



Formation KI

Python

- 1 Base du langage
- 2 Programmation orientée objet
- 3 Modules importants

Pourquoi cette formation ?

Java/C/C++/C#/JS :
&& is and, || is or
Python:



and is and, or is or

[https://kiclubinfo.notion.site/
Formation-Python-219ab0fc71c24f50b06bee30f60f649a?pvs=4](https://kiclubinfo.notion.site/Formation-Python-219ab0fc71c24f50b06bee30f60f649a?pvs=4)



Types de variables

```
1 entier = 1
2 booleen = True
3 flottant = 1.0
4 chaine_de_caractere = "Bonjour les 1A !"
```

- ▶ int pour les **entier**
- ▶ float pour les **décimaux**
- ▶ bool pour les **booléen**
- ▶ string pour les **chaines de caractères**

Opérations et relations binaires

- ▶ $x+y$: **Addition**
- ▶ $x-y$: **Soustraction**
- ▶ $x*y$: **Multiplication**
- ▶ x/y : **Division**, renvoie (toujours) un flottant
- ▶ $x//y$: **Quotient**
- ▶ $x\%y$: **Reste**
- ▶ $x==y$: **Égalité**
- ▶ $<, <=, >$ et $>=$: **Relation d'ordre**

Listes

```
1 liste = [12,24,36,48,60] # Creation de la liste
2
3 print(liste) # Affiche l'ensemble de la liste
4
5 print(liste[0]) # Affiche la premiere valeurs de la liste (ici 12)
6 print(liste[1]) # Affiche la deuxieme valeurs de la liste (ici 24)
7 print(liste[-1]) # Affiche la derniere valeurs de la liste (ici 60)
8
9 print(len(liste)) # Affiche la longueur de la liste (ici 5)
10
11
12 liste_valide = [True, 0, 1.2] # Est valide
13 liste_contenant_une_autre_liste = [0, [True,False], 2.] # Egalement
    valide
```

Dictionnaires

```
1 {"nom": "Ponts", "naissance": 1976, "formations": ["IMI", "GMM", "GCC", "
   VET", "GI", "SEGF"]}
2
3 # Il est également possible d'ecrire un dictionnaire de la maniere
   suivante
4 {
5     1:1,
6     2:4,
7     3:9,
8     4:16,
9     5:25,
10    6:36
11 }
```


Conditions

```
1 x = 1
2 y = 2
3
4 if x==y:
5     # Debut de si (on augmente d'une tabulation)
6     print("x et y sont egaux")
7 # Fin si (on supprime la tabulation)
8 elif x>y:
9     print("x et plus grand que y")
10 else:
11     print("x et plus petit que y")
```

Boucle while

```
1 x=2
2 while x<y:
3     x = 2*x # On double la valeur de x tant qu'elle est plus petite
              que y
4 # x est donc la plus petite puissance de deux superieure a y
```

Boucle for

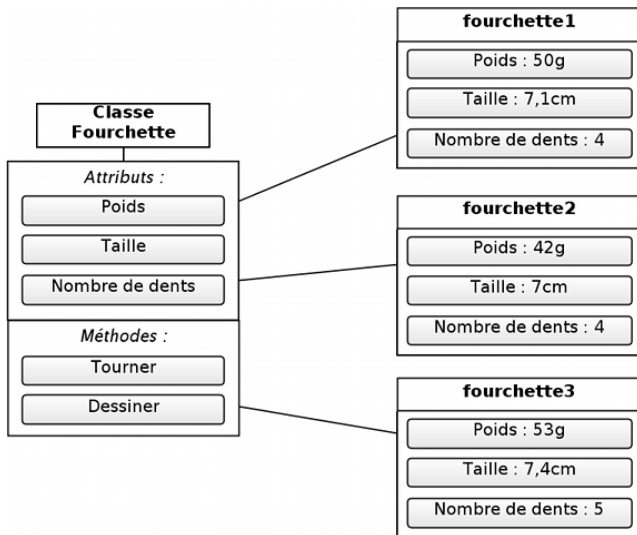
```
1 i=0
2 while i<10:
3     print(i)
4     i += 1 # Similaire a i = i+1
5
6 for i in [1,12,16,20,44]:
7     print(i)
8
9 # Le code precedent avec la boucle while peut s'ecrire :
10 for i in range(10):
11     print(i)
```

Fonctions réutilisables

```
1 def carre(x):  
2     return x**2  
3  
4 print(carre(2)) # Affiche 4
```

```
1 def addition(*args):
2     total = 0
3     for arg in args:
4         total += arg
5     return total
6
7 print(addition(1, 2, 3)) # Affiche 6
8
9 def afficher_noms(**kwargs):
10     for nom, age in kwargs.items():
11         print(f"{nom} a {age} ans")
12
13 afficher_noms(Alice=25, Bob=30, Claire=27)
14 # Affiche :
15 # Alice a 25 ans
16 # Bob a 30 ans
17 # Claire a 27 ans
```

La POO c'est quoi ?






Initialisation

```
1 class Etoile:
2     def __init__(self, masse, position, quantite_mouvement):
3         # __init__ est la fonction qui permet d'initialiser notre
4         objet
5         self.masse = masse
6         self.position = position
7         self.quantite_mouvement = quantite_mouvement
8
9 etoile1 = Etoile(10, [0, 0, 0], [0, 0, 0])
10
11 print(etoile1.masse) # Renvoie la masse de etoile1 (ici 10)
```

Méthodes

```
1 class Etoile:
2     def __init__(self, masse, position, quantite_mouvement):
3         self.masse = masse
4         self.position = position
5         self.quantite_mouvement = quantite_mouvement
6
7     def effort(self, liste_etoiles):
8         effort = [0,0,0]
9         for e in liste_etoiles:
10             effort = [effort[i] + G*self.masse*e.masse*(e.
position[i]-self.position[i])/dist2(self,e) for i in range(3)]
11         return effort
12
13     def deplacement_elem(self, pas, liste_etoiles):
14         effort = self.effort(liste_etoiles)
15         for i in range(3):
16             self.quantite_mouvement[i] += pas*effort[i]
17             self.position[i] += pas*self.quantite_mouvement
[i]/self.masse
```


Quelques modules

- ▶  **NumPy** : Bibliothèque qui permet de manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux.
- ▶  **Matplotlib** : Bibliothèque destinée à tracer et visualiser des données sous forme de graphiques.
- ▶  **Pandas** : Bibliothèque permettant la manipulation et l'analyse des données.

Utilisation (exemple avec *NumPy*)

On place au début de son script la ligne suivante : `import numpy as np`

On utilise les différentes fonction en appliquant `np.` puis la fonction (ou constructeur d'objet). Exemple : `np.array([2,3,4])` ou `np.sqrt(9)`

Merci, à vous de jouer !

`https://kiclubinfo.notion.site/`

`Formation-Python-219ab0fc71c24f50b06bee30f60f649a?pvs=4`

