TP N°6: Les chaînes de caractères

Une *chaîne de caractères* est une suite ou *séquence ordonnée* de caractères Unicode (lettres et symboles) entre simple ou double côtes *non modifiable*. Elle appartient au type `str`. Elles supportent le test d'appartenance, la concaténation, la répétition, l'accès par indice, par tranche, la taille,... mais pas de changement de ses élémentch. On peut convertir une chaîne en une liste et la modifier.

Création

```
ch1 = "Bonjour" # Créer une chaîne de caractères
ch2 = "".join(['v','é','l','o']) # Créer une chaîne de caractères 'vélo' à partir d'une liste
str(['v','é','l','o']) # retourne "['v', 'é', 'l', 'o']"
L = list ("vélo") # Convertir une chaine en une liste ['v','é','l','o']
ch3 = "Aujourd'hui" # chaîne contenant une apostrophe
ch4 = 'Aujourd\'hui'
ch5 = """Première ligne
Deuxième ligne"""  # utiliser """ ou  ''' : pour encadrer une chaîne définie sur plusieurs lignes
ch6 = 'Première ligne\nDeuxième ligne' # utiliser \n pour faire un retour à la ligne
ch7 = "a" * 3 # utiliser l'opérateur * pour répéter une chaîne
ch8 = "IP" + "EIN" # utiliser l'opérateur + pour concaténer une chaîne
```

Remarques :

- `ord('c')` : retourne le code ASCII de 'c'. Exemple : `ord('m')` → `109`
- `chr(code ASCII)` : retourne le caractère correspondant. Exemple : `chr(109)` → `'m'`
- On peut afficher une chaine dans un format prédéfini à l'aide de la fonction `format()`.
  Exemple : `"Bonjour M. {}".format('Ali')` → `"Bonjour M. Ali"`

Exercice 1:

1. Transformer la chaîne de caractère "3.14" en un flottant.
2. En utilisant l'écriture formatée, affichez la phrase «`Bonjour M. Ali`» en utilisant les variables `temps` et `prenom` dont les valeurs sont respectivement "jour", "Ali".

Fonctions et méthodes associées aux chaînes

```
ch = "Bonjour"  # chaîne initiale
ch.index('n')   # renvoie l'index de la première occurrence d'une sous-chaîne;
                # renvoie erreur si non trouvée.
ch.find("jour") # renvoie l'index de la première occurrence d'une sous-chaîne;
                # renvoie -1 si non trouvé.
ch.count('o')   # compte le nombre de sous-chaîne "o" dans la chaîne ch
ch.split('o')   # diviser la chaîne ch en une liste basée sur un délimiteur "o".
ch.replace("jour","soir") # Remplace une sous-chaîne par une autre
ch.upper()      # Changer la casse (mettre la chaîne en Majuscule)  : BONJOUR
ch.lower()      # Changer la casse (mettre la chaîne en miniscule)  : bonjour
ch.capitalize() # Mettre la première lettre en Majuscule            : Bonjour
ch.isalpha()    # Vérifier si la chaîne ne contient uniquement des caractères alphabétiques
ch.isdigit()    # Vérifier si la chaîne ne contient uniquement des caractères numériques
ch.isalnum()    # Vérifier si la chaîne ne contient uniquement des caractères alphanumériques
ch.isspace()    # Vérifier si la chaîne ne contient uniquement des caractères espaces
len(s)          # renvoie le nombre de caractères d'une chaîne
"b" in ch       # Tester avec in et not in si un caractère appartient ou non à une chaîne.
ch == "bonsoir" # Comparer le contenu de deux chaînes
```

Exercice 2

1. Écrire un script qui détermine si une chaîne contient ou non le caractère « e », puis s'il existe, compter son nombre d'occurrences et l'enlever de la chaîne de caractères.

2. Écrire un script qui détermine si une chaîne est palindrome ou non. On rappelle qu'une chaîne de caractères est dite palindrome, si elle se lit de la même manière dans les deux sens. Exemple: non, "12321", "elle".
3. Écrire un script qui détermine si une chaîne de caractères `ch` contient seulement des caractères chiffres.
4. Soit la chaîne de caractères `a='1.0 3.14 7 8.4 0.0'`. Écrire un script qui, à partir de `a`, et en utilisant `float()` et `split()`, construit la liste de réels `[1.0, 3.14, 7.0, 8.4, 0.0]`.

Itération sur les chaînes

```
for x in  S: # On peut itérer directement sur une chaîne  : x représente un élément de S
    print(x)

for i in range(len(S)): # i représente un indice variant de 0 à len(L) - 1
    print(S[i]) # TypeError: 'set' object does not support indexing
```

Exercice 3

1. Écrire un script qui génère une chaîne composée des chiffres de 0 à 9.
2. Écrire un script qui génère une chaîne composée des caractères alphabétiques de *a* à *z*.
3. Écrire un script qui recopie une chaîne (dans un nouvelle variable), en insérant des astérisques entre les caractères. Ainsi par exemple, « Docteur » devra devenir « d*o*c*t*e*u*r »
4. Écrire un script qui recopie une chaîne (dans une nouvelle variable) en l'inversant. Ainsi par exemple, « python » deviendra « nohtyp ».
5. En partant de l'exercice précédent, Écrire un script qui détermine si une chaîne de caractères est un palindrome (c'est-à-dire une chaîne qui peut se lire indifféremment dans les deux sens), comme par exemple « radar » ou « s.o.s »
6. Écrire un script Python qui génère une chaîne de longueur *n* choisie, constituée de caractères tirés au hasard parmi les lettres majuscules. Essayer de deviner cette chaîne.
   Vous pouvez utiliser le module Python `random`.

   ```
   import random
   random.randint(1,10)     # renvoie un nombre entier aléatoire entre 1 et 10
   ```

Exercice 4

1. Générer la liste de tous les *bigrammes* (chaine de deux lettres) possibles avec l'alphabet français. Votre script donnera ce résultat: `bigrammes = ["AA", "AB", "AC", "AD", ..., "BA", "BB", "BC", ..., "ZY", "ZZ"]`
2. Soit la séquence *nucléotidique* ch suivante: ACCTAGCCATGTAGAATCGCCTAGGCTTTAGCTAGCTCTAGC. Déterminer la lettre de l'alphabet la plus fréquente dans la chaîne de caractères ch.
3. Faites un programme qui répertorie dans une liste de tuples tous les mots de 2 lettres qui existent dans une séquence nucléotidique ainsi que leur nombre d'occurrences puis qui les affiche à l'écran. Votre script donnera ce résultat: `[("AC",12),("AB",41),("GA",10), ...]`.

Exercice 5

La *distance de Hamming* entre deux mots est une notion utilisée dans de nombreux domaines (télécommunications, traitement du signal, . . .). Elle est définie, pour deux mots de même longueur, comme le nombre de positions où les deux mots ont un caractère différent. Écrire un programme Python qui calcule la distance de Hamming entre deux mots lorsqu'ils ont la même longueur, et qui renvoie -1 sinon. Par exemple:
- Pour `ch1="aaba"` et `ch2="aaha"` la distance de Hamming = 1,
- Pour `ch1="stylo"` et `ch2="bouteille"` la distance de Hamming = -1.