**TD5 : Tableaux/Listes**

**En Java** les tableaux sont de taille fixe, et homogènes (ils ne contiennent que des valeurs de même type).

Les cases sont indicées par des entiers, commençant par la case numero 0.

*Exemple* int [] T ; on déclare un tableau T contenant des entiers

 T= new int [10] ; on crée un tableau de 10 cases numérotées de 0 à 9

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| null | null | null | null | null | null | null | null | null | null |

Pour accéder à une case, (pour l’affecter par exemple), on utilise T[i] avec i le numéro de la case.

T[3]=12 ;

On obtient :

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| null | null | null | 12 | null | null | null | null | null | null |

Si on veut rajouter une case, c’est impossible

**En Python**, on ne déclare pas les tableaux, leur taille n’est donc pas fixée à l’avance.

*Exemple 1* : T=[ ] on a crée un tableau, qui ne contient rien, qui s’appelle T

*Exemple 2* : T=[8,5,2]

Saisir dans le shell :

T=[8,5,2]

len(T) la fonction len( ) sert à ……………………………………………….

print(T) la fonction print( ) sert à ……………………………………………….

T.append(10) la méthode append( ) sert à ………………………………………………

Lorsqu’on ajoute une valeur dans notre tableau, elle vient systématiquement se placer en dernière position. Il est donc impossible de laisser une « case vide » ou d’intercaler simplement des valeurs. On a tout intérêt à les saisir dans l’ordre. Chaque position est repérée par un indice entier (qui commence à 0).

print( T[1] ) ……….

On peut changer une valeur du tableau :

T[2]=1

print(T) ……………………………..

Les types contenus dans les tableaux Python ne sont pas nécessairement homogènes, on parle donc plus volontiers de liste.

L=["mardi", 11 , "décembre", 2012]

**Exercices:**

**exercice 1:**

On donne la suite $u\_{n}=n^{2}+1$

1-a)Ecrire un algorithme qui calcule et stocke dans une liste L les 10 premières valeurs de $(u\_{n})$.

b)Implémentez le en Python

2-a)Que fait l'algorithme suivant? Pourquoi commence-t-on à i=1

variables: L liste

 i,p entiers

début:

croissante ← True

p ← longueur(L)

Pour i allant de 1 à p-1 répéter:

 Si L[i]-L[i-1] <0 alors:

 croissante ← False

 finSi

fin Pour

Sortie: croissante

fin

b)Implémentez le en Python

c)Cet algorithme peut être optimisé car en effet, des que l'un des calculs a donné un résultat négatif, on est assuré que la suite n'est pas croissante. Il faut donc modifier la boucle Pour en une boucle Tant que (i≤p-1 et croissante=True).

3- Adapter cet algorithme afin de tester non plus si la suite peut être croissante mais si elle peut être arithmétique.

**exercice 2:**

On donne la suite $\left\{\begin{array}{c}u\_{0}=1\\u\_{n+1}=2u\_{n}+1\end{array}\right.$

a)Ecrire un algorithme qui calcule et stocke dans une liste L les 10 premières valeurs de $(u\_{n})$.

b)Implémentez le en Python

**exercice 3:**

Algorithme mystère

variables: motinitial, motfinal listes

 i entier

début:

motinitial=["m","a","r","d","i"]

motfinal=[ ]

Pour i allant de 4 à 0 répéter:

 ajouter à la liste motfinal la valeur motinitial[i]

fin Pour

Afficher motfinal

fin

1-Que fait cet algorithme? Implémentez le en Python.

2-Adaptez le afin que le mot initial soit saisi par l'utilisateur.

3-Transformons le en une fonction qui prend en entrée un mot (sous forme de liste) et retourne le mot final.

On donne la structure d'une fonction en Python:

*def nomdelafonction (paramètres d'entrées): ici def mystere (motinitial):*

 *instructions de la fonction instructions*

 *return valeuràretourner return motfinal*

On peut désormais utiliser cette fonction

print( mystere(["h","e","l","l","o"])) ……………………………………………

Utilisez cette function pour créer une nouvelle fonction qui prend en entrée un mot (sous forme de liste) et teste si c'est un palindrome ( est identique a l'endroit et à l'envers, comme ROTOR, RADAR…)