Python pour les data scientistes

Chapitre 3 : Les chaînes de caractères

Une chaîne de caractères est un objet de la classe (ou de type) str. Une chaîne de caractères peut être définie de plusieurs façons :

```
>>> "je suis une chaine"
'je suis une chaine'
>>> 'je suis une chaine'
'je suis une chaine'
>>> 'pour prendre l\'apostrophe'
'pour prendre l' apostrophe'
>>> "pour prendre l'apostrophe"
"pour prendre l'apostrophe"
>>> " " "ecrire sur plusieurs lignes" " "
'ecrire\nsur\nplusieurs\nlignes'
```

— concaténation

On peut mettre plusieurs chaînes de caractères bout à bout avec l'opérateur binaire de concaténation, noté +.

```
>>> s = 'i vaut'
>>> i = 1
>>> print( s+i )
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
TypeError: cannot concatenate 'str' and 'int' objects

>>> print( s + ' ' + str(i))
i vaut 1

>>> print('*-' * 5)
*-*-*-*-*-
```

Accès aux caractères

Les caractères qui composent une chaîne sont numérotés à partir de zéro. On peut y accéder individuellement en faisant suivre le nom de la chaîne d'un entier encadré par une paire de crochets :

```
>>> "bonjour"[3]; "bonjour"[-1]
'j'
'r'

>>> "bonjour"[2:]; "bonjour"[:3]; "bonjour"[3:5]
'njour'
'bon'
'jo'

>>> "bonjour"[-1::-1];
'ruojnob'
```

— méthodes propres
— len(s) : renvoie la taille d'une chaîne,
— s.find : recherche une sous-chaîne dans la chaîne,
— s.rstrip : enlève les espaces de fin,
— s.replace : remplace une chaîne par une autre,

Les listes

Une liste consiste en une succession ordonnée d'objets, qui ne doivent pas nécessairement être du même type. Les termes d'une liste sont numérotés à partir de 0 (comme pour les chaînes de caractères).

— initialisation

```
>>> []; list();
[]
[]

>>> [1,2,3,4,5]; ['point','triangle','quad'];
[1, 2, 3, 4, 5]
['point', 'triangle', 'quad']

>>> [1,4,'mesh',4,'triangle',['point',6]];
[1, 4, 'mesh', 4, 'triangle', ['point', 6]]

>>> range(10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

>>> range(2,10,2)
[2, 4, 6, 8]
```

Modification

Contrairement aux chaînes de caractères, on peut modifier les éléments d'une liste :

```
>>> l=[1,2,3,4,5]
>>> l[2:]=[2,2,2]
>>> l
[1, 2, 2, 2, 2]
```

Concaténation

```
>>> [0]*7
[0, 0, 0, 0, 0, 0, 0]

>>> L1, L2 = [1,2,3], [4,5]

>>> L1
[1, 2, 3]

>>> L2
[4, 5]

>>> L1+L2
[1, 2, 3, 4, 5]
```

Méthodes propres
— len(L) : renvoie la taille de la liste L,
— L.sort : trie la liste L,
— L.append : ajoute un élément à la fin de la liste L,
— L.reverse : inverse la liste L,
— L.index : recherche un élément dans la liste L,
— L.remove : retire un élément de la liste L,
— L.pop : retire le dernier élément de la liste L,

Copie d'un objet

```
>>> L = ['Dans','python','tout','est','objet']

>>> T = L

>>> T[4] = 'bon'

>>> T
['Dans', 'python', 'tout', 'est', 'bon']

>>> L
['Dans', 'python', 'tout', 'est', 'bon']

>>> L=T[:]

>>> L[4]='objet'

>>> T;L
['Dans', 'python', 'tout', 'est', 'bon']
['Dans', 'python', 'tout', 'est', 'objet']
```

Les modules

Un module est un fichier comprenant un ensemble de définitions et d'instructions compréhensibles par Python (py). Il permet d'étendre les fonctionnalités du langage.

En Python, on peut distinguer trois grandes catégories de module en les classant selon leur éditeur :

- Les modules standards qui ne font pas partie du langage en soi mais sont intégrés automatiquement par Python ;
- Les modules développés par des développeurs externes qu'on va pouvoir utiliser ;
- Les modules qu'on va développer nous-mêmes.

Il existe plusieurs façons d'importer un module :
— import module
— import module as m
— from module import éléments appelés
— from module import * (importe tous les noms sauf les variables et les fonctions privées)

bonjour.py

```
1   nom = "Pierre"
2
3   def disBonjour():
4       print(nom, " vous dit bonjour")
```

```
>>> import bonjour
>>> bonjour.disBonjour()
Pierre  vous dit bonjour
>>> bonjour.nom
'Pierre'
```

```
>>> import bonjour as b
>>> b.disBonjour()
Pierre  vous dit bonjour
```

Le module math

Le module math nous fournit un accès à de nombreuses fonctions permettant de réaliser des opérations mathématiques comme le calcul d'un sinus, cosinus, d'une tangente, d'un logarithme ou d'une exponentielle.

Les fonctions les plus couramment utilisées sont les suivantes :

- Les fonctions ceil() et floor() renvoient l'arrondi du nombre passé en argument en arrondissant respectivement à l'entier supérieur et inférieur ;
- La fonction fabs() renvoie la valeur absolu d'un nombre passé en argument ;
- La fonction isnan() renvoie True si le nombre passé en argument est NaN = Not a Number (pas un nombre en français) ou False sinon ;
- La fonction exp() permet de calculer des exponentielles ;
- La fonction log() permet de calculer des logarithmes ;
- La fonction sqrt() permet de calculer la racine carrée d'un nombre ;
- Les fonctions cos(), sin() et tan() permettent de calculer des cosinus, sinus et tangentes et renvoient des valeurs en radians.

Attention : les fonctions de ce module ne peuvent pas être utilisées avec des nombres complexes. Pour cela, il faudra plutôt utiliser les fonctions du module cmath.

Le module math définit également des constantes mathématiques utiles comme pi accessible via math.pi.

```
>>> import math
>>> math.ceil(3.1)
4
>>> math.ceil(3.9)
4
>>> math.floor(3.1)
3
>>> math.floor(3.9)
3
>>> math.fabs(-4)
4.0
>>> math.pi
3.141592653589793
```

Le module random

Le module random nous fournit des outils pour générer des nombres pseudo-aléatoires de différentes façons.

La fonction random() est le plus utilisée du module. Elle génère un nombre à virgule flottante aléatoire de façon uniforme dans la plage semi-ouverte [0.0, 1.0).

La fonction uniform() va elle générer un nombre à virgule flottante aléatoire compris dans un intervalle. On va lui passer deux tombes en argument : le premier nombre représente la borne basse de l'intervalle tandis que le second représente la borne supérieure. Notez que cette fonction se base sur random().

```
>>> import random
>>> random.random()
0.4493622050556486
>>>
>>> random.random()
0.5548624406475254
>>>
>>> random.uniform(10,100)
71.91953801242744
```