DICTIONNAIRE ET HASHAGE

MP/MP* LYCÉE CARNOT

1. Dictionnaire

1.1. Définition.

Définition 1. Les dictionnaires sont des objets pouvant en contenir d'autres, à l'instar des listes. Cependant, au lieu de disposer ces informations dans un ordre précis, ils associent chaque objet contenu à une clé (la plupart du temps, une chaîne de caractères).

Un dictionnaire est un type modifiable. Nous pouvons donc d'abord créer un dictionnaire vide, puis le remplir. Contrairement aux listes, il n'y a pas de fonction spéciale (telle que `append()`) à appeler pour ajouter un objet.

```
dico = {}
dico['red'] = 'rouge'
dico['yellow'] = 'jaune'
dico['blue'] = 'bleu'
dico['green'] = 'vert'
dico['pink'] = 'rose'
```

On pourra afficher ce dictionnaire à l'écran facilement :

```
print(dico)
```

```
Solution . {'blue': 'bleu', 'pink': 'rose', 'green': 'vert',
 'yellow': 'jaune', 'red': 'rouge'}
```

Le dictionnaire apparaît comme une suite d'éléments séparés par des virgules, le tout entre accolades. Chaque élément est composé d'une paire d'objets (ici deux chaînes de caractères, mais on pourrait avoir d'autres types d'objets) séparés par un double point. Le premier objet est la clé et le second est une valeur.

Date: 2022-23.

On remarquera que les paires n'apparaissent pas dans l'ordre dans lequel elles ont été introduites. Rappelez-vous qu'un dictionnaire n'est pas ordonné.

Pour accéder à une valeur, il suffit de connaître sa clé :

```
print(dico['pink'])
```

affichera rose.

Les dictionnaires sont des objets. On peut donc leur appliquer des méthodes spécifiques. La méthode `keys()` renvoie les clés utilisées dans le dictionnaire :

```
print(dico.keys())
```

Solution .

Pour connaître les valeurs, la méthode à utiliser est `values()` :

```
print(dico.values())
```

Solution .

On peut aussi extraire du dictionnaire une séquence équivalente de tuples avec la méthode `items()`. Cela nous sera utile quand nous voudrons parcourir un dictionnaire avec une boucle.

```
print(dico.items())
```

Solution .

1.2. **Manipuler un dictionnaire.** La fonction intégrée `len()` permet de connaître le nombre d'entrées du dictionnaire :

```
len(dico)
```

donnera comme résultat 5. On a vu comment ajouter des entrées. On peut aussi remplacer la valeur correspondant à une clé. Par exemple :

```
dico['blue'] = 'bleu ciel'
```

Pour résumer, si la clé existe, la valeur correspondante est remplacée ; si elle n'existe pas, une nouvelle entrée est créée. On peut supprimer une entrée avec le mot-clé `del` :

```
del(dico['blue'])
print(dico)
```

Solution .

La méthode `pop()` supprime également la clé précisée, mais elle renvoie en plus la valeur supprimée. Cela peut être utile parfois.

```
couleur = dico.pop('pink')
print(couleur)
```

écrira rose. D'une façon analogue à ce qui se passe avec les listes et les tuples, l'instruction in permet de savoir si un dictionnaire contient une clé déterminée :

```
'red' in dico
```

donnera comme résultat True.
Si l'on donne une clé qui n'est pas dans le dictionnaire, cela provoquera une erreur :

```
print(dico['cyan'])
```

provoquera l'erreur : `KeyError: 'cyan'`
Pour pallier ce problème, il existe la méthode `get()` :

```
print(dico.get('red', 'inconnu'))
```

écrira rouge, car red est dans le dictionnaire.

```
print(dico.get('cyan', 'inconnu'))
```

écrira inconnu, car cyan n'est pas dans le dictionnaire. Le premier argument transmis à cette méthode est la clé de recherche, le second est la valeur que nous voulons obtenir en retour si la clé n'existe pas dans le dictionnaire

1.3. **Parcours d'un dictionnaire.** Vous pouvez utiliser une boucle for pour traiter successivement tous les éléments contenus dans un dictionnaire, mais attention :

- au cours de l'itération, ce sont les clés du dictionnaire qui seront successivement affectées à la variable de travail, et non les valeurs ;
- l'ordre dans lequel les éléments seront parcourus est imprévisible (puisqu'un dictionnaire n'est pas ordonné).

```
for i in dico:
print(i, dico[i])
```

écrira :

Solution .

La manière de procéder suivante est plus élégante, pour un résultat identique :

```
for cle, valeur in dico.items():
print(cle, valeur)
```

La méthode `items()` renvoie une suite de tuples `(clé, valeur)`. On peut aussi ne parcourir que les clés :

```
for cle in dico.keys():
print(cle)
```

ou que les valeurs :

```
for valeur in dico.values():
print(valeur)
```

1.4. **Trier un dictionnaire.** Il n'est pas possible de trier un dictionnaire, puisque, encore une fois, c'est une structure non ordonnée. Par contre, il est possible d'écrire un dictionnaire selon un certain ordre, mais il faut d'abord créer une liste de tuples (une liste, elle, peut être triée sans problèmes). Voici la ligne de code pour créer une liste de tuples triée selon les clés :

```
dico_trie = sorted(dico.items(), key=lambda x : x[0])
print(dico_trie)
```

Le résultat sera :

Solution .

Et voici la ligne de code pour créer une liste de tuples triée selon les valeurs :

```
dico_trie = sorted(dico.items(), key=lambda x : x[1])
print(dico_trie)
```

Le résultat sera :

Solution .

2. Hashage

Un des grand intérêt des dictionnaires est lié à la gestion de la mémoire (vive notamment) et la taille des fichiers. Il est donc souvent intéressant d'avoir une clé de taille réduite permettant d'identifier des fichiers lourds, par exemple un aperçu d'une image plutôt que l'image complète.

La fonction de hachage est une application (non injective) qui pour nous, avec les dictionnaires en Python, aux valeurs associe les clés.

2.1. **Premier exemple : hashage d'une chaîne de caractères par l'entier modulo 256 correspondant.** On considère des chaînes de caractères qui sont codées à partir de l'alphabet ASCII 8 bits. Ainsi, par exemple, puisque les caractères B, l, o et p correspondent aux valeurs 66, 108, 111 et 112 respectivement, le mot "Blop" est associée à l'entier

$$66 * 256^3 + 108 * 256^2 + 111 * 256 + 112 = 1114402672.$$

Exercice 1. (1) Écrivez une fonction qui prend en entrée une chaîne de caractères en ASCII 8 bits et renvoie l'entier associé. (On rappelle que la fonction `ord(c)` renvoie la valeur ASCII du caractère c).

Solution . Remarque : Cette fonction utilise le principe de la méthode de Horner pour évaluer le polynôme

$$a_0X^d + a_1X^{d-1} + ... + a_{d-1}X + ad$$

en $X = 256$, où $a_0, a_1, ...a_d$ sont les entiers correspondant aux lettres de la chaîne s passée en argument.

On utilise la fonction de hashage $h : x \mapsto (x \bmod 255)$.

(2) Écrivez la fonction h en Python qui prend en argument une chaîne de caractère, la converti en entier puis le hache.

Solution . On peut ensuite remarquer que hachage("chien") et hachage("niche") renvoient la même valeur (en l'occurrence 9).

Remarque : Dès que les mots sont un peu longs (c'est déjà le cas pour "chien" et "niche"), le résultat de la fonction hachage est un entier long (ce qui est indiqué en Python par la lettre "L" après la valeur numérique), bien que sa valeur soit petite (comprise entre 0 et 255).

Simplifier alors le calcul en utilisant plus intelligemment les propriétés de $\mathbb{Z}/n\mathbb{Z}$

Solution .

2.2. **Collisions.** Considérons l'exercice classique suivant.

Exercice 2. Si l'on considère un groupe de N personnes, quelle est la probabilité que deux d'entre elles soient nées le même jour de l'année (On

compte 365 jours dans une année) ? (donnez simplement une expression de la probabilité)

Solution .

Exercice 3. Quel est le rapport avec le hashage ?

Définition 2. On dit que l'on a une collision lorsque deux données ont la même valeur de hashage.

La probabilité d'avoir une collision en insérant n clés dans une table de hachage de taille 365 est la même que la probabilité d'avoir un conflit d'anniversaires.

De manière générale, en remplaçant 365 par la taille de la table de hachage, on a la probabilité d'avoir une collision en insérant n valeurs.

Exercice 4. Définir une fonction collisions de paramètres n et m pour qu'elle calcule la probabilité que l'on obtienne une collision en ajoutant n valeurs dans une table de hashage de taille m.

Solution .

Exercice 5. On considère le texte suivant non accentué et non ponctué :

c etait a megara faubourg de carthage dans les jardins d hamilcar les soldats qu il avait commandes en sicile se donnaient un grand festin pour

celebrer le jour anniversaire de la bataille d eryx et comme le maitre etait absent et qu ils se trouvaient nombreux ils mangeaient et ils buvaient en pleine liberte

Écrivez un programme qui compte les occurrences de chacune des lettres de l'alphabet dans ce texte, et qui en fait un histogramme comme l'exemple fictif ci-dessous :

```
  : ||||||||||||||||||||
a : ||||||||||||
d : ||||
e : ||||||||||||||||||||||||||
i : |||||||||
l : |||||||||
n : ||||||||||
o : |||||||
p : ||
r : ||||||||
s : ||||||||
t : ||||||||||||
u : ||||||
```

Solution .