Bilan du travail à la maison

Exercice 1
Avez-vous des question concernant les exercices traités sur codingame ?

Les deux prochains exercices sont a traiter sur une feuille à part, et à rendre.

Exercice 2

1. Décrire ce que va afficher le programme suivant :

```
texte = "Homme libre, toujours tu chériras la mer !"
compteur=0
for lettre in texte :
    if lettre == "e" :
        compteur += 1
print(compteur)
```

2. Que va afficher le programme suivant ?

```
chaine = "Un chasseur sachant chasser doit savoir chasser sans son chien."
print(chaine[:5])
```

3. Donner le numéro des lignes affichant `True`

```
chaine = "Un chasseur sachant chasser doit savoir chasser sans son chien."
print("ch" in chaine)
print("chant" in chaine)
print("Un chien" in chaine)
print("r s" in chaine)
print("un" in chaine)
```

Exercice 3
On considère que `texte` est une variable déjà créé qui contient un texte quelconque.
Donner des instructions python permettant d'afficher (`print`) successivement tous les indices où la lettre "e" apparaît dans `texte`

Exercice 4
Pour le prochain TD, traiter chez vous les feuilles n°11 et 12 (de la page https://www.codingame.com/playgrounds/53303/).

I Échauffement

La feuille de script `TD4_script.py` est fournie avec l'énoncé de ce TD. Elle contient la définition d'une variable `texte` que nous utiliserons lors des tests des différentes fonctions

Exercice 5
Assurez vous de comprendre le programme suivant, donné également dans la feuille de script :

```
def nombre_occ(s, c):
    """
    Retourne le nombre de fois où le caractère c apparait dans la chaîne s
    """
    assert len(c) == 1  # ceci provoque une erreur si c n'est pas un caractère
    compt = 0
    for car in s:
        if car == c:
            compt = compt + 1
    return compt
```

1. Quel sera le résultat de `nombre_occ('un texte court', ' ')` ?
2. En vous référant au TD précédent, compléter la signature de cette fonction par une annotation de types.
3. En notant n la longueur de `s`, combien de comparaison entre caractères devons nous effectuer pour calculer le nombre d'occurrences d'une lettre donnée dans `s` ? de chacune des 26 lettres de l'alphabet dans `s` ?
   Dans la console, évaluer la longueur de la variable `texte` (après avoir exécuté la feuille de script) et donner une application numérique du nombre de comparaisons dans ce cas.
4. Tester dans la console `nombre_occ('un texte court', 'un')`. Le résultat était-il prévisible ?

Exercice 6
On cherche maintenant à savoir combien de fois un mot donné apparaît dans un texte. Pour cela nous allons utiliser les tranches ou “slice”.

1. Tester dans la console (rappel : pour répéter une commande dans la console, on peut utiliser la flèche du haut).

```
texte[0:41]
texte[:41]
texte[10:41]
```

2. On considère deux variables : `mot` et `s` qui contiennent deux chaînes de caractères. et on note i un indice entier. Donner l'instruction python permettant de savoir si la tranche de `s` commençant à l'indice i et de bonne longueur est égale à `mot`.
3. Compléter la fonction `occurrences_mot` en implémentant l'algorithme suivant :
   On note n la longueur de `s` et p celle de `mot`. Pour chaque indice `i` à partir de 0 et jusqu'à un nombre à exprimer en fonction de n et p (ce nombre sera à inclure dans le `range`), on compare la tranche de `s` qui commence à l'indice `i` avec `mot`. En cas d'égalité, on incrémente un compteur.

Tester votre fonction sur un exemple simple, puis sur un exemple où la dernière occurrence de mot fini la chaîne `s`.

II Dictionnaires et analyse de texte

2.1 Structure de données

Un **dictionnaire** est un autre type de données utilisable en python. Il permet de stocker des **valeurs** associées à des **clés**.
Plus précisément, un dictionnaire est un ensemble de (clé, valeur) où chaque valeur est associée à exactement une clé.

2.2 Utilisation en python

Pour créer un dictionnaire, on utilise des accolades de manière similaire aux crochets utilisés pour les listes.

```
d = {}  # un dictionnaire vide
```

```
d = {'a': 12}  # un dictionnaire contenant une paire de (clé, valeur)
# la clé est la chaîne ''a'' et la valeur le nombre 12
d['a']  # c'est la valeur associée à la clé ''a'', ici 12
d[0] = [2 ,4, 6]  # on crée un nouvelle paire (clé, valeur). La clé est un nombre
# et la valeur une liste.
```

- Les clés utilisables pour nous sont : des chaînes, des nombres. En toute généralité, tout objet non modifiable peut être utilisé comme clé.
- Les valeurs sont des objets python quelconques (nous connaissons : nombres entiers ou flottants, booléen, chaîne, liste, dictionnaire)

Pour tester si une clé est présente dans un dictionnaire (et avec le dictionnaire `d` du cadre précédent)

```
'a' in d  # c'est le booléen True car la clé est présente dans d
2 in d # c'est le booléen False
```

2.3 Application aux nombres d'occurrences

Exercice 7
On souhaite, pour une chaîne donnée, compter le nombre d'occurrences de tous ses caractères en une seule boucle. Pour cela nous allons construire un dictionnaire dont les clés sont des caractères et les valeurs des entiers (qui seront les nombres d'occurrences à la fin de l'exécution du code). Chaque valeur sera en fait un compteur.
Nous allons compléter la fonction `dico_occurences(s : str) -> dict`. Son utilisation donne par exemple

```
dico_occurrences("un texte")
# valeur de retour : {'u': 1, 'n': 1, ' ': 1, 't': 2, 'e': 2, 'x': 1}
```

L'algorithme est le suivant :

pour chaque caractère `c` de `s`, on vérifie d'abord si la clé `c` existe dans le dictionnaire en construction.
Si oui, on ajoute 1 au compteur associé, si non on cré une nouvelle paire (clé, valeur).

Exercice 8
Combien de fois l'instruction `if` est-elle exécutée lors de l'appel à `dico_occurrences(texte)` (où `texte` est définie par la feuille de script) ? Comparer à la méthode de la partie précédente qui permettait d'obtenir les nombres d'occurrences de chaque caractère.

Exercice 9
Quelle est le caractère le plus présent dans `texte` ? Et le plus présent hors espace ?

2.4 Mot majoritaire

Si `s` est une variable contenant une chaîne de caractère, l'instruction

```
s.split()
```

retourne une liste contenant les mots de `s`. Plus précisément, les éléments de la liste sont les sous-chaînes de `s` qui sont séparées par un ou plusieurs caractères d'espace (l'espace, $\backslash n$ qui représente le retour à la ligne et $\backslash t$ qui représente la tabulation)

```
"un texte composé de plusieurs      mots".split()
# valeur de retour : ['un', 'texte', 'composé', 'de', 'plusieurs', 'mots']
```

Exercice 10
Adapter la méthode de l'exercice précédent pour compléter la fonction `mot_majoritaire(s: str) -> str`.
Pour itérer sur un dictionnaire, on peut utiliser la structure suivante

```
for cle in d:
    # cle prend comme valeurs successives toutes les clés présentes dans le dictionnaire d
    d[cle]  # retourne la valeur associée à cle, c'est à dire va être successivement
    # toutes les valeurs stockées dans d
```

Quel est le mot le plus présent dans `texte` ? En créant une autre fonction, trouver le plus de 4 lettres ou plus le plus présent dans `texte`.

2.5 Pour aller plus loin

Exercice 11
Trouver les couples de mots de `texte` qui apparaissent exactement le même nombre de fois et au moins 5 fois chacun. On pourra créer une liste dont les éléments sont de la forme `[mot1, mot2]`.
Comment éviter d'obtenir deux fois les mêmes couples ? (Indication : utiliser l'ordre alphabétique)

Exercice 12
On s'intéresse maintenant aux nombres d'occurrences des mots dans `texte`. Combien y a-t-il de mots distincts apparaissant 4 fois dans `texte` ?
Construire la liste des éléments de la forme [k , p] où k est un nombre d'occurrences et p le nombre de mots de `texte` apparaissant exactement k fois. On pourra tester ensuite la commande

```
L.sort()
```

si la liste construite est appelée `L`. Ceci trie la liste `L`.