

Mémo Python

1. Structure d'une fonction

```
def nom_de_la_fonction(parametre1, parametre2):  
    # tout ce qui est INDENTÉ (décalé de 4 espaces ou 1 tabulation)  
    # appartient au corps de la fonction  
    resultat = parametre1 + parametre2  
    return resultat
```

⚠ L'indentation (la tabulation) n'est pas décorative en Python : c'est elle qui délimite un bloc d'instructions. Deux lignes indentées au même niveau appartiennent au même bloc (le corps d'une fonction, d'une boucle, d'un `if...`). Une indentation fautive ou incohérente (mélange d'espaces et de tabulations) fait planter le programme (`IndentationError`). Toutes les lignes d'un même bloc doivent être alignées exactement pareil.

Appeler la fonction ensuite :

```
nom_de_la_fonction(3, 5)          # renvoie 8, sans l'afficher  
print(nom_de_la_fonction(3, 5))  # affiche 8
```

2. Conditions

```
if condition:  
    ...  
elif autre_condition:  
    ...  
else:  
    ...
```

Exemple :

```
def signe(x):  
    if x > 0:  
        return "positif"  
    elif x < 0:  
        return "négatif"  
    else:  
        return "nul"
```

Opérateurs de comparaison : `==` (égal), `!=` (différent), `<`, `>`, `<=`, `>=` Opérateurs logiques : `and`, `or`, `not`

3. Boucles

for : nombre d'itérations connu

```
for i in range(n):          # i prend les valeurs 0, 1, 2, ..., n-1
    print(i)

for i in range(1, n+1):     # i prend les valeurs 1, 2, ..., n
    print(i)

for i in range(a, b, pas):  # de a à b-1, avec un incrément de "pas"
    print(i)
```

while : on répète tant qu'une condition est vraie

```
while condition:
    ...
```

Exemple : compter de 10 en 10 tant qu'on n'a pas dépassé 100

```
n = 0
while n <= 100:
    n = n + 10
```

 **Il faut que la condition finisse par devenir fausse, sinon la boucle tourne indéfiniment (boucle infinie).**

4. Listes

```
L = []                    # liste vide
L = [1, 4, 9, 16]         # liste avec des éléments

L[0]                     # premier élément (les indices commencent à 0)
L[-1]                    # dernier élément
len(L)                   # nombre d'éléments de la liste

L.append(25)              # ajoute 25 à la fin de la liste
L[2] = 100                # modifie l'élément d'indice 2
```

Parcourir une liste

```
for element in L:         # parcourt les valeurs
    print(element)

for i in range(len(L)):   # parcourt les indices
    print(L[i])           (utile si on veut modifier L)
```

Construire une liste terme par terme (très utile pour les suites)

```
L = [1]                  # on part d'une valeur initiale
for i in range(n):
    L.append(L[-1] + 2)   # on ajoute un nouveau terme calculé à partir
                          du précédent
```

5. Opérations mathématiques

```
x + y      # addition
x - y      # soustraction
x * y      # multiplication
x / y      # division (résultat décimal)
x // y     # division entière (quotient)
x % y      # reste de la division euclidienne (modulo)
x ** y     # puissance : x élevé à la puissance y (pas x^y)
abs(x)     # valeur absolue
```

Pour aller plus loin, il faut importer le module `math` :

```
import math

math.sqrt(x)      # racine carrée
math.exp(x)       # exponentielle
math.log(x)       # logarithme népérien
math.pi           # la constante π
math.floor(x)     # partie entière (arrondi en dessous)
```

Arrondir un résultat :

```
round(x, 2)      # arrondit x à 2 décimales
```

6. Autres fonctions utiles

```
print(a, b, c)    # afficher une ou plusieurs valeurs

import random
random.random()   # nombre aléatoire décimal entre 0 et 1
random.randint(a, b) # entier aléatoire entre a et b (inclus)

import matplotlib.pyplot as plt
plt.plot(L)       # tracer les valeurs d'une liste L
plt.show()        # afficher le graphique
```

Récapitulatif

Élément	Syntaxe	Attention
Fonction	<code>def f(x) :</code> puis corps indenté	: en fin de ligne, indentation obligatoire
Condition	<code>if ... : / elif ... : / else :</code>	chaque bloc est indenté
Boucle bornée	<code>for i in range(n) :</code>	<code>range(n)</code> va de 0 à n-1
Boucle non bornée	<code>while condition :</code>	attention aux boucles infinies
Retour de valeur	<code>return valeur</code>	arrête la fonction et renvoie le résultat
Puissance	<code>x**2</code>	pas <code>x^2</code>
Division	<code>/</code> (réelle) et <code>//</code> (entière)	<code>7/2 = 3.5</code> mais <code>7//2 = 3</code>
Liste	<code>L = []</code> , <code>L.append(x)</code> , <code>L[i]</code>	indices commencent à 0