EXERCICE 1

```
## -*- coding: utf-8 -*-
################################################################################


#Tests possibles:
#D1 = { 'Aude': 'Paris', 'Patrice': 'Rennes', 'Sophie': 'Lyon' }
#clear(D1)

#d=copy(D1)
#print(D1)
#print(d)
#D1['Aude']='Strasbourg'
#print(D1)
#print(d)

#print(get(D1,'Aude'))
#print(get(D1,'Tom'))
#print(get(D1,'Tom','Cl?? absente'))

#D1 = [ ('Aude', 'Paris'), ('Patrice', 'Rennes'), ('Sophie', 'Lyon') ]
#print(dict_(D1))
################################################################################


### Fonction qui efface toutes les cl??s et retourne le dictionnaire vide###
def clear(d):
    '''Fonction qui efface toutes les cl??s du dictionnaire d'''
    for k in list(d):
        del d[k]
    return d

### Fonction qui copie toutes les cl??s et retourne un autre dictionnaire ###
def copy(d):
    d_copy={}
    '''Fonction qui copie toutes les cl??s du dictionnaire d'''
    for k in list(d):
        d_copy[k]=d[k]
    return d_copy

### Fonction qui teste la pr??sence d'une cl?? et retourne renvoie default en ca
s d'absence ###
def get(d,cle,default = None):
    '''Fonction qui copie imite la m??thode .get() du dictionnaire d'''
    if cle in d:
        return d[cle]
    return default

### Fonction qui g??n??re un dictionnaire ?? partir d'une liste de 2-uplets ###
def dict_(t):
    '''Fonction qui g??n??re un dictionnaire ?? partir d'une liste de 2-uplet
s'''
    d={}
    for e in t:
        d[e[0]]=e[1]
    return d
```

: **EXERCICE 2**

```
#La fonction gen n'est pas codée par les élèves
def gen(n=1000):#par défaut on lit 1000 lignes
    '''Fonction qui génére et retourne une liste de tuples (prenom, n°TEL)
     à partir du fichier Prenoms.txt fourni'''
    assert n<=10000#on ne dépasse pas 10000 lignes
    tab=[]
    with open('Prenoms.txt' ,'r') as f:
        for i,ligne in enumerate(f):
            num=''
            l=ligne.split()
            if i<n:#les x premieres lignes
                for mot in l[1:]:#on ne récupère pas le nom
                    num+=mot
            else:
                break
            num='0'+num#on ajoute un 0 en tête
            tab.append((l[0],num))
    f.close()
    return tab

annuairetableau=gen(10000)#on change la taille du tableau ici

#QUESTION 1:
def recherche1(p,t):
    for i in range(len(t)):
        if p==t[i][0]:
            return t[i][1]
    return None
print(recherche1('aapo',annuairetableau))
```

```
0678735166
```

:

```
#QUESTION 2:
#on génère le dictionnaire avec notre fonction
annuairedico=dict_(annuairetableau)
#on génère le dictionnaire le constructeur de python
annuairedico=dict(annuairetableau)
#Résultats pour n=5
#{'aaliyah': '0516438897',
#'aapeli': '0177739542',
# 'aapo': '0678735166',
# 'aaren': '0746881696',
# 'aarne': '0574179969'}
```

:

```
#QUESTION 3:
def recherche2(p,d):
    return d.get(p,None)
print(recherche2('aapi',annuairedico))
```

```
None
```

:

```
#QUESTION 4:
from time import perf_counter
#recherche d'un prénom en début de tableau n=10000
ti=perf_counter()
recherche1('aapo',annuairetableau)
tf=perf_counter()
t1=tf-ti
ti=perf_counter()
recherche2('aapo',annuairedico)
tf=perf_counter()
t2=tf-ti
print('Recherche Tableau: ',t1,' Recherche Dico: ',t2)
```

```
Recherche Tableau:  6.481899998789231e-05  Recherche Dico:  3.135200
0007700553e-05
```

]:

```
#recherche d'un prénom en milieu de tableau n=10000
ti=perf_counter()
recherche1('gordon',annuairetableau)
tf=perf_counter()
t1=tf-ti
ti=perf_counter()
recherche2('gordon',annuairedico)
tf=perf_counter()
t2=tf-ti
print('Recherche Tableau: ',t1,' Recherche Dico: ',t2)
```

```
Recherche Tableau:  0.0005222359999947912  Recherche Dico:  3.218599
999854632e-05
```

]:

```
#recherche d'un prénom en fin de tableau n=10000
ti=perf_counter()
recherche1('zvi',annuairetableau)
tf=perf_counter()
t1=tf-ti
ti=perf_counter()
recherche2('zvi',annuairedico)
tf=perf_counter()
t2=tf-ti
print('Recherche Tableau: ',t1,' Recherche Dico: ',t2)
```

```
Recherche Tableau:  0.00141653899999028  Recherche Dico:  3.76219999
7936477e-05
```

La durée d'éxécution pour la recherche en dico est "constante" et de l'ordre de 1e-5 s. Ceci est cohérent avec une complexité théorique O(1)
La durée d'éxécution pour la recherche en tableau croit avec la position de la séquence. Ceci est cohérent avec une complexité théorique O(n)

]:**EXERCICE 3**

```
stock={
9788806222093:{'auteur':'Victor Hugo','titre':"Les misérables",'nombre':5},
9780671201821:{'auteur':'Robert Merle','titre':'Un animal doué de raison','nombr
e':6},
9783518399989:{'auteur':'Alain Fournier','titre':'Le grand Meaulnes','nombre':1
},
9782072864537:{'auteur':'Victor Hugo','titre':'Notre dame de Paris','nombre':4},
9781976166471:{'auteur':'Victor Hugo','titre':'Notre dame de Paris','nombre':3},
9782070518425: {'auteur':'J.K. Rowling','titre':"Harry Potter à l'école des sorc
iers",'nombre':3},
9782070643059: {'auteur':'J.K. Rowling','titre':"Harry Potter et la coupe de fe
u",'nombre':0}
}

#QUESTION 1:
def auteurs(b):
    #cette fonction affiche plusieurs fois Victor Hugo... On peut donc optimiser
    liste=[]
    for cle in b:
        liste.append(b[cle]['auteur'])
    return liste
print("liste des auteurs de la base :",auteurs(stock))
```

```
liste des auteurs de la base : ['Victor Hugo', 'Robert Merle', 'Alai
n Fournier', 'Victor Hugo', 'Victor Hugo', 'J.K. Rowling', 'J.K. Row
ling']
```

]:

```
def auteursv2(b):
    liste=[]
    for cle in b:
        if b[cle]['auteur'] not in liste:
            liste.append(b[cle]['auteur'])
    return liste
print("liste des auteurs de la base :",auteursv2(stock))
```

```
liste des auteurs de la base : ['Victor Hugo', 'Robert Merle', 'Alai
n Fournier', 'J.K. Rowling']
```

]:

```
def auteursv3(b):
    return [b[k]['auteur'] for k in b]
print("liste des auteurs de la base :",auteursv3(stock))
```

```
liste des auteurs de la base : ['Victor Hugo', 'Robert Merle', 'Alai
n Fournier', 'Victor Hugo', 'Victor Hugo', 'J.K. Rowling', 'J.K. Row
ling']
```

]:

```
#Compréhension sans doublons
#On crée un dictionnaire avec les auteurs en clés (pas de doublons!) et None en
 valeur
#On récupère les clés avec la méthode keys()
#On construit une liste avec list()
def auteursv4(b):
    return list({v['auteur']:None for v in b.values()}.keys())
print("liste des auteurs de la base :",auteursv4(stock))
```

```
liste des auteurs de la base : ['Victor Hugo', 'Robert Merle', 'Alai
n Fournier', 'J.K. Rowling']
```

]:

```
#QUESTION 2:
def livresempruntables(b):
    liste=[]
    for cle in b:
        if b[cle]['nombre'] > 0 and ( b[cle]['titre'] not in liste):
            liste.append(b[cle]['titre'])
    return liste
print("liste des titres empruntables :",livresempruntables(stock))
```

```
liste des titres empruntables : ['Les misérables', 'Un animal doué d
e raison', 'Le grand Meaulnes', 'Notre dame de Paris', "Harry Potter
à l'école des sorciers"]
```

]:

```
#QUESTION 3:
def titres_auteurs1(b):
    dic={}
    for cle in b:
        if b[cle]['titre'] not in dic :
            dic [b[cle]['titre']] =b[cle]['nombre']
        else :
            n1=dic [b[cle]['titre']]
            n2=b[cle]['nombre']
            dic [b[cle]['titre']]=n1+n2

    return dic
print("liste des titres auteurs :",titres_auteurs1(stock))
```

```
liste des titres auteurs : {'Les misérables': 5, 'Un animal doué de
raison': 6, 'Le grand Meaulnes': 1, 'Notre dame de Paris': 7, "Harry
Potter à l'école des sorciers": 3, 'Harry Potter et la coupe de fe
u': 0}
```

]:

```
#Par compréhension avec choix de l'auteur
def titres_auteurs2(b,n):
    l=[(b[c]['titre'],b[c]['nombre']) for c in b.keys() if b[c]['auteur']==n]
    d={}
    for elt in l:
        d[elt[0]]=d.get(elt[0],0)+elt[1]#si le titre n'existe pas on 0+elt[1]
    return d
print("liste des titres auteurs :",titres_auteurs2(stock,'Victor Hugo'))
```

```
liste des titres auteurs : {'Les misérables': 5, 'Notre dame de Pari
s': 7}
```

]:

```
#QUESTION 4:
stock2={
9782070643059: {'auteur':'J.K. Rowling','titre':"Harry Potter et la coupe de fe
u",'nombre': 10},
9782070556854: {'auteur':'J.K. Rowling','titre':"Harry Potter et l'Ordre du phén
ix",'nombre': 7},
9782070615360: {'auteur':'J.K. Rowling','titre':"Harry Potter et les Reliques de
la Mort",'nombre': 7}

}
def fusion(b1,b2):
    for cle in b2:
        if cle not in b1:
           b1[cle]=b2[cle]
        else:
            n1=b1[cle]['nombre']
            n2=b2[cle]['nombre']
            b1[cle]['nombre']=n1+n2
    return b1
# Test
print("fusion :",fusion(stock,stock2))
print(stock[9782070643059])
```

```
fusion : {9788806222093: {'auteur': 'Victor Hugo', 'titre': 'Les mis
érables', 'nombre': 5}, 9780671201821: {'auteur': 'Robert Merle', 't
itre': 'Un animal doué de raison', 'nombre': 6}, 9783518399989: {'au
teur': 'Alain Fournier', 'titre': 'Le grand Meaulnes', 'nombre': 1},
9782072864537: {'auteur': 'Victor Hugo', 'titre': 'Notre dame de Par
is', 'nombre': 4}, 9781976166471: {'auteur': 'Victor Hugo', 'titre':
'Notre dame de Paris', 'nombre': 3}, 9782070518425: {'auteur': 'J.K.
Rowling', 'titre': "Harry Potter à l'école des sorciers", 'nombre':
3}, 9782070643059: {'auteur': 'J.K. Rowling', 'titre': 'Harry Potter
et la coupe de feu', 'nombre': 10}, 9782070556854: {'auteur': 'J.K.
Rowling', 'titre': "Harry Potter et l'Ordre du phénix", 'nombre':
7}, 9782070615360: {'auteur': 'J.K. Rowling', 'titre': 'Harry Potter
et les Reliques de la Mort', 'nombre': 7}}
{'auteur': 'J.K. Rowling', 'titre': 'Harry Potter et la coupe de fe
u', 'nombre': 10}
```

EXERCICE 4

]:
```
#QUESTION 1:
#Dict_an_fr
danfr={'one':'un','two':'deux','three':'trois','four':'quatre','five':'cinq','si
x':'six','seven':'sept'}
#Dict_fr_al
dfral={'un':'eins','deux':'zwei','trois':'drei','quatre':'vier','cinq':'fünf','s
ix':'sechs','sept':'sieben'}
def traduction(l,d):
    list=[]
    for mots in l:
        list.append(d[mots])
    return list
print(traduction(['seven','one'],danfr))
```

```
['sept', 'un']
```

]:
```
#QUESTION 2:
def inverse(d):
    dico={}
    for cle in d:
        dico[d[cle]]=cle
    return dico
print(inverse(danfr))
```

```
{'un': 'one', 'deux': 'two', 'trois': 'three', 'quatre': 'four', 'ci
nq': 'five', 'six': 'six', 'sept': 'seven'}
```

]:
```
#QUESTION 3:
def composition1(d1,d2):
    dico={}
    for mot1 in d1:
        dico[mot1]=d2[d1[mot1]]
    return dico
print(composition1(danfr,dfral))
#Par compréhension avec itération simultanée via la fonction zip
def composition2(d1,d2):
    return {c:v for c,v in zip(d1.keys(),d2.values())}
print(composition2(danfr,dfral))
```

```
{'one': 'eins', 'two': 'zwei', 'three': 'drei', 'four': 'vier', 'fiv
e': 'fünf', 'six': 'sechs', 'seven': 'sieben'}
{'one': 'eins', 'two': 'zwei', 'three': 'drei', 'four': 'vier', 'fiv
e': 'fünf', 'six': 'sechs', 'seven': 'sieben'}
```

EXERCICE 5

```
# Question 1
def calculer_indice(mot):
    """ Réordonne les lettres d'un mot par ordre alphabétique : c'est l'indice d
u mot """
    liste = list(mot)
    liste2=sorted(liste)
    indice=" "
    for i in liste2:
        indice =indice+i
    return indice

print("--- Indice d'un mot ---")
mot = "KAYAK"
indice = calculer_indice(mot)
print("mot =",mot)
print("indice =",indice)
```

```
--- Indice d'un mot ---
mot = KAYAK
indice =  AAKKY
```

]:

```
# Question 2:

def anagrammes(mot1,mot2):
    """ Test si deux mots sont annagrammes
    C'est exactemment tester si ils ont le même indice """
    indice1 = calculer_indice(mot1)
    indice2 = calculer_indice(mot2)
    if indice1 == indice2:
        return True
    else:
        return False

print("--- Test anagramme ---")
mot1,mot2 = "PRIERES", "RESPIRE"
print("Les mots ",mot1," et ",mot2)
print("Sont anagrammes ?",anagrammes(mot1,mot2))
```

```
--- Test anagramme ---
Les mots  PRIERES  et  RESPIRE
Sont anagrammes ? True
```

]:

```python
# Question 3:
def dicoindice(liste):
    """ Cosntruit le dictionnaire clé -> forme de la forme
    indice -> valeur=[mot1,mot2,...] """
    dico = {}
    for mot in liste:
        indice = calculer_indice(mot)
        if indice in dico:
            dico[indice].append(mot)
        else:
            dico[indice] = [mot]
    return dico

print("--- Dico des indice d'une liste de mots ---")
liste = ["CRIME","COUCOU","PRIERES","MERCI","RESPIRE","REPRISE"]
dico = dicoindice(liste)
print("liste =",liste)
print("dico =",dico)
```

```
--- Dico des indice d'une liste de mots ---
liste = ['CRIME', 'COUCOU', 'PRIERES', 'MERCI', 'RESPIRE', 'REPRIS
E']
dico = {' CEIMR': ['CRIME', 'MERCI'], ' CCOOUU': ['COUCOU'], ' EEIPR
RS': ['PRIERES', 'RESPIRE', 'REPRISE']}
```