

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`.	<div>Syntax:</div> <div>statement1 and statement2</div> <div>Example:</div> <div>marks = 90 attendance_percentage = 87 if marks >= 80 and attendance_percentage >= 85: print("qualify for honors") else: print("Not qualified for honors") # Output = qualify for honors</div>
Class Definition	Defines a blueprint for creating objects and defining their attributes and behaviors.	<div>Syntax:</div> <div>class ClassName: # Class attributes and methods</div> <div>Example:</div> <div>class Person: def __init__(self, name, age): self.name = name self.age = age</div>
Define Function	A `function` is a reusable block of code that performs a specific task or set of tasks when called.	<div>Syntax:</div> <div>def function_name(parameters): # Function body</div> <div>Example:</div> <div>def greet(name): print("Hello,", name)</div>

Equal(==)	Checks if two values are equal.	<div>Syntax: <pre>variable1 == variable2</pre></div> <div>Example 1: <pre>5 == 5</pre></div> <div>returns True</div> <div>Example 2: <pre>age = 25 age == 30</pre></div> <div>returns False</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>Syntax: <pre>for variable in sequence: # Code to repeat</pre></div> <div>Example 1: <pre>for num in range(1, 10): print(num)</pre></div> <div>Example 2: <pre>fruits = ["apple", "banana", "orange", "grape", "kiwi"] for fruit in fruits: print(fruit)</pre></div>

Function Call	A function call is the act of executing the code within the function using the provided arguments.	<p>Syntax:</p> <pre>function_name(arguments)</pre> <p>Example:</p> <pre>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>variable1 >= variable2</pre> <p>Example 1:</p> <pre>5 >= 5 and 9 >= 5</pre> <p>returns True</p> <p>Example 2:</p> <pre>quantity = 105 minimum = 100 quantity >= minimum</pre> <p>returns True</p>
Greater Than(>)	Checks if the value of variable1 is greater than variable2.	<p>Syntax:</p> <pre>variable1 > variable2</pre> <p>Example 1: 9 > 6</p> <p>returns True</p> <p>Example 2:</p>

		<pre>age = 20 max_age = 25 age > max_age</pre> <p>returns False</p>
If Statement	Executes code block `if` the condition is `True`.	<p>Syntax:</p> <pre>if condition: #code block for if statement</pre> <p>Example:</p> <pre>if temperature > 30: 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>if condition1: # Code if condition1 is True elif condition2: # Code if condition2 is True else: # Code if no condition is True</pre> <p>Example:</p> <pre>score = 85 # Example score if score >= 90: print("You got an A!") elif score >= 80: print("You got a B.") else: print("You need to work harder.") # 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>if condition: # Code, if condition is True else: # Code, if condition is False</pre>

Example:

```
if age >= 18:
    print("You're an adult.")
else:
    print("You're not an adult yet.")
```

Syntax:

```
variable1 <= variable2
```

Example 1:

```
5 <= 5 and 3 <= 5
```

returns True

Example 2:

```
size = 38
max_size = 40
size <= max_size
```

returns True

Syntax:

```
variable1 < variable2
```

Example 1:

```
4 < 6
```

returns True

Example 2:

```
score = 60
passing_score = 65
score < passing_score
```

returns True

Syntax:

```
for: # Code to repeat
    if # boolean statement
        break
for: # Code to repeat
    if # boolean statement
        continue
```

Example 1:

```
for num in range(1, 6):
    if num == 3:
        break
    print(num)
```

Example 2:

```
for num in range(1, 6):
    if num == 3:
        continue
    print(num)
```

Syntax:

```
not variable
```

Example:

```
isLocked = False
print(not isLocked)
```

Loop Controls

'break' exits the loop prematurely. 'continue' skips the rest of the current iteration and moves to the next iteration.

NOT

Returns 'True' if variable is 'False', and vice versa.

		returns True if the variable is False (i.e., unlocked).
Not Equal(!=)	Checks if two values are not equal.	<div>Syntax: <pre>variable1 != variable2</pre></div> <div>Example: <pre>a = 10 b = 20 a != b</pre></div> <div>returns True</div> <div>Example 2: <pre>count=0 count != 0</pre></div> <div>returns False</div>
Object Creation	Creates an instance of a class (object) using the class constructor.	<div>Syntax: <pre>object_name = ClassName(arguments)</pre></div> <div>Example: <pre>person1 = Person("Alice", 25)</pre></div>
OR	Returns `True` if either statement1 or statement2 (or both) are `True`. Otherwise, returns `False`.	<div>Syntax: <pre>statement1 or statement2</pre></div>

		<p>Example:</p> <pre>"Farewell Party Invitation" grade = 12 if grade == 11 or grade == 12: print("Farewell Party Invitation") else: print("Not eligible")</pre> <p>returns True</p>
range()	Generates a sequence of numbers within a specified range.	<p>Syntax:</p> <pre>range(stop) range(start, stop) range(start, stop, step)</pre> <p>Example:</p> <pre>range(5) #generates a sequence of integers from 0 to 4. range(2, 10) #generates a sequence of integers from 2 to 9. range(1, 11, 2) #generates odd integers from 1 to 9.</pre>
Return Statement	'Return' is a keyword used to send a value back from a function to its caller.	<p>Syntax:</p> <pre>return value</pre> <p>Example:</p> <pre>def add(a, b): return a + b 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>try: # Code that might raise an exception except ExceptionType: # Code to handle the exception</pre>

		<p>Example:</p> <pre>try: num = int(input("Enter a number: ")) except ValueError: print("Invalid input. Please enter a valid number.")</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>try: # Code that might raise an exception except ExceptionType: # Code to handle the exception else: # Code to execute if no exception occurs</pre> <p>Example:</p> <pre>try: num = int(input("Enter a number: ")) except ValueError: print("Invalid input. Please enter a valid number") else: 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>try: # Code that might raise an exception except ExceptionType: # Code to handle the exception finally: # Code that always executes</pre> <p>Example:</p> <pre>try: file = open("data.txt", "r") data = file.read() except FileNotFoundError: print("File not found.") finally: 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>while condition: # Code to repeat</pre>

Example:

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



Skills Network

© IBM Corporation. All rights reserved.