

# 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. 1  1. statement1 and statement2</pre> <div>Copied!</div> <p>Example:</p> <pre>1. 1 2. 2 3. 3 4. 4 5. 5 6. 6 7. 7 8. 8 9. 9  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> <div>Copied!</div>
		<p>Syntax:</p> <pre>1. 1  1. class ClassName: # Class attributes and methods</pre> <div>Copied!</div> <p>Example:</p> <pre>1. 1 2. 2 3. 3 4. 4  1. class Person: 2.     def __init__(self, name, age): 3.         self.name = name 4.         self.age = age</pre> <div>Copied!</div>
Class Definition	Defines a blueprint for creating objects and defining their attributes and behaviors.	<p>Syntax:</p> <pre>1. 1  1. def function_name(parameters): # Function body</pre> <div>Copied!</div> <p>Example:</p> <pre>1. 1 2. 2 3. 3 4. 4  1. class Person: 2.     def __init__(self, name, age): 3.         self.name = name 4.         self.age = age</pre> <div>Copied!</div>
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. 1  1. def greet(name): print("Hello,", name)</pre> <div>Copied!</div> <p>Example:</p> <pre>1. 1  1. def greet(name): print("Hello,", name)</pre> <div>Copied!</div> <p>Syntax:</p> <pre>1. 1  1. variable1 == variable2</pre> <div>Copied!</div> <p>Example 1:</p> <pre>1. 1  1. 5 == 5</pre> <div>Copied!</div> <p>returns True</p> <p>Example 2:</p> <pre>1. 1  1. age = 25 age == 30</pre> <div>Copied!</div> <p>returns False</p>
Equal(==)	Checks if two values are equal.	<p>Syntax:</p> <pre>1. 1  1. for variable in sequence: # Code to repeat</pre> <div>Copied!</div>
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.).	

		<div>Example 1:</div> <div><pre>1. 1 2. 2  1. for num in range(1, 10): 2.     print(num)</pre></div> <div>Copied!</div> <div>Example 2:</div> <div><pre>1. 1 2. 2 3. 3  1. fruits = ["apple", "banana", "orange", "grape", "kiwi"] 2. for fruit in fruits: 3.     print(fruit)</pre></div> <div>Copied!</div> <div>Syntax:</div> <div><pre>1. 1  1. function_name(arguments)</pre></div> <div>Copied!</div> <div>Example:</div> <div><pre>1. 1  1. greet("Alice")</pre></div> <div>Copied!</div> <div>Syntax:</div> <div><pre>1. 1  1. variable1 &gt;= variable2</pre></div> <div>Copied!</div> <div>Example 1:</div> <div><pre>1. 1  1. 5 &gt;= 5 and 9 &gt;= 5</pre></div> <div>Copied!</div> <div>returns True</div> <div>Example 2:</div> <div><pre>1. 1 2. 2 3. 3  1. quantity = 105 2. minimum = 100 3. quantity &gt;= minimum</pre></div> <div>Copied!</div> <div>returns True</div> <div>Syntax:</div> <div><pre>1. 1  1. variable1 &gt; variable2</pre></div> <div>Copied!</div> <div>Example 1: 9 &gt; 6</div> <div>returns True</div> <div>Example 2:</div> <div><pre>1. 1 2. 2 3. 3  1. age = 20 2. max_age = 25 3. age &gt; max_age</pre></div> <div>Copied!</div> <div>returns False</div> <div>Syntax:</div> <div><pre>1. 1  1. if condition: #code block for if statement</pre></div> <div>Copied!</div> <div>Example:</div> <div><pre>1. 1 2. 2  1. if temperature &gt; 30:</pre></div>
Function Call	A function call is the act of executing the code within the function using the provided arguments.	
Greater Than or Equal To(>=)	Checks if the value of variable1 is greater than or equal to variable2.	
Greater Than(>)	Checks if the value of variable1 is greater than variable2.	
If Statement	Executes code block `if` the condition is `True`.	

		<pre>2. print("It's a hot day!")</pre> <div>Copied!</div>
		Syntax: <pre> 1. 1 2. 2 3. 3 4. 4 5. 5 6. 6 7. 7 8. 8  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> <div>Copied!</div>
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.	Example: <pre> 1. 1 2. 2 3. 3 4. 4 5. 5 6. 6 7. 7 8. 8 9. 9  1. score = 85 # Example score 2. if score &gt;= 90: 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> <div>Copied!</div>
		Syntax: <pre> 1. 1 2. 2  1. if condition: # Code, if condition is True 2. else: # Code, if condition is False</pre> <div>Copied!</div>
If-Else Statement	Executes the first code block if the condition is `True`, otherwise the second block.	Example: <pre> 1. 1 2. 2 3. 3 4. 4  1. if age &gt;= 18: 2.     print("You're an adult.") 3. else: 4.     print("You're not an adult yet.")</pre> <div>Copied!</div>
		Syntax: <pre> 1. 1  1. variable1 &lt;= variable2</pre> <div>Copied!</div>
		Example 1: <pre> 1. 1  1. 5 &lt;= 5 and 3 &lt;= 5</pre> <div>Copied!</div>
Less Than or Equal To(<=)	Checks if the value of variable1 is less than or equal to variable2.	returns True Example 2: <pre> 1. 1 2. 2 3. 3  1. size = 38 2. max_size = 40 3. size &lt;= max_size</pre> <div>Copied!</div>
Less Than(<)	Checks if the value of variable1 is less than variable2.	returns True Syntax: <pre> 1. 1  1. variable1 &lt; variable2</pre>

		<div>Copied!</div> <div>Example 1:</div> <pre> 1. 1  1. 4 &lt; 6 </pre> <div>Copied!</div> <div>returns True</div> <div>Example 2:</div> <pre> 1. 1 2. 2 3. 3  1. score = 60 2. passing_score = 65 3. score &lt; passing_score </pre> <div>Copied!</div> <div>returns True</div> <div>Syntax:</div> <pre> 1. 1 2. 2 3. 3 4. 4 5. 5 6. 6 7. 7  1. for: # Code to repeat 2.     if # boolean statement 3.         break 4. 5. for: # Code to repeat 6.     if # boolean statement 7.         continue </pre> <div>Copied!</div>
Loop Controls	<div> <div>`break` exits the loop prematurely. `continue` skips the rest of the current iteration and moves to the next iteration.</div> </div>	<div>Example 1:</div> <pre> 1. 1 2. 2 3. 3 4. 4  1. for num in range(1, 6): 2.     if num == 3: 3.         break 4.     print(num) </pre> <div>Copied!</div> <div>Example 2:</div> <pre> 1. 1 2. 2 3. 3 4. 4  1. for num in range(1, 6): 2.     if num == 3: 3.         continue 4.     print(num) </pre> <div>Copied!</div> <div>Syntax:</div> <pre> 1. 1  1. !variable </pre> <div>Copied!</div>
NOT	<div> <div>Returns `True` if variable is `False`, and vice versa.</div> </div>	<div>Example:</div> <pre> 1. 1  1. !isLocked </pre> <div>Copied!</div> <div>returns True if the variable is False (i.e., unlocked).</div> <div>Syntax:</div> <pre> 1. 1  1. variable1 != variable2 </pre> <div>Copied!</div>
Not Equal(!=)	<div> <div>Checks if two values are not equal.</div> </div>	<div>Example:</div> <pre> 1. 1 2. 2 3. 3  1. a = 10 2. b = 20 3. a != b </pre>

		<div>Copied!</div> <pre> returns True  Example 2:  1. 1 2. 2  1. count=0 2. count != 0  Copied!  returns False Syntax:  1. 1  1. object_name = ClassName(arguments)  Copied!  Example:  1. 1  1. person1 = Person("Alice", 25)  Copied!  Syntax:  1. 1  1. statement1    statement2  Copied!  Example:  1. 1 2. 2  1. "Farewell Party Invitation" 2. Grade = 12 grade == 11 or grade == 12  Copied!  returns True Syntax:  1. 1 2. 2 3. 3  1. range(stop) 2. range(start, stop) 3. range(start, stop, step)  Copied!  Example:  1. 1 2. 2 3. 3  1. range(5) #generates a sequence of integers from 0 to 4. 2. range(2, 10) #generates a sequence of integers from 2 to 9. 3. range(1, 11, 2) #generates odd integers from 1 to 9.  Copied!  Syntax:  1. 1  1. return value  Copied!  Example:  1. 1 2. 2  1. def add(a, b): return a + b 2. result = add(3, 5)  Copied!  Syntax:  1. 1 2. 2  1. try: # Code that might raise an exception except 2. ExceptionType: # Code to handle the exception  Copied!  Example:  1. 1 2. 2 3. 3 4. 4 </pre>
Object Creation	Creates an instance of a class (object) using the class constructor.	
OR	Returns `True` if either statement1 or statement2 (or both) are `True`. Otherwise, returns `False`.	
range()	Generates a sequence of numbers within a specified range.	
Return Statement	`Return` is a keyword used to send a value back from a function to its caller.	
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.	

```
1. try:
2.     num = int(input("Enter a number: "))
3. except ValueError:
4.     print("Invalid input. Please enter a valid number.")
```

Copied!

Syntax:

```
1. 1
2. 2
3. 3
```

```
1. try: # Code that might raise an exception except
2.     ExceptionType: # Code to handle the exception
3. else: # Code to execute if no exception occurs
```

Copied!

Example:

```
1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
```

```
1. try:
2.     num = int(input("Enter a number: "))
3. except ValueError:
4.     print("Invalid input. Please enter a valid number")
5. else:
6.     print("You entered:", num)
```

Copied!

Syntax:

```
1. 1
2. 2
3. 3
```

```
1. try: # Code that might raise an exception except
2.     ExceptionType: # Code to handle the exception
3. finally: # Code that always executes
```

Copied!

Example:

```
1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
```

```
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()
```

Copied!

Syntax:

```
1. 1
```

```
1. while condition: # Code to repeat
```

Copied!

Example:

```
1. 1
2. 2
```

```
1. count = 0 while count < 5:
2.     print(count) count += 1
```

Copied!

Try-Except with Else Block

Code in the `else` block is executed if no exception occurs in the try block.

Try-Except with Finally Block

Code in the `finally` block always executes, regardless of whether an exception occurred.

While Loop

A `while` loop repeatedly executes a block of code as long as a specified condition remains `True`.



**Skills** Network