Python break statement

The break is a keyword in python which is used to bring the program control out of the loop. The break statement breaks the loops one by one, i.e., in the case of nested loops, it breaks the inner loop first and then proceeds to outer loops. In other words, we can say that break is used to abort the current execution of the program and the control goes to the next line after the loop.

The break is commonly used in the cases where we need to break the loop for a given condition.

The syntax of the break is given below.

```
#loop statements
break;
```

Example 1

```
list =[1,2,3,4]
count = 0;
for i in list:
    if i == 4:
        print("item matched")
        count = count + 1;
        break
print("found at",count,"location");
```

Output:

```
item matched
found at 2 location
```

Example 2

```
str = "python"
for i in str:
    if i == 'o':
        break
    print(i)
```

Output:

```
p
y
t
h
```

Example 3: break statement with while loop

```
i = 0;
while 1:
    print(i," ",end=""),
    i=i+1;
    if i == 10:
        break;
print("came out of while loop")
```

Output:

```
0  1  2  3  4  5  6  7  8  9  came out of while loop
```

Example 3

```
n=2
while 1:
    i=1;
    while i<=10:
        print("%d X %d = %d\n"%(n,i,n*i));
        i = i+1;
    choice = int(input("Do you want to continue printing the table, press 0 for no?"))
    if choice == 0:
        break;
    n=n+1
```

Output:

```
2 X 1 = 2

2 X 2 = 4

2 X 3 = 6
```

```
2 X 4 = 8

2 X 5 = 10

2 X 6 = 12

2 X 7 = 14

2 X 8 = 16

2 X 9 = 18

2 X 10 = 20

Do you want to continue printing the table, press 0 for no?1

3 X 1 = 3

3 X 2 = 6

3 X 3 = 9

3 X 4 = 12

3 X 5 = 15

3 X 6 = 18

3 X 7 = 21

3 X 8 = 24

3 X 9 = 27

3 X 10 = 30

Do you want to continue printing the table, press 0 for no?0
```

Python continue Statement

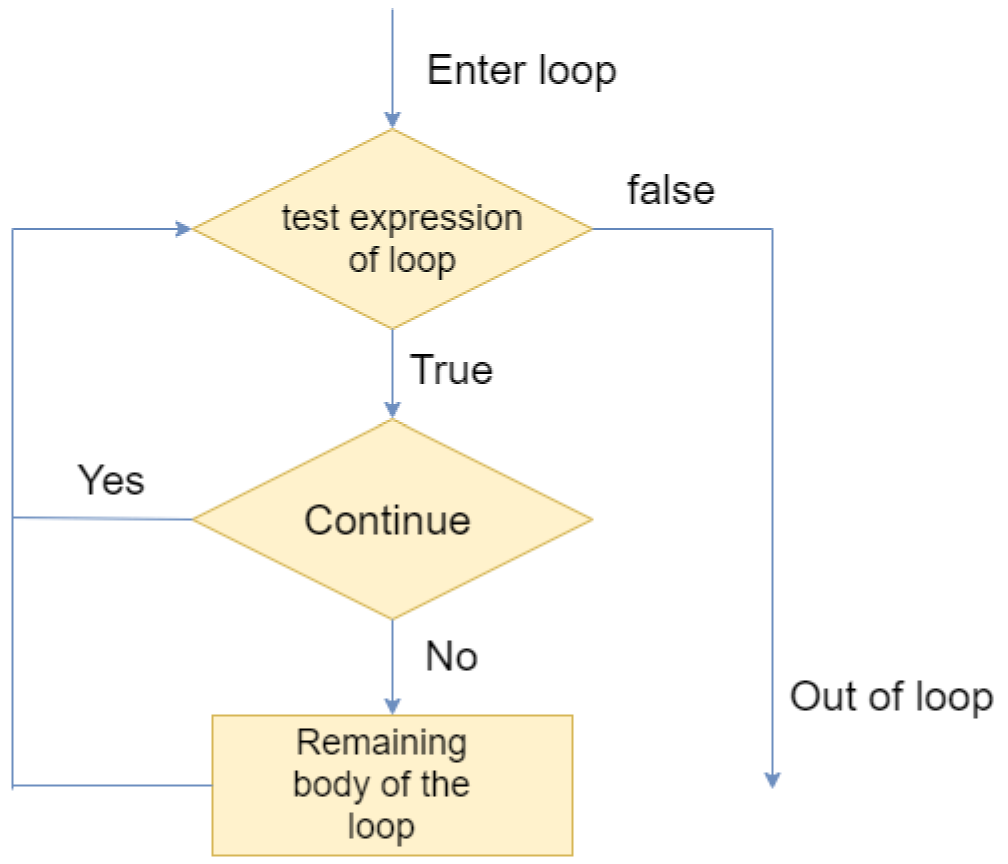
The continue statement in Python is used to bring the program control to the beginning of the loop. The continue statement skips the remaining lines of code inside the loop and start with the next iteration. It is mainly used for a particular condition inside the loop so that we can skip some specific code for a particular condition.The continue statement in Python is used to bring the program control to the beginning of the loop. The continue statement skips the remaining lines of code inside the loop and start with the next iteration. It is mainly used for a particular condition inside the loop so that we can skip some specific code for a particular condition.

Syntax

```
#loop statements
```

```
continue
#the code to be skipped
```

Flow Diagram



Consider the following examples.

Example 1

```
i = 0
while(i < 10):
   i = i+1
   if(i == 5):
      continue
   print(i)
```

Output:

```
1
```

```
2
3
4
6
7
8
9
10
```

Observe the output of above code, the value 5 is skipped because we have provided the **if condition** using with **continue statement** in while loop. When it matched with the given condition then control transferred to the beginning of the while loop and it skipped the value 5 from the code.

Let's have a look at another example:

Example 2

```
str = "Welcome"
for i in str:
    if(i == 'o'):
        continue
    print(i)
```

Pass Statement

The pass statement is a null operation since nothing happens when it is executed. It is used in the cases where a statement is syntactically needed but we don't want to use any executable statement at its place.

For example, it can be used while overriding a parent class method in the subclass but don't want to give its specific implementation in the subclass.

Pass is also used where the code will be written somewhere but not yet written in the program file. Consider the following example.

Example

```
list = [1,2,3,4,5]
flag = 0
for i in list:
    print("Current element:",i,end=" ");
    if i==3:
```

```
    pass
    print("\nWe are inside pass block\n");
    flag = 1
if flag==1:
    print("\nCame out of pass\n");
    flag=0
```

Output:

```
Current element: 1 Current element: 2 Current element: 3
We are inside pass block


Came out of pass

Current element: 4 Current element: 5
```

Python Pass

In Python, the pass keyword is used to execute nothing; it means, when we don't want to execute code, the pass can be used to execute empty. It is the same as the name refers to. It just makes the control to pass by without executing any code. If we want to bypass any code pass statement can be used.

It is beneficial when a statement is required syntactically, but we want we don't want to execute or execute it later. The difference between the comments and pass is that, comments are entirely ignored by the Python interpreter, where the pass statement is not ignored.

Suppose we have a loop, and we do not want to execute right this moment, but we will execute in the future. Here we can use the pass.

Consider the following example.

Example - Pass statement

```
# pass is just a placeholder for
# we will adde functionality later.
values = {'P', 'y', 't', 'h','o','n'}
for val in values:
    pass
```

Example - 2:

```python
for i in [1,2,3,4,5]:
    if(i==4):
        pass
        print("This is pass block",i)
    print(i)
```

Output:

```
1
2
3
This is pass block 4
4
5
```

We can create empty class or function using the pass statement.

```python
# Empty Function
def function_name(args):
    pass
#Empty Class
class Python:
    pass
```