

# Python 2.7 Quick Reference Sheet

ver 2.01 – 110105 (sjd)

## Interactive Help in Python Shell

help()	Invoke interactive help
help( <i>m</i> )	Display help for module <i>m</i>
help( <i>f</i> )	Display help for function <i>f</i>
dir( <i>m</i> )	Display names in module <i>m</i>

## Small Operator Precedence Table

<i>func_name</i> ( <i>args</i> , ...)	Function call
<i>x</i> [ <i>index</i> : <i>index</i> ]	Slicing
<i>x</i> [ <i>index</i> ]	Indexing
<i>x.attribute</i>	Attribute reference
**	Exponentiation
*, /, %	Multiply, divide, mod
+, -	Add, subtract
>, <, <=, >=, !=, ==	Comparison
in, not in	Membership tests
not, and, or	Boolean operators NOT, AND, OR

## Module Import

import <i>module_name</i>
from <i>module_name</i> import <i>name</i> , ...
from <i>module_name</i> import *

## Common Data Types

Type	Description	Literal Ex
int	32-bit Integer	3, -4
long	Integer > 32 bits	101L
float	Floating point number	3.0, -6.55
complex	Complex number	1.2J
bool	Boolean	True, False
str	Character sequence	"Python"
tuple	Immutable sequence	(2, 4, 7)
list	Mutable sequence	[2, x, 3.1]
dict	Mapping	{ x:2, y:5 }

## Common Syntax Structures

<b>Assignment Statement</b> <i>var</i> = <i>exp</i>
<b>Console Input/Output</b> <i>var</i> = input( [ <i>prompt</i> ] ) <i>var</i> = raw_input( [ <i>prompt</i> ] ) print <i>exp</i> [,] ...
<b>Selection</b> if ( <i>boolean_exp</i> ): <i>stmt</i> ... [elif ( <i>boolean_exp</i> ): <i>stmt</i> ...] ... [else: <i>stmt</i> ...]
<b>Repetition</b> while ( <i>boolean_exp</i> ): <i>stmt</i> ...
<b>Traversal</b> for <i>var</i> in <i>traversable_object</i> : <i>stmt</i> ...
<b>Function Definition</b> def <i>function_name</i> ( <i>parameters</i> ): <i>stmt</i> ...
<b>Function Call</b> <i>function_name</i> ( <i>arguments</i> )
<b>Class Definition</b> class <i>Class_name</i> [ ( <i>super_class</i> ) ]: [ <i>class variables</i> ] def <i>method_name</i> ( <i>self</i> , <i>parameters</i> ): <i>stmt</i>
<b>Object Instantiation</b> <i>obj_ref</i> = <i>Class_name</i> ( <i>arguments</i> )
<b>Method Invocation</b> <i>obj_ref.method_name</i> ( <i>arguments</i> )
<b>Exception Handling</b> try: <i>stmt</i> ... except [ <i>exception_type</i> ] [, <i>var</i> ]: <i>stmt</i> ...

## Common Built-in Functions

Function	Returns
abs( <i>x</i> )	Absolute value of <i>x</i>
dict()	Empty dictionary, eg: d = dict()
float( <i>x</i> )	int or string <i>x</i> as float
id( <i>obj</i> )	memory addr of <i>obj</i>
int ( <i>x</i> )	float or string <i>x</i> as int
len( <i>s</i> )	Number of items in sequence <i>s</i>
list()	Empty list, eg: m = list()
max( <i>s</i> )	Maximum value of items in <i>s</i>
min( <i>s</i> )	Minimum value of items in <i>s</i>
open( <i>f</i> )	Open filename <i>f</i> for input
ord( <i>c</i> )	ASCII code of <i>c</i>
pow( <i>x,y</i> )	<i>x</i> ** <i>y</i>
range( <i>x</i> )	A list of <i>x</i> ints 0 to <i>x</i> - 1
round( <i>x,n</i> )	float <i>x</i> rounded to <i>n</i> places
str( <i>obj</i> )	str representation of <i>obj</i>
sum( <i>s</i> )	Sum of numeric sequence <i>s</i>
tuple( <i>items</i> )	tuple of <i>items</i>
type( <i>obj</i> )	Data type of <i>obj</i>

## Common Math Module Functions

Function	Returns (all float)
ceil( <i>x</i> )	Smallest whole nbr >= <i>x</i>
cos( <i>x</i> )	Cosine of <i>x</i> radians
degrees( <i>x</i> )	<i>x</i> radians in degrees
radians( <i>x</i> )	<i>x</i> degrees in radians
exp( <i>x</i> )	e ** <i>x</i>
floor( <i>x</i> )	Largest whole nbr <= <i>x</i>
hypot( <i>x, y</i> )	sqrt( <i>x</i> * <i>x</i> + <i>y</i> * <i>y</i> )
log( <i>x</i> [, <i>base</i> ])	Log of <i>x</i> to <i>base</i> or natural log if <i>base</i> not given
pow( <i>x, y</i> )	<i>x</i> ** <i>y</i>
sin( <i>x</i> )	Sine of <i>x</i> radians
sqrt( <i>x</i> )	Positive square root of <i>x</i>
tan( <i>x</i> )	Tangent of <i>x</i> radians
pi	Math constant pi to 15 sig figs
e	Math constant e to 15 sig figs

## Common String Methods

S.method()	Returns (str unless noted)
capitalize	S with first char uppercase
center(w)	S centered in str w chars wide
count(sub)	int nbr of non-overlapping occurrences of sub in S
find(sub)	int index of first occurrence of sub in S or -1 if not found
isdigit()	bool True if S is all digit chars, False otherwise
islower() isupper()	bool True if S is all lower/upper case chars, False otherwise
join(seq)	All items in seq concatenated into a str, delimited by S
lower() upper()	Lower/upper case copy of S
lstrip() rstrip()	Copy of S with leading/ trailing whitespace removed, or both
split([sep])	List of tokens in S, delimited by sep; if sep not given, delimiter is any whitespace

## Formatting Numbers as Strings

**Syntax:** `"format_spec" % numeric_exp`  
**format\_spec syntax:** `% width.precision type`

- width** (optional): align in number of columns specified; negative to left-align, precede with 0 to zero-fill
- precision** (optional): show specified digits of precision for floats; 6 is default
- type** (required): d (decimal int), f (float), s (string), e (float – exponential notation)
- Examples for `x = 123`, `y = 456.789`

```
"%6d" % x -> ... 123
"%06d" % x -> 000123
"%8.2f" % y -> . . 456.79
"8.2e" % y -> 4.57e+02
"-8s" % "Hello" -> Hello . . .
```

## Common List Methods

L.method()	Result/Returns
append(obj)	Append obj to end of L
count(obj)	Returns int nbr of occurrences of obj in L
index(obj)	Returns index of first occurrence of obj in L; raises ValueError if obj not in L
pop([index])	Returns item at specified index or item at end of L if index not given; raises IndexError if L is empty or index is out of range
remove(obj)	Removes first occurrence of obj from L; raises ValueError if obj is not in L
reverse()	Reverses L in place
sort()	Sorts L in place

## Common Tuple Methods

T.method()	Returns
count(obj)	Returns nbr of occurrences of obj in T
index(obj)	Returns index of first occurrence of obj in T; raises ValueError if obj is not in T

## Common Dictionary Methods

D.method()	Result/Returns
clear()	Remove all items from D
get(k [,val])	Return D[k] if k in D, else val
has_key(k)	Return True if k in D, else False
items()	Return list of key-value pairs in D; each list item is 2-item tuple
keys()	Return list of D's keys
pop(k, [val])	Remove key k, return mapped value or val if k not in D
values()	Return list of D's values

## Common File Methods

F.method()	Result/Returns
read([n])	Return str of next n chars from F, or up to EOF if n not given
readline([n])	Return str up to next newline, or at most n chars if specified
readlines()	Return list of all lines in F, where each item is a line
write(s)	Write str s to F
writelines(L)	Write all str in seq L to F
close()	Closes the file

## Other Syntax

**Hold window for user keystroke to close:**

```
raw_input("Press <Enter> to quit.")
```

**Prevent execution on import:**

```
if __name__ == "__main__":
    main()
```

## Displayable ASCII Characters

32	SP	48	0	64	@	80	P	96	`	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	\$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	105	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o	127	DEL

`\0' = 0, '\t' = 9, '\n' = 10`