Loops

for i in range(x):

for v in list:

for idx, val in enumerate(*iterable*, *start=0*):

[expression for item in list]

while *true expression*:

Conditionals

if condition: ... elif condition: ... else: ...

[a for ... if condition]

[a if condition else b for ..]

10 > a > 100

if (x:=a+b) > c:

Strings

f'Variable {x} substitution'

Regular Expressions

import re

re.compile*(pattern)*	returns a regex object
re.search*(pattern, string)*	Find the first match and return a match object
re.match*(pattern, string)*	Returns a match object if the beginning of the string matches
re.split*(pattern, string, maxsplit)*	Split string by occurrences of pattern, complex!
re.findall*(pattern, string)*	Return all non-overlapping matches as a list of strings
re.finditer*(pattern, string)*	Returns in iterator yielding match objects over non-overlapping matches
re.sub*(pattern, repl, string, count)*	Returns a string, replacing the first occurrence of pattern with the repl string

Regular Expressions (cont)

re.sub*(pattern, repl, string, count)*	Similar to sub but return a tuple (new_string, number_of_subs_made)

Container Types

List	[] or [item1,item2] or list([iterable])	Ordered
Tuple	() or (item1, item2) or tuple([iterable])	Ordered, Immutable
Dictionary	{} or {key: value} or dict(a=1, b=2)	No *a priori* order, unique keys
Collection / Set	set() or {1,2,3} or set{key1, key2}	No *a priori* order, unique keys
String	'' or 'item' or str('item')	String is also iterable, immutable