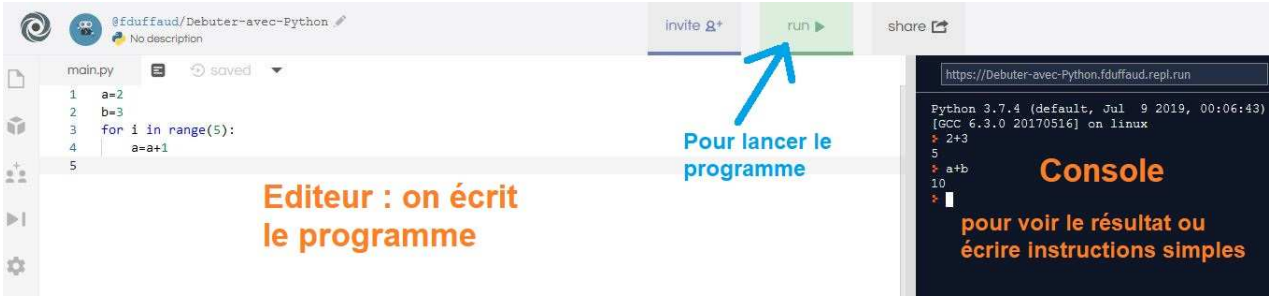
Math93.com

# TD - NSI
# Débuter en Python

## 1 Lancer l'éditeur Python

On ouvre un éditeur Python :

1. avec un éditeur en ligne : www.repl.it
2. avec un éditeur comme Spyder : https ://pypi.org/project/spyder/
3. avec un éditeur comme Edupython : https ://edupython.tuxfamily.org/




## 2 Dans la console

La console se reconnaît facilement. C'est elle qui contient le chevron > (ou le triple >>>) qui est l'invite de Python (prompt en anglais) et qui signifie que Python attend une commande.
L'esprit d'utilisation de la console est un peu le même que celui d'une calculatrice.

```
>>>  2+3
5
>>>  a=5
>>>  a-9
-4
```

**Exercice 2.1**

Lancer les instructions suivantes dans la console en appuyant sur **Enter** à chaque fin de ligne et regarder le résultat :

```
>>>print ( "Hello world !")
>>>x=3
>>>print(x)
>>>4+5
>>>5/2
>>>5//2
>>>print("la valeur de x est", x)
```

## 3 Les types : type(), les variables et les affectations

Voici les différents types de variables que vous devez connaitre :

- Nombres entiers relatifs : **int()**
- Nombres flottant( "décimaux" mais complètement) : **float()**
- Booléens : **bool()**. ( prend seulement **True** ou **False**)
- Les chaînes de caractères : **str()** qui sont entre guillemets " chaîne"
- Tableau dynamique : **list()**

```
# Exemples
# Dans la console Python
>>>type(23)
>>>type(2.3)
>>>type(2<3)
>>>type("2<3")
>>>type([2,3])
```

**#**

Le symbole # (se lit « croisillon », « hash » en anglais , symbole proche du « dièse ») permet de faire figurer dans le corps du programme un commentaire qui ne sera pas pris en compte lors de son exécution.

**=**

Le symbole = n'est pas celui de l'égalité mathématique, il n'est d'ailleurs pas symétrique. Il s'agit d'affecter une valeur à une variable : on stocke une valeur numérique ou du texte dans une mémoire.

```
# Exemples
# Dans la console Python
>>> age=5
>>> type(age)
>>> taille=8.
>>>type(taille)
>>> poids=7.2
>>>type(poids)
>>> nom="khofi"
>>>type(nom)
>>> YeuxBleu=False
>>>type(YeuxBleu )
```

On peut afficher ces variables avec la fonction print() par exemple print(age) ou taper seulement *age* dans la console puis *Enter*

## 4 Pour commencer un programme

- Sur repl.it : cliquez simplement sur : **+ new repl** et donner un nom à votre fichier. Il sera automatiquement enregistré après chaque **Run**.
- Sinon sur un éditeur hors ligne : Cliquer sur Fichier puis Nouveau puis sélectionner Nouveau Module Python. Enregistrer IMMEDIATEMENT votre programme dans votre répertoire de travail avec le nom Mon1erProgramme.py.

Nous allons maintenant faire des programmes très simples pour comprendre la syntaxe de python.
Voici des algorithmes écrit en pseudo code et sa traduction en python. Écrire le code en python dans l'éditeur et lancer le en appuyant sur la flèche verte. Mettre le résultat dans la dernière colonne. Vous pouvez écrire tous les codes à la suite dans le même fichier. Séparer-les par exemple par print("pseudo code i") où i est le numéro du pseudo code.

## 4.1 Pseudo code 1

| **Pseudo Code 1** | **Code Python** | **Affichage** |
|---|---|---|
| Afficher "Entrer votre nom"<br>Entrer Nom<br>Afficher "Votre nom est ",Nom | `Nom=input("Entrer Votre nom")`<br>`print("Votre nom est ",Nom)` | |

## 4.2 Pseudo code 2

| **Pseudo Code 2** | **Code Python** | **Affichage** |
|---|---|---|
| **Pour** N allant de 0 à 5 **faire**<br>Afficher N<br>**Fin Pour** | `for N in range(6):`<br>`  print(N)` | |

**Remarque**

— range(n) fait une liste de $n$ valeurs commençant par 0 .

— ATTENTION les indentations sont obligatoires en python.

## 4.3 Pseudo code 3

| **Pseudo Code 3** | **Code Python** | **Affichage** |
|---|---|---|
| **Pour** N allant de 0 à 5 **faire**<br>Afficher "N"<br>**Fin Pour** | `for N in range(6):`<br>`  print("N")` | |

**Remarque**

— Les chaînes de caractères sont de type **string (str)**.

— Les guillemets affichent ou indiquent le type **string (str)**.

### 4.4 Pseudo code 4

Je voudrais afficher la plage de valeur de mon choix :

**Pseudo Code 4**

Afficher "La dernière valeur?"

Entrer Der
**Pour** N allant de 0 à Der **faire**
  Afficher N
**Fin Pour**

**Code Python**

```
Der=input("La dernière valeur?")
Der=int(Der)
for N in range(0,Der+1):
    print(N)
```

**Affichage**

**Remarque**

- Le **input** permet de saisir une chaîne de caractère (str). Il faut convertir un string (str) en entier (int).
- **int(Der)** permet la conversion. Essayer sans cette ligne pour voir le résultat.

### 4.5 Pseudo code 5 : opérations de base

Je voudrais calculer $a+b$, $a\times b$, $a-b$, $a/b$ (division décimale) $a//b$, (quotient de la division euclidienne) et $a\%b$ (reste de la division euclidienne) sachant que $a = 123456789$ et $b = 987654321$.

**Pseudo Code 5**

$a \leftarrow 123456789$
$b \leftarrow 987654321$
$somme \leftarrow a+b$
$produit \leftarrow a\times b$
$difference \leftarrow a-b$
$quotientD \leftarrow a/b$
$quotientE \leftarrow a//b$
$resteE \leftarrow a\%b$
Afficher "$a+b=$",$somme$
Afficher "$a\times b=$",$produit$
Afficher "$a-b=$", $difference$
Afficher "$a/b=$",$quotientD$
Afficher "$a//b=$",$quotientE$
Afficher "$a\%b=$",$resteE$

**Code Python**

```
a=123456789
b=987654321
somme=a+b
produit=a*b
difference=a-b
quotient=a/b
quotientD=a/b
quotientE=a//b
resteE=a%b
print("a+b=",somme)
print ("a*b=",produit)
print ("a-b=", difference)
print ("a/b=",quotientD)
print ("a//b=",quotientE)
print ("a%b=",resteE)
```

**Affichage**

**Remarque**

- Remarquer que le symbole = sert à l'affectation mais il ne veut pas dire "égal" en python, il veut dire "prendre la valeur ". C'est pour cela qu'on utilise la flèche en pseudo code.
- Le produit s'écrit * et pas × ni x en python!!
- On pouvait aussi éviter "print(somme)" en écrivant simplement *somme* dans la console.

**Aide**

**Une Astuce (à ne pas dévoiler!)**

```
# Méthode 2 :  f (format) devant une chaîne de caractères
#              et les variables entre {}
a=2
b=10
>>>print(f"a-b= {a-b}")
a-b=-8
```

# 5 Faire Une fonction ...

**def nom_ fonction(paramètres) :**

*def nom_fonction(paramètres) :* définit une nouvelle fonction, les deux points entraînent une indentation délimitant la déclaration de la fonction. Le bloc peut servir à effectuer une série d'actions, mais le plus souvent il se termine par *return* pour renvoyer une ou plusieurs valeurs.

## 5.1 Une fonction ... d'Euler

```
# Dans l'éditeur PYTHON
def f(n):
        return n**2-n+41 # n**2=n*n (notation puissance)
```

La fonction définie ci-dessus renvoie l'image de la variable $x$ par la fonction $f$ définie par :

$$f(x) = x^2 - x + 41$$

Tester le programme avec des entiers naturels en écrivant dans la console (*à droite sur ripl.it ou votre logiciel IDLE*) directement $f(5)$ ou $f(8)$ par exemple.

```
# Dans la console PYTHON
>>> f(5)
61
>>> f(8)
97
```

## 5.2 Une liste par compréhension ... première approche

On veut maintenant calculer les valeurs de la fonction $f$ pour $x$ entier variant de 0 à 19 par exemple.

```
# Dans l'éditeur PYTHON
def f(n):
        return n**2-n+41 # n**2=n*n (notation puissance)

valeursx=[x for x in range(20)] # range(20) itère de 0 à 19
valeursy=[f(x) for x in range(20)]
```

Pour visualiser le résultat, écrire simplement valeursx et valeursf dans la console de droite.

```
# Dans la console PYTHON
>>> valeursx
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
 18, 19]
>>> valeursy
[41, 41, 43, 47, 53, 61, 71, 83, 97, 113, 131, 151, 173, 197,
 223, 251, 281, 313, 347, 383]
>>>
```

### 5.3 Des fonctions dans un programme

Un programme long doit être structuré avec des fonctions. Cela permet une meilleur lisibilité et un debbugage efficace. Un programme doit être structuré de la manière suivante :

- En tête de codage des caractères
- Informations diverses
- Définitions des fonctions
  - Il n'y a pas de print dans une fonction de calcul. On doit retourner la valeur avec return.
  - On peut utiliser les fonctions précédentes pour définir une nouvelle fonction.
  - Si on fait un test on retourne une valeur booléenne (bool). Elle ne prend que les valeurs True ou False
- Programme principal. Il doit être très court.

Voici un exemple qui teste si un triangle est rectangle, testez le et essayer de bien comprendre la structure :

**Code Python**

```
#En tête de codage des caractères
# -*- coding: utf-8 -*-
#Informations diverses
"""Created on Sun Jul 07 12:09:54 2019
@author: courtois"""

#Définitions des fonctions
def sumCa(a,b):
    """
    In: deux nombres a et b float
    Out: la somme de a² et b² float
    """
    S=a*a+b*b
    return S

def testTriangleRectangle(a,b,c):
        """
        In: a,b,c (float) longueurs des cotés d'un triangle
        Out: Booléen: True si  triangle rectangle sinon False
        Utilise le theoreme de Pythagore
        """
        if a*a== sumCa(b,c) or b*b== sumCa(a,c) or c*c== sumCa(a,b):
                return True
        else:
                return False
#Programme Principal
M=2.1.
N=4.6
P=25.57
resultat=testTriangleRectangle(M,N,P)
print("MNP est t-il rectangle ? " , resultat)
```

## 5.4 Exercice

1. Calculer à la mains la somme $S(5)$ des entiers de 0 à 5 puis la somme $S(10)$ des entiers de 0 à 10.

$$S(5) = 0+1+\cdots+5 = \cdots \quad \text{et} \quad S(10) = 0+1+\cdots+10 = \cdots$$

2. L' algorithme suivant calcule la somme des premier entiers de 0 à N. A vous maintenant de traduire en python en utilisant une fonction. Attention quand on va de 0 à $N$, il y a $N+1$ valeurs.

| **Pseudo Code 6** somme_entier(N) | **Affichage N=5** | **Affichage N=10** |
|---|---|---|
| K=0<br>**Pour** P allant de 0 à N **faire**<br>$K \leftarrow K+P$<br>**Fin Pour**<br>Retourner K | | |

```
def somme_entier(n):
    '''IN : entier n >=0 (int)
      OUT : somme des entiers de 0 à n (int)
    ...
```

Vous pouvez rajouter dans le code python une partie de programme principal pour tester la fonction ou utiliser la console comme suit.

```
# Dans la console PYTHON
>>> sommeEntier(5)
```

Appeler moi quand vous avez réussi celui ci.

POUR LA SUITE IL FAUDRA TOUJOURS FAIRE DES FONCTIONS

## 6 Boucle Tant Que

On peut aussi utiliser une boucle **Tant Que** si on ne sait pas le nombre d'action à effectuer. Une boucle **Pour** peut être remplacer facilement par une boucle **Tant Que**.
La fonction suivante calcule la valeur d'indice $N$ de la suite définie par

$$\begin{cases} U_0 = 10 \\ U_{n+1} = 5U_n + 3 \end{cases}$$

Coder la fonction avec un programme principal :

**Pseudo Code 7** suite(N)

```
U ← 10
I ← 0
Tant Que I<N faire
    I ← I+1
    U ← 5U+3
Fin Tant Que
Retourner U
```

**Code Python**

```
def suite(N):
    """
    In: N un entier int
    Out: un float
    Nième terme de U
    U_{i+1}=5U_i+3, U0=10
    """
      U=10
       I=0
       while U<N:
              I=I+1
              U=5*U+3
       return U
```

**Affichage $N = 1$ et $N = 3$**

## 7 Saut conditionnel if...elif...else

1. La fonction suivante demande un nombre décimal (float) et teste s'il est positif :

**Pseudo Code 8** is_positif(x)

```
Si x>0 alors
    Retourner Vrai
Sinon
    Retourner Faux
Fin Si
```

**Code Python**

```
def is_positif(x):
    """
     In: un float
     Out: un booléen
    """
    if x>0:
       return True
    else:
       return False
```

**Affichage**

**$x = -5.7$ ; $x = 0$ et $x = 37$**

2. Il y a un petit problème, l'algorithme dit que 0 est négatif alors que c'est un nombre négatif et positif. Il faut un cas particulier pour 0. Voici un nouvel algorithme qui corrige ce problème.

**Pseudo Code 9** is_positif(x)

```
Si x = 0 alors
    Retourner Vrai
Sinon si x > 0 alors
    Retourner Vrai
Sinon
    Retourner Faux
Fin Si
```

**Code Python**

```
def is_positif(x):
    """
     In: un float
     Out: un booléen
    """
    if x==0:
       return True
    elif x>0:
       return True
    else:
       return False
```

**Affichage**

**$x = -5.7$ ; $x = 0$ et $x = 37$**

**Remarque**

— Remarquer la différence entre $x = 0$ et $x == 0$ :
  — le = affecte une valeur à $x$
  — le == vérifie une condition. $x == 0$ est une variable booléenne, elle vaut True ou False.
— On pourrait écrire l'algorithme précédent sans utiliser elif. Comment ?

3. L'algorithme suivant teste si un entier est un nombre premier. On mélange plusieurs instructions.

**Pseudo Code 10** is_prime(x)

$P \leftarrow 2$
**Tant Que** $N/P$ n'est pas un entier
**faire**
  $P \leftarrow P+1$
**Fin Tant Que**
**Si** $P = N$ **alors**
  Retourner Vrai
**Sinon**
  Retourner Faux
**Fin Si**

**Code Python**

```
def is_prime(N):
    """
    In: un int
    Out: un booléen
    """
    P=2
    while N/P!=int(N/P):
        P=P+1
    if P==N:
        return True
    else:
        return False
```

**Affichage**

| N | 2 | 3 | 6 | 23 | 107 |
|---|---|---|---|---|---|
| | | | | | |

**Remarque**

— **!=** veut dire "different de".
— **int(x)** c'est l'entier juste inférieur à x
— **while N/P!=int(N/P)** veut donc dire "Tant que le decimal N/P est différent de l'entier juste inférieur à N/P" ce qui revient à "Tant que N/P n'est pas un entier".

# 8 Quelques exercices

## 8.1 Exercice

La racine carré n'existe pas en python. Pour pouvoir la calculer, on a besoin de rajouter des fonctions qui sont dans un autre fichier python. Il faut pour cela importer le fichier math avec l'instruction import. On peut alors calculer $\sqrt{5}$ avec math.sqrt(5).

```
# Dans la console PYTHON
>>>import math
>>>math.sqrt(5)
```

**Dans l'éditeur Python**

```
import math
print(math.sqrt(5))
```

Coder en python l'algorithme suivant qui compte le nombre de carré parfait inférieur à N :

**Pseudo Code 11** nbCarre(N)

$Nb \leftarrow 0$
**Pour** $i = 0$ à N **faire**
  **Si** $\sqrt{i}$ est un entier **alors**
    $Nb \leftarrow Nb + 1$
  **Fin Si**
**Fin Pour**
Retourner Nb

| **Affichage $N = 10$** | **Affichage $N = 100$** |
|---|---|
| | |

## 8.2 Exercice

Que fait l'algorithme suivant ? Coder le en python :

**Pseudo Code 12** mystere(N)

$P \leftarrow 1$
**Pour** $i = 0$ à N **faire**
  $P \leftarrow P \times i$
**Fin Pour**
Retourner P

| **Affichage $N = 4$** | **Affichage $N = 10$** |
|---|---|
| | |

# 9 Les tableaux dynamiques (list python)

## 9.1 Définition et exemples

**$[val_0, val_1, ..., val_{n-1}]$ :**

$[val_0, val_1, ..., val_{n-1}]$ crée un tableau contenant $n$ valeurs indicées de 0 à n–1. C'est un objet de type list.

**Une liste : L**

Une liste est une suite d'éléments numérotés de même type dont le premier indice est 0. En Python, une liste s'écrit entre crochets [... , ..., ..., ...] avec les éléments séparés par des virgules.

- Le premier élément de la liste est L[0], le 2e est L[1], ...
- Une liste peut être écrite de manière explicite : L = ["Lundi", "Mardi","Mercredi"]
- Sa longueur est donnée par $len(L)$.
- Si les éléments de la liste sont comparables, le max. est donné par *max(L)*, le min. par *min(L)*
- L=[] permet de définir une liste vide.
- Si L est une liste, l'instruction *L.append(x)* va ajouter l'élémentt [x] à la liste L.
- Pour parcourir la liste L , deux solutions sont possibles :
  - un parcours sur les indices :
    for i in range(len(L)):
        L[i] ...
  - un parcours direct sur les éléments :
    for X in L:
        X ...

```
# Dans la console PYTHON
>>>[3,5,7,35]
>>>list("toto")  #transformation d'une chaine en liste
>>>MaListe=[3,5,151,35]
>>>MaListe[2]
```

## 9.2 Opération sur les listes

- appartenance : valeur in $monTableau$
- indice : $monTableau[i]$ (attention le 1er indice est 0)
- Découpage (même chose pour les chaines de caractères)
  - $monTableau[n : p]$ retourne la liste contenue dans $monTableau$ entre le nième exclus et le pième indice inclus.
  - $monTableau[p :]$retourne une liste sans les p premiers indices de $monTableau$
  - $monTableau[: p]$ retourne une liste sans les p derniers indices de $monTableau$
- longueur : len($tableau$)
- affectation : $tableau[i] = valeur$
- Ajout d'une valeur v : $tableau.append(v)$

— Creation d'une liste d'entiers (int) avec un pas $p$ : $list(range(a,b,p)) \subset [a,b[$

Regarder le résultat de :

```
# Dans la console PYTHON
>>>alph="abcdefghij"
>>>Maliste=list(alph)
>>>Maliste
>>>Maliste[2:]
>>>Maliste[:5]
>>>len(Maliste)
>>>Maliste[4]=M
>>>Maliste
>>>Maliste.append(26)
>>>Maliste
>>>"a" in Maliste
>>>45 in Maliste
```

Remarquer que l'on peut faire parfois la même chose avec des chaines de caractères : Essayer ces instructions (elles ne fonctionnent pas toutes) :

```
# Dans la console PYTHON
>>>chaine="Comment allez vous ?"
>>>chaine[2:]
>>>chaine[:5]
>>>len(chaine)
>>>chaine[4]=M
>>>chaine
>>>chaine.append(26)
>>>chaine
>>>chaine=chaine+"26" #cette operation s'appelle la concatenation
>>>chaine
>>>"a" in chaine
>>>"w" in chaine
>>>45 in chaine
```

# 10 Les dictionnaires

**$\{clé_1 : val_1, clé_2 : val_2, ..., clé_n : val_n\}$ :**

$\{clé_1 : val_1, clé_2 : val_2, ..., clé_n : val_n\}$ est une liste dont la clé n'est pas l'indice de position mais qui une valeur qui peut être de n'importe quel type.
- Un dictionnaire en python est donc une sorte de liste mais au lieu d'utiliser des index , on utilise des clés alphanumériques.
- Dans un dictionnaire, les informations ne sont pas stockées dans un ordre précis. Pour accéder aux valeurs, on utilise les clés.

**Exemples** :

```
# Dans la console PYTHON
# Pour initialiser un dictionnaire
>>>dico = {} # ou dico=dict{}

# Pour ajouter des valeurs à un dictionnaire il faut indiquer
# une clé ainsi qu'une valeur :
>>> dico["nom"] = "Turing" # clé1 = "nom" et valeur associée "Turing"
>>> dico["prenom"] = "Alan"
>>> dico["annee_naissance"] = 1912

>>> dico
{'nom': 'Turing', 'prenom': 'Alan', 'annee_naissance': 1912}

# On peut vérifier que la variable dico est bien de type dictionnaire
>>>type(dico)
<class 'dict'>
```

## 10.1 .get() : Comment récupérer une valeur dans un dictionnaire python ? : .

La méthode get vous permet de récupérer une valeur dans un dictionnaire et si la clé est introuvable, vous pouvez donner une valeur à retourner par défaut :

```
>>> dico
{'nom': 'Turing', 'prenom': 'Alan', 'annee_naissance': 1912}

>>> dico
{'nom': 'Turing', 'prenom': 'Alan', 'annee_naissance': 1912}
>>>dico.get("nom")
'Turing'
>>>dico.get("annee_naissance")
1912
>>>dico.get("poids") # si la clé n'est pas trouvée, rien ne s'affiche
```

### 10.2 .key() : Comment récupérer les clés d'un dictionnaire python par une boucle ?

Pour récupérer les clés on utilise la méthode keys .

```
fiche = {"nom":"Wayne", "prenom":"Bruce"}
for cle in fiche.keys():
    print cle
```

### 10.3 .value() : Comment récupérer les valeurs d'un dictionnaire python par une boucle ?

Pour cela on utilise la méthode values .

```
fiche = {"nom":"Wayne", "prenom":"Bruce"}
for valeur in fiche.values():
    print valeur
```

### 10.4 .item() : Comment récupérer les clés et les valeurs d'un dictionnaire python par une boucle ?

Pour récupérer les clés et les valeurs en même temps, on utilise la méthode items qui retourne un tuple .

```
fiche = {"nom":"Wayne","prenom":"Bruce"}
for cle,valeur in fiche.items():
    print cle, valeur
```