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`.	Syntax:
		`1. 1`
		`1. statement1 and statement2`
		Copied!
		Example:
		`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 >= 80 and attendance_percentage >= 85:`
		`5.  print("qualify for honors")`
		`6. else:`
		`7.  print("Not qualified for honors")`
		`8.`
		`9. # Output = qualify for honors`
		Copied!

Class Definition	Defines a blueprint for creating objects and defining their attributes and behaviors.	Syntax:
		`1. 1`
		`1. class ClassName: # Class attributes and methods`
		Copied!
		Example:
		`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`
		Copied!

Define Function	A `function` is a reusable block of code that performs a specific task or set of tasks when called.	Syntax:
		`1. 1`
		`1. def function_name(parameters): # Function body`
		Copied!
		Example:
		`1. 1`
		`1. def greet(name): print("Hello,", name)`
		Copied!

Equal(==)	Checks if two values are equal.	Syntax:
		`1. 1`
		`1. variable1 == variable2`
		Copied!
		Example 1:
		`1. 1`
		`1. 5 == 5`
		Copied!
		returns True
		Example 2:
		`1. 1`
		`1. age = 25 age == 30`
		Copied!
		returns False

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.).	Syntax:
		`1. 1`
		`1. for variable in sequence: # Code to repeat`
		Copied!
		Example 1:
		`1. 1`
		`2. 2`
		`1. for num in range(1, 10):`
		`2.  print(num)`
		Copied!
		Example 2:
		`1. 1`

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

```
1
2
3
4
5
6
7
8
9
```

```
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.
```

Copied!

Syntax:

```
1
2
```

```
if condition: # Code, if condition is True
else: # Code, if condition is False
```

Copied!

If-Else Statement	Executes the first code block if the condition is `True`, otherwise the second block.

Example:

```
1
2
3
4
```

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

Copied!

Syntax:

```
1
```

```
variable1 <= variable2
```

Copied!

Less Than or Equal To(<=)	Checks if the value of variable1 is less than or equal to variable2.

Example 1:

```
1
```

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

Copied!

returns True

Example 2:

```
1
2
3
```

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

Copied!

returns True

Syntax:

```
1
```

```
variable1 < variable2
```

Copied!

Less Than(<)	Checks if the value of variable1 is less than variable2.

Example 1:

```
1
```

```
4 < 6
```

Copied!

returns True

Example 2:

```
1
2
3
```

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

Copied!

returns True

Syntax:

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

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

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

for: # Code to repeat
```

```
6.     if # boolean statement
7.         continue
```

Copied!

Example 1:

```
1. 1
2. 2
3. 3
4. 4
```

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

Copied!

Example 2:

```
1. 1
2. 2
3. 3
4. 4
```

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

Copied!

NOT	Returns `True` if variable is `False`, and vice versa.
Not Equal(!=)	Checks if two values are not equal.
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.

Syntax:

```
1. 1
```

```
1. !variable
```

Copied!

Example:

```
1. 1
```

```
1. !isLocked
```

Copied!

returns True if the variable is False (i.e., unlocked).

Syntax:

```
1. 1
```

```
1. variable1 != variable2
```

Copied!

Example:

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

```
1. a = 10
2. b = 20
3. a != b
```

Copied!

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!
Return Statement	`Return` is a keyword used to send a value back from a function to its caller.	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!
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.	Example: 1. 1 2. 2 3. 3 4. 4 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!
Try-Except with Else Block	Code in the `else` block is executed if no exception occurs in the try block.	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!
Try-Except with Finally Block	Code in the `finally` block always executes, regardless of whether an exception occurred.	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!
While Loop	A `while` loop repeatedly executes a block of code as long as a specified condition remains `True`.	Syntax: 1. 1

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

Copied!

Example:

```
1
2
```

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

Copied!