

Python Cheat Sheet: Keywords

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