Variables

Les types numériques

Les entiers (simple ou long)

Les entiers sont des nombres qui s'étendent sur une machine 32 bits entre les valeurs -2e31+1 et 2e31-1, soit entre -2147483647 et 2147483647.

Bases

- La forme décimale commence par un chiffre de 1 à 9 suivi de chiffre de 0 à 9.
- La base octale commence par "0" suivi de chiffres de 0 à 7.
- La base hexadécimale commence par "0x" suivi de chiffres 0 à 9 et de lettres de A à F.

Les entiers longs

Les entiers longs sont eux limités uniquement par la capacité mémoire de la machine.
Ils sont représentés de la même manière que les entiers mais sont suffixés de la lettre L.
Depuis la version de 2.4 un nombre entier est automatiquement transtypé s'il dépasse la plage autorisée.

Les flottants

- Les flottants permettent de représenter des nombres réels.
- La partie entière est séparée de la partie fractionnelle par un "."

Les nombres complexes

Exemple

```
>>> a = 2 + 1j
>>> b = 1 + 2j
>>> a+b
(3+3j)
>>> a.imag
1.0
>>> a.real
2.0
>>> a.conjugate()
```

```
(2-1j)
```

Les séquences

Une séquence est une collection ordonnée d'éléments indexés par des nombres positifs.
Les index varient de 0 à n-1 pour une séquence de n éléments.

élément	1	2	...	i	...	n-1	n
index	1	2		i-1		n-2	n-1
index	-n	-n+1				-2	-1

Les séquences peuvent être coupées en tranches qui forment alors des sous séquences. sequence[i,j] forme la sequence de l'élement i inclus jusqu'à l'élément j exclu. Un troisième paramètre permet de faire varier le pas.

Exemples

```
sequence[1,3] ne contient que le 2° et 3° élément.
sequence[0:-1] contient tous les éléments sauf le dernier.
sequence[0:-1:2] prend un élément sur 2.
sequence[::-1] inverse la séquence.
```

Primitives

len	retourne le nombre d'élements de la séquence
min	renvoie l'élement à valeur minimale de la séquence
max	renvoie l'élement à valeur maximale de la séquence
sum	renvoie la somme des élements lorsque ceux ci sont additionnables

Il existe 2 grandes familles de séquences, les séquences immuables (string, unicode et tuple) et les séquences modifiables (liste).

Les strings (immuables)

- Les string sont des valeurs alphanumériques, elles sont entourées de guillemets simples ou doubles ou de 3 guillemets simples ou doubles.
- Une string est une séquence ordonnée d'éléments.
- Une string n'est plus modifiable une fois instanciée.
- Les strings sont des séquences de caractères, chaque caractère est codé sur 8 bits ce qui correspond à des valeurs de 0 à 255. Les valeurs de 0 à 127 sont les caractères de la tables ASCII ceux de 128 à 255 les caractères de la table ASCII étendue.
- Il n'existe pas en python de type caractère, un caractère est une string de longueur 1.
- En python 3, les strings sont portés par une nouvelle classe: les bytes.
- Les strings sont des séquences, les différents éléments sont donc accessibles comme décrit

plus haut.
- Les strings sont des séquences immuables, leurs valeurs ne sont donc pas modifiables.

Primitives

chr	renvoi le caractère dont le code ASCII est passé en paramètre
ord	renvoi le code ASCII du caractère passé en paramètre

Liste des méthodes

```
>>> mot = 'Hello'
>>> dir(mot)
['__add__', '__class__', '__contains__', '__delattr__', '__doc__', '__eq__',
'__format__', '__ge__', '__getattribute__', '__getitem__', '__getnewargs__',
'__getslice__', '__gt__', '__hash__', '__init__', '__le__', '__len__',
'__lt__', '__mod__', '__mul__', '__ne__', '__new__', '__reduce__',
'__reduce_ex__', '__repr__', '__rmod__', '__rmul__','__setattr__',
 '__sizeof__', '__str__', '__subclasshook__', '_formatter_field_name_split',
 'capitalize', 'center', 'count', 'decode', 'encode', 'endswith',
'expandtabs',
 'find', 'format', 'index', 'isalnum', 'isalpha', 'isdigit', 'islower',
'isspace',
 'istitle','isupper', 'join', 'ljust', 'lower', 'lstrip', 'partition',
 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip',
 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title',
 'translate', 'upper', 'zfill']
```

Les unicodes (immuables)

Le type unicode fonctionne de la même façon mais sur une plage de 32 bits. Ils sont standards en python 3.x. (format encodage utilisé par défaut).

```
>>> unichr(16000)
u'\u3e80'
>>> print(unichr(16000))
㺀
>>> ord(u'€')
8364
```

Conversion entre string et unicode

La conversion est possible grâce aux méthodes encode et décode à l'aide d'un codec qui permet d’établir la correspondance entre les deux.

```
>>> encode = u'é'.encode('ISO-8859-15')
>>> print(encode)
é
```

```
>>> decode='é'.decode('ISO-8859-15')
>>> print(decode)
é
```

Formatage des chaines de caractères

```
>>> firstname = 'John'
>>> lastname = 'Cleese'
>>> print("Hello {}".format(firstname))
Hello John
>>> print("Hello {} {}".format(firstname, lastname))
Hello John Cleese
```

Il existe certaines expression pour modifier la valeur de l'argument. Exemple, pour les float, la syntaxe est .[P]f où P est un paramètre optionnel qui précise la précion utilisée.

```
>>> pi = 3.14141592654
>>> print ("pi vaut {:.2f}".format(pi))
>>> print ("pi vaut {0:.2f}".format(pi))
```

Les tuples (immuables)

- Les tuples sont des séquences qui contiennent des éléments qui peuvent être hétérogènes.
- Un tuple est encadré par des parenthèses et chaque élément est séparé par une virgule.
- Une fois créé il est impossible de modifier sa valeur.

Exemple

```
>>> actors = ('John Cleese', 'Terry Gilliam', 'Michael Palin')
>>> heterogene = ('John Cleese', 2, u'Terry Gilliam')
>>> len(actors)
3
>>> actors[1]
'Terry Gilliam'

>>> actors[1] = 'Terry Jones'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

Les listes (modifiable)

- Les listes sont des séquences d'élements qui peuvent être hétérogènes et dont la valeur peut être modifiée après sa création.
- Une liste est encadrée par des crochets et chaque élément est séparé par une virgule.

```
>>> liste_vide = []
>>> actors = ['John Cleese']
>>> len(actors)
1
```

Liste des méthodes

```
>>> dir(actors)
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__',
'__delslice__',
 '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__',
'__getitem__',
 '__getslice__', '__gt__', '__hash__', '__iadd__', '__imul__', '__init__',
 '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__',
 '__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__rmul__',
 '__setattr__', '__setitem__', '__setslice__', '__sizeof__',
'__str__','__subclasshook__',
 'append', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse',
'sort']
>>> actors.append('Terry Gilliam')
>>> actors
['John Cleese', 'Terry Gilliam']
>>> new_actors = ['Michael Palin', 'Terry Jones']
>>> actors.extend(new_actors)
>>> actors
['John Cleese', 'Terry Gilliam', 'Michael Palin', 'Terry Jones']
>>> actors.insert(2, 'Eric Idle')
>>> actors
['John Cleese', 'Terry Gilliam', 'Eric Idle', 'Michael Palin', 'Terry
Jones']
>>> actors.remove('John Cleese')
>>> actors
['Terry Gilliam', 'Eric Idle', 'Michael Palin', 'Terry Jones']
>>> actors.pop(2)
'Michael Palin'
>>> actors
['Terry Gilliam', 'Eric Idle', 'Terry Jones']
>>> actors.index('Terry Jones')
2
>>> actors.append(actors[2])
>>> actors
['Terry Gilliam', 'Eric Idle', 'Terry Jones', 'Terry Jones']
>>> actors.count('Terry Jones')
2
>>> actors.sort()
>>> actors
['Eric Idle', 'Terry Gilliam', 'Terry Jones', 'Terry Jones']
>>> actors.reverse()
>>> actors
```

```
['Terry Jones', 'Terry Jones', 'Terry Gilliam', 'Eric Idle']
```

Les ensembles (type set)

Un ensemble est encadré par des accolades et chaque élément est séparé par une virgule.

```
>>> ensemble = {} #dictionnaire
>>> ensemble = {'John Cleese','Terry Gilliam'} # type set
>>> len(ensemble)
2
>>> ensemble.update({'John Cleese'})
>>> len(ensemble)
2
```

Les dictionnaires

- Le mapping, appelé dictionnaire, est une collection d'éléments qui sont identifiés par des clefs uniques.
- Un dictionnaire n'est pas ordonné.
- Les clefs doivent être des objets immuables (chaines de caractère, entier, flottant...)
- Les élements peuvent être hétérogènes
- Un dictionnaire est encadrée par des accolades, chaque élement est séparé par une virgule, Chaque élement est préfixé par sa clef suivi de ":".

Exemples

```
>>> dico_vide = {}
>>> cleese = {'firstname':'John', 'lastname':'Cleese', 'year_of_birth':1939,
'movies':['Monty Python, la vie de Brian', 'Un poisson nommé wanda']}
>>> len(cleese)
4
>>> cleese['firstname']
'John'
```

```
>>> dict(sape=4139, guido=4127, jack=4098)
{'sape': 4139, 'jack': 4098, 'guido': 4127}
>>> dict([('sape', 4139), ('guido', 4127), ('jack', 4098)])
{'sape': 4139, 'jack': 4098, 'guido': 4127}
```

Liste des méthodes

```
>>> a = {}
```

```
>>> dir(a)
>>> ['__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']
```

None

Type à valeur unique qui représente une absence de valeur.

Les booléens

- Représentés par “True” ou “False”
- Tout les types de valeur peuvent être interprétés de manière booléénne

type	False	True
int	0	les autres valeurs
float	.0	les autres valeurs
list	[]	les listes non vides
tuple	()	les tuples non vides
dict	{}	les dicts non vides
type NONE	None	X

Synthèse des types

type	exemple	sequence	modifiable
int	1	NON	NON
float	1.0	NON	NON
complex	1+1j	NON	NON
string	“Hello”	OUI	NON
unicode	u’Hello”	OUI	NON
tuple	(“string”,1)	OUI	NON
list	[“string”,1]	OUI	OUI
dict	{“key”:”valeur”}	NON	OUI

Transtypage

primitive	exemple	description
str(el)	str(1) ⇒ “1”	retourne l’élement sous forme de string
int(el)	int(“1”) ⇒ 1	retourne l’élement sous forme d’entier
float(el)	float(“1”)⇒1.0	retourne l’élement sous forme de float
list(el)	list(“1”)⇒[“1”]	retourne l’élement sous forme de liste
tuple(el)	tuple([1])⇒(1,)	retourne l’élement sous forme de tuple

---

Les opérateurs

Opérateurs mathématiques

Addition	+
Soustraction	-
Multiplication	*
Division	/
Modulo (reste)	%
Puissance	**

Exemples

```
>>> 2+3
5
>>> 3-5
-2
>>> 2*7
14
>>> 'hello'*5
'hellohellohellohellohello'
>>> 'hello' + ' ' + 'world'
'hello world'
>>> 1/6
0
>>> 1.0/6
0.16666666666666666
>>> 10 % 3
1
>>> 2 * * 8
256
```

Opérateurs de comparaison

Inférieur	<
Supérieur	>
Inférieur ou égal	<=

Supérieur ou égal	>=
Egalité	==
Différent	!=
est	is
n'est pas	is not

- Une comparaison travaille sur deux objets et renvoie un résultat booléen.
- Les strings sont immuables, l'affectation mémoire des deux objets est identique.

Exemple

```
>>> a = 'Hello'
>>> b = 'Hello'
>>> id(a)
39756736
>>> id(b)
39756736
>>> a is b
True
```

Autres opérateurs

Négation

```
>>> a = 10
>>> -a
-10
```

Appartenance

S'applique à toutes les séquences.

```
>>> 'y' in 'Terry'
True
>>> 1 in [0,1,2,3,4]
True
```

Trucs & astuces

Interroger le type d'un objet

```
>>> type(variable)
```

Lister les méthodes d'un objet

```
>>> mot = 'Hello'
>>> dir(mot)
['__add__', '__class__', '__contains__', '__delattr__', '__doc__', '__eq__',
'__format__', '__ge__', '__getattribute__', '__getitem__', '__getnewargs__',
'__getslice__', '__gt__', '__hash__', '__init__', '__le__', '__len__',
'__lt__', '__mod__', '__mul__', '__ne__', '__new__', '__reduce__',
'__reduce_ex__', '__repr__', '__rmod__', '__rmul__','__setattr__',
 '__sizeof__', '__str__', '__subclasshook__', '_formatter_field_name_split',
 'capitalize', 'center', 'count', 'decode', 'encode', 'endswith',
'expandtabs',
 'find', 'format', 'index', 'isalnum', 'isalpha', 'isdigit', 'islower',
'isspace',
 'istitle','isupper', 'join', 'ljust', 'lower', 'lstrip', 'partition',
 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip',
 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title',
 'translate', 'upper', 'zfill']
```
