# TP5 : Chaines de caractères

## Exercice 1 : Operations sur les chaines de caractères

Dans le langage python, les chaînes de caractères sont des objets «**str**» et peuvent être écrites de différentes manières:

- ✓ Avec des guillemets simples :`'autorisent les "guillemets"'`
- ✓ Avec des guillemets :`"autorisent les guillemets 'simples'"`
- ✓ Avec des guillemets triples :`'''Trois guillemets simples'''`, `"""Trois guillemets""" Les chaines multi-lignes sont autorisées dans ce cas`

Elles peuvent également être construites à partir du constructeur «**str()**» de la classe «**str**»
Exemple : `maChaine=str("mon texte")`

Le fait d'être des objets permettent aux chaînes de caractères de bénéficier de nombreuses fonctions propres appelées méthodes pour les manipuler. Vous obtiendrez la liste des méthodes disponibles des objets «**str**» simplement par la méthode «**help(str)**» dans une console Python.

Ecrire le script `nomprenom.py` permettant d'entrer le nom puis le prénom de deux personnes successivement. Le script affichera ensuite sur deux lignes « ***Prenom NOM*** » avec le nom en majuscules et la première lettre du prénom en majuscule et les deux lignes seront dans l'ordre alphabétique des noms et éventuellement des prénoms en cas de noms identiques.

Vous pouvez utiliser les méthodes de la classe «**str**» suivantes :

- **str.capitalize()** : renvoie une copie de la chaîne avec son premier caractère en majuscule et le reste en minuscule.
- **str.lower()** : renvoie une copie de la chaîne avec tous les caractères capitalisables convertis en minuscules.
- **str.upper()** : renvoie une copie de la chaîne dont tous les caractères capitalisables sont convertis en capitales.

Pour une comparaison concernant un ordre lexicographique, vous pouvez utiliser les opérateurs de comparaison **<, >, <= et >=**. La comparaison elle-même se fait caractère par caractère. L'ordre dépend de l'ordre des caractères dans l'alphabet, qui dépend aussi de la table de codage des caractères utilisée par votre machine lors de l'exécution du code Python. Ainsi lorsque nous comparons des chaînes de caractères, nous comparons leurs valeurs Unicode. À noter que les comparaisons de chaînes en Python sont sensibles à la casse.

Notez que `str.upper().isupper()` pourrait être False si la chaine contient des caractères non capitalisables

## Exercice 2 : Notes

Ecrivez le programme `Notes.py` qui demande à l'utilisateur de rentrer 5 valeurs de type réel représentant les notes finales avec 5 valeurs de type entier représentant les coefficients de chacune des notes. Votre programme doit permettre la récupération de ces valeurs comme dans l'exemple suivant :

```
Veuillez entrer la note du module 1 et le coefficient
correspondant : 10.5 2
```

Donc la récupération des deux valeurs se fait par le biais d'une seule question ! Pour réaliser cela, vous pouvez utiliser la méthode split() sur la chaine de caractère que vous allez récupérer suite à votre question. Voici un exemple d'utilisation de la méthode split

```
S1 = "15 1.5" # chaine de caractère
S2 = S1.split(" ") # on segmente la chaine suivant le
séparateur espace
S3 = S2[0] # S3 vaut 15
S4 = S2[1]  # S4 vaut 1.5
```

Votre programme doit ensuite calculer la moyenne générale de l'étudiant puis d'évaluer si l'étudiant est admis ou non. L'étudiant est admis s'il a une moyenne générale supérieure à 10 et si aucune de ses notes n'est inférieure à 8.

## Exercice 3 : Palindromes

Un palindrome est un mot que l'on peut lire dans les deux sens. La distinction entre majuscules/minuscules n'a aucune importance pour la lecture d'un palindrome. Si on ne tient pas compte des caractères non alphabétiques (i.e. ' ', ',', '-' et '\''), une phrase complète peut aussi être considérée comme un palindrome.
Exemples de palindromes :

```
Otto
Esope reste ici et se repose
Engage le jeu que je le gagne.
```

Exemples de non-palindromes :

```
Cours de Python
```

Ecrivez un programme `Palindrome.py` qui :
1. lit une chaine de caractères du clavier ;
2. retire les caractères non alphabétiques ;
3. teste si la chaîne ainsi épurée est un palindrome.

Exemple d'exécution :

```
Entrez un mot ou une phrase : Otto
C'est un palindrome !
```

Pour ce programme, il convient d'utiliser plusieurs méthodes prédéfinies de la classe `String`, comme par exemple `lower()`, `isalpha()` et `len()`.

`chaine.lower()` permet de convertir tous les caractères de l'objet `chaine` en minuscule.

**Indication** : l'appel `Character.isalpha(c)`, où 'c' c'est un caractère, permet de tester si 'c' est alphabétique

- N.B Les trois exercices suivants sont issus du TD et ils doivent vous rappeler quelque chose ! C'est à vous de transformer vos solutions algorithmiques en programme Python !

## Exercice 4 : La monnaie (TD1)

Ecrire un programme permettant de lire une valeur entière représentant une somme d'argent et de décomposer cette somme en billets de 100, 50, 10 et en pièces de 2 et 1, en utilisant à chaque fois la plus grande valeur pour chaque type de billet ou pièce. Afficher le résultat de cette décomposition à l'écran.

Exemple : soit la somme de 627 euros. Le message à afficher est : « 627 euros, c'est donc 6 billets de 100, 0 de 50, 2 de 10, 3 pièces de 2 et 1 pièce 1. »

## Exercice 5 : Fiche de paye (TD2)

Ecrire un programme permettant de calculer le salaire d'un ouvrier en fournissant le nombre d'heures travaillées et le salaire horaire. La règle de calcul est la suivante :

- Le salaire horaire est appliqué pour les 160 premières heures.
- Ensuite, il y a une première majoration du salaire horaire de 25% pour toutes les heures entre 161 et 200,
- Puis une dernière majoration de 25% supplémentaire (soit 50%) pour les heures en plus de 200.

## Exercice 6 : Chaînes de caractères (TD4)

Soit T[100] un tableau représentant une chaîne de 100 caractères au maximum (délimiter comme en C par un caractère de fin de chaîne dont le code ASCII est 0).

1. Donner un algorithme qui calcule la taille de la chaîne de caractères T.
2. Calculer le pourcentage de voyelles contenues dans la chaîne de caractères (sans compter le caractère de fin de chaîne)
3. Tester si la chaîne ''wagon'' est une sous-chaîne de T, et donner le début de la première occurrence si elle existe.
4. Calculer le nombre d'occurrences de la chaîne ''wagon'' dans T.

- Pour cet exercice, vous pouvez adapter l'énoncé en considérant votre tableau de chaines de caractères comme une liste de caractères. C'est d'ailleurs le cas avec Python, une chaine de caractère c'est une liste ! Vous faites aussi en sort que c'est l'utilisateur qui donne la chaine à traiter.
- En Python il n'y a pas de caractère de fin de chaine. En effet, une chaine de caractère est vue comme un objet contenant un attribut qui s'incrémente pour chaque caractère

ajouté à la liste. Donc, la fin de chaine c'est tout simplement le dernier élément de la liste.

- Ce n'est pas la peine d'utiliser les méthodes offertes par les listes avec Python (len et find). **C'est à vous de faire le travail !**
- Pour la question 3 et 4, tester votre code avec les phrases suivantes :

- "Le wagon bleu est rapide."
- "Un petit wagon roule vite."
- "Le ciel est bleu, comme un wagon."
- "Wagon après wagon, le train avance."
- "Il y a cinq wagons dans le train."
- "La locomotive tire les wagons."
- "Wagon, wagon, wagon, ciel bleu."
- "Le train avance avec ses wagons."
- "Un wagon rouge file à toute vitesse."
- "Le wagon jaune transporte des marchandises."

## Exercice 7 : Dernière modification

Le module « **os.path** » implémente les fonctions nécessaires à la manipulation des fichiers et leur chemin d'accès.
Pour importer un module il suffit d'utiliser l'instruction « **import** »

```
>>> import nom_du_module
```

Il est possible de n'importer que certaine fonction d'un module en utilisant la clause « **from** » dans l'instruction « **import** »

```
>>> from nom_du_module import nom_fonction1, nom_fonction2
```

Ecrire le script *Verification.py* permettant d'entrer le nom de deux fichiers. Si ces fichiers existent, le script donnera leurs tailles en octets. Le script indiquera ensuite quel est le fichier le plus récent (le fichier modifié le plus récemment) en affichant la date de modification.

Pour cela, vous pouvez utiliser les fonctions du module « **os.path** » suivantes :

- ✓ **os.path.isfile(path)** : renvoie « **True** » si « **path** » est un fichier
- ✓ **os.path.getmtime(path)** : renvoie l'heure de la dernière modification du fichier donnée en paramètre. La valeur retournée est un nombre sous forme de timestamp représentant le nombre de secondes écoulées depuis le 1er janvier 1970 (l'Epoch Unix). Pour formater d'une manière plus lisible cette valeur, vous pouvez utiliser la **`datetime.datetime.fromtimestamp()`** en donnant en argument la date à formater. Vous devez pour cela importer le module **'datetime'**

Pour tester votre programme, créer deux fichiers *f1.txt et f2.txt* dans le répertoire courant (contenant votre fichier python), puis lancer votre code.
Pour avoir un aperçu des informations vous pouvez lancer la commande **ls** au niveau de l'onglet **terminal** de PyChram.