Table des matières

1 Les tableaux (list)

1.1 Echauffement 1

1. Quel est le résultat affiché par ...

```
liste1 =["beans","spams","eggs"]
print(liste1[2])
```

2. Quel est le résultat affiché par ... ?

```
liste1 =["beans","spams","eggs"]
print(len(liste1))
```

3. Quel est le résultat affiché par ...

```
MonTab=[1, True, "a", ["a","b","c"], 3.14]
print(type(MonTab))
```

4. Comment obtenir le 9 de **liste2** ?

```
liste2=[5,2,9]
```

5. Quel est le résultat affiché par ...

```
liste2=[5,2,9]
print(liste1[-1])
```

6. Quel est le résultat affiché par ...

```
liste1 =["beans","spams","eggs"]
liste1[0]="chips"
print(liste1)
```

7. Quel est le résultat affiché par ...

```
liste1 =["a","z","e"]
liste1.append("r")
print(liste1)
```

8. Quel est le résultat affiché par ...

```
liste1 =["a","z","e"]
liste1.pop()
print(liste1)
```

9. Quel est le résultat affiché par ...

```
liste1 =["a","z","e"]
liste1.pop(0)
print(liste1)
```

10. Quel est le résultat affiché par ...

```
liste1 =["beans","spams","eggs"]
for bidule in liste1 :
    print(bidule)
```

1.2 Echauffement 2

1. Quel est le résultat affiché par ...

```
liste1 =["beans","spams","eggs"]
print(len(liste1)==3)
```

2. Quel est le résultat affiché par ...

```
liste1 =["beans","spams","eggs"]
print(len(liste1)=="3")
```

3. Quel est le résultat affiché par ...

```
liste1 =["a","z","e"]
liste1.append("r")
print(liste1)
```

4. Quel est le résultat affiché par ...

```
liste1 =["a","z","e"]
liste1.pop()
liste1.pop()
print(liste1)
```

5. Quel est le résultat affiché par ...

```
liste1 =["a","z","e"]
liste1.pop(0)
liste1.pop(0)
print(liste1)
```

6. Quel est le résultat affiché par ...

```
MonTab=[1, True, "a", ["a","b","c"], 3.14]
print(type(MonTab[2]))
```

7. Quel est le résultat affiché par ...

```
MonTab=[1, True, "a", ["a","b","c"], 3.14]
print(MonTab[3][2])
```

8. Quel est le résultat affiché par ...

```
liste1 =["beans","spams","eggs"]
for bidule in liste1 :
    print(bidule[0])
```

1.3 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.

Exercice n° 2

1. Créer un tableau nommé *planetes* contenant les mots "Mercure", "Venus", "Terre", "Mars", "Jupiter", "Saturne", "Uranus".
2. Faire afficher le $2^{ème}$ élément de *planetes*
3. Faire afficher le dernier élément de *planetes*
4. Faire afficher les 3 derniers éléments de *planetes*.
5. Ajouter "Neptune" et "Pluton" à *planetes*
6. Enlever "Pluton" du tableau
7. Faire afficher les éléments du tableau les uns en dessous des autres à l'aide d'une boucle.

Exercice n° 3

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° 4

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

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

Exercice n° 5

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. Certains prénoms sont présents plusieurs fois dans le tableau. Écrire un algorithme pour les identifier.

Exercice n° 6

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

Exercice n° 7

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

Exercice n° 8

1. Créer un tableau (9x9) dont la première ligne est la table de 1, la deuxième ligne est la table de 2, etc
2. Afficher les tables comme ci-dessous :

```
1 2 3 4 5 6 7 8 9
2 4 6 8 10 12 14 16 18
3 6 9 12 15 18 21 24 27
4 8 12 16 20 24 28 32 36
5 10 15 20 25 30 35 40 45
6 12 18 24 30 36 42 48 54
7 14 21 28 35 42 49 56 63
8 16 24 32 40 48 56 64 72
9 18 27 36 45 54 63 72 81
```

Aide : pour ne pas revenir à la ligne et ajouter un espace après un print on peut ajouter un paramètre : `print("Montest", end=" ")`

1.4 Tableaux par compréhension

Quel est le résultat affiché ?

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

1. Création d'un tableau par compréhension

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

2. On peut aussi ajouter une condition :

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

```
liste=[ 2*i for i in range(10) if i >5 and i<10]
print(liste)
```

3. Autres exemples :

   Dans chaque cas dire ce que contiennent les listes ci-dessous :

```
# a
[i**2 for i in range(10) if i**2<50]
# b
[i*2+1 for i in range(10)]
# c
[7*i for i in range(10)]
```

Exercice n° 9 Créer en compréhension le tableau des ...

1. ... nombres impairs compris entre 50 et 100.
2. ... des carrés des 20 premiers entiers.
3. ... entiers multiples de 3 inférieurs à 50.
4. ... entiers inférieurs à 50 et qui ne sont pas multiples de 3.
5. ... entiers inférieurs à 50 multiples de 3 **ET** pair.
6. ... entiers inférieurs à 50 multiples de 3 **OU** pair.
7. ... puissance de 2 inférieurs à 100000.
8. ... entiers inférieurs à 100 et qui ne sont ni multiples de 2, ni multiples de 3, ni multiples de 5, ni multiples de 7.

2 Chaînes de caractères

Exercice n° 10 Echauffement

1. Tester les commandes ci-dessous :

```
>>>texte="Bonjour TOUT le monde"
>>>texte[0]
>>>len(texte)
>>>texte=texte + "!"
>>>texte
>>>for c in texte :
        print(c)
```

2. Tester les commandes ci-dessous :

```
>>>mot="Bonjour tout le monde!"
>>>mot.split() >>>mot.split(" ") #Il y a un espace entre les guillemets
>>>mot.split("o")
```

3. Quel est le rôle de la méthode split ?
4. Que fait l'algorithme ci-dessous ?

```
>>>mot="Bonjour tout le monde!"
>>>for c in texte :
            if c=="o" :
                cpt=cpt+1
```

5. Que fait l'algorithme ci-dessous ?

```
>>>for c in texte :
            if c=="o" or c=="O" :
                cpt=cpt+1
```

Exercice n° 11

J'ai plaqué mon chêne Comme un saligaud, Mon copain le chêne, Mon alter ego ,On était du même bois Un peu rustique, un peu brut, Dont on fait n'importe quoi Sauf, naturell'ment, les flûtes... J'ai maint'nant des frênes, Des arbres de Judée, Tous de bonne graine, De haute futaie... Mais, toi, tu manque' à l'appel, Ma vieill' branche de campagne, Mon seul arbre de Noël, Mon mât de cocagne ! Auprès de mon arbre, Je vivais heureux, J'aurais jamais dû m'éloigner de mon arbre... Auprès de mon arbre, Je vivais heureux, J'aurais jamais dû le quitter des yeux...

Pour le copier, utiliser le lien : Brassens

1. Combien ce texte contient-il de caractères ?
2. Combien ce texte contient-il de "e" ?
3. Combien ce texte contient-il de voyelles ?
4. Combien ce texte contient-il de mots ?
5. Quel est le pourcentage de mots qui contiennent la lettre "e" ?
6. Créer un dictionnaire donc les clefs sont les caractères du texte et les valeurs le nombre d'occurrences.

Exercice n° 12

1. Écrire une fonction qui prend en paramètre une chaîne de caractère et qui inverse celle-ci.

```
>>>inverse("Hello")
olleH
```

2. Application 1 : Tester pour décoder le texte : mystère
3. Application 2 : Ecrire une fonction qui tester si un mot ou une phrase est un palindrome.
   Tester avec :
   — kayak
   — mon nom
   — engage le jeu, que je le gagne.

3 Les dictionnaires

Lien vers le TP : Les dictionnaires

4 Les piles et les files

4.1 Cours

Lien

4.2 Utilisation des piles

Exercice n° 13 Implémentation d'une pile par un tableau

Rappel : La pile est une structure de type LIFO (Last In, First Out), on accède uniquement au sommet et la première valeur accessible est la dernière ajoutée. Pour manipuler une pile, on définit les fonctions suivantes :

- ***estVide*** : teste si la pile est vide (retourne un booléen)
- ***afficherPile*** : affiche les éléments de la pile.
- ***afficherSommet*** : affiche le sommet de la pile.
- ***empiler*** : ajoute un élément au sommet de la pile
- ***depiler*** : retourne lobjet situé au sommet de la pile et le retire

1. Écrire et tester la fonction est ***estVide***

```
>>>tab1=[]
>>>tab2=["p","y","t","h","o"]
>>> estVide(tab1)
True
>>>estVide(tab2)
False
```

2. Écrire et tester la fonction est ***Sommet***

```
>>>Sommet(tab2)
o
```

3. Écrire et tester la fonction est ***empiler***

```
>>>empiler(tab2,"n")
["p","y","t","h","o","n"]
```

4. Écrire et tester la fonction est ***depiler***

```
>>>depile(tab2)
["p","y","t","h","o"]
>>>depile(tab2)
["p","y","t","h"]
>>>depile(tab2)
["p","y","t"]
```

5. Écrire et tester la fonction est ***afficherPile***

```
>>>afficherPile(tab2)
o
h
t
y
p
```

6. Écrire une fonction qui permet de tester si un élément est présent dans une pile.

```
>>>estPresent(tab2,"y")
True
>>>estPresent(tab2,"a")
False
```

Exercice n° 14

1. En utilisant une structure de pile, écrire une fonction qui prend en paramètre une chaîne de caractère et qui inverse celle-ci.

```
>>>inverse("Hello")
olleH
```

Exercice n° 15

Nous allons étudier ici la vérification des délimiteurs dans une expression.
On distingue ici trois délimiteurs, les parenthèses"(" et ")", les accolades "{" et "}" et les crochets "[" et "]". L'objectif est de déterminer si une chaîne de caractères (qui pourrait être un programme) est correctement délimitée, c'est-à-dire si chacun des délimiteurs ouvrant est suivi par un délimiteur fermant.

1. Les exemples suivants sont-ils bien délimitées ?

   a(b)c ; a[b(c)}d ; {a(b[c[d))]e]} ; {a(b(d[d]))e}

2. Est-il possible de vérifier une chaîne uniquement en comptant les délimiteurs ? Si non, proposez un contre-exemple.
3. Écrivez une fonction ***verifDelim*** qui prend en paramètre une chaîne de caractère est qui renvoie un boolean indiquant si la chaîne est correctement délimitée. Cette fonction ne considérera que les délimiteurs "(" et ")"
4. Écrivez une fonction ***verifDelimStack*** qui réalise vérification de délimiteurs à partir d'une PILE.
5. **[BONUS]** Modifiez votre algorithme de manière à afficher les indices des caractères qui posent problème, par exemple lorsqu'il manque un délimiteur fermant ou qu'il ne s'agit pas du délimiteur fermant correspondant au délimiteur ouvrant.

Exercice n° 16 Notation polonaise inversée : NPI

Avecla NPI,les opérandes (nombres) précèdent l'opérateur (+, -, /,*).
Par exemples :
2 3 + signifie 2 + 3 ou bien 1 2 + 3 4 − ∗ signifie $(1+2)*(3-4)$
Pour effectuer le calcul d'une NPI, on applique l'algorithme suivant :
On lit un élément à la fois.

- Si c'est un opérande (un entier), on l'empile
- Si c'est un opérateur binaire, on dépile deux éléments sur lesquelles on applique l'opérateur et on empile le résultat.

Appliquer cet algorithme pour calculer les expressions ci-dessous :

1. 4 5 ∗ 2 ∗ 1 +
2. 7 3 + 7 3 3 ∗ + −
3. 5 9 3 + 4 2 ∗ ∗ 7 + ∗
4. 3 7 + 6 5 − ∗ 4 2 − 2 3 + ∗ /

Remarques : L'avantage de cette notation est que les calculs peuvent être écrit sans parenthèses.

4.3 Utilisation des files

Exercice n° 17 Implémentation d'une file par tableau

Rappel : La file est une structure de type FIFO (First In, First Out), on accède uniquement au début de la file et la première valeur accessible est la première ajoutée. Pour manipuler une file, on définit les fonctions suivantes

— ***estVide*** : teste si la file est vide (retourne un booléen)
— ***tete*** : affiche la tête de la file.
— ***afficherFile*** : affiche les éléments de la file.
— ***emfiler*** : ajoute un élément au sommet de la file
— ***defiler*** : retourne l'objet situé au sommet de la file et le retire

1. Écrire et tester la fonction est ***estVide***

```
>>>tab1=[]
>>>tab2=["p","y","t","h","o"]
>>> estVide(tab1)
True
>>>estVide(tab2)
False
```

2. Écrire et tester la fonction est ***tete***

```
>>>tab=["H", "e" ,"l", "l"]
>>>tete(tab)
"H"
```

3. Écrire et tester la fonction est ***afficherFile***

```
>>>tab=["H", "e" ,"l", "l"]
>>>afficherPile(tab) "Hell"
```

4. Écrire et tester la fonction est ***emfiler***

```
>>>emfiler(tab,"o")
["H", "e" ,"l", "l", "o"]
```

5. Écrire et tester la fonction est ***defiler***

```
>>>depile(tab)
["H", "e" ,"l", "l"]
>>>depile(tab)
["H", "e" ,"l"]
>>>depile(tab)
["H", "e"]
```

6. Écrire une fonction qui permet de tester si un élément est présent dans une file.

```
>>>tab="azerty"
>>>estPresent(tab,"y")
True
>>>estPresent(tab,"u")
False
```

7. Écrire une fonction qui permet de compter le nombre d'éléments dans une file.