

Python break and continue Statements

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

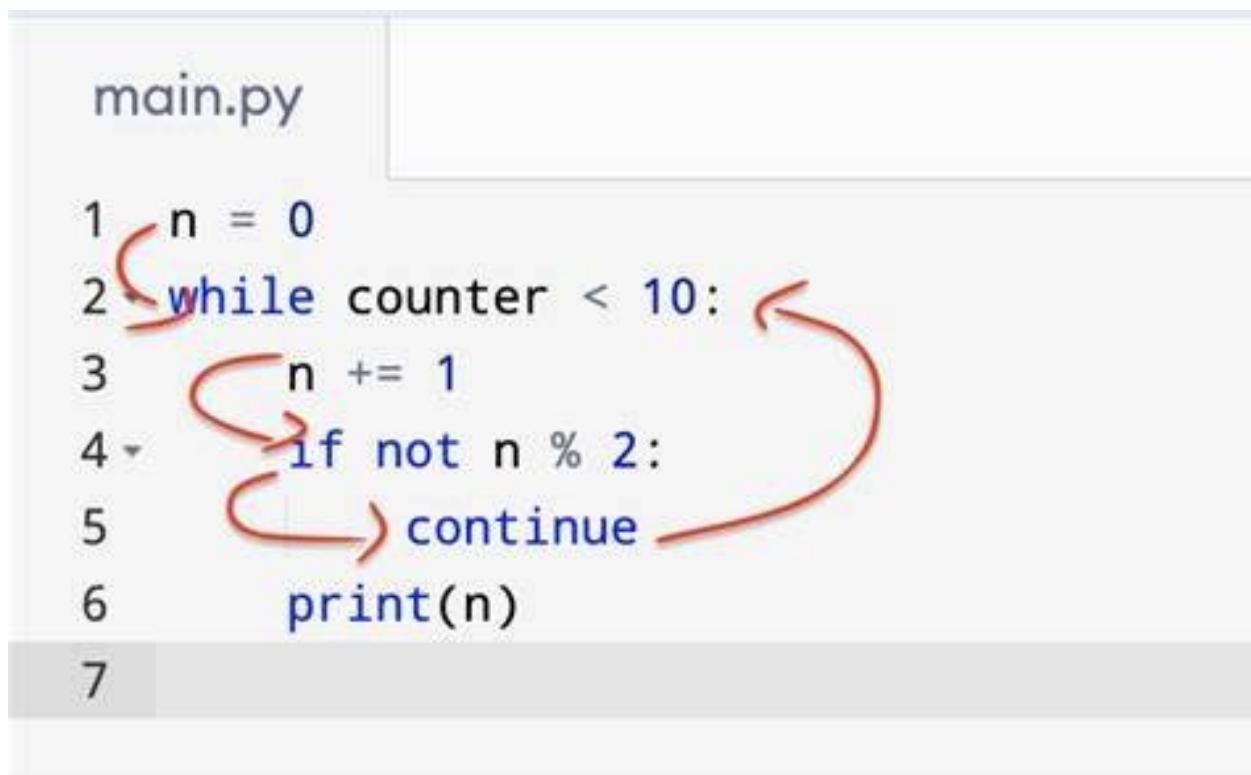
Loops iterate over a block of code until test expression is false, but sometimes we wish to terminate the current iteration or even the whole loop without checking test expression.

The break and continue statements are used in these cases.

Python break Statement

Syntax:

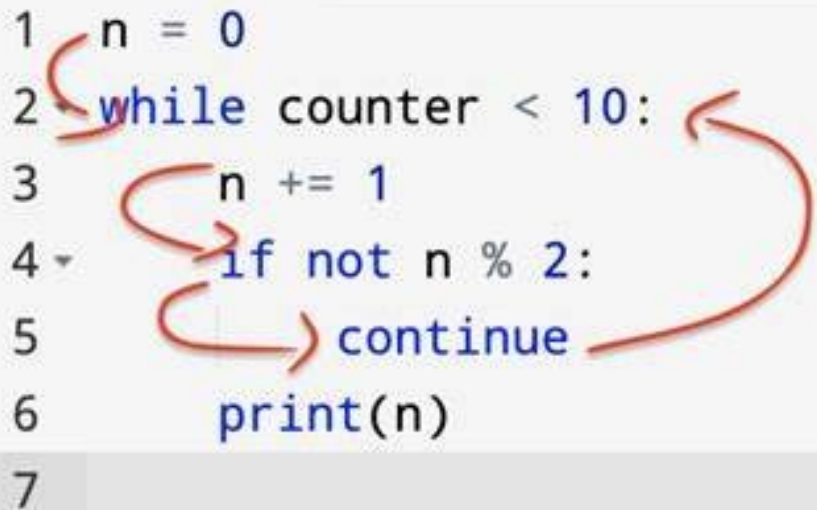
```
break
```



```
main.py
1 n = 0
2 while counter < 10:
3     n += 1
4     if not n % 2:
5         continue
6     print(n)
7
```

main.py

```
1 n = 0
2 while counter < 10:
3     n += 1
4     if not n % 2:
5         continue
6     print(n)
7
```



Example

```
numbers = [1, 2, 3, 4, 5]
for num in numbers:           #iterating over list
    if num == 4:
        continue
    print(num)
else:
    print("in the else-block")
print("Outside of for loop")
```

```
1
2
3
5
in the else-block
Outside of for loop
```

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

```
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))
```

Python Continue Statement

syntax:

```
continue
```

Flow Chart

main.py

```
1 n = 0
2 while counter < 10:
3     n += 1
4     if not n % 2:
5         continue
6     print(n)
7
```

main.py

```
1 n = 0
2 while counter < 10:
3     n += 1
4     if not n % 2:
5         continue
6     print(n)
7
```

```
#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
```