Control Statements (BREAK, CONTINUE, PASS)

Python loops provide for the effective automation and repetition of processes. However, there may occasionally be a situation in which you wish to ignore the condition, skip an iteration, or totally quit the loop.

Loop control statements are one way to accomplish them. Loop control statements alter the order in which operations are carried out. All automatically produced objects inside a scope are deleted upon exiting execution. The following control statements are supported by Python:

- Break statement
- Continue statement
- Pass statement

Break statement: The break statement in Python is used to terminate the loop or statement in which it is present. After that, the control will pass to the statements that are present after the break statement, if available. If the break statement is present in the nested loop, then it terminates only those loops which contain the break statement.

syntax:	**Example:**

for / while loop:
  # statement(s)
  if condition:
  break
  # statement(s)
# loop end 	
	for i in range(1, 5):
	  for j in range(2, 6):
	  if j%i == 0:
	  break
	  print(i, " ", j)

Continue Statement in Python: Like the break statement, continue is also a loop control statement. In contrast to the break statement, the continue statement forces the execution of the loop's subsequent iteration rather than breaking it. The continue statement, as its name implies, compels the loop to carry out the subsequent iteration or continue. The code inside the loop that comes after the continue statement is skipped, and the loop's subsequent iteration starts when the continue statement is executed.

syntax:	**Example:**

for / while loop:
  # statement(s)
  if condition:
  continue
  # statement(s)
	for i in range(1, 11*):*
	  if i == 6:
	  continue
	  else:
	  print(i, end = " ")

Pass Statement in Python: Pass statement, as its name implies, does nothing. When a statement is syntactically necessary but you do not want any code or commands to run, you utilize Python's pass statement. It is similar to a null operation in that it has no effect. Empty loops can also be written using pass statements. Pass is also used for functions, classes, and empty control statements.

syntax:	**Example:**

function/ condition / loop:
  pass
	h = "csit"
	for i in h:
	  pass
	def fun():
	  pass
	fun()
	for i in h:
	  if i == 't':
	  print('Pass executed')
	  pass
	  print(i)

_____

Default Arguments

Python permits default values for function arguments. The parameter is assigned its default value if the function is called without an argument.

Python represents syntax and default values for function parameters in a distinct way. If no argument value is given during the function call, default values mean that the function argument will use that value. Using the assignment(=) operator with the format keywordname=value, the default value is assigned.
Let's use a function student to better grasp this. Two of the three arguments in the function student have default values set to them. Hence, the function student accepts two optional arguments in addition to the necessary one, firstname.
For Example:

```
def student(firstname, lastname ='Kashyap', standard ='Seventh'):
        print(firstname, lastname, 'studies in', standard, 'Standard')

student('Piyush')


student('Piyush', 'Sharma', 'eleventh')


student('Piyush', 'Mishra')
student('Piyush', 'Seventh')
```