APPRENDRE PYTHON EN 5 HEURES

LE GUIDE DES VRAIS DEBUTANTS

PAR
Hicham YOUSSEF

APPRENDRE PYTHON EN 5 HEURES

LE GUIDE DES VRAIS DEBUTANTS

(PROF, PARENT OU AUTODIDACTE

HICHAM YOUSSEF

TABLE DES MATIERES

A PROPOS DE L'AUTEUR

Je m'appelle Hicham, J'ai longtemps travaillé en tant qu'ingénieur dans le secteur de l'aéronautique. Mon travail consistait principalement à faire de la modélisation et simulation numérique, par exemple, dimensionner des pièces d'avion, prévoir le comportement mécanique / thermique / acoustique de structures avionique, prévoir la durée de vie des pièces … et plusieurs autres activités.

Maintenant j'ai tourné cette page, je suis enseignant de mathématiques et blogueur dans des domaines qui me passionnent, comme la cuisine, les maths et la programmation python.

Le but à travers la réalisation du site https://cours-maths-python.com est d'offrir aux enseignants, parents et élèves, des ressources pratiques en mathématiques, programmation python et activités scratch.

A LIRE ATTENTIVEMENT

INTRODUCTION

Apprendre à programmer peut sembler complexe, surtout quand on n'a jamais touché à une ligne de code. Pourtant, avec les bons outils et une méthode claire, Python devient un langage accessible, logique, et même… amusant !

Ce guide s'adresse à vous si :

- Vous êtes **enseignant(e)** et vous souhaitez intégrer un peu de code dans vos cours de mathématiques ou de technologie.
- Vous êtes **parent** et vous aimeriez aider votre enfant à s'initier à la programmation.
- Vous êtes **autodidacte** et vous voulez enfin comprendre ce que fait ce langage qu'on retrouve partout : dans les sciences, les jeux vidéo, l'intelligence artificielle, et même l'éducation.

L'objectif est simple : vous faire découvrir les bases du langage Python en 5 sessions d'environ 1h, à travers des activités concrètes, sans jargon inutile.

Aucune connaissance technique n'est requise. Vous n'avez même pas besoin d'installer quoi que ce soit si vous utilisez un éditeur en ligne.

À la fin de ce parcours, vous saurez :

- écrire des programmes interactifs,
- automatiser des petits calculs,
- créer un mini-jeu éducatif,
- sauvegarder des résultats dans un fichier,
- et pourquoi pas, accompagner d'autres personnes dans leurs premiers pas avec Python.

🛠️ MATERIEL NECESSAIRE

Bonne nouvelle : pour suivre ce guide, **pas besoin d'un ordinateur dernier cri** ni d'être un expert en informatique. Voici ce qu'il vous faut :

- **Un ordinateur** (Windows, Mac ou Linux) ou une tablette avec clavier ;
- **Une connexion internet** pour accéder aux plateformes en ligne ;
- **Un navigateur web récent** (Chrome, Firefox, Safari…) ;
- **Un compte gratuit** sur replit.com ou programiz.com si vous ne souhaitez rien installer.

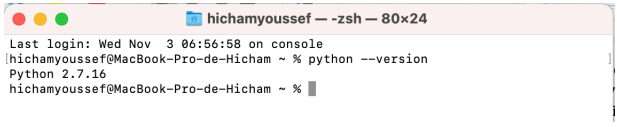
💡 Si vous préférez coder en local, vous pouvez aussi installer Python depuis python.org et utiliser un éditeur simple comme **Visual Studio Code**.

Tout est fait pour que vous puissiez commencer à programmer en quelques minutes, sans frustration.

1ERE SEANCE. INSTALLATION ET PREMIER PROGRAMME

MACOS. LINUX

Python est un langage de programmation multiplateforme. Si le système d'exploitation utilisé est Linux ou MacOS, il y a de forte chance que python y soit déjà installé. Et pour le savoir c'est très simple : ouvrez un terminal de commande et tapez python --version.



On voit afficher sur le terminal la version de python utilisée par défaut. Dans le cas de l'exemple ci-dessus la version par défaut est python 2.7.16.

Il est important de comprendre et savoir les différentes versions qui existent. Et il est très important de bien considérer ses besoins avant d'utiliser une version autre que la version de production actuelle. Aujourd'hui, la version à utiliser est python 3.10. Pour l'avoir il faut se rendre dans le site officiel de python.org et la télécharger puis de l'installer.

Windows

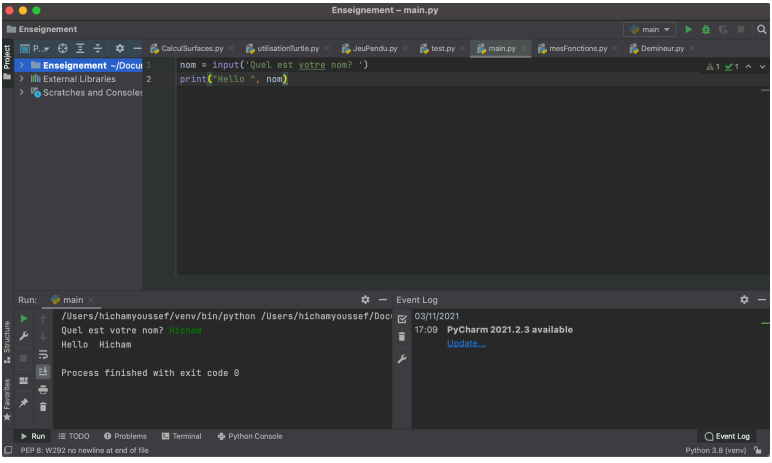
En revanche, si vous utilisez **Windows**, il va vous falloir **installer python**. En effet, Il suffit d'aller sur le site python.org et télécharger la **version de python** qui vous convient, double cliquer pour commencer **l'installation** et suivez les étapes jusqu'à la fin de l'installation.

Notez que Python.3.10 n'est pas compatible avec Window7 ou une autre version antérieure.

Installer un IDE

Un environnement de développement intégré IDE (**Integrated Development Environment**) est tout simplement une application qui permet de **coder facilement**. Voici quelques caractéristiques de base qu'un bon IDE devrait avoir:

- Une **interface graphique**,
- Un système capable de détecter automatiquement les **mots-clés** et de les colorer
- Un **débogueur** pour **corriger le programme** si nécessaire
- Et bien sûr, cela permet de sauvegarder et d'ouvrir le script exactement comme quand on l'a laissé.



Il existe plusieurs IDE qui permettent de **programmer avec python**, personnellement, j'ai utilisé pendant très longtemps **python x, y**. Cette distribution est gratuite et permet de **développer facilement** des **calculs scientifiques** et **techniques**, l'analyse de données et la visualisation de données basées sur le langage de programmation Python. Mais malheureusement, cet IDE ne prend pas en compte la dernière version de python.

Pour utiliser la dernière version de python, je recommande (classé par ordre de préférence) :

1. Visual studio code
2. Pycharm
3. ou Edupython.

Visual Studio Code

PyCharm
The Python IDE
for Professional Developers
DOWNLOAD

EDUPYTHON

Pycharm et **visual studio code** permettent de faire des développements dans un **environnement professionnel**. Et Edupython convient plus aux milieux éducatifs.

Pour la préparation de ce guide, j'ai utilisé l'IDE Pycharm.

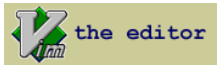
Je ne reviendrai pas sur l'installation et l'utilisation de Pycharm ou Edupython. On peut trouver très facilement sur intenet des tutoriels qui montrent leurs utilisations.

PLATEFORMES DE DEVELOPPEMENT EN LIGNE

- **Avantage** : pas besoin d'installer ni python ni aucun IDE. Tout se passe enligne
- **Inconvénient** : on ne peut pas utiliser certaines bibliothèque de python
- Solution suffisante pour apprendre quelques bases de la programmation mais très limitée pour le travail à long terme.

- Quelques plateformes :
    - https://trinket.io
    - https://replit.com/languages/python3
    - https://www.programiz.com/python-programming/online-compiler/
    - https://www.online-python.com

ÉCRIRE UN SCRIPT PYTHON

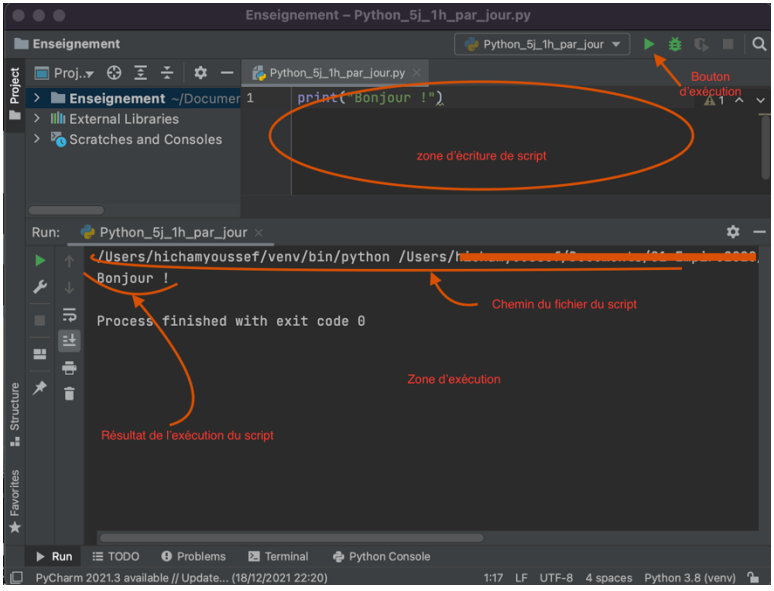


On peut utiliser un simple éditeur de texte pour écrire un script python puis de l'exécuter dans un invité de commande

- Notepad++ https://notepad-plus-plus.org
- Vim,

LE PREMIER PROGRAMME

On va commencer par un exemple très simple pour comprendre la programmation python ainsi que sa syntaxe. Mais avant de commencer, voici une petite présentation de l'interface de Pycharm dans laquelle nous allons écrire puis exécuter les scripts python. (Les autres IDE se présentent de la même façon quasiment.)



On va commencer par une instruction toute simple, on va demander d'afficher une consigne.

L'instruction qui permet de faire cela est la fonction print().

Code python :

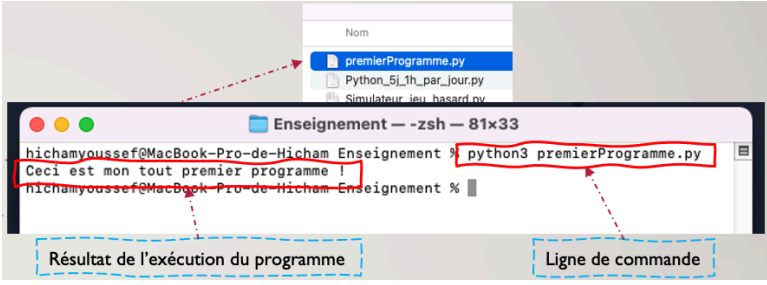
print(‘‘Ceci est mon tout premier programme :)’’)

TRAVAUX PRATIQUES

1ère partie

Si vous n’avez rien installé sur votre terminal ou si vous préférez travailler avec un smartphone ou une tablette allez directement à la 2ème partie

- Ouvrez un éditeur de texte et écrivez la ligne suivante:
- print("Ceci est mon tout premier programme :)")
- Enregistrez le fichier avec l’extension .py (exemple: premierProgramme.py)
- Ouvrez u terminal ou un invité de commande à l’endroit où vous avez enregistrer le fichier
- Taper la commande suivante: **Python3 premierProgramme.py**

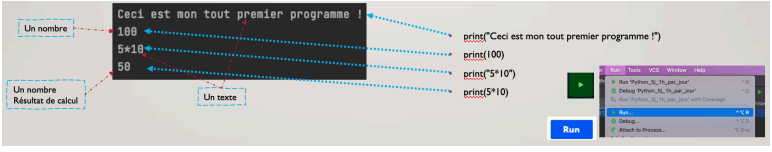


2ème partie

- Ouvrez un IDE ou allez sur une des plateformes de développement en ligne ;
- Créez un nouveau fichier de script python et donnez-lui un nom avec l'extension .py
- Dans la zone d'écriture de script, tapez le programme suivant :

```
print("Ceci est mon tout premier programme !")
print(100)
print("5*10")
print(5*10)
```

- Exécutez avec le bouton RUN ;
- Dans la zone d'exécution, admirez le résultat de votre premier code python.



Print() : Fonction qui permet d'afficher des données

Les données peuvent être de **type texte, nombre ou autres types.**

LES TYPES DE DONNEES

Les données texte sont de type string str()

- Ne peuvent pas subir des opérations mathématiques.
- On utilise soit les doubles quottes ‘’ ‘’ ou des simples quottes ‘ ‘ pour définir une donnée de type string.
- Les données nombres :

- Entier: type integer (int)
- Fractionaire : type float

Possible de transformer les données texte en données nombre en utilisant les fonctions int() ou float()

D'autres types de données seront détaillés plus tard.

3EME PARTIE
Tapez le programme suivant, puis exécutez le.

```
nom = input("quel est votre nom ?")
age = input("quel est votre age ?")
print("vous vous appelez ", nom)
print("et vous avez ", age, " ans !")
```

résultat de l'exécution :

```
quel est votre nom ? Tom
quel est votre age ? 23
vous vous appelez  Tom
et vous avez  23  ans !
```

- Les données : Tom et 23 sont entrées au clavier.
- Les paramètres nom et age sont des variables.
- Le signe « = » permet d'affecter une valeur à une variable
- On peut utiliser le contenu d'une variable pour l'afficher ou pour effectuer des calculs avec
- On peut afficher plusieurs données avec la fonction print(). Il suffit de les espacer par des virgules.
- La fonction input() permet d'interagir avec l'utilisateur.

- Les données obtenues par la fonction input() sont toujours de type string. Il est possible de les transformer en integer ou float en utilisant les fonctions int() ou float()

4EME PARTIE. EXERCICE

Compléter le programme pour obtenir le résultat suivant :

```
entrer la valeur de a : 10
entrer la valeur de a : 15
S =  150
a+b =  25
a-b =  -5
a/b =  0.6666666666666666
a puissance b =  1000000000000000
```

… = int(……..("entrer la valeur de a : "))

b = …… (……..(…entrer la valeur de b : …))

S = a*b

print(" …… ", S)

……("a+b = ", ……)

…… …… …… …… …… …… …… ……

…… …… …… …… …… …… …… ……

Notes

2EME SEANCE. VARIABLES, LISTES ET TESTS LOGIQUES

LES VARIABLES

Une variable doit donc avoir un nom, celui-ci est choisi librement par le codeur à condition que ce nom ne soit pas déjà réservé par une instruction du langage.

CHOIX DU NOM DE VARIABLE

- Le nom de la variable doit obligatoirement commencer par une lettre ;
- La casse compte, une lettre écrite en majuscule n'est pas équivalente à la même lettre écrite en minuscule ;
- Il est possible d'utiliser des lettres accentuées comme (é, à, …). Mais je ne recommande pas cette pratique.
- On ne peut utiliser des caractères spéciaux comme @, !, …
- On peut utiliser un mélange de chiffres et de lettres. On peut utiliser le caractère spécial Under score(_).
- On ne peut pas utiliser un mot réservé par une instruction du langage. Cependant, il y a une convention qui régule le choix des noms de variables et qui est plus ou moins respectée par les programmeurs :
    - le nom de la variable doit donner des indications sur la variable
    - le nom de la variable doit commencer par une lettre minuscule, on utilise des majuscules à l'intérieur du nom pour le rendre plus lisible (surfaceRectangle par exemple)

AFFECTATION

L'affectation est le fait de donner une valeur à la variable.

Le signe = est le moyen d'affectation dans le langage python.

EXERCICE 1

Qu'affiche le code suivant :

```
a = 15
b = 3
c = a + b
a = b
b = c
b = b – a
print(b)
```

Quelle affectation doit-on donner à la variable a pour que le programme affiche 11 ?

EXERCICE 2

Tester le code suivant :

```
largeur, longueur = 10, 15.5
surface = largeur * longueur
print(type(largeur), type(longueur), type(surface))
```

La fonction print() est une fonction d'affichage.

Quel est le rôle de la fonction type() ?

Solution Exercice 1

```
a = 15          # créer la variable a et lui affecter la valeur 15
b = 3           # créer la variable b et lui affecter la valeur 3
c = a + b       # créer la variable c et lui affecter la somme de a + b
a = b           # affecter à la variable a la valeur de b
b = c           # affecter à la variable b la valeur de c
b = b – a       # affecter à la variable b la valeur de b – a
print(b)        # afficher la valeur de b
```

la dernière valeur que b prend est a+b-b = a. donc pour que le programme affiche 11, il faudra affecter à b la valeur 11.

Solution Exercice 2

La fonction type() permet de donner le type des arguments qu'elle prend. (variables ou valeurs mis entre parenthèses). Dans le cas de l'exercice elle affichera :

<class 'int'> <class 'float'> <class 'float'>

Operateurs de calculs mathematiques

- + : L'addition
- - : La soustraction
- * : La multiplication
- / : La division réelle (résultat est du type float)
- // : La division entière tronquée
- ** : L'exponentiation (puissance)
- % : Le modulo (reste de la division)

```
>>> 25/9
2.7777777777777777
>>> 25//9
2
>>> 25%9
7
>>> 3*2
6
>>> 3**2
9
```

LES VARIABLES DE TYPE LISTE

- Ces variables sont très utiles lorsque l'on veut classer des données de manière ordonnée.
- Schématiquement, on peut considérer une liste python comme un armoire dans lequel on trouve des tiroirs, et dans chaque tiroir on range une donnée. Chaque tiroir de l'armoire est indexé par un numéro pour pouvoir l'identifier facilement.

- Ces numéros, que l'on appellera indices, sont les clés qui permettent d'ouvrir les tiroirs pour obtenir leurs contenus.

COMMENT CREER UNE VARIABLE DE TYPE LISTE

Pour créer une variable de type liste python, il suffit de donner un nom à cette variable suivie de (=), puis mettre entre deux crochets […, …, …] les éléments de la liste, espacés de virgules (,).

EXEMPLE:

- **note_Maths = [13.5, 18.5, 17, 17.5, 15.5]**
- Le nom de la variable : note_Maths.
- elle contient 5 éléments.
- Il ne faut pas confondre le point qui est utilisé comme séparateur décimal et la virgule qui sépare les éléments de la liste python.
- Les éléments de la liste python sont indexés de 0 à 4.
- Pour obtenir le premier élément de la liste on utilise l'indice 0 de la manière suivante :

- noteMaths[0] = 13.5
- Les indices négatifs parcourent la liste du dernier au premier éléments

PROPRIETES DES LISTES

- **La fonction len()** donnela **longueur d'une liste** (nombre d'éléments de la liste)
- Les **indices d'une liste** commencent par 0, pour avoir le **premier élément** on utilisera **[0]**, le deuxième [1], le troisième [2] et ainsi de suite.

```
>>> note_Maths = [13.5, 18.5, 17, 17.5, 15.5]
>>> print(note_Maths[0])
13.5
>>> print(note_Maths[-1])
15.5
>>> print(note_Maths[1])
18.5
```

- Les éléments d'une même liste peuvent être de types différents

```
>>> liste_1 = [25, 1.5, "texte"]
>>> type(liste_1)
<class 'list'>
>>> type(liste_1[0])
<class 'int'>
>>> type(liste_1[1])
<class 'float'>
>>> type(liste_1[2])
<class 'str'>
```

- il est possible de modifier les éléments de la liste séparément

```
>>> liste_1[2]="99"
>>> liste_1
[25, 1.5, '99']
```

QUELQUES OPERATIONS SUR LES LISTES PYTHON

Il existe une grande variété d'opérations qu'on peut appliquer aux liste, et c'est en codant des application de plus en plus compliquées qu'on est amené à les utiliser, voici à mon sens les trois fonctions les plus utilisées pour modifier les listes

LA FONCTION INSERT()

Imaginons que l'on souhaite ajouter un élément à une liste, et en plus on souhaite choisir l'emplacement de cet élément dans la liste. Ceci est rendu possible grâce à la fonction insert(). Cette fonction prend 2 arguments, le premier correspond à l'emplacement où l'on souhaite placer l'élément. Et le second argument correspond à l'élément lui-même qu'on veut ajouter à la liste.

```
>>>
>>> a = [2., "3", 5]
>>> a.insert(1,15.4)
>>> a
[2.0, 15.4, '3', 5]
>>>
>>>
```

LA FONCTION APPEND()

Cette fonction permet d'insérer un élément à la fin d'une liste. Elle augmentera la longueur de la liste de 1.

Et pour accéder au dernier élément ajouté, on peut utiliser l'indice [-1]. La fonction append() ne prend qu'un seul

argument. On ne peut donc ajouter qu'un seul élément à la fois.

L'opérateur += permet de fusionner 2 listes, il peut donc faire exactement le même travail que la fonction append(). Il est aussi utilisé pour ajouter plusieurs éléments à la fois, chose qui n'est pas possible de faire avec la fonction append.()

```
>>> a += [1]
>>> a
[2.0, '3', 5, 1]
>>> a += [12, 11, 17]
>>> a
[2.0, '3', 5, 1, 12, 11, 17]
>>>
```

La fonction pop()

Cette dernière fonction que je présente ici est la fonction pop(), celle-ci permet tout simplement de supprimer un élément de la liste. La fonction pop() peut prendre au maximum 1 seul argument, et cet argument doit correspondre à l'indice de l'élément que l'on souhaite supprimer de la liste.

```
>>> a = [2., "3", 5]
>>> a.pop(1)
'3'
>>> a
[2.0, 5]
>>> a.pop()
5
>>> a
[2.0]
>>>
```

On peut faire beaucoup de choses sur les listes, notamment les calculs statistiques comme la moyenne, le maximum de la

liste et minimum… nous verrons tout cela avec des applications concrètes le moment venu. Vous pouvez consulter le site python.org pour plus d'aide sur les listes python.

EXERCICE 3

i. **Répondez aux questions sans coder le script:**

```
a = [2., "3", 5]
a.append(15.4)
print(a)
```

Résultat qui sera affiché: ----------

```
a += [22, 1.2]
print(a)
```

Résultat qui sera affiché: ----------

```
a.insert(3,21)
print(a)
```

Résultat qui sera affiché: ----------

```
a.pop(-3)
print(a)
```

Résultat qui sera affiché: ----------

```
a.pop()
print(a)
```

Résultat qui sera affiché: ----------

```
a[1]=int(a[1])
print(a)
```

Résultat qui sera affiché: ----------

ii. **Vérifiez vos réponses en recopiant le script dans un fichier python**

OPERATION SUR LES LISTES

- list_1 + list_2 : Renvoie une liste contenant les éléments de list_1 suivis des éléments de list_2.
- list_1 * n : Renvoie une liste contenant les éléments de list_1 répétés n fois
- sorted(list_1) : retourne ne liste ordonnée contenant les éléments de la liste list_1
    - **Création d'une nouvelle liste**
- list_1.sort() : Ordonne la liste_1 par ordre croissant des valeurs
    - **Pas de création d'une nouvelle liste**
- list_1.count(x) : Renvoie le nombre d'occurrence de x dans la liste list _1
- list_1.upper() : met en majuscule les lettres de list_1
- list_1.lower() : met en minuscule les lettres de list_1
- list_1.find(x) : Renvoie l'indice de la première occurrence de x dans la liste list _1. si x n'est pas dans la liste list_1, il renvoie -1.
- list_1.index(x) : Renvoie l'indice de la première occurrence de l'élément x dans la liste list _1
    - Attention : un message d'erreur sera envoyé par la console python si l'élément x n'est pas dans la liste.
- list_1.remove(x) : Supprime la première occurrence de l'élément x dans la liste list _1
    - Attention : un message d'erreur sera envoyé par la console python si on essai de retirer un élément qui n'est pas dans la liste.

- Il sera pertinent de tester si l'élément existe avant de le supprimer de la liste. (voire partie Tests logiques)

TESTS LOGIQUES

Un test logique renvoie un 1 ou True si le test est vrai. Et un 0 ou False si le test est faux.

```
>>> "ceci" == "cela"
False
>>> "ceci" == "cEci"
False
>>> "ceci" == "ceci"
True
```

Exemple: Reproduisez puis exécutez le script suivant: (puis analysez les résultats retournés)

```
jours=["Lundi","Mardi","Mercredi","Jeudi","Vendredi"]
Test1 = "Samedi" in jours
print(Test1)
Test2 = "Lundi" in jours
print(Test2)
Test3 = "mardi" in jours
print(Test3)
Test4 = "Mardi" in jours
print(Test3)
```

```
>>> "mardi" in jours
False
>>> "Mardi" in jours
True
>>> 20 == 4*5
True
```

EXERCICE 4

i. **Répondez aux questions sans coder le script suivant :**

```
a = [2., "3", 5]
list_1 = ["Téléviseur", "Téléphone", "Tablette", "Ordinateur",
"Switch", "PS5"]
list_2 = [1, 40, 75.5]
list_3 = list_1 + list_2
print(list_3)
```

Résultat qui sera affiché: ----------

```
list_4 = list_2 * 3
print(list_4)
```

Résultat qui sera affiché: ----------

```
test1 = list_2 == list_4
print(test1)
```

Résultat qui sera affiché: ----------

```
test2 = list_2[0] == list_4[0]
print(test2)
```

Résultat qui sera affiché: ----------

```
list_5 = sorted(list_4)
test3 = list_5 == list_4.sort()
print(test3)
```

Résultat qui sera affiché: ----------

```
test4 = "Tél" in list_1[0] and "Tél" in list_1[1]
print(test4)
```

Résultat qui sera affiché: ----------

```
list_1.remove("Téléviseur")
list_1.append("Livre")
test5 = "Tél" in list_1[0] and "Tél" in list_1[1]
print(test5)
```

Résultat qui sera affiché: ----------

```
test6 = "Tél" in list_1[0] or "Tél" in list_1[1]
print(test6)
```

Résultat qui sera affiché: ----------

ii. **Vérifiez vos réponses en recopiant le script dans un fichier python, puis en l'exécutant**

Notes

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Notes

3EME SEANCE. CHAINES DE CARACTERES, TESTS LOGIQUES ET BOUCLES

LES CHAINES DE CARACTERES

Les propriétés des listes s'appliquent également aux chaines de caractères. Les chaînes de caractères peuvent également être indexées, ce qui signifie qu'on peut accéder aux caractères par leurs positions.

Le premier caractère d'une chaîne possède l'indice 0, le deuxième 1 etc…

```
>>> chaine = "j'aime le chocolat !"
>>> chaine[2]
'a'
>>> chaine[-1]
'!'
>>> chaine.find("le")
7
>>> chaine.replace("le", "ce")
"j'aime ce chocolat !"
>>> chaine
"j'aime le chocolat !"
```

STRUCTURES CONDITIONNELLES. (IF, ELIF, ELSE)

Elles permettent d'exécuter un bloc de code si une certaine condition est vérifiée. Il y'a trois schéma d'utilisation possibles :

- La condition if ("si")
- La condition if…else ("si…sinon")
- La condition if…elif…else ("si…sinon si… sinon").

TRAVAUX PRATIQUES.

Reproduisez puis exécutez le programme suivant :

```
nom = input("quel est votre nom ? ")
age = int(input("quel est votre age ? "))
if age <= 16:
   print("Bonjour ", nom, " , vous pouvez entrer c'est gratuit
!")
elif age <= 25:
   print("Bonjour ", nom, " , vous devez payer 10 euros !")
else:
   print("Bonjour ", nom, " , vous devez payer 15 euros !")
```

Testez le programme en répondant 12, puis 23 et enfin 45 à la deuxième question.

Notez à chaque fois la réponse affichée.

---------- ---------- ----------

Qu'affichera le programme si l'on répondra 75 à la deuxième question ? Répondez sans utiliser le programme.

---------- ---------- -------

Adaptez le programme pour que les personnes âgées de plus de 65 ans n'auront rien à payer

LA CONDITION IF ("SI")

if suivi d'un test logique, suivi de 2 points « : ». Si le résultat du test est True, alors le bloc indenté (décalé vers la droite par espace ou tabulation) sera exécuté.

LA CONDITION IF…ELSE ("SI…SINON")

Comme la structure de if toute seule, sauf que si le résultat du test est False alors le bloc indenté en dessous de else qui sera exécuté

La condition if…elif…else ("si…sinon si… sinon").

Plusieurs tests logiques se succèdent, et seulement un seul bloc qui sera exécuté, celui end essous du test dont le résultat est True. Et si aucun test logique n'est vrai alors c'est le bloc en dessous du else qui sera exécuté.

```
if test1 :
    print("test1 est vrai")
elif test2:
    print("test2 est vrai")
elif test3:
    print("test3 est vrai")
else:
    print("aucun test n'est vrai")
```

Exercice

Écrire un script qui permet de résoudre la problématique suivante :

Le patron d'un restaurant souhaite installer une borne interactive à l'entrée. La borne est équipée d'un programme qui filtre les entrées des clients en fonction des paramètres suivants :

- Seuls les clients qui ont réservé et qui sont vaccinés ou munis d'un test négatif datant de moins de 24 heures qui peuvent entrer dans le restaurant.
- Les clients âgés de moins de 12 ans n'ont pas besoin de vaccin ni de test.

- Le programme doit interagir avec le client, c'est-à-dire il pose des questions au client, et en fonction des réponses il lui affiche à la fin :
    - « merci vous pouvez entrer »,
    - ou « désolé, vous ne pouvez pas entrer »
- Toutes les éventualités doivent être traitées.

LES OPERATEURS DE TESTS LOGIQUES

Opérateur	Définition
==	Tester l'égalité (valeur et type)
!=	Tester la différence en valeur ou en type
<	Tester si une valeur est strictement inférieure à une autre
>	Tester si une valeur est strictement supérieure à une autre
<=	Tester si une valeur est inférieure ou égale à une autre
>=	Tester si une valeur est supérieure ou égale à une autre

LES BOUCLES

Les boucles sont très utiles dans la programmation. Elles évitent aux **programmeurs** de répéter un bloc d'instructions, inutilement, un certain nombre de fois.

Il existe 2 types de boucles :

- Des boucles avec un nombre indéfini de répétition (boucle while) tant qu'un test logique est vrai, on répète l'exécution d'un bloc de code.
- Des boucles avec un nombre défini de répétition. (boucle for). On répète l'exécution d'un bloc de code un certain nombre de fois prédéfini.

STRUCTURE DE BOUCLE CONDITIONNELLE (WHILE)

C'est un bloc d'instructions qui sera exécuté autant de fois que le test est vrai

Voici un code qui montre comment utiliser la boucle conditionnelle while:

```
x = 1
while x <= 3:
   print("x =", x)
   x = x+1
print("fin de la boucle. x = ", x)
```

Explication du code en langage naturel :

- Initialiser x à 1
- Répéter tant que x est inférieure ou égale à 3
- Afficher x = valeur de x
- Incrémenter x de 1
- Afficher: fin de la boucle. x = valeur de x

Résultat :

```
x = 1
x = 2
x = 3
fin de la boucle. x =  4
```

STRUCTURE DE LA BOUCLE FOR

C'est un bloc d'instructions qui sera exécuté un nombre de fois prédéfini

Voici un code qui fait la même chose que celui du paragraphe précédent.

```
for x in range(4):
   print("x =", x)

print("fin de la boucle. x = ", x)
```

Explication du code en langage naturel :

- Répéter pour chaque x égale à la valeur de la liste des nombres de 0 à 3
- Afficher x = valeur de x
- Afficher : fin de la boucle. x = valeur de x

Résultat :

```
x = 0
x = 1
x = 2
x = 3
fin de la boucle. x =  3
```

LA FONCTION RANGE()

La fonction range() permet de créer une liste de valeurs espacées par un intervalle régulier. Elle peut prendre un deux ou trois arguments.

- Le premier argument correspond à la valeur initiale de la liste ;
- le deuxième correspond à la valeur finale de la liste ;

- et le dernier correspond au pas (l'intervalle entre deux valeurs consécutives).

Ainsi pour avoir la liste de valeurs suivantes : [0, 1, 2, ..., 19]

Il faut utiliser la fonction de cette manière : list(range(20))

Le premier élément de la liste est « 0 ». Et le nombre d'éléments égale au nombre entre les parenthèses de range()

Notes

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Notes

4EME SEANCE. TRAVAUX PRATIQUES. JEUX ET APPLICATION

CREER UN JEU DE DEVINETTE

- Créer une variable avec un nombre quelconque
- Demander à l'utilisateur de deviner le nombre
- S'il le trouve, afficher « Bravo vous avez trouvé le nombre en x tentative »
- Sinon, afficher « Ce nombre est très grand (ou très petit ça dépend), essayez encore »
- S'il dépasse un nombre maximal d'essais, afficher « dommage, vous avez perdu! Le nombre à deviner est: nombre »

SOLUTION:

Voici une proposition de solution avec boucle while, faites le avec la boucle for

```
nombre = 13
input("j'ai choisi un nombre entre 0 et 20, essayez de le deviner")
nbrMax = 5
essai = 0
N = ""
while essai<nbrMax and N!= nombre:
   essai += 1
   N = int(input("proposez un nombre"))
   if N == nombre:
      print("Bravo vous avez trouvé le nombre en ",essai , " tentative")
   elif N<nombre:
      print("Ce nombre est petit !")
```

```
else:
    print("Ce nombre est grand !")
```

Notes

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

LES MODULES PYTHON

Les modules python, aussi appelés bibliothèque, sont très important dans la programmation. Ils peuvent être comparés à un énorme magasin de bricolage avec un très grand nombre de rayons. Chaque rayon est spécialisé dans un domaine précis.

Il existe des modules dans quasiment tous les domaines : statistique, finance, mathématique, physique, mécanique, aéronautique, développement web, jeux, …

Les modules utilisés dans le cadre de ce cours sont tous installés en même temps que python. Mais il est tout à fait possible d'avoir besoin d'utiliser un module qui n'est pas installé. Dans ce cas, il faut ouvrir une console et saisir la ligne de code suivante : pip unstall nom_du module

COMMENT UTILISER UN MODULE. « RANDOM »

Pour pouvoir utiliser un module, il faut l'importer en entier ou seulement certaines de ses fonctions.

random est un module qui traite les phénomènes aléatoires. La fonction **randint** de ce module permet de choisir aléatoirement des nombres entiers dans un intervalle donné.

Voici un exemple d'utilisation de randint.

Code	Explication
from random **import** randint nombre = randint(0,20) print(nombre)	• Importer la fonction randint à partir du module random • Choisir un nombre aléatoire entre 0 et 20 • Afficher ce nombre

EXERCICE

Utilisez la fonction **randint** dans votre solution du jeu de devinette.

APPLICATIONS

Voici un script qui permet de réviser les tables de multiplication :

```
from random import randint
from random import choice
iterMax = 5
nbrQuestion = 10
msgCourage = ["Tu y es presque, essaie un autre nombre : ",
"Courage, tu vas finir par trouver la bonne réponse : ",
"Essaie encore, ne te décourage pas : ", "Ah dommage, tu y es
presque, essaie encore : "]
msgFelicit = ["Bravo ! Tu es un(e) champion(ne)", "Et encore
une bonne réponse !", "Oui oui oui ouiiiiii", "C'est tout
simplement Génial ! ", "ça c'est un travail de PRO !", "Trop
trop fort, ne change rien !"]
q=0
while q < nbrQuestion:
   nbr1 = randint(2, 9)
   nbr2 = randint(2, 9)
   sol = int(input(str(nbr1) + " x " + str(nbr2) + " = "))
   it = 0
   while it < iterMax:
      if sol == nbr1*nbr2:
         print(choice(msgFelicit))
         print("Félicitations !")
         break
      elif it == iterMax-1:
         print("la réponse juste est ", nbr1*nbr2)
         break
```

```
    else:
        print(choice(msgCourage))
        sol = int(input("Courage, nouvelle réponse :"))
        it = it+1
  q = q+1
```

QUESTIONS :

1. Recopier le script puis l'exécuter
2. Quelle variable permet de réduire ou augmenter le nombre de questions posés par le programme

---

3. Changer le programme pour qu'il ne pose que 3 questions
4. Que se passe-t-il si la variable q dépasse la valeur de la variable nbrQuestion?

---

5. Que se passe-t-il si la variable it dépasse la valeur de la variable iterMax?

---

6. Comment sont obtenues les variables nbr1 et nbr2?
7. Proposer une explication du fonctionnement de la fonction choice importée depuis le module random

---

8. Changer le programme pour qu'il renseigne le nombre des réponses justes et le nombre de réponses fausses.
9. Changer le bloc de la deuxième boucle while par une boucle for

10. Que se passe-t-il si l'utilisateur entre une lettre au lieu d'un chiffre au clavier?

---

11. Changer le programme pour palier à cette éventualité (voir l'aide proposée ci-dessous.)

```
>>> "toto".isnumeric()
False
>>> "5".isnumeric()
True
```

12. Changer le programme pour qu'il donne également le temps mis par l'utilisateur pour répondre à toutes les questions. (Importer la bibliothèque time…)

5EME SEANCE. FONCTIONS ET MANIPULATION DES FICHIERS TEXTE ET CSV

LES FONCTIONS

Quand on est amené à faire des tâches répétitives un certain nombre de fois, mais juste en changeant certains paramètres, il et plus facile de créer une fonction spécifique pour cette tâche.

La fonction peut prendre zéro, un ou plusieurs paramètres en entrée, et renvoyer un résultat en sortie.

Syntaxe pour définir une fonction :

```
def nom_Fonction(paramètres):
  bloc d'instructions
```

Exemple : on dispose d'une liste de plusieurs éléments de type string. On cherche à déterminer l'élément le plus long (qui se compose d'un plus grand nombre de lettre)

```
def plus_grand_mot(liste_mots):
  nbr_lettre = 0
  for m in liste_mots:
    n = len(m)
    if n > nbr_lettre:
      nbr_lettre = n
      ind = liste_mots.index(m)
  return liste_mots[ind]
liste = ["Téléviseur", "Téléphone", "Tablette",
"console de jeux", "Switch", "PS5"]
mot = plus_grand_mot(liste)
print(mot)
```

Résultat : console de jeux

EXERCICE:

Créer une fonction qui affiche la table de multiplication d'un nombre x. la fonction prendra x en argument

SOLUTION :

Voici une proposition de solution. À vous d'en proposer une autre.

```
def table_multiplication(x):
  n = 1
  while n <= 10 :
```

```
    print(x, " x ", n, " = ", n*x)
    n = n + 1
```

résultat :

```
>>> table_multiplication(9)
9  x  1  =  9
9  x  2  =  18
9  x  3  =  27
9  x  4  =  36
9  x  5  =  45
9  x  6  =  54
9  x  7  =  63
9  x  8  =  72
9  x  9  =  81
9  x  10  =  90
```

MANIPULATION DES FICHIERS TEXTE

La manipulation des fichiers texte peut s'avérer rapidement très utile quand on commence à programmer avec python. On peut par exemple avoir besoin d'extraire des données ou d'écrire des informations.

FONCTION OPEN()

Que ce soit pour lire ou pour écrire on utilise la fonction open(). Cette fonction prend en entrée plusieurs arguments.

Open(chemin vers le fichier, comment l'ouvrir) :

Les 2 premiers arguments sont les plus importants, le premier concerne le chemin du fichier qu'on veut ouvrir, en lecture ou en écriture. Et le deuxième concerne justement la manière dont on veut ouvrir le fichier.

- r: pour ouvrir le fichier en lecture seul.
- w: pour ouvrir le fichier en écriture seule. Si le fichier n'existe pas il sera créé, et s'il existe il sera écrasé et recréé
- a: pour continuer l'écriture dans le fichier. Si le fichier n'existe pas il sera créé, et s'il existe l'écriture continuera en fin de fichier

METHODE .READLINES() ET .CLOSE()
Cette méthode permet de fermer le fichier à la fin de son utilisation. Il est très important de ne pas l'oublier.

Exemple 1:

```
dosTravail = "/Users/Documents/Entrees"
F = open(dosTravail+ "/DonneesEntree.txt", "r")
lines = F.readlines()
print(lines)
F.close()
```

Dans ce programme, j'ai préféré utiliser une variable à laquelle j'affecte le chemin vers un dossier de travail. Cette variable est de type string (str). J'ai enregistré préalablement dans ce dossier, un fichier texte (.txt) nommé DonneesEntree.txt.

J'ai créé ensuite une nouvelle variable (F) qui servira comme base d'ouverture du fichier.

Ensuite, pour obtenir le contenu du fichier, j'ai appliqué la méthode **.readlines()** à la variable (F). Cette méthode permet de lire d'un seul coup l'ensemble des lignes du fichier, elle retourne un résultat sous forme de liste.

J'ai créé et affecté cette liste à une variable lines. La méthode .readline() permet de lire le fichier ligne après l'autre. Pour

lire un fichier en entier il faudra la mettre à l'intérieur d'un bloc d'une boucle de répétition (while ou for)

Et enfin, j'ai appliqué la méthode **.close()** à la variable F pour fermer le fichier.

L'INSTRUCTION WITH

Cette instruction est très efficace car elle permet d'ouvrir et gérer la fermeture du fichier dès qu'on a fini de l'utiliser.

Exemple 2:

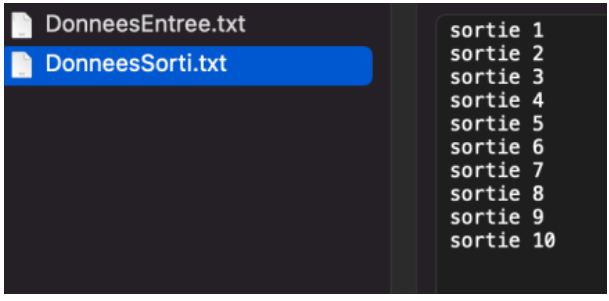
```
with open(dosTravail+ "/DonneesEntree.txt", "r") as F2:
  lines2 = F2.readlines()
  print(lines2)
```

CREER PUIS ECRIRE DANS UN FICHIER

Voici un programme qui permet de créer un fichier puis d'écrire dedans :

```
i = 0
with open(dosTravail+ "/DonneesSortie.txt", "w") as F2:
  while i < 10:
    i = i+1
    F2.write("sortie" + str(i) + "\n")
```

Résultat:



TRAVAUX PRATIQUES:

Inspirez-vous de l'exercice sur les tables de multiplication pour écrire toutes les tables de multiplication dans un document texte. (Tables de multiplication de 1 à 9).

LES FICHIERS CSV:

En général, les informations dans un fichier csv sont rangées par colonnes. Donc elles peuvent être affichées sous forme d'un tableau que l'on peut ouvrir avec un tableur comme excel ou google sheet.

La minipulation des fichiers csv se fait de la même manière que les fichiers texte. Pour créer des données de cette manière, on espace simplement les colonnes par des points virgules (;).

Conclusion

Félicitations, vous êtes allé(e) jusqu'au bout de ce guide ! 🎉

Vous connaissez désormais les bases de Python et vous avez même créé plusieurs mini-projets concrets. Vous pouvez être fier : ce sont des compétences que beaucoup n'osent même pas commencer à explorer. Que vous soyez enseignant, parent, ou apprenant motivé, vous avez maintenant de quoi aller plus loin avec sérénité.

💡 **Les corrigés complets de tous les exercices** ainsi qu'une **section "Pour aller plus loin"** (avec des idées de projets, des ressources complémentaires et des vidéos bonus) vous attendent sur cette page de mon blog : https://cours-maths-python.com/correction-exercices-bonus-apprendre-python-en-5-heures/

N'hésitez pas à me faire des retours de vos succès et de vos difficultés en laissant des commentaires en dessous des articles du blog ou en m'envoyant un mail au

hicham@cours-maths-python.com

Et si on allait plus loin ensemble ?

Vous avez aimé ce guide mais vous aimeriez :

- Être accompagné pas à pas dans votre apprentissage du code ?
- Créer vos propres séquences pédagogiques avec Python ou Scratch ?
- Intégrer le numérique dans votre enseignement, ou aider votre enfant à progresser efficacement ?

📩 Je propose un accompagnement personnalisé : **coaching individuel**, **formations**, ou **ateliers sur mesure**, en fonction de votre profil et de vos besoins.

👉 Intéressé ? rendez-vous sur cette page :

https://cours-maths-python.com/aller-plus-loin/

Cliquez sur le lien ou copier-coller le sur un navigateur.

Au plaisir de vous accompagner dans la suite de votre aventure avec Python !

Hicham Youssef

hicham@cours-maths-python.com