

Built-in data types

None	NoneType	None object
bool	Boolean	True, False
int	Integer	-2, -1, 0, 1, 2
long	Long Integer	123456789012345L
float	Floating point	3.1415
complex	complex	3 + 4*1j
str	String	'alice', 'bob'
unicode	Unicode String	u'alice', u'bob'
list	List	[1, 'two', 3.0]
tuple	Tuple	(1, 'two', 3.0)
dict	Dictionary	{'name': 'alice', 'age': 7}

Indentation matters

```
if Cheshire.is_visible:
    print('I can see you')
    alice.talks()
else:
    print("I can't see you")
```

Four spaces is the recommended by PEP 8 (Style Guide for Python Code)

String Literals

```
'alice'
"bob"
"The Hatter's Tea"
"""Twinkle, twinkle, little bat!
How I wonder what you're at!"""
```

String Methods

s.lower()	Lowercased copy
s.upper()	Uppercased copy
s.isalpha()	True if alphabetic
s.isalnum()	True if alphanumeric
s.split(t)	Split s using t as separator
s.zfill(w)	Padding with leading zeros
s.strip()	Removes beg/end spaces

Lists

lst.append(x)	Append x to the end of lst
lst.count(x)	How many times x is in lst
lst.extend(itr) Same as lst += itr	Append all of iterable itr to lst
lst.index(x)	Index position of the first occurrence
lst.insert(i,x)	Insert item x into lst at index i
lst.pop()	Removes the last item in lst
lst.pop(i)	Removes item with index i
lst.remove(x)	Removes the first occurrence of x in lst

Lists (cont)

lst.reverse()	Reverse the list lst in-place
lst.sort()	Sorts lst in-place

List Comprehensions

```
>>> [2**x for x in range(10)]
[1, 2, 4, 8, 16, 32, 64, 128, 256, 512]
```

```
>>> [2**x for x in range(10) if x%2 == 0]
[1, 4, 16, 64, 256]
```

Dictionaries

d.clear()	All items from dict d
d.copy()	Shallow copy of dict d
d.keys()	Read-only iterable of all keys
d.values()	Read-only iterable of all values
d.items()	Read-only iterable of all (key, value)
d.pop(k)	Removes (k, value) and returns d[k]

Conditionals

```
if x > 0 and y > 0:
    print('First Quadrant')
elif x < 0 and y > 0:
    print('Second Quadrant')
elif y < 0:
    print('Third and Fourth')
else:
    print('On the axis')
```

Loops (for)

```
a=1
b=2
for i in range(10):
    c=a+b
    print c
    a=b; b=c
```



Loops (while)

```
import random
x=1.0
while True:
    x*=random.random()
    print x
    if x<0.5:
        x*=3
        continue
    if x>0.8:
        break
```

You can use 'break' to escape from the inner loop and 'continue' to jump to the next item in the loop

Classes

```
class Quaternion(object):

    def __init__(self, four_tuple):
        self.values=four_tuple

    @property
    def real(self):
        return self.values[0]
```

Special Methods

<code>__bool__(self)</code>	True or False return
<code>__init__(self, args)</code>	Object Initialization
<code>__hash__(self)</code>	To be use as key
<code>__repr__(self)</code>	String eval(repr(x))==x
<code>__str__(self)</code>	Human readable string

Comparison Special Methods

<code>__lt__(self, other)</code>	<
<code>__le__(self, other)</code>	<=
<code>__gt__(self, other)</code>	>
<code>__ge__(self, other)</code>	>=
<code>__eq__(self, other)</code>	==
<code>__ne__(self, other)</code>	!=

os Module

<code>os.listdir(path)</code>	Equivalent to 'ls'
<code>os.getcwd()</code>	Current Working Directory (pwd)
<code>os.mkdir(path)</code>	Make directory (mkdir)
<code>os.chdir(path)</code>	Change directory (cd)
<code>os.path.isfile(path)</code>	Check if path is a file
<code>os.path.isdir(path)</code>	Check if path is directory
<code>os.path.exists(path)</code>	Check existence as file or directory

sys Module

<code>sys.argv</code>	List of arguments
<code>sys.version</code>	Python version
<code>sys.platform</code>	OS platform
<code>sys.maxint</code>	Max integer
<code>sys.stdout</code>	Standard output
<code>sys.stderr</code>	Standard error

itertools Module

<code>product('ABCD', repeat=2)</code>	AA AB AC AD BA BB BC BD CA CB CC CD DA DB DC DD
<code>permutations('ABCD', 2)</code>	AB AC AD BA BC BD CA CB CD DA DB DC
<code>combinations('ABCD', 2)</code>	AB AC AD BC BD CD
<code>combinations_with_replacement('ABCD', 2)</code>	AA AB AC AD BB BC BD CC CD DD

math Module

<code>sin, cos, tan</code>	Trigonometrical
<code>asin, acos, atan</code>	Inv. Trigonometrical
<code>sinh, cosh, tanh</code>	Hyperbolic
<code>asinh, acosh, atanh</code>	Inverse Hyperbolic
<code>ceil, floor, trunc</code>	Truncation
<code>log, log10</code>	Logarithms
<code>sqrt, exp, pow</code>	Exponentials
<code>degrees, radians</code>	Angular conversion
<code>pi, e</code>	Math Constants

json Module

```
>>> import json
>>> dic={'name':'Alice', 'age': 7.5}
>>> dic_json=json.dumps(dic,
                        sort_keys=True,
                        indent=4,
                        separators=(',', ': '))
>>> print dic_json
{
  "age": 7.5,
  "name": "Alice"
}
>>> dic2=json.loads(dic_json)
>>> dic2
{u'age': 7.5, u'name': u'Alice'}
```



By **Guillermo** (guilleaf)
cheatography.com/guilleaf/
bit.ly/guilleaf

Not published yet.
 Last updated 15th June, 2014.
 Page 2 of 2.

Sponsored by **Readability-Score.com**
 Measure your website readability!
<https://readability-score.com>