

# Python Programming Fundamentals Cheat Sheet

Package/Method	Description	Syntax and Code Example
AND	Returns `True` if both statement1 and statement2 are `True`. Otherwise, returns `False`.	<p>Syntax:</p> <pre>1 statement1 and statement2</pre> <p>Example:</p> <pre>1 marks = 90 2 attendance_percentage = 87 3 4 if marks &gt;= 80 and attendance_percentage &gt;= 85: 5     print("qualify for honors") 6 else: 7     print("Not qualified for honors") 8 9 # Output = qualify for honors</pre>
Class Definition	Defines a blueprint for creating objects and defining their attributes and behaviors.	<p>Syntax:</p> <pre>1 class ClassName: # Class attributes and methods</pre> <p>Example:</p> <pre>1 class Person: 2     def __init__(self, name, age): 3         self.name = name 4         self.age = age</pre>
Define Function	A `function` is a reusable block of code that performs a specific task or set of tasks when called.	<p>Syntax:</p> <pre>1 def function_name(parameters): # Function body</pre> <p>Example:</p> <pre>1 def greet(name): print("Hello,", name)</pre>
Equal(==)	Checks if two values are equal.	<p>Syntax:</p> <pre>1 variable1 == variable2</pre> <p>Example 1:</p> <pre>1 5 == 5</pre> <p>returns True</p> <p>Example 2:</p> <pre>1 age = 25 age == 30</pre> <p>returns False</p>
For Loop	A `for` loop repeatedly executes a block of code for a specified number of iterations or over a sequence of elements (list, range, string, etc.).	<p>Syntax:</p> <pre>1 for variable in sequence: # Code to repeat</pre> <p>Example 1:</p> <pre>1 for num in range(1, 10): 2     print(num)</pre> <p>Example 2:</p> <pre>1 fruits = ["apple", "banana", "orange", "grape"]</pre>

		<pre> 2     for fruit in fruits: 3         print(fruit) </pre>
Function Call	A function call is the act of executing the code within the function using the provided arguments.	<p>Syntax:</p> <pre> 1     function_name(arguments) </pre> <p>Example:</p> <pre> 1     greet("Alice") </pre>
Greater Than or Equal To(>=)	Checks if the value of variable1 is greater than or equal to variable2.	<p>Syntax:</p> <pre> 1     variable1 &gt;= variable2 </pre> <p>Example 1:</p> <pre> 1     5 &gt;= 5 and 9 &gt;= 5 </pre> <p>returns True</p> <p>Example 2:</p> <pre> 1     quantity = 105 2     minimum = 100 3     quantity &gt;= minimum </pre> <p>returns True</p>
Greater Than(>)	Checks if the value of variable1 is greater than variable2.	<p>Syntax:</p> <pre> 1     variable1 &gt; variable2 </pre> <p>Example 1: 9 &gt; 6</p> <p>returns True</p> <p>Example 2:</p> <pre> 1     age = 20 2     max_age = 25 3     age &gt; max_age </pre> <p>returns False</p>
If Statement	Executes code block `if` the condition is `True`.	<p>Syntax:</p> <pre> 1     if condition: #code block for if statement </pre> <p>Example:</p> <pre> 1     if temperature &gt; 30: 2         print("It's a hot day!") </pre>
If-Elif-Else	Executes the first code block if condition1 is `True`, otherwise checks condition2, and so on. If no condition is `True`, the else block is executed.	<p>Syntax:</p> <pre> 1     if condition1: 2         # Code if condition1 is True 3 4     elif condition2: 5         # Code if condition2 is True 6 7     else: 8         # Code if no condition is True </pre> <p>Example:</p> <pre> 1     score = 85 # Example score 2     if score &gt;= 90: </pre>

		<pre> 3         print("You got an A!") 4     elif score &gt;= 80: 5         print("You got a B.") 6     else: 7         print("You need to work harder.") 8 9     # Output = You got a B. </pre>
If-Else Statement	Executes the first code block if the condition is `True`, otherwise the second block.	<p>Syntax:</p> <pre> 1     if condition: # Code, if condition is True 2     else: # Code, if condition is False </pre> <p>Example:</p> <pre> 1     if age &gt;= 18: 2         print("You're an adult.") 3     else: 4         print("You're not an adult yet.") </pre>
Less Than or Equal To(<=)	Checks if the value of variable1 is less than or equal to variable2.	<p>Syntax:</p> <pre> 1     variable1 &lt;= variable2 </pre> <p>Example 1:</p> <pre> 1     5 &lt;= 5 and 3 &lt;= 5 </pre> <p>returns True</p> <p>Example 2:</p> <pre> 1     size = 38 2     max_size = 40 3     size &lt;= max_size </pre> <p>returns True</p>
Less Than(<)	Checks if the value of variable1 is less than variable2.	<p>Syntax:</p> <pre> 1     variable1 &lt; variable2 </pre> <p>Example 1:</p> <pre> 1     4 &lt; 6 </pre> <p>returns True</p> <p>Example 2:</p> <pre> 1     score = 60 2     passing_score = 65 3     score &lt; passing_score </pre> <p>returns True</p>
Loop Controls	`break` exits the loop prematurely. `continue` skips the rest of the current iteration and moves to the next iteration.	<p>Syntax:</p> <pre> 1     for: # Code to repeat 2         if # boolean statement 3             break 4 5     for: # Code to repeat 6         if # boolean statement 7             continue </pre> <p>Example 1:</p>

		<pre> 1   for num in range(1, 6): 2       if num == 3: 3           break 4       print(num) </pre> <p>Example 2:</p> <pre> 1   for num in range(1, 6): 2       if num == 3: 3           continue 4       print(num) </pre>
NOT	Returns `True` if variable is `False`, and vice versa.	<p>Syntax:</p> <pre> 1   !variable </pre> <p>Example:</p> <pre> 1   !isLocked </pre> <p>returns True if the variable is False (i.e., unlocked).</p>
Not Equal(!=)	Checks if two values are not equal.	<p>Syntax:</p> <pre> 1   variable1 != variable2 </pre> <p>Example:</p> <pre> 1   a = 10 2   b = 20 3   a != b </pre> <p>returns True</p> <p>Example 2:</p> <pre> 1   count=0 2   count != 0 </pre> <p>returns False</p>
Object Creation	Creates an instance of a class (object) using the class constructor.	<p>Syntax:</p> <pre> 1   object_name = ClassName(arguments) </pre> <p>Example:</p> <pre> 1   person1 = Person("Alice", 25) </pre>
OR	Returns `True` if either statement1 or statement2 (or both) are `True`. Otherwise, returns `False`.	<p>Syntax:</p> <pre> 1   statement1    statement2 </pre> <p>Example:</p> <pre> 1   "Farewell Party Invitation" 2   Grade = 12 grade == 11 or grade == 12 </pre> <p>returns True</p>
range()	Generates a sequence of numbers within a specified range.	<p>Syntax:</p> <pre> 1   range(stop) 2   range(start, stop) 3   range(start, stop, step) </pre> <p>Example:</p>

		<pre> 1  range(5) #generates a sequence of integers frc 2  range(2, 10) #generates a sequence of integers 3  range(1, 11, 2) #generates odd integers from 1 </pre>
Return Statement	`Return` is a keyword used to send a value back from a function to its caller.	<p>Syntax:</p> <pre> 1  return value </pre> <p>Example:</p> <pre> 1  def add(a, b): return a + b 2  result = add(3, 5) </pre>
Try-Except Block	Tries to execute the code in the try block. If an exception of the specified type occurs, the code in the except block is executed.	<p>Syntax:</p> <pre> 1  try: # Code that might raise an exception exce 2  ExceptionType: # Code to handle the exception </pre> <p>Example:</p> <pre> 1  try: 2      num = int(input("Enter a number: ")) 3  except ValueError: 4      print("Invalid input. Please enter a valid </pre>
Try-Except with Else Block	Code in the `else` block is executed if no exception occurs in the try block.	<p>Syntax:</p> <pre> 1  try: # Code that might raise an exception exce 2  ExceptionType: # Code to handle the exception 3  else: # Code to execute if no exception occurs </pre> <p>Example:</p> <pre> 1  try: 2      num = int(input("Enter a number: ")) 3  except ValueError: 4      print("Invalid input. Please enter a valid 5  else: 6      print("You entered:", num) </pre>
Try-Except with Finally Block	Code in the `finally` block always executes, regardless of whether an exception occurred.	<p>Syntax:</p> <pre> 1  try: # Code that might raise an exception exce 2  ExceptionType: # Code to handle the exception 3  finally: # Code that always executes </pre> <p>Example:</p> <pre> 1  try: 2      file = open("data.txt", "r") 3      data = file.read() 4  except FileNotFoundError: 5      print("File not found.") 6  finally: 7      file.close() </pre>
While Loop	A `while` loop repeatedly executes a block of code as long as a specified condition remains `True`.	<p>Syntax:</p> <pre> 1  while condition: # Code to repeat </pre> <p>Example:</p> <pre> 1  count = 0 while count &lt; 5: 2      print(count) count += 1 </pre>

© IBM Corporation. All rights reserved.