AND		1 statement1 and statement2 🕰 Example:
AND	Returns `True` if both statement1 and statement2 are `True`. Otherwise, returns `False`.	<pre>1 marks = 90 2 attendance_percentage = 87 3</pre>
		<pre>if marks >= 80 and attendance_percentage >= 85: print("qualify for honors") else:</pre>
		<pre>print("Not qualified for honors") # Output = qualify for honors</pre>
		Syntax: 1 class ClassName: # Class attributes and methods
Class Definition	Defines a blueprint for creating objects and defining their attributes and behaviors	Example:
Define Function	A 'function' is a reusable block of code that performs a specific task or set of tasks when called. Checks if two values are equal. 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.).	<pre>class Person: definit(self, name, age): self.name = name colf.ago = ago</pre>
		Syntax:
		1 def function_name(parameters): # Function body Example:
		1 def greet(name): print("Hello,", name)
		Syntax: 1 variable1 == variable2
Equal(==) For Loop		Example 1: 1
		returns True Example 2:
		1 age = 25 age == 30
		returns False Syntax:
		1 for variable in sequence: # Code to repeat 💪 Example 1:
		<pre>for num in range(1, 10): print(num)</pre>
		Example 2: 1 fruits = ["apple", "banana", "orange", "grape", "kiwi"]
		for fruit in fruits: print(fruit)
Function Call Greater Than or Equal To(>=)	A function call is the act of executing the code within the function using the provided arguments. Checks if the value of variable1 is greater than or equal to variable2. Checks if the value of variable1 is greater than variable2.	Syntax: 1 function_name(arguments)
		Example: 1 greet("Alice")
		Syntax:
		1 variable1 >= variable2
		1 5 >= 5 and 9 >= 5
		returns True Example 2:
		1 quantity = 105 2 minimum = 100 3 quantity >= minimum
Greater Than(>)		quantity >= minimum returns True
		Syntax: 1 variable1 > variable2
		Example 1: 9 > 6
		returns True Example 2:
		1 age = 20 2 max_age = 25 3 age > max_age
		3 age > max_age returns False
		Syntax: 1 if condition: #code block for if statement
f Statement	Executes code block `if` the condition is `True`.	Example:
		<pre>1 if temperature > 30: 2 print("It's a hot day!")</pre>
		Syntax: 1 if condition1:
	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.	<pre>2 # Code if condition1 is True 3 4 elif condition2:</pre>
		5 # Code if condition2 is True 6
		7 else: 8 # Code if no condition is True Example:
lf-Elif-Else		1 score = 85 # Example score 2 if score >= 90:
		<pre>print("You got an A!") elif score >= 80:</pre>
		<pre>print("You got a B.") else: print("You need to work harder.")</pre>
		8 9 # Output = You got a B.
		Syntax: 1 if condition: # Code, if condition is True
If-Else Statement Less Than or Equal To(<=)		2 else: # Code, if condition is False Example:
	Executes the first code block if the condition is `True`, otherwise the second block.	<pre>1 if age >= 18: 2 print("You're an adult.")</pre>
		else: print("You're not an adult yet.")
		Syntax: 1 variable1 <= variable2
		Example 1:
	Checks if the value of variable1 is less than or equal to variable2.	1 5 <= 5 and 3 <= 5
		Example 2:
		1
		returns True
Less Than(<)		Syntax: 1 variable1 < variable2
		Example 1:
	Checks if the value of variable1 is less than variable2.	1 4 < 6 returns True
	Checks if the value of variable1 is less than variable2.	Example 2: 1 score = 60
		<pre>passing_score = 65 score < passing_score</pre>
		returns True
		SVDTAY.
		Syntax: 1 for: # Code to repeat 2 if # boolean statement
		1 for: # Code to repeat
		<pre>for: # Code to repeat if # boolean statement break 4</pre>
Loop Controls	`break` exits the loop prematurely. `continue` skips the rest of the current iteration and moves to the next iteration	<pre>1 for: # Code to repeat 2 if # boolean statement 3 break 4</pre>
Loop Controls	`break` exits the loop prematurely. `continue` skips the rest of the current iteration and moves to the next iteration.	<pre>1 for: # Code to repeat 2 if # boolean statement 3 break 4</pre>
Loop Controls		<pre>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 for: # Code to repeat if # boolean statement continue Example 1:</pre>
Loop Controls		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
Loop Controls		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) Cal
Loop Controls		1
		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 for num in range(1, 6): if num == 3: continue print(num) Syntax: 1 !variable Example:
	iteration.	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 for num in range(1, 6): if num == 3: continue print(num) Syntax: 1 !variable Example:
	iteration.	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 yrint(num) Syntax: 1 !variable
	iteration.	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) for num in range(1, 6): if num == 3: continue for num in range(1, 6): if num == 3: continue print(num) for num in range(1, 6): if num == 3: continue print(num) for num in range(1, 6): if num == 3: continue print(num) for num in range(1, 6): if num == 3: continue print(num) for num in range(1, 6): if num == 3: continue print(num) for num in range(1, 6): if num == 3: continue print(num) for num in range(1, 6): if num == 3: continue print(num) for num in range(1, 6): if num == 3: continue print(num) for num in range(1, 6): if num == 3: continue print(num) for num in range(1, 6): if num == 3: continue for num in range(1, 6): if num == 3: co
NOT	Returns `True` if variable is `False`, and vice versa.	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) for num in range(1, 6): if num == 3: continue for num in range(1, 6): if num == 3: continue print(num) for num in range(1, 6): if num == 3: continue print(num) for num in range(1, 6): if num == 3: continue print(num) for num in range(1, 6): if num == 3: continue print(num) for num in range(1, 6): if num == 3: continue print(num) for num in range(1, 6): if num == 3: continue print(num) for num in range(1, 6): if num == 3: continue print(num) for num in range(1, 6): if num == 3: continue print(num) for num in range(1, 6): if num == 3: continue print(num) for num in range(1, 6): if num == 3: continue for num in range(1, 6): if num == 3: co
NOT	iteration.	for: # Code to repeat if # boolean statement break for: # Code to repeat if # boolean statement for num in range(1, 6): if num == 3: break print(num) Example 2: for num in range(1, 6): if num == 3: continue for num in range(1, 6): if num == 3: continue print(num) Syntax: 1 !variable Example: 1 !isLocked returns True if the variable is False (i.e., unlocked). Syntax: 1 variable1 != variable2 Example: Example: 1 a = 10 2 b = 20
NOT	Returns `True` if variable is `False`, and vice versa.	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) for num in range(1, 6): if num == 3: continue for num in range(1, 6): if num == 3: continue print(num) Syntax: livariable Example: lisLocked returns True if the variable is False (i.e., unlocked). Syntax: variable1!= variable2 Example: a = 10 b = 20 a a!= b returns True
NOT	Returns `True` if variable is `False`, and vice versa.	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: lvariable Example: lisLocked returns True if the variable is False (i.e., unlocked). Syntax: variable1 != variable2 Example: a = 10 b = 20 a a != b returns True Example 2:
NOT	Returns `True` if variable is `False`, and vice versa.	for: # Code to repeat if # boolean statement break for: # Code to repeat if # boolean statement continue Example 1: 1
NOT Not Equal(!=)	Returns `True` if variable is `False`, and vice versa.	1 for: # Code to repeat 2 if # boolean statement 3 break 4 5 for: # Code to repeat 6 if # boolean statement 7 continue Example 1: 1 for num in range(1, 6): 2 if num == 3: 3 break 4 print(num) Example 2: 1 for num in range(1, 6): 2 if num == 3: 3 continue 4 print(num) Syntax: 1 !variable Example: 1 listocked returns True if the variable is False (i.e., unlocked). Syntax: 1 variable1! = variable2 Example: 1 a = 10 2 b = 20 3 a != b returns True Example 2: 1 count=0 2 count != 0 returns False Syntax: 1 object_name = className(arguments) ② Example:
NOT Not Equal(!=)	Returns 'True' if variable is 'False', and vice versa. Checks if two values are not equal.	1 for: # Code to repeat 2 if # boolean statement 3 break 4 5 for: # Code to repeat 6 if # boolean statement 7 continue Example 1: 1 for num in range(1, 6): 2 if num == 3: 3 break 4 print(num) Example 2: 1 for num in range(1, 6): 2 if num == 3: 3 continue 4 print(num) Syntax: 1 !variable Example: 1 listocked returns True if the variable is False (i.e., unlocked). Syntax: 1 variable1!= variable2 Example: 1 a = 10 2 b = 20 3 a != b returns True Example 2: 1 count=0 2 count!= 0 returns False Syntax: 1 object_name = className(arguments) ② Example:
NOT Not Equal(!=)	Returns 'True' if variable is 'False', and vice versa. Checks if two values are not equal.	1
Not Equal(!=) Object Creation	Returns 'True' if variable is 'False', and vice versa. Checks if two values are not equal.	for: # Code to repeat if # boolean statement break
Not Equal(!=) Object Creation	Returns 'True' if variable is 'False', and vice versa. Checks if two values are not equal. Creates an instance of a class (object) using the class constructor.	for: # Code to repeat if # boolean statement break
Not Equal(!=) Object Creation	Returns 'True' if variable is 'False', and vice versa. Checks if two values are not equal. Creates an instance of a class (object) using the class constructor.	for: # Code to repeat if # boolean statement break
Not Equal(!=) Object Creation OR	Returns 'True' if variable is 'False', and vice versa. Checks if two values are not equal. Creates an instance of a class (object) using the class constructor. Returns 'True' if either statement1 or statement2 (or both) are 'True', Otherwise, returns 'False'.	for: # Code to repeat if # boolean statement break for: # Code to repeat if # boolean statement continue Example 1: 1 for num in range(1, 6): if num == 3: break print(num) Example 2: 1 for num in range(1, 6): if num == 3: continue print(num) Example 2: 1 for num in range(1, 6): if num == 3: continue print(num) Syntax 1 variable Example: 1 !istocked returns True if the variable is False (i.e., unlocked). Syntax 1 variable1 = variable2 Example: 1 a = 10 b = 20 a b 20 a
Not Equal(!=) Object Creation OR	Returns 'True' if variable is 'False', and vice versa. Checks if two values are not equal. Creates an instance of a class (object) using the class constructor.	for: a code to repeat if a boolean statement break for: a code to repeat if a boolean statement continue Example 1: 1 for num in range(1, 6): if num == 3: break 4 print(num) Example 2: 1 for num in range(1, 6): if num == 3: continue 4 print(num) Example: 1 Ivariable Example: 1 Istocked returns True if the variable is False (i.e., unlocked). Syntax 1 variable1 = variable2 Example: 1 a = 10 b = 20 a
Not Equal(!=) Object Creation OR	Returns 'True' if variable is 'False', and vice versa. Checks if two values are not equal. Creates an instance of a class (object) using the class constructor. Returns 'True' if either statement1 or statement2 (or both) are 'True', Otherwise, returns 'False'.	for: a code to repeat if a boolean statement break for: a code to repeat if a boolean statement continue Example 1: for num in range(1, 6): if num == 3: break print(num) Example 2: 1 for num in range(1, 6): 2 if num == 3: continue 4 print(num) Syntax: 1 Variable ② Example: 1 !istocked returns True if the variable is False (i.e., unlocked). Syntax: 1 variable1 = variable2 ② Example: 1 a = 10 2 b = 20 3 a = b returns True Example 2: 1 count=0 2 count = 0 returns False Syntax: 1 object_name = Classilame(arguments) ③ Example: 1 person1 = Person("Alice", 25) ③ Syntax: 1 statement1 statement2 ② Example: 1 "Farewell Party Invitation" Grade = 12 grade == 11 or grade == 12 returns Irue Syntax: 1 range(stop) 2 range(start, stop) 3 range(start, stop) 3 range(start, stop, step) Example: Example:
Not Equal(!=) Object Creation OR	Returns 'True' if variable is 'False', and vice versa. Checks if two values are not equal. Creates an instance of a class (object) using the class constructor. Returns 'True' if either statement1 or statement2 (or both) are 'True', Otherwise, returns 'False'.	for: # Code to repeat break for: # Code to repeat for a 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: l
Not Equal(!=) Object Creation OR	Returns 'True' if variable is 'False', and vice versa. Checks if two values are not equal. Creates an instance of a class (object) using the class constructor. Returns 'True' if either statement1 or statement2 (or both) are 'True', Otherwise, returns 'False'.	for: a Code to repeat if # boolean statement break
Not Equal(!=) Object Creation OR	Returns True if variable is False', and vice versa. Checks if two values are not equal. Creates an instance of a class (object) using the class constructor. Returns 'True' if either statement1 or statement2 (or both) are 'True'. Otherwise, returns 'Talse'. Generates a sequence of numbers within a specified range.	for: # Code to repeat if # boolean statement break for: # Code to repeat if # boolean statement for: # Code to repeat if # boolean statement continue Example 1: 1
Not Equal(!=) Object Creation OR	Returns True if variable is False', and vice versa. Checks if two values are not equal. Creates an instance of a class (object) using the class constructor. Returns 'True' if either statement1 or statement2 (or both) are 'True'. Otherwise, returns 'Talse'. Generates a sequence of numbers within a specified range.	for: # Code to repeat if # boolean statement break for: # Code to repeat if # boolean statement continue Example !: 1 for num in range(1, 0): if num == 3:
Not Equal(!=) Object Creation OR Return Statement	Returns 'True' if variable is Talse', and vice versa. Checks if two values are not equal. Creates an instance of a class (object) using the class constructor. Returns 'True' if either statement1 or statement2 (or both) are 'True'. Otherwise, returns 'Talse'. Generates a sequence of numbers within a specified range. 'Return' is a keyword used to send a value back from a function to its caller. Tries to execute the code in the try block. If an exception of the specified type occurs, the code in the	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:
Not Equal(!=) Object Creation OR Return Statement	Returns True' if variable is 'False', and vice versa. Checks if two values are not equal. Creates an instance of a class (object) using the class constructor. Returns 'True' if either statement1 or statement2 (or both) are 'True'. Otherwise, returns 'Talse'. Cenerates a sequence of numbers within a specified range.	for: # Code to repeat break for: # boolcan statement break for: # Code to repeat if # boolcan statement continue Comple: for num in range(1, 6): if num — 3: break print(num) Example 2: 1 for num in range(1, 6): if num — 3: continue Example 2: 1 for num in range(1, 6): if num — 3: continue d print(num) Syntax: 1 Ivariable Example: 1 listocked current rue if the variable's False (i.e., unlocked). Syntax: 1 variable != variable2 Example: 1 a = 10 2 b = 20 3 a i = b returns True Example 2: 1 count-0 count != 0 returns False Syntax: 1 object_name - classflame(arguments) Example: 1 persont - Person("Alice", 25) Syntax: 1 statement1 statement2 Example: 1 range(stop) 2 range(start, stop) 3 r
Not Equal(!=) Object Creation OR Return Statement	Returns 'True' if variable is Talse', and vice versa. Checks if two values are not equal. Creates an instance of a class (object) using the class constructor. Returns 'True' if either statement1 or statement2 (or both) are 'True'. Otherwise, returns 'Talse'. Generates a sequence of numbers within a specified range. 'Return' is a keyword used to send a value back from a function to its caller. Tries to execute the code in the try block. If an exception of the specified type occurs, the code in the	1
NOT Not Equal(!=) Object Creation OR Return Statement	Returns 'True' if variable is Talse', and vice versa. Checks if two values are not equal. Creates an instance of a class (object) using the class constructor. Returns 'True' if either statement1 or statement2 (or both) are 'True'. Otherwise, returns 'Talse'. Generates a sequence of numbers within a specified range. 'Return' is a keyword used to send a value back from a function to its caller. Tries to execute the code in the try block. If an exception of the specified type occurs, the code in the	for: a Code to repeat break for: a Code to repeat fir a boolean statement for: a Code to repeat fir a boolean statement continue Comple t: for num in range(1, 6): if num = 3: break
Not Equal(!=) Object Creation OR	Returns 'True' if variable is Talse', and vice versa. Checks if two values are not equal. Creates an instance of a class (object) using the class constructor. Returns 'True' if either statement1 or statement2 (or both) are 'True'. Otherwise, returns 'Talse'. Generates a sequence of numbers within a specified range. 'Return' is a keyword used to send a value back from a function to its caller. Tries to execute the code in the try block. If an exception of the specified type occurs, the code in the	for: a Code to repeat for: boolean statement break for: d Code to repeat for: man in range(1, 6): for mum in range(1, 6): break print(num) Example 2: I for num in range(1, 6): frum == 3: continue
Not Equal(!=) Object Creation OR Return Statement Try-Except Block	Returns 'True' if variable is Talse', and vice versa. Checks if two values are not equal. Creates an instance of a class (object) using the class constructor. Returns 'True' if either statement1 or statement2 (or both) are 'True'. Otherwise, returns 'Talse'. Generates a sequence of numbers within a specified range. 'Return' is a keyword used to send a value back from a function to its caller. Tries to execute the code in the try block. If an exception of the specified type occurs, the code in the	for: a Code to repeat if a boolean statement break for: a Code to repeat for: a Code to
Not Equal(!=) Object Creation OR	Returns True' if watistate is Take', and vice seess. Checks if two values are not repail. Checks if it is a leave of a class (https:() using the class constructor. Returns True' if either statement or statement2 (or both) are True'. Otherwise, returns Take'. Generales a sequence of numbers within a specified range. This to execute the code in the try block, if an exception of the specified type occurs, the code in the recorpt latest is executed.	for: # Code to repeat if # boolean statement break for: # Code to repeat if # boolean statement continue Comple: for mun in range(1, 6): if mun = 3: break print(nun) Fromple : 1 for mun in range(1, 6): if mun = 3: break print(nun) Syntax: if nun = 3: continue print(nun) Syntax: if it wariable is False (i.e., unlocked). Syntax: variable != variable2 Example: 1 a = 10 b = 20 a = 1-b returns True Fromple 2: 1 count=0 count != 0 returns False Syntax: i object_name = ClassHame(arguments) Cample: 1 person1 = Person("Allice", 72) Syntax: i object_name = ClassHame(arguments) Cample: 1 range(start, stop) range(s
Not Equal(!=) Object Creation OR Return Statement Try-Except Block	Returns True' if watistate is Take', and vice seess. Checks if two values are not repail. Checks if it is a leave of a class (https:() using the class constructor. Returns True' if either statement or statement2 (or both) are True'. Otherwise, returns Take'. Generales a sequence of numbers within a specified range. This to execute the code in the try block, if an exception of the specified type occurs, the code in the recorpt latest is executed.	for: a Code to repeat
Not Equal(!=) Object Creation OR Return Statement Try-Except Block	Returns True' if watistate is Take', and vice seess. Checks if two values are not repail. Checks if it is a leave of a class (https:() using the class constructor. Returns True' if either statement or statement2 (or both) are True'. Otherwise, returns Take'. Generales a sequence of numbers within a specified range. This to execute the code in the try block, if an exception of the specified type occurs, the code in the recorpt latest is executed.	for: # Code to repeat if # boolean statement break for: # Code to repeat continue for out in range(1, 6): if num = 7: if num = 7: if num = 7: if num = 4: continue if print(num) Syntac il twariable Loample: I listocked returns True if the variable is false (i.e., unlocked). Syntac variable != variable? Loample: I variable != variable? Cample: I count != 0 returns True Syntac i object_name = Classiane(arguments) for return True Syntac i opject_name = Classiane(arguments) Syntac i opject_name = Classiane(arguments) Syntac i opject_name = 1 or grade = 12 returns false Syntac i person! = Person("Alice", 25) Syntac i person! = Person("Alice", 25) Syntac i range(2) agenerates a sequence of integers from 2 to "ange(1, 11, 2) agenerates a sequence of integers from 2 to "ange(1, 11, 2) agenerates odd integers from 1 to 9. Syntac range(1, 11, 2) agenerates a sequence of integers from 2 to "ange(1, 11, 2) agenerates odd integers from 1 to 9. Syntac i return value Comple: Loample: Loample: 1 range(2) agenerates a sequence of integers from 2 to "ange(1, 11, 2) agenerates odd integers from 1 to 9. Syntac i return value Comple: 1 range(2) agenerates a sequence of integers from 2 to "ange(1, 11, 2) agenerates odd integers from 1 to 9. Syntac i return value Comple: 1 try: code that might raise an exception except secept valuetror: i print("Invalid input. Please enter a valid number.") i accept valuetrore: i print("Invalid input. Please enter a valid number.") i occept valuetrore: i print("Invalid input. Please enter a valid number.") i occept valuetrore: i print("Invalid input. Please enter a valid number.") i occept valuetrore: i print("Invalid input. Please enter a valid number.") i occept valuetrore: i print("Invalid input. Please enter a valid number.") i occept valuetrore: i print("Invalid input. Please enter a valid number.") i occept valuetrore: i prin
Not Equal(!=) Object Creation OR Return Statement Try-Except Block	Returns True' if watistate is Take', and vice seess. Checks if two values are not repail. Checks if it is a leave of a class (https:() using the class constructor. Returns True' if either statement or statement2 (or both) are True'. Otherwise, returns Take'. Generales a sequence of numbers within a specified range. This to execute the code in the try block, if an exception of the specified type occurs, the code in the recorpt latest is executed.	for: a Code to repeat if a boolean statement brook for: a Code to repeat for man in range(1, 6): If non = 3: brook print(num) Syntac variable % variable % variable % variable variable % frample: listocked % variable variable % frample: variable % variable variable %
Not Equal(!=) Object Creation OR Return Statement Try-Except Block	Heturns 'Ince' threshold is halse,' and vice versa. Checks if too values are not equal. Checks if it either statement or statement? for both) are 'True', Otherwise, returns 'Trabe'. Checks if too values as sequence of numbers will in a specified range. Check is a sequence of numbers will in a specified range. Returns 'is a knywood used to send a value back from a function to its caller. Check is the value take code in the try block. If an exception of the specified type occurs, the code in the except back is executed. Check in the 'elan' block is executed if no exception occurs in the try block.	for: # Code to repeat if a tooclean statement break for: # Code to repeat if a boolean statement continue Example t:
Not Equal(!=) Object Creation OR Try-Except Block Try-Except with Else Block	Returns True' if watistate is Take', and vice seess. Checks if two values are not repail. Checks if it is a leave of a class (https:() using the class constructor. Returns True' if either statement or statement2 (or both) are True'. Otherwise, returns Take'. Generales a sequence of numbers within a specified range. This to execute the code in the try block, if an exception of the specified type occurs, the code in the recorpt latest is executed.	for a Code to repeat for a First Code to repeat
NOT Not Equal(!=) Object Creation OR Try-Except Block Try-Except with Else Block	Heturns 'Ince' threshold is halse,' and vice versa. Checks if too values are not equal. Checks if it either statement or statement? for both) are 'True', Otherwise, returns 'Trabe'. Checks if too values as sequence of numbers will in a specified range. Check is a sequence of numbers will in a specified range. Returns 'is a knywood used to send a value back from a function to its caller. Check is the value take code in the try block. If an exception of the specified type occurs, the code in the except back is executed. Check in the 'elan' block is executed if no exception occurs in the try block.	for: # Code to repeat
Not Equal(!=) Object Creation OR Try-Except Block Try-Except with Else Block	Heturns 'Ince' threshold is halse,' and vice versa. Checks if too values are not equal. Checks if it either statement or statement? for both) are 'True', Otherwise, returns 'Trabe'. Checks if too values as sequence of numbers will in a specified range. Check is a sequence of numbers will in a specified range. Returns 'is a knywood used to send a value back from a function to its caller. Check is the value take code in the try block. If an exception of the specified type occurs, the code in the except back is executed. Check in the 'elan' block is executed if no exception occurs in the try block.	1 For: a Code to repeat 1 1 1 1 1 1 1 1 1
Not Equal(!=) Object Creation OR Try-Except Block Try-Except with Else Block	Heturns 'Ince' threshold is halse,' and vice versa. Checks if too values are not equal. Checks if it either statement or statement? for both) are 'True', Otherwise, returns 'Trabe'. Checks if too values as sequence of numbers will in a specified range. Check is a sequence of numbers will in a specified range. Returns 'is a knywood used to send a value back from a function to its caller. Check is the value take code in the try block. If an exception of the specified type occurs, the code in the except back is executed. Check in the 'elan' block is executed if no exception occurs in the try block.	for: # Code to repeat