

WHILE LOOPS

An Introduction to Computer Science



Let's learn about While Loops.

Another Kind of Iteration

```
while condition:  
    pass
```



FOR loops allow us to iterate through each element of a list.
They are the most common form of iteration in Python.
However, there is another kind of iteration, called While loops.

Syntax

While keyword

Conditional

Colon

```
while condition:
```

```
    pass
```

Body

4 spaces



Notice how the syntax for a WHILE loop is similar to that of an IF statement. We have the word WHILE, followed by a condition, ending with a colon. Then we can have any number of statements inside the body. Remember that this body, like a FOR loop or an IF statement, is indented with 4 spaces.

Like IF Statements

```
count_down = 4
while count_down > 0:
    count_down -= 1
    print(count_down)
print("Reached the end")
```

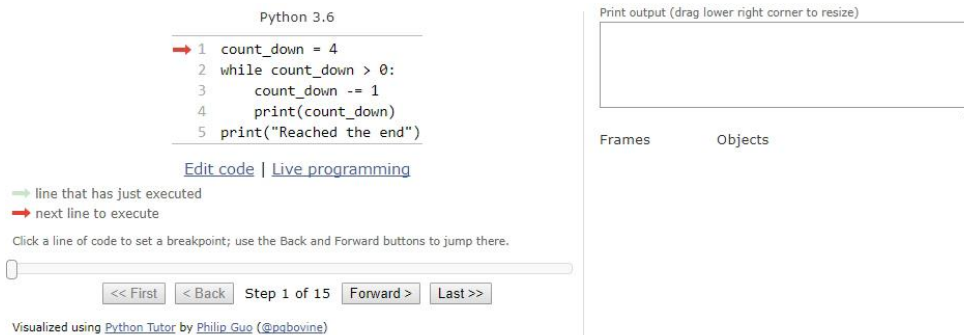


WHILE statements are actually very similar to IF statements.

However, instead of executing once or none, they will execute repeatedly until the condition is false.

Note that a loop won't end until the condition evaluates to false – even if the condition would be false during the loop body, the loop can't end until the end of the body.

Tracing the Flow



The screenshot displays the Python Tutor interface for Python 3.6. The code being executed is a while loop that counts down from 4 to 0. The first line, `count_down = 4`, is highlighted with a red arrow, indicating it is the next line to execute. The subsequent lines are `while count_down > 0:`, `count_down -= 1`, `print(count_down)`, and `print("Reached the end")`. Below the code, a legend explains the symbols: a green arrow for the line just executed and a red arrow for the next line to execute. A progress bar shows the current step as 'Step 1 of 15'. Navigation buttons include '<< First', '< Back', 'Forward >', and 'Last >>'. The output area on the right is empty, and the bottom status bar indicates the visualization is using Python Tutor by Philip Guo (@pgbovine).

```
Python 3.6
1 count_down = 4
2 while count_down > 0:
3     count_down -= 1
4     print(count_down)
5 print("Reached the end")
```

[Edit code](#) | [Live programming](#)

→ line that has just executed
→ next line to execute

Click a line of code to set a breakpoint; use the Back and Forward buttons to jump there.

<< First < Back Step 1 of 15 Forward > Last >>

Visualized using [Python Tutor](#) by [Philip Guo](#) (@pgbovine)

When a While loop is executed, the program follows a similar looping behavior as a FOR loop.

Here we can see that behavior in a WHILE loop to count down from 10.

The program tests the condition, and if it is true, then it moves through each statement in the body.

Then, it returns to the top and tests the condition again.

If it is still, true, it repeats the previous loop. Otherwise, it skips to the end of the WHILE body.

Looping Forever

```
count_down = 4

while count_down > 0:
    count_down += 1
    print(count_down)

print("Reached the end")
```

Increases the variable, so it never reaches 0!



A tricky thing about WHILE loops is that it is easy to loop forever by accident.

Consider the code below.

Because the variable "count_down" is only ever increasing, it will never reach zero and the loop never ends.

Unnecessary While Loops

WHILE loop

```
index = 0
while index < 10:
    index += 1
    print(index)
```

FOR loop

```
for index in range(10):
    print(index)
```



A major advantage of For Loops instead of While Loops is that they terminate based on a list, so they are very unlikely to loop forever.

When possible, you should probably use a FOR loop instead of a WHILE loop.

As an example, consider the counting code shown before.

Instead, that can be efficiently replaced with a FOR loop that calls the "range" function, which consumes a integers and produces a list of integers counting up to that number.

User Input Loop

```
command = ""  
while command != "EXIT":  
    command = input("Input a word, or write EXIT:")  
    print("You wrote", command)  
  
print("No more words!")
```



One of the very few cases where a WHILE loop is more useful than a FOR loop is dealing with repeated user input.

For instance, the Read-Evaluate-Print-Loop that powers a command line uses a While loop. The pattern shown here is one way to handle repeated user input.

This repeatedly asks for words until the string "EXIT" is given, at which point it will exit the loop.