PARCOURS D'UNE CHAINE DE CARACTERES ET RECHERCHE D'UN CARACTERE DANS LA CHAINE

Niveau : Première NSI

Durée : 1h

Objectifs :

- Recherche d'un élément dans une donnée indexée (chaîne de caractères)
- Parcours séquentiel d'une chaîne de caractères
- Etude du coût

Prerequis :

- Représentation d'un caractère
- Constructions élémentaires de Python : Affectation, séquences conditionnelles, boucles, fonctions
- Notions de compléxité

1 - Chaînes de caractères

On appelle **caractère** tout symbole qui peut être écrit :

- les lettres de l'alphabet latin : abcd...xyzABCD...XYZ
- les chiffres décimaux : 0123456789
- les symboles de ponctuation (y compris l'espace) : .,;:!?
- les symboles de parathésage : (){}[]
- et bien d'autres caractères comme les lettres accentuées àéùèÀÈ... et les lettres d'autres alphabets ou caratères spéciaux 千ʒ¢©§

Un **chaîne de caractère** (*string* en anglais) est une séquence de caractères, c'est-à-dire des caractères qui se suivent les uns derrière les autres. Une **chaîne vide** est une chaîne qui ne contient aucun caractère.

En python, les chaînes de caratères doivent être placées entre **quotes** :

- apostrophes simples (')
- guillemets doubles (")
- les triples apostrophes(''')
- ou des triples guillemets (""")

In [ ]:

```
chaine1 = 'Bonjour'
chaine1
```

In [ ]:

```
chaine2 = "Aujourd'hui il fait chaud"
chaine2
```

</br>

Il existe des caractères particuliers qui n'entrainent aucun affichage. On trouve entre autres :

- **\n** permet d'insérer des **sauts à la ligne**

In [ ]:

```
print("Une chaîne de caractères \nsur plusieurs \nlignes")
```

</br>

- **\t** permet d'insérer des marques de **tabulation** dans une chaîne

In [ ]:

```
print ("Une chaîne\t avec \ttabulations")
```

Pour connaître la taille d'une chaîne de caractères, il est possible d'utiliser, sous Python, la **fonction len()**.

In [ ]:

```
print(len(''))
print(len('a'))
chaine = "Bienvenue en NSI !"
print(len(chaine))
```

2 - Parcourir une chaîne de caractères

2.1 - Notation indicielle

Dans Python, une chaîne est manipulée comme une séquence indexée de caractères. Ainsi, **chaque caractère est accessible directement par son index ou indice**.

Le premier caractère (le plus à gauche) d'une chaîne de **longueur** n a pour **indice 0** et le dernier l'**indice** $n-1$.

Question : Justifier les résultats obtenus pour les lignes de code suivantes :

In [ ]:

```
chaine = "Bienvenue en NSI !"
print(chaine[0])
print(chaine[5])
print(chaine[17])
print(chaine[20])
```

En plus de cet accès unitaire aux caractères, il est possible d'accéder à des sous-chaînes en précisant la tranche souhaitée par l'indice de son premier caractère et l'indice du caractère suivant le dernier caractère de cette tranche.

Question : Justifier les résultats obtenus pour les lignes de code suivantes :

In [ ]:

```
chaine = "Bienvenue en NSI !"
print(chaine[0:8])
print(chaine[:8])
print(chaine[13:])
```

2.2 - Parcourir une chaîne de caractères

Il est très souvent nécessaire de parcourir une chaîne de caractères afin d'obtenir la réponse à un problème donné. Par exemple, si on veut compter le nombre d'espaces, le nombre d'occurences de tel ou tel caractère.

Question : Expliquer le rôle de la fonction suivante

In [ ]:

```
def parcours1(chaine,caractere) :
    for i in range (0,len(chaine)) :
        if chaine[i] == caractere :
            return True
    return False
```

Question : Tester la fonction pour les arguments suivants. Conclure sur le rôle de la fonction **parcours1**.

In [ ]:

```
parcours1('Bonjour','n')
```

In [ ]:

```
parcours1('Bonjour',' ')
```

In [ ]:

```
parcours1('Bienvenue en NSI !',' ')
```

Question : Modifier le programme précédent afin de retourner l'indice de la première occurence d'un caractère dans la chaîne. Si le caractère n'est pas dans la chaîne il faudra retourner -1.

In [ ]:

```
def parcours2(chaine,caractere) :
    for i in range (0,len(chaine)) :
        if chaine[i] == caractere :


```

Question : Tester la fonction pour les cas suivants.

In [ ]:

```
parcours2('Bienvenue en Numérique et Sciences Informatiques !','z')
```

In [ ]:

```
parcours2('Bienvenue en Numérique et Sciences Informatiques !','B')
```

In [ ]:

```
parcours2('Bienvenue en Numérique et Sciences Informatiques !','n')
```

Question : Quel cas (cas le pire), le nombre d'opérations élémentaires réalisées est maximal ? Donner la valeur maximale de ce nombre d'opérations pour les cas précédents

Réponse :

Question : Donner l'ordre de grandeur de la compléxité de l'algorithme **"parcours2"**.

Réponse :

Question : Modifier la fonction **"parcours2"** afin de comptabiliser le nombre de boucles exécutées par la fonction.

In [ ]:

```
def parcours3(chaine,caractere) :

    for i in range (0,len(chaine)) :

        if chaine[i] == caractere :
            return i,
    return -1,
```

Question : Tester la fonction pour un pire des cas et justifier l'ordre de la compléxité donné plus haut.

In [ ]:

```
parcours3('Bienvenue en Numérique et Sciences Informatiques !','z')
```

Réponse :

3 - Recherche de l'occurence d'un caractère dans une chaîne de caractères

La fonction **occurence(chaine,caractere)** permet de retourner le nombre d'occurence du caractère dans la chaîne passés en paramètre.

Question : Compléter, ci-dessous, la fonction "occurence".

In [ ]:

```
def occurence(chaine,caractere) :

    for i in range (0,len(chaine)) :
        if chaine[i] == caractere :

    return nb_occurence
```

Question : Tester la fonction pour différents cas.

In [ ]:

```
occurence('Bienvenue en Numérique et Sciences Informatiques !','z')
```

In [ ]:

```
occurence('Bienvenue en Numérique et Sciences Informatiques !','n')
```

In [ ]:

```
occurence('Bienvenue en Numérique et Sciences Informatiques !','i')
```

Question : Donner l'ordre de grandeur de la compléxité dans le pire des cas.

Réponse :

4 - Itérabilité des chaînes de caractères

Sous Python, une chaîne de caractère est un **objet itérable** c'est-à-dire qu'il est possible de la **parcourir directement à l'aide d'une boucle for**.

Le programme de recherche de caractère dans une chaîne, **"parcours1"**, peut alors s'écrire sous la forme :

In [ ]:

```
def parcours1_V2(chaine,caractere) :
    for lettre in chaine :
        if lettre == caractere :
            return True
    return False
```

Question : Tester la fonction pour différents arguments

In [ ]:

```
parcours1_V2('Bonjour','n')
```

In [ ]:

```
parcours1_V2('Bonjour',' ')
```

In [ ]:

```
parcours1_V2('Bienvenue en NSI !',' ')
```

Question : Sur le même principe modifier le programme **"parcours2"** qui permet de retourner l'indice de la première occurence. Tester son bon fonctionnement.

In [ ]:

```
def parcours2_V2(chaine,caractere) :
```

In [ ]:

```
parcours2_V2( , )
```

In [ ]:

```
parcours2_V2( , )
```

Question : Toujours sur le même principe modifier le programme "occurence".

In [ ]:

```
def occurence_V2(chaine,caractere) :
```

Question : Tester la fonction pour différents cas.

In [ ]:

```
occurence_V2( , )
```

In [ ]:

```
occurence_V2( , )
```

In [ ]:

```
occurence_V2( , )
```

5 - Pour aller plus loin

Question : Modifier le programme **"parcours2_V2"** qui permet de retourner les indices de chacune des occurences du caractère dans la chaîne. Tester le programme.

In [ ]:

```
def parcours2_V3(chaine,caractere) :
```

In [ ]:

```
recherche2_V3( , )
```

In [ ]:

```
recherche2_V3( , )
```

Question : Modifier le programme "occurence_V2" qui permet de retourner les occurences de chacun des caractères présent dans la chaîne. Tester le programme.

In [ ]:

```
def occurence_V3(chaine) :
```