NSI, leçon : les chaînes de caractères.

1.Définition et méthodes

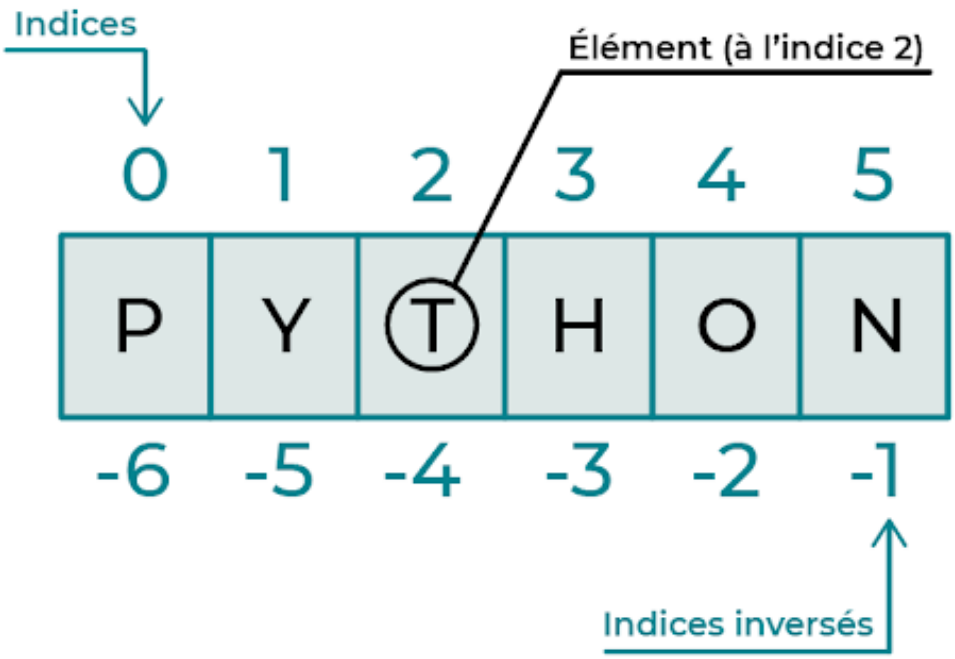
En python on définit une chaîne de caractères à l'aide de guillemets :

```
>>> texte="L'art de la programmation"
>>> type(texte)
<class 'str'>
```

On préferera les doubles guillemets pour déclarer une chaîne de caractères aux simples guillemets.

Les carctères d'une chaîne de caractères sont repérés par leur **indice ou index**.

Le 1er caractère a pour index ou indice : 0, Le second caractère a pour index ou indice : 1.....



```
>>> chaine="L'art de la programmation"
>>> chaine[0]
'L'
>>> chaine[2]
'a'
>>> chaine[1]
"'"
>>> len(chaine) # donne la longueur de la chaine soit son nombre de caractères
25
>>> chaine[24] # donne le dernier caractère
'n'
>>> chaine[-1] # donne également le dernier caractère
'n'
```

Une chaîne de caractères n'est pas **mutable** en Python, c'est à dire que l'**on ne peut pas la modifier**.

```
>>> texte="NSI"
>>> texte[0]
'N'
>>> texte[0]="P"
Traceback (most recent call last):
  File "<pyshell>", line 1, in <module>
TypeError: 'str' object does not support item assignment
```

Une chaîne de caractères est **itérable** en python, c'est à dire que l'**on peut la parcourir** à l'aide d'une boucle **for**

Par exemple, si on considère la fonction suivante :

```
def parcourir_chaine(texte):
    for caractere in texte:
        print(caractere)
```

On a alors :

```
>>> parcourir_chaine("AZERTY")
A
Z
E
R
T
Y
```

On peut parcourir également une chaîne de caractères à l'aide de ses indices :

```
def parcourir_chaine_indice(texte):
    for indice in range(len(texte)):
        print(texte[indice])
```

```
>>> parcourir_chaine_indice("AZERTY")
A
Z
E
R
T
Y
```

2. Quelques méthodes qui peuvent servir

```
>>> a="123"
>>> type(a)
<class 'str'>
>>> int(a) # transforme une chaîne de caractère en entier
123
>>> str(123) # transforme un int en str
'123'
>>> float("12.2")
12.2
>>> str(12.2)
'12.2'
```

3. La découpe de chaîne de caractères

```
>>> chaine="l'art du slice"
>>> chaine[2:5]
'art'
>>> chaine[:5]
"l'art"
>>> chaine[5:]
' du slice'
>>> chaine[6:8]
'du'
```

4. Exercices

On considère les fonctions suivantes, détermier le retour des commandes associées :

```
def fonction1(texte):
    retour=""
    for caract in texte:
        retour=caract+retour
    return retour
```

```
>>> fonction1("azerty")

```

```
def fonction2(texte):
    retour=0
    for caract in texte:
        retour+=1
    return retour
```

```
>>> fonction2("azerty")

```

```
def fonction3(n):
    retour=""
    for i in range(n):
        retour=retour+"*"
    return retour
```

```
>>> fonction3(4)

```

```
def fonction4(texte,caractere):
    s=0
    for carac in texte:
        if carac==caractere:
            s+=1
    return s
```

```
>>> fonction4("L'art de la programmation","a")

>>> fonction4("L'art de la programmation","b")

```

4. TP

On peut utiliser les chaînes de caractères pour stocker des données comme en.csv par exemple.

On considère la fonction suivante qui permet de déterminer si un nombre est premier :

```
import math
def est_premier(nombre):
    if nombre==1:
        return False
    for diviseur in range(2, int(math.sqrt(nombre))+1):
        if nombre%diviseur==0:
            return False
    return True
```

On peut l'utiliser pour créer une fonction qui permettra de déterminer la liste des nombres premiers jusqu'à une certaine valeur.

```
def liste_nombres_premier(n):
    '''
    retourne la liste des nombres premiers inférieurs
    ou égaux à n
    : entrée : n : int
    : sortie : str
    >>> liste_nombres_premier(11)
    '2,3,5,7,11'
    >>>liste_nombres_premier(100)
    '2,3,5,7,11,13,17,19,23,29,31,37,
    41,43,47,53,59,61,67,71,73,79,83,89,97'
    '''
```

Compléter cette fonction pour qu'elle réponde à sa doctring.