

Python Loops

Review the topic of loops in Python with this e-book. Check out the table of contents to navigate to each topic.

Contents

- 1: Python Loops
- 2: The 'for' loop
- 4: The 'range' function
- 5: The 'break' and 'continue' statements
- 6: The 'pass' statement
- 7: The 'else' keyword in loops
- 8: Nested loops

Loops

Used to run blocks of code multiple times.

There are two kinds of loops in Python: while loops and for loops.

The while loop

Runs until the given conditional evaluates to **False**.

```
while <condition>:  
    <block of code>
```

Extra Resources

- 1. [What is a loop? Why do we need loops?](#)
- 2. [While loops](#)

Examples



```
counter = 10
while counter > 0:
    print(f"{counter} seconds left until New Year!")
    counter -= 1
print("Happy New Year!")
```

In the next example, the counter never changes, causing the loop to run until we stop it manually.



```
counter = 10
while counter > 0:
    print(f"{counter} seconds left until New Year!")
```

The for loop

Runs a specific number of times.

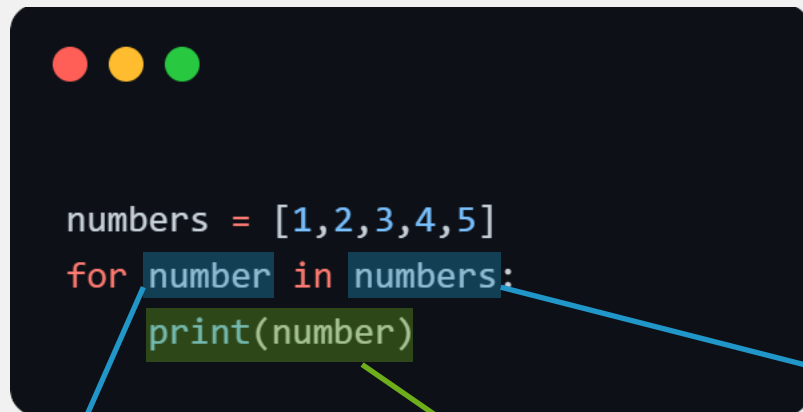
Often used to go over all the elements of an iterable.

An iterable is an object that can be iterated over e.g. a list.



```
for <variable> in <iterable>:
    <block_of_code>
```

Example



```
numbers = [1,2,3,4,5]
for number in numbers:
    print(number)
```

Variable used to iterate
over an iterable

An iterable that is being iterated over

The block of code that will be
executed until the whole iterable
has been iterated over

Extra Resources

1. [Defining a for loop](#)

The range function

Returns a sequence of integers in a given range.

Example

```
for i in range(start, stop, step):  
    print(i)
```

Specifies the increment between two integers in a sequence (optional)

Defines a start of the sequence (optional)

Defines the end of the sequence (required)

We are going to analyze different outcomes of the example code in the table below it.

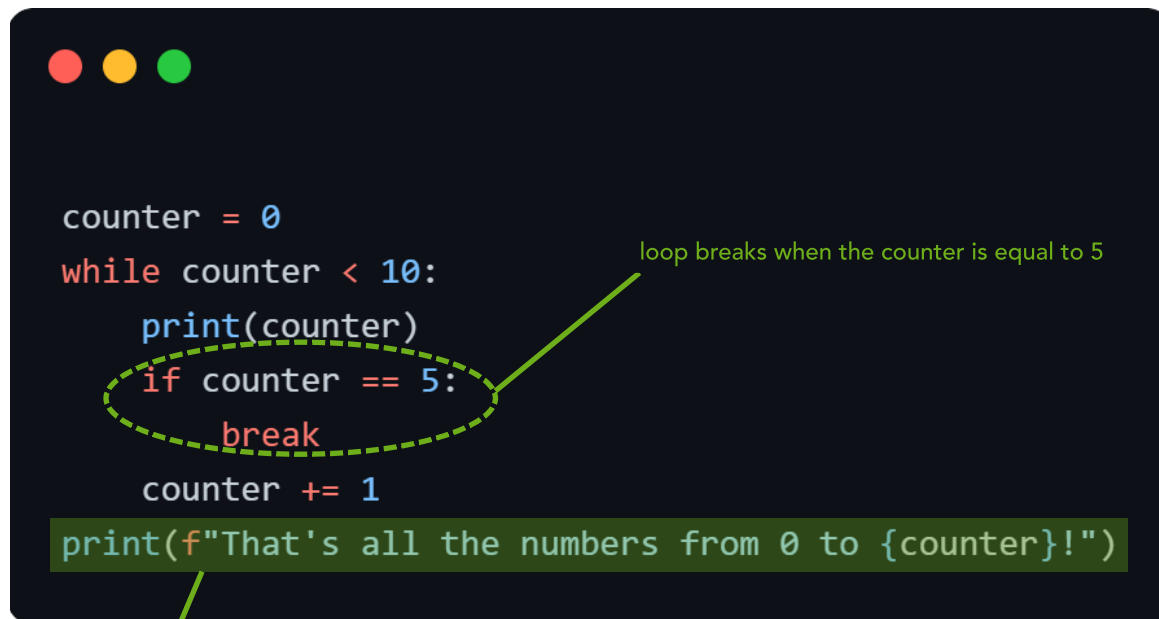
Function used	Result	Explanation
<code>range(0, 5)</code>	0, 1, 2, 3, 4	Start at 0 end at stop-1. Default step is 1.
<code>range(5)</code>	0, 1, 2, 3, 4	Default start is 0 and default step is 1.
<code>range(1, 6, step=2)</code>	1, 3, 5	Increment between integers is step = 2

Extra Resources

1. [The range function. Using range in for loops.](#)

The break statement

Used to stop the loop from iterating.

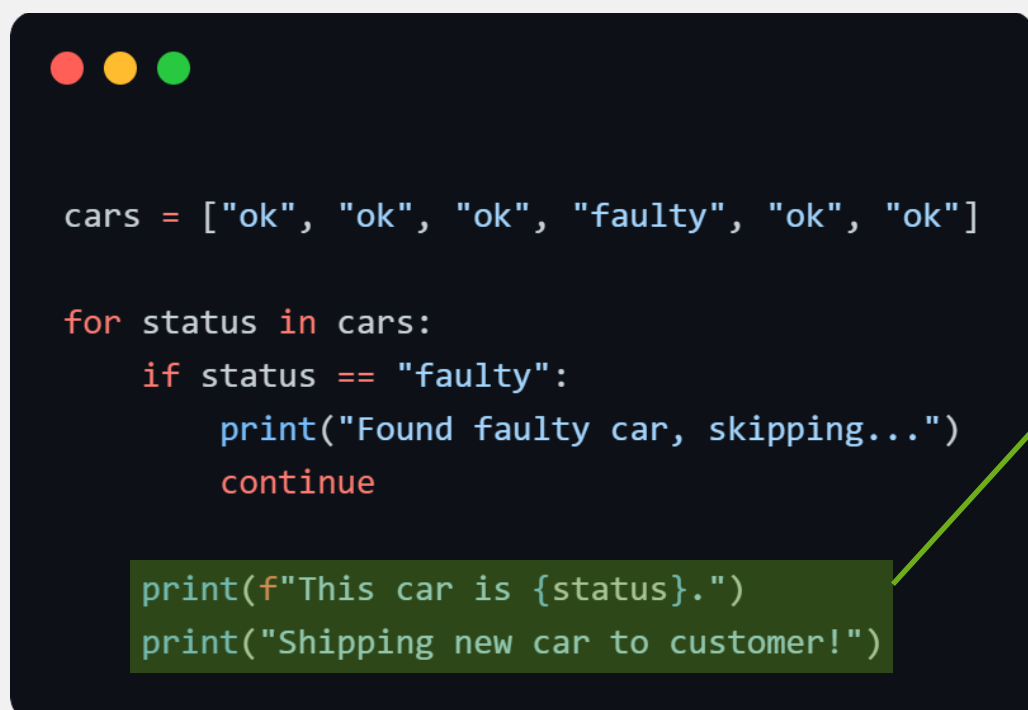


```
counter = 0
while counter < 10:
    print(counter)
    if counter == 5:
        break
    counter += 1
print(f"That's all the numbers from 0 to {counter}!")
```

After the break, this line is executed

The continue statement

Terminates the current iteration and moves onto the next one.



```
cars = ["ok", "ok", "ok", "faulty", "ok", "ok"]

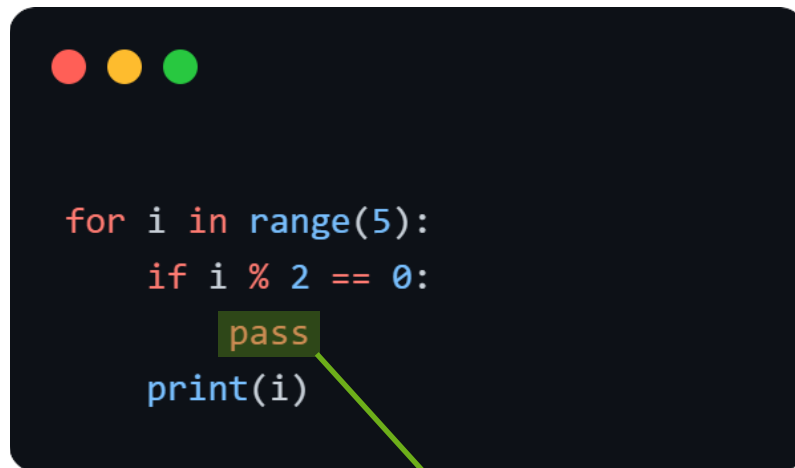
for status in cars:
    if status == "faulty":
        print("Found faulty car, skipping...")
        continue
    print(f"This car is {status}.")
    print("Shipping new car to customer!")
```

This part of the code will be skipped when the status is "faulty"

The pass statement

Used to skip over a part of the code while executing the program.

Usually used as a placeholder at a place where some code is required from the syntax side, but the code doesn't exist yet.



```
for i in range(5):  
    if i % 2 == 0:  
        pass  
    print(i)
```

This if statement doesn't do anything yet

Extra Resources

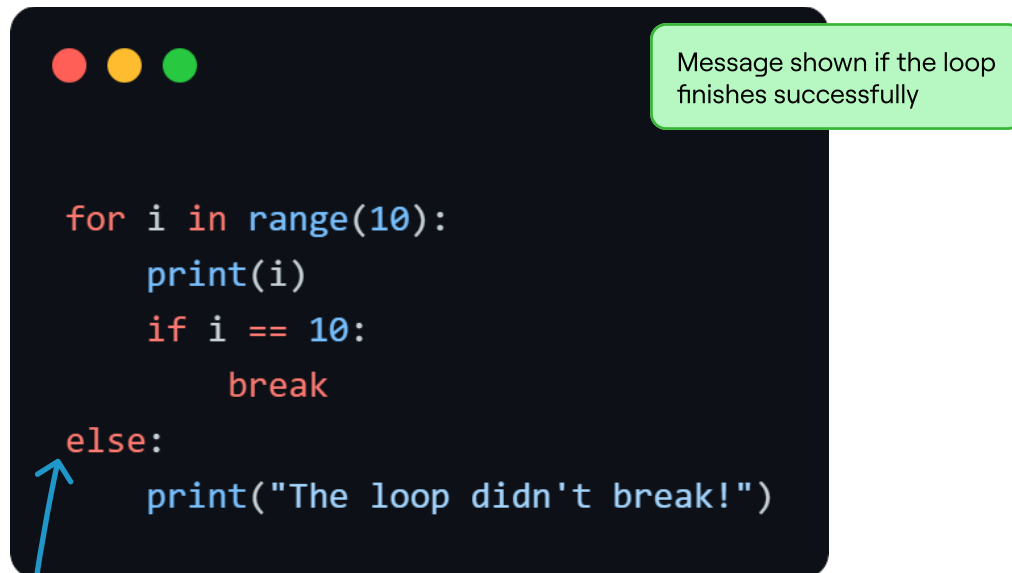
1. [The break statement](#)
2. [The continue keyword](#)

The else keyword and for loop

The `else` keyword can be used with `for` loops to check if the loop has finished running without breaking.

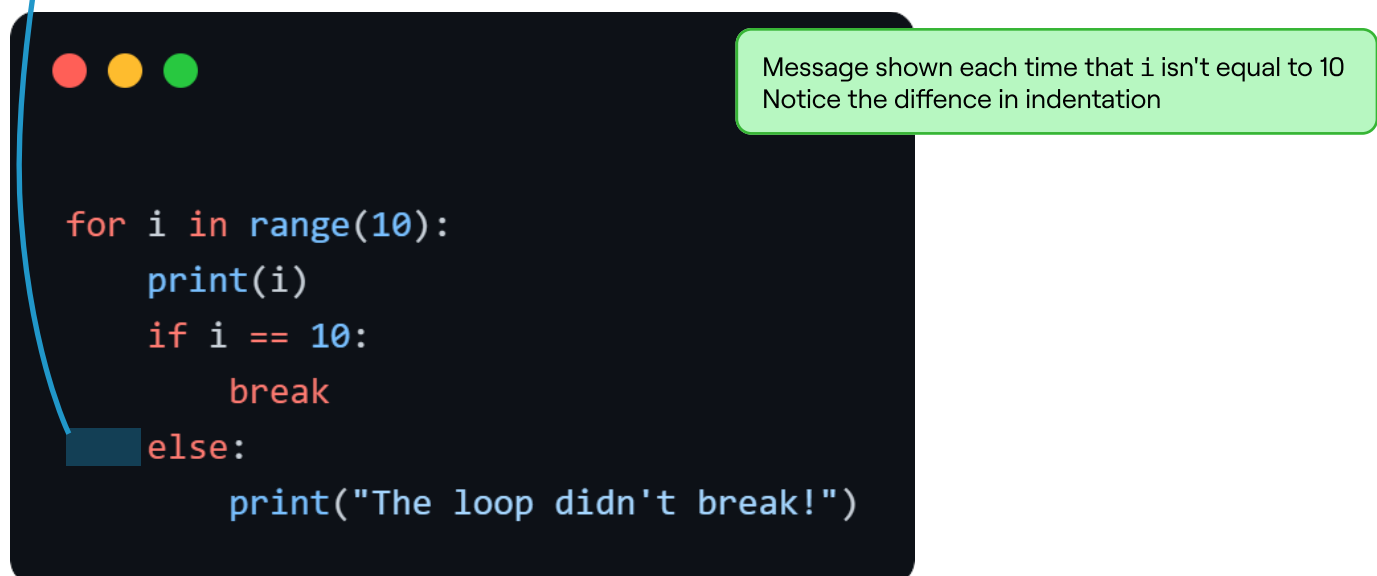
The `else` keyword should be indented by the same number of spaces as `for` loop.

Examples



```
for i in range(10):  
    print(i)  
    if i == 10:  
        break  
else:  
    print("The loop didn't break!")
```

Message shown if the loop finishes successfully



```
for i in range(10):  
    print(i)  
    if i == 10:  
        break  
else:  
    print("The loop didn't break!")
```

Message shown each time that i isn't equal to 10
Notice the difference in indentation

Nested loops

Loops can be used one inside another, regardless of their type.

Example

Creating a matrix of given dimensions using a nested loop.

```
rows = 3
columns = 2
matrix = []
for i in range(rows):
    row = []
    for j in range(columns):
        row.append(f"{i}-{j}")
    matrix.append(row)
print(matrix)
```

Output

```
[
    ['0-0', '0-1'],
    ['1-0', '1-1'],
    ['2-0', '2-1']
]
```