

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

Assignment Statement <i>var</i> = <i>exp</i>
Console Input/Output <i>var</i> = input([<i>prompt</i>]) <i>var</i> = raw_input([<i>prompt</i>]) print <i>exp</i> [,] ...
Selection if (<i>boolean_exp</i>): <i>stmt</i> ... [elif (<i>boolean_exp</i>): <i>stmt</i> ...] ... [else: <i>stmt</i> ...]
Repetition while (<i>boolean_exp</i>): <i>stmt</i> ...
Traversal for <i>var</i> in <i>traversable_object</i> : <i>stmt</i> ...
Function Definition def <i>function_name</i> (<i>parmameters</i>): <i>stmt</i> ...
Function Call <i>function_name</i> (<i>arguments</i>)
Class Definition 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>
Object Instantiation <i>obj_ref</i> = <i>Class_name</i> (<i>arguments</i>)
Method Invocation <i>obj_ref.method_name</i> (<i>arguments</i>)
Exception Handling 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</i> , <i>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</i> , <i>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</i> , <i>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</i> , <i>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	<i>S</i> with first char uppercase
center(<i>w</i>)	<i>S</i> centered in str <i>w</i> chars wide
count(<i>sub</i>)	int nbr of non-overlapping occurrences of <i>sub</i> in <i>S</i>
find(<i>sub</i>)	int index of first occurrence of <i>sub</i> in <i>S</i> or -1 if not found
isdigit()	bool True if <i>S</i> is all digit chars, False otherwise
islower() isupper()	bool True if <i>S</i> is all lower/upper case chars, False otherwise
join(<i>seq</i>)	All items in <i>seq</i> concatenated into a str, delimited by <i>S</i>
lower() upper()	Lower/upper case copy of <i>S</i>
lstrip() rstrip()	Copy of <i>S</i> with leading/ trailing whitespace removed, or both
split([<i>sep</i>])	List of tokens in <i>S</i> , delimited by <i>sep</i> ; if <i>sep</i> 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(<i>obj</i>)	Append <i>obj</i> to end of <i>L</i>
count(<i>obj</i>)	Returns int nbr of occurrences of <i>obj</i> in <i>L</i>
index(<i>obj</i>)	Returns index of first occurrence of <i>obj</i> in <i>L</i> ; raises ValueError if <i>obj</i> not in <i>L</i>
pop([<i>index</i>])	Returns item at specified <i>index</i> or item at end of <i>L</i> if <i>index</i> not given; raises IndexError if <i>L</i> is empty or <i>index</i> is out of range
remove(<i>obj</i>)	Removes first occurrence of <i>obj</i> from <i>L</i> ; raises ValueError if <i>obj</i> is not in <i>L</i>
reverse()	Reverses <i>L</i> in place
sort()	Sorts <i>L</i> in place

Common Tuple Methods

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

Common Dictionary Methods

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

Common File Methods

F.method()	Result/Returns
read([<i>n</i>])	Return str of next <i>n</i> chars from <i>F</i> , or up to EOF if <i>n</i> not given
readline([<i>n</i>])	Return str up to next newline, or at most <i>n</i> chars if specified
readlines()	Return list of all lines in <i>F</i> , where each item is a line
write(<i>s</i>)	Write str <i>s</i> to <i>F</i>
writelines(<i>L</i>)	Write all str in seq <i>L</i> to <i>F</i>
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	106	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`