

Module Management	
<code>dir()</code>	list all objects in the programs namespace
<code>import module</code>	import <i>module</i> into its own namespace
<code>import module as alt</code>	import <i>module</i> into the namespace <i>alt</i>
<code>from module import func</code>	import the function <i>func</i> from <i>module</i> into the programs namespace
<code>dir(module)</code>	list all objects in the namespace <i>module</i>
<code>reload module</code>	reinitialise <i>module</i>

Useful Modules (General)	
<code>sys</code>	interacting with the system command line
<code>os</code>	interacting with the operating system
<code>fileinput</code>	simple input file processing
<code>argparse</code>	easily manage command line options and arguments
<code>random</code>	tools for generating pseudo-random data
<code>collections</code>	an expanded set of helpful data structures
<code>glob</code>	working with wildcards
<code>re</code>	regular expression searching/matching
<code>datetime</code>	time and date handling/processing
<code>urllib / urllib2</code>	working with webpages

Useful Modules (Data Handling/Plotting/Biology)	
<code>numpy</code>	tools for working with numerical data
<code>scipy</code>	scientific computing
<code>pandas</code>	improved data handling
<code>sklearn</code>	extensive machine learning module
<code>blaze</code>	working with very large datasets
<code>matplotlib</code>	powerful plotting library
<code>bokeh</code>	simple interactive plotting
<code>seaborn</code>	easy plotting for statistical data
<code>ggplot</code>	plotting similar to ggplot2 in R
<code>Bio</code>	Biopython (see biopython.org)

Statements - Simple	
<code>assert</code>	Used in debugging to check expressions
<code>pass</code>	Do nothing
<code>del</code>	Delete name or value
<code>print</code>	Output to STDOUT
<code>return</code>	Used in functions to return a value
<code>yield</code>	Used in generator functions
<code>break</code>	Terminate inner-most enclosing loop
<code>continue</code>	Skip to next iteration of enclosing loop
<code>raise</code>	Raise an exception
<code>global</code>	Mark variable as globally available
<code>exec</code>	Execute code contained in a string

Statements - Compound	
<code>if:elif:else:</code>	Conditional execution of code blocks
<code>while:else:</code>	Repeat code block while expression is true
<code>for in:else:</code>	loop through entries in a sequence
<code>try...except...finally</code>	exception handling of risky code
<code>with</code>	define context for a block of code
<code>def</code>	define a new function object
<code>class</code>	define a new class

Logical Tests	
<code>==</code>	is equal to
<code>!= , <></code>	is not equal to
<code>and</code>	and
<code>or</code>	or
<code>not</code>	not
<code>></code>	greater than
<code>>=</code>	greater than or equal to
<code><</code>	less than
<code><=</code>	less than or equal to
<code>is</code>	true if the operands refer to the same object
<code>is not</code>	opposite of the above
<code>in</code>	true if first operand contained within the second operand (a sequence)
<code>not in</code>	opposite of the above



Python Programming Language Cheat Sheet	
Standard Operators	
<code>+</code>	Addition (or string concatenation)
<code>-</code>	Subtraction (or unary minus)
<code>*</code>	Multiplication (or string repetition)
<code>/</code>	Division
<code>%</code>	Modulus (remainder)
<code>//</code>	Floor division
<code>**</code>	Exponentiation

Assignment Operators	
<code>x = y</code>	standard assignment
<code>x += y</code>	<code>x = x + y</code>
<code>x -= y</code>	<code>x = x - y</code>
<code>x *= y</code>	<code>x = x * y</code>
<code>x /= y</code>	<code>x = x / y</code>
<code>x %= y</code>	<code>x = x % y</code>
<code>x //= y</code>	<code>x = x // y</code>
<code>x <<= y</code>	<code>x = x << y</code>
<code>x >>= y</code>	<code>x = x >> y</code>
<code>x &= y</code>	<code>x = x & y</code>
<code>x = y</code>	<code>x = x y</code>
<code>x ^= y</code>	<code>x = x ^ y</code>

Bitwise Operators	
<code>&</code>	Bitwise AND
<code> </code>	Bitwise OR
<code>^</code>	Bitwise XOR
<code>~</code>	Bitwise NOT
<code>x << n</code>	Shift bits of <i>x</i> left by <i>n</i> bits
<code>x >> n</code>	Shift bits of <i>x</i> rightt by <i>n</i> bits

Common Datatype Functions

<code>len(item)</code>	returns the length of <i>item</i>
<code>max(item)</code>	returns the maximum value contained in a list or tuple
<code>min(item)</code>	returns the minimum value contained in a list or tuple
<code>any()</code>	returns true if any element is true
<code>all()</code>	returns true if all elements are true

Dictionary Methods

<code>clear()</code>	remove all entries in the dictionary
<code>copy()</code>	create a 'shallow' copy of dictionary
<code>get(key[, default])</code>	returns value associated with <i>key</i> , or default if <i>key</i> is not in the dictionary
<code>has_key(key)</code>	returns True if <i>key</i> exists in dictionary
<code>items()</code>	returns list of key-value pairs as tuples
<code>keys()</code>	returns a list of the keys in the dictionary

<code>pop(key[, default])</code>	removes entry for <i>key</i> & returns associated value or <i>default</i> if <i>key</i> is not in dictionary
<code>setdefault(key[, default])</code>	returns value associated with <i>key</i> , or creates <i>key-default</i> entry & returns <i>default</i> if <i>key</i> is not in dictionary
<code>update(dict2)</code>	adds entries of <i>dict2</i> into dictionary
<code>values()</code>	returns all values in a list

Indices, Slices, Ranges

<code>a[N]</code>	returns Nth item in sequence <i>a</i>
<code>a[-1]</code>	returns the last element in <i>a</i>
<code>a[-4]</code>	returns the 4th-from-last element of <i>a</i>
<code>a[N:]</code>	returns from Nth element to the end
<code>a[:N]</code>	returns from first to N-1th element
<code>a[N:M]</code>	returns from Nth to M-1th element
<code>range(N)</code>	returns list of integers from 0 - N-1
<code>range(M, N)</code>	returns list of integers from M - N-1
<code>range(M, N, L)</code>	returns list of integers from M - N-1 in steps of size <i>L</i>

List Methods

<code>append(item)</code>	adds <i>item</i> to the end of the list
<code>count(value)</code>	returns count of occurrences of <i>value</i> in list
<code>extend(lst2)</code>	adds elements in <i>lst2</i> to end of list
<code>index(value)</code>	returns index of 1st occurrence of <i>value</i> in list
<code>insert(i, item)</code>	inserts <i>item</i> into list at position <i>i</i>
<code>pop(i)</code>	removes and returns ith value
<code>remove(item)</code>	removes first occurrence of <i>item</i>
<code>reverse()</code>	reverses the order of the list
<code>sort()</code>	sorts list elements by value

String Methods (Formatting)

<code>capitalize()</code>	capitalise the first letter of every word
<code>center(width[, fillchar])</code>	center-align string in a field of <i>width</i> characters, filled by <i>fillchar</i> or spaces by default
<code>lower()</code>	convert all letters to lowercase
<code>rstrip()</code>	removes whitespace from the righthand side of the string
<code>strip()</code>	removes whitespace from both sides of the string
<code>swapcase()</code>	swaps uppercase to lower and <i>vice versa</i>
<code>upper()</code>	converts all letters to uppercase

String Methods (Searching)

<code>count(sub[, start[, end]])</code>	count the number of occurrences of <i>sub</i> in string, between <i>start</i> and <i>end</i> positions
<code>find(sub[, start[, end]])</code>	returns position of <i>sub</i> between <i>start</i> and <i>end</i> , or -1 if not found
<code>index(sub[, start[, end]])</code>	returns position of <i>sub</i> between <i>start</i> and <i>end</i> , raises error if not found
<code>replace(old, new)</code>	replace all instances of <i>old</i> in string with <i>new</i>
<code>split(sep[, maxitems])</code>	returns list of substrings split up by <i>sep</i> , optionally into <i>maxitems</i> items
<code>join(seq)</code>	returns a string of all entries in seq concatenated, with string between entries

String .format() Placeholder Codes

<code>{}</code>	inserts the value of the next method argument
<code>{N}</code>	inserts value of the Nth method argument
<code>{N:d}</code>	inserts Nth value as a decimal integer
<code>{N:.4f}</code>	inserts Nth value as a floating point to 4 decimal places
<code>{:,}</code>	splits large numbers into thousands by commas
<code>{:.2%}</code>	inserts value represented as a percentage, to two decimal places
<code>{N:x<10s}</code>	inserts a string, left-aligned in a field of width 10, with any remaining space filled with 'x' characters

String Methods (Testing)

<code>startswith(sub)</code>	true if string starts with substring <i>sub</i>
<code>endswith(sub)</code>	true if string ends with substring <i>sub</i>
<code>isalnum()</code>	true if all characters are alphanumeric
<code>isalpha()</code>	true if all characters are alphabetic
<code>isdigit()</code>	true if all characters are numeric
<code>islower()</code>	true if all characters are lowercase
<code>isspace()</code>	true if all characters are whitespace
<code>istitle()</code>	true if string is titlecase (all words are capitalised)
<code>isupper()</code>	true if all characters are uppercase

File Methods

<code>close()</code>	close connection to file
<code>readlines()</code>	read contents one line at a time
<code>readline()</code>	read the next line in the file
<code>read(size)</code>	read size bytes from the file
<code>fileno()</code>	returns the file number from the OS
<code>truncate(size)</code>	delete file contents after <i>size</i>
<code>write(string)</code>	write <i>string</i> to file
<code>writelines(list)</code>	write strings in <i>list</i> as lines to file