1 Listes en Python

1.1 Définition des listes

En Python, une liste est une « variable composée » (comme les chaînes de caractères par exemple). On les note **entre crochets**.

■ **Exemple 1:**

```
prenom = ["Louise", "Antoine", "Alexandre"]
```

Le nom de la liste est `prenom`, elle est composée de trois chaines de caractères. Chaque élément de la liste est indexé par un numéro. L'élément `"Louise"` est affecté de l'indice 0, `"Antoine"` de l'indice 1, etc...

■ **Exemple 2:**

Après avoir entré la liste prenom dans Python, exécuter les commandes suivantes :

```
>>> print(prenom[1])
>>> prenom[1]="Louis"
>>> print(prenom)
```

Avec cette dernière commande, observer que la liste est affichée entre crochets.

Remarque 1 (liste vide).

```
>>> a=[]
```

Définit une liste vide dont le nom est a.

La fonction `"len()"` permet d'obtenir la longueur d'une liste.

■ **Exemple 3:**

```
>>> len(prenom)
```

1.2 Opérations sur les listes

1.2.1 Accéder à un terme

Pour changer le terme d'une suite, il suffit d'utiliser la syntaxe proposée dans l'exemple 2, 2e ligne.

Remarque 2 (numérotage négatif).

Les éléments d'une liste sont numérotés en positif et négatif.

■ **Exemple 4:**

Cela est utile lorsqu'on veut appeler le dernier élément :

```
>>> print(prenom[-1])
```

1.2.2 Addition de listes

On peut ajouter deux listes :

■ **Exemple 5:**

```
>>> parents=["Arnaud", "Mélanie"]
>>> prenom=parents+prenom
>>> print(prenom)
```

1.2.3 Ajouter un élément

Pour ajouter un élément en bout de liste, on peut utiliser la commande `.append()`

■ **Exemple 6:**

```
>>> prenom.append("Alma")
>>> print(prenom)
```

Observer cette nouvelle syntaxe :

```
nomdeliste.append(élément)
```

1.2.4 Insérer un terme

On peut insérer des éléments avec la commande `.insert()`

■ **Exemple 7:**

```
>>> prenom.insert(1, "Bruno")
```

Décrire l'action de cette commande :

```
liste.insert(a,b)
```

1.2.5 Extraire un élément de la liste

On peut supprimer un terme de la liste avec la commande `.pop()`. Attention, la commande `.pop` renvoie une valeur qu'il faut stocker dans une nouvelle variable.

■ **Exemple 8:**

```
>>> frangin=prenom.pop(1)
>>> print(frangin)
>>> print(prenom)
```

La syntaxe `liste.pop(a)` renvoie le terme de rang a de la liste, et le supprime dans la liste.

1.2.6 Supprimer un élément

On peut supprimer le terme de rang a de la liste grâce à la commande `.remove(a)`

■ **Exemple 9:**

```
>>> prenom.remove(-1)
>>> print(prenom)
```

1.2.7 Test d'appartenance

Les commandes `in` et `not in` testent si des valeurs sont présentes ou pas dans une liste. Elles renvoient des booléens (Vrai ou Faux).

■ **Exemple 10:**

```
>>> "Arnaud" in prenom
>>> "Arnaud" not in prenom
>>> "Jean-Claude" in prenom
```

1.3 Parcours d'une liste

On peut utiliser les listes dans une boucle `for` :

■ **Exemple 11:**

```
>>> for i in prenom :
>>> print(i)
```

2 Exercices

▶ Exercice 1

Liste de nombres :

1. On veut créer une liste contenant les 20 premiers nombres entiers.
   ```
   entiers=[]
   for i in range(...) :
       entiers.append(...)
   ```
2. En partant d'une liste vide (`pairs=[]`), et en utilisant une boucle, générer la liste des 100 premiers nombres pairs.
3. Même consigne pour générer la liste des 100 premiers nombres impairs.
4. Que fait le programme suivant ?
   ```
   mult=[]
   for i in range(100,200) :
       if i % 7 == 0 :
           mult.append(i)
   ```

▶ Exercice 2

Génération et stockage des termes d'une suite récurrente

On considère la suite (u_n) définie pour tout $n \in \mathbb{N}$ par $\begin{cases} u_0 = 1 \\ u_{n+1} = 0,8u_n + 1 \end{cases}$.

1. Pour générer les 10 premiers termes de la suite (u_n), on peut employer une boucle `for` :

```
u=...
for i in range(...) :
    u=0.8*u+1
    print(u)
```

2. En modifiant le programme précédent, stocker les termes de (u_n) dans une liste.
3. Exécuter le programme suivant :

```
liste=[1]
for i in range(10) :
    liste.append(0.8*liste[-1]+1)
    print(liste)
```

▶ Exercice 3

Moyenne :

On considère le tableau suivant :

Notes	6	8	5	9	8
Coefficients	2	1	2	1	4

1. Créer deux listes, `notes` et `coefs` contenant les valeurs des notes et des coefficients.
2. Que fait la fonction suivante ?

```
def tot(liste):
    S=0    for i in liste :
        S=S+i    return S
```

3. Créer une fonction qui calcule la moyenne des valeurs précédentes.

▶ Exercice 4

Suite de Fibonacci

La suite de Fibonacci est définie pour tout entier n par $\begin{cases} u_0 = 1, \quad u_1 = 1 \\ u_{n+2} = u_{n+1} + u_n \end{cases}$. Ainsi, à partir du troisième, chaque terme est la somme des deux précédents.

1. Créer une liste contenant les 20 premiers termes de la suite de Fibonacci.
2. Créer la liste des quotients des termes successifs de la suite de Fibonacci. Que remarque-t-on ?