Ecole Robots
Apprendre à coder avec les robots éducatifs

Kit de programmation
CURSUS EDUCATION NATIONALE
Python - Niveau lycée

Les bases de la programmation

La syntaxe Python - partie 1

Prénom : _____ Nom : _____ Classe : _____

Les programmes nécessaires à la réalisation des robots sont disponibles en téléchargement sur le site www.ecolerobots.com.

Toutes les boîtes et les pièces détachées sont aussi disponibles sur le site www.ecolerobots.com.

Les bases de la programmation
La syntaxe Python – partie 1

Montage, programmation, robotique.
École Robots – Cursus Éducation Nationale

Ecole Robots
Apprendre à coder avec les robots éducatifs

Chapitre 1 : Avant d'apprendre le langage Python

1. Quels liens entre la programmation et la société ?

Vous les utilisez tous les jours sur votre smartphone ou votre tablette, mais savez-vous au juste comment fonctionnent les applications de messagerie, les réseaux sociaux ou encore les plateformes vidéo ?

Vous utilisez et accédez à ces services, via un navigateur Web ou une application dédiée, grâce à des **serveurs** qui sont des ordinateurs sur lesquels s'exécutent des programmes complexes.

Comme ces services font désormais partie de notre quotidien, il est légitime de se demander comment ces nouveaux services basés sur des technologies de pointe, comme l'IA (Intelligence Artificielle) ou la « blockchain », impactent et font évoluer notre société.

Certains chercheurs se sont déjà penchés sur la question et prédisent que, dans un avenir proche, les ordinateurs et les robots occuperont très probablement des postes aujourd'hui occupés par des humains en raison du développement de l'IA. Outre le fait que de nouveaux emplois liés à l'utilisation de ces technologies devraient émerger, il est probable que les emplois créatifs que seuls des humains peuvent occuper et les emplois demandant des compétences en communication élevées deviennent de plus en plus importants.

Toutes ces technologies qui annoncent cette nouvelle ère impliquent **la programmation**. Or, dans le même temps, on annonce également une pénurie de personnes capables de

comprendre l'informatique et plus particulièrement la programmation pourtant nécessaire au fonctionnement de ces nouvelles technologies.

Avant d'aborder la programmation, examinons de plus près les métiers et tâches nécessitant des compétences en programmation et les avantages qu'il y a à apprendre la programmation.

2. Les métiers nécessitant des compétences en programmation

De nombreux emplois exigent des compétences en programmation. Regardons quelques exemples dans le tableau ci-dessous.

Tâches nécessitant des compétences en programmation	**À quoi sert la programmation ?**
Développement d'applications pour smartphone	Transformer les idées de services et de jeux en applications.
Contrôle de robots et de dispositifs IoT (*)	Utiliser des capteurs pour obtenir des informations sur des sites à distance ou piloter à distance des moteurs pour contrôler une machine.
Automatisation de tâches simples de traitement de données	Agréger automatiquement les données d'une enquête et distribuer automatiquement des e-mails à des heures précises.
Production d'art numérique	Contrôler la lumière et le son pour créer des œuvres numériques.

*IoT : acronyme pour Internet of Things, c'est-à-dire Internet des objets. Désigne les objets connectés à Internet qui s'échangent des informations pour se contrôler mutuellement.

3. Les langages de programmation utilisés dans la société

D'innombrables programmes en cours d'exécution dans le monde sont écrits dans divers langages de programmation. Examinons certaines des fonctionnalités des langages de programmation les plus fréquemment utilisés et ce qu'on peut faire avec.

- **Python**

Langage de programmation né aux Pays-Bas en 1991. Ce langage est très apprécié des débutants parce qu'il permet d'écrire du code source (*) concis et facile à lire. Ce langage est bien adapté aux opérations numériques et est devenu populaire grâce au développement de programmes d'intelligence artificielle tels que l'analyse de données, l'apprentissage automatique (machine learning) et l'apprentissage profond (deep learning) qui sont de plus en plus demandés depuis ces dernières années.

**Le code source : chaîne de caractères, texte ou fichier texte d'un programme écrit dans un*

langage de programmation.

- **Java**

Langage de programmation développé aux États-Unis au début des années 90. Beaucoup de systèmes ont été développés en Java, resté longtemps populaire dans le monde entier. Il permet de développer une très grande variété de systèmes, que ce soient des systèmes Web, des applications Android, des systèmes professionnels utilisés dans les entreprises ou encore des systèmes de contrôle de machines. Les programmes écrits dans ce langage fonctionnent sous Windows et Mac.

- **Langage C**

Langage de programmation né aux États-Unis en 1972. Ce langage permet aussi bien de développer des logiciels « bas niveau » (logiciels très réactifs embarqués dans des avions, des voitures, etc.) que des logiciels « haut niveau » (applications de bureautique, jeux, etc.). Bien que le temps de développement soit long et sa grammaire compliquée, il a souvent été présenté comme le premier langage de programmation à apprendre depuis le milieu des années 1990. L'environnement utilisé dans ce cours pour écrire des programmes en Python est écrit en langage C.

- **JavaScript**

Langage développé aux États-Unis et apparu au milieu des années 90. Ce langage fonctionne dans un navigateur. Lorsque vous affichez une page Web contenant du code JavaScript, le code change dynamiquement la page. Il permet, par exemple, de changer le texte et les images en réponse aux clics de l'utilisateur, d'ajouter une animation qui produit des mouvements complexes, etc.

- **Ruby**

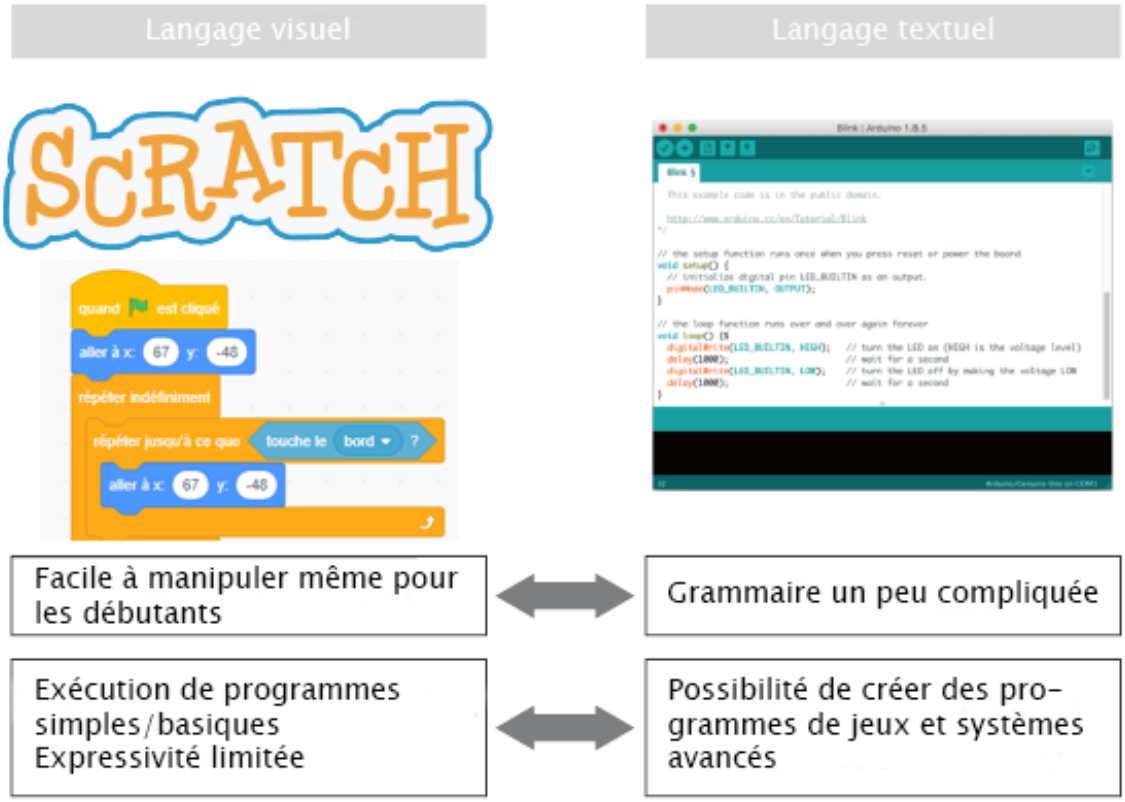
Langage de programmation né au Japon en 1992. Polyvalent et flexible, il est utilisé dans de nombreux systèmes Web, quelle que soit leur taille. Comparé à d'autres langages, il est intéressant pour implémenter de nombreuses fonctions en très peu de code.

- **Swift**

Nouveau langage de programmation développé en 2014 pour les produits Apple. Il reprend les avantages des différents langages de programmation et se maîtrise relativement facilement.

4. Pourquoi apprendre le langage Python ?

Le langage de programmation visuel appelé Scratch est souvent utilisé pour faire ses premiers pas en programmation. Scratch permet de créer un programme simplement en glissant des blocs sur un écran. Bien qu'il soit facile d'utilisation, même pour les débutants, il ne convient pas au développement de jeux et de systèmes complexes. En revanche, si vous maîtrisez des langages de programmation écrits comme Python, vous pourrez créer et réaliser ce que vous souhaitez, même si la syntaxe est un peu compliquée.



Aujourd'hui, Python est l'un des langages de programmation les plus utilisés dans le développement pour créer des applications, contrôler des robots et coder l'intelligence artificielle. C'est pourquoi Python compte un grand nombre d'utilisateurs qui forment une communauté active. Ce langage présente à la fois l'avantage d'obtenir facilement un code source ouvert et de haute qualité, mais aussi celui de raccourcir le temps de développement.

Comme la société évolue avec le développement des technologies de l'information et l'émergence de nouveaux modèles d'intelligence artificielle, de robots ou encore de monnaie virtuelle, il devient de plus en plus urgent et nécessaire d'élargir sa culture informatique et d'acquérir des compétences en programmation. La programmation deviendra, en effet, l'un des piliers de notre société future et sera un sujet abondamment traité dans les journaux télévisés, les médias web et magazines. Pour offrir des clés de compréhension face à ces nouveaux enjeux de société, ce manuel et les suivants vous proposent de maîtriser les bases du langage Python.

Chapitre 2 : Les données en Python

1. Ce que vous allez apprendre dans cette leçon

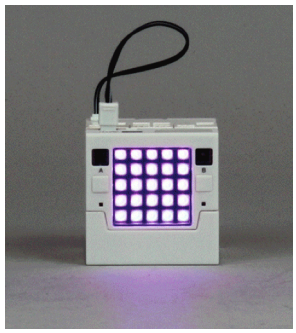
Après une présentation du matériel et du logiciel que vous utiliserez au fil de ces manuels pour apprendre le langage Python, nous nous intéresserons aux données numériques les plus utilisées et aux chaînes de caractères. À la fin de ce chapitre, vous écrirez un programme en Python pour afficher votre nom sur le panneau LED de la carte.

2. Présentation du matériel

Dans ce chapitre, vous acquerrez non seulement les bases du langage Python, mais aussi les commandes de base pour contrôler le panneau LED et le buzzer de la carte ESPeRobo.

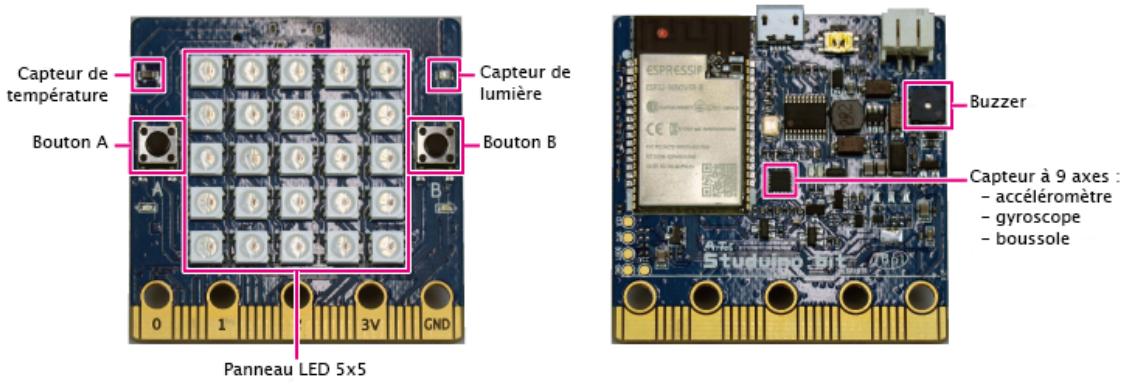
La carte ESPeRobo

La carte ESPeRobo comporte diverses fonctionnalités qui feront l'objet d'exercices de programmation. Vous programmerez, par exemple, les LED à s'allumer en changeant leur couleur selon une séquence que vous aurez établie.



En plus de contrôler le panneau LED et le buzzer de la carte, vous serez amené, au fil de votre apprentissage, à écrire des programmes pour recueillir et mesurer des informations tirées de l'environnement de la carte grâce à ses boutons intégrés, son capteur de lumière, son capteur de température, son accéléromètre, son gyroscope et sa boussole.

Les composants de la carte ESPeRobo
Voyons où les composants se trouvent sur la carte ESPeRobo :



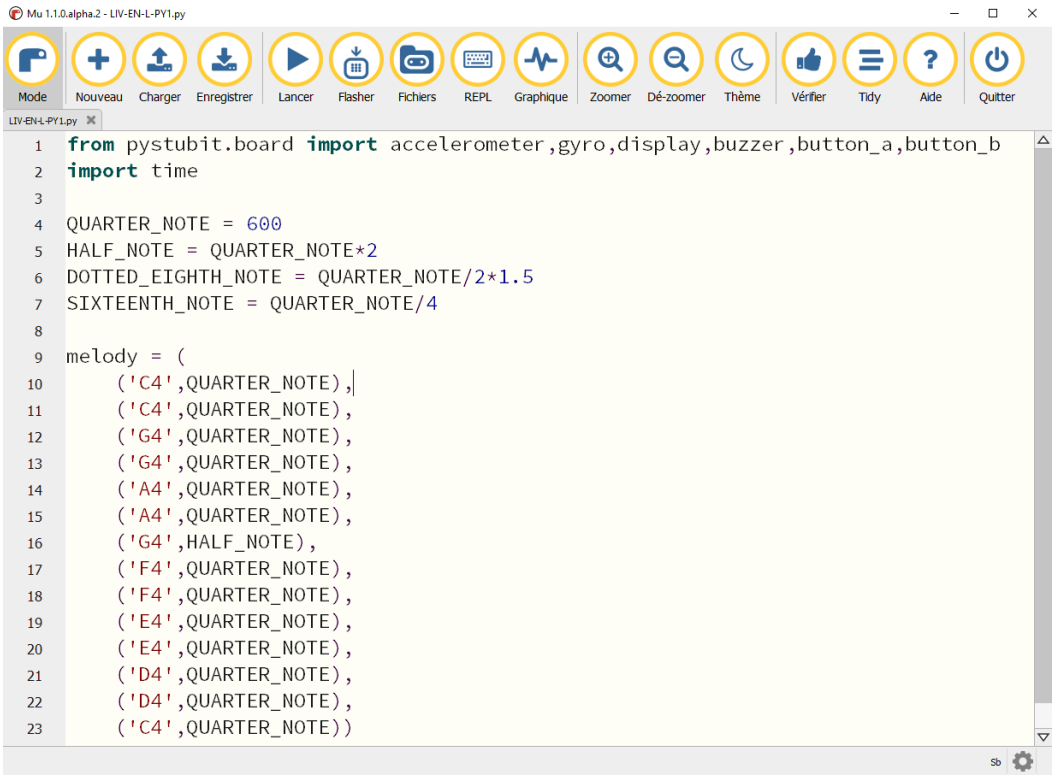
3. Ouvrir un environnement de programmation pour écrire en Python

Pour écrire des programmes en Python, il faut un environnement de programmation. Dans ce cours, vous utiliserez l'environnement de programmation « Mu » distribué en open source (*).

(*) Open source : logiciel dont le code source est publié gratuitement et peut, selon sa licence, être librement modifié ou redistribué par n'importe qui.

3.1. L'interface de l'environnement de programmation « Mu »

L'interface de l'environnement de programmation « Mu » est assez simple et facile à prendre en main par les débutants parce qu'il ne propose aucune fonctionnalité avancée.



3.2. Installer l'environnement de programmation « Mu »

Mu est disponible en version Windows et Mac. Si Mu n'est pas installé sur votre PC, téléchargez le programme d'installation correspondant au système d'exploitation de votre PC à partir du lien ci-dessous et installez-le.

Télécharger Mu pour Windows
https://www.artec-kk.co.jp/artecrobo2/data/mu-editor_64bit.exe

Télécharger Mu pour Mac
https://www.artec-kk.co.jp/artecrobo2/data/mu-editor.zip

Pour apprendre les bases de la programmation en Python, vous intégrerez, dans un 1er temps, quelques exemples de programme dans l'environnement de programmation et les exécuterez pour observer les résultats.

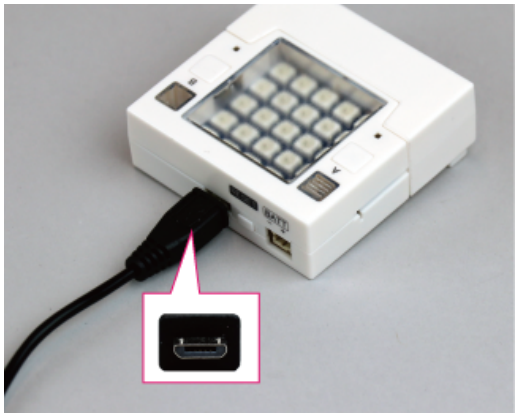
4. Préparatifs avant chaque début des cours

Désormais, avant chaque début de cours, vous devrez faire les préparatifs décrits ci-dessous.

4.1. Connecter l'ESPeRobo

Le programme que vous aurez créé dans l'environnement de programmation devra être transféré de votre PC vers l'ESPeRobo. Pour cela, connectez votre ordinateur à l'ESPeRobo en utilisant le câble USB fourni.

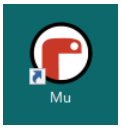
Connectez la grande fiche du câble USB à votre ordinateur.



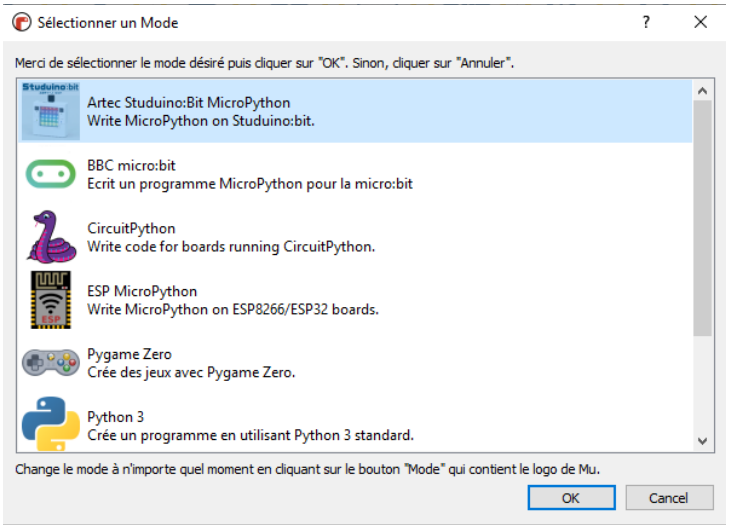
Connectez la petite fiche du câble USB à l'ESPeRobo.

4.2. Ouvrir l'environnement de programmation Mu

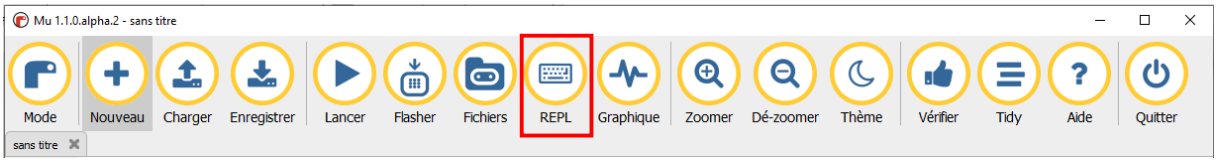
1) Double-cliquez sur l'icône suivante pour lancer « Mu ».



2) Lors du 1er démarrage, une fenêtre vous proposera de sélectionner le mode de démarrage. Sélectionnez Studuino:bit dans la liste et cliquez sur le bouton « OK ». Lors du 2nd démarrage, l'environnement de programmation s'ouvrira automatiquement dans le mode précédemment sélectionné.

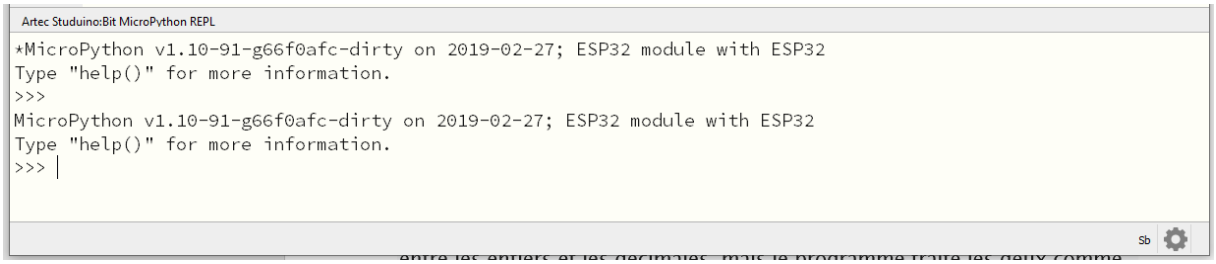


3) Cliquez sur le bouton "REPL*" en haut au centre de l'écran affiché.



* REPL est une abréviation pour « Read-Eval-Print Loop ». Il permet d'envoyer des commandes au robot pendant l'exécution du programme et d'afficher les résultats dans la console.

4) Quand REPL est activé, une console interactive s'ouvre en bas de l'écran, qu'on appelle aussi terminal. À ce stade, le PC et l'ESPeRobo sont toujours en communication. Ne débranchez donc pas le câble USB. Pour quitter le REPL, cliquez à nouveau sur le bouton « REPL ».



5. Les données numériques en Python

En programmation, divers types de données peuvent être traités : nombres, lettres, images, musique ou encore vidéos.

5.1. Traitement des données numériques

La calculatrice est le premier outil auquel on pense pour réaliser des calculs sur un ordinateur. Vous pouvez cependant également faire divers calculs en Python, tout comme sur une calculatrice.

■ Les types de données numériques

Lorsque vous utilisez une calculatrice, vous ne faîtes pas de différence entre les entiers et les décimales. Le programme, lui, en revanche, les traite comme des types de données différents.

- Le nombre entier 0, les nombres naturels (1, 2, ...) et les entiers négatifs (-1, -2, ...) sont traités comme des données entières.
- Les nombres à virgule, les valeurs numériques avec des informations après le point décimal, sont traités, quant à eux, comme des données à virgule.

■ Les opérateurs numériques

En Python, différentes opérations arithmétiques peuvent être effectuées, notamment les quatre opérations arithmétiques : addition, soustraction, multiplication et division.

Dans les langages de programmation, les signes utilisés pour les calculs (ex : +, - , etc.) sont appelés « opérateurs ». Les opérateurs d'addition et de soustraction « + » et « - » sont les mêmes. En revanche, les opérateurs pour les multiplications et divisions ne sont pas

les signes « × » et « ÷ », mais l'astérisque « * » pour la multiplication et la barre oblique « / » pour la division.

Type de calcul	Opérateur
Addition	+
Soustraction	-
Multiplication	*
Division	/

Maintenant, écrivons et exécutons un programme qui effectue des calculs.

■ Addition entière

Commençons par faire une simple addition « 1 + 2 ». Un programme ne nécessitant qu'une seule ligne de code comme ce calcul peut être écrit dans la fenêtre REPL et exécuté immédiatement. Cette fenêtre est généralement appelée « console » ou « terminal ».

Cliquez à côté des signes « >>> » dans la dernière ligne du terminal et écrivez le calcul suivant :

```
>>> 1+2
```

Appuyez sur la touche « Entrée » pour exécuter ce programme. Vérifiez que 3 est bien le résultat qui s'affiche.

```
>>> 1+2
3
>>>
```

* Si vous ne parvenez pas à faire « Entrée » dans le terminal, c'est que la communication entre le PC et l'ESPeRobo a échoué. Assurez-vous que le câble USB est correctement branché, appuyez une 1ère fois sur le bouton REPL pour fermer le terminal, puis une 2nde fois pour le rouvrir.

■ Addition de nombres à virgule

Écrivons maintenant l'opération « 1.2 + 1.8 ». Une fois la formule écrite, appuyez sur la touche « Entrée » pour exécuter le code. Attention, la virgule s'écrit avec « . » et non « , » !

* Comme ce qui est affiché sur le terminal ne peut pas être supprimé, entrez la formule à la suite de la 1ère comme ci-dessous.

```
>>> 1+2
3
>>> 1.2+1.8
3.0
>>>
```

Comme les données sont traitées comme des nombres à virgule, le résultat s'affiche cette fois-ci avec un point décimal « 3.0 ».

■ Soustraction d'entiers et de nombres à virgule

Calculons maintenant la soustraction décimale « 4.9 - 0.9 ».

```
>>> 1+2
3
>>> 1.2+1.8
3.0
>>> 4.9-0.9
4.0
>>>
```

Si le nombre à soustraire est un nombre à virgule, le résultat de l'opération est également traité comme un nombre à virgule.

■ Multiplication d'entiers

Multiplions maintenant les entiers 4 et 9. Pour la multiplication, utilisez l'astérisque « * » au lieu de « x ».

```
>>> 1+2
3
>>> 1.2+1.8
3.0
>>> 4.9-0.9
4.0
>>> 4*9
36
>>>
```

Dans une multiplication, le résultat d'une opération est traité comme un entier si les nombres sont des entiers et comme un nombre à virgule si ce sont des nombres à virgule.

■ Division d'entiers

Calculons la division d'entiers « 54 ÷ 6 ». Pour la division, utilisez la barre oblique « / » au lieu du signe « ÷ ».

```
>>> 1+2
3
>>> 1.2+1.8
3.0
>>> 4.9-0.9
4.0
>>> 4*9
36
>>> 54/6
9.0
>>>
```

Notez que la réponse est « 9.0 » au lieu de « 9 ». Dans la division, même si le quotient est entier, le résultat est toujours traité comme un nombre à virgule.

■ Les autres opérateurs

Outre ces quatre opérateurs, Python dispose d'autres opérateurs servant à d'autres calculs numériques. Un calcul numérique désigne les calculs réalisés sur un système informatique.

Type de calcul	**Opérateur**
Calcul pour trouver le quotient entier d'une division (sans le reste)	//
Calcul du reste d'une division	%
Calcul de la puissance (ex : $10^2 = 100$)	**

Exemples de calculs :

```
>>> 10//3
3
>>> 10%3
1
>>> 2**3
8
>>>
```

■ Priorité des opérateurs

Les opérateurs ont une priorité de calcul. Lorsque plusieurs opérateurs sont inclus dans une expression, l'opération commence par l'opérateur ayant la priorité la plus élevée.

Priorité des opérateurs Python (du plus prioritaire au moins prioritaire) :

Opérateur	Type de calcul
**	Puissance
*, /, //, %	Multiplication, division, division (quotient entier), reste
+, -	Addition, soustraction

Comme avec l'arithmétique normale, si une opération comporte une addition « + » et une multiplication « * », la multiplication « * » a la priorité, sauf si l'opération comporte une expression incluse « ( ) ». Dans ce cas, c'est cette dernière qui a la priorité sur la multiplication.

Exemple de calculs :

```
>>> 9+2*3
15
>>> (9+2)*3
33
>>>
```

5.2. Convertir des données de type numérique

Nous avons pu voir par le calcul de diverses opérations que les nombres entiers et ceux à virgule sont traités comme des données différentes. Dans des langages de programmation comme Python, toutes les données sont traitées selon le type auquel elles appartiennent.

■ Obtenir le type de données numériques

Utilisez la commande type() pour connaître le type auquel appartiennent ces données.

```
type (données dont vous voulez connaître le type)
```

Lancez le programme suivant pour savoir à quel type appartiennent ces données, nombre entier ou nombre à virgule :

```
>>> type(1)
<class 'int'>
>>> type(1.1)
<class 'float'>
>>>
```

'int' et 'float' précédés de « class » désignent un type de données. 'int' désigne un entier et 'float' désigne un nombre à virgule.

* En anglais, les entiers sont appelés « integer number » et les nombres à virgule « floating-point number », d'où les mots 'int' et 'float'.

■ Conversion de données de type numérique

Comment convertir un nombre à virgule (« 9.0 ») en nombre entier (« 9 ») et inversement ? Python propose les instructions suivantes pour convertir un type de données numériques en un autre :

Pour convertir des données en type entier (int), utilisez la commande suivante :

```
int(données à convertir en type entier)
```

Pour convertir des données en type de nombre à virgule (float), utilisez la commande suivante :

```
float(données à convertir en type de nombre à virgule)
```

Tapez les commandes suivantes pour savoir si les types de données sont convertis.
* Notez que les instructions int() et float() sont insérées dans l'instruction type() qui sert à vérifier le type auquel appartiennent ces données.

```
>>> type(float(1))
<class 'float'>
>>> type(int(1.1))
<class 'int'>
>>>
```

Pour mieux comprendre ces résultats, notamment le dernier, regardons comment agit la fonction int() :

```
>>> int(1.1)
1
>>>
```

On peut voir que int() transforme un nombre à virgule en nombre entier en ne conservant que la partie avant la virgule, opération que l'on appelle « tronquer ». Quelle que soit la partie X après la virgule, int(1.X) renverra 1.

6. Les chaînes de caractères en Python

Dans cette partie, nous allons découvrir le type permettant de manipuler du texte en Python (les chaînes de caractères) et comment le convertir vers des types numériques et inversement.

6.1. Créer des chaînes de caractères

Dans les langages de programmation, une chaîne de caractères est un type de donnée qui permet de traiter du texte dans les programmes. Dans cette partie, nous allons voir comment manipuler ce type en Python.

■ Créer des chaînes de caractères

Comme indiqué dans le code suivant, une erreur se produira si la chaîne est écrite et exécutée telle quelle :

```
>>> hello
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'hello' isn't defined
>>>
```

L'erreur « NameError: name 'hello' isn't defined » indique que les données nommées 'hello' ne sont pas définies. La raison de cette erreur sera expliquée au chapitre 3 dans lequel seront abordées les variables. Lorsque vous manipulez des chaînes en Python, il suffit d'utiliser des guillemets simples '' ou des guillemets doubles " " pour que l'erreur ne se produise plus.

```
>>> 'hello'
'hello'
>>>
```

En Python, aucune différence n'est faite entre une chaîne encadrée de guillemets simples ou doubles.

■ Afficher une chaîne de caractères dans le terminal

En Python, la commande print() permet d'afficher une chaîne de caractères dans un terminal.

```
print(chaîne de caractères)
```

Exécutez le code suivant pour voir comment cela fonctionne :

```
>>> print('hello')
hello
>>>
```

Notez qu'il n'y a pas de guillemets dans le résultat, contrairement au précédent programme, parce que print() affiche le contenu qui se trouve entre ses guillemets.

■ Gérer la présence de caractères spéciaux dans des chaînes de caractères

Il convient de prendre des précautions lorsque le texte à l'intérieur d'une chaîne de caractères contient des guillemets. Par exemple, si vous tentez de créer la chaîne 'hello ' python'', une erreur se produira :

```
>>> print('hello 'python")
Traceback (most recent call last):
  File "<stdin>", line 1
SyntaxError: invalid syntax
>>>
```

Cette erreur est due au fait que le guillemet ['] situé devant « python » est considéré comme un guillemet simple qui ferme le guillemet ['] placé devant « hello ». Pour éviter une erreur, ce guillemet simple doit être traité de manière particulière grâce à l'une des solutions suivantes.

- Solution n°1

Si vous souhaitez créer une chaîne de caractères contenant des guillemets simples, placez l'ensemble de la chaîne entre guillemets doubles. Inversement, si vous souhaitez afficher des guillemets doubles dans une chaîne, placez l'ensemble de la chaîne entre guillemets simples.

```
>>> print("hello 'python'")
hello 'python'
>>> print('hello "python"')
hello "python"
>>>
```

- Solution n°2

En mettant un antislash « \ » devant un guillemet, le guillemet sera traité comme des guillemets de citation. Dans ces cas-là, l'antislash est appelé un caractère d'échappement. Il sert à indiquer que le guillemet qui suit ne doit pas être interprété comme un délimiteur de chaîne (pour marquer le début ou la fin d'une chaîne).

```
>>> print('hello \'python\'')
hello 'python'
>>>
```

L'antislash « \ » peut également être utilisé pour afficher un saut de ligne entre deux parties d'une chaîne de caractères en le faisant suivre de la lettre « n », comme suit : « \n ». Le caractère « n » est alors interprété comme un saut de ligne. Il s'agit là aussi d'une séquence d'échappement. Cette fois-ci, le caractère d'échappement « \ » associe un comportement spécial au caractère « n » (ici le saut de ligne).

```
>>> print('hello \npython')
hello
python
>>>
```

■ Concaténer des chaînes

Vous pouvez concaténer, c'est-à-dire assembler, deux chaînes avec l'opérateur « + » utilisé précédemment pour additionner des nombres. Dans l'exemple ci-dessous, la chaîne « abcdef » est créée en concaténant les chaînes « abc » et « def ».

```
>>> 'abc'+'def'
'abcdef'
>>>
```

■ Répéter une chaîne

Quand l'opérateur « * » est utilisé avec un nombre N et une chaîne S, il crée une nouvelle chaîne de caractères dont le contenu est celui de S répété N fois. Dans l'exemple ci-dessous, la chaîne « abc » est multipliée par « 3 » :

```
>>> 'abc'*3
'abcabcabc'
>>>
```

6.2. Convertir une valeur numérique en chaîne de caractères et inversement

La valeur numérique « 123 » et la chaîne de caractères « '123' » sont traitées comme des données complètement différentes. Les valeurs numériques peuvent être utilisées pour le calcul en tant que données numériques, mais il peut arriver de vouloir les traiter en chaînes de caractères. Dans ce cas, il faut convertir ces données numériques en chaînes de caractère.

■ Convertir des nombres en chaînes

Utilisez la commande len() pour obtenir le nombre de caractères d'une chaîne.

```
len(chaîne de données)
```

* « len » est une abréviation de « length » en anglais qui signifie « longueur ».

Essayez, par exemple, d'obtenir le nombre de caractères contenus dans la chaîne « python » en utilisant la commande len().

```
>>> len('python')
6
>>>
```

Avec cette même commande, vous pouvez obtenir le nombre de chiffres qui composent des données numériques. La commande len() ne prend, cependant, pas en charge les données numériques. Une erreur se produira si elle est spécifiée telle quelle :

```
>>> len(123456)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: object of type 'int' has no len()
>>>
```

Il faut donc commencer par convertir les données numériques en données de chaîne de caractères. Pour réaliser cette conversion, utilisez la commande str().
* « str » est une abréviation pour « string » qui signifie « chaîne » en anglais.

```
Convertir les données spécifiées en chaîne de caractères...
str(données que vous voulez convertir en chaîne)
```

Exécutez le code suivant :

```
>>> len(str(123456))
6
>>>
```

■ Convertir une chaîne de caractères en nombre

Pour repasser d'une chaîne de caractères à un nombre, utilisez les commandes int() ou float() que vous connaissez déjà.

Dans l'exemple ci-dessous, les chaînes "123" et "456" sont converties en nombres avec la commande int() et additionnées pour obtenir le résultat "579".

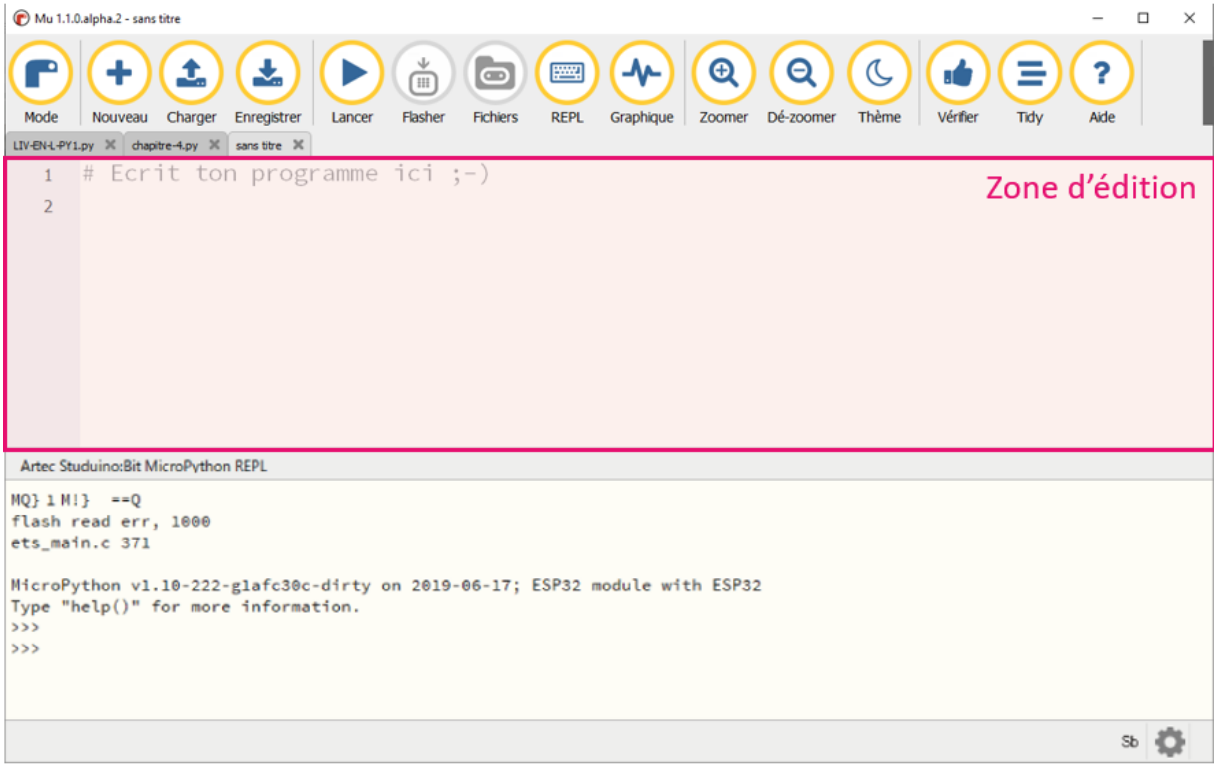
```
>>> int('123')+int('456')
579
>>>
```

7. Exercice : Afficher son nom sur le panneau LED de l'ESPeRobo

Dans cet exercice, nous allons utiliser les données de chaîne pour faire défiler votre nom sur le panneau LED.

7.1. Créer un programme

Jusqu'à présent, vous n'avez écrit et exécuté qu'une seule ligne de code à la fois dans le terminal. Or, dans cet exercice, vous devrez écrire plusieurs lignes de code pour contrôler la carte ESPeRobo. À partir de maintenant, nous utiliserons donc la zone d'édition. Cliquez dans la zone d'édition mise en évidence dans l'image ci-dessous pour écrire vos lignes de code.



Recopiez les deux lignes de code ci-dessous dans la zone d'édition. Ces deux lignes de code permettent de faire défiler du texte sur le panneau LED de la carte ESPeRobo.
* Une explication détaillée de ce code sera faite petit à petit dans les leçons suivantes.

```
1  from pystubit.board import display
2  display.scroll('Ecrivez votre nom ici.')
```

Dans la 1ère ligne de code, une instruction importe l'objet « display » qui sert à contrôler le panneau LED :

```
from pystubit.board import display
```

La 2e ligne de code exécute la commande scroll() qui sert à afficher et faire défiler la chaîne de caractères spécifiée entre parenthèses :

```
display.scroll('Ecrivez votre nom ici.')
```

Remplacez le texte inscrit entre guillemets 'Ecrivez votre nom ici' par votre nom.

Si vous souhaitez afficher des nombres avec la commande scroll(), spécifiez-les sous forme de chaînes plutôt que de chiffres. Pour cela, utilisez l'une des méthodes suivantes :

- Placez les chiffres entre guillemets pour en faire une chaîne :

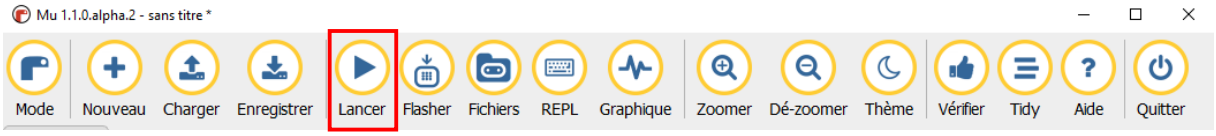
```
1  from pystubit.board import display
2  display.scroll('123')
```

- Convertissez les chiffres en chaîne avec la commande str() :

```
1  from pystubit.board import display
2  display.scroll(str(123))
```

7.2. Exécuter le programme

Pour exécuter le programme, cliquez sur le bouton « Lancer » en haut de l'écran.

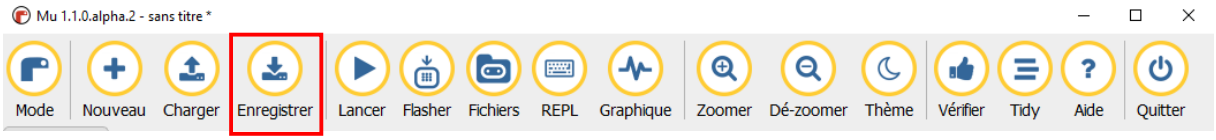


Lorsque l'exécution est lancée, une fenêtre de terminal s'ouvre et affiche le message suivant. Votre nom défilera ensuite sur le panneau LED.

```
MicroPython v1.10-91-g66f0afc-dirty on 2019-02-27; ESP32 module with ESP32
Type "help()" for more information.
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
raw REPL; CTRL-B to exit
>OK
```

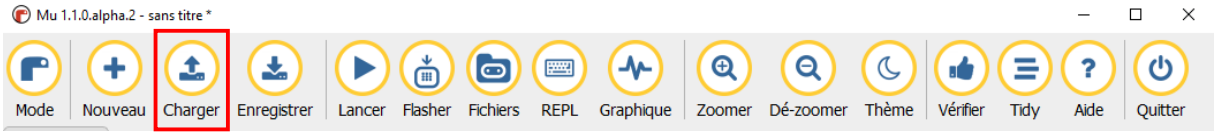
7.3. Sauvegarder le programme

Pour sauvegarder le programme, cliquez sur le bouton « Enregistrer » en haut de l'écran. Vous pouvez spécifier le dossier de destination et le nom du fichier.



Les fichiers de programme Python ont une extension « .py ». L'ordinateur se sert de cette extension pour identifier le type de fichier. Les fichiers portant l'extension « .py » sont donc reconnus comme des fichiers de programme Python.

Le fichier de programme enregistré peut être ouvert et modifié à l'aide du bouton « Charger » situé en haut de l'écran.



Résumé du chapitre

Dans ce chapitre, vous avez appris à :

- repérer les éléments composant la carte ESPeRobo.
- utiliser le logiciel Mu pour créer et exécuter des programmes en Python.
- manipuler des données numériques.
- traiter des chaînes de caractères.

Python peut gérer d'autres types de données, telles que les « listes », les « tuples » et les « dictionnaires ». Nous expliquerons en détail comment traiter ces données au chapitre 6.

Notes

Apprendre à programmer des robots pour comprendre le monde d'aujourd'hui et de demain.

Les machines programmées, de plus en plus intelligentes, font partie intégrante de notre vie de tous les jours. Elles nous accompagnent, nous entourent et ont envahi tous les domaines de notre vie quotidienne. Maîtriser le monde, ce n'est pas les utiliser, mais avant tout comprendre comment elles fonctionnent.

Comment fonctionnent-elles ?
Selon quelle logique ? Selon quels algorithmes ?
Comment sont conçus les programmes qui leur dictent leurs actions et réactions ?

C'est ce que vous apprendrez tout au long de ces livrets d'apprentissage. Et pas seulement "en théorie" : vous allez vous-même concevoir et programmer vos propres robots : des actions simples aux plus complexes, vous apprendrez à programmer des robots amusants et originaux que vous aurez conçus vous-même. Une seule limite : votre créativité !

L'École Robots permet à tous de s'initier à la programmation en s'amusant, un enjeu majeur, aujourd'hui et demain.

Ecole Robots
Apprendre à coder avec les robots éducatifs
CURSUS EDUCATION NATIONALE

Pour en savoir plus : www.ecolerobots.com

Réf : LIV-EN-LY-PY-1 (v.1)