Python break and continue Statements

In Python, break and continue statements can alter the flow of a normal loop.

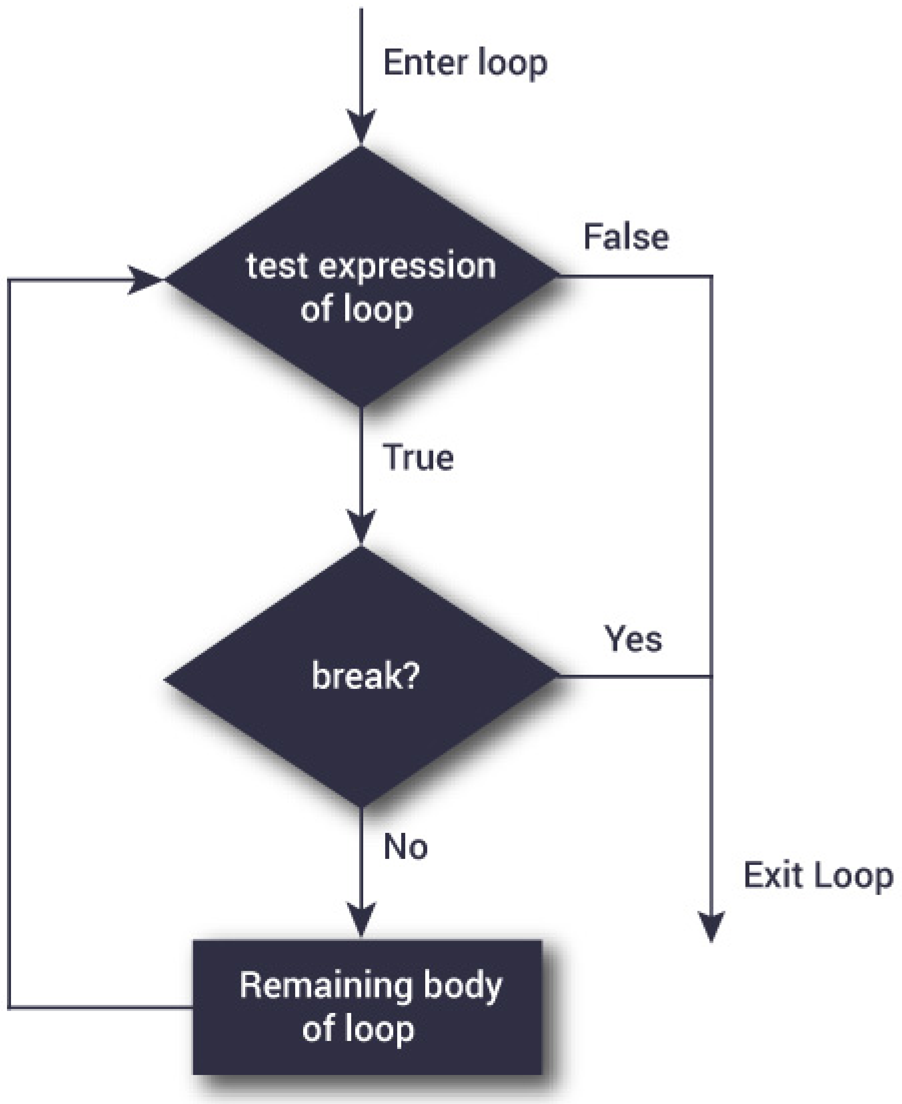
Loops iterate over a block of code until test expression is false, but sometimes we wish to terminate the current iteration or even the whole loop without cheking test expression.

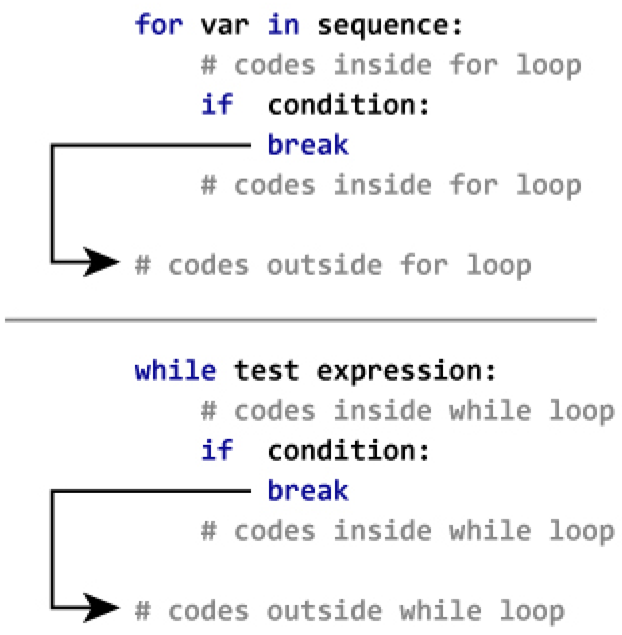
The break and continue statements are used in these cases.

Python break Statement

Syntax:

```
break
```


Enter loop
test expression of loop
False
True
break?
Yes
No
Exit Loop
Remaining body of loop



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

# codes outside for loop
```

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

# codes outside while loop
```

Example

In [1]:

```python
numbers = [1, 2, 3, 4]
for num in numbers:          #iterating over 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 [2]:

```
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))
```

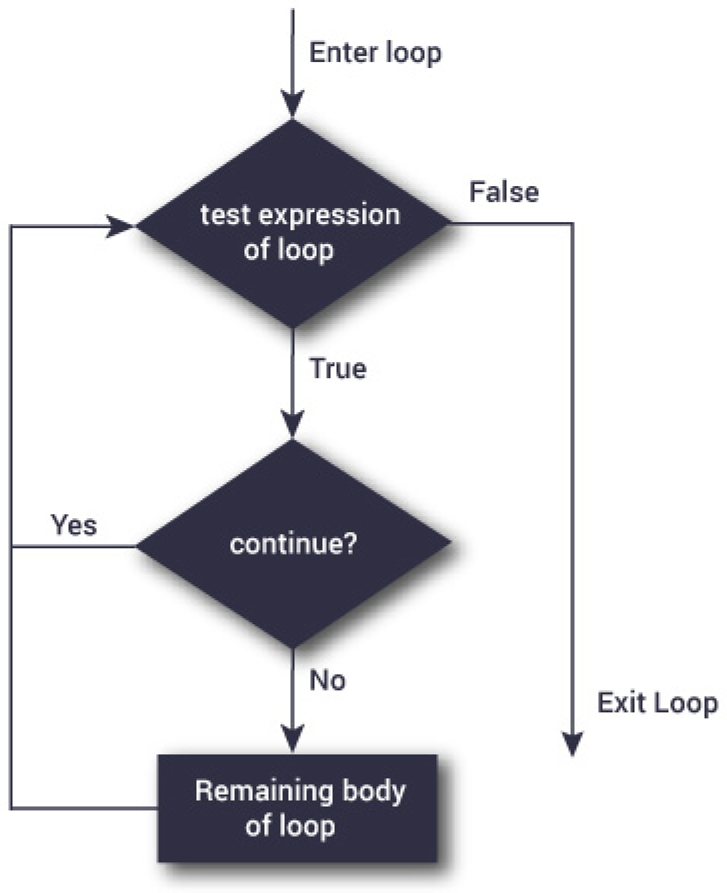
```
Enter a number: 10
10 is divisible by 2
10 is NOT a Prime number
```

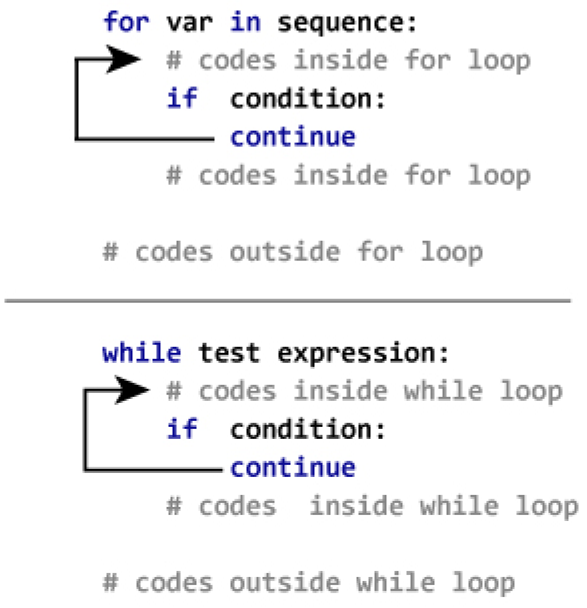
Python Continue Statement

syntax:

```
continue
```

Flow Chart





```
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
```

Example

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
```

In [ ]:

```
```