Introduction à Python

M1 IEF - 2024

1. Qu'est ce que Python ?

2. La nécessité de l'indentation

3. Qu'est ce que Jupyter ?

4. Quels sont les IDEs (Integrated Development Environments) à ma disposition ?
    1. (Default) IDLE
    2. (Mon choix) Visual Studio Code (VSCode)
    3. PyCharm
    4. Sublime Text
    5. Jupyter
    6. Spyder
    7. Et bien d'autres ...

Plan

01 Les variables

02 Les structures de contrôle

03 Les fonctions

04 Manipuler les dataframes

05 Les séries temporelles

06 Les APIs

1. Les types
   - Chaines de caractères: **str**
   - Numériques: **int, float, complex**
   - Booleens: **bool**
   - Vide: **None**

2. Les listes : [1, 2.0, 'hi']

3. Afficher un résultat: **print**
   Concaterner du texte avec les fstring : f"{var}"

4. Importer un fichier
   - Chemin relatif vs absolue (./, ../)
   - csv, xlsx
   - Vérifier les types des colonnes

5. Les modules
   Deux incontournables : pandas (pour les dataframes) et numpy (pour les aspects matriciels)

```
# Affecte la valeur 5 à la variable 'a'
a = "5"
a <- "5"

# Affiche le type de 'a’
type(a)

# Incrémente la variable 'a' de 2
a = a + 2
a+=2

# Change le type de 'a'
a = int(a)
a = float(a)

# Affichage dynamique basé sur la valeur de 'a'
print( "La valeur de a est", a)
print( f"La valeur de a est {a} ")

# Importer un module
import pandas as pd
import numpy as np
```

1. Les boucles
   - for : parcours tous les éléments d'un vecteur et execute autant de fois un code donné

     for i in vecteur:
         code

   - while : Exécute un codé donné tant qu'une condition n'est pas remplie

     while condition:
         code

2. Les conditions: execute ou non un code en fonction d'une condition

   if condition:
       code
   elif condition:
       code
   else:
       code

```
# parcours le vecteur c(1,2,3,4) et affiche les valeurs
for i in range(1,5):
  print(i)

# parcours un autre vecteur et affiche les valeurs
for i in ["Bonjour","tout","le","monde","!"]:
  print(i)


# Incrémente i de 1 jusqu'a ce que i vaille 5
i = 1
while i < 5:
  print(i)
  i += 1


# Affichage des valeurs de a sous certaines conditions
a = 7

if a == 7:
  print("a = 7")
elif  a == 10:
  print("a = 10")
else:
  print("a n'est ni égal à 7 ni à 10.")
```

1. Vérifier si un nombre est premier ou pas. Pour savoir si un nombre est divisible par un autre, utilisez le modulo (% en Python)

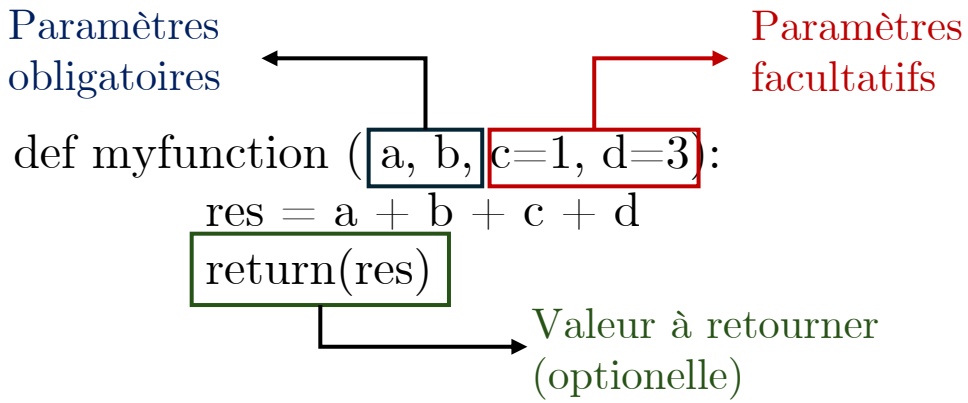
2. Les deux premiers termes de la suite de Fibonacci sont tous deux un. Les termes suivants de la séquence sont trouvés en additionnant les deux termes immédiatement précédents. Écrivez un code qui écrit les n termes de la séquence de Fibonacci pour tout $n \geq 3$

3. Écrivez un code qui simule le lancer d'une pièce avec une probabilité p associée aux piles, jusqu'à obtenir fâce, et renvoie le nombre de piles obtenues. Ensuite, reproduisez l'exécution de ce code 100 fois et afficher le nombre maximum de piles obtenus lors d'une même séquence.

   Pour générer une probabilité entre 0 et 1, utiliser :

   import random
   random.random()

Une fonction informatique est un ensemble d'instructions regroupées sous un nom et s'exécutant à la demande (l'appel de la fonction).




```python
# Déclaration d'une fonction
def myFunction(a:int, b:int, c:list) -> int:
    """ Additionne 3 valeurs
      Args:
        - a: la premiere valeur
        - b: la seconde valeur
        - c: la derniere valeur
    """
    res = a + b + c
    return res

# Différents appels à ma fonction
myFunction(a = 1, b = 2)
myFunction(1, 2)
xx = myFunction(a = 1, c = 2, b = 3)
```

Toutes les variables contenues dans une fonction (paramètres inclus) sont 'locales' et ne sont accessibles que dans cette fonction. Elles sont créées à l'appel de la fonction et detruites à la fin de son exécution. Par opposition aux variables déclarées partout ailleurs dans le code, dites 'globales' et accessibles partout (y compris dans une fonction).

1. Vérifier si un nombre est premier ou pas. Pour savoir si un nombre est divisible par un autre, utilisez le modulo (%% en R)

   => Transformer votre réponse en utilisant une fonction avec un seul parametre **obligatoire** définissant le nombre à vérifier.

2. Les deux premiers termes de la suite de Fibonacci sont tous deux un. Les termes suivants de la séquence sont trouvés en additionnant les deux termes immédiatement précédents. Écrivez un code qui écrit les n termes de la séquence de Fibonacci pour tout n≥3

   => Transformer votre réponse en utilisant une fonction avec un seul parametre **obligatoire** n définissant le nombre de termes à ajouter à la suite de Fibonacci.

3. Écrivez un code qui simule le lancer d'une pièce avec une probabilité p associée aux piles, jusqu'à obtenir fâce, et renvoie le nombre de piles obtenues. Ensuite, reproduisez l'exécution de ce code 100 fois et afficher le nombre maximum de piles obtenus lors d'une même séquence. Pour générer une probabilité entre 0 et 1, utilisez runif(1).

   => Transformer votre réponse en utilisant une fonction avec deux paramètres **facultatifs**. Un premier pour définir la probabilité associée aux piles (par défaut 0.5), et un second pour le nombre de lancers (par défaut 100).