Chapitre 7 :
Types séquentiels 2 :
chaînes de caractères et dictionnaires

I Définitions et types

1.1 Les chaînes de caractères

Les chaînes de caractères sont délimitées par des guillemets "..." ou des apostrophes '...'. Elles contiennent des objets de type alphanumériques.
Voici quelques exemples de chaînes de caractères :

```
>>> c1='bonjour'
>>> c2="quel temps fait-il aujourd'hui ?"
>>> c3='2+2=4'
```

Le type des chaînes de caractères est : str.

1.2 Les dictionnaires

Un dictionnaire est une collection non ordonnée d'objets, délimités par des accolades {...}. Chaque valeur d'un dictionnaire correspond à une clé. Par exemple :

```
>>> d= {"voiture" : 4, "vélo" : 2}
```

d est un dictionnaire composé des clés "voiture" et "vélo" et des valeurs 4 et 2.

Le type des dictionnaires est : dict.

II Opérations basiques

2.1 Longueur

Comme pour les listes et les tuples, la commande len(...) donne la longueur d'une chaîne de caractères ou d'un dictionnaire :

```
>>> c='vive Python'
>>> len(c)
11
>>> t=(1,[5,6],'vive Python !')
>>> len(t)
3
>>> d= {"voiture" : 4, "vélo" : 2}
>>> len(d)
2
```

2.2 Extraction

2.2.1 Chaînes de caractères

Ce cas est identique à celui des listes et des tuples : soit c une chaîne de caractères.
La commande c[k] permet d'accéder à l'élément situé en $k^{\text{ième}}$ position dans la chaîne de caractères.

La numérotation des éléments des chaînes de caractères commence à 0.

On peut utiliser des valeurs négatives de k pour compter à partir de la fin, on commence alors la numérotation à -1.

La commande c[i :j] permet de construire la chaîne constituée des éléments de l d'indice variant entre i et j exclus.

La commande c[i :j :k] permet de construire la chaîne constituée des éléments de l d'indice variant entre i et j exclus avec un pas k.

```
>>> l=[1,0.2,'python',[2,4],8,7]
>>> c='vive Python'
>>> l[1]
0.2
>>> c[5]
'P'
>>> c[0 :len(c) :2]
'vv yhn'
```

2.2.2 Dictionnaires

L'accès aux informations sera différent pour les dictionnaires car il s'agit d'un type non ordonné.
Pour accéder aux valeurs d'un dictionnaire, on doit utiliser les clés.
Soit d un dictionnaire, soit c une clé de d, la commande d[c] permet d'accéder à la valeur associée à la clé c.

```
>>> d= {"voiture" : 4, "vélo" : 2}
>>> d["voiture"]
4
```

2.3 Test d'appartenance

L'opérateur in permet de tester l'appartenance à une chaîne de caractères ou à un dictionnaire. Son résultat est un booléen. Dans le cas d'un dictionnaire, il s'agit d'un test d'appartenance de clés et pas de valeurs.

```
>>> c='vive Python'
>>> d= {"voiture" : 4, "vélo" : 2}
>>> v in c
Traceback (most recent call last) :
File "<console>", line 1, in <module>
NameError : name 'v' is not defined
>>> 'v' in c
True
>>> "voiture" in d
True
>>> 4 in d
False
```

La commande in est identique pour tous les types séquentiels mais son coût est moins important pour les dictionnaires. En effet en raison de l'implémentation des dictionnaires, l'accès aux clés se fait à un coût constant, alors que pour les autres types séquentiels, il dépendra de la longueur de la séquence.

```
import time

def TestIn(n) :
    l=[k for k in range(n)]
    d={k :k for k in range(n)}
    t=time.time()
    n in l
    print('pour une liste de longueur n :',time.time()-t)
    t=time.time()
    n in d
    print('pour un dictionnaire de longueur n :',time.time()-t)

>>> TestIn(10**8)
pour une liste de longueur n : 16.755928993225098
pour un dictionnaire de longueur n : 0.0
```

2.4 Concaténation

Comme pour les listes et les tuples, il est possible de concaténer les chaînes de caractères avec l'opérateur +. La concaténation ne fonctionne pas avec les dictionnaires.

```
>>> c1="Bonjour"
>>> c2=" "
>>> c3="bonsoir"
>>> c1+c2+c3
"Bonjour bonsoir"
```

2.5 Conversion de type

On peut transformer une séquence en chaîne de caractères avec la commande `str`. Cela peut être utile pour trouver les chiffres de l'écriture décimale d'un nombre.

```
>>> n=148
>>> c=str(n)
>>> c
'148'
>>> c[1]
'4'
```

2.6 Itérables

2.6.1 Chaînes de caractères

Les chaînes de caractères sont itérables.

```
1 def fonction1(c) :
2     for x in c :
3         print(x)
```

`fonction1(c)` affiche chaque caractère de la chaîne `c`.

2.6.2 Dictionnaires

Un dictionnaire n'est pas itérable, on doit utiliser les méthodes `keys` ou `items` pour travailler sur un objet itérable.
On suppose ici qu'on a défini un dictionnaire `d`.

- `d.keys()` renvoie les clés du dictionnaire `d`,
- `d.items()` renvoie une version itérable du dictionnaire, c'est-à-dire une version sur laquelle on pourra faire des boucles.

```
>>> d = {"voiture" : 4, "vélo" : 2}
>>> d.keys()
dict_keys(['voiture', 'vélo'])
>>> d.items()
dict_items([('voiture', 4), ('vélo', 2)])
```

```
1 def fonction2(d) :
2     for i in d.items() :
3         print(i)
```

```
1 def fonction3(d) :
2     for cle in d.keys() :
3         print(cle)
```

```
1 def fonction4(d) :
2     for cle, valeur in d.items() :
3         print("l'élément de clé", cle, "vaut", valeur)
```

On a les résultats suivants :

```
>>> d={ "voiture" : 4, "vélo" : 2 }
>>> fonction2(d)
('voiture', 4)
('vélo', 2)
>>> fonction3(d)
voiture
vélo
>>> fonction4(d)
l'élément de clé voiture vaut 4
l'élément de clé vélo vaut 2
```

III Données mutables et données immuables

3.0.1 Définition

On rappelle que les séquences Python sont mutables ou immuables :

- immuables : leur valeur ne peut pas changer (à moins de les redéfinir entièrement)
- mutables : leur valeur peut changer.

On a déjà vu que les listes sont mutables et les tuples sont immuables.

Les listes et les dictionnaires sont mutables,
les tuples et les chaines de caractères sont immuables.

```
>>> c='abc';d={"voiture" : 4, "vélo" : 2}
>>> c[1]='d'
Traceback (most recent call last) :
File "<console>", line 1, in <module>
TypeError : 'str' object does not support item assignment
>>> d["voiture"]=6
>>> d
{'voiture' : 6, 'vélo' : 2}
>>> d["tricycle"]=3
>>> d
{'voiture' : 6, 'vélo' : 2, 'tricycle' : 3}
```

3.0.2 Copie

Les dictionnaires étant mutables, on rencontre le même problème de copie que celui vu pour les listes.
Afin de remédier à ce problème, on utilisera encore la méthode `copy` qui faut partie du module `copy`.

```
>>> import copy
>>> d1= {"voiture" : 4, "vélo" : 2}
>>> d2=copy.copy(d1)
>>> d1["voiture"]=6
>>> d1
{"voiture" : 6, "vélo" : 2}
>>> d2
{"voiture" : 4, "vélo" : 2}
```

IV Un algorithme classique : recherche d'un mot dans une chaîne de caractères

On considère deux chaînes de caractères `chaine` et `mot`. La fonction suivante renvoie `True` si `mot` est dans `chaine` et `False` sinon.

```
def recherche(chaine,mot) :
    assert len(chaine)>=len(mot)
    n=len(chaine)
    p=len(mot)
    for i in range(n-p) :
        v=True
        for j in range(p) :
            if mot[j] !=chaine[i+j] :
                v=False
        if v==True :
            return True
    return False
```

Cette fonction utilise la notion de code mort pour arrêter la boucle for avant la fin de son exécution.

V Manipulation de fichiers

On va travailler sur un fichier contenant des données textuelles. Le fichier sera au format TXT. On utilisera les mêmes commandes que pour un fichier contenant des données numériques :

- open : ouverture du fichier. On précise un argument qui peut être :
  - 'r' : pour une ouverture en lecture (read).
  - 'w' : pour une ouverture en écriture (write). Dans ce cas, à chaque ouverture, le fichier est écrasé. Si le fichier n'existe pas, il sera automatiquement créé.
  - 'a' : pour une ouverture en mode ajout à la fin du fichier (append). Si le fichier n'existe pas, il sera automatiquement créé.

  On peut également préciser l'encodage afin de ne pas avoir de problème avec les caractères spéciaux.
- read : lecture du fichier. Les informations sont copiées dans une variable.
- close : fermeture du fichier. Cette commande permet de libérer les ressources dont on ne se sert plus.
- readline : lecture d'une ligne du fichier.
- readlines : création d'une liste contenant toutes les lignes du fichier.
- split : extraction des différents champs d'une ligne via des séparateurs.
- write : écriture à la fin d'un fichier.

Pour illustrer ses commandes, on utilisera le fichier Lamartine.txt suivant :

```
Que me font ces vallons, ces palais, ces chaumières,
Vains objets dont pour moi le charme est envolé?
Fleuves, rochers, forêts, solitudes si chères,
Un seul être vous manque, et tout est dépeuplé!
```

```
>>> fic=open("Lamartine.txt",'r',encoding='utf-8')
>>> f=fic.read()
>>> f
'Que me font ces vallons, ces palais, ces chaumières,\n Vains objets dont pour moi le
charme est envolé?\n Fleuves, rochers, forêts, solitudes si chères,\n Un seul être vous
 manque, et tout est dépeuplé! '
>>> fbis=fic.read()
>>> fbis
''#La commande read vide le contenu de fic et le stocke dans une variable.
```

```
>>> fic=open("Lamartine.txt",'r',encoding='utf-8')
>>> fic.readline()
'Que me font ces vallons, ces palais, ces chaumières,\n'
>>> fic.readline()
'Vains objets dont pour moi le charme est envolé?\n'
>>> fic.readlines()
['Fleuves, rochers, forêts, solitudes si chères,\n', 'Un seul être vous manque, et
 tout est dépeuplé! ']
```

```
>>> fic=open("Lamartine.txt",'r',encoding='utf-8')
>>> f=fic.read()
>>> t=f.split('\n')
>>> print(t)
['Que me font ces vallons, ces palais, ces chaumières,', 'Vains objets dont pour moi le
 charme est envolé ?', 'Fleuves, rochers, forêts, solitudes si chères,', 'Un seul être vous
 manque, et tout est dépeuplé ! ']
>>> tbis=f.split(',')
['Que me font ces vallons', ' ces palais', ' ces chaumières', '\nVains objets dont pour
 moi le charme est envolé ?\nFleuves', ' rochers', ' forêts', ' solitudes si chères', '\nUn
 seul être vous manque', ' et tout est dépeuplé ! ']
```

```
>>> fic=open("Lamartine.txt",'r',encoding='utf-8')
>>> fic.write('Lamartine')
Traceback (most recent call last) :
File "<console>", line 1, in <module>
io.UnsupportedOperation : not writable
>>> fic=open("Lamartine.txt",'w',encoding='utf-8')
>>> fic.write('Lamartine')
>>> f=fic.read()
Traceback (most recent call last) :
File "<console>", line 1, in <module>
io.UnsupportedOperation : not readable
>>> fic=open("Lamartine.txt",'r',encoding='utf-8')
>>> f=fic.read()
>>> f
'Lamartine'
```

```
>>> fic=open("Lamartine.txt",'a',encoding='utf-8')
>>> fic.write('Lamartine')
>>> fic=open("Lamartine.txt",'r',encoding='utf-8')
>>> f=fic.read()
>>> f
'Que me font ces vallons, ces palais, ces chaumières,\nVains objets dont pour moi le
charme est envolé ?\nFleuves, rochers, forêts, solitudes si chères,\nUn seul être vous
manque, et tout est dépeuplé !Lamartine'
```