

# Python Cheat Sheet: Keywords

“A puzzle a day to learn, code, and play” → Visit [finxter.com](https://finxter.com)

Keyword	Description	Code example
False, True	Data values from the data type Boolean	<code>False == (1 &gt; 2), True == (2 &gt; 1)</code>
and, or, not	Logical operators: (x and y) → both x and y must be True (x or y) → either x or y must be True (not x) → x must be false	<pre>x, y = True, False (x or y) == True      # True (x and y) == False    # True (not y) == True       # True</pre>
break	Ends loop prematurely	<pre>while(True):     break # no infinite loop     settings and more ... "ld")</pre>
continue	Finishes current loop iteration	<pre>while(True):     continue     print("43") # dead code</pre>
class	Defines a new class → a real-world concept (object oriented programming)	<pre>class Beer:     def __init__(self):         self.content = 1.0     def drink(self):         self.content = 0.0</pre>
def	Defines a new function or class method. For latter, first parameter ("self") points to the class object. When calling class method, first parameter is implicit.	<pre>becks = Beer() # constructor - create class becks.drink() # beer empty: b.content == 0</pre>
if, elif, else	Conditional program execution: program starts with "if" branch, tries the "elif" branches, and finishes with "else" branch (until one branch evaluates to True).	<pre>x = int(input("your value: ")) if x &gt; 3: print("Big") elif x == 3: print("Medium") else: print("Small")</pre>
for, while	<pre># For loop declaration for i in [0,1,2]:     print(i)</pre>	<pre># While loop - same semantics j = 0 while j &lt; 3:     print(j)     j = j + 1</pre>
in	Checks whether element is in sequence	<code>42 in [2, 39, 42] # True</code>
is	Checks whether both elements point to the same object	<pre>y = x = 3 x is y # True [3] is [3] # False</pre>
None	Empty value constant	<pre>def f():     x = 2 f() is None # True</pre>
lambda	Function with no name (anonymous function)	<code>(lambda x: x + 3)(3) # returns 6</code>
return	Terminates execution of the function and passes the flow of execution to the caller. An optional value after the return keyword specifies the function result.	<pre>def incrementor(x):     return x + 1 incrementor(4) # returns 5</pre>