**Loop Control Statements**

Continue Statement

-- Python Continue Statement returns the control to the beginning of the loop.

-- **Python Continue statement** is a loop control statement that forces to execute the next iteration of the loop while skipping the rest of the code inside the loop for the current iteration only, i.e. when the continue statement is executed in the loop, the code inside the loop following the continue statement will be skipped for the current iteration and the next iteration of the loop will begin.

# *Continue statement*

*for* var in "Helloo World":

*if* var == "o":

*continue*

    print(var)

# *loop from 1 to 10*

*for* i in range(1, 11):

    # *If i is equals to 6,*

    # *continue to next iteration*

    # *without printing*

*if* i == 6:

*continue*

*else*:

        # *otherwise print the value*

        # *of i*

        print(i, end=" ")

Break Statement

-- **Python break** is used to terminate the execution of the loop.

-- **break statement in python** is used to bring the control out of the loop when some external condition is triggered.

-- break statement is put inside the loop body (generally after if condition).  It terminates the current loop, i.e., the loop in which it appears, and resumes execution at the next statement immediately after the end of that loop.

-- If the break statement is inside a nested loop, the break will terminate the innermost loop.

# *Break Statement*

*for* i in range(10):

    print(i)

*if* i == 2:

*break*

# *Python program to demonstrate break statement*

s = 'Helllooo World'

# *Using for loop*

*for* letter in s:

    print(letter)

    # *break the loop as soon it sees 'e'*

    # *or 's'*

*if* letter == 'l' or letter == 'o':

*break*

print("Out of for loop"    )

print()

i = 0

# *Using while loop*

*while* True:

    print(s[i])

    # *break the loop as soon it sees 'e'*

    # *or 's'*

*if* s[i] == 'e' or s[i] == 's':

*break*

    i += 1

print("Out of while loop ")

num = 0

*for* i in range(10):

    num += 1

*if* num == 8:

*break*

    print("The num has value:", num)

print("Out of loop")

Pass Statement

-- The Python pass statement is a null statement. But the difference between pass and comment is that comment is ignored by the interpreter whereas pass is not ignored.

-- When the user does not know what code to write, So user simply places a pass at that line.

-- Sometimes, the pass is used when the user doesn’t want any code to execute. So users can simply place a pass where empty code is not allowed, like in loops, function definitions, class definitions, or in if statements. So using a pass statement user avoids this error.

-- If we do not use pass or simply enter a comment or a blank here, we will receive an **IndentationError** error message.

n = 26

if n > 26:

    # write code your here

print('Geeks')

# *Pass Statement*

def *my\_function*():

*pass*

n = 10

*for* i in range(n):

  # *pass can be used as placeholder*

  # *when code is to added later*

*pass*

a = 10

b = 20

*if*(a<b):

*pass*

*else*:

  print("b<a")