Les chaînes de caractères

Objectifs

- parcourir et comparer des chaînes de caratères
- copier les éléments d'une chaîne de caratères
- approfondir la concaténation de chaînes
- approfondir les structures de contrôle : CHOISIR, PARCOURIR, RÉPÉTER
- découvrir les fonctions natives du langage pour manipuler les chaînes
- approfondir la notion de fonctions

Connaissances et savoir-faire à maîtriser

1) Définition

Une chaîne de caractère peut être vue comme un tableau indéxé par la position de la lettre dans la chaîne. Ces tableaux (on parle de **séquences** en Python) commencent à l'index 0 :

Index	0	1	2	3	4	5
Chaîne	p	y	t	h	o	n

Ainsi, l'accès à la lettre `t` s'effectue via l'instruction `chaine[2]` .

2) Parcourir une chaîne

On récupère la longueur d'une chaîne via une fonction native du langage Python : `len(chaine)` . On peut alors énumérer chaque lettre de la chaîne en utilisant une boucle FOR :

```
chaine = "Python"
for index in range(0, len(chaine)):
    print(chaine[index])
for index in range(len(chaine) - 1, -1, -1): # parcours inversé de 6 à 0
    print(chaine[index])
```

Remarque 1 *: Un parcours inversé part de la fin de la chaîne, donc de l'index* `len(chaine) - 1` *pour finir sur l'index* 0 *(rappelez-vous que la borne indiquée en second paramètre n'est pas comptée dans une boucle FOR). Le* −1 *en* 3^{ième} *paramètre correspond au pas (ici* −1*).*

On peut aussi parcourir une chaîne en utilisant un raccourci syntaxique permis par le langage Python :

```
for lettre in "Python":
    print(lettre)
```

3) Comparer deux chaînes

On utilisera pour cela les opérateurs classiques de comparaison ==, !=, <, >. Ces opérateurs sont basés sur l'ordre lexicographique qui est une extension de l'ordre alphabétique tenant compte des différences entre majuscules et minuscules, chiffres et nombres, ponctuation, ...

```
"toto" == "Toto" # renvoit False
"hello" > "Hello" # renvoit True
```

4) Copier une chaîne

Les chaînes de caractères font partie des structures de donnée **non mutables** c'est-à-dire non modifiables. Autrement dit, il est impossible de modifier un caractère de la chaîne via l'instruction `chaine[2] = 'e'` par exemple. Si l'on veut copier et/ou modifier une chaîne, il est nécessaire d'en créer une nouvelle et d'y concaténer les caractères à copier :

```
def copie(chaine):
    ch = ""
    for lettre in (chaine):
        ch = ch + lettre
    return ch
```

5) Méthodes de chaînes et langage objet

En Python, une chaîne de caractères est en réalité un **objet informatique** auquel on peut envoyer des messages lui indiquant de modifier son état, sa représentation en mémoire. Ses messages sont appelés des **méthodes d'objets** et s'utilisent pour manipuler les chaînes de caractères. Il existe notamment la méthode `replace` que nous allons recoder dans les exercices et bien d'autres encore dont une liste complète se trouve à l'adresse suivante :
https://docs.python.org/fr/3/library/stdtypes.html?highlight=split#str
On retiendra à ce stade que les objets sont les structures de données fondamentales des langages objets dont fait partie le langage Python et que **la programmation orientée objet** ou POO est un type de programmation permettant une bien meilleure structuration, lisibilité et maintenance du code par les équipes de développpements logiciels.

6) Signature de fonction

En informatique, une signature de fonction permet de définir les paramètres d'entrées d'une fonction ainsi que leur type mais également les valeurs qui seront retournées par la fonction.
Dans les exercices, nous utiliserons la syntaxe suivante pour mettre en œuvre cette signature et décrire ce que devront réaliser les fonctions que vous devrez coder :

```
nom_fonction(param1, param2, ...)
    |- ENTREES : description des paramètres et de leur type
    |- SORTIES : nature et description de la valeur retournée
```

Par exemple, la signature de la fonction `copie` définie plus haut pourrait se présenter de la manière suivante :

```
copie(chaine)
    |- ENTREES : chaine est de type str
    |- SORTIES : une copie exacte de la chaine passée en paramètre
```

Exercices à réaliser

Pour chaque exercice, ouvrez le fichier mentionné dans l'énoncé et lisez les consignes qui reprennent ou complètent les notions de ce document. Les conseils du TP précédent restent bien évidemment toujours valables, à savoir :

- respecter les niveaux d'indentation
- utiliser des noms de variables explicites
- utiliser des commentaires afin de documenter votre programme via le symbole #
- user et abuser des `print` pour débugger, les mettre en commentaire lorque votre programme fonctionne
- bien analyser les messages d'erreur de python qui vous renseignent sur la nature de l'erreur lors de l'exécution

Exercice 1

Ouvrir le fichier `TP-chaines-exo1.py`, lire les consignes et répondre aux questions suivantes :

1. Demander à l'utilisateur de saisir son prénom.
2. Écrire une fonction `compterLettres` qui compte le nombre de lettres du prénom et dont la signature est la suivante :

```
compter_lettres(mot)
    |- ENTREES : une variable mot de type str
    |- SORTIES : un entier indiquant le nombre de lettres
                 de la chaîne passée en paramètre
```

   Remarque : vous éviterez évidemment d'utiliser la fonction native `len(chaine)` .

3. Écrire une fonction `compter` qui compte le nombre de consonnes et de voyelles du prénom et dont la signature est la suivante :

```
compter(mot, flag)
    |- ENTREES : mot et flag sont des strings
                 flag (drapeau en français) indique la nature des
                 lettres à compter ("consonnes" ou "voyelles")
    |- SORTIES : un entier indiquant le nombre de consonnes ou de voyelles
                 de la chaîne passée en paramètre
```

4. Écrire une fonction `index` qui renvoit le le plus petit rang dans le prénom d'une certaine lettre et dont la signature est la suivante :

```
index(mot, lettre)
    |- ENTREES : mot et lettre sont des strings
    |- SORTIES : le rang de la lettre dans le mot
                 Par ex. :
                 index("toto", "o") renvoit 1
                 index("toto", "a") renvoit -1
```

Exercice 2

Ouvrir le fichier `TP-chaines-exo2.py`, lire les consignes et répondre aux questions suivantes :

1. Demander à l'utilisateur de saisir le mot de son choix.
2. Écrire une fonction `remplacer` qui remplace une lettre par une autre dans une chaîne et dont la signature est la suivante :


```
remplacer(mot, l1, l2)
    |-ENTREES : trois variables de type str
    |-SORTIES : une chaîne dans laquelle chaque lettre l1 du mot aura
                été remplacée par l2.
                Par ex., remplacer("exercice", "e", a") renvoit "axarcica"
```

3. Écrire une fonction `inverser` qui applique un effet miroir sur la chaîne passée en paramètre et dont la signature est la suivante :

```
inverser(mot)
    |-ENTREES : une variable mot de type str
    |-SORTIES : une chaîne dont les lettres ont été inversées.
                Par ex. : inverser(python) renvoit "nohtyp"
```

4. Écrire une fonction `est_palindrome` qui compte le nombre de consonnes et de voyelles du prénom et dont la signature est la suivante :

```
est_palindrome(mot)
    |-ENTREES : une variable mot de type str
    |-SORTIES : un booléen True ou False indiquant si mot est un palindrome
```

Remarque : un palindrome est un mot qui peut se lire dans les deux sens, par ex. : kayak.

Compétences à valider

Les chaînes de caractères					
Manipuler	Je sais parcourir une chaîne				
	Je sais faire un parcours inversé				
	Je sais comparer deux chaînes				
	Je sais copier les éléments d'une chaîne vers une autre et modifier éventuellement la copie				
Faciliter	Je connais la fonction native `len(chaine)`				
	J'ai compris ce qu'est une méthode de chaîne et leur utilité				
	J'ai compris la notion de signature de fonction				