Matrices

Liens

- Guides bonnes conduites

- GTK Python tutorial
- Bases de données sqlite / Python
- GTK Python
- Prog éven. Tk
- Python admin sys linux
- Processur et sous processus

Créer une matrice

```
ragnarok@Fenrir:~$ python3
Python 3.7.6 (default, Dec 19 2019, 09:25:23)
[GCC 9.2.1 20191130] on linux
>>> score =
[['joueur1','joueur2','joueur3','joueur4'],[150.0,-50.0,-50.0,-50.0],[-20.0,
-20,0,-20,0,60]]
>>> print(score)
[['joueur1', 'joueur2', 'joueur3', 'joueur4'], [150.0, -50.0, -50.0, -50.0],
[-20.0, -20, 0, -20, 0, 60]]
>>> type(score)
<class 'list'>
```

La matrice a été créée. On voit bien qu'il s'agit en fait d'une liste de listes.

Accéder à un élément de la matrice

```
>>> score[2][2]
0
>>> score[2][3]
-20
>>> score[0][0]
'joueur1'
```

Ces listes peuvent contenir tous types d'éléments

```
>>> type(score[0][0])
<class 'str'>
>>> type(score[1][1])
<class 'float'>
```

```
>>> score[1:][1:]
[[-20.0, -20, 0, -20, 0, 60]]
```

Parcourir la matrice

Parcourir les lignes

```
>>> for lignes in score :
...     print(lignes)
...
['joueur1', 'joueur2', 'joueur3', 'joueur4']
[150.0, -50.0, -50.0, -50.0]
[-20.0, -20, 0, -20, 0, 60]
```

Parcourir la matrice :

```
>>> for lignes in score :
...     for col in lignes:
...             print(col)
...
joueur1
joueur2
joueur3
joueur4
150.0
-50.0
-50.0
-50.0
-20.0
-20
0
-20
0
60
```

Parcourir la matrice avec index

```
>>> for i_line, line in enumerate(score):
...     print(i_line, ' : ', line)
...
0  :  ['joueur1', 'joueur2', 'joueur3', 'joueur4']
1  :  [150.0, -50.0, -50.0, -50.0]
2  :  [-20.0, -20, 0, -20, 0, 60]
```

```
>>> for i_line, line in enumerate(score):
```

```
...     for i_col, col in enumerate(line):
...             print (i_col, ' : ', col)
...
0  :  joueur1
1  :  joueur2
2  :  joueur3
3  :  joueur4
0  :  150.0
1  :  -50.0
2  :  -50.0
3  :  -50.0
0  :  -20.0
1  :  -20
2  :  0
3  :  -20
4  :  0
5  :  60
```

Dictionnaires

Créer un ensemble de joueur.

```
[ragnarok@asgard-freeBSD:~] $ python3.6
Python 3.6.9 (default, Jul  9 2019, 01:18:18)
[GCC 4.2.1 Compatible FreeBSD Clang 6.0.1 (tags/RELEASE_601/final 335540)]
on freebsd12
Type "help", "copyright", "credits" or "license" for more information.
>>> joueur = dict()
>>> for i in range(0,3):
...      joueur["nom{}".format(i+1)] = 100+i*100
```

...

La liste de joueur est crée (ici avec des scores définis par des multiples de 100

```
>>> joueur
{'nom1': 100, 'nom2': 200, 'nom3': 300}
>>>
```

Parcourir les clés

```
 >>> for j in joueur:
...     print(j)
...
nom1
nom2
```

```
nom3
```

ou

```
>>> for j in joueur.keys():
...     print(j)
...
nom1
nom2
nom3
```

Parcourir les valeurs

```
>>> for values in joueur.values():
...     print(values)
...
100
200
300
```

Parcourir les clefs et les valeurs

```
>>> for cle,valeur in joueur.items():
...     print("La cle est: {} et la valeur est: {}.".format(cle,valeur))
...
La cle est: nom1 et la valeur est: 100.
La cle est: nom2 et la valeur est: 200.
La cle est: nom3 et la valeur est: 300.
```

Afficher une valeur

```
>>> joueur['nom1']
100
```

Remplacer une valeur

```
>>> joueur['nom1']
100

>>> joueur['nom1'] = 5000
>>> joueur['nom1']
5000
```

Au final

On peut parcourir un dictionnaire grâce aux méthodes **keys** (parcourt les clés),**values** (parcourt les valeurs) ou **items** (parcourt les couples clé-valeur).

From:
https://cbiot.fr/dokuwiki/ - **Cyrille BIOT**

Permanent link:
https://cbiot.fr/dokuwiki/python:aide-memoire?rev=1579967209

Last update: **2020/01/25 15:46**