Python Programming Language Cheat Sheet

Logical tests

==	is equal to
!=	is not equal to
<>	is not equal to
and	and
or	or
not	not
>	greater than
>=	greater than or equal to
<	less than
<=	less than or equal to
is	true if the operands refer to the same object
is not	true if the operands refer to the different objects
in	true if the first operand is one of the elements of the second, which must be a sequence
not in	the opposite of in

Module management

dir()	lists all objects in the program's namespace
import <module>	imports <module> into its own namespace
import <module> as <new name>	imports <module> into the namespace <new name>
from <module> import *	import <module> into program's namespace
from <module> import func	import function into the program's namespace
dir(<module>)	lists all objects in the namespace <module>
reload <module>	Reinitialise module

Useful modules

Tkinter (& PWM)	Cross platform user interfaces
numpy	Efficient large arrays
scipy	Scientific data analysis including statistics
random	Random number generation
PIL	Python imaging library
matplotlib	2-D plotting and image production
os.path	Pathname manipulation
pickle	Object persistence
sys	System specific parameters and functions
os	Operating system interfaces
sqlite3	Access SQLite databases
MySQLdb	Access MySQL databases
re	Regular expressions

String format conversions

d or i	Signed integer, can also use u, but this is obsolete
o	Octal number
x or X	Lower or upper case hexadecimal format
e or E	Floating point exponential
f or F	Floating point decimal
g or G	Floating point decimal or exponential
c	single character
r	Any python data type (converted with repr())
s	Any python data type (converted with str())
%	A percentage sign

String format modifiers

n	format the number in a field of width n
0*n*	Format the number in a field of width n with leading zeroes
<space>	leaves a space for positive numbers or puts in a minus sign for negative numbers
+	Forces a plus for positive and a minus for negative numbers
#	Displays in an alternate format (depending on the data type)
-	left justifies the number not right justify

Assignment Operators

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

Bitwise Operators

&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR (XOR)
~	Bitwise NOT
x << n	Shift bits of *x* left by *n* bits
x >> n	Shift bits of *x* right by *n* bits

sys Variables

argv	Command line arguments (list)
builtin_module_names	Hardwired builtins
byteorder	Native byte order
check_interval	Signal check frequency
exec_prefix	Root directory
executable	Name of executable file
exitfunc	Exit function name
modules	*Loaded* modules
path	Search path used for finding modules
platform	Current platform or operating system
stdin, stdout, stderr	File objects for standard IO streams
version_info	Python version number
winver	Version number

os Variables

altsep	Alternative directory separator
curdir	Current directory string (.)
defpath	Default search path
devnull	Path of null device
extsep	Extension separator (.)
linesep	Line separator (\n unix-like, \r windows)
name	Name of operating system
pardir	Parent directory string (..)
pathsep	Separator for directories in search path
sep	Separator for folders in path string (/ unix-like, \ windows)

Operators

+	Addition (or concatenation or strings)
-	Subtraction (on unary minus)
*	Multiplication (or string repetition)
/	Division
%	Modulus (remainder) (or interpolation of strings)
//	Floor division
**	Exponentiation

Information gathered from various Python websites (mainly python.org) by Peter D. Ashton.

String tests

endswith(sub)	true if the string ends with <sub>
isalnum()	true if all characters are alphanumeric
isalpha()	true if all characters are alphabetic
isdigit()	true if all characters are digits
islower()	true if all characters are lowercase
isspace()	true is there is only whitespace in the string
istitle()	true if string is titlecased
isupper()	true if all characters are uppercase
startswith(sub)	true if string starts with <sub>

String formatting methods (return a new string with...)

.capitalize()	all lowercase with first character uppercase
center(width[, fillchar])	centre the string in a field <width> wide
expandtabs()	converts tab characters to multiple spaces
ljust(width[,fillchar])	left justifies string and fills to width with fillchar
lower()	all letters lowercase
lstrip()	leading whitespace removed
rjust(width)	original string right justified in a field of width
rstrip()	trailing whitespace removed
strip()	leading and trailing whitespace removed
swapcase()	lowercase changed to upper and vice versa
title()	string in title case
translate(table)	string translated according to table
upper()	all characters uppercase
zfill(width)	left fills the string with zeroes to a maximum of width characters

String searching methods

count(sub[,start[,end]])	counts the occurrences of <sub> between start and end
find(sub,start,end)	returns position of sub between start and end, or -1 if not found
index(sub,start.end)	returns position of sub between start and end, or raises error if not found
join(seq)	returns a string which is the concatenation of all the strings in the sequence
partition(sep)	splits string at first <sep>
replace(old,new)	returns string with all <old> changed to <new>
rfind(sub,start,end)	finds last occurrence of sub between start and end
rindex(sub.start.end)	finds last occurrence of sub between start and end
rpartition(sep)	splits string at last <sep>
rsplit(sep[, maxitems])	returns that last <maxitems> words
split(sep[, maxitems])	returns that first <maxitems> words
splitlines()	splits string at line breaks

List methods

append(item)	adds item to the end of the list
count(value)	returns a count of elements that match <value>
extend(list)	adds all the elements in <list> to the list
index(value)	returns the index of the first <value> in the list
insert(position, item)	inserts item in the list before <position>
pop(position)	removes and returns the element at position
remove(item)	removes the first occurrence of item in the list
reverse()	reverses the order of the elements in place
sort()	sorts the elements by value in place

Dictionary methods

clear()	removes all items from the dictionary
copy()	makes a "shallow copy" of the dictionary
get(key[, default])	returns d[key] if it exists, or default
has_key(key)	true if key exists in the dictionary
items()	returns all key, value pairs as a list of tuples
keys()	return a list of the keys
pop(key,[value])	returns the value for key, and deletes it
setdefault(key [,default])	returns d[key] and sets it to the default if it doesn't exist
update(d2)	merges d2 into the dictionary
values()	returns all values as a list

File methods

close()	closes the file
flush()	flushes the buffer to disk
fileno()	returns the operating system file number
isatty()	true if the file is a terminal
next()	return the next line from the file
read(size)	reads <size> bytes from the file
readline(size)	reads a line of at most <size> characters
readlines(size)	returns a list of strings of the lines in the file
seek(offset)	moves the next read to <offset> in the file
tell()	returns the current offset in the file
truncate(size)	sets the file size to <size> and deletes the rest
write(string)	writes the string to the file
writelines(list)	writes the list of string to the file as lines

Common datatype functions

len()	returns the length of the list/string/tuple/dictionary
max()	returns the maximum value in a list/tuple
min()	returns the minimum value in a list/tuple
any()	return true if any element is true
all()	returns true if all elements are true

Python statements - simple

expressions	Used to interactively evaluate expressions
assignment	Used to bind values to names
assert	Used in debugging to check expressions
pass	Do nothing
del	Deletes names and values
print	Output the following expressions
return	Return a value from a function
yield	Used in generator functions
break	Terminate the innermost enclosing loop
continue	Skips to the next iteration of a for or while loop
raise	Raises the last exception
global	Marks variable names as containing global data
exec	Execute python code contained in a string

Python statements - compound

if: elif: else:	Executes one of several blocks of code
while: else:	Repeat a block of code while expression is true
for in: else:	Loops over a list of values
try..except..finally	Exception handling of risky code
with	define a context for a block of code
def	define a new python function object
class	define a new class

Indices, slices & ranges

All of the below work on both strings and lists.

a[N]	returns the Nth item (character) in the array (string)
a[-1]	returns the last item
a[-4]	returns the 4th last item
a[N:]	returns from the Nth item to the end
a[:N]	returns up to the N-1th item
a[N:M]	returns from the Nth item to the M-1th
range(N)	list of values from 0..N-1
e.g., range(10)	[0,1,2,3,4,5,6,7,8,9]
range(start, N)	list of values from start..N-1
e.g., range(4,10)	[4,5,6,7,8,9]
range(start,N,step)	list of values from start..N-1 but increasing by step
e.g., range(4,10,2)	[4,6,8]