TP – CHAPITRE 4 - Dictionnaires

Sommaire

TP1 - Premier exemple de dictionnaire

- Sur votre compte, dans le dossier **NSI** , créer un sous-dossier **TP-Chapitre4** dans lequel on rangera les TP de ce chapitre.
- Récupérer les fichiers **fichier1.csv**, **dico1.py** et **TP1 .py** donnés en ressource et les enregistrer dans le dossier **TP-Chapitre4** sur votre compte.
- Dans le fichier **TP1 .py** , le dictionnaire qui est chargé est un dictionnaire pour lequel les clés sont les prénoms et les valeurs sont les âges.
- Compléter le premier **pass** afin de rajouter une nouvelle entrée dans le dictionnaire **dico1** avec pour clé : **votre propre prénom** et pour valeur : **votre âge**.
- Compléter le deuxième **pass** afin d'obtenir dans la console l'affichage ci-dessous, la dernière ligne doit comporter votre prénom et votre âge.

```
Bob a 17 ans
...
...
Enzo a 19 ans
____ a __ ans
```

Lever la main pour valider ce TP.

TP2 - Filtre dans un dictionnaire

- Dupliquer le fichier **TP1 .py** en **TP2 .py** .
- Modifier le code pour obtenir dans la console l'affichage ci-dessous de tous les prénoms des personnes du dictionnaire ayant 17 ans.

```
Liste des personnes du dictionnaire ayant 17 ans :
Bob
Téo
Louis
Luc
Célia
```

Lever la main pour valider ce TP.

TP3 - Scrabble

- Récupérer les fichiers **scrabble.csv** et **TP3 .py** en ressource et les enregistrer dans le dossier **TP-Chapitre4** sur votre compte.
- Compléter le code de la fonction **points(mot)** pour que cette fonction renvoie le nombre de points du mot en majuscule passé en paramètre au scrabble, c'est à dire la somme des points de chaque caractères du mot. Dans le dictionnaire **dico_scrabble** on charge à partir du fichier **scrabble.csv** les valeurs au scrabble de toutes les lettres majuscules de l'alphabet. Par exemple : **dico_scrabble['A'] = 1** .
- Exemple de retours attendus dans la console :

```
>>> point('BON')
5
>>> point('ZOO')
12
>>> point('ZIZANIE')
25
```

Lever la main pour valider ce TP.

TP4 - Occurences dans une séquence

- Récupérer le fichier **TP4 .py** en ressource et l'enregistrer dans le dossier **TP-Chapitre4** sur votre compte.
- Compléter le code de la fonction **occurences(sequence)** pour que cette fonction renvoie un dictionnaire dont les clés sont les éléments de la séquence (de type **str** ou **list** ou **tuple**) et les valeurs le nombre de fois que l'élément apparaît dans la séquence.
- Exemple de retours attendus dans la console :

```
>>> print(occurences('radar'))
 {'r': 2, 'a': 2, 'd': 1}
>>> print(occurences([1, 1, 1, 2, 2, 5]))
 {1: 3, 2: 2, 5: 1}
>>> print(occurences(('a', 'a', 5, True, True, True)))
 {'a': 2, 5: 1, True: 3}
```

Lever la main pour valider ce TP.

TP5 - Ecrire des mots en morse

Le code morse est composé de deux symboles, **le point (.)** et l**e tiret (-)**. Le séparateur entre les codes de deux lettres est un espace et le séparateur entre deux mots est souvent le symbole / .

Exemple : `BONJOUR MONSIEUR` va s'écrire en morse :

```
-... --- -. .--- --- ..- .-./-- --- -. ... .. . ..- .-.
```

Ce code peut-être utiliser sous de nombreuses formes. Les signes graphiques (point ou tiret) peuvent être traduit en impulsions électriques, sonores ou lumineuses.

Conventions :

- Le **point** est traduit par une **impulsion brève**.
- Le **tiret** est traduit par une **impulsion longue** dont la durée est égale à trois impulsions brèves.
- L'espace entre deux signaux (point ou tiret) se traduit par un **silence court** dont la durée est égale à une impulsion brève.
- L'**espace** qui sépare le code morse de deux lettres se traduit par un **silence moyen** dont la durée est égale à trois impulsions brèves.
- Le **slash** qui sépare le code morse de deux mots se traduit par un **silence long** dont la durée est égale à sept impulsions brèves.

Code morse pour les lettres majuscules :

A .-	J .---	S ...
B -...	K -.-	T -
C -.-.	L .-..	U ..-
D -..	M --	V ...-
E .	N -.	W .--
F ..-.	O ---	X -..-
G --.	P .--.	Y -.--
H	Q --.-	Z --..
I ..	R .-.	

-

-

- Récupérer le fichier **TP5 .py** en ressource et l'enregistrer dans le dossier **TP-Chapitre4** sur votre compte.
- Compléter le code de la fonction **morse(mot)** pour que cette fonction renvoie une chaîne de caractères correspondant à l'écriture en morse du mot passé en paramètre qui est aussi de type **str**. Utiliser le dictionnaire **dico_morse** dont les clés sont les lettres majuscules de

l'alphabet et les valeurs sont l'écriture en morse de chacunes de ces lettres. Le **séparateur** entre deux codes morses est l'**espace.**

- Exemple de retours attendus dans la console :

```
>>> morse('BONJOUR')
'-... --- -. .--- --- ..- .-.'
>>> %Run TP5-morse.py
>>> morse('LE ZOO')
'.-.. . / --.. --- ---'
```

Lever la main pour valider ce TP.

TP6 - Inverser un dictionnaire

- Dupliquer le **TP5 .py** en **TP6 .py** .
- Rédiger une nouvelle fonction **inverser_dico(dico)** qui prend en paramètre un dictionnaire nommé **dico** et qui renvoie le dictionnaire dont les clés sont les valeurs de **dico** et les valeurs sont les clés de **dico** .
- Affichage dans la console :

```
>>> dico1 = inverser_dico(dico_morse)
>>> print(dico1)
{'.-': 'A', '-...': 'B', '-.-.': 'C', '-..': 'D', '.': 'E
', '..-.': 'F', '--.': 'G', '....': 'H', '..': 'I', '.---
': 'J', '-.-': 'K', '.-..': 'L', '--': 'M', '-.': 'N', '-
--': 'O', '.--.': 'P', '--.-': 'Q', '.-.': 'R', '...': 'S
', '-': 'T', '..-': 'U', '...-': 'V', '.--': 'W', '-..-':
'X', '-.--': 'Y', '--..': 'Z', '/': ' '}
```

Lever la main pour valider ce TP.

TP7 - Déchiffrer des mots en morse

- Dupliquer le **TP6 .py** en **TP7 .py .**
- Rédiger une nouvelle fonction **dechiffrer(mot_en_morse)** qui prend en paramètre une chaîne de caractères donnant l'écriture en morse d'un mot ou d'une succession de mots. Cette fonction doit retourner le mot déchiffré ou la succession de mots déchiffrée.
- Exemple de retours attendus dans la console :

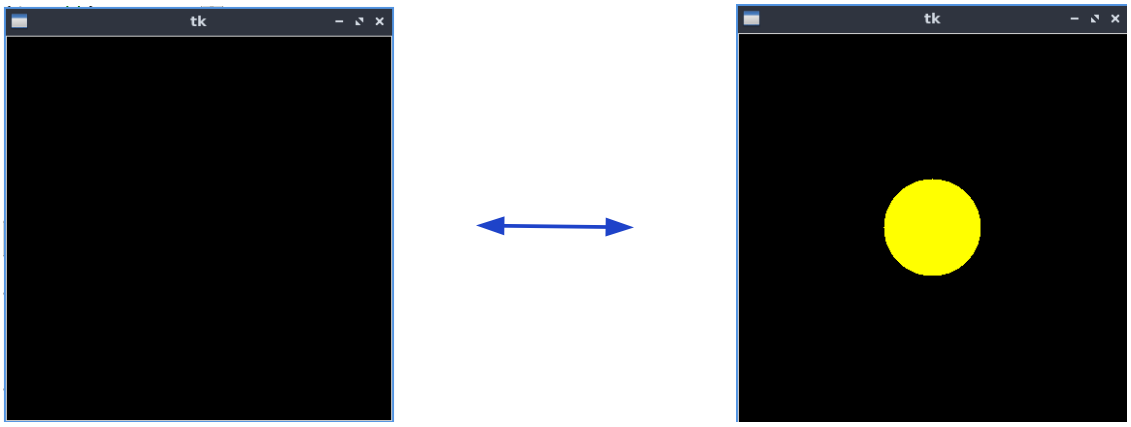
```
>>> m1 = morse('BONJOUR')
>>> print(m1)
 -... --- -. .--- --- ..- .-.
>>> dechiffrer(m1)
'BONJOUR'
>>> dechiffrer('.--')
'W'
>>> dechiffrer('-... .. . -.')
'BIEN'
>>> dechiffrer('-... . .- ..- / - . -- .--. ...')
'BEAU TEMPS'
```

Lever la main pour valider ce TP.

TP8 - * Traduire en morse, puis en signal lumineux

- Dupliquer le **TP7 .py** en **TP8 .py .**
- Importer la bibliothèque **tkinter** .
- Afficher une fenêtre carré de côté 400 pixels à fonc noir.
- Faire saisir à l'utilisateur un texte en **majuscule** sans ponctuation. Convertir ce texte en texte en morse avec la fonction **morse( )** et l'enregistrer dans une chaine de type **str**.
- Rédiger des fonctions et utiliser la méthode **canvas.after()** pour créer une animation qui envoie un signal lumineux correspondant au texte en morse. Le signal lumineux sera symbolisé par un cercle jaune qui clignote, en respectant les impulsions et les silences mentionnés dans les conventions du TP5.
- Ci-dessous l'affichage dans la console attendu, puis animation attendue qui alterne entre les deux captures d'écran ci-après.

```
>>> %Run TP8-morse-lumineux.py
  Saisir un texte en majuscule :
  SOS
```



Lever la main pour valider ce TP.

TP9 - Fonction récursive pour la suite de Fibonacci

- Créer un nouveau fichier **TP9 .py** et l'enregistrer dans le dossier **TP-Chapitre4** sur votre compte.
- Rédiger une fonction **f(n)** récursive basique sans dictionnaire, comme dans le chapitre 2 qui calcule les valeurs de la suite de Fibonacci.
- Depuis la bibliothèque **time** importer la fonction time, afin de rajouter dans le code le calcul du temps d'exécution, comme dans l'exemple du cours sur les coefficients binomiaux.
- Exemple d'affichage attendu dans la console :

```
f( 36 ) =  14930352
Temps d'exécution en secondes : 5.2570648193359375
```

Lever la main pour valider ce TP.

TP10 - Fonction récursive améliorée pour la suite de Fibonacci

- Dupliquer le fichier **TP9 .py** en **TP10 .py** .
- Améliorer la rapidité d'exécution de la fonction récursive **f(n)** , en utilisant un dictionnaire pour stocker les valeurs déjà calculées.
- Exemple d'affichage attendu dans la console :

```
f( 36 ) =  14930352
Temps d'exécution en secondes : 0.0002677440643310547
```

Lever la main pour valider ce TP.

TP11 - * Fonction récursive pour la suite de Stern-Brocot

- Créer un nouveau fichier **TP11 .py** et l'enregistrer dans le dossier **TP-Chapitre4** sur votre compte.
- Rédiger une fonction récursive **s(n)** qui calcule les valeurs de la suite de **Stern-Brocot**.
  La suite de Stern-Brocot est définie par :

  $s(0) = 0$
  $s(1) = 1$
  $s(2n) = s(n)$ pour tout entier naturel n non nul
  $s(2n+1) = s(n) + s(n+1)$ pour tout entier naturel n non nul

- Optimiser l'exécution de la fonction **s(n)** en utilisant un **dictionnaire** pour mémoriser les valeurs déjà calculées.
- Premier affichage attendu dans la console :

```
s(0) = 0
s(1) = 1
s(2) = 1
s(3) = 2
s(4) = 1
s(5) = 3
:
s(45) = 12
s(46) = 7
s(47) = 9
s(48) = 2
s(49) = 9
s(50) = 7
```

- Deuxième affichage attendu dans la console :

```
>>> s(10**20)
263945319
>>> s(10**30)
2641828456999
>>> s(10**40)
1562016307138296 9
>>> s(10**50)
167927140074779758557
```

Lever la main pour valider ce TP.