NSI - Les dictionnaires

November 10, 2019

Le cours est aussi disponible sur https://www.math93.com/lycee/nsi-1ere/nsi-1ere.html

1 Cours et activités

1.1 Définition

Un **dictionnaire** en python est une sorte de liste mais au lieu d'utiliser des index, on utilise des clés alphanumériques.

L'ensemble des couples **clé** − **valeur** sont enregistrés sous la forme *clé:valeur* , l'ensemble de ces couples étant séparés par une virgule et placés entre deux accolades.

Les clés peuvent être des entiers (int), des chaînes de caractères (str) et même des tuples.

Par exemple :

[1]:
```
d={'clé1':'valeur1','clé2':'valeur2'}
print(d)
```

```
{'clé1': 'valeur1', 'clé2': 'valeur2'}
```

1.2 Comment créer un dictionnaire python ?

Pour **initialiser un dictionnaire** , on utile la syntaxe suivante qui va créer une variable de type dictionnaire, vide.

[2]:
```
a=dict()
b={}
```

On peut vérifier le type de la variable créée.

[3]:
```
type(a)
type(b)
# On peut par exemple entrer des noms en clés et les numéros de téléphone en␣
 ↪valeur.
a={'Marie': 62111111 ,'Moussa': 62111122,'Chérine': 62111111}
print(a)
```

```
{'Marie': 62111111, 'Moussa': 62111122, 'Chérine': 62111111}
```

1.3 Comment ajouter des valeurs dans un dictionnaire python ?

Pour **ajouter des valeurs** à un dictionnaire il faut indiquer une **clé** ainsi qu'une **valeur** sous la forme **nom_dico['clé']='valeur'**.

On peut par exemple entrer des noms en clés et les numéros de téléphone en valeur.

Remarque : un entier ne commence pas par 0 en python, si on veurt stocker la valeur complète 0611223344 il faudra passer les numéros en variables str.

[4]:
```
a['Pierre']=611223344
a['Franck']=611223599
a['Yannis']=611223355
print(a)
```

```
{'Marie': 62111111, 'Moussa': 62111122, 'Chérine': 62111111, 'Pierre':
611223344, 'Franck': 611223599, 'Yannis': 611223355}
```

- La **longueur d'un dictionnaire** est le nombre de clés. On l'obteint avec la fonction *len(nom_dico)*.
- L'ordre dans lequel apparaissent les paires clé-valeur dans le dictionnaire n'a pas d'importance, car on accède pas aux éléments d'un dictionnaire par un index, mais par une clé, qui n'est pas nécessairement un nombre.

[5]:
```
len(a)
```

[5]: 6

1.4 Un premier exercice

[6]:
```
## Exercice 1 : Question 1
# 1.a Créer un dictionnaire vide nommé *dico*.
# 1.b Ce dictionnaire comprendra en **clés** des mots en anglais et en␣
 ↪**valeurs** une traduction possible en français.
# 1.c Ajouter les clés-valeurs : 'house' : 'maison' ,   'kitchen' : 'cuisine' ,␣
 ↪  'bunk beds' : 'lits superposés' ,   'bed room' : 'pièce lit'
# 1.d Vérifier le résultat par un affichage du dictionnaire.
# 1.e Afficher sa longueur.
```

1.5 Comment récupérer une valeur dans un dictionnaire python?

Deux façons de procéder pour récupérer une valeur dans un dictionnaire :

- la **méthode get** : nom_dico.get('clé').
- la **méthode directe** : nom_dico['clé']

[7]:
```
a.get('Pierre')
```

[7]:
```
611223344
```

[8]:
```
a['Franck']
```

[8]:
```
611223599
```

Si la clé est introuvable, la méthode get ne renvoie rien. Vous pouvez donner une valeur à retourner par défaut.

[9]:
```
a.get('Marcus',"Ce nom n'existe pas danns le dictionnaire")
```

[9]:
```
"Ce nom n'existe pas danns le dictionnaire"
```

Faites de même avec votre dictionnaires dico.

[10]:
```
## Exercice 1 : Question 2
# Accéder aux valeurs dar les clés et tester un appel d'une clé qui n'est pas␣
 ↪dans votre dictionnaire.
```

1.6 Comment modifier une valeur dans un dictionnaire python?

On peut bien sûre modifier la valeur d'une clé.

[11]:
```
a['Pierre']=799887744
print(a)
```

```
{'Marie': 62111111, 'Moussa': 62111122, 'Chérine': 62111111, 'Pierre':
799887744, 'Franck': 611223599, 'Yannis': 611223355}
```

On pourrait par exemple dans le dictionnaire dico précédent modifier la valeur de la clé 'bed room' par 'chambre' ce qui est plus convenable.

[12]:
```
## Exercice 1 : Question 3
# Modifier la valeur de la clé 'bed room' par 'chambre'
```

1.7 Comment supprimer une valeur dans un dictionnaire python?

Pour supprimer un élément :

- on peut utiliser l'instruction intégrée **del** ,
- ou la méthode **pop()**.

Par exemple, si on souhaite supprimer la paire 'Pierre' : '611223344', on peut exécuter l'instruction : *del a['Pierre']* ou *a.pop('Pierre')*

[13]:
```
print('avant', a)
del a['Pierre']
print('après le del ',a)
a.pop('Yannis')
print('après le pop',a)
```

```
avant {'Marie': 62111111, 'Moussa': 62111122, 'Chérine': 62111111, 'Pierre':
799887744, 'Franck': 611223599, 'Yannis': 611223355}
après le del  {'Marie': 62111111, 'Moussa': 62111122, 'Chérine': 62111111,
'Franck': 611223599, 'Yannis': 611223355}
après le pop {'Marie': 62111111, 'Moussa': 62111122, 'Chérine': 62111111,
'Franck': 611223599}
```

[14]:
```
## Exercice 1 : Question 4
# Supprimer une entrée de votre dictionnaire
```

1.8 Comment parcourir les clés ou les valeurs dans un dictionnaire python?

1.8.1 Méthode key()

La méthode **keys()** renvoie une séquence contenant les clés du dictionnaire. Si nécessaire, cette séquence peut être convertie en une liste à l'aide de la fonction intégrée list() ou en tuple à l'aide de la fonction intégrée tuple().

[15]:
```
a.keys()
```

[15]:
```
dict_keys(['Marie', 'Moussa', 'Chérine', 'Franck'])
```

[16]:
```
list(a.keys())
```

[16]:
```
['Marie', 'Moussa', 'Chérine', 'Franck']
```

1.8.2 Méthode value()

La méthode **values()** renvoie une séquence contenant les valeurs mémorisées dans le dictionnaire.

On peut, comme pour les clés, utiliser les méthodes list() ou tuple() pour transformer ces séquences en listes ou en tuples.

[17]:
```
a.values()
```

[17]:
```
dict_values([62111111, 62111122, 62111111, 611223599])
```

1.8.3 Méthode item()

La méthode **items()** retourne une séquence de tuples, chaque tuple contenant deux éléments, la clé et la valeur correspondante. Par exemple :

[18]:
```
a.items()
```

[18]:
```
dict_items([('Marie', 62111111), ('Moussa', 62111122), ('Chérine', 62111111),
('Franck', 611223599)])
```

1.8.4 Parcours par une boucle

On peut aussi parcourir le dictionnaire par une boucle en utilisant les différentes méthodes.

[19]:
```
for clé in a:
    print("La clé ", clé, "est associée à la valeur", a[clé])
```

```
La clé  Marie est associée à la valeur 62111111
La clé  Moussa est associée à la valeur 62111122
La clé  Chérine est associée à la valeur 62111111
La clé  Franck est associée à la valeur 611223599
```

[20]:
```
for clé in a.keys():
    print("Les clés du dictionnaire sont ", clé)
```

```
Les clés du dictionnaire sont  Marie
Les clés du dictionnaire sont  Moussa
Les clés du dictionnaire sont  Chérine
Les clés du dictionnaire sont  Franck
```

[21]:
```
for valeur in a.values():
    print("Les valeurs du dictionnaire sont ", valeur)
```

```
Les valeurs du dictionnaire sont  62111111
Les valeurs du dictionnaire sont  62111122
Les valeurs du dictionnaire sont  62111111
Les valeurs du dictionnaire sont  611223599
```

[22]:
```
## Exercice 1 : Question 5
# 5.1 Afficher les clés et valeurs de votre dictionnaire avec les différentes␣
 ↪méthodes proposées
# 5.2 Afficher les clés de votre dictionnaire avec une boucle.
# 5.3 Afficher les valeurs de votre dictionnaire avec une boucle.
```

1.9 Comment copier un dictionnaire : méthode copy()

Supposons que l'on souhaite effectuer une copie du dictionnaire a précédent sous le nom a2.

[23]:
```
print(a)
a2=a
print(a2)
```

```
{'Marie': 62111111, 'Moussa': 62111122, 'Chérine': 62111111, 'Franck':
611223599}
{'Marie': 62111111, 'Moussa': 62111122, 'Chérine': 62111111, 'Franck':
611223599}
```

[24]:
```
# On va modifier le dictionnaire a2 copié, on remarque qu'alors le dictionaire␣
↪a aussi est modifié.
a2['Lina']=611111111
print('a= ',a)
print('a2= ',a2)
```

```
a=  {'Marie': 62111111, 'Moussa': 62111122, 'Chérine': 62111111, 'Franck':
611223599, 'Lina': 611111111}
a2=  {'Marie': 62111111, 'Moussa': 62111122, 'Chérine': 62111111, 'Franck':
611223599, 'Lina': 611111111}
```

On remarque que les deux dictionnaires ont été modifiés. Cela montre qu'il n'y a en fait qu'un seul objet et que dico et dico_2 sont en fait des références vers le même objet. Ces instructions ne créent pas une copie du dictionnaire dico. Elles se contentent de créer ce que l'on appelle un alias, c'est-à-dire un autre nom pour désigner le même objet.

Pour obtenir une vraie copie d'un dictionnaire, il faut utiliser, comme avec les listes, la méthode *copy()* :

[25]:
```
a3=a.copy()
a3['Pablo']=622222222
print('a= ',a)
print('a3= ',a3)
```

```
a=  {'Marie': 62111111, 'Moussa': 62111122, 'Chérine': 62111111, 'Franck':
611223599, 'Lina': 611111111}
a3=  {'Marie': 62111111, 'Moussa': 62111122, 'Chérine': 62111111, 'Franck':
611223599, 'Lina': 611111111, 'Pablo': 622222222}
```

[26]:
```
## Exercice 1 : Question 5
# Copier votre dictionnaire dico dans un autre dictinnaire dico2 et modifier␣
↪des valeurs.
# Vérifier alors les remarques du cours.
```

1.10 Création d'un dictionnaire par compréhension

On peut aussi constuire un dictionnaire par compréhension, comme avec les listes.

[27]:
```
comprehension={x*x: x for x in range(10)}
print(comprehension)
```

```
{0: 0, 1: 1, 4: 2, 9: 3, 16: 4, 25: 5, 36: 6, 49: 7, 64: 8, 81: 9}
```

1.11 Concaténer deux dictionnaires : update()

Il est possible de concaténer deux dictionnaires avec la méthode **update()**:

[28]:
```
d = {'apples': 1, 'oranges': 3, 'pears': 2}
ud = {'pears': 4, 'grapes': 5, 'lemons': 6}
d.update(ud)
d
```

[28]:
```
{'apples': 1, 'oranges': 3, 'pears': 4, 'grapes': 5, 'lemons': 6}
```

2 Résumé des notions abordées

- Un dictionnaire est un objet conteneur associant des clés à des valeurs.
- Pour créer un dictionnaire, on utilise la syntaxe :
  - dictionnaire = { cle1:valeur1, cle2:valeur2, cleN:valeurN }
- Pour récupérer une valeur dans un dictionnaire python.
  - la méthode get : nom_dico.get('clé').
  - la méthode directe : nom_dico['clé']
- On peut ajouter ou remplacer un élément dans un dictionnaire:
  - dictionnaire[cle] = valeur
- On peut supprimer une clé (et sa valeur correspondante) d'un dictionnaire en utilisant:
  - le mot-clé del
  - la méthode pop()
- On peut parcourir un dictionnaire grâce aux méthodes:
  - keys() : parcourt les clés
  - values() : parcourt les valeurs
  - items() : parcourt les couples clé-valeur
- On peut concaténer deux dictionnaires grâce à la méthode update():
  - dico1.update(dico2)

[29]:
```
# Pour accéder à toutes les fonctions et méthodes liées à la classe␣
↪dictionnaire (la notion de classe sera vue plus tard)
# il suffit de taper la commande dir(nom_dico) :

dir(a)
```

[29]:
```
['__class__',
 '__contains__',
 '__delattr__',
 '__delitem__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__setattr__',
 '__setitem__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'clear',
 'copy',
 'fromkeys',
 'get',
 'items',
 'keys',
 'pop',
 'popitem',
 'setdefault',
 'update',
 'values']
```

Méthodes	Actions
clear()	Supprime tous les éléments du dictionnaire
copy()	Renvoie une copie du dictionnaire
fromkeys()	Renvoie un dictionnaire avec les clés et valeurs spécifiées
get()	Renvoie la valeur de la clé spécifiée

Méthodes	Actions
items()	Retourne un objet qui affiche une liste des paires de tuples (clé, valeur) du dictionnaire
keys()	Renvoie une liste contenant les clés du dictionnaire
pop()	Removes the element with the specified key
popitem()	Supprime l'élément de la clé spécifiée

Méthodes	Actions
setdefault()	Renvoie la valeur de la clé spécifiée. Si la clé n'existe pas : insère la clé, avec la valeur spécifiée.
update()	Mise à jour du dictionnaire avec les paires clé-valeur spécifiées
values()	Renvoie une liste de toutes les valeurs du dictionnaire

3 TD - Les dictionnaires sous Python

3.1 Exercice 2

Voici un dictionnaire qui permet de coder les 26 lettres de l'alphabet, l'espace et l'apostrophe '.

[30]:
```
codage={'a':111,'b':121,'c':211,'d':311,'e':151,'f':411,'g':113,'h':11,'i':
↪21,'j':71,'k':19,'l':81,'m':100,'n':9,'o':1,'p':8,'q':1234,'r':1119,'s':
↪4111,'t':5111,'u':555,'v':321,'w':0,'x':20,'y':30,'z':40,' ':951,"'":753}
```

A l'aide de ce dictionnaire, on veut coder le message suivant : 'bonjour' 1. Entrainez-vous sur quelques lettres. 2. Ecrire une fonction f qui va renvoyer le mot codé sous forme d'une liste d'entiers.

Par exemple on devra avoir :

```
python >>>f('bonjour') [121, 1, 9, 71, 1, 555, 1119]
```

[31]:
```
# Exercice 2 : suestion 2
# ------------------------
def f(mot):
    ''' IN  : mot, un str
        OUT : le codage de mot avec le dictionnaire dans une liste'''
```

On peut facilement construire un dictionnaire permettant de décoder un message en passant les valeurs du premier en clé de l'autre car toutes les valeurs sont différentes. On utilise la métgode de construcion par compréhension vue dans le cours.

[32]:
```
decodage = {codage[clé] : clé for clé in codage  }
print(decodage)
```

```
{111: 'a', 121: 'b', 211: 'c', 311: 'd', 151: 'e', 411: 'f', 113: 'g', 11: 'h',
21: 'i', 71: 'j', 19: 'k', 81: 'l', 100: 'm', 9: 'n', 1: 'o', 8: 'p', 1234: 'q',
1119: 'r', 4111: 's', 5111: 't', 555: 'u', 321: 'v', 0: 'w', 20: 'x', 30: 'y',
40: 'z', 951: ' ', 753: "'"}
```

3. Ecrire une fonction g de décodage qui renvoie le mot codé. Par exemple on devra avoir :

```
>>>g([121, 1, 9, 71, 1, 555, 1119])
'bonjour'
```

[33]:
```
# Exercice 2 : suestion 3
# ------------------------
def g(liste):
    ''' IN  : liste, une liste d'entiers correspondants aux code
        OUT : le mot décodé (un str)'''
```

3.2 Exercice 3

Voici un extrait de Wikipédia concernant le *jeu de cartes*.

Le jeu de cartes français ou jeu de 52 cartes est un jeu de cartes organisées en deux couleurs: noir et rouge et en quatre enseignes françaises : pique, cœur, carreau, trèfle.

On considère que la valeur d'une carte correspond à 2 si c'est un deux, 3 si c'est un trois, ..., 10 pour un dix, 11 pour un valet, 12 pour une dame, 13 pour un roi et 14 pour un as.

1. Créer un dictionnaire nommé *valeur_carte* dont les clés sont les noms des cartes (2,3,...,10,valet,Dame,Roi,As) et les valeurs les valeurs de ces clés. Par exemple votre dictionnaire devra renvoyer :

```
>>> valeur_carte['valet']
11
```

[34]:
```
# Exercice 3 : suestion 1
# ------------------------
```

2. Créer un dictionnaire nommé *jeu_carte* dont :
   - les clés sont les 52 tuples (nom,enseigne) des cartes d'un jeu de 52 cartes : ('2','pique') , ... ,('as','pique'), ('2','coeur'), ... ,('as','coeur')...
   - et les valeurs les valeurs de ces clés. Par exemple votre dictionnaire devra renvoyer :
     ```
     >>> jeu_carte[('dame','pique')]
     12
     ```

[35]:
```
# Exercice 3 : suestion 2
# ------------------------
```

3. Si on veut sélectionner aléatoirement des éléments (créer un échantillon) sans remise d'une liste ou d'un dictionnaire on peut utiliser la fonction random.sample(). Par exemple pour obtenir un échantillon de taille 2 du dictionnaire test :

[36]:
```
test={('2','pique'):2 , ('as','pique'):14, ('3','coeur'):3,  ('as','coeur'):14}
import random
echant=random.sample(test.keys(), 2)
print(echant)
```

```
[('3', 'coeur'), ('as', 'pique')]
```

[37]:
```
# Exercice 3 : suestion 3
# ------------------------
```

4. Créer une fonction bataille(jeu_cartes) qui va choisir au hasard deux cartes distinctes du jeu, les afficher, comparer leur valeurs et donner la carte gagante. Par exemple on pourra avoir :

   ```
   >>> bataille(jeu_cartes)
   ('dame','coeur') de valeur 12 bat ('9','pique') de valeur 9
   ```

[38]:
```
# Exercice 3 : suestion 4
# ------------------------
def bataille(jeu_cartes):
    ''' IN  : un dictionnaire jeu de cartes
        OUT : carte1 choisie, carte 2 choisie , les valeurs et la carte la plus␣
 ↪forte
        Action :  va choisir au hasard deux cartes distinctes du jeu,
        les afficher, comparer leur valeurs et afficher la carte gagante.'''
```