Entrainement sur les boucles while, appliqué aux chaines de caractère.

A partir des types de base, se constituent des types construits. On effectue des ensembles de valeurs, lorsque les données sont nombreuses ou bien parfois il est intéressant d'en regrouper certaines.

Partons de la chaîne de caractères pour aller jusqu'à la notion de liste.

Partie 1 : la chaine de caractères

La chaîne de caractères est constituées d'entités plus petites, les caractères. Selon les cas, on pourra traiter la chaine de caractères comme un seul objet ou bien comme une suite ordonnée d'éléments. Et dans le cas de la suite d'éléments, nous pourrions désirer accéder à chaque élément, ici un caractère, à titre individuel.

En fait, les chaînes de caractères font partie d'une catégorie d'objets Python que l'on appelle des *séquences*, et dont font partie aussi les *listes* et les *tuples*. On peut effectuer sur les séquences tout un ensemble d'opérations similaires.

Opération1 : l'indiçage et l'extraction

a) Que renvoie python lorsqu'on saisit les lignes du programme suivant ? Noter le résultat.

```
>>> nom = 'Informatique'
>>> print(nom[1], nom[5], nom[8])
```

Dans cet exemple on voit que chaque caractère de la séquence occupe une position bien définie dans la chaîne et que la numérotation de cette position démarre à …….. la position est indiquée entre ……

b) On peut aussi indicer (désigner l'emplacement d'un caractère) depuis le dernier caractère, de la façon suivante :

```
>>> nom = 'Informatique'
>>> print(nom[-1], nom[-2], nom[-4], nom[-6])
```

Saisir ces lignes dans python et noter le résultat.

Dans cet exemple, on voit que la position [-1] correspond au ……..caractère ; [-2] correspond ……

c) On peut extraire une partie de la chaine

Que renvoie python lorsqu'on saisit les lignes du programme suivant ? Noter le résultat.

```
>>> nom = 'Informatique'
>>> print(nom[0:3])
>>> print(nom[:3])
>>> print(nom[3:])
```

Opération 2 : déterminer la longueur de la chaîne : à l'aide de la fonction intégrée len ()

Que renvoie python lorsqu'on saisit les lignes du programme suivant ? Noter le résultat.

```
>>> nom = 'Informatique'
>>> print(len(nom))
```

Opération 3 : la concaténation : obtenir une chaine plus longue à partir de chaines plus courtes

Que renvoie python lorsqu'on saisit les lignes du programme suivant ? Noter le résultat.

```
>>> a = 'Sciences'
>>> b = ' Informatiques'
>>> c = a + b
>>> print (c)
```

Remarque : L'espace est considéré comme un caractère.

Opération 4 : la répétition

Que renvoie python lorsqu'on saisit les lignes du programme suivant ? Noter le résultat.

```
>>> a = 'Numérique '
>>> b = a * 3
>>> print (b)
```

Opération 5 : convertir une chaine de caractères en nombre entier int () ou nombre à virgule float ()

Que renvoie python lorsqu'on saisit les lignes du programme suivant ? Noter le résultat.

```
>>> nombreeleves ='26'
>>> print (1 + nombreeleves)
```

Ou

```
>>> nombreeleves = int ('26')
>>> print (1 + nombreeleves)
```

Entrainement sur les boucles while, appliqué aux chaines de caractère.

Exercices à faire d'abord sur papier (écrire algorithme) puis à tester sur python.

N°1 : En utilisant une boucle WHILE, écrire un script qui détermine si une chaîne de caractères contient ou non la caractère « e »

```
chaine = "nsiforever"

# vous devez faire deux affectations ici avant de commencer la boucle

while
```

```
if ……

    print("Le caractère est présent")

else:

    print("Le caractère est introuvable")
```

N°2 : En modifiant assez peu le script précédent, Ecrire un script qui compte le nombre d'occurrences du caractère « e » dans une chaîne.

N°3 : En utilisant une boucle WHILE , érire un script qui recopie une chaîne (dans une nouvelle variable), en insérant des astérisques entre les caractères. Par exemple « NSI » deviendra « N*S*I ». Votre algorithme doit fonctionner pour n'importe quelle longueur de mot.

Remarque 1 : il existe des « méthodes » qui permettent de remplacer certains des programmes demandés précédemment

Par exemple : **pour vérifier si la chaine contient 'e' on peut utiliser :**

```
#exercice 1 (la chaine contient-elle "e"?)
chaine = "bonjour"
caractere = 'e'
# l'instruction in permet de determiner la présence d'une chaine de
caractère dans une autre chaine de caractère.
if (caractere in chaine):
     print (chaine, "contient", caractere)
else:
     print (chaine, "ne contient pas", caractere)
```

Pour compter le nombre d'occurrences 'e', on peut utiliser :

```
#exercice 2 (.count pour compter le nombre de "e" dans la chaine)
chaine = input ('Saisir une chaîne de caractères : ')
caractere = 'e'
nombre=chaine.count('e')
print ('le nombre de', caractere, 'dans', chaine, 'est égal à ',
nombre)
```

Remarque 2 : on ne peut pas remplacer un caractère par un autre dans une chaine de caractères

```
chaine = 'bonjour'
chaine[0]='B'
print (chaine)
```

une erreur est renvoyée TypeError: 'str' object does not support item assignment

On peut réussir en créant une nouvelle chaine, dans laquelle figurera le caractère remplaçant et le reste de la chaine.

```
chaine = 'bonjour'
chaine = 'B' + chaine[1:]
print (chaine)
```

Ou bien on transforme la chaine de caractères en liste avec **list(chaine),** on peut modifier cette liste et on retransforme la liste en chaine avec **''.join(liste)**

```
chaine = 'bonjour'
liste=list(chaine)
liste[0]='B'
chainebis = ''.join(liste)
print(liste , chainebis)
```

(On peut vérifier sous quelle forme apparait le mot bonjour lorsque la chaine est transformée en liste ; on peut aussi vérifier le type renvoyé pour chacune des étapes de transformation)