Python break and continue

- In Python, break and continue statements can alter the flow of a normal loop.
- Loops iterate over a block of code until test expression is false, but sometimes we wish to terminate the current iteration or even the whole loop without checking test expression.
- The break and continue statements are used in these cases.

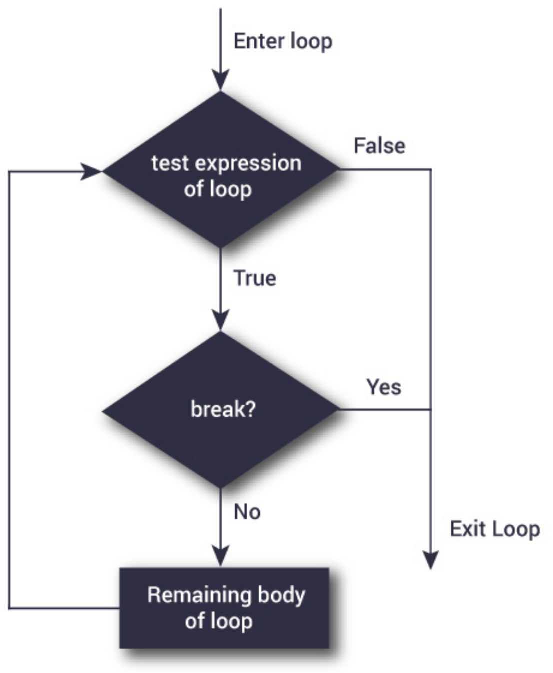
Python break statement

- The break statement terminates the loop containing it. Control of the program flows to the statement immediately after the body of the loop.
- If break statement is inside a nested loop (loop inside another loop), break will terminate the innermost loop

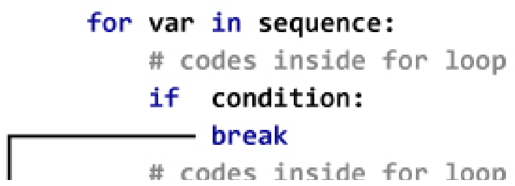
Syntax:

```
break
```

Flowchart:



The working of break statement in for loop and while loop is shown below.



```
for var in sequence:
    # codes inside for loop
    if  condition:
        break
    # codes inside for loop
```

In [1]:
```python
numbers = [1, 2, 3, 4]
for num in numbers:          #iterating ovhttps://cdn.programiz.com/sites/tuto
rial2program/files/continue-statement-flowchart.jpger list
    if num == 4:
        break
    print(num)
else:
    print("in the else-block")
print("Outside of for loop")
```

```
1
2
3
Outside of for loop
```

Python Program to check given number is Prime number or not (using break)

In [7]:
```python
num = int(input("Enter a number: "))        #convert string to int


isDivisible = False;

i=2;
while i < num:
    if num % i == 0:
        isDivisible = True;
        print ("{} is divisible by {}".format(num,i) )
        break; # this line is the only addition.
    i += 1;

if isDivisible:
    print("{} is NOT a Prime number".format(num))
else:
    print("{} is a Prime number".format(num))
```

```
Enter a number: 97
97 is a Prime number
```

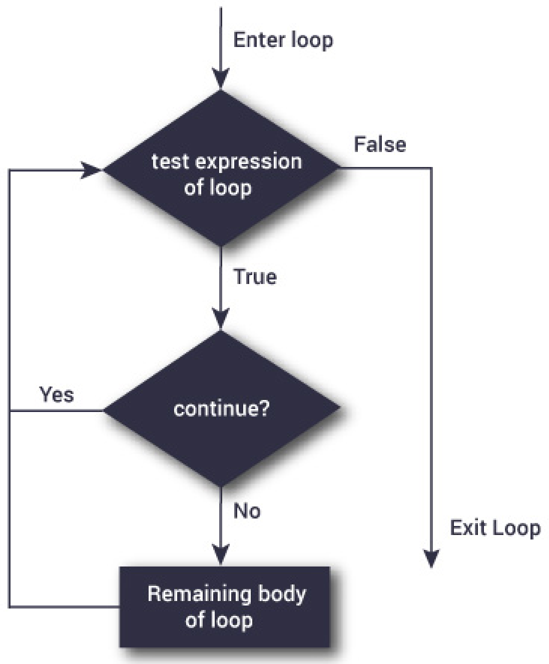
Python Continue Statement

- The continue statement is used to skip the rest of the code inside a loop for the current iteration only.
- Loop does not terminate but continues on with the next iteration.

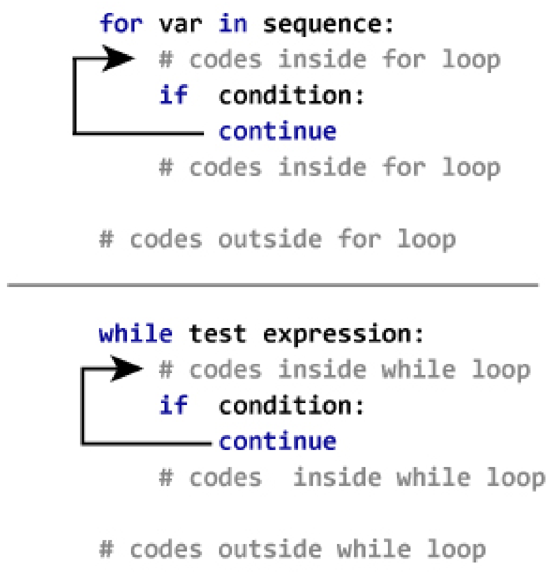
Syntax:

```
continue
```

Flowchart:



The working of continue statement in for and while loop is shown below.



```
for var in sequence:
    # codes inside for loop
    if  condition:
        continue
    # codes inside for loop

# codes outside for loop
```

```
while test expression:
    # codes inside while loop
    if  condition:
        continue
    # codes  inside while loop

# codes outside while loop
```

```
In [3]: #print odd numbers present in a list
        numbers = [1, 2, 3, 4, 5]

        for num in numbers:
            if num % 2 == 0:
                continue
            print(num)
        else:
            print("else-block")
```

```
1
3
5
else-block
```

Python pass statement

- In Python programming, pass is a null statement. The difference between a comment and pass statement in Python is that, while the interpreter ignores a comment entirely, pass is not ignored.
- However, nothing happens when pass is executed. It results into no operation (NOP).

- We generally use it as a placeholder.
- Suppose we have a loop or a function that is not implemented yet, but we want to implement it in the future. They cannot have an empty body. The interpreter would complain. So, we use the pass statement to construct a body that does nothing.

```
In [1]: # pass is just a placeholder for
        # functionality to be added later.
        sequence = {'p', 'a', 's', 's'}
        for val in sequence:
            pass
```

We can do the same thing in an empty function or class as well.

```
In [5]: def function(args):
            pass
```

```
In [6]: class example:
            pass
```