
CodroidHub Summer Training

Title:-

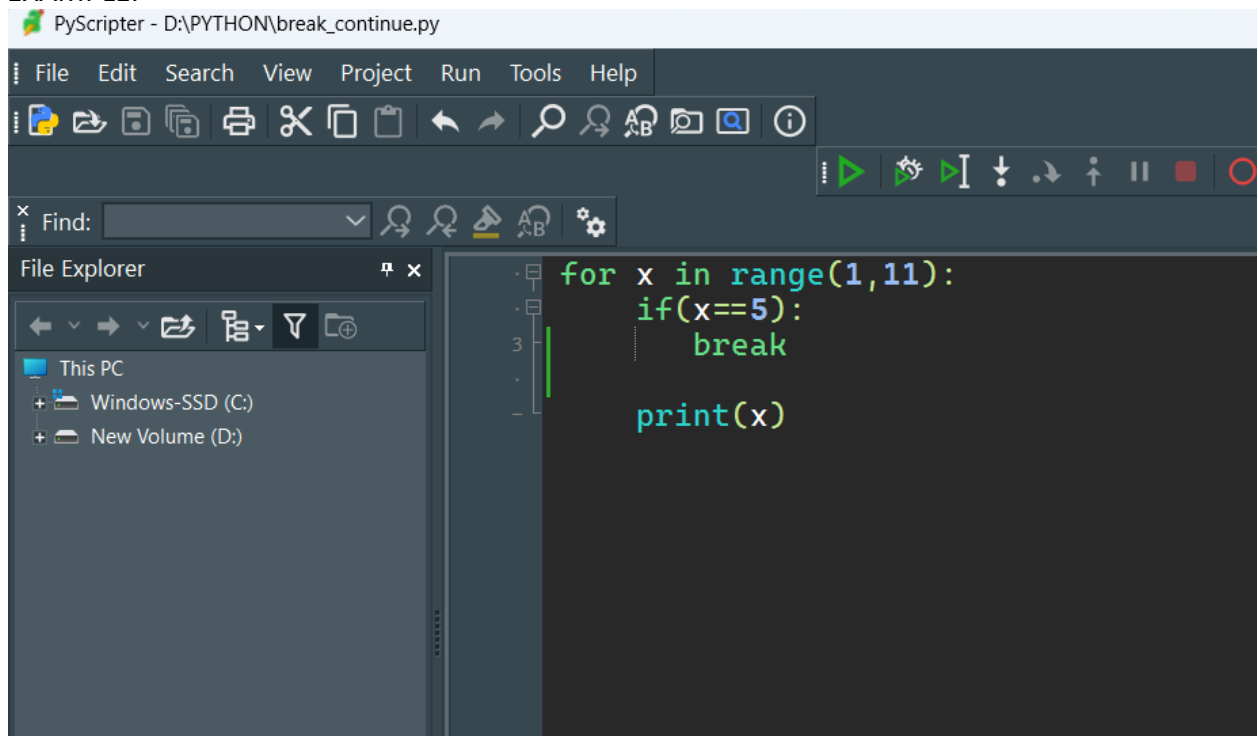
BREAK statement in Python
CONTINUE statement in Python

BREAK STATEMENT IN PYTHON

In Python, the break statement is used to immediately exit a loop when a certain condition is met. When working with nested loops, the break statement can be used to break out of both the inner and outer loops.

If a break statement is encountered in the inner loop, only the inner loop will be exited and the outer loop will continue to iterate. However, if the break statement is included in the outer loop, both the outer and inner loops will be exited and the program will continue executing after the loop.

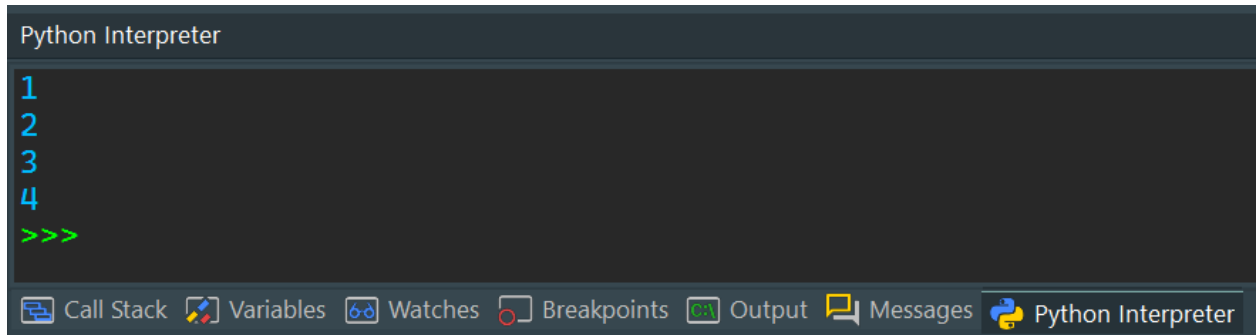
EXAMPLE:



The screenshot shows the PyScripter IDE interface. The title bar reads "PyScripter - D:\PYTHON\break_continue.py". The menu bar includes File, Edit, Search, View, Project, Run, Tools, and Help. The toolbar contains icons for file operations, editing, and execution. The File Explorer on the left shows "This PC" with drives "Windows-SSD (C:)" and "New Volume (D:)". The main editor displays the following Python code:

```
for x in range(1,11):  
    if(x==5):  
        break  
    print(x)
```

OUTPUT:



```
Python Interpreter
1
2
3
4
>>>
```

Call Stack Variables Watches Breakpoints Output Messages Python Interpreter

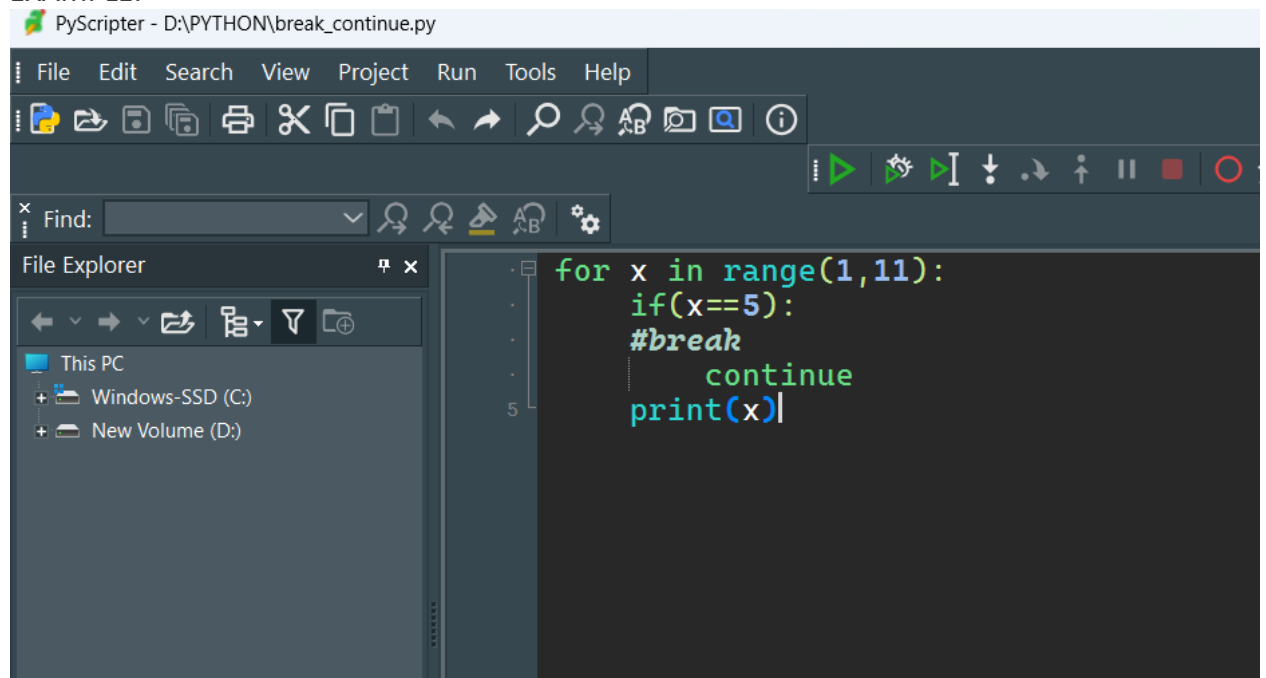
Key takeaways

- *Break is a loop control statement along with continue and pass.*
- *You can use break to exit for loops and while loops.*
- *Break only exits the innermost loop in a nested loop.*
- *You can't use break to exit an if statement unless the if statement is inside of a loop.*

Continue Statement in Python

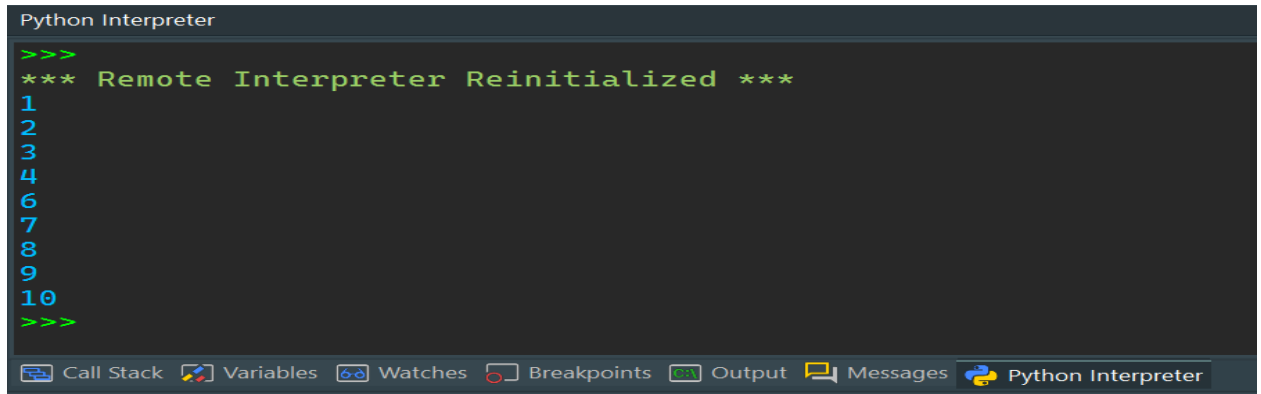
- The **continue** statement is used to skip the remaining code inside a loop for the current iteration only. **For instance, let's use continue instead of a break statement in the previous example.**

EXAMPLE:



```
PyScripter - D:\PYTHON\break_continue.py
File Edit Search View Project Run Tools Help
[Icons]
[Run, Debug, Step Through, Step Over, Step Into, Step Out, Stop]
Find:
File Explorer
This PC
+ Windows-SSD (C:)
+ New Volume (D:)
5
for x in range(1,11):
    if(x==5):
        #break
        continue
    print(x)
```

OUTPUT:



```
>>>
*** Remote Interpreter Reinitialized ***
1
2
3
4
6
7
8
9
10
>>>
```

When the condition `x == 5` becomes `True`, the `continue` statement gets executed. The remaining code in the loop is skipped only for that iteration. That's why Iteration: 5 is missing from the above output.

Therefore, the `continue` statement works opposite to the `break` statement. Instead of terminating the loop, it forces it to execute the next iteration of the loop.