Table des matières

1 Les tableaux (list)

1.1 Bac à sable

Voici quelques exemples de tableaux :

```
>>>liste1 =["beans","spams","eggs"]
>>>liste2=[5,2,9]
>>>liste3=[1, True, "azerty", ["a","b","c"]]
>>>liste4=[]
>>>type(liste4)
```

1. Lire un élément :

```
>>>liste1[2]
>>>liste1[0]
>>>liste3[4]
```

Comment obtenir le 9 de **liste2** ?
Comment obtenir le "b" **liste3** ?

2. Modifier un élément :

```
>>>liste1[0]="chips"
>>>liste1
```

Remplacer le 9 de **liste2** par 2020.

3. Ajouter un élément :

```
>>>liste1.append("mushroom")
>>>liste1
>>>liste4.append("Coucou")
>>>liste4
```

A quel endroit sont ajoutés les éléments ?

4. Supprimer le dernier élément :

```
>>>liste1.pop()
>>>liste1
>>>liste1.pop()
>>>liste1
>>>liste1.pop()
>>>liste1
```

5. Supprimer le premier élément :

```
>>>liste1 =["beans","spams","eggs"]
>>>liste1.pop(0)
>>>liste1
>>>liste1.pop(0)
>>>liste1
>>>liste1.pop(0)
>>>liste1
```

6. Nombre d'éléments présents :

```
>>>len(liste1)
>>>len(liste2)
>>>len(liste3)
```

7. Parcourir les éléments d'une liste

```
>>> for bidule in liste1 :
        print(bidule)
```

1.2 Exercices

Exercice n° 1

```
def enigme(a,b) :
        L=[ ]
        L.append(a+b)
        L.append(a*b)
        L.append(a**b)
        return L
```

— Tester enigme(3,4)
— Justifier le 81.
— Quelle est l'intérêt d'utiliser une liste ?

Exercice n° 2

On considère le tableau L=[0,1,1,1,1,0,1,1,1,1,1,0,0,0,0,1,1,0,1,0,1,0,0,1,1,0,0,0,0,0,0,1,1,1,]

1. Écrire un algorithme en Python qui renvoie le nombre de 0 dans le tableau.

Exercice n° 3

On considère le tableau L=["z","i","n","g","z", "l", "r", "o"]

1. Ajouter un "u" puis un "f".
2. Écrire un algorithme qui affiche les éléments de L un par un.
3. Avec un algorithme, remplacer les "z" par des "e".
4. Écrire un algorithme qui renvoie un tableau donc les éléments de L ont été écrit en sens inverse.

Exercice n° 4

A l'aide du fichier : prenom.py

1. Combien y a t-il de prénoms dans le tableau ?
2. A quel indice ce trouve le prénom "Charlie" ?
3. Écrire une fonction **estPresent** qui teste si un prénom est présent dans la liste. (La réponse doit être un booléen)
4. Écrire une fonction **OuEst(prenom)** qui prend en paramètre un prénom(str) et qui affiche la phrase " prenom est caché à l'indice ..."
5. Certains prénoms sont présents plusieurs fois dans le tableau. Écrire un algorithme pour les identifier.

Exercice n° 5

Créer une liste qui contient tous les nombres pairs inférieurs à 100.

Exercice n° 6

On considère le tableau : Tab=["a","z","r","t","y","z"].
Écrire un algorithme qui insère un "e après chaque "z".

1.3 Tableaux par compréhension

1. Création d'un tableau par compréhension

```
>>>liste=[ i for i in range(10)]
>>>liste
```

Ce code est équivalent à :

```
>>>liste=[]
>>>for i in range(10) :
        liste.append(i)
>>>liste
```

2. On peut aussi ajouter une condition :

```
>>>liste=[ i for i in range(10) if i %2 ==0]
>>>liste
```

```
>>>liste=[ i for i in range(10) if i >5]
>>>liste
```

3. Autres exemples :

```
>>>liste=[ i**2 for i in range(10)]
>>>liste
```

```
>>>liste=[ i*2+1 for i in range(10)]
>>>liste
```

```
>>>liste=[ 7*i for i in range(10)]
>>>liste
```

Exercice n° 7 En utilisant la compréhension de listes ...

1. Donner la liste de tous les nombres impairs compris entre 50 et 100.

2. Donner la liste des entiers inférieurs à 50 multiples de 3.

3. Donner la liste des entiers inférieurs à 50 et qui ne sont pas multiples de 3.

4. Donner la liste des entiers inférieurs à 50 multiples de 3 **ET** pair.

5. Donner la liste des entiers inférieurs à 50 multiples de 3 **OU** pair.

6. Donner la liste des puissance de 2 inférieurs à 100000.

7. Donner la liste des entiers inférieurs à 100 et qui ne sont ni multiples de 2, ni multiples de 3, ni multiples de 5, ni multiples de 7.