python™

“Vers l’infini et au delà”

Buzz l’Éclair

Samuel SORIN
contact@samuelsorin.fr

Sommaire

Présentation

Histoire

- Créé en 1989, par Guido van Rossum
- février 1991, la première version publique
- Le nom a été inspiré par série télévisée “Monty Python's Flying Circus”
- Actuellement géré par la Python Software Foundation
- licence libre proche de la licence BSD
- Actuellement Python 3.13.1 *(janvier 2025)*
- Plus d’info sur https://fr.wikipedia.org/wiki/Python_(langage)

Et alors, c’est quoi exactement python ?

- langage de programmation polyvalent, haut niveau, interprété,
- un des langages de programmation les plus connus et les plus utilisés au monde,
- multi-paradigme
- multiplateformes
- orientée objet
- typage dynamique fort

On fait quoi avec ?

- script (admin sys)
- logiciel (bureautique)
- Intelligence artificiel
- data science (deap learning)
- calcul scientifique
- jeux (https://www.pygame.org/)
- web

Le Zen de Python

Mieux vaut beau que laid.
Mieux vaut explicite que implicite.
Mieux vaut simple que complexe.
Mieux vaut complexe que compliqué.
Mieux vaut plat qu'imbriqué.
Mieux vaut clairsemé que dense.
La lisibilité compte.
Les cas spéciaux ne sont pas assez spéciaux pour enfreindre les règles.
Bien que la praticité l'emporte sur la pureté.
Les erreurs ne doivent jamais passer sous silence.
À moins qu'elles ne soient explicitement réduites au silence.
Face à l'ambiguïté, refusez la tentation de deviner.

Il devrait y avoir une - et de préférence une seule - façon évidente de le faire.
Bien que cette façon puisse ne pas être évidente au premier abord, à moins que vous ne soyez néerlandais.
Mieux vaut maintenant que jamais.
Même si jamais est souvent mieux que *tout de suite*.
Si l'implémentation est difficile à expliquer, c'est une mauvaise idée.
Si l'implémentation est facile à expliquer, c'est peut-être une bonne idée.
Les espaces de noms sont une idée géniale - faisons-en plus !

https://peps.python.org/pep-0020/
https://inventwithpython.com/blog/2018/08/17/the-zen-of-python-explained/

Et il y a des gens qui l'utilisent ?

- YouTube, Instagram, Google, La NASA, Spotify, Netflix, moi :) …
- Meilleurs langages en 2024 selon l'IEEE : Python leader pour la cinquième année consécutive
- Python est N°1 des langages de programmation en forte croissance
- Python est N°1 des langages les plus populaires dans la communauté open source
- source:
  https://www.blogdumoderateur.com/classement-langages-informatiques-tiobe-mars-2023/
  https://www.tiobe.com/tiobe-index/,
  https://spectrum.ieee.org/top-programming-languages-2024
  https://programmation.developpez.com/actu/364508/Python-detrone-JavaScript-et-devient-le-langage-de-programmation-le-plus-utilise-sur-GitHub-grace-a-l-explosion-des-projets-d-IA-et-de-l-arrivee-de-nouveaux-utilisateurs-exterieurs-a-la-communaute-des-codeurs/

A quoi ça ressemble Python (attention spoil)

PHP	PYTHON

<?php
// Comparaison de variable
$a = 200;
$b = 33;
 	
if ($a > $b) {
echo "a est plus grand que b";
} elseif ($a == $b) {
echo "a est égal à b";
} else {
echo "a est plus petit que b";
}
?>
	# Comparaison de variable
	a = 200
	b = 33
	
	if a > b:
	    print("a est plus grand que b")
	elif a == b:
	    print("a est égal à b")
	else:
	    print("a est plus petit que b")

Toute première fois

Installer Python

1 - Si vous êtes sur Windows , changez de système d'exploitation :)

2 - Télécharger la dernière version de python depuis https://www.python.org/ puis l'installer (https://docs.python.org/fr/3/using/windows.html)

3 - et voilà !

L’interpréteur

Windows:

- lancer l’invite de commandes
- lancez la commande "py"

Linux :

- lancer un terminal
- lancez la commande "python"

Exercice 1

1. Afficher **“Hello World”** à l’aide de la fonction **print()**
2. Tester les calculs avec python dans l’interpréteur

Éditeurs et IDE

- PyCharm
- Visual Studio Code
- Atom
- Sublime Text
- …

Exercice 2

1. Installer l’éditeur de votre choix (Pycharm ou VS Code)
2. Créer un dossier de travail sur votre disque
3. Créer un fichier **test.py**
   a. écrire le script qui permet d’afficher **“Hello World”**
   b. écrire le script qui calcul et affiche les résultats de la table de multiplication de votre choix

```
5
10
15
...
```

Calculs et variables

Calculs - les opérateurs mathématiques

symbole	nom	types	exemples
+	Additionner	entier, réel / chaîne de caractères	6+4 == 10 "a" + "b"== "ab"
-	Soustraire	entier, réel	6-4 == 2
*****	Multiplier	entier / réel /chaîne de caractères	6*4 == 24 1.2 * 1 == 1.2 3 * "s" == "sss"
******	Puissance	entier, réel	12**2 == 144
/	Diviser	entier / réel	9/2 == 4.5 9./2 == 4.5
//	Résultat entier d'une division	entier, réel	9//2 == 4
%	Modulo	entier, réel	9%2 == 1

Les variables - généralités

Sorte de “*boîte virtuelle*” dans laquelle on peut mettre une (ou plusieurs) donnée(s).

Permet de stocker temporairement une donnée pour travailler avec.

Pour la machine, une variable est une adresse mémoire où sont stockées les informations.

Les variables - affectations (ou assignation)

Affectations	a = 12
Affectations multiples	a = b = 5
Affectations parallèles	a, b = 4, 8.33

Les variables - incrémentation

Incrémentation	+=	a = 1 a += 2 >> 3
Décrémentation	-=	a = 10 a -= 3 >> 7

Les variables - nommage

Autorisé :

lettres de l'alphabet, les chiffres le caractère "_" et "-"

Interdit :

accents, signe de ponctuation, signe @.

De plus, les chiffres ne doivent jamais se trouver en première position.

Mots réservés :

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

Convention de nommage :

`my_variable`

https://pep8.org/

https://openclassrooms.com/fr/courses/4425111-perfectionnez-vous-en-python/4464230-assimilez-les-bonnes-pratiques-de-la-pep-8

Les variables - types

Typage dynamique fort

Dynamique : python se charge de choisir le meilleur type pour la variable déclarer
Fort : Python détecte les erreurs de typage

Types de variables

integer	int	Nombres entiers
string	str	Chaîne de caractères
float	float	Nombres à virgules (1.512)
boolean	bool	Bouléen (True ou False)

Les variables - types

Détermination d'un type avec la fonction type()

```
>>> a = 3
>>> type(a)
  <type 'int'>
```

Les variables - Conversion des types

- int() : permet de modifier une variable en entier.
- str() : permet de transformer la plupart des variables d'un autre type en chaînes de caractère.
- float() : permet la transformation en flottant.

```
>>> a = “3”
‘3’
>>> a = int(a)
3

>>> b = 5
5
>>> b = str(b)
‘5’
```

Built-in Functions - https://docs.python.org/3/library/functions.html

L'interpréteur Python a un certain nombre de fonctions natives et de types intégrés qui sont disponibles par défaut.

On en a déjà vu quelques unes : **print()**, **int()**, **str()**, **float()**

Et on va en utiliser plein d'autres comme **input()** par exemple.

Liste et doc de ces fonctions : https://docs.python.org/fr/3/library/functions.html

Les variables - exercices 3

1. À l’aide de la fonction native input(), demander le prénom de l’utilisateur, et afficher la phrase “Bonjour et bienvenue [PRENOM]”.

2. Reprenez le script de multiplication fait précédemment
   a. demander à l’utilisateur quelle table de multiplication il souhaite afficher.
   b. Afficher la table de multiplication à l’aide du chiffre donné par l’utilisateur.

Formater les chaînes de caractères

Formatage de chaîne - concaténation

Il est possible de concaténer plusieurs chaîne de caractères avec le symbole +

```
>>> a = “bonjour”
>>> b = “Comment ça va ?”
>>> c = a + b
print(c)

“bonjourcomment ça va”

>>> c = a + “ ” + b
print(c)

“bonjour comment ça va ?”
```

Formatage de chaîne - Ajout de variable

Il est possible d'ajouter des variables à une chaîne de différentes façon :

1 - %-formatting

L'inconvénient est qu'il peut vite devenir illisible

```
>>> name = "Eric"
>>> "Hello, %s." % name
'Hello, Eric.'
```

```
>>> name = "Eric"
>>> age = 74
>>> "Hello, %s. You are %s." % (name, age)
'Hello Eric. You are 74.'
```

Formatage de chaîne - Ajout de variable

Il est possible d'ajouter des variables à une chaîne de différentes façon :

1 - f-Strings *(Méthode conseillée)*

```
>>> name = "Eric"
>>> age = 74

>>> f"Hello, {name}. You are {age}."
'Hello, Eric. You are 74.'

>>> f"Le résultat de 2x37 est {2 * 37}"
'Le résultat de 2x37 est 74'
```

Les conditions - exercice 4

Reprenez l’exercice précédent et utiliser le **fstring** pour afficher la table de multiplication

Les conditions

Les conditions - if else

Permet de tester une condition et de n'exécuter les instructions que si cette condition est vérifiée

Il est possible de donner des instructions quelque soit les choix possibles avec le mot clé else

```
if a > 10 :
    print("a est plus grand que dix")
else:
    print("a n'est pas plus grand que dix")
```

Les conditions - elif

Permet de tester une nouvelle condition si la condition précédente n’est pas passée.

elif est la contraction de "else" et "if", qu'on pourrait traduire par "sinon si".

Il possible d'ajouter autant de conditions elif que l'on souhaite.

```
>>> a = 5
>>> if a > 5:
...     a = a + 1
... elif a == 5:
...     a = a + 1000
... else:
...     a = a - 1
>>> a
1005
```

Les conditions - les opérateurs de comparaison

<	strictement inférieur
>	strictement supérieur
<=	inférieur ou égal
>=	supérieur ou égal
==	égal
!=	différent
<>	différent, on utilisera de préférence !=
:=	Opérateur d'affectation walrus operator ou "opérateur morse". Si l'expression est exacte, la variable prend directement la valeur. *(nouveau depuis Python 3.8).*

note : Il est possible d'enchaîner les opérateurs : X < Y < Z, dans ce cas, c'est Y qui est pris en compte pour la comparaison avec Z et non pas l'évaluation de (X < Y) comme on pourrait s'y attendre dans d'autres langages.

Les conditions - AND / OR

Permet d’affiner une condition avec les mots clé AND et OR

```
>>> v = 15
>>> v > 5 and v < 10

False
```

```
>>> v = 11
>>> v > 5 or v > 100

True
```

```
>>> v = 7
>>> v > 5 and v < 10

True
```

```
>>> v = 1
>>> v > 5 or v > 100

False
```

Les conditions - chaîner les comparateurs

```
>>> a, b, c = 1, 10, 100

>>> a < b < c
True

>>> a > b < c
False
```

Les conditions - exercice 5 : *Trouver un nombre*

Règles du jeu :

- L'ordinateur tire un nombre entier au hasard entre 0 et 100.
- L'utilisateur doit le trouver et pour cela propose des valeurs.
- L'ordinateur indique pour chaque valeur proposée si la valeur est trop petite, trop grande ou s'il a trouvé.
- L'utilisateur a le droit à 3 tentatives.

> *Écrire un programme en Python pour jouer à ce jeu.*

Les boucles

Les boucles - while

Permet de répéter un bloc de code tant qu'une ou plusieurs conditions sont vraies.

```
>>> i = 0
>>> while i < 10:
...     print(i)
...     i = i + 1
```

Les boucles - for

La boucle for permet de faire des itérations sur un élément, comme une chaîne de caractères par exemple ou une liste .

```
>>> phrase = "Bonjour toi"
>>> for lettre in phrase:
...     print(lettre)
```

```
>>> for i in range(10):
...     print(i)
```

Les boucles - la fonction range

Si vous devez itérer sur une suite de nombres, la f**onction native range()** est faite pour cela. Elle génère des suites arithmétiques :

```
>>> for i in range(5):
...     print(i)
0
1
2
3
4
```

Le dernier élément fourni en paramètre ne fait jamais partie de la liste générée ; range(10) génère une liste de 10 valeurs, dont les valeurs vont de 0 à 9

Les boucles - break et continue

L'instruction "break" permet d'arrêter une boucle avant sa fin.

```
for i in range(5):
    if i == 3:
        break
    print i
```

L'instruction "continue" permet de passer à l'itération suivante.

```
for i in range(5):
   if i == 3:
        continue
   print i
```

Les boucles - exercice 6

Reprendre le script de table de multiplication et l'améliorer :

- en utilisant la boucle "for"
- en utilisant la boucle "while"
- en demandant à l'utilisateur combien de résultat il souhaite afficher

Les boucles - exercice 7a / 7b

Reprendre le script pour trouver un nombre au hasard et l'améliorer

1. en utilisant la boucle "while"
2. en utilisant la boucle "for"

Les boucles - exercice 8

Créer un script qui permet de demander à un utilisateur de saisir 2 nombres, puis les actions qu'il souhaite effectuer avec.

- On affiche un message récapitulant l'action à effectuer.
- Tant que l'utilisateur ne demande pas lui-même d'arrêter, le programme continue de tourner.
- Les actions sont :
    A = Additionner,
    M = Multiplier,
    S = Soustraire,
    E = Exit
- Selon l'action, afficher le message de l'opération à effectuer et son résultat

(par exemple : "2 + 2 = 4")

Pierre Papier Ciseaux

Exercice qui récapitule toutes les notions vus dans la partie 1 : variable, boucle, conditions

Instruction :

- implémentez la logique de base du jeu Pierre Papier Ciseaux (pierre bat ciseaux, papier bat pierre, ciseaux bat papier)
- puis demandez un input clavier.
- L'ordinateur choisit un coup au hasard avec une fonction du package random, pour finir affichez qui a gagné la partie.