TP1

Les bases de Python

1 – Présentation du TP

Objet du TP

Ce TP contient 10 exercices de difficulté croissante et s'étale sur 2 séances. L'objectif est de faire l'intégralité de ces exercices à l'issue des 2 séances.

Compétences travaillées dans ce TP

- Syntaxe de base de Python : fonctions, structures de contrôle
- Structures de données : séquences, dictionnaires

2 – Création du projet

Lancez PyCharm et cliquez sur « New Project ».

Nommez votre projet « tp1 ».

Avant de cliquer sur « Create », faites les vérifications suivantes :

- Au besoin, déroulez la partie « Python Interpreter »
- Vérifiez que la case cochée n'est **pas** « New environment using… ». Si c'est le cas, faites les opérations indiquées dans le TP 0.
- Vérifiez que la case « Create a main.py welcome script » est **décochée**.

Faites ces vérifications à chaque TP. Le plus important est de ne pas créer d'environnement virtuel.

Dans la partie gauche, faites un clic droit sur « tp1 », sélectionnez « New » puis « Python File ».

Entrez le nom « tp1 » (en minuscules, sans accents ni espaces). Ceci va créer le fichier « tp1.py » (l'extension « .py » est ajoutée automatiquement).

3 – Exercices

Pour chaque exercice, des exemples d'utilisation vous sont données pour tester vos fonctions ***a minima***. Mais il est fortement conseillé d'enrichir ces exemples avec vos propres tests.

Exercice 1 : Convertion de température

Sachant que la formule de conversion des degrés Celsius aux degrés Fahrenheit est :

$$°F = (\ °C \times 9/5\ ) + 32$$

Implémentez **et testez** une fonction :

```
def celsius_fahrenheit(celcius) :
```

qui convertit la température `celcius` (exprimée en degrés Celsius) en degrés Fahrenheit.

Attention : la fonction celsius_to_fahrenheit() ne doit faire aucun affichage, mais renvoyer le résultat.

Exemples d'utilisation :

```
temperature_celcius = 22
temperature_fahrenheit = celsius_fahrenheit(temperature_celcius)
print(temperature_fahrenheit)  # affiche : 71.6
```

Ou simplement :

```
print( celsius_fahrenheit(22) )   # affiche : 71.6
```

Exercice 2 : Mention

Implémentez et testez une fonction :

```
def mention(note) :
```

qui renvoie, sous la forme d'une chaîne de caractères, la mention correspondante à la `note` donnée en argument, selon les règles suivantes :

- *insuffisant* pour une note inférieure à 10,
- *passable* pour une note comprise entre 10 et 12,
- *assez bien* pour une note comprise entre 12 et 14,
- *bien* pour une note entre 14 et 16,
- *très bien* pour une note entre 16 et 20.

Astuce :

En Python, on peut directement tester des encadrements avec des conditions de la forme :

```
if a < x < b :
```

Ce qui est équivalent à :

```
if a < x and x < b :
```

Exemple d'utilisation :

```
print( mention(13) )   # affiche : assez bien
```

Exercice 3 : Nombres pairs

Implémentez et testez une fonction :

```
def nombres_pairs(n) :
```

qui renvoie, sous la forme d'une liste, tous les nombres pairs inférieurs ou égaux à n.

Exemple d'utilisation :

```
print( nombres_pairs(10) )   # affiche : [0, 2, 4, 6, 8, 10]
```

Amélioration : si vous avez utilisé un `if`, essayez d'implémenter une version sans `if`

Exercice 4 : Remplacement d'un caractère

Implémentez et testez une fonction :

```
def remplacer(texte, caractere, nouveau_caractere) :
```

qui renvoie une chaîne de caractères obtenue en remplaçant toutes les occurrences de `caractere` par `nouveau_caractere` dans le `texte` fourni.

Contrainte : ne pas utiliser la méthode replace() du type `str`

Exemple d'utilisation :

```
print( remplacer("toto", "o", "i") )   # affiche : titi
```

Amélioration : utiliser un ternaire

Exercice 5 : Palindromes

Un palindrome est un mot dont l'ordre des lettres reste le même qu'on le lise de gauche à droite ou de droite à gauche. Par exemple, le mot *kayak* est un palindrome.

Implémentez et testez une fonction :

```
def est_palindrome(mot) :
```

qui retourne `True` si le contenu de la chaîne `mot` est un palindrome, `False` Sinon.

Exemples d'utilisation :

```
print( est_palindrome("ressasser") )   # affiche : True
print( est_palindrome("palindrome") )  # affiche : False
```

Amélioration : Si ce n'est pas déjà fait, essayez d'écrire cette fonction en 1 ligne de code en utilisant une tranche.

Exercice 6 : Triplets pythagoriciens

Un triplet (a,b,c) est ***pythagoricien*** si ces composantes vérifient les relations suivantes :

- la relation d'ordre a < b < c,
- la relation de Pythagore : $a^2 + b^2 = c^2$.

Exemple : (3,4,5) est *pythagoricien* car 3 < 4 < 5 et $3^2 + 4^2 = 5^2$

Implémentez et testez une fonction :

```
def triplets_pythagoriciens(n) :
```

qui retourne les triplets pythagoriciens d'entiers de 1 à n, sous la forme d'une liste de tuples.

Exemple d'utilisation :

```
# affiche : [(3, 4, 5), (5, 12, 13), (6, 8, 10), (8, 15, 17), (9, 12,
15), (12, 16, 20)]
print( triplets_pythagoriciens(20) )
```

Exercice 7 : Diviseurs d'un nombre

Implémentez et testez une fonction :

```
def diviseurs(n) :
```

qui retourne les diviseurs du nombre n, sous la forme d'une liste.

Exemple d'utilisation :

```
# affiche : [1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30, 60]
print( diviseurs(60) )
```

Bonus : si ce n'est déjà fait, optimiser le temps d'exécution du code pour que la complexité soit en racine de n.

Exercice 8 : Nombre parfait

Un nombre est dit **parfait** s'il est égal à la somme de ses diviseurs *propres* (diviseurs autres que le nombre lui-même). Par exemple, 28 est un nombre parfait car 28 = 1+2+4+7+14.

Implémentez et testez une fonction :

```
def est_parfait(n) :
```

qui retourne `True` si le nombre n est pair, `False` Sinon.

Astuce : `sum(l)` fait la somme des éléments de la liste `l`

Exemples d'utilisation :

```
print( est_parfait(6) )      # affiche : True
print( est_parfait(42) )     # affiche : False
print( est_parfait(8128) )   # affiche : True
```

Exercice 9 : Comptage de lettres

Implémentez et testez une fonction :

```
def occurrences_lettres(mot) :
```

qui renvoie un dictionnaire donnant le nombre d'occurrences des lettres composant le `mot` fourni.

Exemple d'utilisation :

```
# affiche : {'A': 1, 'N': 5, 'T': 5, 'I': 3, 'C': 1, 'O': 2, 'S': 1, 'U': 1,
'E': 3, 'L': 2, 'M': 1}
print( occurrences_lettres("ANTICONSTITUTIONNELLEMENT") )
```

Exercice 10 : Score d'un mot au Scrabble

Dans cet exercice, nous allons considérer une version simplifiée du Scrabble.

Le Scrabble est un jeu dont l'objectif est de marquer des points en créant des mots à partir de lettres tirées au hasard.

Chaque lettre possède une valeur. Pour obtenir le score d'un mot, on ajoute la valeur de chacune des lettres le composant. Par exemple, le mot « POMME » rapporte 3 + 1 + 2 + 2 + 1 = 9 points.

Dans la version francophone du jeu de Scrabble, les lettres ont les valeurs suivantes :

- lettres valant 1 point : E, A, I, N, O, R, S, T, U, L
- lettres valant 2 points : D, M, G
- lettres valant 3 points : B, C, P
- lettres valant 4 points : F, H, V
- lettres valant 8 points : J, Q
- lettres valant 10 points : K, W, X, Y, Z

Pour vous faire gagner du temps, on vous fournit ces informations sous la forme du dictionnaire Python ci-dessous, qui associe une liste de lettres à un nombre de points :

```
points_lettres = {
    1 : ["E", "A", "I", "N", "O", "R", "S", "T", "U", "L"],
    2 : ["D", "M", "G"],
    3 : ["B", "C", "P"],
    4 : ["F", "H", "V"],
    8 : ["J", "Q"],
    10 : ["K", "W", "X", "Y", "Z"]
}
```

Implémentez une fonction :

```
def calculer_score(mot, points_lettres) :
```

qui calcule et renvoie le score d'un `mot` au Scrabble en utilisant le dictionnaire `points_lettres` fourni.

Exemples d'utilisation :

```
print( calculer_score("POMME", points_lettres) )    # affiche : 9
print( calculer_score("JUKEBOX", points_lettres) )  # affiche : 34
```