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'.	<pre>Example: 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</pre>
Class Definition	Defines a blueprint for creating objects and defining their attributes and behaviors.	Syntax: class ClassName: # Class attributes and methods Example: class Person: definit(self, name, age): self.name = name self.age = age
Define Function	A 'function' is a reusable block of code that performs a specific task or set of tasks when called.	Syntax: def function_name(parameters): # Function body Example: def greet(name): print("Hello,", name)

		Syntax:
		variable1 == variable2
		Example 1:
		5 == 5
Equal(==)	Checks if two values are equal.	
		returns True
		Example 2:
		age = 25 age == 30
		returns False
		Syntax:
		for variable in sequence: # Code to repeat
		Example 1:
		for num in range(1, 10): print(num)
For Loop	A 'for' loop repeatedly executes a block of code for a specified number of iterations or over a sequence of elements	
Tor Loop	specified number of iterations of over a sequence of elements	
	(list, range, string, etc.).	Example 2:
	(list, range, string, etc.).	fruits = ["apple", "banana", "orange", "grape", "kiwi"]
	(list, range, string, etc.).	
	(list, range, string, etc.).	<pre>fruits = ["apple", "banana", "orange", "grape", "kiwi"] for fruit in fruits:</pre>
	(list, range, string, etc.).	<pre>fruits = ["apple", "banana", "orange", "grape", "kiwi"] for fruit in fruits:</pre>
	(list, range, string, etc.).	<pre>fruits = ["apple", "banana", "orange", "grape", "kiwi"] for fruit in fruits:</pre>
	(list, range, string, etc.).	<pre>fruits = ["apple", "banana", "orange", "grape", "kiwi"] for fruit in fruits:</pre>
Function Call	(list, range, string, etc.). A function call is the act of executing the code within the	<pre>fruits = ["apple", "banana", "orange", "grape", "kiwi"] for fruit in fruits:</pre>

1	•	
		<pre>Example: greet("Alice")</pre>
		Syntax:
		Syntax: variable1 >= variable2
		Example 1: 5 >= 5 and 9 >= 5
Greater Than or Equal To(>=)	Checks if the value of variable 1 is greater than or equal to variable 2.	
		returns True Example 2: quantity = 105 minimum = 100 quantity >= minimum
		returns True
Greater Than(>)	Checks if the value of variable 1 is greater than variable 2.	Syntax: variable1 > variable2
		Example 1: 9 > 6 returns True Example 2: age = 20 max_age = 25 age > max_age

		returns False Syntax: if condition: #code block for if statement
If Statement	Executes code block `if` the condition is `True`.	<pre>Example: if temperature > 30: print("It's a hot day!")</pre>
		Syntax: if condition1: # Code if condition1 is True elif condition2: # Code if condition2 is True else: # Code if no condition is True
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: 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.
If-Else Statement	Executes the first code block if the condition is 'True', otherwise the second block.	Syntax: if condition: # Code, if condition is True else: # Code, if condition is False
		Example:

<pre>if age >= 18: print("You're an adult.")</pre>	
else: print("You're not an adult yet.")	
Syntax:	
<pre>variable1 <= variable2</pre>	
Example 1:	
5 <= 5 and 3 <= 5	
Than or Equal Checks if the value of variable1 is less than or equal to variable2.	
returns True	
Example 2:	
size = 38 max_size = 40	
size <= max_size	
returns True	
Than(<) Checks if the value of variable1 is less than variable2. Syntax:	
variable1 < variable2	
Example 1:	
4 < 6	
returns True	
Example 2:	
score = 60	
l I	I

Example 2: for num in range(1, 6): if num == 3: continue print(num)			
returns True Symma: for: # Code to copeae	1	1	passing_score = 65 score < passing_score
Syntax: for: # Code to repeat if # boolean statement book repeat if # boolean statement for: in range(1, 6): if not men is range(1, 6): if not men is break print(nun) Example 1: for men in range(1, 6): if not men 3: break print(nun) Example 2: for non in range(1, 6): if not men 3: continue print(nun) NOT Returns 'True' if variable is 'False', and vice versa. Syntax: not variable Example: interked = +alse	1	1	ı
Syntax: for: # Code to repeat if # boolean statement book repeat if # boolean statement for: in range(1, 6): if not men is range(1, 6): if not men is break print(nun) Example 1: for men in range(1, 6): if not men 3: break print(nun) Example 2: for non in range(1, 6): if not men 3: continue print(nun) NOT Returns 'True' if variable is 'False', and vice versa. Syntax: not variable Example: interked = +alse	1	1	1
Syntax: for: # Code to repeat if # boolean statement book repeat if # boolean statement for: in range(1, 6): if not men is range(1, 6): if not men is break print(nun) Example 1: for men in range(1, 6): if not men 3: break print(nun) Example 2: for non in range(1, 6): if not men 3: continue print(nun) NOT Returns 'True' if variable is 'False', and vice versa. Syntax: not variable Example: interked = +alse	ĺ		1
Syntax: for: # Code to repeat if # boolean statement book repeat if # boolean statement for: in range(1, 6): if not men is range(1, 6): if not men is break print(nun) Example 1: for men in range(1, 6): if not men 3: break print(nun) Example 2: for non in range(1, 6): if not men 3: continue print(nun) NOT Returns 'True' if variable is 'False', and vice versa. Syntax: not variable Example: interked = +alse	l I	Į į	1
Syntax: for: # Code to repeat if # boolean statement book repeat if # boolean statement for: in range(1, 6): if not men is range(1, 6): if not men is break print(nun) Example 1: for men in range(1, 6): if not men 3: break print(nun) Example 2: for non in range(1, 6): if not men 3: continue print(nun) NOT Returns 'True' if variable is 'False', and vice versa. Syntax: not variable Example: interked = +alse	l I	Į į	1
Syntax: for: # Code to repeat if # boolean statement book repeat if # boolean statement for: in range(1, 6): if not men is range(1, 6): if not men is break print(nun) Example 1: for men in range(1, 6): if not men 3: break print(nun) Example 2: for non in range(1, 6): if not men 3: continue print(nun) NOT Returns 'True' if variable is 'False', and vice versa. Syntax: not variable Example: interked = +alse	[Į į	returns True
for: a code to repose if a boolean statement break for: Solds to repose for a code to repose		1	
Example 1: False Example Example Example	l I	1	for: # Code to repeat
Example 1: for num is range(1, 6): if num = 3: print(num) Example 2: for num is range(1, 6): if num = 3: print(num) Example 2: for num in range(1, 6): if num = 3: print(num) Example 2: for num in range(1, 6): if num = 3: print(num) NOT Returns 'True' if variable is 'False', and vice versa. Syntux: not variable Example: is tocked = False	[Į į	if # boolean statement break
Example 1: for one of range(1, 6): from seed; exits the loop prematurely, 'continue' skips the rest of the current iteration and moves to the next iteration. Example 2: for num in range(1, 6): if num == 3: continue print(num) continue print(num) continue print(num) continue	1		if # boolean statement
Threak' exits the loop prematurely. 'continue' skips the rest of the current iteration and moves to the next iteration. Example 2: for num in range(1, 6): if num == 3: print(num) Example 2: for num in range(1, 6): if num == 8: continue print(num) NOT Returns 'True' if variable is 'False', and vice versa. Syntax: not variable Example: islocked = False	1		continue
Threak' exits the loop prematurely. 'continue' skips the rest of the current iteration and moves to the next iteration. Example 2: for num in range(1, 6): if num == 3: print(num) Example 2: for num in range(1, 6): if num == 8: continue print(num) NOT Returns 'True' if variable is 'False', and vice versa. Syntax: not variable Example: islocked = False		Į į	1
Threak' exits the loop prematurely. 'continue' skips the rest of the current iteration and moves to the next iteration. Example 2: for num in range(1, 6): if num == 3: print(num) Example 2: for num in range(1, 6): if num == 8: continue print(num) NOT Returns 'True' if variable is 'False', and vice versa. Syntax: not variable Example: islocked = False	[Į į	i l
Threak' exits the loop prematurely. 'continue' skips the rest of the current iteration and moves to the next iteration. Example 2: for num in range(1, 6): if num == 3: print(num) Example 2: for num in range(1, 6): if num == 8: continue print(num) NOT Returns 'True' if variable is 'False', and vice versa. Syntax: not variable Example: islocked = False	1	Į į	ı
Threak' exits the loop prematurely. 'continue' skips the rest of the current iteration and moves to the next iteration. Example 2: for num in range(1, 6): if num == 3: print(num) Example 2: for num in range(1, 6): if num == 8: continue print(num) NOT Returns 'True' if variable is 'False', and vice versa. Syntax: not variable Example: islocked = False	[Į į	i l
Threak' exits the loop prematurely. 'continue' skips the rest of the current iteration and moves to the next iteration. Example 2: for num in range(1, 6): if num == 3: print(num) Example 2: for num in range(1, 6): if num == 8: continue print(num) NOT Returns 'True' if variable is 'False', and vice versa. Syntax: not variable Example: islocked = False	[Į į	i l
Loop Controls Threak' exits the loop prematurely. 'continue' skips the rest of the current iteration and moves to the next iteration. Example 2: for num in range(1, 6): if num == 3: continue print(num) NOT Returns 'True' if variable is 'False', and vice versa. Syntax: not variable Example: isLocked = False		Į į	
Loop Controls Threak' exits the loop prematurely. 'continue' skips the rest of the current iteration and moves to the next iteration. Example 2: for num in range(1, 6): if num == 3: continue print(num) NOT Returns 'True' if variable is 'False', and vice versa. Syntax: not variable Example: isLocked = False		Į į	for num in range(1, 6): if num == 3:
Example 2: for num in range(1, 6): if num == 3: continue print(num) NOT Returns 'True' if variable is 'False', and vice versa. Syntax: not variable Example Example Example Syntax: not variable Example Example	[Į į	break print(num)
Example 2: for num in range(1, 6): if num == 3: continue print(num) NOT Returns 'True' if variable is 'False', and vice versa. Syntax: not variable Example: islocked = False	Loop Controls	'break' exits the loop prematurely. 'continue' skips the rest of the current iteration and moves to the next iteration.	1
NOT Returns 'True' if variable is 'False', and vice versa. Syntax: not variable Example: islocked = False	1	the current netation and me to to at a second	ı
NOT Returns 'True' if variable is 'False', and vice versa. Syntax: not variable Example: islocked = False		Į į	1
NOT Returns 'True' if variable is 'False', and vice versa. Syntax: not variable Example: islocked = False	[Į į	i l
NOT Returns 'True' if variable is 'False', and vice versa. Syntax: not variable Example: islocked = False		Į į	1
NOT Returns 'True' if variable is 'False', and vice versa. Syntax: not variable Example: islocked = False	1		F
NOT Returns 'True' if variable is 'False', and vice versa. Syntax: not variable Example: isLocked = False		Į į	
NOT Returns 'True' if variable is 'False', and vice versa. Syntax: not variable Example: isLocked = False		Į į	if num == 3:
not variable Example: isLocked = False	1	Į į	
not variable Example: isLocked = False	1	1	1
not variable Example: isLocked = False	1	Į į	ı
not variable Example: isLocked = False	[Į į	i l
not variable Example: isLocked = False	I		ı
not variable Example: isLocked = False	[Į į	i l
not variable Example: isLocked = False	NOT	Returns 'True' if variable is 'False', and vice versa.	Syntax:
isLocked = False	I		
isLocked = False	I		ı
isLocked = False		Į į	1
isLocked = False		Į į	1
isLocked = False	1	Į į	ı
isLocked = False	1	Į į	ı
isLocked = False	[Į į	Example:
print(not isLocked)	[Į į	isLocked = False
	[Į į	print(not isLocked)
	[Į į	i l
	[Į į	i l
	I		ı
	I		ı
	1	1	1
·	ı	1	ı

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

		Grade = 12 grade == 11 or grade == 12
		returns True
		Syntax:
		range(stop)
		range(start, stop) range(start, stop) range(start, stop, step)
		range(scare, scop, scep)
range()	Generates a sequence of numbers within a specified range.	
		Example:
		<pre>range(5) #generates a sequence of integers from 0 to 4. range(2, 10) #generates a sequence of integers from 2 to 9.</pre>
		range(1, 11, 2) #generates odd integers from 1 to 9.
		Syntax:
		return value
Return Statement	'Return' is a keyword used to send a value back from a function to its caller.	Example:
	Talletion to its caller.	def add(a, b): return a + b
		result = add(3, 5)
Try-Except Block	Tries to execute the code in the try block. If an exception of	Syntax:
	the specified type occurs, the code in the except block is executed.	try: # Code that might raise an exception except
		ExceptionType: # Code to handle the exception
		Example:
		try: num = int(input("Enter a number: "))
		<pre>num = int(input("Enter a number: ")) except ValueError: print("Invalid input. Please enter a valid number.")</pre>
		print(invalid input. Please enter a valid number.")
		'

		Syntax: try: # Code that might raise an exception except ExceptionType: # Code to handle the exception else: # Code to execute if no exception occurs
Try-Except with Else Block	Code in the 'else' block is executed if no exception occurs in the try block.	<pre>try: num = int(input("Enter a number: ")) except ValueError: print("Invalid input. Please enter a valid number") else: print("You entered:", num)</pre>
		Syntax: try: # Code that might raise an exception except ExceptionType: # Code to handle the exception finally: # Code that always executes
Try-Except with Finally Block	Code in the 'finally' block always executes, regardless of whether an exception occurred.	<pre>Example: 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'.	Syntax: while condition: # Code to repeat
1		Example:

	<pre>count = 0 while count < 5: print(count) count += 1</pre>



© IBM Corporation. All rights reserved.