INTRODUCTION A LA PROGRAMMATION PYTHON

#6. Liste

> **Objectifs:**
>
> - Introduire le concept de liste
> - Programmer en utilisant des listes

Nous avons vue en #2 les variables et les types de base: int , float, str et bool.

Mais nous avons parfois besoin de manipuler non pas une seule donnée mais plutôt un **ensemble de données**.

Pour cela, nous pouvons utiliser un type construit (pas de base) Python: **list**

Une liste Python est un ensemble dont les éléments sont:

- ordonnés
- indexés par la position dans la liste (le premier indice est **0**)
- d'un type de base ou d'un type construit par exemple une liste ...

Voici des exemples illustrant comment manipuler des listes:

```
# création d'une liste
maListe1 = [ ]                          # création d'une liste vide
maListe2 = [ 1 , 2 , 3 ]                # création d'une liste avec 3 données (int, int, int)
maListe3 = [ 1 , 'Hello' , True ]       # création d'une liste avec 3 données (int, str, bool)
maListe4 = [ maListe2 , maListe3 , 3 ]  # création d'une liste avec 3 données (list, list, int)
print(maListe1,maListe2,maListe3,maListe4)

# ajout d'un élément à la fin d'une liste: méthode append()
maListe1.append(7)           # on ajoute 7 (int) à  maListe1 (qui était vide)
maListe2.append(4)           # on ajoute 4 (int) à la fin de maListe2
maListe3.append(3.14)        # on ajoute 3.14 (float) à la fin de maListe3
maListe4.append(maListe2)    # on ajoute maListe2 (list) à la fin de maListe2
print(maListe1,maListe2,maListe3,maListe4)

# la longueur ( = nombre d'éléments) de la liste: méthode len()
print(len(maListe1),len(maListe2),len(maListe3),len(maListe4))

# récupération du ième élément de la liste: list[i]
print(maListe2[0],maListe2[1],maListe2[2], maListe2[3]) # premier indice: 0 ; dernier indice: len(list)-1
print(maListe4[1])                      # deuxième élément de maListe4: c'est une liste
print(maListe4[1][2])                   # troisième élément du deuxième élément de maListe4

# modification du ième élément de la liste: list[i] =
maListe2[3] = 5
print(maListe2)

# suppression du dernier élémentd'une liste: méthode pop()
maListe2.pop()
print(maListe2)

# boucle bornée sur les éléments d'une liste
for i in range(len(maListe2)):
    print(maListe2[i])
# on peut aussi itérer directement sur les éléments de la liste (au lieu des indices)
for elt in maListe2:
    print(elt)
```

```
[] [1, 2, 3] [1, 'Hello', True] [[1, 2, 3], [1, 'Hello', True], 3]
[7] [1, 2, 3, 4] [1, 'Hello', True, 3.14] [[1, 2, 3, 4], [1, 'Hello', True, 3.14], 3, [1, 2, 3, 4]]
```

```
1 4 4 4
1 2 3 4
[1, 'Hello', True, 3.14]
True
[1, 2, 3, 5]
[1, 2, 3]
1
2
3
1
2
3
```

Et ci dessous un exemple pour illustrer comment boucler sur une liste:

```python
# itération sur les indices: de 0 à len(list)-1
for i in range(len(maListe2)):
    print(maListe2[i])

# itération directement sur les éléments de la liste
for elt in maListe2:
    print(elt)
```

Mission 6.1.

> Créer une variable de type liste.
>
> Lui ajouter des informations à demander à l'utilisateur: nom, prénom et date de naissance.
>
> Puis afficher la variable.

Mission 6.2.

> On reprend la mission 5.1 en intégrant une nouvelle fonction similaire au programme de 6.1.
>
> Programmer une fonction qui demande à l'utilisateur son nom, son prénom et sa date de naissance et renvoie une liste les contenant: `inputUtilisateur( ) -> information`
>
> > paramètre: aucun
> > retour: information (list) liste contenant le nom (str), le prénom (str) et l'année de naissance (int)
>
> Reprendre la fonction qui renvoie un message de bienvenue : `messageBienvenue( nom , prenom, annee ) -> message`
>
> Puis utiliser les deux fonctions pour afficher un message de bienvenue à 10 utilisateurs.
>
> NB: nous avons vu en #5 qu'une fonction ne renvoyait qu'une seule donnée, si on veut en renvoyer plusieurs, il suffit de les intégrer à une donnée de type construit comme une liste

Mission 6.3.

> On étend la fonction de 5.2 à une liste de nombre.
>
> Programmer une fonction qui retourne une liste contenant le carré d'une liste de nombres : `carreList( listX ) -> listY`
>
> > paramètre: listX (list) liste de nombres (int ou float)
> > retour: listY (list) liste des carrés des nombres de listX (int ou float)
>
> Puis l'utiliser pour afficher le carré :
>
> > des 20 premiers entiers positifs
> > des entiers entre -10 et 10
>
> NB: vous pouvez afficher le graphique de la fonction carré avec la fonction `plot(listX, listY)` de la bibliothèque numplot

Vidéo

>