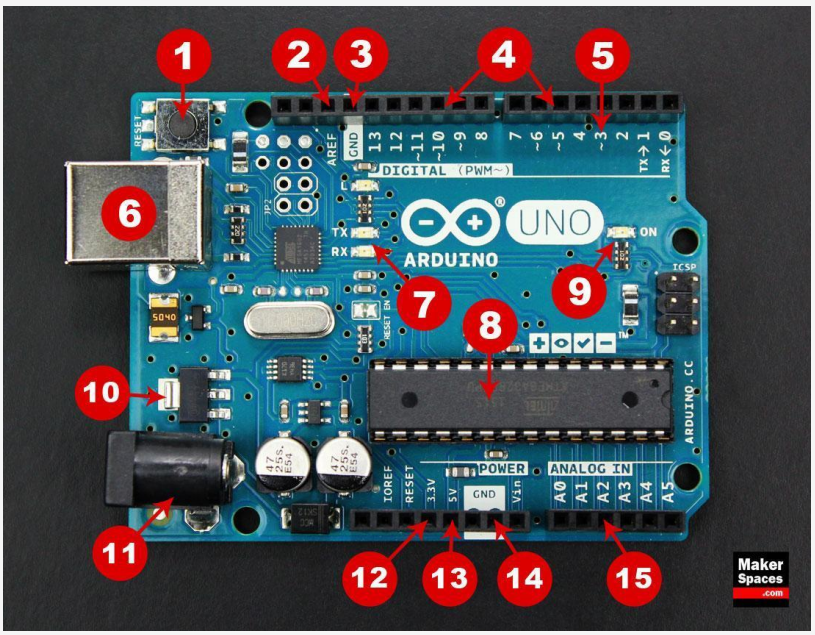


Image 19 – Arduino UNO.
Source: https://www.makerspaces.com/arduino-uno-tutorial-beginners/

1. Bouton de réinitialisation - Redémarre tout code qui a été chargé sur la carte Arduino
2. AREF - Signifie "Analog Reference" et définit une tension de référence externe
3. Broche de terre (GND) - Ferme le circuit électrique et fournit une référence commune tout au long
4. Entrée/sortie numérique - Les broches 0-13 peuvent être utilisées pour l'entrée ou la sortie numérique
5. PWM - Les broches marquées du symbole (~) peuvent simuler une sortie analogique
6. Connexion USB - Alimente votre Arduino et télécharge des croquis
7. TX/RX - Transmet et reçoit l'indication de données des LED
8. Microcontrôleur ATmega - Stocke les programmes (cerveau de l'Arduino)
9. Indicateur LED d'alimentation - S'allume lorsque la carte est branchée à une source d'alimentation
10. Régulateur de tension - Contrôle la quantité de tension entrant dans la carte Arduino
11. DC Power Barrel Jack - Alimente votre Arduino avec une source d'alimentation
12. Broche 3,3 V - Fournit 3,3 volts d'alimentation à vos projets
13. Broche 5V - Fournit 5 volts d'alimentation à vos projets
14. Broches de terre - Ferme le circuit électrique et fournit une référence commune tout au long

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

15. Broches analogiques - Lit le signal d'un capteur analogique et le convertit en numérique

Les cartes Arduino Uno doivent être connectées à une source d'alimentation pour fonctionner. Il existe différentes manières de connecter la carte à une source d'alimentation, par exemple directement via un ordinateur via USB ou dans le cas de projets mobiles via une batterie 9V. La dernière méthode nécessite l'utilisation d'une alimentation 9V AC pour fonctionner.

Image 20 – Sources d'alimentation de Arduino.
Source: https://www.makerspaces.com/arduino-uno-tutorial-beginners/

Un autre composant important de la carte Arduino est la *breadboard*, également appelée *breadboard* sans soudure. Elle est utilisée dans la phase de prototypage pour évaluer la fonctionnalité du circuit et permet la création et l'expérimentation temporaires de différentes conceptions de circuits. Les points d'attache (trous) du boîtier en plastique contiennent des clips métalliques reliés les uns aux autres par des bandes de matériaux conducteurs. Cependant, la *breadboard* n'est pas alimentée par elle-même et doit être connectée à la carte Arduino via des fils de connexion. En outre, ces fils sont utilisés pour former le circuit en connectant des résistances, des commutateurs et d'autres composants.

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

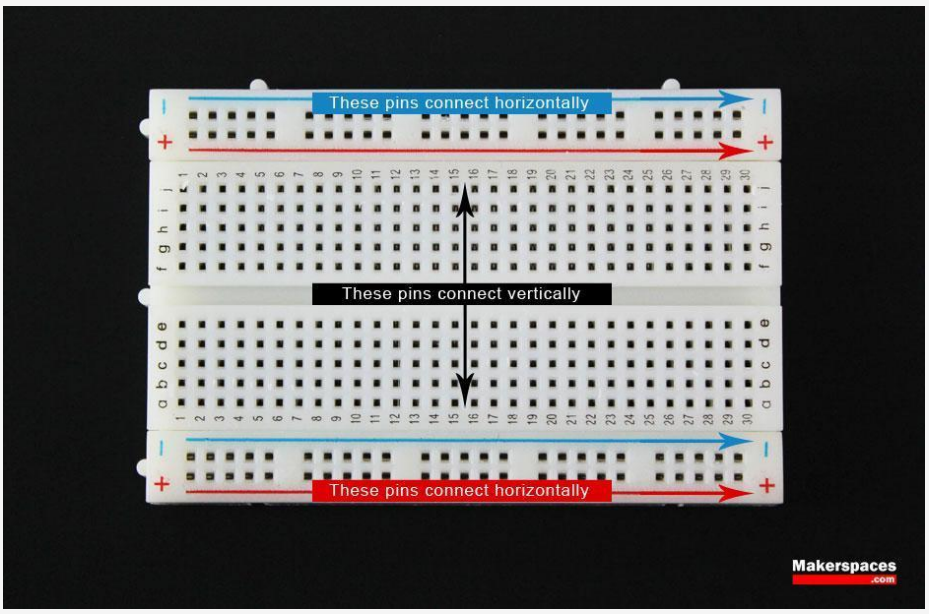


Image 21 – *Breadboard*.
Source: https://www.makerspaces.com/arduino-uno-tutorial-beginners/

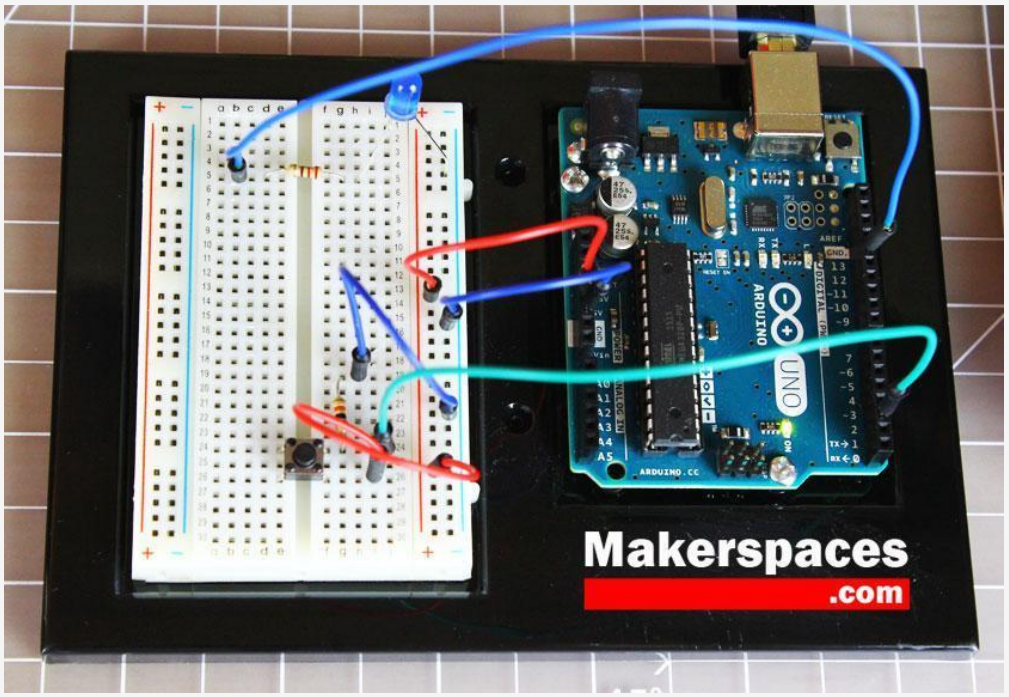


Image 22 – Circuit Arduino terminé.
Source: https://www.makerspaces.com/arduino-uno-tutorial-beginners/

1.3. Concepts : entrée, sortie, analogique, numérique

Entrées et sorties (E/S) dans un contexte de programmation

Les entrées et les sorties représentent les interactions entre un robot/une machine et le monde réel. En termes plus simples, les entrées sont les données que le robot/la machine reçoit et les sorties sont les résultats que nous pouvons voir.

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Les entrées sont généralement transmises par des capteurs tels que des interrupteurs, des potentiomètres ou des caméras, tandis que les sorties se réfèrent à des actions immédiates déclenchées par des moteurs, telles que l'allumage d'une LED ou l'extinction de l'alarme.

Prenons l'exemple suivant : Les touches d'un clavier représentent les entrées. Lors de la frappe, l'ordinateur reçoit des informations qui sont commandées. Cependant, ce processus ne nous est pas visible. L'ordinateur ou la machine prend l'entrée (c'est-à-dire tout ce qui est tapé) et l'affiche à l'écran.

Dans le contexte Arduino : un programme peut prendre un bouton en entrée, qui allumera une lumière LED s'il est activé. Les informations traitées par le bouton ne vous sont pas visibles, mais la lumière LED qui s'allume ou s'éteint est la preuve de la sortie.

Il existe deux types d'entrées/sorties : E/S analogiques et numériques.

Quelle est la différence entre les E/S analogiques et numériques

Il existe deux manières de différencier un signal analogique d'un signal numérique :

1. Par type de capteur
2. Par la méthode de traitement (c'est-à-dire le temps et la résolution)

1. Capteurs analogiques vs. capteurs numériques : une LED peut être allumée/éteinte ou avoir une intensité variable, comme allumée avec une faible intensité ou allumée avec une intensité élevée.

    ⇒ Si le capteur est un interrupteur placé sur la LED, il contrôlera si la LED est allumée ou éteinte et ne détectera rien d'autre. Il s'agit d'une entrée numérique.

    ⇒ Si le capteur est un LDR (résistance dépendante de la lumière), il détectera la lumière et la convertira en une valeur analogique comprise entre 0 et 255. Nous pouvons détecter si une LED est à 0 % (éteinte) ou à 100 % (pleine luminosité) et également lire de nombreuses autres valeurs intermédiaires (par exemple, 20, 23 %, 86 %). Il s'agit d'une entrée analogique.

2. Traitement analogique vs. traitement numérique : pour déterminer s'il s'agit d'un traitement analogique ou numérique, vérifiez simplement leur traitement en fonction de:

    ⇒ Temps : Le traitement analogique traite en continu les informations et chaque fois que l'entrée est modifiée, la sortie est instantanément modifiée en fonction de l'entrée. La mise à jour est immédiate. Pour le numérique, le traitement aura un court délai jusqu'à ce que le système enregistre le

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

changement. Ce retard est établi par la "fréquence d'échantillonnage" et il est contrôlé par une horloge.

⇒ Résolution : Le traitement analogique a des résolutions infinies car il ne s'arrête jamais et a de nombreuses valeurs différentes. En revanche, la résolution numérique fonctionne avec des nombres binaires, ce qui signifie qu'elle ne traite que deux valeurs et signaux.

Exemples avec lumière LED :

- Allumer/éteindre avec un traitement analogique : lorsque le gradateur est activé, le circuit modifie immédiatement l'intensité de la LED en conséquence.
- Allumer/éteindre avec traitement numérique : avec une fréquence d'échantillonnage définie, toutes les 5 secondes, le système lira l'interrupteur et indiquera si la lumière est allumée ou éteinte, quelle que soit l'intensité et la luminosité de la lumière. Lorsque vous changez le gradateur dans ces 5 secondes, la lumière de la LED ne changera pas.

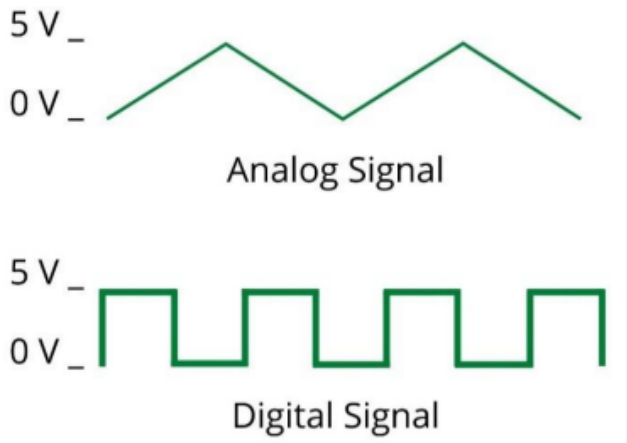


Image 23 – Signal analogique vs numérique

Résumé

Dans le monde réel, les signaux analogiques sont les plus couramment utilisés. Cependant, les signaux et le traitement numériques sont recommandés en matière de robotique. Les raisons sont les suivantes : 1) le traitement numérique est moins cher et offre plus de flexibilité par rapport au traitement analogique ; 2) les programmes fonctionnent sous une forme binaire similaire aux signaux numériques ; et 3) les signaux analogiques sont plus susceptibles d'être affectés par le bruit du circuit électrique.

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Pour ces raisons, les signaux analogiques sont convertis en signaux numériques la plupart du temps. Il convient de noter que tous les signaux analogiques, entrées et sorties, peuvent être convertis en signaux numériques, mais pas l'inverse.

2. Principes de base de la programmation avec Arduino IDE

⇒ **Nombre de participants :** 1 à 10 par animateur

⇒ **Durée :** 3h

⇒ **Méthodes d'enseignement:** Présentation, Instruction guidée, Apprentissage expérientiel

⇒ **Matériel requis :** présentation, IDE et cartes Arduino, ordinateur (1 par participant) et connexion Internet stable

2.1. Configuration de l'IDE Arduino et des commandes de base

Dans cette section, nous apprendrons comment télécharger et exécuter l'IDE Arduino pour la première fois. Nous passerons également en revue certaines fonctions de base de l'IDE Arduino afin de nous familiariser avec le programme.

Comment télécharger l'IDE Arduino :

Les instructions suivantes sont fournies pour télécharger et configurer l'IDE Arduino :

1. Allez sur https://www.arduino.cc/
2. Cliquez sur Logiciel. Il ouvrira automatiquement la page illustrée ci-dessous dans la figure 7.

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

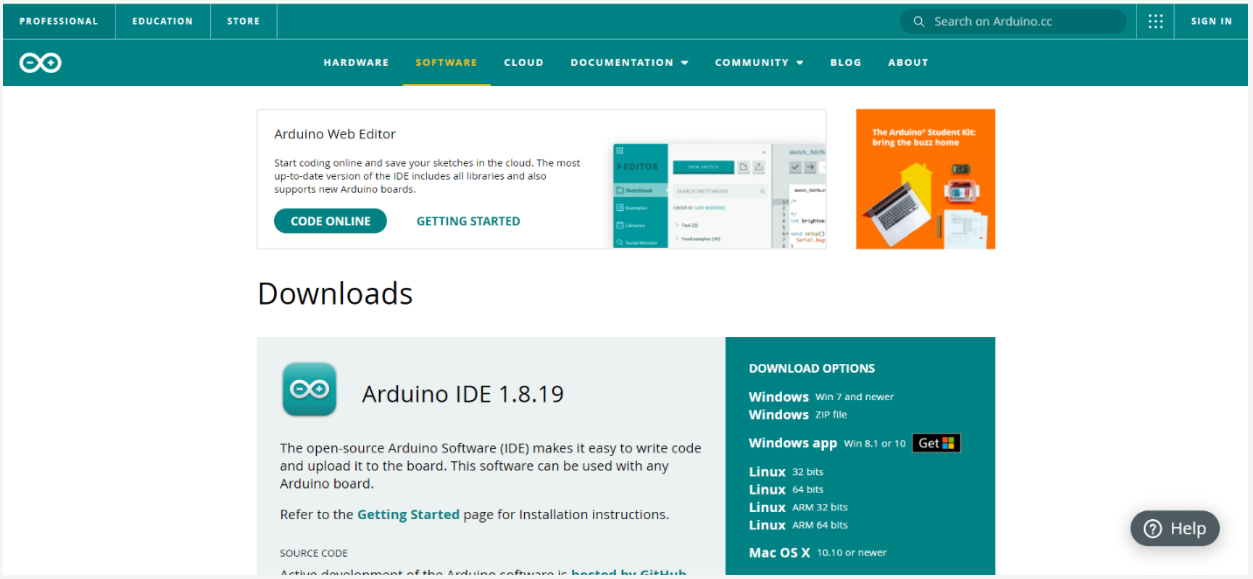


Image 24 – Téléchargement de l'IDE Arduino

3. Cliquez sur l'option de téléchargement qui convient à votre système d'exploitation (voir Figure 7). Si vous avez des doutes, cliquez sur "Getting Started" pour lire plus d'informations sur l'installation du logiciel pour votre ordinateur.

Arduino IDE est utilisé pour créer, ouvrir et modifier des croquis. La carte est définie par les croquis que nous écrivons sur l'ordinateur via l'IDE Arduino. Vous pouvez soit utiliser les boutons affichés en haut de l'IDE ou les éléments du menu.

Une fois le téléchargement terminé, la page ci-dessous s'ouvrira automatiquement :

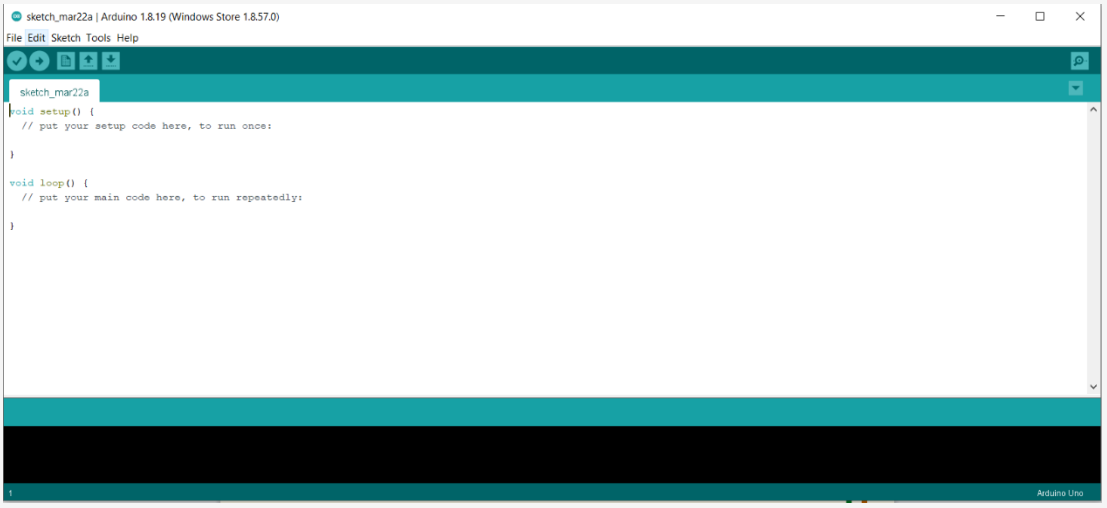


Image 25 – Ouvrir l'IDE Arduino pour la première fois

Comment changer la langue:

1. Cliquez sur FICHIER et sélectionnez PRÉFÉRENCES.

2. À côté de la langue de l'éditeur, un menu déroulant des langues actuellement prises en charge s'affiche.

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

3. Sélectionnez votre langue préférée dans le menu.

4. Redémarrez le logiciel pour utiliser la langue sélectionnée.

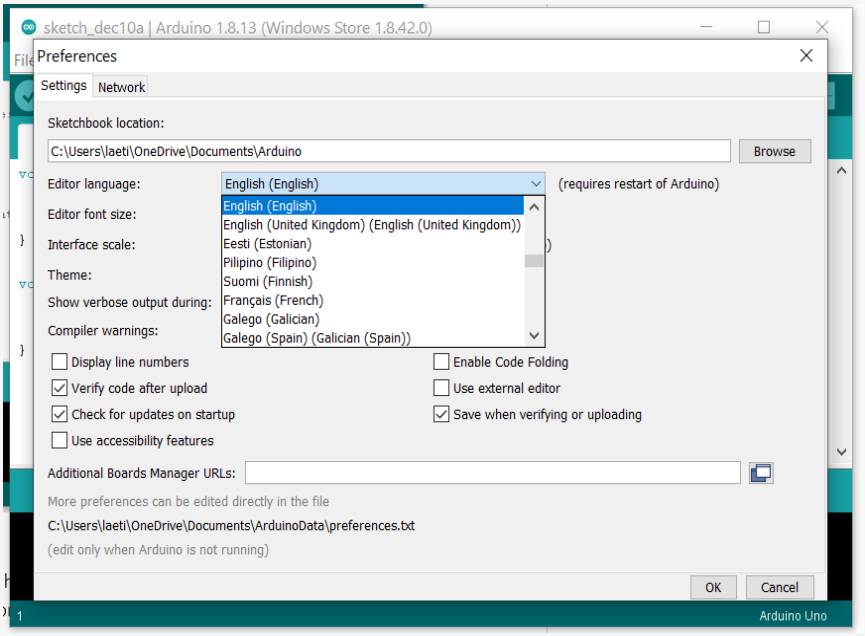


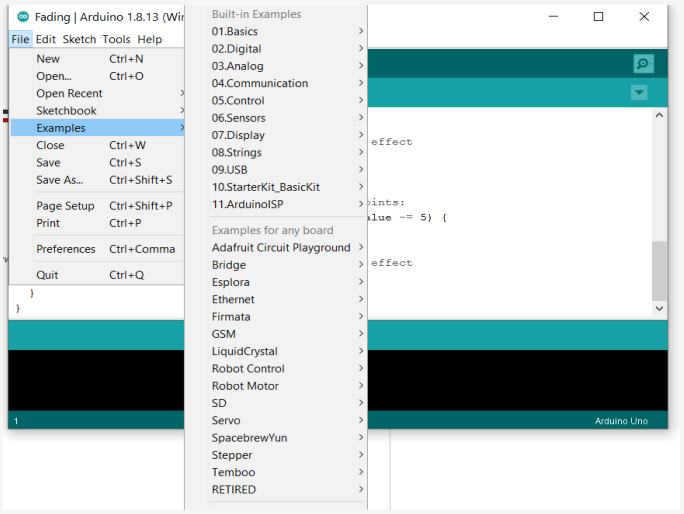
Image 26 – Modification des paramètres de langue dans Arduino IDE

Comment créer un nouveau projet:

Fichier -> Nouveau

Il est également possible d'utiliser des exemples déjà développés pour s'en inspirer ou les reproduire :

Fichier -> Exemples



Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Image 27 – Création d'un nouveau projet dans Arduino IDE

Comment vérifier un projet :

Cliquez à gauche sous l'onglet FICHIER. Il deviendra orange pendant la compilation des informations. Attendez qu'il revienne à sa couleur initiale.

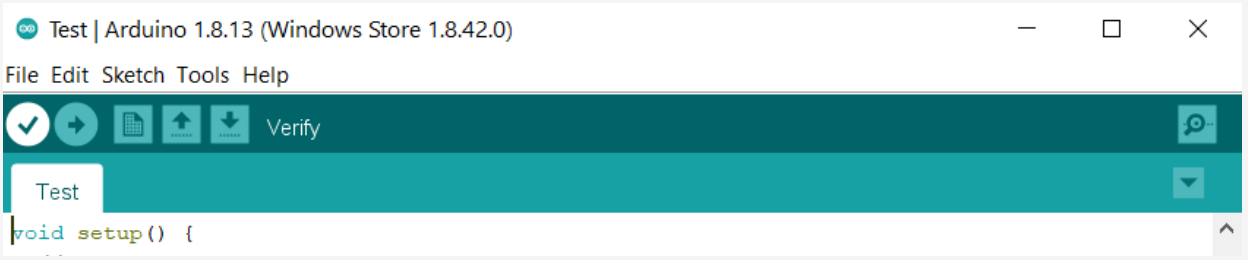


Image 28 – Vérification d'un projet dans Arduino IDE

Comment télécharger un programme:

La carte Arduino doit être branchée à votre ordinateur avec un câble USB pour télécharger le programme. Pour télécharger le programme, vous devez cliquer sur la flèche horizontale :

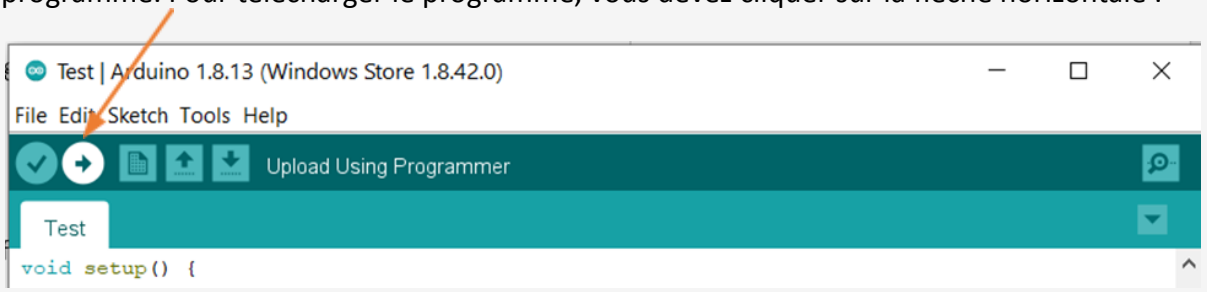


Image 29 – Télécharger un programme dans l'IDE Arduino

Lorsque le programme est chargé, la flèche reprend sa couleur initiale. Si vous avez branché et configuré votre carte Arduino en fonction de ce que vous avez programmé, vous pourrez regarder votre programme en action.

2.2. Programmation dans Arduino et téléchargement de programmes sur la carte

Comment programmer sur Arduino :

Après avoir compris le matériel de la carte Arduino UNO et téléchargé le logiciel Arduino, nous sommes prêts à commencer la programmation.

Les programmes Arduino peuvent être divisés en trois parties principales :

1. Structure,
2. Valeurs (variables et constantes)
3. Fonctions.

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Dans cette session, nous découvrirons le programme logiciel Arduino, étape par étape, et comment nous pouvons écrire le programme sans aucune erreur de syntaxe ou de compilation.

Arduino – Structure du programme

Commençons par la structure. La structure du logiciel se compose de deux fonctions principales :

Setup( ) function *Fonction Configuration*

Loop( ) function *Fonction Boucle*

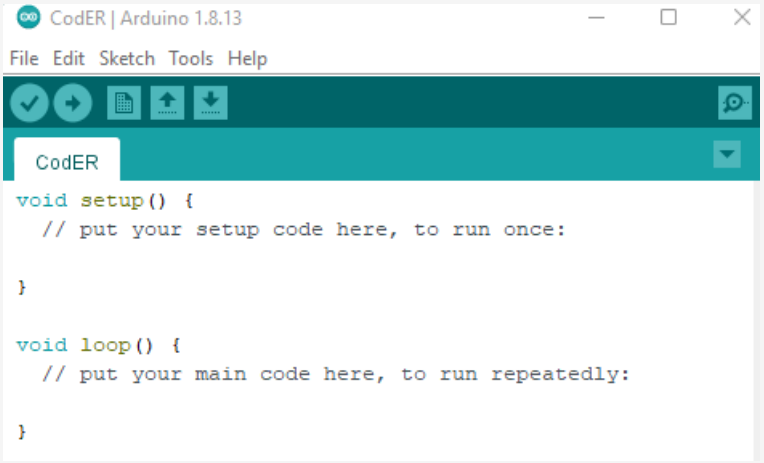


Image 30 - Arduino – Structure du programme

La fonction ***setup()*** est appelée au démarrage d'un sketch. Nous l'utilisons pour initialiser les variables, les modes de broches, commencer à utiliser les bibliothèques, etc. La fonction de configuration ne s'exécutera qu'une seule fois après chaque mise sous tension ou réinitialisation de la carte Arduino. Après avoir créé une fonction setup(), qui initialise et définit les valeurs initiales, la fonction ***loop()*** fait précisément ce que son nom suggère et boucle consécutivement, permettant à votre programme de changer et de répondre. Utilisez-le pour contrôler activement la carte Arduino.

Types de données

L'environnement Arduino est similaire à C++ avec un support de bibliothèque et des hypothèses intégrées sur l'environnement cible pour simplifier le processus de codage. Vous trouverez ci-dessous une liste avec certains types de données couramment utilisés dans Arduino :

- boolean (8 bit): true/false *booléen (8 bits) : vrai/faux*
- byte (8 bit): unsigned number from 0-255 *octet (8 bits) : nombre non signé de 0 à 255*
- char (8 bit): signed number from -128 to 127 *char (8 bits) : nombre signé de -128 à 127.*

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

- word (16 bit): unsigned number from 0-65535 *mot (16 bits) : nombre non signé de 0 à 65535*
- int (16 bit): signed number from -32768 to 32767 *int (16 bits) : nombre signé de -32768 à 32767*
- unsigned long (32 bit) - unsigned number from 0-4,294,967,295 *non signé long (32 bits) - nombre non signé de 0 à 4 294 967 295*

Arduino – Variables

Les variables du langage de programmation C, utilisées par Arduino, ont une propriété appelée portée. Une portée est une région du programme, et il y a trois endroits où les variables peuvent être déclarées. Ils sont:

- A l'intérieur d'une fonction ou d'un bloc, qu'on appelle des variables locales.
- Dans la définition des paramètres de fonction, qui sont appelés paramètres formels.
- En dehors de toutes les fonctions, qui sont appelées variables globales.

Exemple de variables :

Cet exemple crée un entier appelé "countUp", initialement défini comme le nombre zéro. La variable augmente de un dans chaque boucle.

```
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);
}
```

Image 31 – Exemple de variables Arduino.
Source: https://www.arduino.cc/reference/en/language/variables/data-types/int/

Un opérateur est un symbole qui indique au compilateur d'effectuer des fonctions mathématiques ou logiques spécifiques. Le langage C est riche en opérateurs intégrés et fournit les types d'opérateurs suivants :

- Arithmetic Operators *Opérateurs arithmétiques*
- Comparison Operators *Opérateurs de comparaison*
- Boolean Operators *Opérateurs booléens*
- Bitwise Operators *Opérateurs au niveau du bit*
- Compound Operators *Opérateurs composés*

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Plus de détails sur chaque type d'opérateur peuvent être récupérés ici :
https://www.tutorialspoint.com/arduino/arduino_operators.htm

2.3. LED clignotante avec Arduino

Dans cette partie, nous créons et téléchargeons un croquis simple qui fera clignoter une LED à plusieurs reprises en l'allumant et en l'éteignant pendant des intervalles d'une seconde.

Pas à pas:

- Connectez l'Arduino à l'ordinateur à l'aide du câble USB.
- Ouvrez l'IDE.
- Choisissez Tools4Serial Port. Assurez-vous que le port USB est sélectionné et que la carte Arduino est correctement connectée.
- Connectez une LED à la broche numérique 13 de l'Arduino (comme indiqué dans la figure ci-dessous). Une broche numérique peut soit détecter un signal électrique, soit générer une commande. Dans ce petit projet, nous allons générer un signal électrique qui allumera la LED.

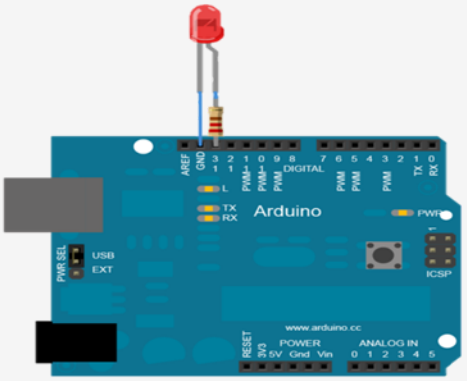


Image 32 – Exemple de broche numérique Arduino avec LED clignotante
Source: https://www.instructables.com/How-to-Blink-LED-Using-Arduino/

- Entrez ce qui suit dans votre croquis entre les accolades { et }, sous le void setup() : pinMode(13, SORTIE); // définit la broche numérique 13 sur la sortie

Le numéro 13 dans la liste représente la broche numérique à laquelle vous vous adressez. Vous réglez cette broche sur OUTPUT, ce qui générera (émettra) un signal électrique. Si vous vouliez qu'il détecte un signal électrique entrant, vous utiliseriez plutôt INPUT. Notez que la fonction pinMode() se termine par un point-virgule (;). Chaque fonction de vos croquis Arduino se termine par un point-virgule.

- Enregistrez à nouveau votre croquis pour vous assurer de ne rien perdre de votre travail en choisissant

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

- Fichier > Enregistrer sous.
- Entrez un nom court pour votre croquis, puis cliquez sur OK.

N'oubliez pas que notre objectif est de faire clignoter la LED à plusieurs reprises. Nous allons créer une fonction de boucle pour dire à l'Arduino d'exécuter une instruction à plusieurs reprises jusqu'à ce que l'alimentation soit coupée ou que quelqu'un appuie sur le bouton RESET.

- Entrez le code affiché en gras après la section void setup() dans la liste suivante pour créer une fonction de boucle vide.
- Terminez cette nouvelle section avec une autre accolade (}), puis enregistrez à nouveau votre croquis.

```
void setup()
{
pinMode(13, OUTPUT); // set digital pin 13 to output régler la broche numérique 13 sur la sortie
}
void loop()
{
// place your main loop code here: placez votre code de boucle principale ici :

}
```

- Ensuite, entrez les fonctions réelles dans la boucle vide () pour que l'Arduino s'exécute.
- Entrez ce qui suit entre les accolades de la fonction de boucle, puis cliquez sur Vérifier pour vous assurer que vous avez tout saisi correctement :

```
digitalWrite(13, HIGH); // turn on digital pin 13 activer la broche numérique 13
delay(1000); // pause for one second faire une pause d'une seconde
digitalWrite(13, LOW); // turn off digital pin 13 désactiver la broche numérique 13
delay(1000); // pause for one second faire une pause d'une seconde
```

La fonction **digitalWrite()** contrôle la tension qui est sortie d'une broche numérique : dans ce cas, la broche 13 vers la LED. En réglant le deuxième paramètre de cette fonction sur **HIGH**, une tension numérique "high" est émise ; alors le courant sortira de la broche et la LED s'allumera.

Avec la LED allumée, la lumière s'arrête pendant 1 seconde avec retard (1000). La fonction **delay ()** fait que le croquis ne fait rien pendant un certain temps, dans ce cas, 1 000 millisecondes ou 1 seconde.

- Ensuite, nous éteignons la tension de la LED avec digitalWrite(13, LOW);

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

- Enfin, nous nous arrêtons à nouveau pendant 1 seconde pendant que la LED est éteinte, avec un retard (1000) ;

Le croquis terminé devrait ressembler à ceci :

```
void setup()
{
pinMode(13, OUTPUT); // set digital pin 13 to output
}
void loop()
{
digitalWrite(13, HIGH); // turn on digital pin 13
delay(1000); // pause for one second
digitalWrite(13, LOW); // turn off digital pin 13
delay(1000); // pause for one second
}
```

- Enregistrez votre croquis.
- Vérifiez votre croquis, pour vous assurer qu'il a été écrit correctement afin que l'Arduino puisse comprendre.
- Une fois le croquis vérifiée, une note doit apparaître dans la fenêtre de message, comme illustré ci-dessous :

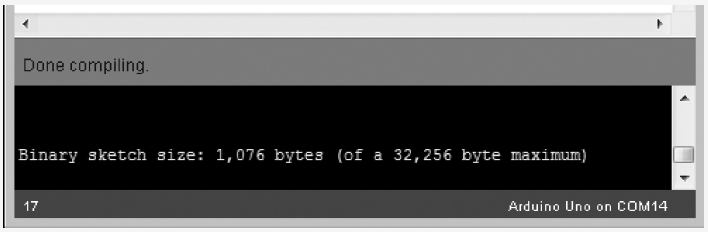


Image 33 – Vérification du croquis dans l'exemple de la LED clignotante

- Assurez-vous que votre carte Arduino est connectée et cliquez sur Télécharger dans l'IDE. L'IDE peut vérifier à nouveau votre croquis et le télécharger sur votre carte Arduino.

Les LED TX/RX de votre carte doivent clignoter pendant ce processus, indiquant que votre programme fonctionne correctement.

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

3. Applications

⇒ **Nombre de participants:** 1-10 par animateur
⇒ **Duration:** 5.5h
⇒ **Méthodes d'enseignement:** Présentation, Instruction guidée, Apprentissage expérientiel

⇒ **Required materials:** Présentation, cartes Arduino et IDE, ordinateur (1 par participant), connexion Internet stable et matériel pertinent en fonction du projet (par exemple, paintbot)

3.1. Qu'est-ce qu'est la robotique?

La robotique est un domaine où la science, l'ingénierie et la technologie se croisent. L'objectif est de produire des machines, appelées robots, capables de se substituer, de reproduire ou d'assister les actions humaines. À l'origine, les robots ont été construits pour gérer des tâches monotones, en particulier dans l'industrie. Mais depuis leur création, ils se sont étendus au-delà de leurs usages initiaux. De nos jours, nous pouvons trouver une énorme variété de robots capables d'effectuer un large éventail de tâches telles que la lutte contre les incendies, le nettoyage des maisons et l'assistance aux médecins lors d'opérations chirurgicales incroyablement complexes. Chaque robot comprend un niveau d'autonomie différent, allant des robots qui sont contrôlés par l'homme dans toutes leurs tâches, aux robots entièrement autonomes qui effectuent des tâches sans contrôle externe.

Le monde de la robotique est clairement en pleine expansion mais on retrouve tout de même des caractéristiques cohérentes :

1. "Un robot est un système mécanique, ce qui lui permet d'interagir avec l'environnement physique et par conséquent d'accomplir certaines tâches précises" (Built In, 2022, §3).
1. "Tout robot nécessite des composants électriques qui contrôlent son alimentation ainsi que l'aspect mécanique" (Built In, 2022,§3).
2. "Les robots contiennent au moins un certain niveau de programmation informatique. Sans un ensemble de code lui disant quoi faire, un robot ne serait qu'un morceau de simple machinerie. Insérer un programme dans un robot lui donne la capacité de savoir quand et comment effectuer une tâche " (Built In, 2022, §3).

3.2 Types de robots

Dans le monde d'aujourd'hui, nous pouvons trouver une gamme de robots utilisés pour diverses applications. Au fur et à mesure que la technologie progresse, les robots deviennent de plus en plus importants dans notre vie quotidienne où ils jouent un rôle important. Il existe de nombreux types de robots et ils varient beaucoup selon la taille, la

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

fonction, du petit robot domestique aux robots industriels géants, du robot pédagogique au robot de loisirs.

Nous pouvons classer les robots de différentes manières, en tenant compte de leurs types et de leurs applications. Chaque robot a ses propres caractéristiques uniques et sa taille, sa forme et ses capacités peuvent varier considérablement. Néanmoins, de nombreux robots partagent un éventail de fonctionnalités. Voici quelques catégories que nous pouvons utiliser pour classer les robots.

Robots industriels

Dans le domaine de l'industrie nous pouvons retrouver 6 types de robots: les robots articulés, les robots cartésiens, les robots dits SCARA, les robots cylindriques, le robot à configuration delta et les robots polaires.

- Robot articulé: Dès par leur forme et leur façon d'opérer, ces robots ressemblent à un bras humain. Le bras est lié à une base capable de pivoter. Le nombre total d'articulations peut varier, d'un minimum de 2 à un maximum de 10, chaque articulation fournit au robot un degré de liberté supplémentaire (Analytics Insight, 2021).

Image 34 – Robot articulé
Source: https://diy-robotics.com/article/articulated-robots/

- Cartesien: Aussi appelés robots rectilignes ou portiques, les robots cartésiens ont trois articulations linéaires qui utilisent le système de coordonnées cartésien (X, Y et Z). Ils peuvent également avoir une partie mobile supplémentaire pour permettre un mouvement de rotation. Les trois articulations prismatiques délivrent un mouvement linéaire le long de l'axe (Analytics Insight, 2021).

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

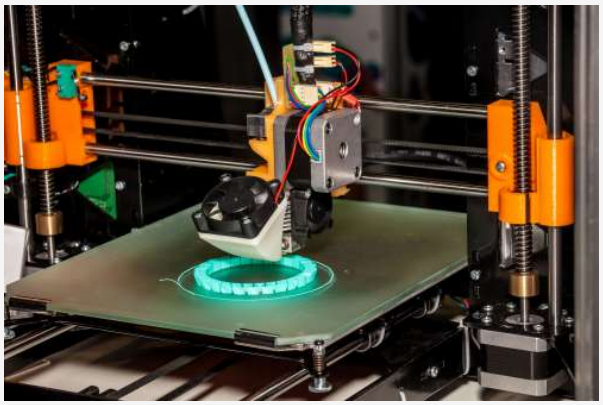


Image 35 – Robot cartesien
Source: https://diy-robotics.com/article/what-you-should-know-about-cartesian-robot/

- SCARA: Le robot de type SCARA est plus couramment utilisé à des fins d'assemblage dans le monde entier en raison de son montage facile et sans obstruction (Analytics Insight, 2021).

Image 36 – Bras SCARA
Source: https://diy-robotics.com/article/scara-robots/

- Cylindrique : Ces robots sont généralement utilisés à des fins d'assemblage, de soudage par points et de moulage mécanique sous pression. Ils ont au minimum un joint tournant à la base et au moins un joint prismatique pour relier les maillons. Le joint rotatif utilise un mouvement de rotation le long de l'axe du joint, tandis que le joint prismatique se déplace dans un mouvement linéaire. (Analytics Insight, 2021).

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

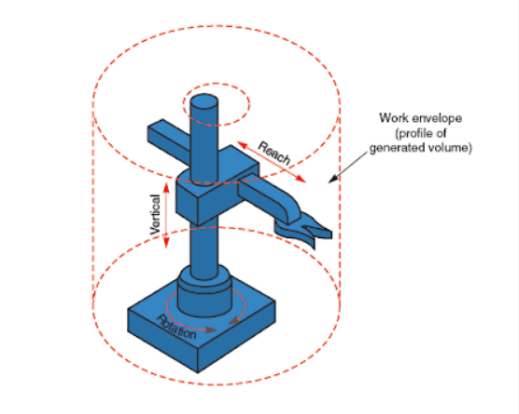


Image 37 – Robot cylindrique
Source: https://learnmech.com/cylindrical-robot-diagram-construction-applications/

- Robots Delta: Aussi appelés «robots araignées», ils utilisent trois moteurs montés sur la base pour actionner les bras de commande qui positionnent le poignet. Les robots Delta de base sont des unités à 3 axes, mais des modèles à 4 et 6 axes sont également disponibles. (Analytics Insight, 2021).

Image 38 – Robots Delta
Source: https://diy-robotics.com/article/top-six-types-industrial-robots-2020/

- Polar: Dans cette configuration, le bras est relié à la base par une articulation torsadée et une combinaison de deux articulations rotatives et d'une articulation linéaire. Les axes forment un système de coordonnées polaires et créent un environnement de travail de forme sphérique(Analytics Insight, 2021).

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

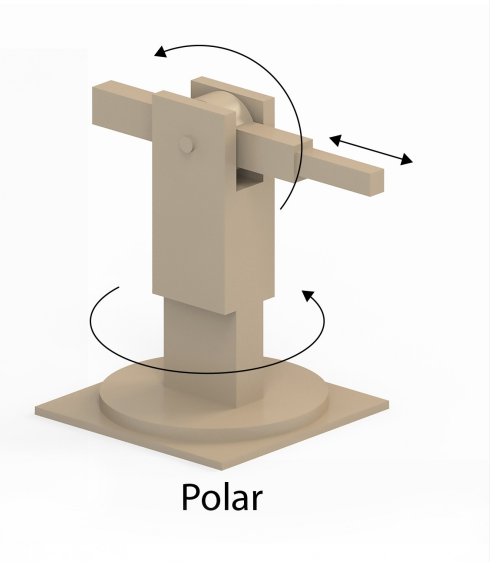


Image 39 – Robots polaires
Source: https://www.howtorobot.com/expert-insight/industrial-robot-types-and-their-different-uses

Outre les robots industriels, il existe un nombre illimité de types de robots dans tous les domaines tels que l'éducation, la sécurité, les sciences spatiales, la médecine, etc. Vous trouverez ici quelques exemples de ces robots, mais rappelez-vous que vous pouvez en trouver beaucoup plus.

Robots domestiques

Aussi appelés robots grand public, ce sont des robots que vous pouvez acheter et utiliser uniquement pour vous amuser ou pour vous aider dans vos tâches quotidiennes. Ces robots sont utilisés pour effectuer des tâches ménagères telles que des robots de nettoyage de piscine, des robots aspirateurs, des robots de nettoyage de gouttières, des robots assistants alimentés par l'IA et une variété croissante de jouets et de kits robotiques. Cette catégorie peut inclure des robots de divertissement, ils sont conçus pour susciter des émotions humaines pour nous divertir.

Robots militaires et robots d'intervention en cas de catastrophe

Les robots sont également utilisés dans le contexte militaire ou les interventions d'urgence lorsque la situation peut être trop dangereuse pour les humains. Les drones robots, les détecteurs de mines et les dispositifs de détection sont courants sur le champ de bataille mais nécessitent un contrôle direct par les humains. Les véhicules téléguidés empêchent la perte de vies humaines qui se produirait si les soldats étaient sur le terrain au lieu d'être derrière les contrôleurs (Stanford Edu, 2022). Néanmoins, l'utilisation de robots et de l'IA dans les opérations militaires est assez controversée et peut soulever de nombreuses préoccupations éthiques légitimes. De plus, ces types de robots sont capables d'endurer des conditions extrêmes et de traverser des terrains difficiles. Ces robots effectuent des tâches

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

dangereuses comme la recherche de survivants à la suite d'une urgence et l'aide à d'autres activités cruciales sur le site de la catastrophe. (Analytics Insight, 2021).

Robots médicaux

Les robots ont eu un impact énorme sur la médecine. Ils ont commencé il y a environ 35 ans lorsque l'objectif était d'insérer une sonde dans le cerveau pour réaliser une biopsie. "Aujourd'hui, les robots médicaux sont bien connus pour leur rôle en chirurgie, en particulier l'utilisation de robots, d'ordinateurs et de logiciels pour manipuler avec précision des instruments chirurgicaux à travers une ou plusieurs petites incisions pour diverses interventions chirurgicales. Une vue agrandie en 3D haute définition du champ opératoire permet au chirurgien d'opérer avec une précision et un contrôle élevés » (NCBI, 2019). L'un des robots médicaux les plus célèbres est celui produit par da Vinci, approuvé par la FDA en 2000 et il aurait été utilisé pour effectuer plus de 6 millions de chirurgies dans le monde (NCBI, 2019).

Robots de service

« L'Organisation internationale de normalisation définit un « robot de service » comme un robot « qui effectue des tâches utiles pour les humains ou des équipements, à l'exclusion des applications d'automatisation industrielle » (IFR, 2022). Les types de robots de service les plus courants sont les robots de conciergerie et les robots d'accueil. Guardforce Hong Kong, par exemple, a lancé un robot concierge qui automatise l'enregistrement des visiteurs, contrôle les points d'accès grâce à la reconnaissance faciale et offre des informations multimédias, ce qui le rend parfait pour les bâtiments commerciaux, les événements et les expositions. La même entreprise possède un robot d'accueil pour les centres commerciaux qui fournit un guide d'achat, une fonction de poignée de main et la possibilité pour les clients de télécharger des coupons (Guardforce, 2022).

Cobots

Les robots collaboratifs ou « cobots » travaillent avec les humains dans un environnement partagé pour effectuer leurs tâches. Par exemple, le bras robotique Sawyer aide les travailleurs des serres à cueillir les plantes. Un robot Mitsubishi propose du café au kiosque Café X à Hong Kong.

Drones

Aussi appelés véhicules aériens sans pilote (UAV), il s'agit d'aéronefs sans pilote humain à bord. Les drones peuvent être de différentes tailles et avoir différents niveaux d'autonomie. Ce type de robot a envahi un large éventail d'industries à travers le monde et il aide à mener à bien de nombreuses activités telles que le nettoyage de l'atmosphère, la photographie aérienne et la livraison.

Robots humanoïdes

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

C'est probablement le type de robot auquel la plupart des gens pensent lorsqu'ils pensent à un robot. Ces robots ressemblent et peuvent également imiter le comportement humain. Ils effectuent généralement des activités de type humain (comme courir, sauter et porter des objets) et sont parfois conçus pour nous ressembler, même avec des visages et des expressions humains (Built In, 2022).

Robots éducatifs

La robotique éducative est une nouvelle discipline conçue pour initier les élèves à la robotique et à la programmation de manière interactive dès le plus jeune âge. La robotique éducative fournit aux élèves tout ce dont ils ont besoin pour construire et programmer facilement un robot capable d'effectuer diverses tâches, la complexité de la discipline étant toujours adaptée à l'âge des élèves (Iberdrola, 2022).

3.3 Utilisation d'un moteur à courant continu avec un motor shield

En règle générale, les robots et les appareils automatisés contiennent des pièces mobiles. Le mouvement est activé par des moteurs qui sont programmés pour fonctionner d'une certaine manière. Dans cette section, nous explorons la programmation d'un moteur à courant continu de base, un moteur qui peut tourner dans le sens des aiguilles d'une montre et dans le sens inverse des aiguilles d'une montre. Les moteurs à courant continu sont généralement utilisés dans les jouets ainsi que dans les appareils électriques tels que les mixeurs et autres gadgets de cuisine.

Il est possible de faire fonctionner un moteur à courant continu avec une carte programmable (ex. Arduino) en utilisant seulement quelques composants. Cependant, pour des raisons de simplicité, il est généralement admis que la meilleure solution consiste à coupler le moteur à courant continu avec un *motor shield*.

Il existe deux stratégies principales pour câbler un moteur à courant continu à une carte Arduino. La première consiste à utiliser un Mosfet et une diode, la seconde repose sur un composant électronique appelé *motor shield*.

Un *motor shield* nous permet de contourner un certain nombre de câblages compliqués qui sont nécessaires pour faire fonctionner un ou plusieurs moteurs à courant continu.

Il existe plusieurs *motor shields*. Pour l'exercice suivant, vous aurez besoin du *motor shield Adafruit V1*. Attention à bien choisir la Version 1 et non la Version 2 du *motor shield Adafruit*, car la programmation de l'un n'est pas compatible avec l'autre.

Le *motor shield Adafruit V1* contient deux puces L293D. Pour cette raison, il peut piloter jusqu'à 4 moteurs à courant continu simultanément.

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

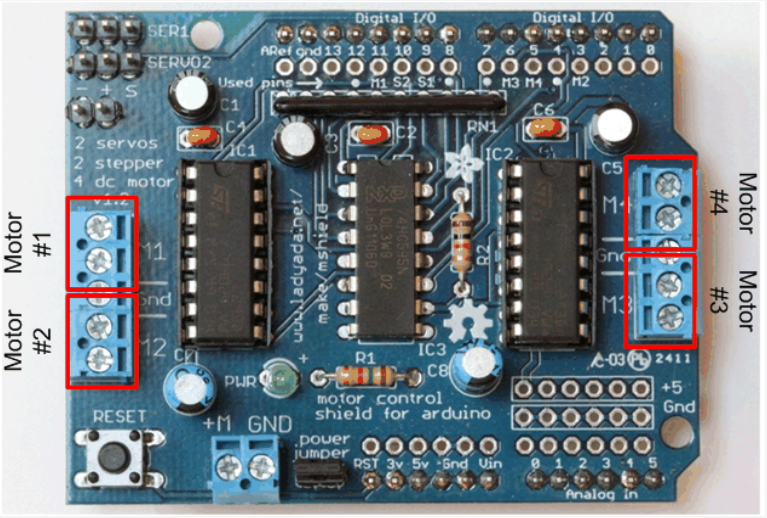


Image 40 – *Motor shield Adafruit V1*
Source: https://learn.adafruit.com/adafruit-motor-shield/af-dcmotor-class

Outre les moteurs à courant continu, il est possible de piloter des servomoteurs et des moteurs pas à pas avec le *motor shield Adafruit*.

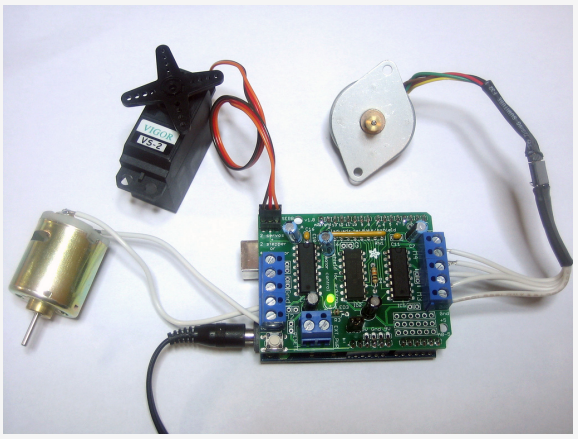


Image 41 – Adafruit motor shield V1 conntected
Source: https://learn.adafruit.com/adafruit-motor-shield/af-dcmotor-class

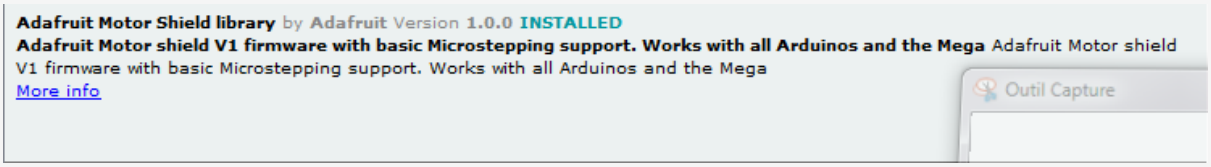
Vous pouvez connecter un moteur à courant continu au *shield* en reliant les câbles du moteur à M1, M2, M3 ou M4.

Lorsque vous avez terminé, vous pouvez commencer la tâche de codage.
Afin de programmer le moteur à courant continu connecté à notre motor shield, nous devons d'abord installer la bibliothèque *Adafruit motor shield*. Vous la trouverez dans le répertoire de la bibliothèque en recherchant « AF motor ».

Allez dans Croquis > Inclure la bibliothèque > Gérer les bibliothèques...

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne



A ce stade, vous pouvez commencer à programmer le *shield*. Un code simple pour faire tourner le moteur à courant continu est le suivant :

```
#include <AFMotor.h>

AF_DCMotor motor1(1); // Nous définissons un moteur attaché à M1 sur le motor shield

void setup()
{
motor1.setSpeed(100); // Nous définissons la vitesse à laquelle le moteur va tourner

}
void loop()

{
motor1.run(BACKWARD); // Le moteur tournera dans le sens des aiguilles d'une montre ou dans le sens inverse des aiguilles d'une montre, selon la façon dont vous l'avez connecté au motor shield. Pour le faire tourner dans le sens opposé, vous devez remplacer BACKWARD par FORWARD.

delay(10000); // Il tournera pendant 10 secondes

 motor1.run(RELEASE); // Il s'arrêtera pendant 1 seconde et recommencera à partir du haut de la fonction de boucle (loop)

delay(1000);
 }
```

3.4. Construisez un robot peinture à l'aide d'un moteur à courant continu et d'Arduino

Il est possible de mettre en œuvre la programmation d'un moteur à courant continu dans un cadre artistique pour réaliser un projet interactif et créatif. Un robot peinture fusionne les arts avec l'électronique, c'est une machine qui utilise de la peinture pour créer des œuvres d'art uniques.

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

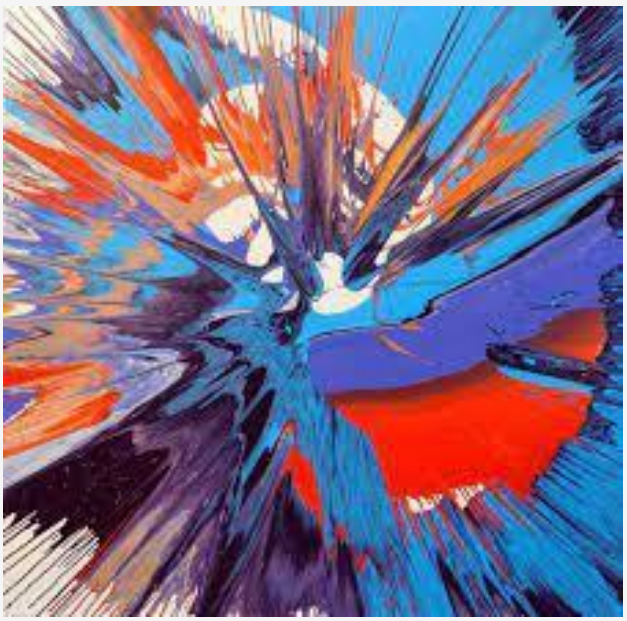


Image 42– Création à l'aide d'un robot peinture
Source: https://girlsinstem.eu/

Le robot peinture se compose d'une table tournante, tournant à des vitesses variables. Sur le plateau tournant, vous pouvez placer une feuille de papier ou une petite toile. En déposant soigneusement des gouttes de peinture au-dessus de la feuille rotative, vous obtenez un chef-d'œuvre unique créé par vous et le robot peinture.

Image 43 – Robots peinture
Source: https://girlsinstem.eu/

Créez la boîte

A. Prenez un morceau de fil rouge et un morceau de fil noir d'au moins 30 cm chacun et connectez-les au moteur. (Assurez-vous de ne pas couper les morceaux trop petits car cela complique la connexion du moteur du robot au reste du circuit électronique. Plus votre boîte est grande, plus les fils doivent être longs)
B. Utilisez la boîte à œufs comme support du moteur

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

1. Retirez le couvercle de la boîte à œufs, nous n'en avons pas besoin.
2. Nous avons besoin de l'un des dessus de la boîte à œufs et de ses 4 pointes environnantes. Couper entre la 3ème/4ème pointe et le dessus adjacent.
3. Coupez la pointe du dessus. Il est préférable de commencer à couper un petit morceau et de l'ajuster ensuite plutôt que de trop couper.
4. Placez le moteur du bas vers le haut avec les fils vers le bas. Le moteur doit être bien ajusté dans le trou que vous venez de couper. Vous pouvez essayer de le pousser avec précaution ou de couper un peu plus du haut. Il est important que le moteur s'adapte fermement et que la construction soit robuste car c'est la forme la base de la table tournante. Pour donner plus de stabilité et de soutien, vous pouvez utiliser du ruban adhésif ou vous pouvez mettre des cure-dents sous le moteur à travers la boîte à œufs.

C. Faites un petit trou au milieu du carton. Passez les fils du moteur à travers le trou allant de l'intérieur de la boîte vers l'extérieur ou le bas de la boîte. Placez le support de la boîte à œufs dessus.

D. Assurez-vous que le support de la boîte à œufs est placé au milieu de la boîte et fixez-le fermement à la boîte. Ceci est très important car il doit résister à la force du moteur en rotation.

E. Placez un disque en carton (rond ou toute autre forme) sur le disque de connexion du moteur rouge (imprimé en 3D). Assurez-vous qu'il est plus petit que la boîte afin qu'il puisse tourner librement.

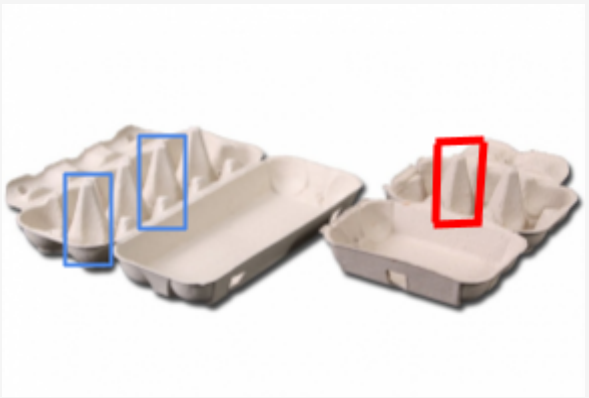
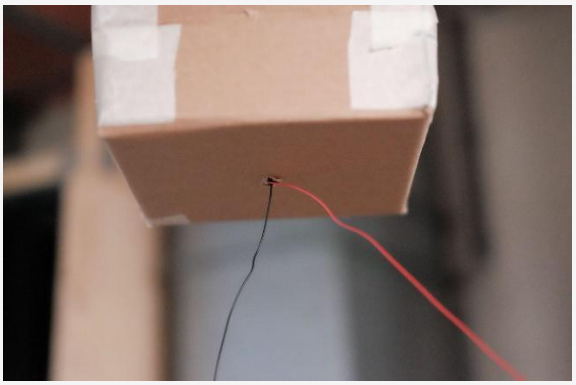


Image 44 – Étape B de construction du robot



Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Image 45 – Étape C de construction du robot

Image 46 – Étape D de construction du robot

Connecter l'électronique

Après avoir créé la boîte, vous pouvez simplement connecter le moteur à courant continu qui est attaché au *shield* du moteur et le programmer pour changer de vitesse et de direction afin de créer votre œuvre d'art !

3.5 Construire un jouet en papier interactif avec un servomoteur

Un servomoteur est un moteur dont la position est vérifiée en continu et corrigée en fonction de la mesure.

On retrouve généralement 3 composants à l'intérieur d'un servomoteur: un moteur DC, un circuit de contrôle et un potentiomètre. Dans un servo moteur, le moteur est connecté à une boîte à vitesses afin d'augmenter la vitesse ainsi que la force de torsion. Le moteur fait pivoter les engrenages. En même temps la puce électronique interprète les signaux qu'elle enregistre et le potentiomètre indique à la puce la position exacte des engrenages, ce qui participe au contrôle de la position exacte du servomoteur.

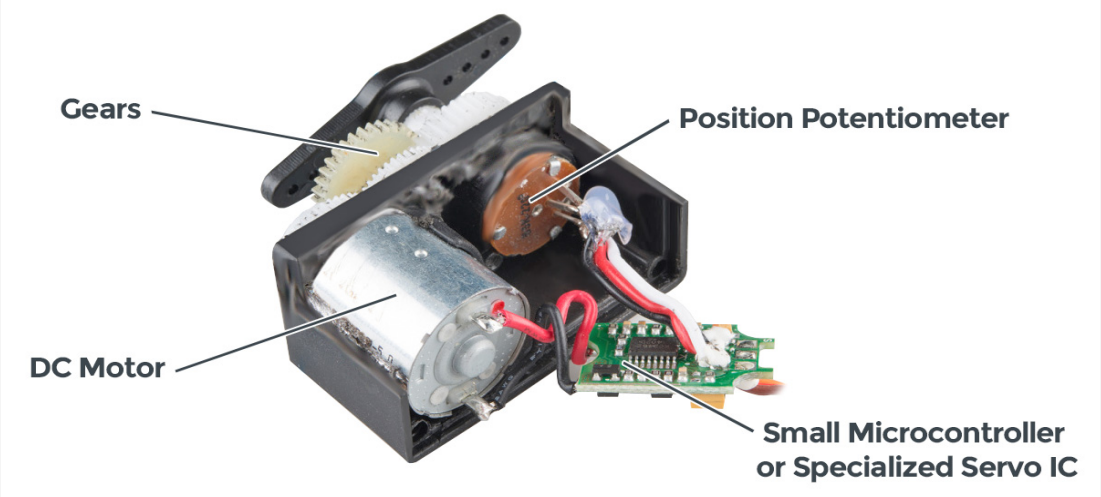


Image 47 - Servo moteur
Source: https://www.sparkfun.com/servos

Les servomoteurs ont d'innombrables applications. Dans cette section, nous illustrons comment utiliser un servomoteur pour créer un jouet en papier interactif dont vous pouvez tourner la tête à gauche ou à droite en agissant sur un potentiomètre.

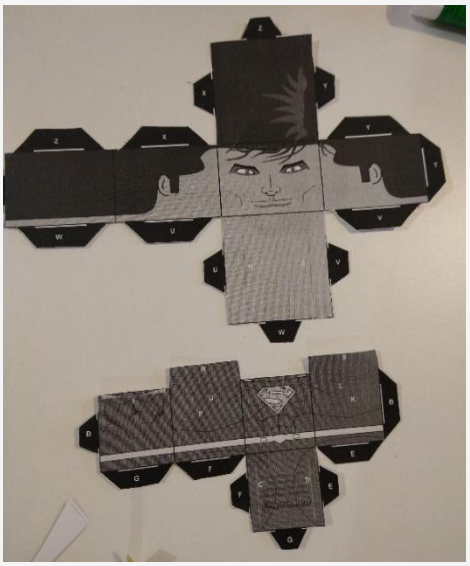
Il n'y aura pas de programmation pour ce projet, cependant, il est possible d'obtenir exactement les mêmes résultats en couplant le servomoteur avec une carte Arduino.

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Première étape: Créer le jouet en papier

Nous allons commencer par créer notre jouet en papier. Tout d'abord, vous devez choisir un modèle, puis découper le modèle dans du papier ou du carton et enfin assembler le jouet en papier selon les instructions.
Différents modèles au choix sont disponibles sur Cubeecraft.

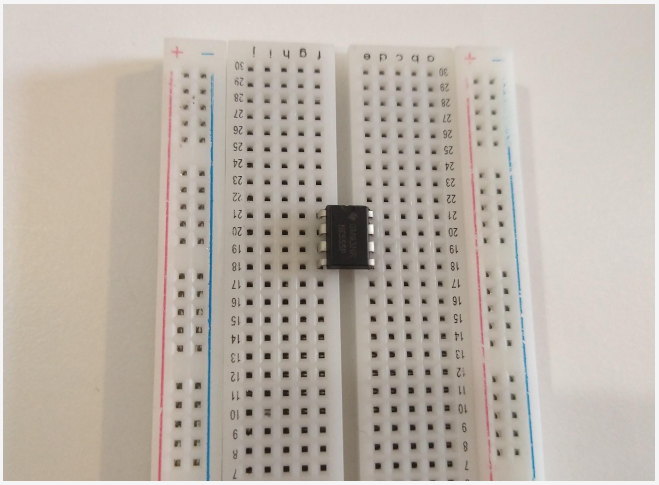


Deuxième étape: Construire le circuit électronique

Nous commençons par placer la puce NE555 au milieu d'une breadboard, comme sur la photo ci-dessous.

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne



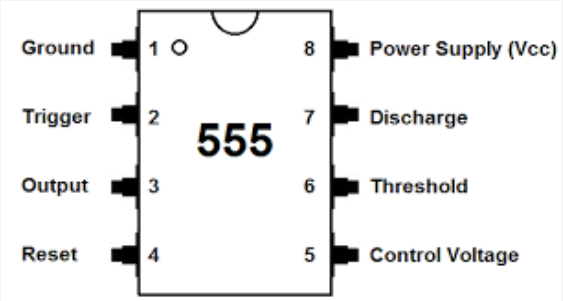
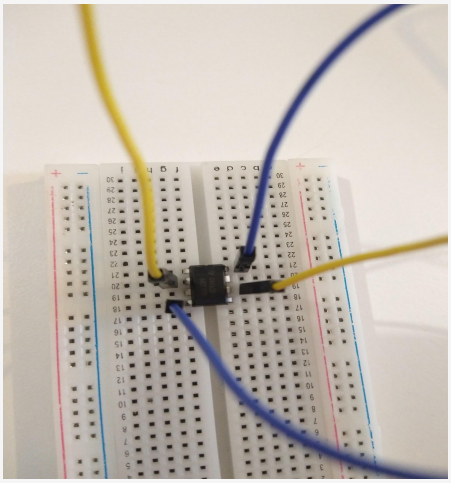
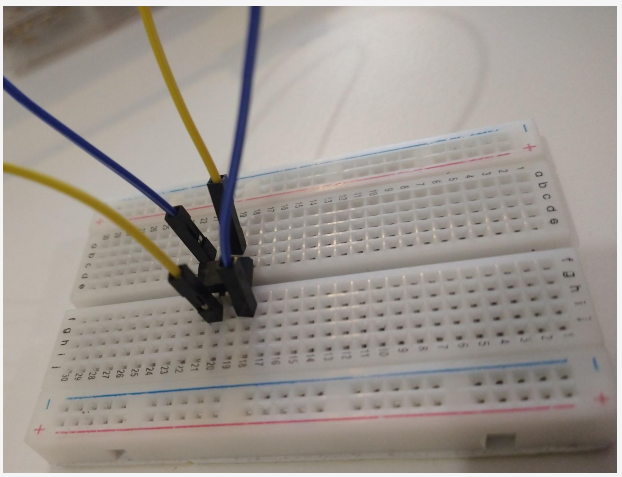


Image 48 – Puce NE555

Source: http://www.learningaboutelectronics.com/Articles/555-timer-pinout.php

Ensuite nous ajoutons deux câbles, un qui relie la patte 4 et la patte 8 du NE555 et un antre qui relie les pattes 2 et 6.

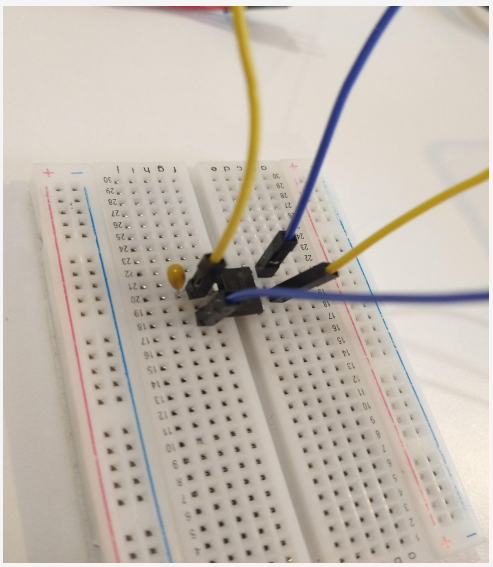


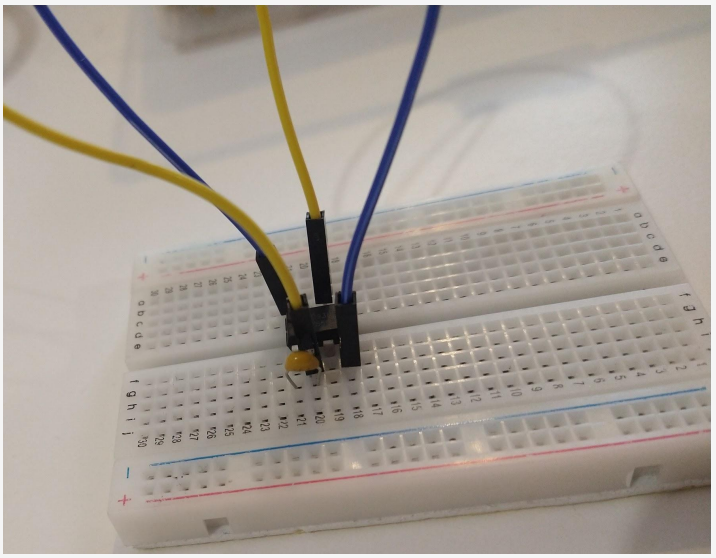


Ensuite, nous ajoutons un condensateur qui relie les pattes 1 et 2.

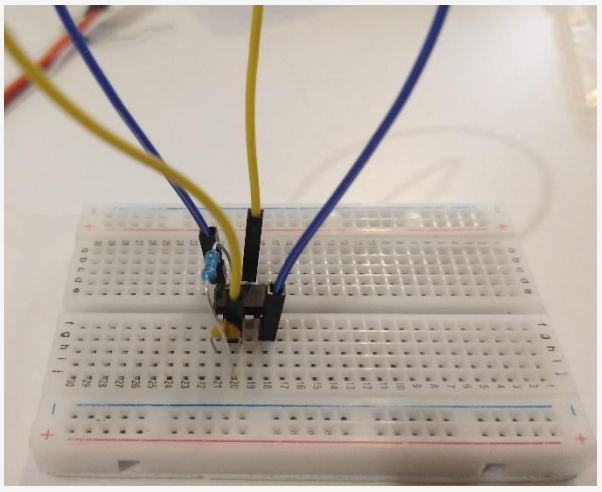
Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

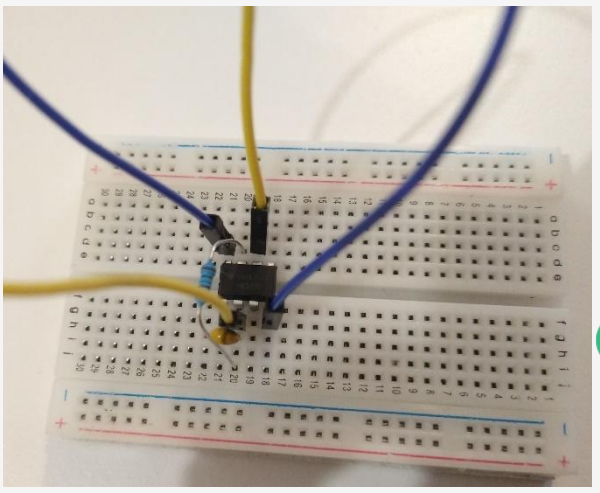
Cofinancé par l'Union européenne





Nous ajoutons une résistance de 220 kohm qui relie les pattes 2 et 7.

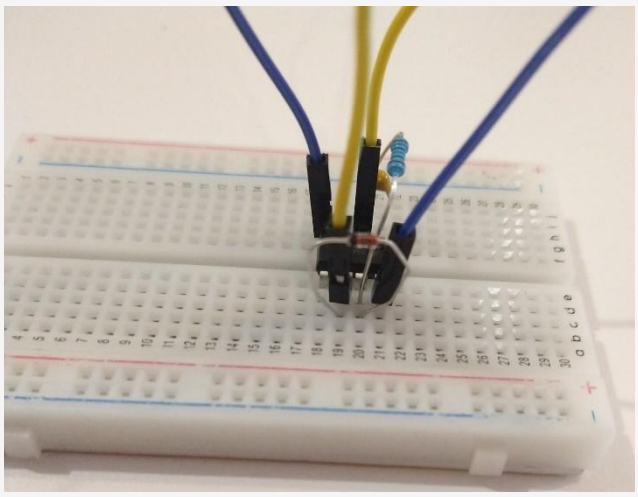


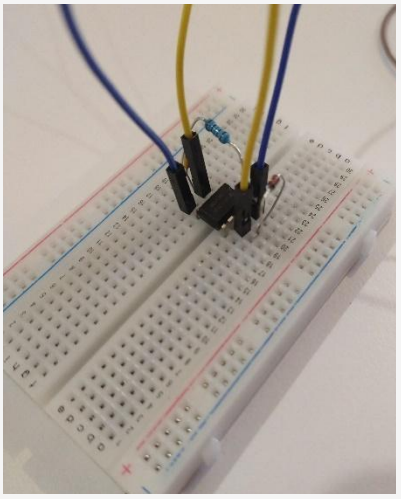


Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

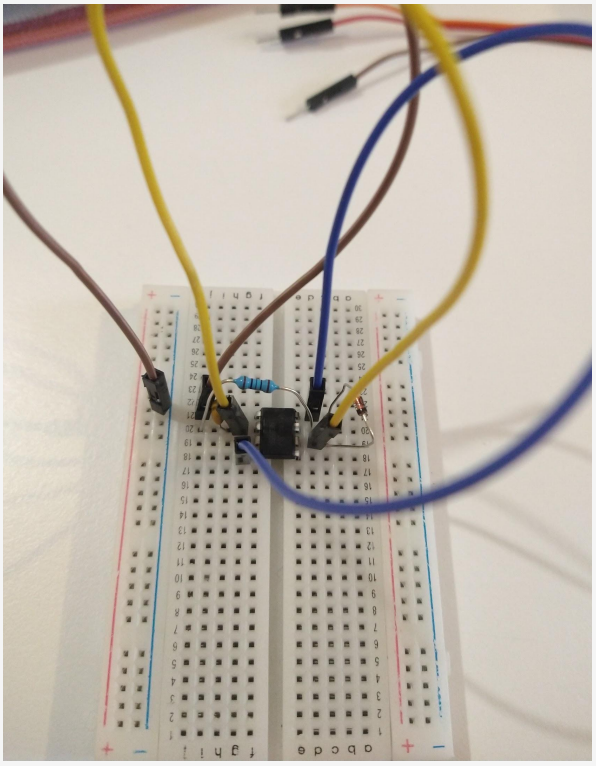
Cofinancé par l'Union européenne

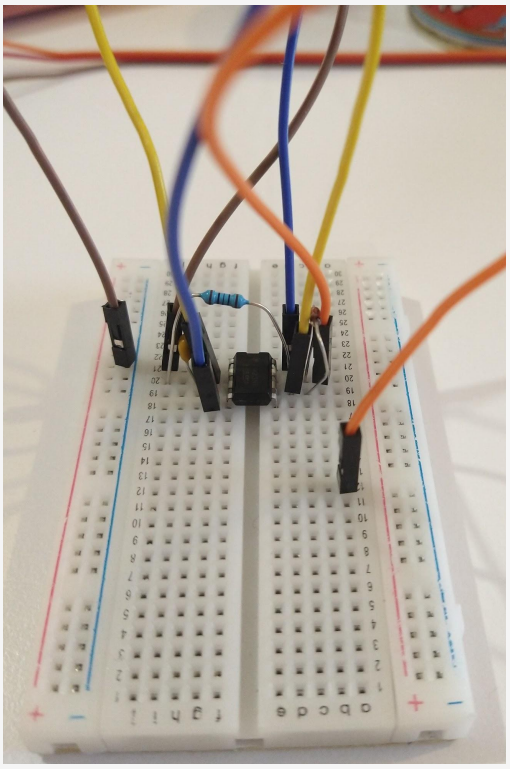
Ensuite, nous rajoutons une diode Zener qui relie les pattes 6 et 7 du NE555 telle que sur les photos. Veuillez mettre la diode dans la bonne direction, c'est -à -dire avec la bande noire sur la patte 6.





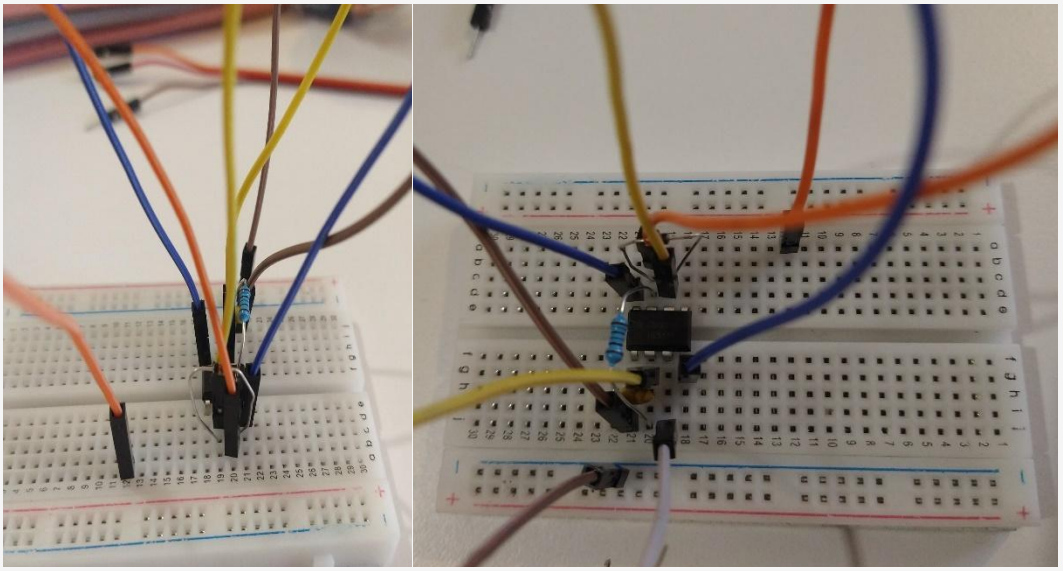
Nous rajoutons un câble qui va de la patte 1 jusqu'au pôle négatif puis un autre qui connecte la patte 7 à une autre ligne plus éloignée de la breadboard. Un nouveau câble va être branché d'un côté à la pâte 3 du NE555 (pour l'instant on ne s'occupe pas de l'autre extrémité du câble).



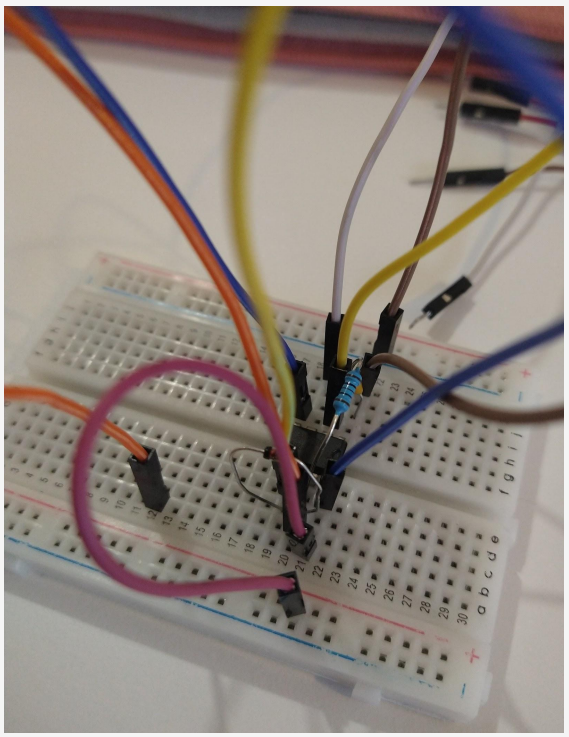


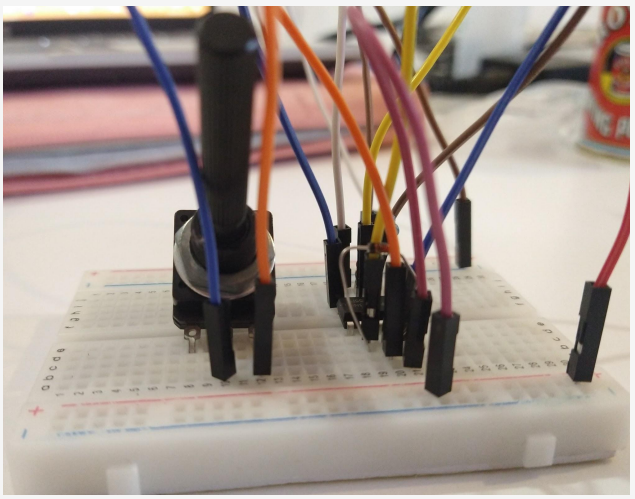
Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne



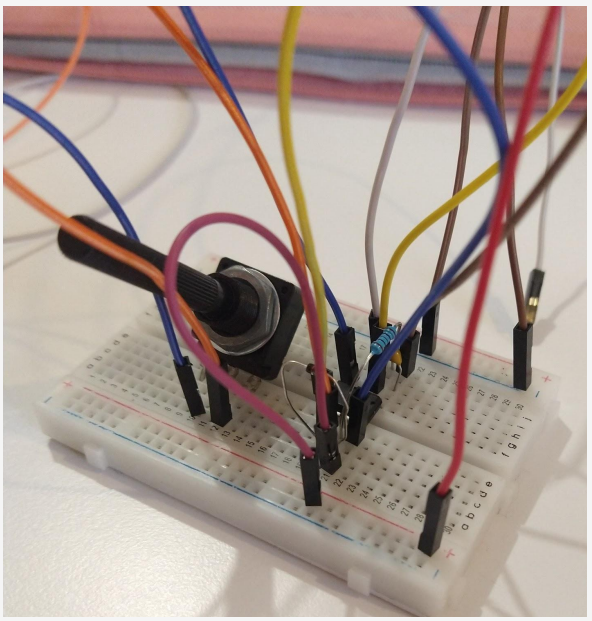
Nous ajoutons un câble qui relie la patte 1 au pôle positif. Puis, nous positionnons le potentiomètre tel que sur la photo ci-dessous. Avec une patte sur la même ligne du câble (orange) qui partait de la patte 7 du NE555. On rajoute, sur la ligne où se situe la patte du milieu du potentiomètre un autre câble qui va rejoindre le pôle positif.



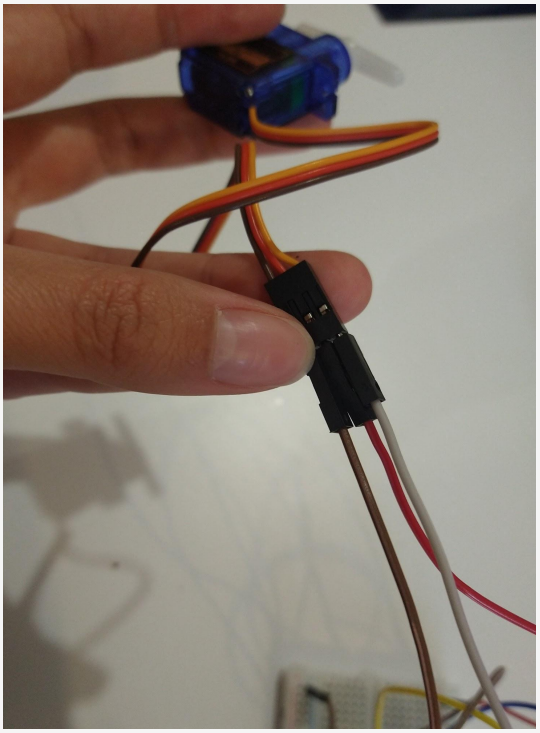


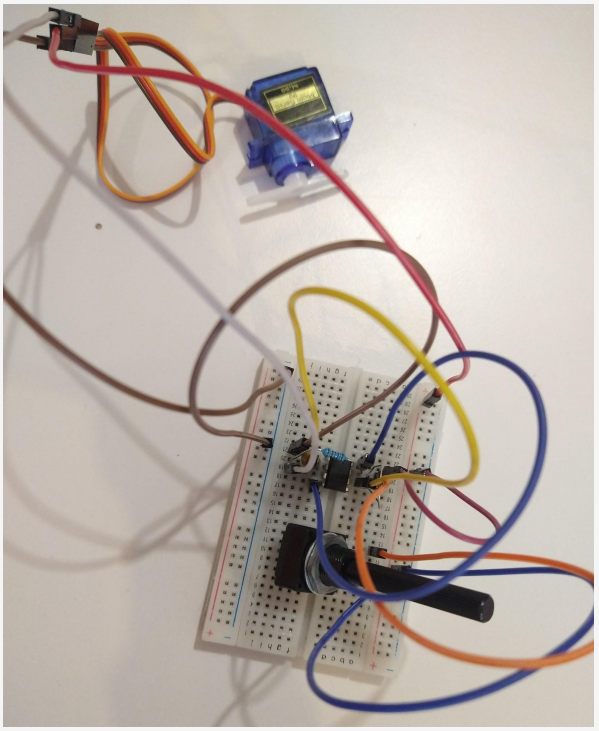
Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne



Nous prenons maintenant le câble (blanc) qui était branché à la pâte 3 du NE555 d'une extremité et on le connecte au câble orange du servo moteur. Aussi on branche un câble au pôle positif de la breadboard et on le relie au câble rouge du servo. On fait pareil avec un câble qui sort du pôle négatif de la breadboard et qui se connecte au câble marron du servo moteur. (Voir les images ci-dessous).

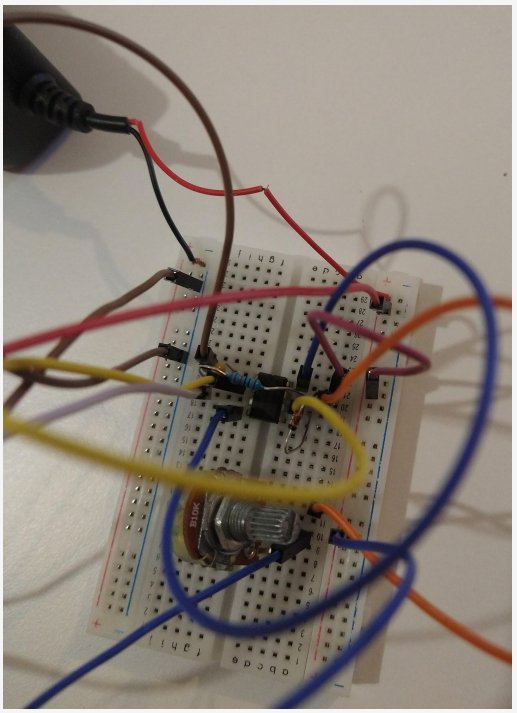




Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Le dernier pas consiste à prendre un adaptateur de 5V, couper l'embout et enlever le plastique pour avoir le câble rouge et le câble noir à disposition. On dénude un peu le deux bouts qu'on va également introduire dans le pôle positif et négatif de manière respective.



Voilà, notre circuit est fini!

Pour terminer le projet, connectez le servomoteur à la tête du jouet en papier et cachez le circuit électronique à l'intérieur d'une boîte en carton. Assurez-vous de percer un trou pour le potentiomètre.

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

3.6. Serrure de porte télécommandée

La communication IR ou infrarouge est une technologie de communication sans fil, peu coûteuse et facile à utiliser. La lumière infrarouge est très similaire à la lumière visible, sauf qu'elle a une longueur d'onde légèrement plus longue. Cela signifie que l'IR est indétectable à l'œil humain - parfait pour la communication sans fil.

Afin de comprendre le fonctionnement de la technologie IR, dans la section suivante, vous allez construire une serrure de porte connectée à une carte Arduino.

Construire une serrure de porte télécommandée qui fonctionne avec la technologie infrarouge

Dans ce projet, nous utiliserons les informations recueillies précédemment sur la technologie infrarouge pour créer une serrure de porte télécommandée.

Vous utiliserez une télécommande infrarouge, un récepteur infrarouge et un servomoteur pour réaliser ce projet.

Première étape: Installez la bibliothèque

Rendez-vous sur ce site Web et téléchargez la bibliothèque. Ensuite, nous devons installer la bibliothèque sur le logiciel Arduino IDE. Accédez à Arduino IDE → Croquis → Inclure une bibliothèque → IRremote.

Deuxième étape: Découvrez la clé de votre télécommande

Afin de comprendre comment votre carte Arduino interprète les signaux électriques envoyés par votre télécommande, nous devons télécharger le code suivant sur la carte.

```
/* Trouver les codes clés de votre télécommande. More info: https://www.makerguides.com
*/

#include <IRremote.h> // Inclure la librairie IRremote

#define RECEIVER_PIN 13 // Définir la broche du récepteur IR
IRrecv receiver(RECEIVER_PIN); // Créer un récepteur de l'IR
decode_results results; // Créer un objet de résultats

void setup() {
  Serial.begin(9600); // Commencer la communication série avec un débit en bauds de 9600
  receiver.enableIRIn(); // Activer le récepteur
```

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

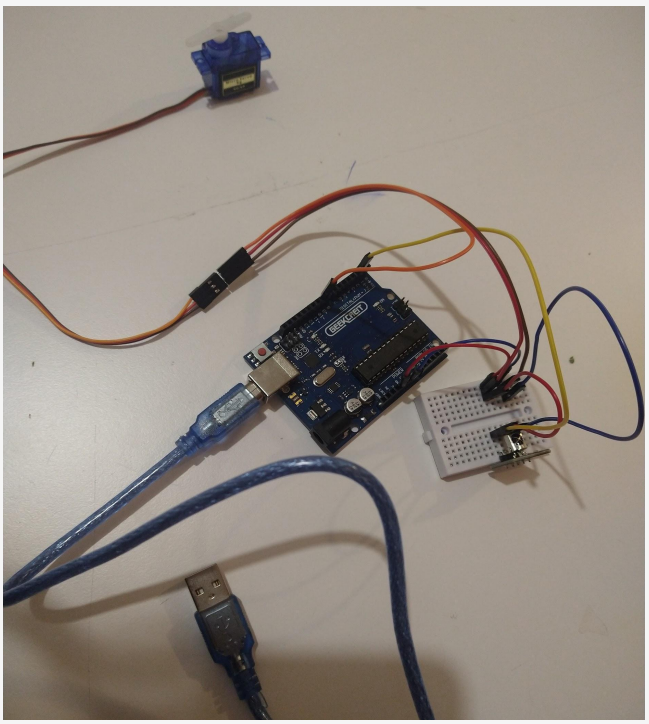
```
 receiver.blink13(true); // Activer le clignotement de la LED intégrée lorsqu'un signal IR est reçu
}

void loop() {
 if (receiver.decode(&results)) { // Décoder le signal reçu et le stocker dans les résultats
  Serial.println(results.value, HEX); // Afficher les valeurs dans le moniteur série
  receiver.resume(); // Réinitialiser le récepteur pour le code suivant
 }
}
```

Ensuite, vous pouvez ouvrir le moniteur série Arduino et en cliquant sur les boutons de la télécommande, vous pourrez voir la clé qui s'affiche sur le moniteur série. Chaque bouton de votre télécommande correspond à une touche différente. Prenez note des différentes clés car vous aurez besoin de ces informations plus tard.

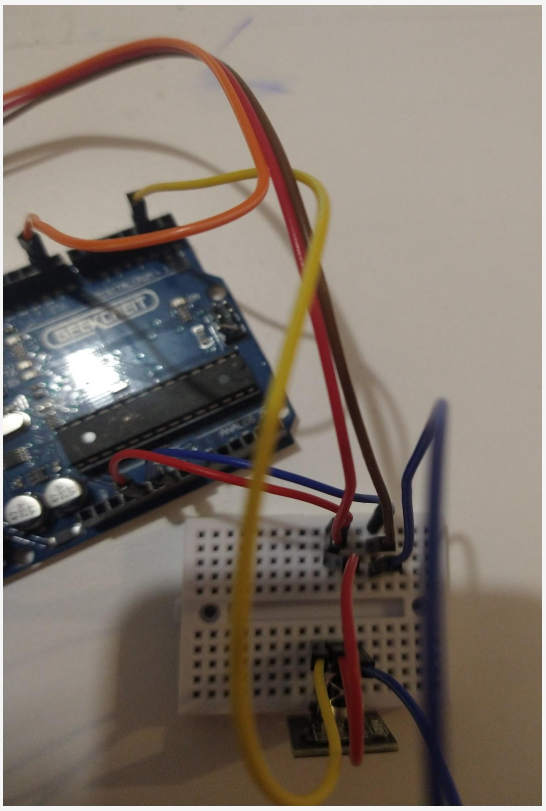
Troisième étape: Câbler tous les composants

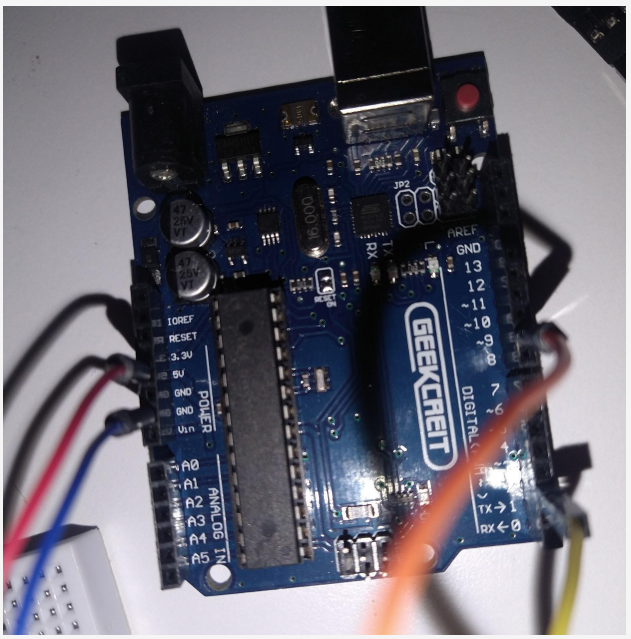
Câblez tous les composants comme illustré dans les images ci-dessous. Soyez prudent avec la réception IR : la sortie positive va au 5V de la carte Arduino, la sortie négative à la terre (*ground*) et la sortie du signal à la broche numérique 2 (voir code ci-dessous).
Notez que la sortie du signal du servomoteur est connectée à la broche numérique 9 de la carte Arduino.

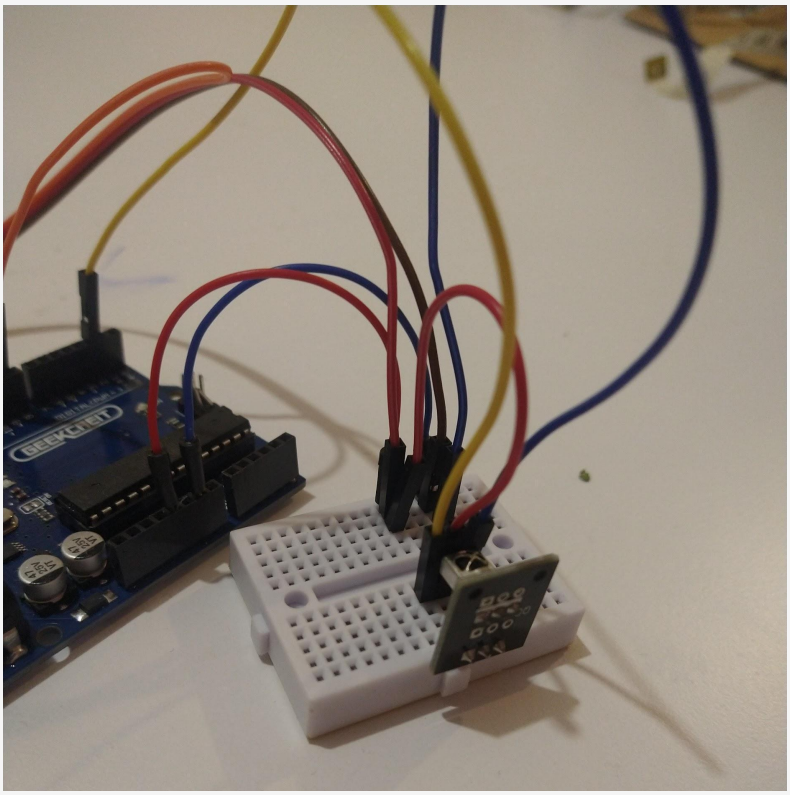


Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne







Quatrième étape: Le code

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Nous devons télécharger le code suivant sur le tableau :

```
#include <IRremote.h>      //Il faut inclure la bibliothèque IRremote dans les bibliothèques Arduino
#include <Servo.h>
#define up 0xFF906F   //Définir le bouton de rotation dans le sens des aiguilles d'une montre
#define down 0xFFE01F //Définir le bouton de rotation dans le sens inverse des aiguilles d'une montre

int RECV_PIN = 2;     //Définir la broche du récepteur IR
Servo servo;
int val;            //Angle de rotation
bool cwRotation, ccwRotation; //Les états de rotation

IRrecv irrecv(RECV_PIN);

decode_results results;

void setup()
{
 Serial.begin(9600);
 irrecv.enableIRIn(); //Démarrer le récepteur
 servo.attach(9);     //Pin du servo moteur
}

void loop()
{
 if (irrecv.decode(&results)) {
   Serial.println(results.value, HEX);
   irrecv.resume(); // Recevoir la valeur suivante

   if (results.value == up)
   {
    cwRotation = !cwRotation;     //Basculer la valeur de rotation
    ccwRotation = false;        //Pas de rotation dans ce sens
   }

   if (results.value == down)
   {
    ccwRotation = !ccwRotation;   //Basculer la valeur de rotation
    cwRotation = false;          //Pas de rotation dans ce sens
```

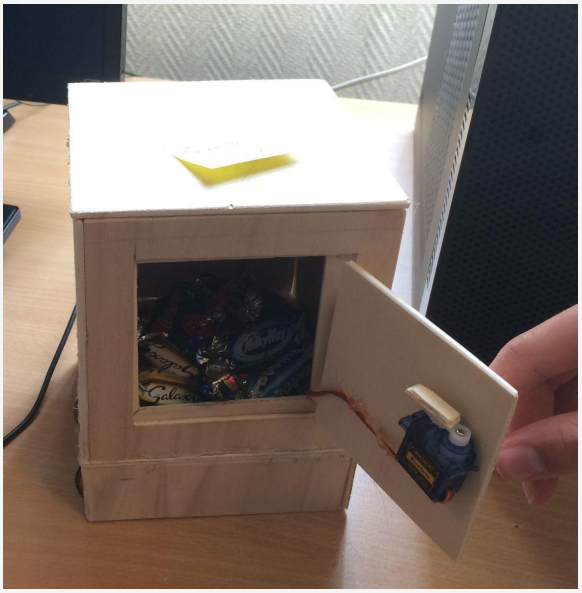
Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

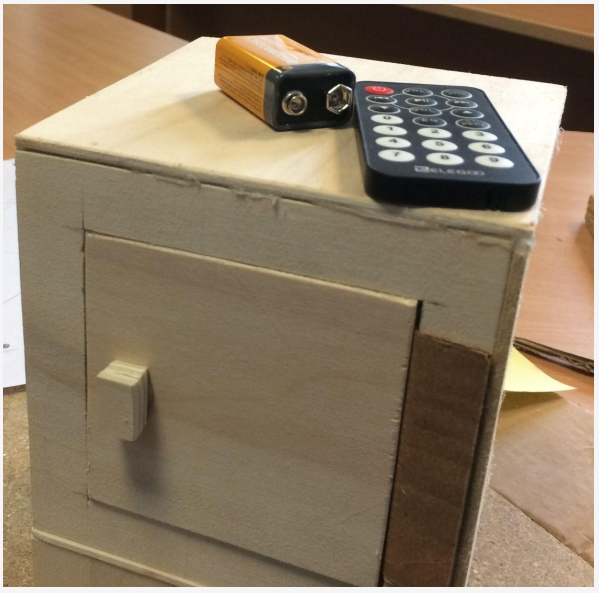
Cofinancé par l'Union européenne

```
  }
 }
 if (cwRotation && (val != 175))  {
  val++;                    //Pour bouton dans le sens des aiguilles d'une montre
 }
 if (ccwRotation && (val != 0))  {
  val--;                    //Pour bouton dans le sens contraire des aiguilles d'une montre
 }
 servo.write(val);
 delay(20);         //Vitesse générale
}
```

Cinquième étape: Créer la serrure de la porte

Vous devriez maintenant avoir testé votre projet. Si le servomoteur répond aux commandes envoyées par la télécommande, alors il est temps de penser à la serrure de porte. Vous pouvez être créatif ici et imaginer n'importe quel type de serrure de porte. Nous suggérons une solution qui consiste à fixer le servomoteur directement sur une porte, comme dans

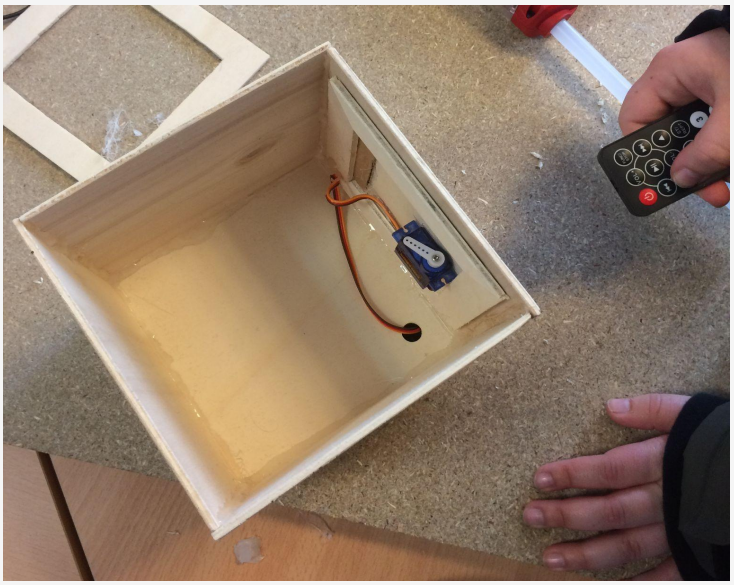




Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

l'image ci-dessous.



3.7. Mesurer la température, l'humidité, la lumière et la couleur

Dans les projets précédents, nous avons utilisé des moteurs pour obtenir différents résultats, allant de la simple conduite d'un moteur dans le sens des aiguilles d'une montre ou dans le sens inverse des aiguilles d'une montre à la construction de gadgets plus sophistiqués tels que le jouet interactif en papier ou la serrure de porte télécommandée.

Maintenant, les moteurs sont des actionneurs, ce qui signifie qu'ils sont destinés à effectuer une certaine action. En plus des actionneurs, on pourrait également choisir d'employer des capteurs. Nous avons en fait exploré quelques capteurs dans le chapitre précédent. Le premier était le potentiomètre, le second était le capteur infrarouge qui reçoit les signaux IR et les traduit en code informatique pour la carte Arduino.

Cette section concerne les capteurs. L'objectif est de vous familiariser avec ce type de composants électroniques pour pouvoir faire passer vos propres projets au niveau supérieur.

En particulier, nous explorerons des capteurs qui mesurent la température, la lumière, ainsi que l'humidité et la couleur.

3.7.1. Utiliser un capteur avec Arduino

Le capteur de lumière est un type de résistance, on l'appelle une résistance dépendante de la lumière. Il est utilisé pour détecter la lumière et aussi pour mesurer le niveau de luminosité d'un certain environnement.

Un capteur de lumière a deux broches et comme il s'agit essentiellement d'une résistance, nous n'avons pas besoin de faire la distinction entre ces deux broches.

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Plus l'intensité de la lumière est élevée, plus la résistance enregistrée par le capteur de lumière est faible. Par conséquent, en mesurant la résistance du capteur de lumière, nous pouvons connaître la luminosité de l'environnement.

Voici comment câbler une photorésistance sur une carte Arduino.

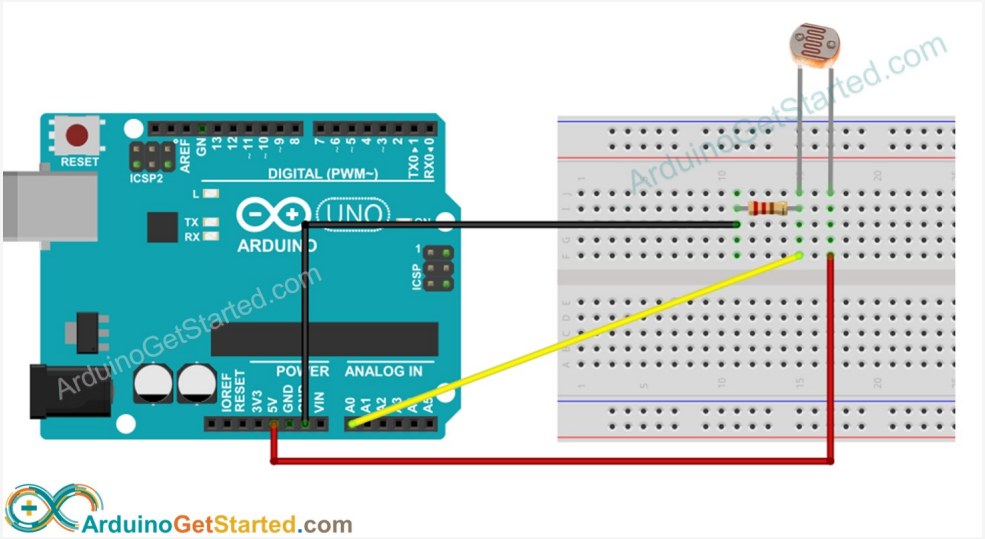


Image 49 – Connecter une photorésistance à une carte Arduino
Source: https://arduinogetstarted.com/tutorials/arduino-light-sensor

Et c'est un code simple qui permet de visualiser les valeurs enregistrées par le capteur de lumière via le moniteur série de l'IDE Arduino.

```
void setup() {
  //Démarrer la communication série à 9600 bits par seconde :
  Serial.begin(9600);
}
void loop() {
  //Lit l'entrée sur la broche analogique A0 (valeur entre 0 et 1023)
  int analogValue = analogRead(A0);
  Serial.print("Analog reading: ");
  Serial.print(analogValue);   //La lecture analogique brute
  delay(500);
}
```

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

3.7.2. Construire un thérémine avec Arduino et un capteur de lumière

Le thérémine est un synthétiseur qui émet un son lorsque vous agitez vos mains devant lui. Il s'agit essentiellement d'un instrument de musique électronique.

Dans ce projet, nous allons fabriquer un instrument similaire qui changera la hauteur de la note lorsque vous agiterez votre main devant.

Nous aurons besoin d'un buzzer piézo, d'un capteur de lumière et d'une résistance de 1k ohm.

Première étape: Câblage

Compléter le câblage selon le schéma ci-dessous:

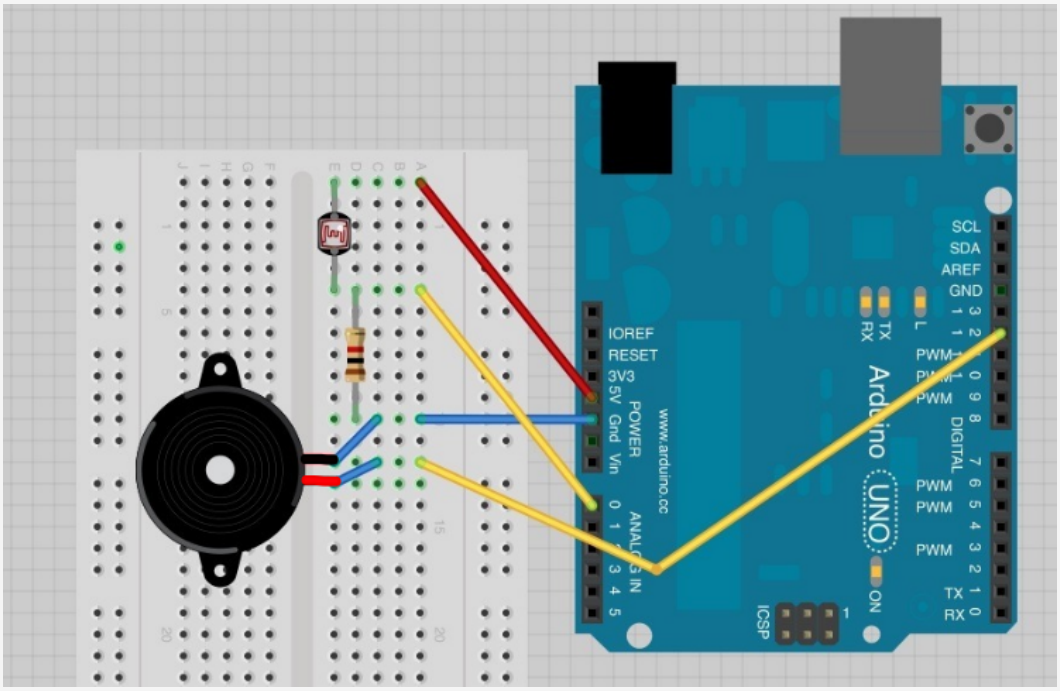


Image 50 – Câblage d'un theremin avec Arduino

Source:
https://learn.adafruit.com/adafruit-arduino-lesson-10-making-sounds/pseudo-theramin

Deuxième étape: Code

```
int speakerPin = 12;

int photocellPin = 0;
```

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

```
void setup()
{
}

void loop()
{
  int reading = analogRead(photocellPin);
  int pitch = 200 + reading / 4;
  tone(speakerPin, pitch);
}
```

Le croquis est en fait très simple. Nous prenons simplement une lecture analogique de A0, pour mesurer l'intensité lumineuse. Cette valeur sera comprise entre quelque chose comme 0 à 700.

Nous ajoutons 200 à cette valeur brute pour faire de 200 Hz la fréquence la plus basse et ajoutons simplement la lecture divisée par 4 à cette valeur pour nous donner une plage d'environ 200 Hz à 370 Hz.

Afin de produire différents effets, vous pouvez essayer de modifier la valeur par laquelle la lecture du capteur de lumière est divisée. Par exemple, remplacez 4 par 2 et voyez ce qui se passe.

3.7.3. Robots sensibles aux couleurs

Comme son nom l'indique, le tri par couleur consiste à trier des objets selon leur couleur.

Évidemment, cela peut être réalisé en regardant chaque objet et en décidant de le placer à un endroit ou à un autre, cependant, lorsque les objets deviennent trop nombreux, cela peut être une tâche fastidieuse et incroyablement répétitive. Dans ce cas, les machines de tri automatique des couleurs peuvent s'avérer très utiles.

Ces machines ont des capteurs de couleur pour détecter la couleur de n'importe quel objet. Après avoir détecté la couleur, un moteur ou un système de moteurs saisit l'objet et le place dans le réceptacle respectif. Les machines de tri des couleurs peuvent être utilisées dans différents domaines où l'identification des couleurs, la distinction des couleurs et le tri des couleurs sont importants. Certains des domaines d'application comprennent l'industrie agricole (tri des grains sur la base de la couleur), l'industrie alimentaire, l'industrie du diamant et de l'exploitation minière, le recyclage, etc.

Machines industrielles de tri par couleur

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Jetons un coup d'œil à certaines machines industrielles de tri par couleur.

Image 51 - Machines industrielles de tri par couleur

Source: http://hugeacademy.com

Les trieurs peuvent être divisés en trieurs de couleur de type goulotte et de type tapis.

Les trieurs de couleur à bande cassent un plus petit pourcentage du matériau (important pour les noix) et le produit reste relativement statique pendant le processus de transport lorsqu'il se déplace horizontalement sur la bande. Dans le type à goulotte, le matériau glisse sur la goulotte provoquant des collisions, des frottements et des mouvements verticaux plus importants, aggravant ainsi le rapport de matériau cassé. La structure de la courroie rend la transmission douce et stable sans rebondissement du matériau.

Les trieurs de couleur de type goulotte sont plus courants, en particulier pour les aliments, car les prix sont plus bas, les capacités sont plus élevées et les produits peuvent être vus plus facilement des deux côtés, ce qui est important lorsqu'un grain décortiqué n'a une coque que d'un côté. Les trieurs de goulotte sont généralement utilisés pour des produits spécifiques, car la goulotte est conçue avec des canaux spéciaux pour ce type de matériau en fonction des tailles et des formes du matériau. Par exemple, des goulottes de 5 mm sont utilisées pour le riz, les céréales et les granulés de plastique. Les goulottes plates conviennent aux flocons de plastique, tels que le PET ou les flocons de bouteilles de lait.

3.7.4. Présentation du capteur DHT11

Il existe un capteur pour à peu près tout, y compris pour mesurer la température !

Le capteur DHT11 est un capteur de température et d'humidité, ce qui signifie qu'il peut mesurer les deux paramètres. Dans ce projet, nous apprenons à câbler un capteur DHT11 à une carte Arduino et à programmer la carte afin de visualiser les valeurs de température enregistrées par le capteur en temps réel.

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Première étape: Câblage

Câblez tous les composants selon l'image ci-dessous :

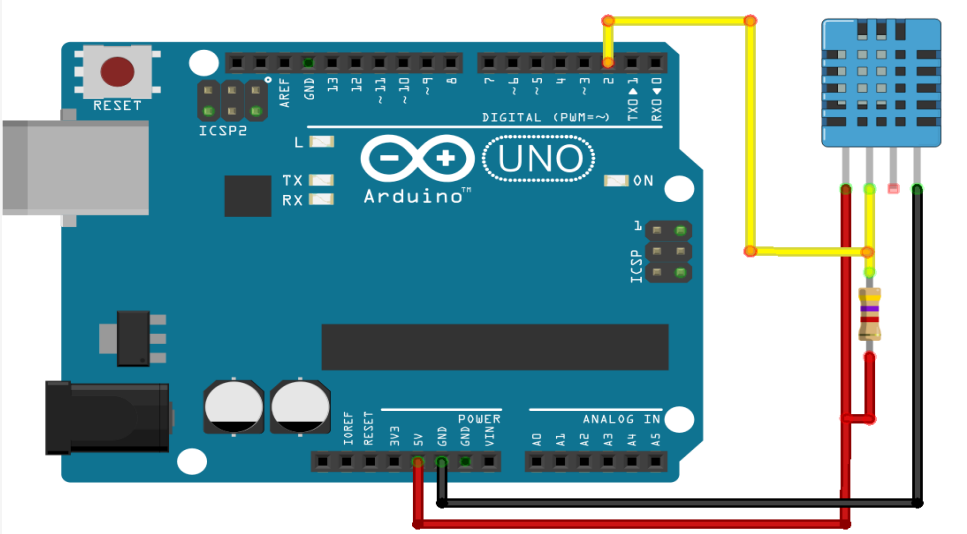


Image 52- Capteur DHT11 sur Arduino UNO
Source:
https://randomnerdtutorials.com/complete-guide-for-dht11dht22-humidity-and-temperature-sensor-with-arduino/

Il est possible d'ignorer la résistance dans le schéma, qui est une résistance de 4,7 kohms.

Deuxième étape: Installer les bibliotèques

Pour lire à partir du capteur DHT, nous utiliserons la bibliothèque DHT d'Adafruit. Pour utiliser cette bibliothèque, vous devez également installer la bibliothèque Adafruit Unified Sensor. Suivez les étapes suivantes pour installer cette bibliothèque.

Ouvrez votre IDE Arduino et accédez à Croqui> Inclure la bibliothèque> Gérer les bibliothèques. Le gestionnaire de bibliothèque devrait s'ouvrir.

Recherchez "DHT" dans la zone de recherche et installez la bibliothèque DHT d'Adafruit.

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

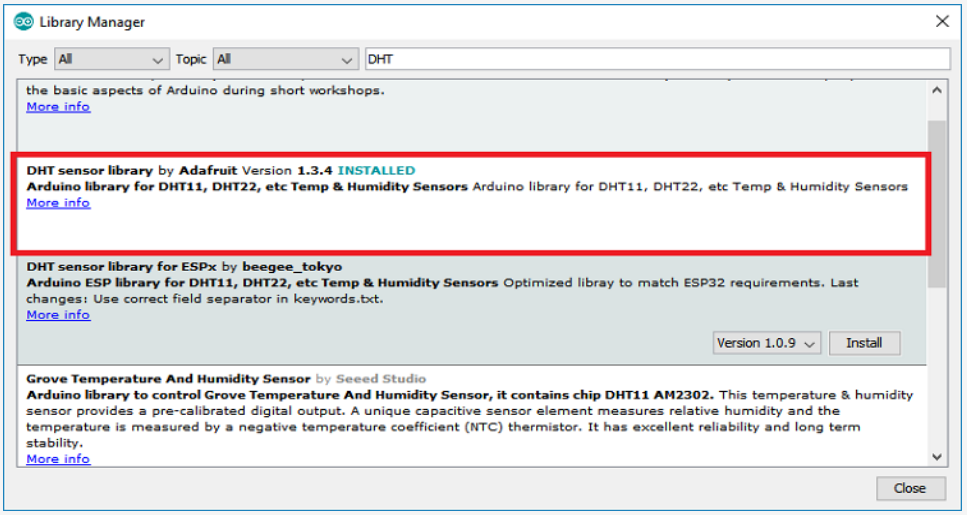


Image 53 – Installer la bibliothèque DHT d'Adafruit

Source:

https://randomnerdtutorials.com/complete-guide-for-dht11dht22-humidity-and-temperature-sensor-with-arduino/

Troisième étape : Le code

```
#include "DHT.h"
#define DHTPIN 2     // Broche à laquelle nous sommes connectés
#define DHTTYPE DHT11   // DHT 11
//Démarrer le capteur DHT pour Arduino 16 MHz
DHT dht(DHTPIN, DHTTYPE);
void setup() {
  Serial.begin(9600);
  Serial.println("DHTxx test!");

  dht.begin();
}
void loop() {
  // Attendez quelques secondes entre les mesures.
  delay(2000);

  // La lecture de la température ou de l'humidité prend environ 250 millisecondes !
```

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

```
  // Les lectures du capteur peuvent correspondre à la réalité des 2 secondes précédentes (c'est un capteur très lent)


  // Lire la température en degrés Celsius
  float t = dht.readTemperature();


  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.print(" *C ");
}
```

3.7.5. Construire un ventilateur de refroidissement intelligent

Dans ce projet, vous allez utiliser vos connaissances sur les capteurs et les actionneurs pour créer un gadget utile et unique : un ventilateur de refroidissement intelligent. Vous utiliserez un moteur à courant continu entraîné par un *motor shield* (nous l'avons exploré dans le chapitre 3.3) plus un capteur DHT11 pour enregistrer la température de votre pièce.

Le résultat final est un ventilateur de refroidissement qui fonctionne lorsqu'un certain seuil de température est atteint, vous pourrez bien sûr programmer le système à votre guise et déterminer vous-même ce seuil de température.

Première étape: Impression 3D du ventilateur de refroidissement

Nous avons trouvé cette conception sur thingiverse. Il s'agit d'un mini ventilateur de refroidissement qui utilise un moteur à courant continu. Il est également possible d'imprimer en 3D un boîtier de batterie pour loger une pile 9v. Cependant, cela ne sera pas nécessaire pour nos besoins car nous utiliserons la pile 9v pour alimenter le *motor shield*.

Essayez d'imprimer le ventilateur et le support. Si vous n'avez pas accès à une imprimante 3D, il est bien sûr possible de créer ces deux éléments en carton ou en contreplaqué ou tout autre matériau suffisamment solide.

Deuxième étape: Tout câbler

Câblez tous les composants selon l'image ci-dessous :

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

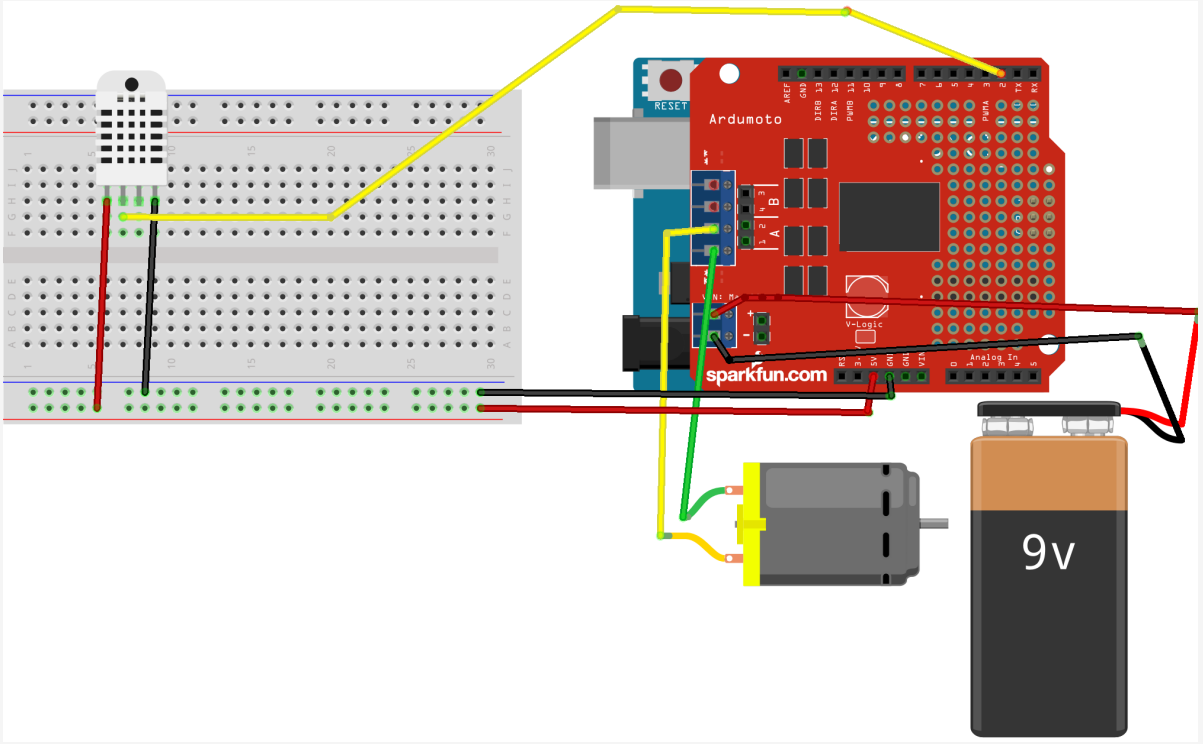


Image 54 – Câblage du ventilateur de refroidissement intelligent
Source: Digijeunes

Troisième étape: Le code

Avec ce programme, nous demandons à la carte Arduino de faire fonctionner le moteur à courant continu si la température enregistrée par le capteur DHT11 est égale ou supérieure à 24°. Sinon, le moteur à courant continu ne tournera pas.

```
#include "DHT.h"

#include "AFMotor.h"

#define DHTPIN 2     //Broche à laquelle nous sommes connectés


AF_DCMotor motor1(1); // Nous définissons un moteur attaché à M1 sur le motorshield


#define DHTTYPE DHT11   // DHT 11


// Démarrer le capteur DHT pour Arduino 16 MHz

DHT dht(DHTPIN, DHTTYPE);
```

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

```
void setup() {
  Serial.begin(9600);

  motor1.setSpeed(100); // Nous définissons la vitesse à laquelle le moteur va tourner

  dht.begin();
}

void loop() {
  // Attendez quelques secondes entre les mesures.
  delay(2000);

  // La lecture de la température ou de l'humidité prend environ 250 millisecondes !
  //  Les lectures du capteur peuvent correspondre à la réalité des 2 secondes précédentes (c'est un capteur très lent)

 // Lire la température en degrés Celsius
  float t = dht.readTemperature();

  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.print(" *C ");

  if (t >= 24)
  {
    motor1.run(BACKWARD);
  }
```

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

```
 else
 {
   motor1.run(RELEASE);
 }
}
```

Références

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/

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/

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

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

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et sera mis en œuvre de décembre 2021 à novembre 2023. Cette publication reflète les opinions des auteurs et la Commission européenne ne peut être tenue responsable de l'utilisation qui pourrait en être faite des informations qui y sont contenues (Code projet : 2021-1-FR02-KA220-YOU-000028696)

Cofinancé par l'Union européenne

Cofinancé par
l'Union européenne

Le projet #CodER est cofinancé par le programme ERASMUS+ de l'Union européenne et est mis en œuvre de décembre 2021 à novembre 2023. Cette publication n'engage que son auteur et la Commission n'est pas responsable de l'usage qui pourrait être fait des informations qui y sont contenues.
Project Number: 2021-1-FR02-KA220-YOU-000028696