Python 3.1 Quick Reference

available at http://pedrokroger.net/python-quick-reference/

Built-in functions

abs(*x*)
all(*iter*)
any(*iter*)
ascii(*obj*)
bin(*x*)
bool(*[x]*)
bytearray(*[arg, [encoding, [errors]]]*)
bytes(*[arg, [encoding, [errors]]]*)
chr(*i*)
classmethod(*fn*)
compile(*source, filename, mode*)
complex(*real, [imag]*)
dict(*[arg]*)
dir(*[object]*)
divmod(*x, y*)
enumerate(*iter, start=0*)
eval(*source, [globals, [locals]]*)
filter(*fn, iter*)
float(*[x]*)
format(*alue, [format_spec]*)
frozenset(*[iter]*)
getattr(*object, name, [default]*)
globals()
hasattr(*object, name*)
hash(*object*)
help(*[object]*)
hex(*x*)
id(*object*)
input(*[prompt]*)
int(*[number | string, [base]]*)
isinstance(*object, classinfo*)
issubclass(*class, classinfo*)
iter(*object, sentinel*)
len(*object*)
list(*[iter]*)
locals()
map(*func, iter, ...*)
max(*iter, [args...], *, [key]*)
memoryview(*obj*)
min(*iter, [args...], *, [key]*)
next(*iterator, [default]*)
object()
oct(*number*)
open(*file, **keys*)
ord(*c*)
pow(*x, y, [z]*)
print(*[obj, ..., sep, end, file]*)
property(***keys*)
range(*[start,] stop, [step]*)
repr(*object*)
reversed(*sequence*)
round(*x, [n]*)
set(*iter*)
setattr(*object, name, value*)
slice(*[start], stop, [step]*)
sorted(*iter, [key], [reverse]*)
staticmethod(*unction*)
str(*[object, [encoding, [errors]]]*)
sum(*terable, [start]*)
super(*[type, [object-or-type]]*)
tuple(*[iter]*)
type(*object*)
type(*name, bases, dict*)
vars(*[object]*)
zip(**iters*)

Built-in constants

False, True, None
Ellipsis, NotImplemented

Boolean operations

```
not x
x or y
x and y
```

Comparisons

```
x < y
x <= y
x > y
x >= y
x == y
x != y
x is y
x is not y
x < y < z
```

Numeric operations

x + y	sum
x - y	subtraction
x * y	multiplication
x / y	quotient
x // y	floored quotient
x % y	remainder
-x	negation
+x	identity
x ** y	x to the power y

Bit-string Operations

x | y	and
x & y	and
x ^ y	exclusive or
x << n	bitwise left-shift
x >> n	right-shift
~x	bitwise invert (integers)

Extended Assignment

```
x += y     x /= y
x &= y     x >>= y
x -= y     x %= y
x |= y     x <<= y
x *= y     x **= y
x ^= y     x //= y
```

Sequence Assignment

```
w = [1, 2, 3, 4]
 a, *b = w      (1, [2, 3, 4])
 a, *b, c = w   (1, [2, 3], 4)
```

Sequence Operations

x in s	membership
x not in s	membership
s + t	concatenation
s * n, n * s	copy s n times
s[n]	nth item of s
s[i:j]	from i to j
s[i:j:k]	from i to j, step k

Slice examples

```
s = ['a', 'b', 'c']
 s[0]    'a'          first
 s[-1]   'c'          last
 s[1:]   ['b', 'c']   rest
 s[:-1]  ['a', 'b']   butlast
```

Float methods

float.as_integer_ratio()
float.hex()
float.fromhex(*s*)

List methods

append(*obj*)
extend(*iter*)
count(*value*)
index(*value, [start, [stop]]*)
insert(*pos, obj*)
pop(*[index]*)
remove(*value*)
reverse()
sort(*key=None, reverse=False*)

String methods

capitalize()
center(*width, [fillchar]*)
count(*sub, [start, [end]]*)
decode(*[encoding, [errors]]*)
encode(*[encoding[,errors]]*)
endswith(*suffix, [start, [end]]*)
expandtabs(*[tabsize]*)
find(*sub, [start, [end]]*)
format(**args, **kwargs*)
index(*sub, [start, [end]]*)
isalnum()
isalpha()
isdigit()
islower()
isspace()
istitle()
isupper()
join(*iterable*)
ljust(*width, [fillchar]*)
lower()
lstrip(*[chars]*)
partition(*sep*)
replace(*old, new, [count]*)
rfind(*sub [,start [,end]]*)
rindex(*sub, [start, [end]]*)
rjust(*width, [fillchar]*)
rpartition(*sep*)
rsplit(*[sep [,maxsplit]]*)
rstrip(*[chars]*)
split(*[sep, [maxsplit]]*)
splitlines(*[keepends]*)
startswith(*prefix, [start, [end]]*)
strip(*[chars]*)
swapcase()
title()
translate(*table, [deletechars]*)
upper()
zfill(*width*)

Set methods

add(*elem*)
clear()
copy()
difference(*other,...*)
difference_update(*other,...*)
discard(*elem*)
intersection(*other,...*)
intersection_update(*other,...*)
isdisjoint(*other*)
issubset(*other*)
issuperset(*other*)
pop()
remove(*elem*)
symmetric_difference(*other*)
symmetric_difference_update(*other*)
union(*other,...*)
update(*other,...*)

Dictionary methods

clear()
copy()
fromkeys(*seq, [value]*)
get(*key, [default]*)
items()
keys()
popitem()
pop(*key, [default]*)
setdefault(*key, [default]*)
update(*[other]*)
values()

File methods

close()
flush()
fileno()
isatty()
next()
read(*[size]*)
readline(*[size]*)
readlines(*[sizehint]*)
xreadlines()
seek(*offset, [whence]*)
tell()
truncate(*[size]*)
write(*str*)
writelines(*sequence*)

From future (python 2.6)

```
from __future__ import <f>
 absolute_import    PEP 328
 division           PEP 238
 print_function     PEP 3105
 unicode_literals   PEP 3112
```

Keywords (keyword.kwlist)

```
False None True and as assert
break class continue def del in
elif else except finally for is
from global if import lambda
nonlocal not or pass raise
return try while with yield
```

Built-in exceptions

```
BaseException
 ├─ SystemExit
 ├─ KeyboardInterrupt
 ├─ GeneratorExit
 ├─ Exception
 └─ StopIteration
```

```
BaseException
 ├─ AssertionError
 ├─ AttributeError
 ├─ BufferError
 ├─ ArithmeticError
 │   ├─ FloatingPointError
 │   ├─ OverflowError
 │   └─ ZeroDivisionError
 ├─ EnvironmentError
 │   ├─ IOError
 │   ├─ OSError
 │   ├─ WindowsError (Windows)
 │   └─ VMSError (VMS)
 ├─ EOFError
 ├─ ImportError
 ├─ LookupError
 │   ├─ IndexError
 │   └─ KeyError
 ├─ MemoryError
 ├─ NameError
 │   └─ UnboundLocalError
 ├─ ReferenceError
 ├─ RuntimeError
 │   └─ NotImplementedError
 ├─ SyntaxError
 │   ├─ IndentationError
 │   └─ TabError
 ├─ SystemError
 ├─ TypeError
 ├─ ValueError
 │   ├─ UnicodeError
 │   ├─ UnicodeDecodeError
 │   ├─ UnicodeEncodeError
 │   └─ UnicodeTranslateError
 ├─ Warning
 ├─ DeprecationWarning
 ├─ PendingDeprecationWarning
 ├─ RuntimeWarning
 ├─ SyntaxWarning
 ├─ UserWarning
 ├─ FutureWarning
 ├─ ImportWarning
 ├─ UnicodeWarning
 └─ BytesWarning
```