IngeSUP - TD 10 - Les structures de données 2

> *“Le langage structure tout de la relation inter-humaine.”*

Jacques Lacan

Exercice 10.1

Soit le dictionnaire :

```
d = {'nom': 'Dupuis', 'prenom': 'Jacque', 'age': 30}
```

1. Corriger l'erreur dans le prénom, la bonne valeur est 'Jacques'.

Entrée [15]:

```
d = {'nom': 'Dupuis', 'prenom': 'Jacque', 'age': 30}

d["prenom"]="Jacques"

print(d)
```

```
{'nom': 'Dupuis', 'prenom': 'Jacques', 'age': 30}
```

2. Afficher la liste des clés du dictionnaire.

Entrée [5]:

```
for i in d.keys():
    print(i)
```

```
nom
prenom
age
```

3. Afficher la liste des valeurs du dictionnaire.

Entrée [6]:

```
for i in d.values():
    print(i)
```

```
Dupuis
Jacques
30
```

4. Afficher la liste des paires clé/valeur du dictionnaire.

Entrée [16]:

```
for i,j in d.items():
    print(i,j)

# solution alternative
for i in d.keys():
    print(i, d[i])
```

```
nom Dupuis
prenom Jacques
age 30
nom Dupuis
prenom Jacques
age 30
```

5. Ecrire la phrase "Jacques Dupuis a 30 ans" en utilisant `print` sur les clés du dictionnaire ***d***.

Entrée [10]:

```
print(d["prenom"], d["nom"], "a", d["age"], "ans")
```

```
Jacques Dupuis a 30 ans
```

Exercice 10.2

On représente des matrices carrées sous la forme de listes de listes de nombres.

Entrée [11]:

```
carre3 = [
    [2, 7, 3],
    [9, 5, 1],
    [4, 3, 8]
]

carre4 = [
    [4, 5, 11, 14],
    [15, 10, 8, 1],
    [6, 3, 13, 12],
    [9, 16, 2, 7]
]

print(len(carre4))
```

```
4
```

1. Quelle est la valeur de la longueur de carre4 ?

Réponse: 4

2. Supposons qu'on veuille accéder à la deuxième valeur de la deuxième ligne de *carre3*. Quelle instruction python doit-on taper ?

Entrée [12]:

```
print(carre3[1][1])
```

```
5
```

3. Supposons qu'on veuille accéder à la troisième valeur de la première ligne de *carre3*. Quelle instruction python doit-on taper ?

Entrée [ ]:

```
print(carre3[0][2])
```

4. Quelle instruction permet de récupérer la valeur 3 de *carre4* ?

Entrée [ ]:

```
print(carre4[2][1])
```

5. On propose le code suivant :

Entrée [23]:

```
def somme_ligne(carre, n):
    """
    carre est une liste de listes de nombres
    n est un nombre entier
    """
    somme = 0
    for nombre in carre[n]:
        somme = somme + nombre
    return somme
```

Que vaut *somme_ligne(carre4, 2)* ? À quoi sert cette fonction ?

Réponse: 34

6. Définissez la fonction `lignes_magiques` qui prend un carré en paramètre et qui vérifie que les sommes des nombres de chaque ligne sont égales.

Entrée [51]:

```
def lignes_magiques(carre):
    e=set({})
    for i in range(len(carre)):
        e.add(somme_ligne(carre, i))

    if len(e)==1:
        return True
    return False

lignes_magiques(carre4)

print(lignes_magiques(carre4))

#alternative sans ensemble :

def lignes_magiques(carre):
    resultat=True
    ref=somme_ligne(carre,0)
    for i in range(1,len(carre)):
        if somme_ligne(carre,i)!=ref:
            resultat=False
    return resultat

print(lignes_magiques(carre4))
```

```
True
True
```

Exercice 10.3

Les nombres complexes sont décrits par un couple de réels qui définit leur partie réelle et leur partie imaginaire.

On pourra définir un complexe par un tuple[float,float]. Ainsi le nombre complexe $2 + 3i$ sera représenté par le tuple $(2.0, 3.0)$, le nombre i par $(0.0, 1.0)$ et un réel r par $(r, 0.0)$.

1. Ecrivez les fonctions `partie_relle()` et `partie_imaginaire()` renvoyant respectivement la partie réelle et la partie imaginaire d'un nombre complexe `c` défini comme un tuple.

Entrée [35]:

```
def partie_reelle(c):
    return c[0]

def partie_imaginaire(c):
    return c[1]

exemple=(2,3)   # test pour 2+3i
print(partie_reelle(exemple))
print(partie_imaginaire(exemple))
```

```
2
3
```

2. Ecrivez la fonction `addition_complexe()` renvoyant la somme de deux nombres complexes passés en argument.

Entrée [36]:

```
def addition_complexe(c1,c2):
    partieR=c1[0]+c2[0]
    partieI=c1[1]+c2[1]
    return (partieR,partieI)

print(addition_complexe((2,3),(5,6)))
```

```
(7, 9)
```

On rappelle que le produit de deux nombres complexes $(a + bi)$ et $(c + di)$ est donné par

$$(a + bi) * (c + di) = (ac - bd) + (ad + bc)i$$

.

3. Ecrire une fonction `produit_complexe()` telle que produit_complexe(c1, c2) renvoie le produit des nombres complexes `c1` et `c2` .

Entrée [38]:

```
def produit_complexe(c1,c2):
    partieR=c1[0]*c2[0]-c1[1]*c2[1]
    partieI=c1[0]*c2[1]+c1[1]*c2[0]
    return (partieR,partieI)

print(produit_complexe((2,3),(5,6)))
```

```
(-8, 27)
```

Exercice 10.4

On définit un point par un tuple (x, y) et une liste de points par une liste de tuple **points**.

1. Ecrivez une fonction `distance()` qui détermine la distance entre tous les points distincts de la liste `points` entrée en paramètre. On affichera la distance maximale entre deux points distincts. Testez votre fonction avec `points=[(1,3),(0,1),(2,4)]` .

On rappelle que la distance d entre un point P1 de coordonnées (x_1, y_1) et un point P2 de coordonnées (x_2, y_2) s'obtient par :

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

On ne s'occupera pas de la redondance de certains calculs. (En effet la distance entre les points P1 et P2 est identique à la distance entre les points P2 et P1).

Entrée [76]:

```
def distance(points):
    #on va faire la liste des distances entre deux points
    L=[]
    for point1 in points:
        for point2 in points:
            dist=(((point1[0]-point2[0])**2)+((point1[1]-point2[1])**2))**(1/2)
            L.append(dist)
    return max(L)

distance([(1,3),(0,1),(2,4)])
```

Out[76]:

```
3.605551275463989
```

Exercice 10.5

---

1. Ecrivez une fonction nommée `compte_car()` qui accepte une chaine de caractères et qui renvoie l'occurence des caractères contenus dans la chaine sous forme de dictionnaire.

Entrée [60]:

```
def compte_car(chaine):
    dico={}
    for lettre in chaine:
        if lettre not in dico.keys(): #si la lettre ne fait pas encore partie du dictionnaire
            dico[lettre]=1
        else:
            dico[lettre]+=1
    return dico

print(compte_car("afgzy,&)!!djfijaijdjqokojdijiofjoeijf!"))    # TEST
```

```
{'a': 2, 'f': 4, 'g': 1, 'z': 1, 'y': 1, ',': 1, '&': 1, ')': 1, '!': 3, 'd': 3, 'j': 8, 'i': 5, 'q': 1, 'o': 4, 'k': 1,
'e': 1}
```

2. Ecrivez une fonction nommée `compte_mots_ligne()` qui accepte une chaine de caractères et qui renvoie l'occurence des mots contenus dans la chaine sous forme de dictionnaire. (Rappel: Les mots sont séparées par le caractère espace).

Entrée [61]:

```
def compte_mots_ligne(chaine):
    liste=chaine.split(" ")

    dico={}
    for mot in liste:
        if mot not in dico.keys(): #si la lettre ne fait pas encore partie du dictionnaire
            dico[mot]=1
        else:
            dico[mot]+=1
    return dico

print(compte_mots_ligne("Bonjour ceci est un texte et ceci est une phrase"))
```

```
{'Bonjour': 1, 'ceci': 2, 'est': 2, 'un': 1, 'texte': 1, 'et': 1, 'une': 1, 'phrase': 1}
```

À l'aide de la fonction précédente, il est maintenant possible de déterminer les mots les plus représentés (hors caractères spéciaux) dans un fichier composé de plusieurs lignes ou dans une variable.

3. Ecrivez une fonction `compte_mots_texte()` qui accepte une variable et retourne l'occurence des mots contenus dans la variable sous forme de dictionnaire. Testez avec la variable `lafontaine_txt` entrée ci-dessous.

Entrée [62]:

```
lafontaine_txt = """Rien ne sert de courir ; il faut partir à point.\
Le Lièvre et la Tortue en sont un témoignage.\
Gageons, dit celle-ci, que vous n'atteindrez point\
Sitôt que moi ce but. - Sitôt ? Etes-vous sage ?\
Repartit l'animal léger.\
Ma commère, il vous faut purger\
Avec quatre grains d'ellébore.\
- Sage ou non, je parie encore.\
Ainsi fut fait : et de tous deux\
On mit près du but les enjeux :\
Savoir quoi, ce n'est pas l'affaire,\
Ni de quel juge l'on convint.\
Notre Lièvre n'avait que quatre pas à faire ;\
J'entends de ceux qu'il fait lorsque prêt d'être atteint\
Il s'éloigne des chiens, les renvoie aux Calendes,\
Et leur fait arpenter les landes.\
Ayant, dis-je, du temps de reste pour brouter,\
Pour dormir, et pour écouter\
D'où vient le vent, il laiss la Tortue\
Aller son train de Sénateur.\
Elle part, elle s'évertue ;\
Elle se hâte avec lenteur.\
Lui cependant méprise une telle victoire,\
Tient la gageure à peu de gloire,\
Croit qu'il y va de son honneur\
De partir tard. Il broute, il se repose,\
Il s'amuse à toute autre chose\
Qu'à la gageure. A la fin quand il vit\
Que l'autre touchait presque au bout de la carrière,\
Il partit comme un trait ; mais les élans qu'il fit\
Furent vains : la Tortue arriva la première.\
Eh bien ! lui cria-t-elle, avais-je pas raison ?\
De quoi vous sert votre vitesse ?\
Moi, l'emporter ! et que serait-ce\
Si vous portiez une maison ?"""
```

Entrée [74]:

```
#je ne suis pas sur de voir la différence avec l'exercice précédent ? la chaine de caractère n'ayant pas de retour à la ligne
print(lafontaine_txt)

#a part qu'on néttoie les caractères spéciaux

def compte_mots_texte(texte):
    for i in ";.,':?-!":
        texte=texte.replace(i," ")
    return compte_mots_ligne(texte)


compte_mots_texte(lafontaine_txt)
```

```
Rien ne sert de courir ; il faut partir à point.Le Lièvre et la Tortue en sont un témoignage.Gageons, dit celle-ci, que vou
s n'atteindrez pointSitôt que moi ce but. - Sitôt ? Etes-vous sage ?Repartit l'animal léger.Ma commère, il vous faut purger
Avec quatre grains d'ellébore.- Sage ou non, je parie encore.Ainsi fut fait : et de tous deuxOn mit près du but les enjeux
:Savoir quoi, ce n'est pas l'affaire,Ni de quel juge l'on convint.Notre Lièvre n'avait que quatre pas à faire ;J'entends de
ceux qu'il fait lorsque prêt d'être atteintIl s'éloigne des chiens, les renvoie aux Calendes,Et leur fait arpenter les land
es.Ayant, dis-je, du temps de reste pour brouter,Pour dormir, et pour écouterD'où vient le vent, il laiss la TortueAller so
n train de Sénateur.Elle part, elle s'évertue ;Elle se hâte avec lenteur.Lui cependant méprise une telle victoire,Tient la
gageure à peu de gloire,Croit qu'il y va de son honneurDe partir tard. Il broute, il se repose,Il s'amuse à toute autre cho
seQu'à la gageure. A la fin quand il vitQue l'autre touchait presque au bout de la carrière,Il partit comme un trait ; mais
les élans qu'il fitFurent vains : la Tortue arriva la première.Eh bien ! lui cria-t-elle, avais-je pas raison ?De quoi vous
sert votre vitesse ?Moi, l'emporter ! et que serait-ceSi vous portiez une maison ?
```

Out[74]:

```
{'Rien': 1,
 'ne': 1,
 'sert': 2,
 'de': 9,
 'courir': 1,
 '': 43,
 'il': 8,
 'faut': 2,
 'partir': 2,
 'à': 5,
 'point': 1,
 'Le': 1,
 'Lièvre': 2,
 'et': 4,
 'la': 8,
 'Tortue': 2,
 'en': 1,
 'sont': 1,
 'un': 2,
 'témoignage': 1,
 'Gageons': 1,
 'dit': 1,
 'celle': 1,
 'ci': 1,
 'que': 4,
 'vous': 5,
 'n': 3,
 'atteindrez': 1,
 'pointSitôt': 1,
 'moi': 1,
 'ce': 2,
 'but': 2,
 'Sitôt': 1,
 'êtes': 1,
 'sage': 1,
 'Repartit': 1,
 'l': 5,
 'animal': 1,
 'léger': 1,
 'ma': 1,
 'commere': 1,
 'purgerAvec': 1,
 'quatre': 2,
 'grains': 1,
 'Sage': 1,
 'ou': 1,
 'non': 1,
 'je': 3,
 'm': 1,
 'encore': 1,
 'Ainsi': 1,
 'fut': 1,
 'fait': 3,
 'tous': 1,
 'deuxOn': 1,
 'mit': 1,
 'près': 1,
 'du': 2,
 'les': 4,
```

4. Déterminez l'occurence des mots **Lièvre** et **Tortue** dans la variable `lafontaine_txt` .

```
Entrée [1]:
```

```
# j'ai pas le fichier a dispo mais je tente une correction sans pouvoir tester :
fichier=open(lafontaine_txt.txt, "r")
texte=fichier.read()
texte=texte.replace("\n"," ") # je remplace les retours a la ligne par des espaces
#une autre méthode serait d'utiliser la fonction de la question 2, en faisant un dictionnaire par liste
#puis de concaténer les dictionnaires avec .update

dico=compte_mots_texte(texte)
print(dico("Lièvre"))
print(dico("Tortue"))
```

Corrigé du TD 10

Vous pouvez retrouver le corrigé de ce TD ici (Corrig%C3%A9s/Corrig%C3%A9_TD%2010.ipynb).