Python premier pas

Suite du traitement des données

Sommaire

- Les conditions
- Les listes
- Les boucles
- Objets et méthodes
- Applications

Les conditions

if test : effectue les instructions qui suivent si le test est respecté.

On remarque que les instructions qu'on veut exécuter dans la structure conditionnelle ne sont pas alignées avec le if mais sont en retraits : on dit que ces instructions sont indentées.

- Structure conditionnelle sous Python

```
a = 'DevData'

if a == 'DevData':
    print('Vive la Data avec un grand D')
else :
    print("Ho ! Es ce qu'il y a de la data à Cannes ?")
```

```
Vive la Data avec un grand D
```

Cela effectue les instructions1 lorsque le test est vérifié, sinon effectue les instructions2.

- La ligne qui précède l'indentation se finit toujours par deux points. Appuyer sur la touche

Entrée après avoir tapé « : » effectue automatiquement l'indentation.

Les conditions

```
a = 'devdata'

if a == 'DevData':
    print('Vive la Data avec un grand D')
elif a == 'devdata':
    print('Vive la data')
else :
    print("Ho ! Es ce qu'il y a de la data à Cannes ?")
```

```
Vive la data
```

Cela effectue les instructions1 indentées lorsque le test1 est vérifié, sinon effectue le test2 et, si celui-ci est vérifié, effectue les instructions2 indentées.

Remarques :

- On peut enchaîner autant de "elif" que nécessaire.
- On peut terminer une série de "elif" par un "else" afin d'être sûr de traiter tous les cas.

Les listes

- **Liste** : Variable contenant une liste **ordonnée** d'éléments

```
maListe = [ 4.7, 53, 2019, "formation", "Montpellier" ]
print(maListe)
```

```
[4.7, 53, 2019, 'formation', 'Montpellier']
```

```
print(maListe[0])
print(maListe[1])
print(maListe[2])
print(maListe[3])
print(maListe[4])
```

```
4.7
53
2019
formation
Montpellier
```

```
maListe[0] = 'DevData'
print(maListe[0])
```

```
DevData
```

La boucle for

```
for i in maListe:
    print(i)
```

```
4.7
53
2019
formation
Montpellier
```

```
a = 'DevData'

for i in a:
    print(i)
```

```
D
e
v
D
a
t
a
```

Pour tout éléments i dans maListe :
    afficher le ième élément

Pour tout i dans la chaîne de caractère a :
    afficher la ième lettre

La boucle while

```
x = 1

while x < 10:
    print("x a pour valeur", x)
    x = x * 2

print("Fin")
```

```
x a pour valeur 1
x a pour valeur 2
x a pour valeur 4
x a pour valeur 8
Fin
```

Tant que x est inférieur à 10
    afficher « x à pour valeur », la de valeur x
    x prend la valeur x multiplié par 2

Afficher « Fin »

La fonction range()

L’instruction range(début,fin,pas) génère une suite d’entiers (les paramètres début et pas sont optionnels)

- Dans l’intervalle [0 ; fin[si un seul paramètre est renseigné. range(5) va créer la suite [0, 1, 2, 3, 4] de 5 termes
- Dans l’intervalle [début ; fin[si 2 paramètres sont renseignés
- range(1,5) va créer la suite [1, 2, 3, 4], le premier terme sera 1 et le dernier 5.
- Dans l’intervalle [début ; fin[mais de pas en pas, si les 3 paramètres sont renseignés.

La fonction range()

```
for i in range(5):
    print(i)
```

```
0
1
2
3
4
```

```
for i in range(5, 10):
    print(i)
```

```
5
6
7
8
9
```

```
for i in range(5, 10, 2):
    print(i)
```

```
5
7
9
```

```
maListe = [i**3 + 2 for i in range(10)]
print(maListe)
```

```
[2, 3, 10, 29, 66, 127, 218, 345, 514, 731]
```

Objets et méthodes

La programmation orientée objet est un style de programmation dans lequel les données et les opérations qui les manipulent sont organisées en **classes et méthodes**.

- Un objet est comparable à une variable qui appartient à une classe bien définit. **En python tout est objet.**
- Une class est un type d'objet (Chaîne de caractère, valeur numérique, liste, …) auquel on peut appliquer des méthodes qui lui sont propre après avoir fait une instanciation.
- Les méthodes sont des fonctions qui sont associées de manière explicite à une classe. Elles ont comme particularité un accès privilégié aux données de la classe elle-même.

En pratique ça marche comment

- *maListe.append(x) ajoute l'élément x à la fin de maListe*
- *maListe[i] donne le ième élément de maListe*
- *maListe.insert(i, x) insère x à la position i dans maListe*
- *maListe.remove(x) supprime le premier élément égal à x s'il y est*
- *maListe.count(x) compte le nb d'occurrences de x dans maListe*
- *maListe.sort() tri maListe*
- *maListe.reverse() renverse l'ordre de maListe*

Pratique

1. Ecrire un programme qui à partir d'une saisie d'un nombre vous dit s'il est paire ou impaire.
2. Etant donné les longueurs des cotés d'un triangle (hypoténuse, coté adjacent) écrire un programme qui vérifie si un triangle est rectangle.
3. Créer un programme qui donne le prix TTC après avoir saisie le prix HT. Ce programme doit se répéter pour pouvoir entrer plusieurs prix à la suite et ne s'arrêter que si l'utilisateur rentre 0.
4. Soit les listes L1 = [2, 7, 3, 1, 5, 7, 1] et L2 = [8, 5, 9, 4, 1, 1, 5]. Produisez une liste L3 qui contient une seul et unique fois tout les éléments de L1 et L2.
5. Ecrire un programme qui vérifie si la liste [1, 2, 3, 4, 3, 2, 1] et un palindrome.