TD_3

December 3, 2019

TD — correction interactive

1 TD Listes / Dictionnaires

1.1 Dictionnaires

5. Comptez (dans un dictionnaire) le nombre d'occurrence de chaque élément contenu dans une liste. Par exemple ["a","c","c","b","a"] -> {"a":2, "b":1, "c":2}
6. Prendre une liste de quadruplets (nom,prenom,année,profession) et créer un dictionnaire indexé par un couple nom prénom et qui liste année et profession pour chaque entrée. si le même nom et prénom apparait plusieurs fois, on n'enregistre que la première valeur rencontrée.
7. Faire une table qui enregistre les combinaisons de deux dés à six faces, avec comme clefs la valeur de la somme des deux dés et comme valeurs les combinaisons de deux dés.
8. Faire le crible d'Eratosthène, enregistré dans un dictionnaire : on commence avec comme clefs tous les entiers de 1 à n (fixé), et finalement on ne garde que les clefs qui sont des nombres premiers.

9. Faire l'inversion (clefs/valeurs) d'une table

[8]:
```
def comptage(liste):
    res = {}
    for element in liste:
        if element in res:
            res[element] = res[element] + 1
        else:
            res[element] = 1
    return res


def comptage2(liste):
    res = {}
    for element in liste:
        res[element] = res.get(element,0) + 1
    return res


print(comptage2(["a","c","c","b","a"]))
```

```
{'a': 2, 'c': 2, 'b': 1}
```

[10]:
```python
liste = [("Bogart","Humphrey",1922,"acteur mythique"),
        ("Bacall","Lauren",1935,"actrice mythique"),
        ("De Funes","Louis",1766,"clown")]

def index(liste):
    res = {}
    for (nom,prenom,annee,prof) in liste:
        if (nom,prenom) not in res:
            res[(nom,prenom)] = (annee,prof)
    return res

imdb = index(liste)

print(imdb.get(("Bogart","Humphrey")))
print(imdb.get(("Carey","Jim"),"qui ?"))
```

```
(1922, 'acteur mythique')
qui ?
```

[13]:
```python
def tirage():
    res = {}
    for i in range(1,7):
        for j in range(1,7):
            if (i+j) not in res:
                res[i+j] = [(i,j)]
            else:
                res[i+j].append((i,j))
    return res

from pprint import pprint
pprint(tirage())
```

```
{2: [(1, 1)],
 3: [(1, 2), (2, 1)],
 4: [(1, 3), (2, 2), (3, 1)],
 5: [(1, 4), (2, 3), (3, 2), (4, 1)],
 6: [(1, 5), (2, 4), (3, 3), (4, 2), (5, 1)],
 7: [(1, 6), (2, 5), (3, 4), (4, 3), (5, 2), (6, 1)],
 8: [(2, 6), (3, 5), (4, 4), (5, 3), (6, 2)],
 9: [(3, 6), (4, 5), (5, 4), (6, 3)],
 10: [(4, 6), (5, 5), (6, 4)],
 11: [(5, 6), (6, 5)],
 12: [(6, 6)]}
```

[15]:
```
probs = {}
t = tirage()
for v in t:
    probs[v] = len(t[v])/36

pprint(probs)
```

```
{2: 0.027777777777777776,
 3: 0.05555555555555555,
 4: 0.08333333333333333,
 5: 0.1111111111111111,
 6: 0.1388888888888889,
 7: 0.16666666666666666,
 8: 0.1388888888888889,
 9: 0.1111111111111111,
 10: 0.08333333333333333,
 11: 0.05555555555555555,
 12: 0.027777777777777776}
```

[16]:
```
def inverse(table):
    res = {}
    for x in table:
        if table[x] in res:
            res[table[x]].append(x)
        else:
            res[table[x]] = [x]
    return res

inverse({2:3,5:6,8:3,1:4,11:6})
```

[16]: {3: [2, 8], 6: [5, 11], 4: [1]}

[22]:
```
def eratosthene(n):
    crible = {}
    for i in range(2,n+1):
        crible[i] = True

    k = 2
    while k*k<= n:
        for q in range(2,(n//k)+1):
            if k*q in crible:
                del crible[k*q]
        k = k + 1

    return list(crible.keys())

print(eratosthene(100))
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73,
79, 83, 89, 97]
```