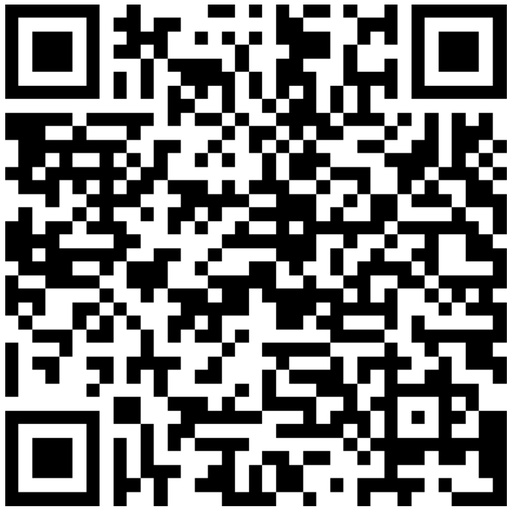
Atelier développement

ou programmation Python



Pour retrouver les morceaux de code présentés

AT

État d’esprit du développeur

Rechercher une réponse à son problème seul

- Formuler le besoin « unitaire » plutôt que le problème en lui-même

Savoir copier-coller

Explorer les résultats petit à petit, découvrir et appréhender par soit même

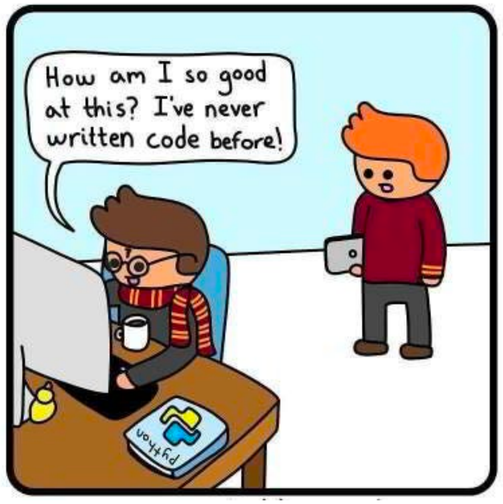
Lire les documentations

Essayer, se tromper, se corriger

Développer avec Python

En mode « notebook » avec Google Colab

En mode « normal » avec Spyder , Pycharm



Les variables

Le développement implique de **stocker** et **travailler** avec **différents types de données**. On fait cela en utilisant des **variables**

Une variable stocke une donnée et lui donne un nom spécifique

```
variable = "valeur"
```

On va pouvoir ensuite afficher cette variable

```
print(variable)
```

On peut également demander à l'utilisateur de renseigner une variable

```
var_utilisateur = input("Entrez une valeur : ")
```

Types de données

```
variable = "valeur"

nb_eleves = 32                  # Numérique, int
progression = 0.01              # Numérique, float
nom_cours = "Atelier dév"       # Chaîne de caractères, string
cours_termine = False           # Booléen, bool
```

Les opérations

Sur les nombres

```
addition = 1 + 2
soustraction = 1 - 2
multiplication = 1 * 2
division = 1 / 2
modulo = 3 % 2              # reste d'une division
```

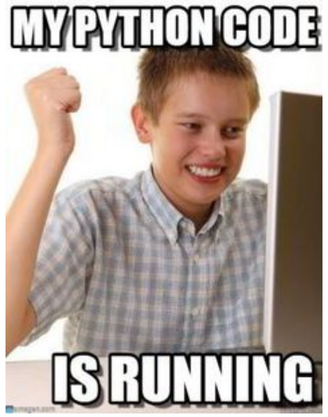
Les opérations

Sur le texte

```
debut_phrase = "j'adore le "
fin_phrase = "python"
phrase = debut_phrase + fin_phrase
print(phrase)                # « j'adore le python »

phrase = debut_phrase + fin_phrase + " !!!"
print(phrase)                # « j'adore le python !!!»
```

Exercice



Les variables

Les listes

Les listes sont des ensembles d'objets

```
ma_liste = [1, 2, 3, 4, 5]
ma_liste = ["a", "b", "c", "d", "e"]
ma_liste_vide = []
```

On peut **accéder** à un élément spécifique d'une liste en utilisant un **indice**

```
print(ma_liste[0])      # a

ma_liste[2] = "uwu"
print(ma_liste)         # ["a", "b", "uwu", "d", "e"]
```

Les variables

Les listes

On va pouvoir **ajouter**, **trier** et **supprimer** des éléments à une liste

```
ma_liste = ["Pierre", "Paul", "Jacques"]
ma_liste.append("Marie")
print(ma_liste)    # ["Pierre", "Paul", "Jacques", « Marie »]

ma_liste.sort()
print(ma_liste)    # ["Jacques", "Marie", "Paul", "Pierre"]

ma_liste.remove("Paul")
print(ma_liste)    # ["Pierre", "Jacques", « Marie"]
```

Exercice

Les variables

Les dictionnaires

Un dictionnaire est similaire à une liste mais on va retrouver des **clés à la place des indices**

```
mon_dico = {"prenom": "Pierre", "nom": "Dupont"}
print(mon_dico["prenom"]) # "Pierre"
```

On peut créer un **dictionnaire vide** et, également, **une liste de dictionnaires**

Les variables

Les dictionnaires

On peut **créer** une nouvelle paire clé-valeur, **affecter une nouvelle valeur** et **supprimer** une paire

```
mon_dico["age"] = 18
print(mon_dico) # {"prenom": "Pierre", "nom": « Dupont",
                  "age": 18}

mon_dico["age"] = 19
print(mon_dico) # {"prenom": "Pierre", "nom": « Dupont",
                  "age": 19}

del mon_dico["age"]
print(mon_dico) # {"prenom": "Pierre", "nom": "Dupont"}
```

Exercice

Les structures conditionnelles

L’instruction « if » **exécute un code spécifié si son expression est vraie**

```
mon_age = 19

if mon_age > 18:
    print("Vous êtes majeur")
```

Les structures conditionnelles

« else » permet d'exécuter un code spécifié **si la précédente condition « if » n'est pas vraie**

```
mon_age = 19

if mon_age > 18:
    print("Vous êtes majeur")
else:
    print("Vous êtes mineur")
```

Les structures conditionnelles

« elif » permet de spécifier **plusieurs conditions dans une même structure**

```
mon_age = 77

if mon_age > 18:
    print("Vous êtes majeur")
elif mon_age > 32:
    print("Vous êtes trop agé")
elif mon_age < 13:
    print("Vous êtes trop jeune")
else:
    print("Vous êtes mineur")
```

Les opérateurs

Un **opérateur logique** (« not », « and », « or ») permet de vérifier plusieurs conditions

```
if not cours_termine:
    print("Le cours n'est pas terminé")

if mon_age > 18 and mon_age < 32:
    print("Vous êtes majeur et jeune")

if mon_os == "Windows" or mon_os == "MacOS":
    print("Vous utilisez Windows ou MacOS")
```

Les opérateurs

Un **opérateur de comparaison** permet de comparer une variable à quelque-chose

```
a == 1  # strictement égal à
a != 1  # différent de
a > 1  # strictement supérieur à
a >= 1  # supérieur ou égal à
a < 1  # strictement inférieur à
a <= 1  # inférieur ou égal à

mon_age = 21
if mon_age == 21:
    print("Vous avez 21 ans")
```

Exercice

Les structures itératives

Les boucles servent à **répéter du code**

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

while not cours_termine:
    print("Le cours n'est pas terminé »)
    cours_termine = True
```

Les structures itératives

On peut également **parcourir une liste**

```
ma_liste = ["Pierre", "Paul", "Jacques"]
for eleve in ma_liste:
    print(eleve)
    # Pierre
    # Paul
    # Jacques
```

Exercice

Les fonctions

Une fonction sert à regrouper du code qui sera exécuté quand la fonction sera appelée

```
def dire_bonjour(nom):
  print("Bonjour " + nom + " !")

dire_bonjour("Pierre") # Bonjour Pierre !

def ajoute(nb):
  return nb + 1

nouveau_nb = ajoute(2)
print(nouveau_nb) # 3
```

Exercice

Les librairies

Il existe des libraires pour faire… à peu près **tout** et leur utilisation est très **simple**, on a des librairies « natives »

```
import random

nombre = random.randint(0, 100)
print(nombre)
```

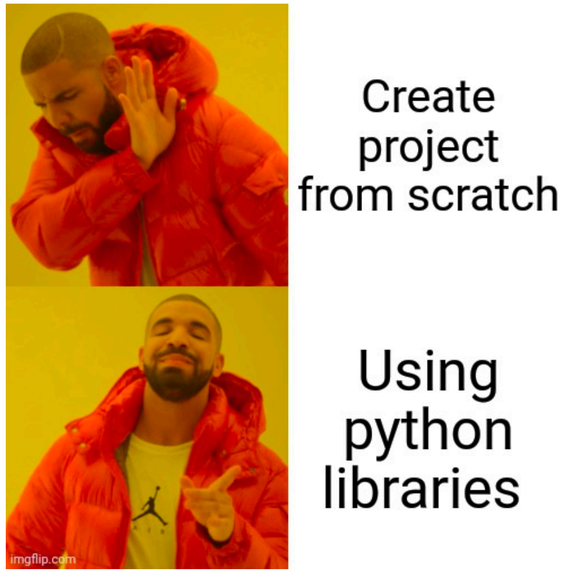
Les librairies

Et des librairies **à télécharger**, par exemple, via PyPi avec « *pip install* » dans la console ou en précédent la commande d'un « *!* » sur Colab

```
pip install fuzzywuzzy
```

```
from fuzzywuzzy import fuzz

ratio = fuzz.ratio("vitalik buterin",
"vitalik butherin")
print(ratio) # 97
```



Exercice

Exploiter des sources de données

Depuis un fichier

On peut lire tout type de fichier en Python, il existe plusieurs méthodes

Ici, on lit un fichier .csv

```
import csv

with open('mon_fichier.csv') as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=';')
    for row in csv_reader:
        print(row)
```

Exploiter des sources de données

Depuis un fichier

On peut lire tout type de fichier en Python, il existe plusieurs méthodes

Ici, on lit un fichier .json

```
import json

with open('mon_fichier.json', 'r') as fichier_json:
    donnees_json = json.load(fichier_json)
```

Exploiter des sources de données

Depuis une API

On peut également interroger des services web prévus à cet effet, souvent leur réponse est au format .json

```
import requests

url = ("https://recherche-entreprises.api.gouv.fr/search"
       "?categorie_entreprise=PME,ETI"
       "&departement=83")

donnees = requests.get(url).json()

print(donnees)  # 'results': [{'siren': '775676182', 'nom_...
```

Exercice

Analyser les données

Exercice