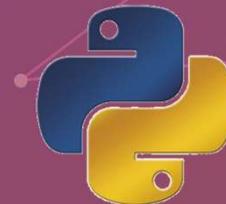


Initiation à la Programmation

Le langage



"Any fool can write code that a computer can understand. Good programmers write code that humans can understand."

Vinod Kumar Nair

Objectif



L'objectif global est de donner aux participants une base solide pour commencer à programmer en Python, en leur fournissant les compétences nécessaires pour résoudre des problèmes simples et pour continuer à explorer et à développer leurs connaissances en programmation par la suite.

Une progression graduelle et une approche pratique seront appliquées, l'accent sera mis sur la résolution de problèmes.

AGENDA



1

Introduction

- Langage de programmation
- Compilation vs Interprétation
- Installation et configuration de l'environnement de programmation

2

Le langage Python (Syntaxe de base)

- Variables
- Opérateurs
- Type de données
- Entrées/Sorties
- Structures de Contrôle

3

The power of code

Visionnez la vidéo



vidéo <https://youtu.be/nKlu9yen5nc>



The power of code



vidéo <https://youtu.be/nKlu9yen5nc>



Répondez aux questions suivantes :

- Quel est le message principal de la vidéo sur l'importance d'apprendre à coder ?
- Pourquoi les intervenants de la vidéo pensent-ils que savoir coder est important aujourd'hui ?
- Quels sont les secteurs ou domaines mentionnés dans la vidéo où la programmation est utilisée ?
- Selon vous, pourquoi la programmation est-elle souvent qualifiée de "compétence essentielle" dans la vidéo ?
- D'après la vidéo, quelles qualités ou compétences peuvent être développées grâce à la programmation ?
- Quels obstacles ou idées reçues concernant l'apprentissage de la programmation sont évoqués dans la vidéo ?

The power of code

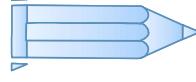
Votre avis personnel :

- Avez-vous été surpris par le fait que certaines personnes présentées dans la vidéo n'avaient pas de formation en informatique avant de commencer à coder ? Pourquoi ?
- Comment imaginez-vous que l'apprentissage de la programmation pourrait transformer votre avenir ou celui des autres ?
- Après avoir vu cette vidéo, êtes-vous plus motivé(e) à apprendre la programmation ? Pourquoi ?
- Pouvez-vous penser à un problème de votre quotidien ou de votre communauté que vous aimeriez résoudre grâce à la programmation ?
- Selon vous, quel impact pourrait avoir l'apprentissage généralisé de la programmation dans votre pays ou dans le monde ?

Langage de programmation ...

... est une notation permettant d'écrire des algorithmes pour qu'ils puissent être exécutés par un ordinateur.

Les langages de programmation et le langage naturel partagent certaines similitudes dans la manière dont ils permettent aux humains de communiquer avec les ordinateurs ou avec d'autres humains. Un langage de programmation est composé de :

- | | | |
|------------|-------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------|
| alphabet |  | – ensemble de symboles utilisés pour construire des mots. |
| lexis |  | – ensemble de mots que la langue offre à ses utilisateurs (vocabulaire). |
| syntaxe |  | – ensemble de règles utilisées pour déterminer si une certaine chaîne de mots forme une phrase valide (grammaire). |
| sémantique |  | – ensemble de règles déterminant si une certaine phrase a du sens. |

Le langage de programmation doit fournir un environnement de traduction permettant de le rendre compréhensible par la machine.

Caractéristiques du langage Python ...

- 01** Multiplateforme
- 02** Gratuit
- 03** Paradigmes de programmation multiples
- 04** Bibliothèque standard robuste

Fonctionne sur de nombreux systèmes d'exploitation : Windows, Mac OS X, Linux, Android, iOS, depuis les mini-ordinateurs Raspberry Pi jusqu'aux supercalculateurs.

Peut-être installé sur autant d'ordinateurs que nécessaire

Permet l'utilisation de plusieurs styles de programmation. Il est possible de structurer et écrire du code qui reflète une philosophie ou un ensemble de principes différents (impératif, orienté-objet, fonctionnel)

La bibliothèque standard est très vaste et très bien documentée

Un peu d'histoire de Python ...



Python a été créé par Guido van Rossum, un programmeur néerlandais. Il a commencé à travailler sur Python en décembre 1989 et a publié la première version publique, Python 0.9.0, en février 1991.

L'origine du nom "Python" est liée à l'admiration de Guido van Rossum pour le groupe de comédie britannique Monty Python. Le nom n'a aucun lien avec le serpent du même nom.

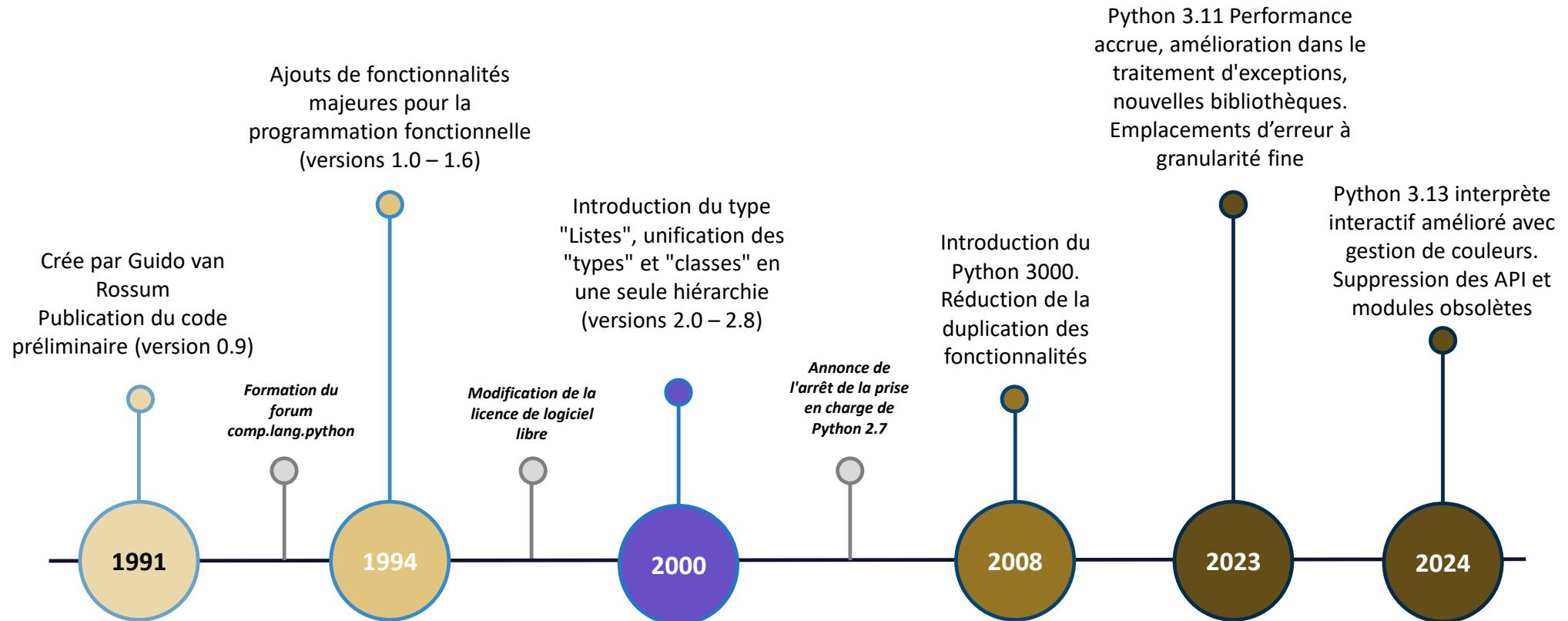


Les principaux objectifs de Guido van Rossum lorsqu'il a créé Python étaient de concevoir un langage de programmation avec une syntaxe simple, lisible et élégante. Il voulait un langage qui permettrait aux programmeurs d'écrire du code clair et concis, tout en étant facile à apprendre et à utiliser.

Depuis sa création, Python a connu une adoption rapide et une croissance significative en popularité.

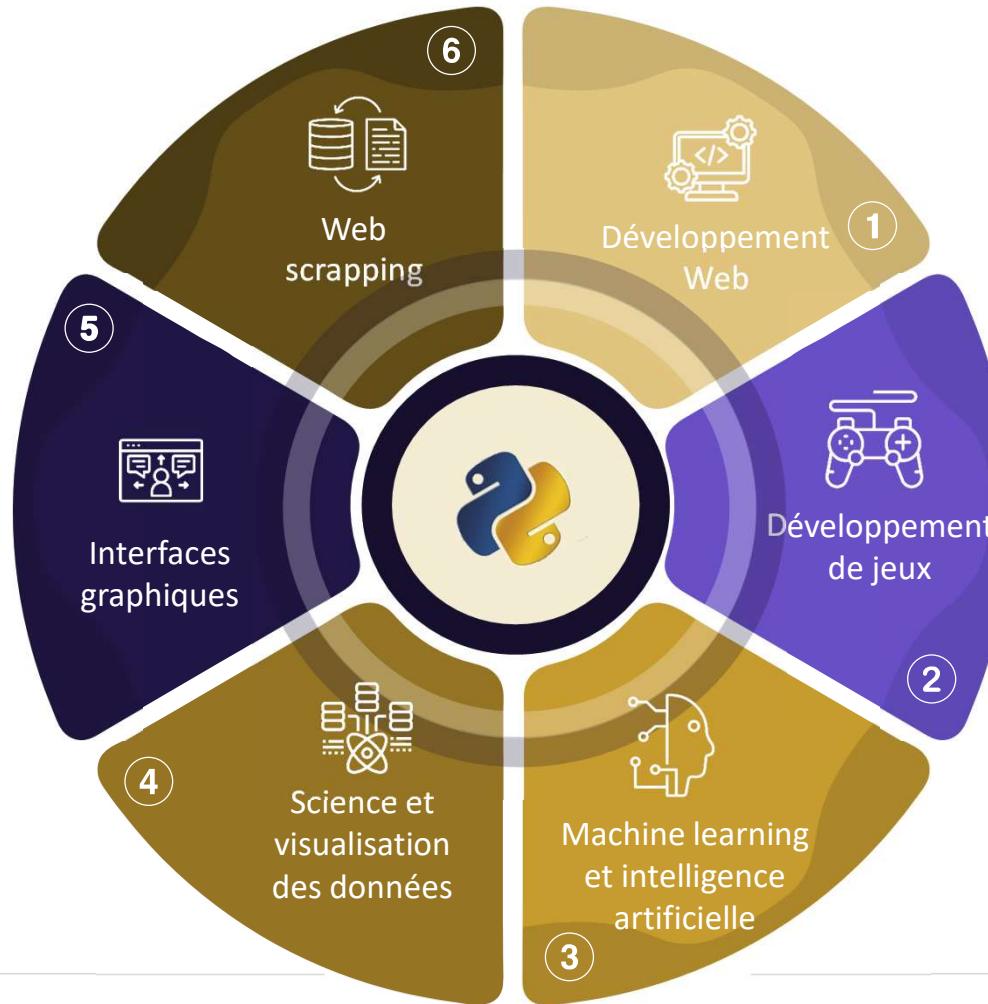
Roadmap de Python ...

10



Les usages de Python ...

11



Le langage Python ...

... est un langage de programmation de haut niveau dont la syntaxe est simple.

Il est composé de :

- Familles d'instructions :
 - l'affectation de variables
 - la lecture / écriture
 - les conditionnelles
 - les boucles
- Types de données
- Bibliothèques de fonctions
- Mécanismes de définition des fonctions

Résumé : structure d'un programme Python type

```

# -*- coding:Utf8 -*-
#####
# Programme Python type          #
# auteur : G.Swinnen, Liège, 2009 #
# licence : GPL                  #
#####

#####
# Importation de fonctions externes :
from math import sqrt

#####
# Définition locale de fonctions :
def occurrences(car, ch):
    "Cette fonction renvoie le \
    nombre de caractères <car> \
    contenus dans la chaîne <ch>"

    nc = 0
    i = 0
    while i < len(ch):
        if ch[i] == car:
            nc = nc + 1
        i = i + 1
    return nc

#####

# Corps principal du programme :
print("Veuillez entrer un nombre :")
nbr = eval(input())
print("Veuillez entrer une phrase :")
phr = input()
print("Entrez le caractère à compter :")
cch = input()

no = occurrences(cch, phr)
rc = sqrt(nbr**3)

print("La racine carrée du cube", end=' ')
print("du nombre fourni vaut", end=' ')
print(rc)

print("La phrase contient", end=' ')
print(no, "caractères", cch)

```

Un programme Python contient en général les blocs suivants, dans l'ordre :

- Quelques instructions d'initialisation (importation de fonctions et/ou de classes, définition éventuelle de variables globales).
- Les définitions locales de fonctions et/ou de classes.
- Le corps principal du programme.

Le programme peut utiliser un nombre quelconque de fonctions, lesquelles sont définies localement ou importées depuis des modules externes. Vous pouvez vous-même définir de tels modules.

- La définition d'une fonction comporte souvent une liste de PARAMÈTRES. Ce sont toujours des VARIABLES, qui recevront leur valeur lorsque la fonction sera appelée.

Une boucle de répétition de type 'while' doit toujours inclure au moins quatre éléments :

- l'initialisation d'une variable 'compteur' ;
- l'instruction while proprement dite, dans laquelle on exprime la condition de répétition des instructions qui suivent ;
- le bloc d'instructions à répéter ;
- une instruction d'incrémentation du compteur.

- La fonction "renvoie" toujours une valeur bien déterminée au programme appelant. Si l'instruction 'return' n'est pas utilisée, ou si elle est utilisée sans argument, la fonction renvoie un objet vide : 'None'.

Le programme qui fait appel à une fonction lui transmet d'habitude une série d'ARGUMENTS, lesquels peuvent être des valeurs, des variables, ou même des expressions.

Traitement du langage...

Il existe deux façons différentes de transformer un programme d'un langage de programmation de haut niveau en langage machine :

- **COMPILEATION** : le programme source est vérifié entièrement et traduit une fois en obtenant un fichier exécutable (contenant le code machine). Le programme qui effectue cette traduction est appelé **compilateur** ou **traducteur** ;
- **INTERPRÉTATION** : chaque ligne du code source est vérifiée, traduite et exécutée. Le programme effectuant ce type de transformation est appelé **interpréteur**.

Python est un langage interprété

Image : <https://edube.org/learn/python-essentials-1>

L'environnement Python ...

Navigateur ou Installation

Pour construire et exécuter un programme écrit en Python , il est nécessaire de disposer de quelques outils

- **Interprète en ligne**

Programiz



<https://www.programiz.com/python-programming/online-compiler/>

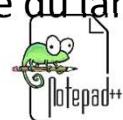


Trinket

<https://trinket.io/python3>

- **Un éditeur syntaxique** : il s'agit d'un éditeur de texte avancé qui prend en charge un large éventail de langages de programmation. L'éditeur colore différemment les éléments du code source (mots-clés, variables, chaînes de caractères, commentaires, etc.) selon leur rôle dans le langage de programmation. Il applique un formatage visuel distinctif à chaque élément, ce qui facilite la lecture, la compréhension et l'identification rapide des parties du code. Dans ce cas il faut installer l'interprète du langage.

Notepad ++



<https://notepad-plus-plus.org/downloads/>



Visual Studio Code

<https://code.visualstudio.com/>

L'environnement Python ...

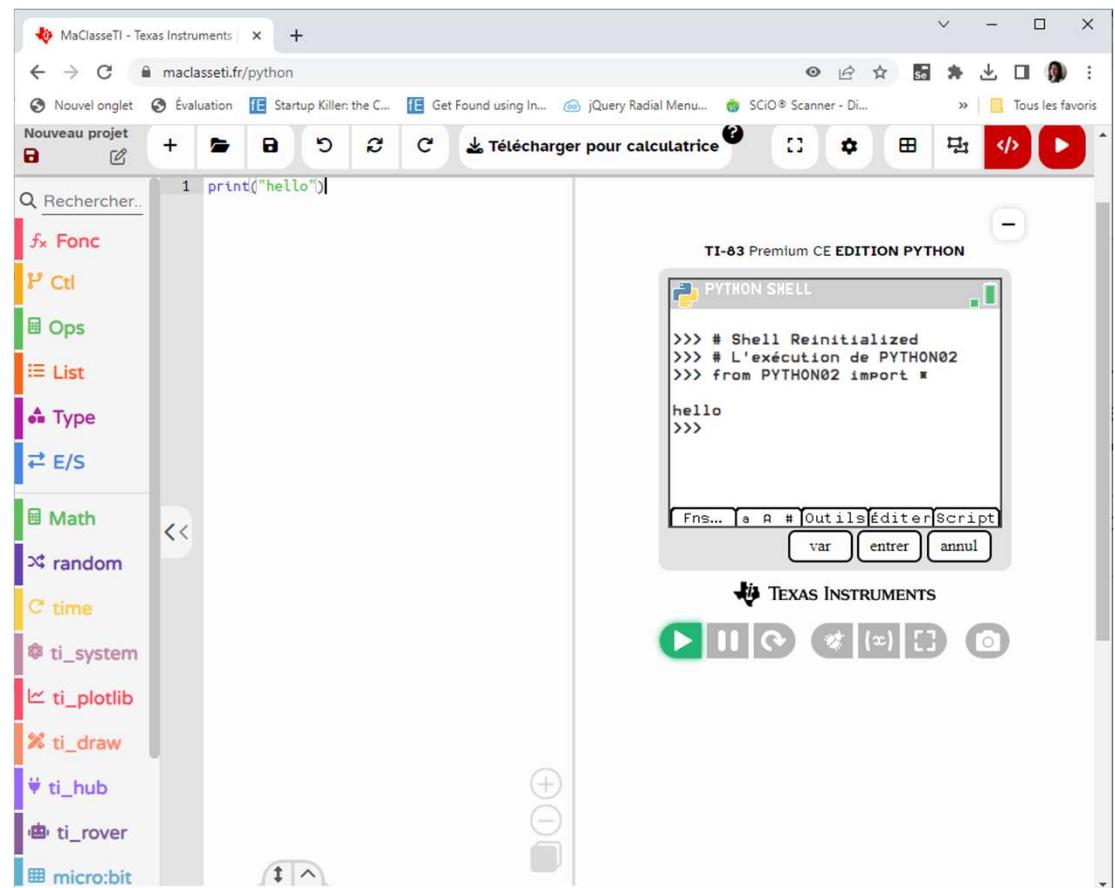
- **IDE (*Integrated Development Environment*)** : est un ensemble d'outils lequel permet d'assister et formaliser le travail de création de logiciels (éditeur, débuggeur, exécution).

Eclipse  [eclipse](https://eclipseide.org/)
<https://eclipseide.org/>

Autres outils Python ...

Il existe plusieurs outils pour travailler avec Python et des calculatrices.

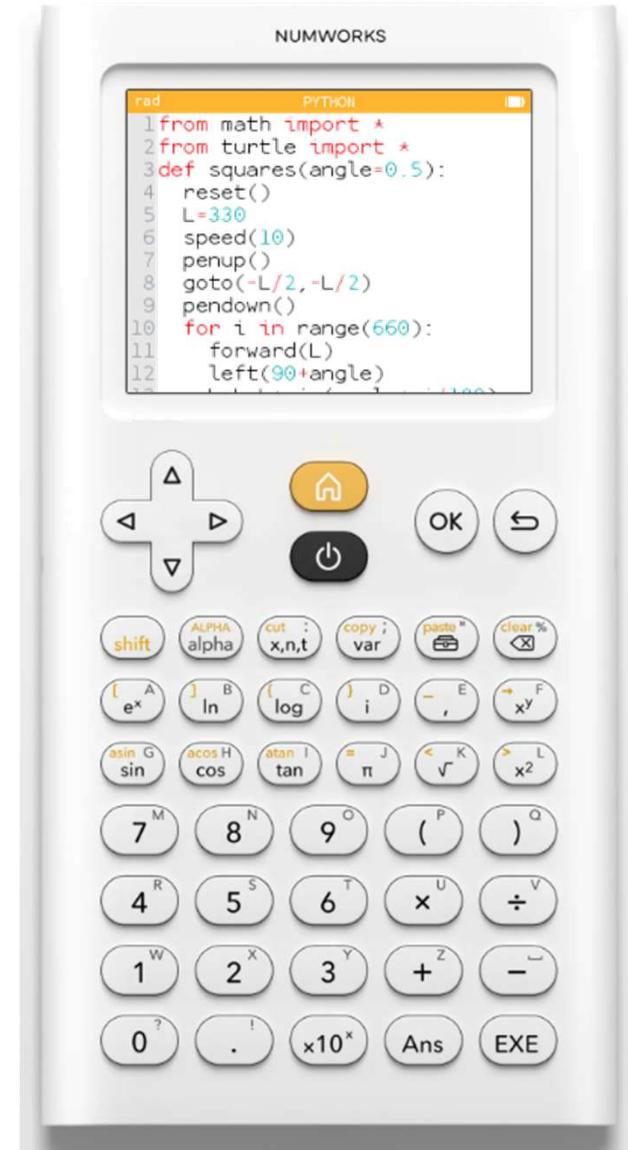
- **MicroPython** sur les calculatrices programmables : Certaines calculatrices programmables, comme la TI-83/84 ou la Casio fx-9860GII, permettent l'installation de MicroPython. Cela permet d'écrire et d'exécuter des programmes Python directement sur la calculatrice. La calculatrice TI-83 Premium CE propose également un émulateur en ligne (<https://maclasseti.fr/python>)



Autres outils Python ...

- **Numworks** : offre un émulateur de la calculatrice, en ligne, et qui permet également l'écriture et l'exécution du code python :

<https://www.numworks.com/fr/simulateur/>



Autres outils Python ...

Il existe plusieurs outils pour travailler avec Python et des calculatrices.

- **Émulateurs de calculatrices** : il est possible d'utiliser des émulateurs de calculatrices sur l'ordinateur pour développer et tester des programmes Python avant de les transférer sur une calculatrice réelle. Les émulateurs TI-83/84 et TI-Nspire sont couramment utilisés. L'émulateur TI-Nspire est fourni avec la calculatrice, il possible également d'acquérir la licence séparément
- **Jupyter Notebooks** : il est possible d'utiliser des Jupyter Notebooks pour développer et exécuter du code Python de manière interactive. Ils peuvent être exécutés localement sur l'ordinateur, mais il existe également des solutions en ligne.

Autres outils Python ...

Il existe plusieurs interprètes Python disponibles pour les téléphones mobiles. Voici quelques-uns d'entre eux :

1. **Pydroid** : Pydroid est une application Android qui fournit un environnement de développement Python complet sur votre téléphone. Elle comprend un éditeur de code, un interpréteur Python, et prend en charge des bibliothèques populaires (il existe en version gratuite avec des publicités et une version payante).
2. **QPython** : QPython est une application Android qui permet d'exécuter du code Python sur les appareils mobiles. Elle offre un éditeur de code, un interpréteur Python, et prend en charge certaines bibliothèques tierces.
3. **Pythonista (iOS)** : Pythonista est une application iOS qui permet de développer et d'exécuter du code Python sur les appareils Apple. Elle propose un éditeur de code avancé et prend en charge des fonctionnalités puissantes.
4. **Juno (iOS)** : Juno est une application iOS conçue pour travailler avec des **notebooks Jupyter** sur iPad. Il est possible d'écrire et d'exécuter du code Python de manière interactive.

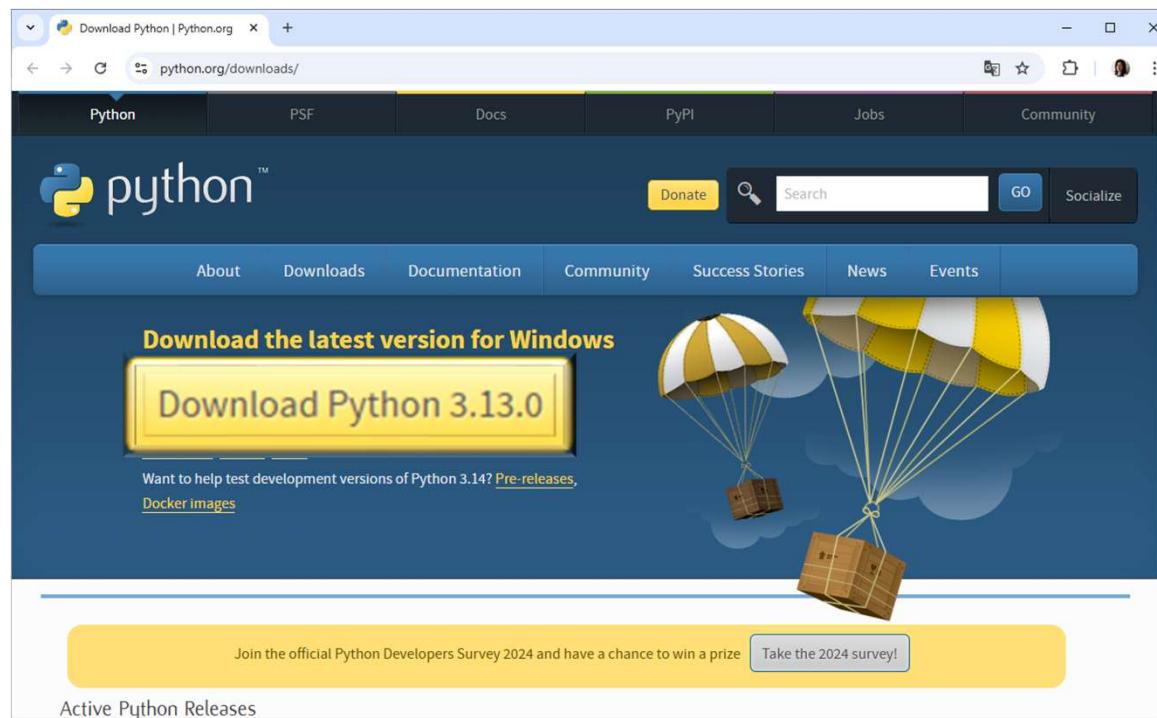
Installation

Sous Windows

L'installation de Python ...

sur Windows :

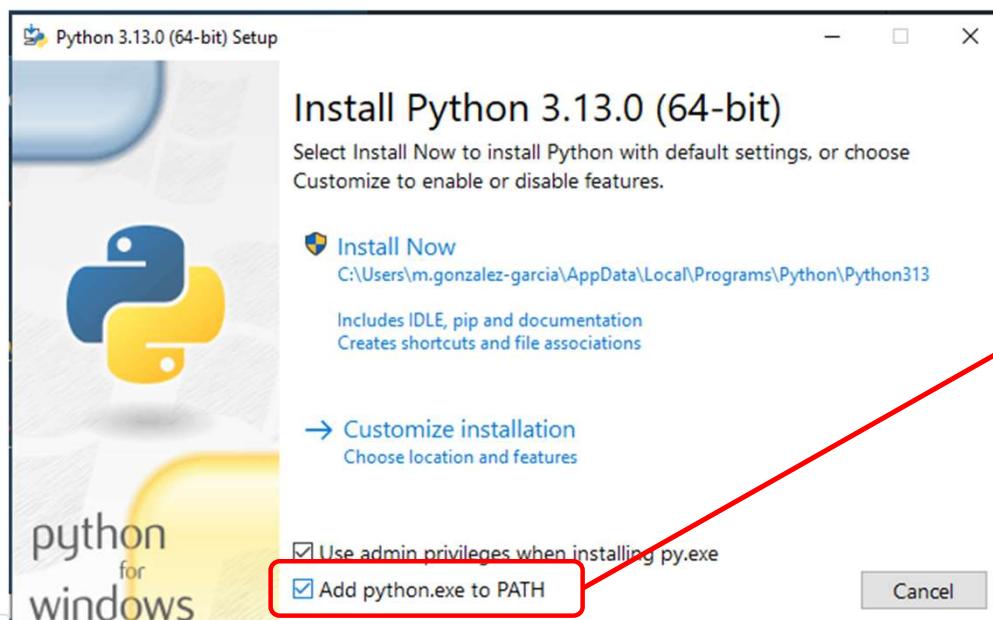
- Rendez-vous sur le site officiel de Python (<https://www.python.org/downloads/>) et téléchargez la dernière version stable de Python pour Windows.



L'installation de Python ...

sur Windows :

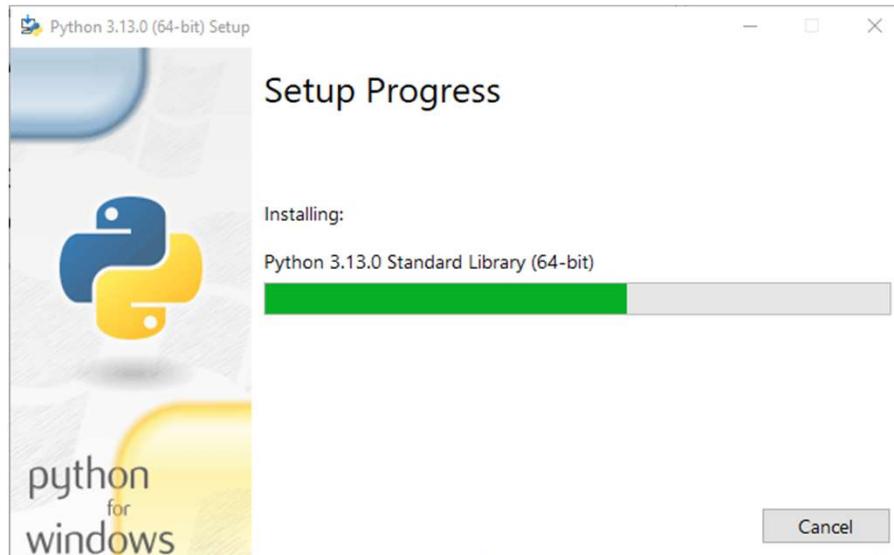
- Exécutez le fichier d'installation téléchargé (par exemple, python-3.13.0-amd64.exe).
- Cochez la case "**Add python.exe to PATH**" (Ajouter **python.exe** au PATH) pendant l'installation pour pouvoir utiliser Python à partir de n'importe quel emplacement ou dans la ligne de commande.



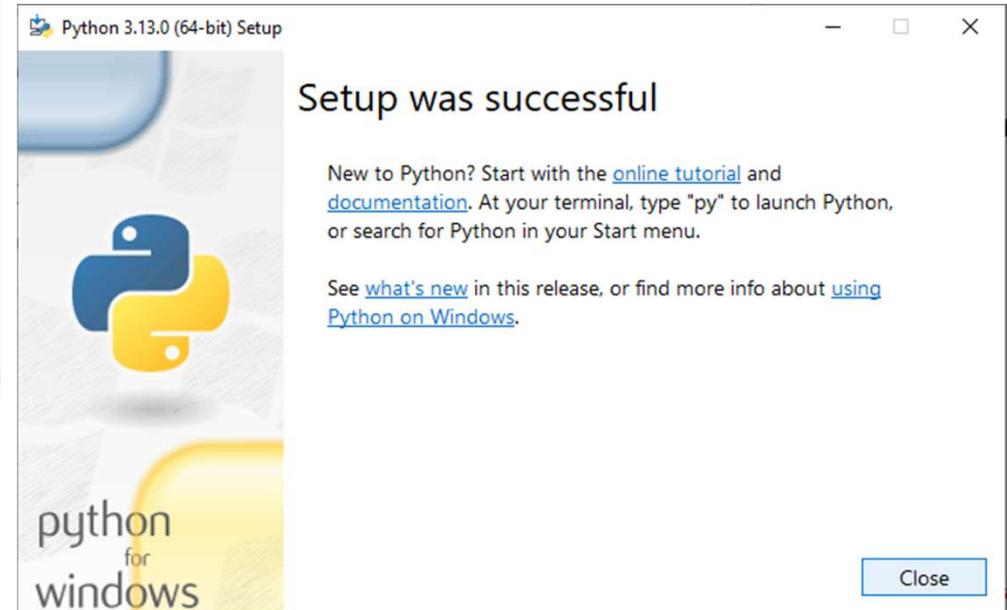
- **PATH** est une variable d'environnement sur les systèmes d'exploitation qui spécifie les répertoires dans lesquels le système d'exploitation recherche les fichiers exécutables lorsqu'une commande est entrée dans la ligne de commande ou dans l'invite de commandes

L'installation de Python ...

sur Windows :

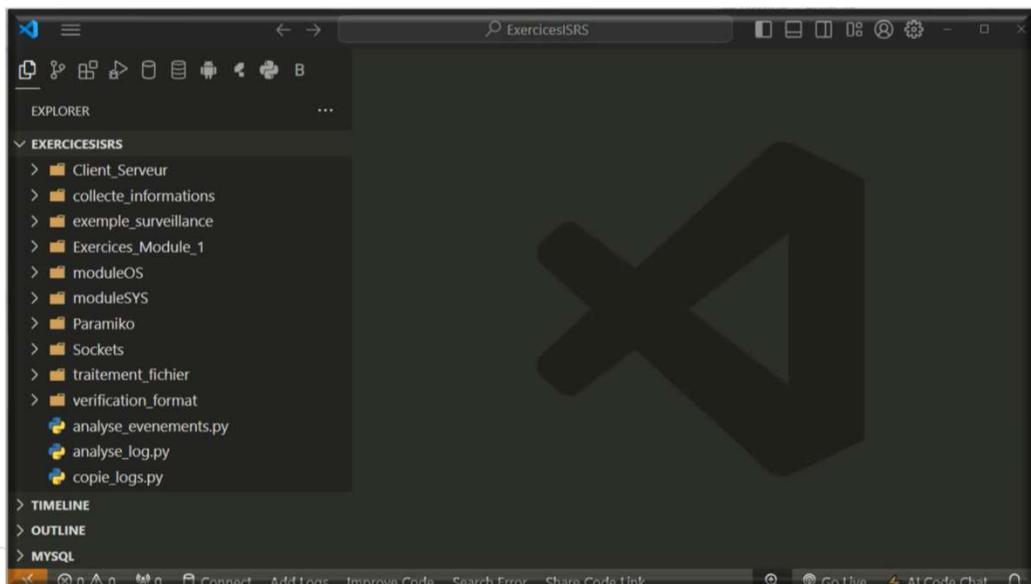


Suivez les instructions de l'assistant d'installation pour terminer le processus.



Configuration de l'environnement de travail ...

- Choix d'un éditeur de code ou de l'IDE: exemple **VSCode**
- Configuration de l'éditeur de code :
 - **Installer l'extension Python pour VSCode** : cela permettra à VSCode de fournir des fonctionnalités spécifiques à Python, telles que la mise en évidence de syntaxe, le débogage, la gestion des environnements virtuels, etc. Pour installer l'extension Python, accédez à l'onglet Extensions (représenté par  dans la barre latérale gauche de VSCode, recherchez "Python" et installez l'extension proposée par Microsoft.

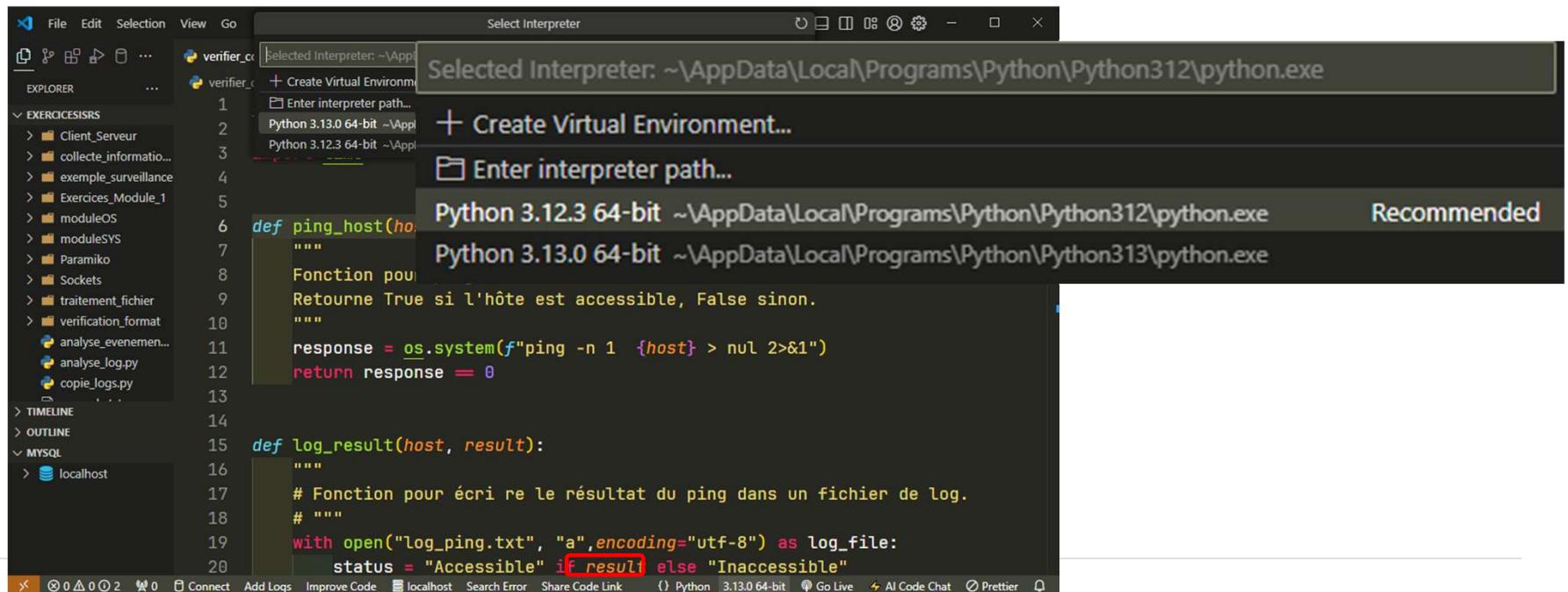


VSCode fournit un ensemble d'extensions qui permettent d'augmenter la productivité au moment du codage :

- **autoDocstring** : génère rapidement des *docstrings* pour les fonctions Python.
- **Python Indent** : Correction de l'indentation de Python
- **Indent-Rainbow** : Pour rendre l'indentation plus lisible

Configuration de l'environnement de travail ...

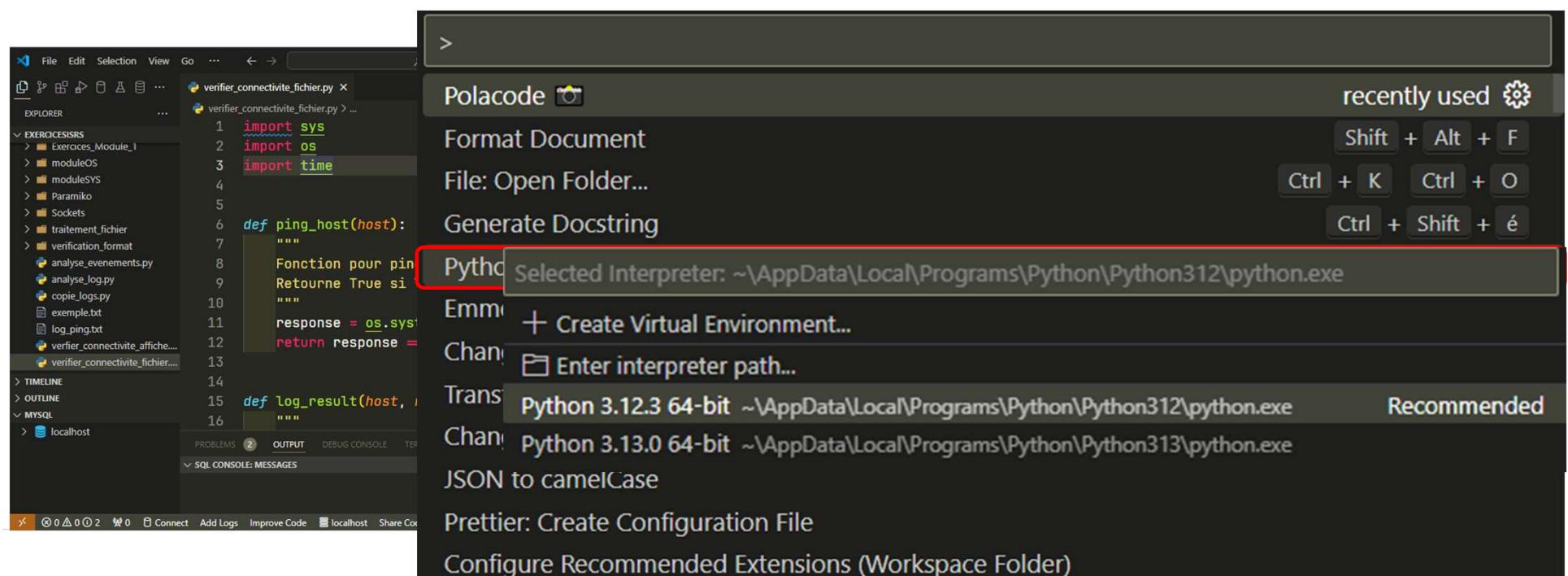
- Choix d'un éditeur de code ou de l'IDE: exemple **VSCode**
- Configuration de l'éditeur de code :
 - **Configurer l'interprète Python** : cliquez sur l'icône qui représente la version actuelle de Python (ex . **3.13.0 64-bit**) dans la barre d'outils inférieure de **VSCode**. Sélectionnez l'interprète Python de la version que vous souhaitez utiliser.



Configuration de l'environnement de travail ...

30

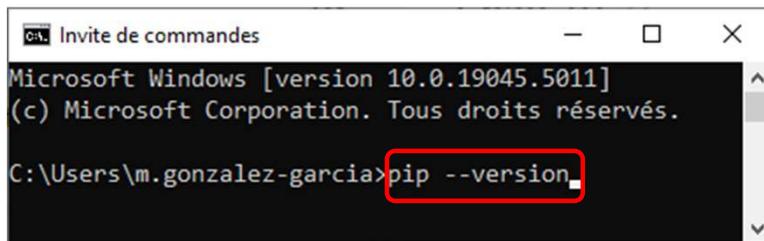
- Choix d'un éditeur de code ou de l'IDE: exemple **VSCode**
- Configuration de l'éditeur de code :
 - Configurer l'interprète Python : utilisez le raccourci "**Ctrl + Shift + P**" pour ouvrir la palette de commandes et tapez "**Python: Select Interpreter**". Sélectionnez l'interprète Python de la version que vous souhaitez utiliser.



Configuration de l'environnement de travail ...

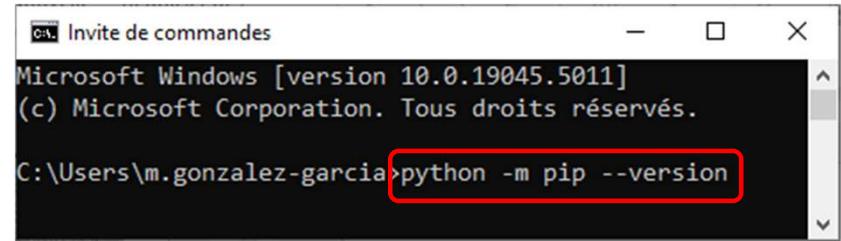
- Installation de **pip** (package installer for Python) : **pip** est le système de gestion de paquets Python, il permet d'installer et de gérer des bibliothèques et des outils supplémentaires.

1. Vérifier si **pip** est déjà installé : Dans l'invite de commande, écrivez



```
Invite de commandes
Microsoft Windows [version 10.0.19045.5011]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\m.gonzalez-garcia>pip --version
```

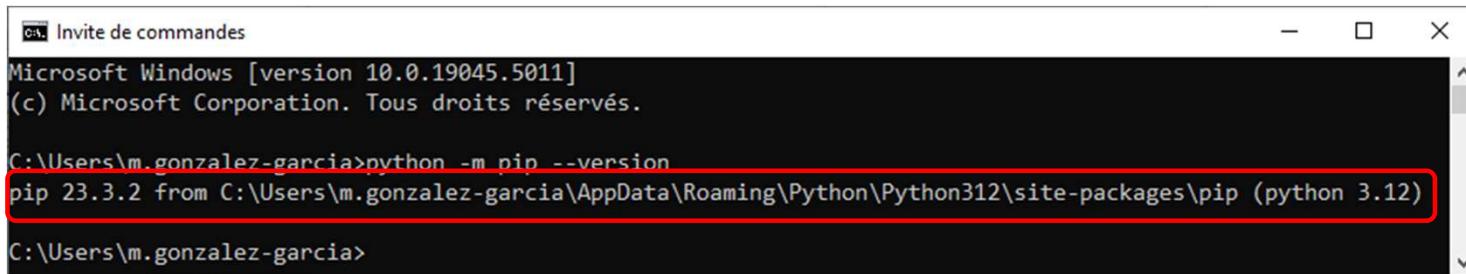


OU

```
Invite de commandes
Microsoft Windows [version 10.0.19045.5011]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\m.gonzalez-garcia>python -m pip --version
```

Si **pip** est installé, les informations sur la version seront affichées



```
Invite de commandes
Microsoft Windows [version 10.0.19045.5011]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\m.gonzalez-garcia>python -m pip --version
pip 23.3.2 from C:\Users\m.gonzalez-garcia\AppData\Roaming\Python\Python312\site-packages\pip (python 3.12)

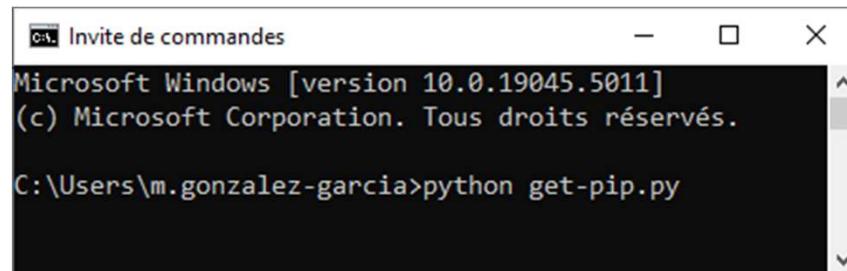
C:\Users\m.gonzalez-garcia>
```

Configuration de l'environnement de travail ...32

- Installation de **pip** (package installer for Python) : **pip** est le système de gestion de paquets Python, il permet d'installer et de gérer des bibliothèques et des outils supplémentaires.

2. Installez **pip** si nécessaire :

- Télécharger le script [get-pip.py](#) à partir de ce lien et placez-le dans un répertoire facile d'accès.
- Ouvrir l'invite de commande, allez dans le répertoire où get-pip.py a été téléchargé, puis exécuter :



```
Microsoft Windows [version 10.0.19045.5011]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\m.gonzalez-garcia>python get-pip.py
```

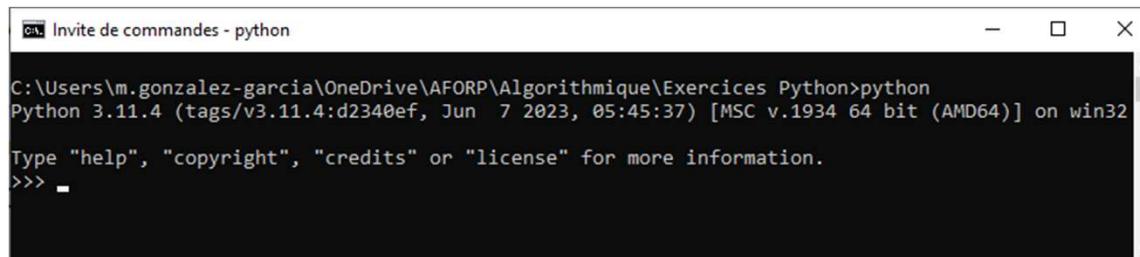
3. Ajouter **pip** au PATH (si nécessaire)

Premier contact avec Python ...

L'interprète Python analyse la ligne de commande et l'environnement à la recherche de différents paramètres.

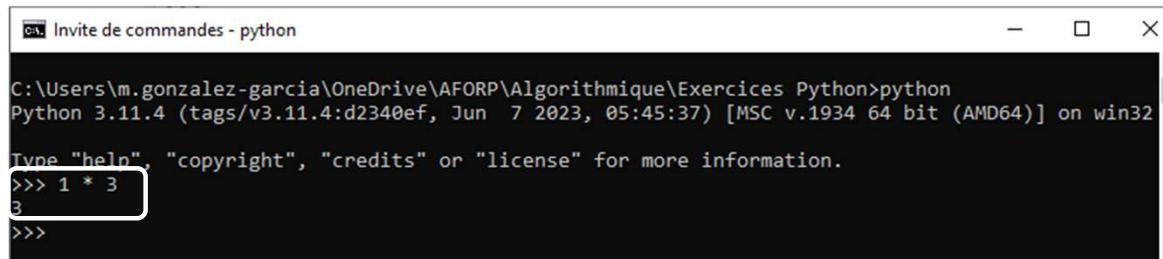
L'interface de l'interpréteur ressemble à celle du **shell UNIX**

- La commande "**python**" permet d'initier l'exécution de l'interprète
- Les trois chevrons "**>>>**" est l'invite de commande (prompt en anglais) de l'interprète Python. Ici, Python attend une commande qui doit être saisi au clavier.



```
C:\Users\m.gonzalez-garcia\OneDrive\AFORP\Algorithmique\Exercices Python>python
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun  7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> -
```

- Python exécute la commande directement et affiche le résultat.



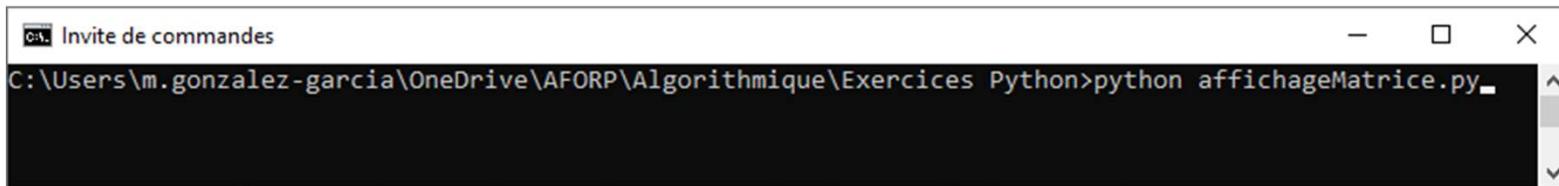
```
C:\Users\m.gonzalez-garcia\OneDrive\AFORP\Algorithmique\Exercices Python>python
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun  7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 1 * 3
3
>>>
```

La commande exit permet de quitter l'interprète

Premier contact avec Python ...

Exécution d'un programme :

- Lorsque la suite d'instructions à exécuter est plus complexe, il est préférable de les enregistrer dans un fichier.
- L'extension du fichier qui contient le code source est ".py"
- Pour exécuter le programme, la commande "**python**", suivie du chemin permettant accéder au fichier , est utilisée



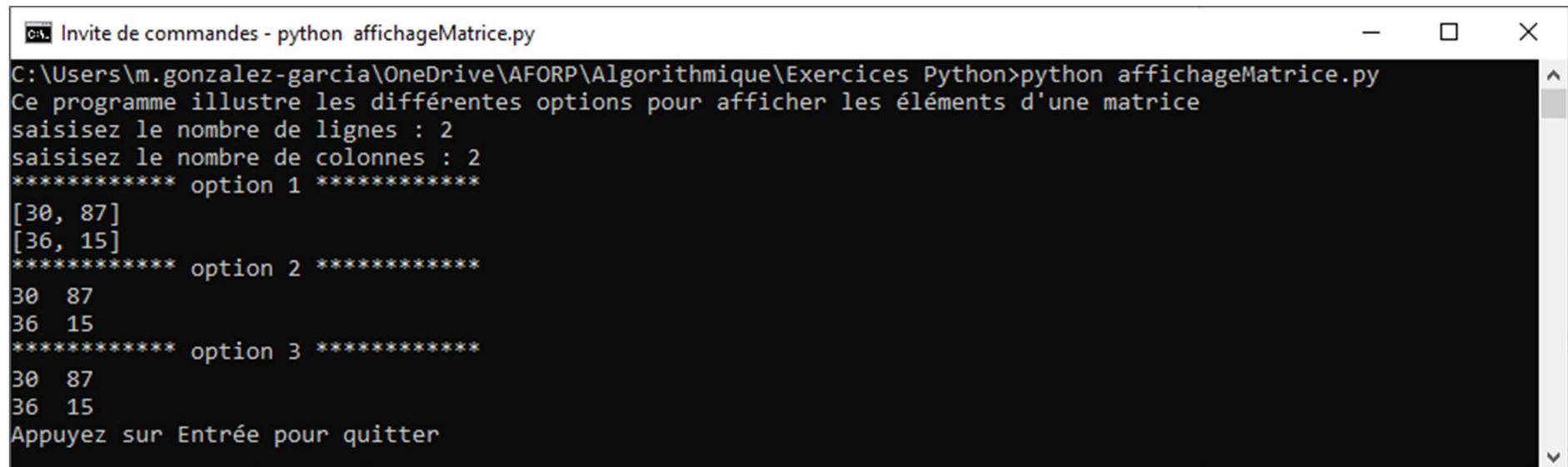
A screenshot of a terminal window titled "Invite de commandes". The window shows the command "C:\Users\m.gonzalez-garcia\OneDrive\AFORP\Algorithmique\Exercices Python>python affichageMatrice.py" being typed. The terminal has a dark background and light-colored text. There are standard window controls (minimize, maximize, close) at the top right.

Dans l'exemple le nom du fichier qui contient le code source du script est "**affichageMatrice.py**"

La commande exit permet de quitter l'interprète

Premier contact avec Python ...

Le résultat :



```
Invite de commandes - python affichageMatrice.py
C:\Users\m.gonzalez-garcia\OneDrive\AFORP\Algorithmique\Exercices Python>python affichageMatrice.py
Ce programme illustre les différentes options pour afficher les éléments d'une matrice
saisissez le nombre de lignes : 2
saisissez le nombre de colonnes : 2
***** option 1 *****
[30, 87]
[36, 15]
***** option 2 *****
30 87
36 15
***** option 3 *****
30 87
36 15
Appuyez sur Entrée pour quitter
```

La commande exit permet de quitter l'interprète

La syntaxe

AGENDA

45



1

Introduction

- Langage de programmation
- Compilation vs Interprétation
- Installation et configuration de l'environnement de programmation

2

Le langage Python (Syntaxe de base)

- Variables
- Opérateurs
- Type de données
- Entrées/Sorties
- Structures de Contrôle

Le langage (mécanismes de documentation) ...

Les commentaires

Permettent d'expliquer le code, ils ne sont pas interprétés par Python. L'utilisation de commentaires fait partie des bonnes pratiques de programmation. Ils doivent être synchronisés avec le code, c'est-à-dire que si le code est modifié, les commentaires doivent l'être aussi.

En Python, les commentaires commencent avec un caractère dièse, "#", et s'étendent jusqu'à la fin de la ligne. Un commentaire peut apparaître au début d'une ligne ou à la suite d'un espace ou de code, mais pas à l'intérieur d'une chaîne de caractères littéraux.

Ceci est un commentaire

```
# Cet algorithme permet d'échanger les valeurs des variables "VarA" et "VarB"
VarA, VarB = input("Saisissez la première valeur de VarA : "), input(
    "Saisissez la deuxième valeur de VarB : ")
# Affichage des valeurs avant l'échange
print("Avant l'échange : ", "\n", "VarA = ", VarA, "\n", "VarB = ", VarB)
# Pour l'échange il nécessaire d'utiliser une variable auxiliaire VarC
VarA, VarB = VarB, VarA
# Affichage des valeurs après l'échange
print("Après l'échange : ", "\n", "VarA = ", VarA, "\n", "VarB = ", VarB)
```

Le langage (mécanismes de documentation) ...

Les Docstrings :

Un **Docstring** est une chaîne utilisée pour fournir des descriptions détaillées des modules, des classes, des fonctions ou des méthodes Python. Les **Docstrings** sont délimités par trois guillemets """

Exemple :

Ceci est un
bloc DocString

```
def estBissextile(annee:int):
    """_summary_ cette fonction indique si l'année envoyée en paramètre est bissextile

    Args:
        annee (_type_): entier

    Returns:
        _type_: booléen
    """
    if (annee % 4 == 0 and annee % 100 != 0) or (annee % 400 == 0):
        return True
    else:
        return False
```

Le langage (mécanismes de documentation) ...

Les Docstrings :

- Les **Docstrings** sont placées sous le bloc d'en-tête d'une fonction, classe, module, etc.
- Contrairement aux commentaires, les **Docstrings** sont accessibles en tant qu'objets Python à partir du code lui-même. Ils peuvent être récupérés en utilisant l'attribut `__doc__` de l'objet.
- Les **Docstrings** sont utilisées pour générer automatiquement une documentation plus formelle, souvent avec l'aide d'outils tels que *Sphinx* ou *pdoc*, qui peuvent extraire et formater les informations contenues dans les **Docstrings** pour créer des documents dans un format lisible par les humains, par exemple du HTML.

Le langage (les commentaires) ...

Des outils tels que "pdoc" permettent de générer une page html avec la documentation

Exemple (à exécuter depuis la ligne de commandes):

```
pdoc --html estBissextileFonction.py
```

The screenshot shows a web browser window displaying a generated API documentation page. The title bar says "estBissextileFonction API document". The page has a header "Module estBissextileFonction". On the left, there's an "Index" section and a "Functions" section containing a single entry "estBissextile". On the right, there's a detailed view of the "estBissextile" function. It shows the function definition: `def estBissextile(annee: int)`. Below it is a summary: "summary cette fonction indique si l'année envoyée en paramètre est bissextile". Under "Args", it shows the parameter `annee : _type_` as "entier". Under "Returns", it shows the return type as "`_type_` booléen". There are "EXPAND SOURCE CODE" buttons at the bottom of each section.

La programmation (conventions de nommage)

Convention : “ce qui est admis d'un commun accord, tacite ou explicite”

Une **convention de nommage** est un ensemble de règles destinées à choisir les noms des éléments du programme (variables, fonctions, constantes), dans le code source et la documentation.

Dans un projet elles permettent à l'équipe de travailler ensemble. L'objectif étant :

- rendre le code source plus facile à lire et à comprendre avec moins d'efforts ;
- améliorer l'apparence du code source (par exemple, en interdisant les noms trop longs ou les abréviations) ;

“You should name a variable using the same care with which you name a first-born child.”

Robert C. Martin

La programmation (conventions de codage)

51



CommitStrip.com



Il est important de **les identifier**, de **les faire évoluer** et de favoriser leur **acceptation** afin qu'elles puissent être **appliquées** par l'équipe de développement

La programmation (conventions de nommage)

Quelques conventions :

- **Camel case (camelCase) :**

Est une notation consistant à écrire l'ensemble de mots des variables en les liant sans espace ni ponctuation, et en mettant en capitale la première lettre de chaque mot (*upper camel case*). La première lettre du premier mot peut être en bas ou haut de casse mot (*lower camel case*), selon la convention.



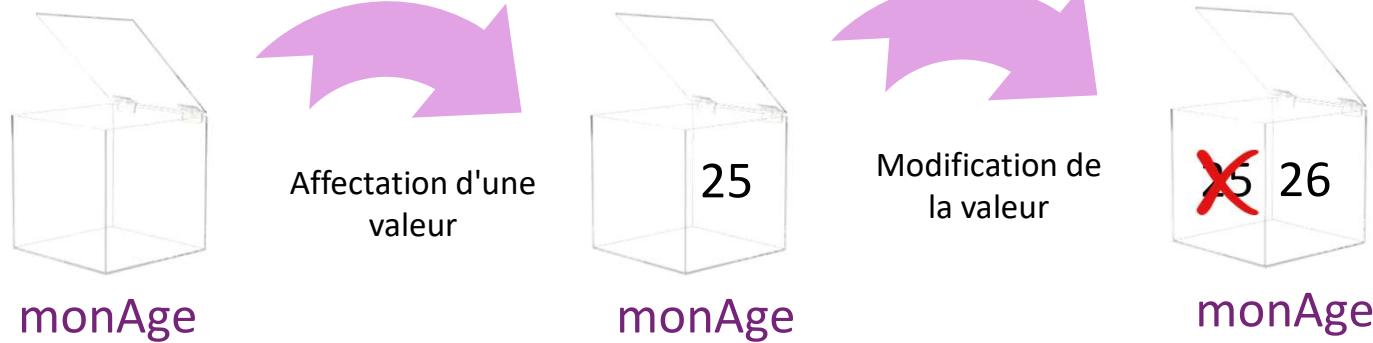
- **Snake case (snake_case) :**

Est une notation consistant à écrire l'ensemble de mots des variables généralement, en minuscules (`lower_snake_case`) ou en majuscules (`UPPER_SNAKE_CASE`) en les séparant par des tirets bas .

Exemples : <https://google.github.io/styleguide/jsguide.html>
<https://google.github.io/styleguide/pyguide.html>

Le langage (les variables) ...

En programmation, une variable est un **nom symbolique** qui est utilisé pour référencer une **valeur** ou une **donnée** laquelle est stockée dans la mémoire de l'ordinateur. Ce nom est une représentation d'une **l'adresse de la mémoire de l'ordinateur**.



Les variables sont essentielles pour manipuler et gérer les données dans un programme informatique. Elles permettent de conserver des informations qui peuvent être utilisées, modifiées et référencées tout au long de l'exécution du programme.

Le langage (les variables) ...

Les variables possèdent un certain nombre de propriétés :

- un **nom** le symbole qui permet d'identifier la déclarée la donnée ;
- un **type**, permet de définir la nature des valeurs que peut prendre la variable, ainsi que les opérateurs qui peuvent lui être appliqués (exemple : entier, chaîne de caractères,...)
- une **taille** de mot : Le type de la variable spécifie aussi la longueur de cette séquence (8 bits, 32 bits, 64 bits) ;
- une **valeur**, c'est la séquence de bits elle-même, la valeur peut être : une constante, une autre variable ou une expression.
- une **adresse**, c'est l'endroit dans la mémoire où elle est stockée ;
- une **portée**, c'est la portion de code source où elle est accessible;
- une **durée de vie**, c'est le temps d'exécution pendant laquelle la variable existe.

Dans certains langages de programmation ces propriétés sont définies lors de la déclaration

Le langage (le nom des variables) ...

Python ne possède pas de syntaxe explicite pour "déclarer" une variable, elles sont créées automatiquement au moment où on leur affecte une valeur.

- Le **nom** doit respecter les règles suivantes :
 - Commencer par une lettre ou par le caractère souligné (*underscore (_)*) ;
 - Ne doit contenir que des caractères alphanumériques courants;
 - Ne doit pas contenir des espaces;
 - Ne pas utiliser de mots "réservés" (ex : **if**, **while**, **def**, **True**,...);
 - Le nom doit être descriptif et significatif;
 - Le nom doit suivre un **style de nommage** commun à tout le code.
- Les noms sont sensibles à la casse (ex: `ma_Variable` ≠ `Ma_Variable`)

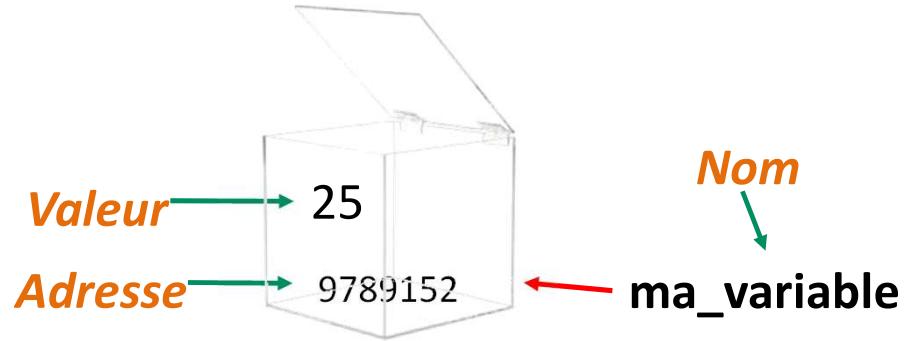
Exemple : `ma_variable = 0`

The diagram shows the Python assignment statement `ma_variable = 0`. Annotations with arrows point to specific parts:

- An arrow points to the identifier `ma_variable` with the label "nom".
- An arrow points to the assignment operator `=` with the label "Opérateur d'affectation".
- An arrow points to the value `0` with the label "valeur".

Le langage (les variables) ...

À la création d'une variable en Python, une « **carte d'identité** » lui est attribuée, cette « **carte d'identité** » permet de savoir, où la variable est gardée en mémoire et son type.



Les fonctions Python ci-dessous permettent d'afficher la « carte d'identité »

Si **ma_variable** est une variable

- **ma_variable** : permet de connaître la valeur
- **id(ma_variable)** : permet de connaître son identifiant, qui représente sa position en mémoire
- **type(ma_variable)** : permet de connaître son type

Le langage (l'affectation) ...

L'affectation est l'action d'associer une valeur à une variable.

Exemple :

`b = 1`

L'instruction d'affectation procède en deux temps :

- Le symbole "b" est le nom de la variable.
- Le symbole « = » est l'opérateur d'affectation.
- L'expression située à droite de l'opérateur d'affectation est évaluée, c'est-à-dire calculée en fonction de l'état de la mémoire à cet instant et le résultat est placé en mémoire avec un identifiant. L'expression peut correspondre à une valeur, une autre variable, une opération complexe,...
 - `b = 1 + 2`
 - `b = a`
 - `b = "exemple d'affectation"`
- Ensuite, et seulement ensuite, l'interpréteur affecte au nom situé à gauche de l'opérateur d'affectation l'objet obtenu après évaluation de l'expression de droite

Le langage (les variables) ...

Utilisez l'interprète Python en mode interactif pour créer une variable et afficher sa carte d'identité une fois que une valeur lui est affectée:



- Créer la variable nommée **ma_variable**
- Affecter la valeur 25 à la variable créée précédemment.
- Afficher la valeur de la variable
- Utiliser la fonction `id()` pour afficher l'identifiant de la variable
- Utiliser la fonction `type()` pour afficher le type de la variable

Le langage (l'affectation) ...

Python permet d'effectuer des *affectations parallèles*, c'est-à-dire affecter plusieurs valeurs à plusieurs variables dans une même instruction :

Syntaxe :

```
nom_variable_1, nom_variable_2, nom_variable_3 = valeur1, valeur2, valeur3
```

Exemples :

- `mon_prenom, mon_age, mon_numero = "Martha", 28, 16`

Le langage (le type des variables) ...

Python définit trois types de valeurs numériques, le type détermine l'ensemble de valeurs et les opérations qui peuvent être réalisées :

- Le type ***int*** qui représente tout entier positif ou négatif ;
- Le type ***float*** qui représente les nombres avec une partie décimale, compris entre 10^{-308} et 10^{308} . La valeur spéciale `math.inf` représente l'infini. Le type ***float*** permet de représenter certaines expressions scientifiques.
- Le type ***complex*** qui représente les nombres complexes.

Le **type** d'une variable en Python est déterminé par la **valeur** que lui est affectée :

Les autres types de données seront étudiés ultérieurement



Le langage (le type des variables) ...

- Le type **str** qui représente une séquence de caractères alphanumériques. Les chaînes sont des objets immuables, ce qui signifie qu'une fois qu'une chaîne est créée, elle ne peut pas être modifiée.
- Chaque caractère dans une chaîne est accessible par son index, et vous pouvez effectuer différentes opérations sur les chaînes, telles que la concaténation, la répétition, la découpe (*slicing*) et la recherche.
- Le type **str** offre de nombreuses opérations intégrées qui facilitent le traitement et la manipulation de texte.

Exemple : "J'apprends le Python!"

Les autres types de données seront étudiés ultérieurement



Le langage (le type des variables) ...

Python reconnaît certains types de variable automatiquement (entier, décimal). Par contre, pour une chaîne de caractères, il faut l'entourer de guillemets (doubles, simples, voire trois guillemets successifs doubles ou simples) afin d'indiquer à Python le début et la fin de la chaîne de caractères.

Les autres types de données seront étudiés ultérieurement



Opérations arithmétiques

Opération	Notation en Python
Addition	+
Soustraction	-
Multiplication	*
Division	/
Division entière	//
Puissance	**
Reste de la division (modulo)	%

Opérateurs d'affectation

Opérateur	Exemple	Équivalent à	Description
=	x = 1	x = 1	Affecte 1 à la variable x
+=	x += 1	x = x + 1	Ajoute 1 à la dernière valeur connue de x et affecte la nouvelle valeur à x
-=	x -= 1	x = x - 1	Enlève 1 à la dernière valeur connue de x et affecte la nouvelle valeur à x
*=	x *= 2	x = x * 2	Multiplie par 2 la dernière valeur connue de x et affecte la nouvelle valeur à x
/=	x /= 2	x = x / 2	Divise par 2 la dernière valeur connue de x et affecte la nouvelle valeur à x
%=	x %= 2	x = x % 2	Calcule le reste de la division entière de x par 2 et affecte ce reste à x
//=	x //= 2	x = x // 2	Calcule le résultat entier de la division de x par 2 et affecte ce résultat à x

Le langage (les variables) ...

Utilisez l'interprète Python en mode interactif pour :



- Affecter la valeur 25 à une variable nommée **mon_age**
- Afficher la valeur de la variable
- Afficher le type de la variable
- Affecter la valeur "Martha" à la variable **mon_age**
- Afficher la valeur de la variable
- Afficher le type de la variable

Les opérations d'entrée/sortie

Le langage (les entrée/sorties) ...

La fonction ***print()*** : est une fonction native qui affiche une ligne à l'écran, la ligne à afficher est donnée comme argument. La fonction convertit l'argument en une forme lisible (une chaîne de caractères) par l'homme si nécessaire.

Exemple :

- `print("Bonjour")`
- `print(1)`
- `print()`
- `print(ma_variable)`

Par défaut la fonction ***print()*** ajoute un saut à la ligne

Le langage (les entrée/sorties) ...

Dans les chaînes Python, l'antislash « \ » est un caractère spécial, également appelé caractère « escape ». Il est utilisé pour représenter certains caractères : " \t " est une tabulation, "\n " est une nouvelle ligne , "\r " est un retour chariot et "\\" permet d'afficher l'apostrophe

Exemple : `print("J\'apprends\tle\tPython")`

```
c:\> Invite de commandes - python
>>> print('"J\'apprends\tle\tPython"')
"J'apprends      le      Python"
>>>
```

Le langage (les entrée/sorties) ...

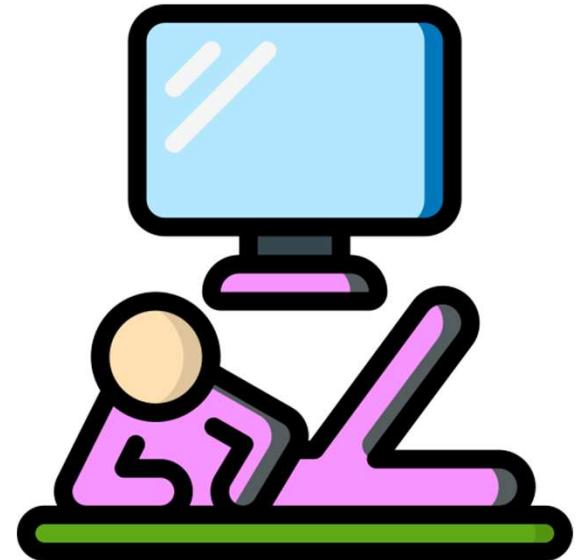
Exercice : utilisez la fonction `print()` pour afficher à l'écran.



```
"J'apprends"  
"le"  
""""Python"""
```

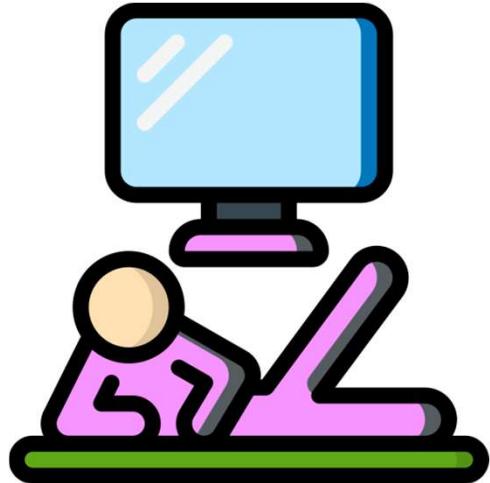
Le langage (les entrées/sorties) ...

1. utilisez la fonction **print()** pour afficher le texte ***Je suis nouveau en Python!*** à l'écran. Utilisez des guillemets doubles (" ") autour de la chaîne. Que se passe-t-il si des guillemets (' ') simples sont utilisés pour remplacer les guillemets doubles (" ") ?
2. utilisez à nouveau la fonction **print()**, mais cette fois affichez votre prénom;
3. supprimez les guillemets exécutez votre code. Regardez la réponse de Python. Quel type d'erreur est indiqué ?
4. supprimez les parenthèses, remettez les guillemets doubles et exécutez à nouveau votre code. Quel type d'erreur est indiqué cette fois-ci?
5. utilisez plusieurs fonctions **print()** sur la même ligne, puis sur des lignes différentes. Voyez ce qui se passe.
6. écrivez l'instruction **print("je suis", end = " ")**, ajoutez une autre instruction **print("nouveau")**. Quel est le résultat ?



Le langage (les entrée/sorties) ...

74



Recherchez l'utilisation de l'argument **sep** dans la fonction ***print()***.

Le langage (les entrée/sorties) ...

Corrigé :

Utilisation de l'argument **sep**;

Par défaut, la fonction **print()** affiche plusieurs objets séparés par des espaces. Cette fonction accepte optionnellement l'argument '**sep**', lequel permet d'indiquer une autre chaîne à utiliser pour séparer les éléments à afficher (dans le cas où il y a plus d'un élément).

```
# utilisation du paramètre sep
print(11, 34, 50)
print(11, 34, 50, sep=':')
print('18', '09', '2020')
print('18', '09', '2020', sep=' - ')
```

Le langage (les entrée/sorties) ...

Challenge :



Modifier le code pour afficher une chaine deux fois plus large et deux fois plus haute (Atn : garder les proportions).

Le langage (les entrée/sorties) ...

Écriture formatée

L'écriture formatée est un mécanisme permettant d'afficher des variables avec un certain format.
Python a introduit les ***f-strings*** pour mettre en place l'écriture formatée
f-string est la contraction de *formatted string literals*.

Exemple :

```
x = 25
nom = "Martha"
print(f"{nom} a {x} ans")
```

La chaîne est précédée de la lettre **f**

Les variables sont entourées par des {}

Résultat : **Martha a 25 ans**

Les variables entre les {} sont remplacées par leur valeur.

Il ne faut pas omettre le **f** avant le premier guillemet, sinon Python prendra cela pour une chaîne normal de caractères et ne mettra pas en place ce mécanisme de remplacement



Le langage (les entrée/sorties) ...

Exemple : il est possible d'indiquer combien de décimales seront affichés

```
prop = (4500 + 2575) / 14800
print("La proportion de GC est", prop)
print(f"La proportion de GC est {prop:.2f}")
print(f"La proportion de GC est {prop:.3f}")
```



Indique la quantité de décimales

Résultat :

```
La proportion de GC est 0.4780405405405405
La proportion de GC est 0.48
La proportion de GC est 0.478
```

Le langage (les entrée/sorties) ...

Il est possible de préciser sur combien de caractères le résultat doit être écrit et comment se fait l'alignement (à gauche (<), à droite (>) ou centré (^)).

Exemple :

```
print(f"{10:>6d}") ; print(f"{1000:>6d}")
```

quantité de caractères
(6), alignés à droite (>)

```
10  
1000
```

```
print(f"{10:<6d}") ; print(f"{1000:<6d}")
```

quantité de caractères (6),
alignés à gauche (<)

```
10  
1000
```

```
print(f"{10:^6d}"); print(f"{1000:^6d}")
```

quantité de caractères
(6), alignés centré (^)

```
10  
1000
```

Le langage (les entrée/sorties) ...

Il est possible également de préciser le nombre de décimales et l'alignement (à gauche (<), à droite (>) ou centré (^)).

Exemple :

```
# pour indiquer le nombre de décimales et l'alignement
prop = (4500 + 2575) / 14800
print(f"La proportion de GC est {prop:<5.2f}")
print(f"La proportion de GC est {prop:<5.3f}")
```

quantité de caractères (5),
alignés à gauche (<)

Places décimales

```
La proportion de GC est 0.48
La proportion de GC est 0.478
```

Le langage (les entrée/sorties) ...

Le formatage est également possible sur des chaînes de caractères en utilisant la lettre **s** (comme string)

Exemple :

```
print(f"mois {'janvier':>8s}") ; print(f"mois {'décembre':>8s}")
```

Résultat

```
mois janvier
mois décembre
```

Le langage (les entrée/sorties) ...

La fonction ***input()*** : est capable de lire les données saisies par l'utilisateur et de renvoyer les mêmes données au programme en cours d'exécution.



- Cette fonction provoque une interruption dans l'exécution du programme courant.
- L'utilisateur est invité à entrer des caractères au clavier et à terminer en appuyant sur la touche <Enter>. Ensuite l'exécution du programme se poursuit
- Si la fonction ***input()*** est appelée sans arguments; la fonction basculera la console en mode d'entrée et attend la saisie des données, jusqu'à ce que la touche Entrée est appuyée; toutes les données saisies seront envoyées à votre programme via le résultat de la fonction.
 - `mon_age = input()`
- Pour consulter la valeur de la variable ***mon_age*** la fonction ***print(mon_age)*** peut être utilisée.

Le langage (les entrée/sorties) ...

La fonction ***input()*** : est capable de lire les données saisies par l'utilisateur et de renvoyer les mêmes données au programme en cours d'exécution.



- Si la fonction ***input()*** est appelée avec un argument - il s'agit d'une chaîne contenant un message ; le message sera affiché sur la console avant que l'utilisateur n'ait la possibilité d'entrer quoi que ce soit.
➤ **mon_age = input("Saisir votre âge: ")**
- Pour consulter la valeur de la variable **mon_age** la fonction **print(mon_age)** peut être utilisée.

Le langage (les entrée/sorties) ...

La fonction ***input()*** : est capable de lire les données saisies par l'utilisateur et de renvoyer les mêmes données au programme en cours d'exécution.



- Pour saisir une chaîne de caractères l'utilisateur doit le faire avec les guillemets inclus
 - `mon_nom = input("Saisir votre nom complet : ")`
- Pour consulter la valeur de la variable `mon_nom` la fonction `print(mon_nom)` peut être utilisée.

Le langage (les entrée/sorties) ...

La fonction ***input()*** : est capable de lire les données saisies par l'utilisateur et de renvoyer les mêmes données au programme en cours d'exécution.

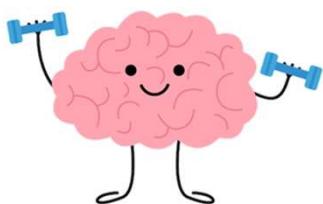


- La fonction `input()` renvoi toujours une chaîne de caractères (**string**)
 - `mon_nom = input("Saisir votre nom complet : ")`
- Pour consulter le type d'une variable la fonction **`type()`**, peut être utilisée
 - `type(mon_nom)`

Le langage (les entrée/sorties) ...

La fonction ***input()*** lit les données saisies par l'utilisateur et les envoie au programme en cours d'exécution.

Écrivez et testez le programme ci-dessous et répondez les questions :



```
maVariable = input("Saisissez un chiffre : ")
monResultat = maVariable ** 2
print(maVariable, "au carré est :", monResultat)
```

- Quel est le résultat affiché ?
- Quelle explication pouvez-vous donner ?



Dans cet exemple la convention de nommage utilisée est ***camelCase***

Le langage (Conversion de types) ...

Python offre deux fonctions simples pour convertir le type de données de la donnée d'entrée et résoudre ce problème :

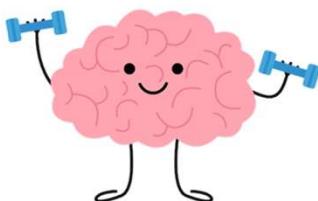
- la fonction **int()** prend un argument en entrée et **tente** de le convertir en entier.
 - **Exemple** : `int(ma_variable)`
- la fonction **float()** prend un argument en entrée et **tente** de le convertir en décimal.
 - **Exemple** : `float(ma_variable)`.

La structure d'un programme ...

Les racines du polynôme de deuxième dégrée

$$A x^2 + B x + C$$

Peuvent être calculées



- $x = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$
- $\Delta = B^2 - 4AC$ (Δ est appelé le discriminant « delta »)
- $x_1 = (-B + \sqrt{\Delta})/2A$
- $x_2 = (-B - \sqrt{\Delta})/2A$
- Écrire l'algorithme pour calculer les racines x_1, x_2 du trinôme
- Investiguer quelle est la fonction Python permettant de calculer la racine carrée d'un nombre.
- Traduire l'algorithme en Python
- Choisissez des données de tests et testez le script

Le langage (instructions simples) ...

- Python supporte des instructions sur une même ligne à condition qu'elles soient séparées par des ";".

Exemple :

```
print(x); print(y); print(z)
```

Exercice ...

Algorithmes & Fonctions

Réaliser en Python les programmes suivants :

Situation 1 : : On donne un nombre, l'ordinateur affiche son image par la fonction $f : x \mapsto 2x - 6$. (Tester le programme pour les valeurs 0, 1, 2, -1 et -2.)

Situation 2 : : Améliorer le programme de l'exercice 1 pour que l'ordinateur affiche « Bravo ! Vous êtes un as du calcul mental ! » si l'image obtenue vaut 4. (Trouver la valeur qui affiche ce message.)

Situation 3 : : Modifier le programme précédent pour réaliser la même chose mais avec la fonction $g : x \mapsto \frac{x+1}{x-2}$. (Tester le programme pour les valeurs 0, 1, 2, -1 et -2 et trouver la valeur de x qui affiche le message de félicitations.)

Situation 4 : : Modifier le programme de la situation 2 pour réaliser la même chose mais avec la fonction $h : x \mapsto \sqrt{3x+1}$. (Tester le programme pour les valeurs 0, 1, 2, -1 et -2 et trouver la valeur de x qui affiche le message.)

Les structures conditionnelles

Structures de contrôle...

Conditionnel

97



*Quelle est la définition de la **structure conditionnelle** et dans quel contexte elle est utilisée ?*

vidéo : <https://youtu.be/fVUL-vzrlcM>

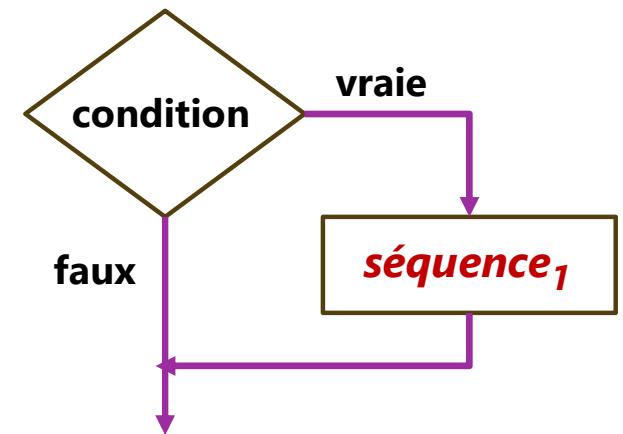
Structures de contrôle...

Conditionnel

98

.... permet d'exécuter une séquence d'instructions, seulement si une **condition** est satisfaite.

- Forme 1 : Si **condition** alors ...
- La **condition** est une expression **booléenne** :
 - la condition est **évaluée**;
 - si la condition est **vraie**, la séquence (**séquence₁**) est exécutée puis le contrôle passe à la suite;
 - si la condition est **fausse**, le contrôle passe à la suite, sans exécuter la (**séquence₁**).



Structures de contrôle...

Conditionnel

99

- Une expression **booléenne** est une expression qui évalue à une valeur booléenne, c'est-à-dire à soit **vrai** soit **faux**.
- Une expression booléenne peut être construite en utilisant des opérateurs de comparaison, des opérateurs logiques et des valeurs booléennes directes.
- Les expressions booléennes jouent un rôle fondamental dans les tests conditionnels, qui permettent à un programme de prendre des décisions en fonction de certaines conditions spécifiées. Elles permettent de contrôler l'exécution du code en fonction de la véracité ou de la fausseté de certaines conditions.

Opérateurs de comparaison ...

.... permettent d'exprimer des expressions booléennes, sont utilisés pour l'écriture des conditions.

Expression (pour vérifier)	Opérateur	Exemple
égalité	<code>==</code>	<code>X == 2</code>
différent de	<code>!=</code>	<code>X != 2</code>
strictement inférieur à	<code><</code>	<code>X < 2</code>
inférieur ou égal à	<code><=</code>	<code>X <= 2</code>
strictement supérieur à	<code>></code>	<code>X > 2</code>
supérieur ou égal à	<code>>=</code>	<code>X >= 2</code>

Utilisation des opérateurs (A, B et Delta sont des variables de type numérique) :

`if A <= B :`

`if Delta == 0`

Structures de contrôle...

Conditionnel

101

En **Python** : Une conditionnelle est écrite à l'aide du mot clé **if**.

Syntaxe :

1. Si ... Alors

if condition :

Instruction A



Condition est une expression booléenne, évaluée à **True** ou **False**

Chaque en-tête de clause commence par un mot-clé spécifique et se termine par le caractère deux-points (:)

*En Python, un bloc est défini par une **indentation** obtenue en **décalant** le début des instructions vers la droite grâce à des espaces en début de ligne .*

Toutes les instructions d'un même bloc doivent être indentées exactement au même niveau.

Les instructions peuvent être écrites sur la même ligne que l'en-tête si elles sont simples, dans ce cas elles doivent être séparées par des ";"

Un bloc peut contenir une ou plusieurs instructions, et notamment des instructions composées.

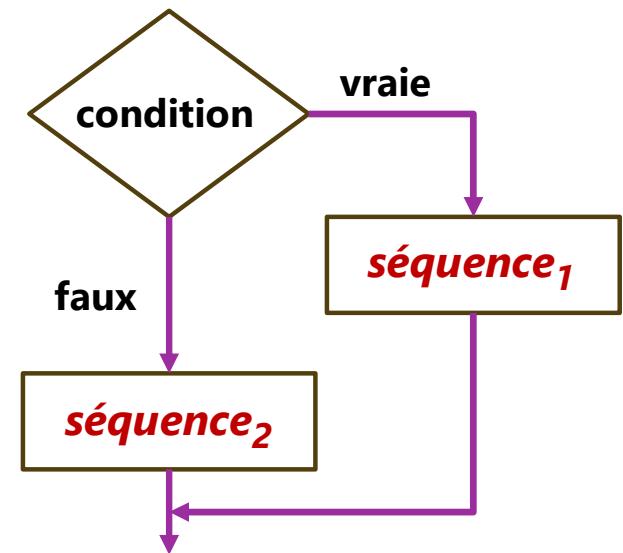
Structures de contrôle...

Conditionnel

102

.... permet d'exécuter une séquence d'instructions, seulement si une **condition** est satisfaite.

- Forme 2 : Si **condition** alors... sinon
- La **condition** est une expression **booléenne** :
 - la condition est **évaluée**;
 - si la condition est **vraie**, la séquence (**séquence₁**) est exécutée puis le contrôle passe à la suite de la **séquence₂**;
 - si la condition est **fausse**, le contrôle passe à la suite (**séquence₂**) , sans exécuter la (**séquence₁**).



Structures de contrôle...

Conditionnel

103

En **Python** : Une conditionnelle est écrite à l'aide des mots clés **if else**.

Syntaxe :

1. Si...Alors... Sinon

if condition :

 Instruction A

else :

 Instruction B



Condition est une expression booléenne, évaluée à **True ou False**

Chaque en-tête de clause commence par un mot-clé spécifique et se termine par le caractère deux-points (:) .

*En Python, un bloc est défini par une **indentation** obtenue en **décalant** le début des instructions vers la droite grâce à des espaces en début de ligne .*

Toutes les instructions d'un même bloc doivent être indentées exactement au même niveau.

Les instructions peuvent être écrites sur la même ligne que l'en-tête si elles sont simples, dans ce cas elles doivent être séparées par des ";"

Un bloc peut contenir une ou plusieurs instructions, et notamment des instructions composées.

Structures de contrôle...

Conditionnel

104

En **Python** : Une conditionnelle est écrite à l'aide des mots clés **if else**. Il est possible d'imbriquer plusieurs conditions en ajoutant le mot clé **elif**, contraction de "**else**" et "**if**".

Syntaxe :

1. Si...Alors... Sinon

if condition1 :

 Instruction A

elif condition2 :

 Instruction B

else :

 Instruction C

Exemple :

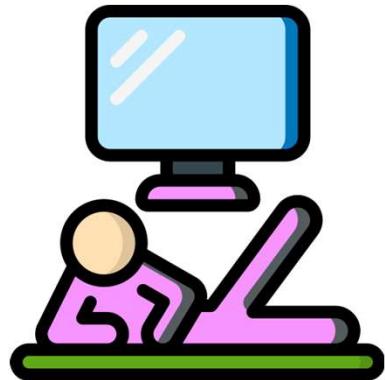
```
note = float(input("Saisi ta note : "))
if note == 20 :
    print ("Tu es un génie")
elif note > 15 :
    print("Très Bien !")
elif note > 13 :
    print( "Bien")
elif note > 11 :
    print( "Assez Bien")
else :
    print("Tu devras recopier 3 fois toutes les décimales de PI")
```



L'indentation et les ":"

Instruction conditionnelle ...

105



Écrire un programme en Python qui demande l'âge d'un enfant à l'utilisateur. Ensuite, il l'informe de sa catégorie :

- « Poussin » de 6 à 7 ans
- « Pupille » de 8 à 9 ans
- « Minime » de 10 à 11 ans
- « Cadet » après 12 ans

Instruction conditionnelle ...

106



Écrire un programme en Python qui permette

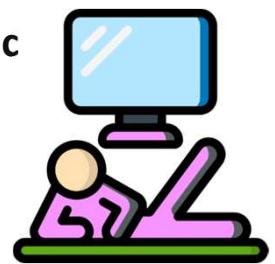
1. Entrer deux nombres.
2. Choisir une opération (+, -, *, /).
3. Selon l'opération choisie, effectuer le calcul correspondant.
4. Gérer le cas spécial de la division par zéro.
5. Afficher le résultat.

Instruction conditionnelle ...

Pour respecter la norme **RT2012** des maisons BBC (Bâtiments Basse Consommation), il faut que la résistance thermique des murs notée **R** soit supérieure ou égale à 4. Pour calculer cette résistance thermique, on utilise la relation : $R = e/c$ où **e** désigne l'épaisseur de l'isolant en mètre et **c** désigne le coefficient de conductivité thermique de l'isolant. Ce coefficient permet de connaître la performance de l'isolant.

Le prestataire a choisi comme isolant la laine de verre dont le coefficient de conductivité thermique est : **c = 0,035**. Il souhaite mettre 15 cm de laine de verre sur ses murs. La maison respecte-t-elle la norme RT2012 ?

- Écrire un programme Python, qui affiche "**norme respectée**" si la norme est respectée et "**norme non respectée**" sinon.
- Modifiez le programme afin de permettre à l'utilisateur de saisir les valeurs pour **e** et **c**



Instruction conditionnelle ...

108



Écrire un programme en Python qui calcule le nombre de radiateurs nécessaires pour chauffer une pièce.

On sait qu'un radiateur est capable de chauffer 8 m³. L'utilisateur donnera la **longueur**, la **largeur** et la **hauteur** de la pièce (en mètres).

Instruction conditionnelle ...

110



Un magasin de reprographie facture 0,10 € les dix premières photocopies, 0,09 € les vingt suivantes et 0,08 € au-delà.

Écrire un programme en Python qui demande à l'utilisateur le nombre de copies et qui calcule le prix à payer

Expressions booléennes...

Opérateurs logiques 112

Il est possible de combiner plusieurs conditions

Exemple : A <= B and B <= C

- **La conjonction** : la condition est vérifiée si toutes les sous-conditions sont vérifiées à leur tour

- Table de vérité :

		Expression 1	
		VRAI	FAUX
ET/and		VRAI	FAUX
Expression 2	VRAI	VRAI	FAUX
	FAUX	FAUX	FAUX

Une table de vérité donne la valeur d'une ou de plusieurs expressions booléennes, construite à partir de variables et d'opérateurs booléens, en fonction des valeurs des variables booléennes.

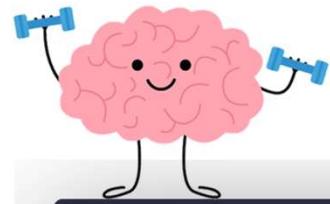
Instruction conditionnelle ...

Indiquer quelle est la fonctionnalité de ce programme



```
A = int(input("Saisissez une valeur pour A : "))
B = int(input("Saisissez une valeur pour B : "))
C = int(input("Saisissez une valeur pour C : "))
print("Valeurs rentrées : A = ", A, " B = ", B, " C = ", C)
if (A > B) and (B >= C):
    print("A est le supérieur ")
elif (B > A) and (A >= C) :
    print("B est le supérieur ")
elif (C > A) and (A >= B) :
    print ("C est le supérieur ")
else :
    print ("Les trois valeurs sont égaux ")
```

Instruction conditionnelle ...



```
A = int(input("Saisissez une valeur pour A : "))
B = int(input("Saisissez une valeur pour B : "))
C = int(input("Saisissez une valeur pour C : "))
print("Valeurs rentrées : A = ", A, " B = ", B, " C = ", C)
if (A > B) and (B >= C):
    print("A est le supérieur ")
elif (B > A) and (A >= C) :
    print("B est le supérieur ")
elif (C > A) and (A >= B) :
    print ("C est le supérieur ")
else :
    print ("Les trois valeurs sont égaux ")
```

- Testez le programme montré à l'image avec les données :
 - 12 12 8
 - 12 8 12
 - 8 12 12
- Quels sont les résultats obtenus ?
- Sont-ils corrects ?
- Réalisez les modifications le cas échéant

Instruction conditionnelle ...

Compléter le programme ci-dessous :

```
A = int(input("Saisissez une valeur pour A : "))
B = int(input("Saisissez une valeur pour B : "))
C = int(input("Saisissez une valeur pour C : "))
print("Valeurs rentrées : A = ", A, " B = ", B, " C = ", C)
if (A <= B) and (B <= C):
    print("la séquence est ordonnée de façon ascendante : ")
elif [REDACTED] :
    print("la séquence est ordonnée de façon descendante : ")
else :
    print ("la séquence n'est pas ordonnée : ")
```



Instruction conditionnelle ...

Il est possible de combiner plusieurs conditions

- **La disjonction** : la condition est vérifiée si au moins une des sous-conditions est vérifiée

Exemple : A <= B or B <= C

- Table de vérité :

		Expression 1	
		OU/or	VRAI
Expression 2	VRAI	VRAI	VRAI
	FAUX	VRAI	FAUX

Une table de vérité donne la valeur d'une ou de plusieurs expressions booléennes, construite à partir de variables et d'opérateurs booléens, en fonction des valeurs des variables booléennes.

Instruction conditionnelle ...

Réécrivez le programme ci-dessous pour utiliser les opérateurs logiques (**and** et **or**). Testez-le, réalisez les améliorations où c'est possible

```
note = int(input("Saisie ta note : "))
if note == 20 :
    print ('tu es un génie')
elif note > 15 :
    print( "Très Bien ! ")
elif note > 13 :
    print("Bien")
elif note > 11 :
    print( "Assez Bien")
else :
    print("Tu devras recopier 3 fois toutes les décimales de pi ! ")
```



Envoyez un fichier texte avec votre code à m.gonzalez-garcia@aforp.fr

Instruction conditionnelle ...

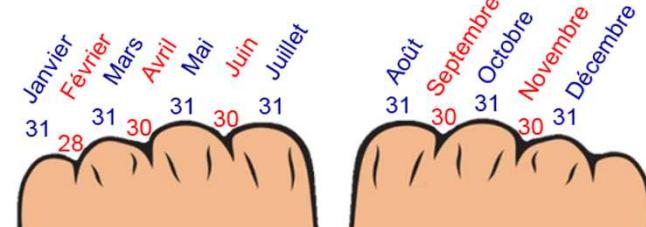
Compléter le programme ci-dessous pour afficher le nombre de jours du mois dont le numéro est saisi en entrée (sans tenir compte des années bissextiles).

```
mois = int(input("Saisir le numéro du mois : "))
if mois == 1

    Jours = 31
elif mois == 4
    Jours = 30
else :
    Jours = 28
print ("Le nombre de jours du mois de", mois, "est :",
Jours)
```



*Si l'utilisateur introduit le chiffre **3** qui représente le mois de **mars**, l'algorithme doit afficher **31**, qui correspond au nombre de jours du mois de **mars***

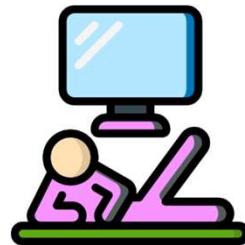


Crédit image : <https://www.l instit.com/lecon-mathematiques-mesures-durees-an-mois-jour.html>

Envoyez un fichier texte avec votre code à m.gonzalez-garcia@aforp.fr

Instruction conditionnelle ...

Testez le programme de l'exercice précédent avec les valeurs :



1. mois = 0
 2. mois = 20
- Que se passe-t-il ?
 - Apporter les corrections nécessaires

Envoyez un fichier texte avec votre code à m.gonzalez-garcia@aforp.fr

Instruction conditionnelle ...

un chiffre "A" est divisible par un chiffre "B" si le **reste** (% en Python) de la division entière (// en Python) est égal à "0"

Une année est bissextile si elle est :

- divisible par 4 et non divisible par 100
- ou
- divisible par 400 dans le cas du début du siècle.



Exemple :

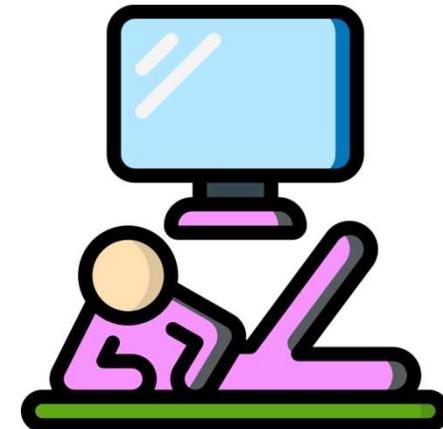
- l'année 2000 était bissextile : 2000 débute un siècle ($2000/400 = 5$);
- l'année 2100 ne sera pas bissextile.

Écrire en Python la condition permettant d'indiquer si une année est **bissextile**

Envoyez un fichier texte avec votre code à m.gonzalez-garcia@aforp.fr

Instruction conditionnelle ...

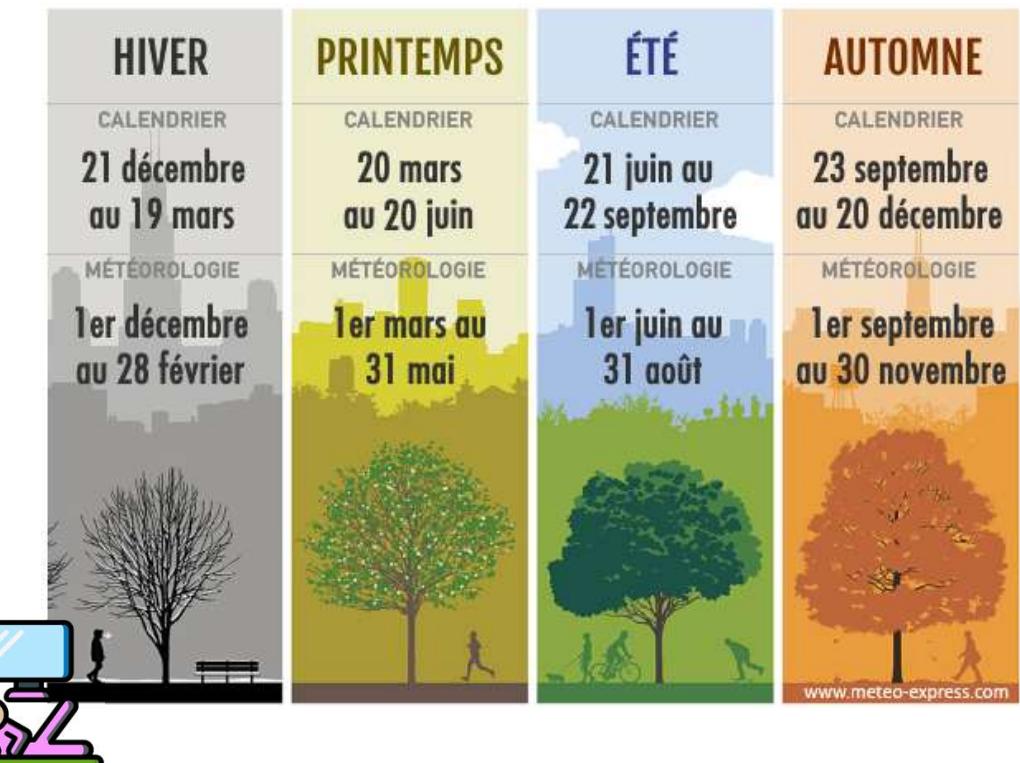
Modifiez le programme de l'exercice précédent pour considérer les années **bissextiles**



Instruction conditionnelle ...

Écrire un programme en Python qui demande une date sous la forme de **2 nombres entiers** (le premier nombre étant le jour et le deuxième le mois) et affiche le nom de la saison (ex : si l'utilisateur saisit 12 et puis 02, le programme doit afficher "hiver").

On supposera que le premier jour de la saison est toujours le **21**.



Credit image : <https://meteo-express.com/article/automne-meteo-ce-jeudi-quelles-differences-avec-le-calendrier>

Instruction conditionnelle ...



Écrire un programme en Python qui demande une date sous la forme de **2 nombres entiers** (le premier nombre étant le jour et le deuxième le mois) et affiche le nom de la saison (ex : si l'utilisateur saisit 12 et puis 02, le programme doit afficher "hiver").

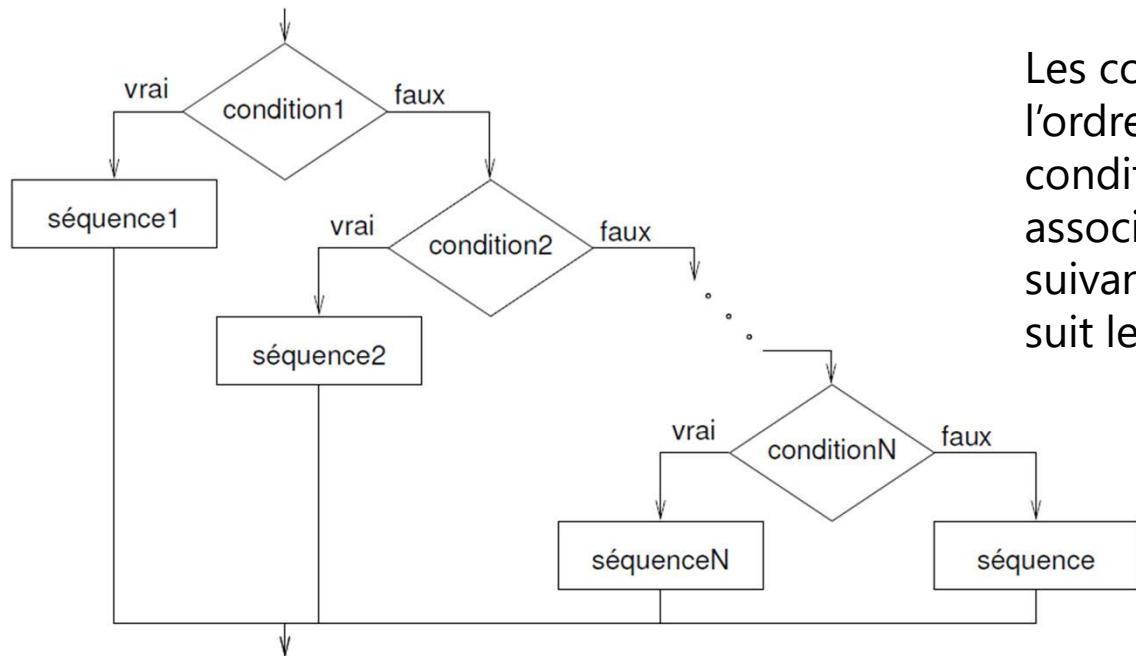
On supposera que le premier jour de la saison est toujours le **21**.

Exemple de la condition en utilisant la syntaxe d'AlgoBox

```
▼ SI ((MOIS==12 ET JOUR >=21) OU MOIS == 1 OU MOIS ==2 OU (MOIS ==3 ET JOUR <21)) ALORS
  DEBUT_SI
    AFFICHER "C'est l'hiver, il fait froid"
  FIN_SI
```

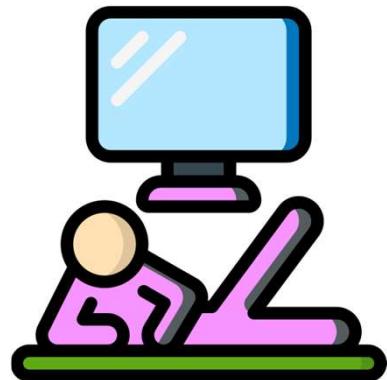
Instruction conditionnelle ...

Les conditions peuvent être imbriquées



Les conditions sont évaluées dans l'ordre d'apparition. Dès qu'une condition est **vraie**, la séquence associée est exécutée. L'instruction suivante à exécuter sera alors celle qui suit le **FinSi**

Instruction conditionnelle ...



Écrire un programme en Python qui pour trois valeurs réelles a , b et c , détermine si a , b et c peuvent être les côtés d'un triangle ; et si le triangle en question est un **triangle rectangle**, un triangle **isocèle** ou un autre triangle.

On suppose que **c** représente le côté le plus long du triangle.

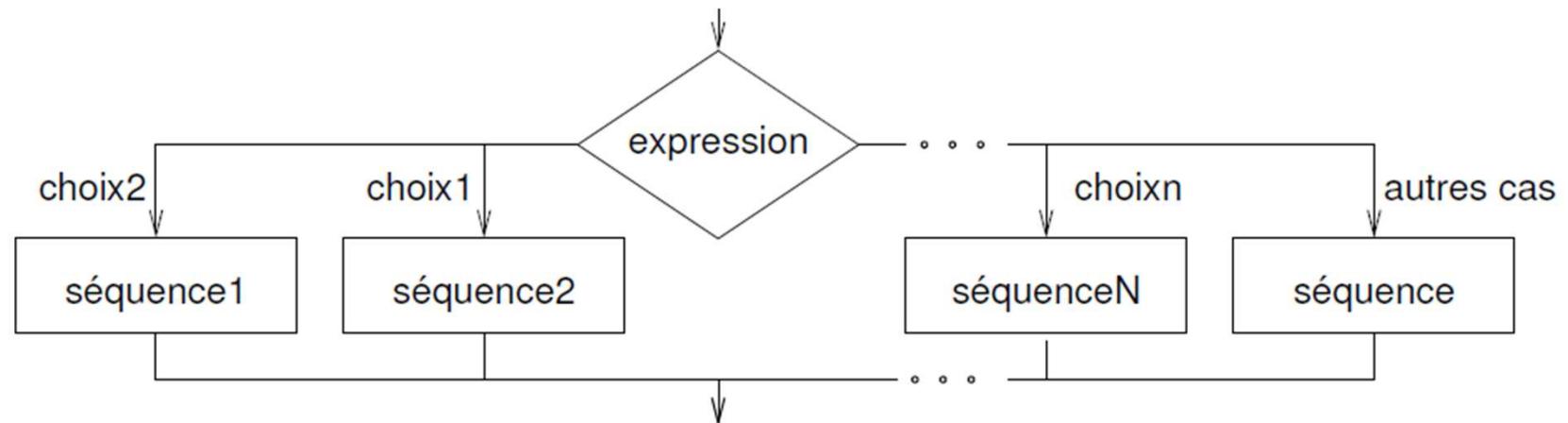
Pour cet exercice, certains des résultats géométriques suivants peuvent être utiles :

- si $a + b \geq c$, les longueurs correspondent à un triangle
- si $a^2 + b^2 = c^2$, le triangle est rectangle
- si $a^2 + b^2 = c^2$ et $a = b$, le triangle est isocèle et rectangle

Instruction conditionnelle...

match ◆ 135

L'instruction **match** permet de faire plusieurs tests de valeurs sur le contenu d'une **même expression**



Instruction conditionnelle...

match  136

L'instruction **match** permet de faire plusieurs tests de valeurs sur le contenu d'une **même expression**. L'expression (**jour**) est évaluée selon le patron (après le mot clé **case**)

```
match jour :  
    case "lundi" : print(2)  
    case "mardi" : print(3)  
    case "mercredi" : print(4)  
    case "jeudi" : print(5)  
    case "vendredi" : print(6)  
    case "samedi" : print(7)  
    case "dimanche" : print(1)  
    case _: print("erreur")
```



Disponible à partir de la version 3.10

Structure conditionnelle...

match ...

137



Réécrivez l'exercice de l'année bissextile en utilisant la structure "match"



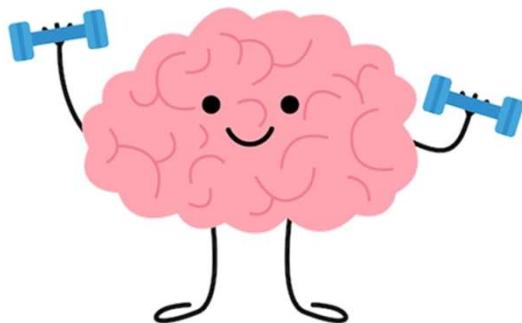
Disponible à partir de la version 3.10

Instruction conditionnelle ...

Calcul des frais de déplacement selon plusieurs tarifs.

Le calcul des frais de déplacement est fonction de la puissance du véhicule et du nombre de km parcourus.

- Si le nombre de km est <= à 100 , **frais de déplacement = 0**
- Si le nombre de km est > à 100 , le tarif à appliquer est fonction de la puissance du véhicule :



Puissance fiscale	Tarif
1 à 3	Tarif : 1
4 à 6	Tarif : 1,5
7 à 8	Tarif : 2
9 à 12	Tarif : 2,5
>12	Tarif : 3

Calcul des frais = Nombre de km * tarif

Les boucles

Structures de contrôle itératives...

... Les itérations permettent d'exécuter une même séquence d'instructions plusieurs fois .



*Il est important de garantir que la répétition se termine effectivement
vidéo : <https://youtu.be/jsUN0NV5RfQ>*



Structures de contrôle itératives...

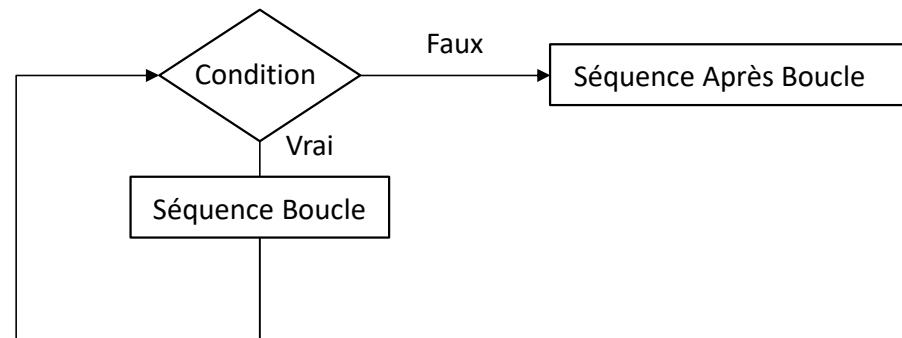
boucle for

143

... Les itérations permettent d'exécuter plusieurs fois une même séquence d'instructions.



Le nombre maximal d'itérations est connu



Chaque élément de fournit par l'itérateur

Son évaluation produit un objet itérable et un itérateur

est exécutée pour chaque élément de la target list

```
"for" target list "in" expression list ":" suite  
["else" ":" suite]
```

optionnel

est exécutée quand la boucle se termine

Structures de contrôle itératives... boucle for

144

Exemple 1 : utilisation de la fonction `range()`

crée une séquence de nombres, à partir de 0 par défaut, et incrémenté de 1 (par défaut), et se termine à un nombre spécifié et renvoie un itérateur approprié sur ces entiers

```
for x in range(6): print(x)
```

Dans l'exemple l'objet produit est une séquence
(0, 1, 2, 3, 4, 5)

Structures de contrôle itératives...

boucle for

145

Exemple 2 : utilisation de la fonction **range()**

```
for x in range(2, 6): print(x)
```

```
for x in range(2, 6, 2): print(x)
```

Indique
l'incrément.

```
for x in range(6, 2, -2): print(x)
```

Structures de contrôle itératives...

boucle for

146

Exemple 3 : La boucle **for** effectue des affectations aux variables de la liste cible, ce qui écrase toutes les affectations antérieures de ces variables

```
for i in range(10):
    print(i)
    i = 5
```

Quelles valeurs sont affichées?

Structures de contrôle itératives...

boucle for

Exemple 4 : La boucle **for** ne nécessite pas de variable d'indexation à définir au préalable.

```
jours = ["Dimanche", "Lundi", "Mardi", "Mercredi", "Jeudi", "Vendredi", "Samedi"]
for x in jours:
    print(x)
```

Évalue la liste et produit un itérateur

À chaque itération l'itérateur renvoi un élément qui est assigné à la variable **x**

Exemple 5 : La boucle **for** permet d'itérer sur un chaîne

```
for x in "Dimanche":
    print(x)
```

Structures de contrôle itératives...

boucle for

148



Additionner les **N** premiers nombres positifs.

Structures de contrôle itératives...

boucle for

149

```
# Programme 1

for i in range(100):
    for i in range(100):
        print("*")
```

```
# Programme 2
```

```
for i in range(100):
    print("*")
for i in range(100):
    print("*")
```

< Introduction Python

Lorsque le sondage est actif, répondez à PollEv.com /marthagonzalez201



Quel programme affiche plus de "*" ?

Programme	Percentage
Programme 1	0%
Programme 2	0%

Powered by  Poll Everywhere

Structures de contrôle itératives...

Une légende de l'Inde ancienne raconte que le jeu d'échecs a été inventé par un vieux sage, que son roi voulu remercier en lui affirmant qu'il lui accorderait n'importe quel cadeau en récompense. Le vieux sage demanda qu'on lui fournisse simplement un peu de riz pour ses vieux jours, et plus précisément le nombre de grains de riz suffisant pour qu'on puisse déposer :

- 1 grain sur la première case
- 2 grains sur la deuxième case
- 4 grains sur la troisième case
- 8 grains sur la quatrième case ...

Et ainsi de suite (toujours deux fois plus) jusqu'à la 64^{ème} case.

1. Écrivez un programme en Python qui affiche le nombre exact (entier) de grains à déposer sur chaque case du jeu.
2. Modifiez le pour faire calculer le nombre total de grains de riz.



Structures de contrôle itératives...

boucle while

152

... permet de répéter une série d'instructions tant qu'un *objectif soit atteint*

Est utilisée quand le nombre d'itérations n'est pas connu en avance

- **Explication :**

- La condition (assignment expression) est **évaluée avant** l'entrée dans la boucle
- Le bloc d'instructions (suite1) est exécuté, **si la condition est vérifiée**
- Si la condition du est fausse dès le début, le bloc suite1 n'est **jamais** exécuté
- Quand la condition est **fausse** la boucle **termine**

Syntaxe

```
"while" assignment expression ":" suite1
["else" ":" suite]
```

optionnel

est exécutée quand la boucle se termine



Il est indispensable de garantir que la condition d'arrêt soit vérifiée à un moment ou à un autre, sans quoi le programme va rentrer dans une «boucle infinie»

Structures de contrôle itératives...

boucle while

153

Exemple :

```
i = 1
while i < 6 :
    print(i)
    i += 1
else:
    print("la boucle est finie i est égal à" , i)
```



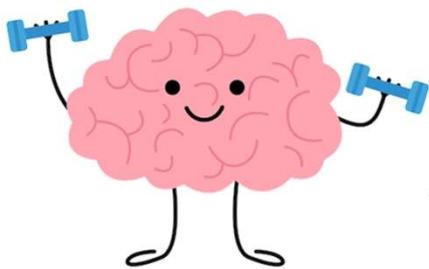
*Il est indispensable de s'assurer que la condition d'arrêt soit vérifiée
à un moment ou à un autre, sans quoi le programme va rentrer
dans une «boucle infinie»*

Structures de contrôle...

boucle while

156

Écrire des programmes python pour :



1. Un escargot veut monter au sommet d'un mur de 10 m. de hauteur. Pendant la journée il avance de 3 m. Mais pendant la nuit il recule de 2 m. Combien de jours il nécessite pour y arriver ?
2. Écrire un algorithme qui demande à l'utilisateur la saisie d'une suite de nombres, et qui lui dise ensuite quel était le plus grand parmi ces nombres. La saisie des nombres s'arrête lorsque l'utilisateur entre un zéro.
3. Modifiez l'algorithme précédent pour que le programme affiche également dans quelle position avait été saisi le plus grand nombre : « C'était le nombre n° 2 »

Quelle boucle choisir ?

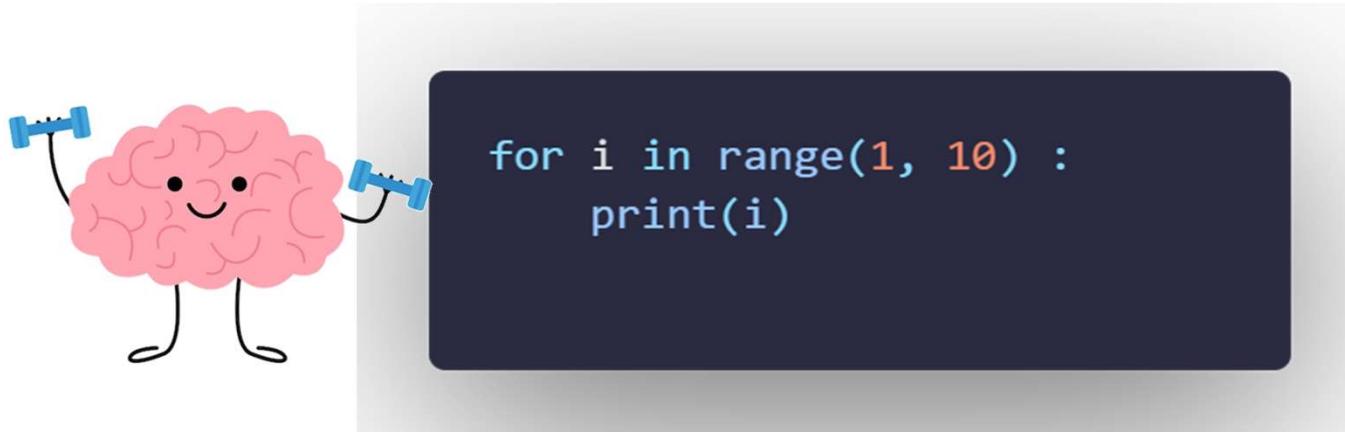
157

Le choix du type de boucle à utiliser dans un problème se fait de la manière suivante :

- Si le nombre d'itérations à effectuer dans la boucle est connu, on utilisera une boucle **for**
- Si la poursuite dans la boucle est dépendante d'une condition, on utilisera de préférence une boucle **while**

Structures de contrôle itératives...

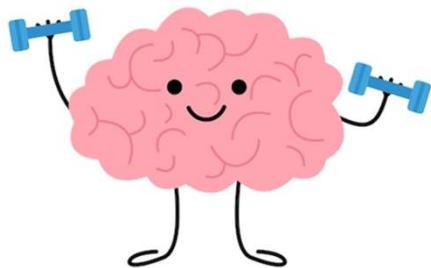
Réécrivez l'algorithme ci-dessous en utilisant la boucle **while**



Est-il possible de toujours transformer la boucle du type **while** en une boucle **for** ?

Structures de contrôle itératives...

159



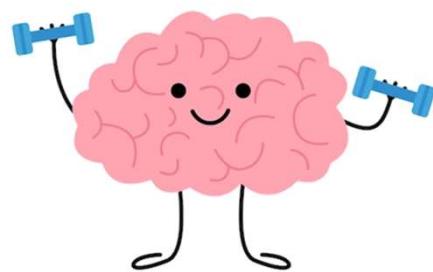
On souhaite réaliser la saisie d'un numéro qui corresponde à un mois (compris entre 1 et 12) avec vérification.

Si la saisie est incorrecte, le programme affiche un message expliquant l'erreur et demande à l'utilisateur de resaisir la donnée.

Écrire le programme en Python pour résoudre le problème

Structures de contrôle itératives...

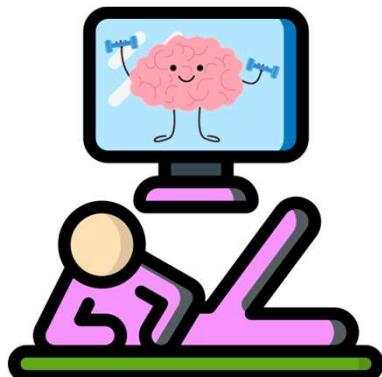
Écrire des programmes en Python pour :



Demande à l'utilisateur d'indiquer un nombre de départ, et qui ensuite affiche les dix nombres suivants. Par exemple, si l'utilisateur entre le nombre 17, le programme affichera les nombres de 18 à 27.

Structures de contrôle itératives...

Écrire des programmes en Python pour :



1. Demander à l'utilisateur de saisir 10 entiers et compter la quantité d'entiers positifs, et afficher ce résultat.
2. Demander à l'utilisateur de saisir une séquence d'entiers, additionner les positifs, et arrêter dès qu'un entier négatif est saisi en affichant le résultat.
3. Modifier ce dernier algorithme pour afficher la moyenne de la série d'entiers positifs saisis.
4. Calculer la factorielle de **N** (**N!**)

On rappelle que :

$$0! = 1$$

$$N! = 1 \times 2 \times 3 \times \dots \times N$$

Structures de contrôle itératives...

162

Écrire des programmes en Python pour :

1. Demande des nombres à l'utilisateur et qui indique, à la fin de la saisie, le nombre le plus grand et son rang. L'utilisateur met fin à la saisie en entrant la valeur zéro.

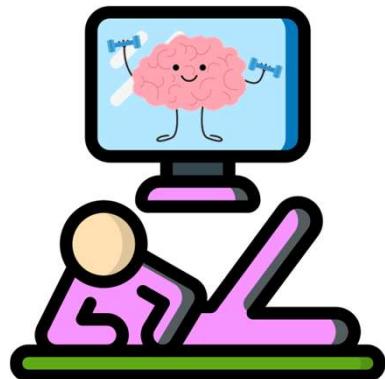
Par exemple, si l'utilisateur entre les nombres suivants:

6 8.5 52.3 95 23.8 0

L'algorithme affichera :

Le nombre le plus grand est 95, situé au rang 4.

2. Indiquer le nombre d'années à partir de l'an 2013 pour que la ville de Poitiers (Vienne) atteigne 100 000 habitants, sachant que la ville compte actuellement 87 697 habitants et que le taux de croissance annuel est de 0.89%.



Structures de contrôle itératives...

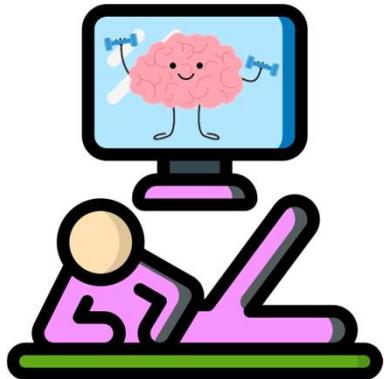
Écrire un programme en Python pour résoudre le problème décrit ci-dessous :



1. Mon ami m'a prêté la somme de 2500€ (sans intérêts). Pour rembourser mon ami, j'ai prévu de lui remettre 110 euros par mois. Je me demande :
 1. Combiens de mois ça va me prendre.
 2. Quel sera le montant à rembourser le dernier mois.

Structures de contrôle itératives...

164



- On lance une balle d'une hauteur initiale de 400 cm.
- On suppose qu'à chaque rebond, la balle perd 10% de sa hauteur (la hauteur est donc multipliée par 0.9 à chaque rebond).
- On cherche à savoir le nombre de rebonds nécessaire pour que la hauteur atteinte par la balle soit inférieure ou égale à 10 cm.

Écrire un programme en Python pour résoudre le problème.

Structures de contrôle itératives...

boucle while

166

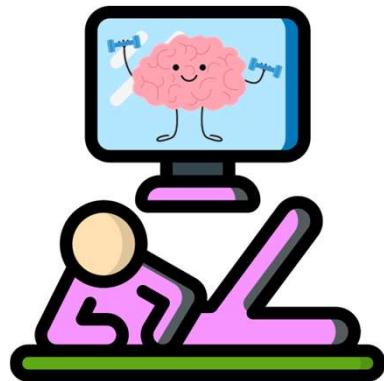
Exemple : le programme ci-dessous permet à un joueur de deviner un nombre entre 1 et 10 choisi de façon aléatoire par l'ordinateur

```
import random
# importe la bibliothèque qui permet de travailler avec des nombres aléatoires
nombre_a_deviner = random.randint(1,10)
# l'ordinateur génère un entier aléatoire entre 1 et 10
ma_prediction = 0 # cette variable représente la prédition du joueur
while nombre_a_deviner != ma_prediction :
    # cette boucle fini quand le joueur devine le nombre proposé par l'ordinateur
    ma_prediction = int(input("Indiquez votre prédition, un mombre entre 1 et 10 : "))
    if ma_prediction > nombre_a_deviner :
        print(f"Reesayez à nouveau, votre prediction {ma_prediction} est trop haute")
    elif ma_prediction < nombre_a_deviner :
        print(f"Reesayez à nouveau, votre prediction {ma_prediction} est trop base")
    else :
        print(f"You avez gagné, le nombre à déviner était {nombre_a_deviner}")
```

Structures de contrôle...

boucle while

167

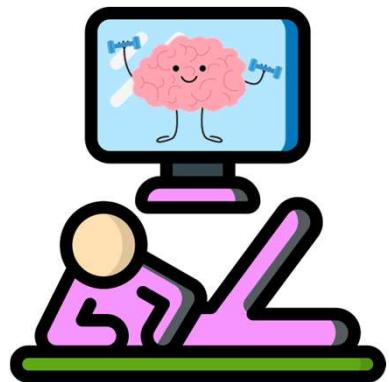


Modifier l'exemple précédent pour permettre à l'ordinateur de deviner le chiffre

Structures de contrôle...

boucle **while**

171



Ecrire un programme python pour déterminer si un nombre entier est un nombre premier.

*Un nombre entier **n** est premier s'il n'est divisible que par 1 et par lui-même*

Bonnes pratiques de codage



Liens intéressants

Médiagraphie ...

174

- Le site officiel : <https://www.python.org/>
- <https://www.pedagogie.ac-nantes.fr/mathematiques/mutualisation/logiciels/python-1049620.kjsp>
- <https://www.france-ioi.org/lycee/progresser/index.html>
- <https://openclassrooms.com/fr/courses/7168871-apprenez-les-bases-du-langage-python?archived-source=235344>
- <https://amienspython.tuxfamily.org/profs.html>
- <https://replit.com/languages/python3>