Break, continue and Pass in Python

Using loops in python automates and repeats the tasks in an efficient manner. But Sometimes there may arise a condition where you want to exit the loop completely, skip an iteration or ignore that condition. These can be done by loop control statements.

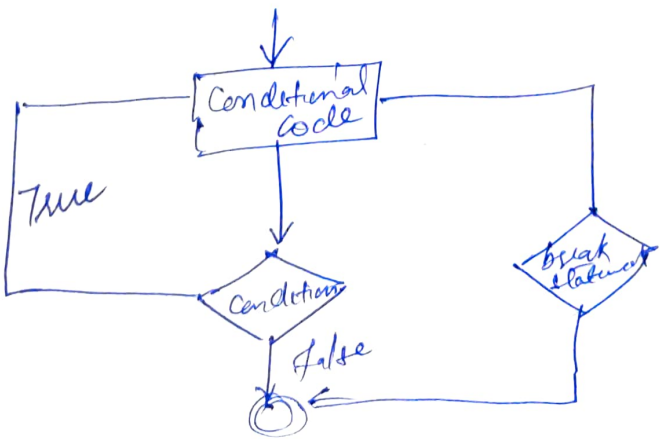
Loop control statements change execution from its normal sequence.

When execution leaves a scope, all automatic objects that were created in that scope are destroyed.

Break

The break statement 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 in nested loop

If the break statement is present in nested loop then it terminate only those loops which contains break statement.



```
s = 'geeksforgeeks'  # using for loop
for letter in s:
    print(letter)
    if letter == 'e' or letter == 's':
        break
print("Out of for loop")
print()
i = 0

# using while loop
i = 0
while True:
    print(s[i])
    if s[i] == 'e' or s[i] == 's':
        break
    i += 1
print('Out of while loop')
```

Continue Statement (Opposite of break Stat

Instead of terminating the loop, it forces to execute the next iteration of the loop.

```
for i in range(1, 11):
    if i == 6:
        continue
    else:
        print(i, end=" ")
```

O/p 1 2 3 4 5 7 8 9 10

pass Statement

It is like null operation, as nothing will happen if it is executed. Pass statement can also be used for writing empty loops. Pass is also used for empty control statement, function and classes.

```
s = "geeks"
for i in s:          # empty loop
    pass             # No error will be raised

# empty function
def fun():
```

fun() # No error will be raised

```
for i in s:
    if i == 'k':
        print('Pass executed')
        pass
        print(i)
```

O/P

```
g
e
e
pass executed
k
s
```