Python	Description	Complexité
`d[k] = v`	Ajout (ou modification) d'une association de `k` à `v`	O(1) en moyenne
`d[k]`	Accès à la valeur de clé `k`	O(1) en moyenne
`len(d)`	Nombre de clés de `d`	O(1) en moyenne
`for k in d:`	Parcourir les clés `k` de `d`	O(n) où n est le nombre de clés
`k in d`	Test si `k` est une clé de `d`	O(1) en moyenne
`d.copy()`	Copie de `d`	O(n) où n est le nombre de clés

- Un **dictionnaire** est une structure de données qui à chaque **clé** associe une **valeur**. Il possède les opérations suivantes :
  - Ajouter une association (clé, valeur).
  - Supprimer une association (clé, valeur).
  - Obtenir les valeurs associée à une clé donnée.
- Les dictionnaires de Python sont implémentés par table de hachage, ce qui demande une fonction de hachage pour les clés. Les types de base immuables (non modifiables) comme `int` ou `string` ont une fonction de hachage déjà définie. Les types mutables (modifiables) comme `list` ou `dict` ne doivent pas être utilisés comme clé d'un dictionnaire, mais peuvent être utilisés comme valeur.
- Exemple d'utilisation de dictionnaire :

```
d = {"rouge": (255, 0, 0), "jaune": (255, 255, 0)}
# "rouge" est une clé de  valeur associée (255, 0, 0)
d["rouge"] # donne (255, 0, 0)
d["blabla"] # donne une erreur
"rouge" in d # renvoie True
for k in d:
    print(k) # affiche "rouge" et "jaune"
len(d) # nombre de clés
```

- Exercice : Écrire une fonction `inverse(d)` renvoyant un dictionnaire `d2` « inverse » de `d`, c'est-à-dire tel que : `d[k] = v` $\iff$ `d2[v] = k`.

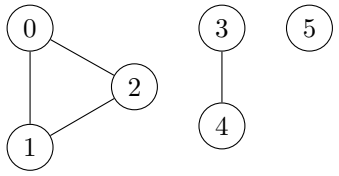
```
def inverse(d):
    d2 = {}
    for k in d:
        d2[d[k]] = k
    return d2
```

- Exercice : Écrire une fonction `frequent(L)` renvoyant l'élément le plus fréquent dans une liste `L` de taille n, en complexité O(n).

Solution : Sans dictionnaire, on pourrait compter combien de fois apparaît chaque élément de `L` mais cela demanderait une complexité O(n^2).

```
def frequent(L):
    d = {} # d[e] = nombre d'occurences de e dans L
    for e in L:
        if e in d:
            d[e] += 1
        else:
            d[e] = 1
    max = 0
    for k in d:
        if d[k] > max:
            max = d[k]
            res = k
    return res
```

- Au lieu de la représentation par matrice/liste d'un graphe G, on peut représenter G par un dictionnaire `d`, où `d[v]` est la liste (ou l'ensemble) des voisins du sommet `v`.
Exemple :




```
d = {0 : [1, 2], 1: [0, 2], 2: [0, 1],
     3: [4], 4: [3], 5: []}
```