

FOR LOOPS

An Introduction to Computer Science



Let's learn about For Loops.

Doing Things Multiple Times

```
costs = [100, 25, 25, 50, 10]
```

```
print(costs[0])
```

```
print(costs[1])
```

```
print(costs[2])
```

```
print(costs[3])
```

```
print(costs[4])
```



```
for a_cost in costs:  
    print(a_cost)
```

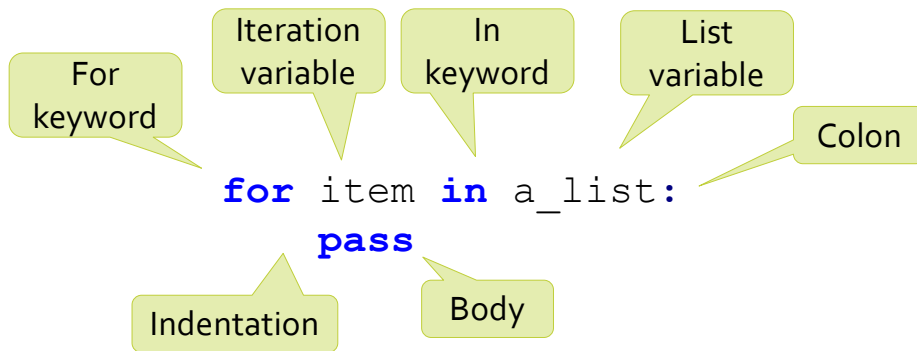
Observe the code below, which prints out the following 5 numbers.

We had to write a print statement 5 times, once for each number.

Surely there's a way that will let us stop writing the same chunk of code each time?

For Loops are an amazingly useful tool that let us perform an action on each element of a list.

For Loop Syntax



Here's how we write a For loop.

First, we write the word 'for'.

Then we make a new variable named the iteration variable.

We'll come back to this later.

Next, we write the word 'in'.

Then, we put the list variable or value that we want to iterate over.

After that, we need to put a colon.

Finally, we put any statements to run inside the body of the loop.

The Iteration List

```
for item in a_list:  
    pass
```

Each element in turn
is assigned to the
iteration variable

10	10	20	1	1
----	----	----	---	---

A crucial element of any For loop is the iteration list.

This is the data that we want to iterate over.

When the loop executes, each item of the list will be assigned to the iteration variable in turn.

It doesn't matter if the list has no items, ten items, or a thousand items; Python will still process each one in turn.

The Iteration Variable

```
for a_temperature in temperatures:
```

```
for a_book in books:
```

```
for a_name in names:
```

```
for a_fruit in fruits:
```

One of the hardest things to understand about For loops is the iteration variable.

The iteration variable represents the generalized version of each item in the list.

If you have a list of temperatures, it is "a temperature".

If you have a list of books, it is "a book".

By performing operations on this generalized version of a list item, you can work on the entire list while only having to think about one element at a time.

The iteration variable is created by the For loop when it executes.

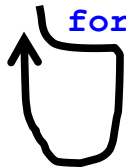
The type of the iteration variable is the same type as the element type of the list.

The Body

```
for a_price in prices:  
    ↓ adjusted = a_price * .9  
    print (adjusted)
```

Just like an If statement, we can put statements inside of a For loop. Each statement is indented with four spaces, and each statement will be executed one-by-one, from top to bottom.

The Flow of a For Loop



```
for a_price in prices:  
    adjusted = a_price * .9  
    print(adjusted)
```

Previously, we said that a program was like a river, running from the top to the bottom. If statements made the stream split into two directions. For loops also make the stream split, but one of the new streams will move back up. This is the crucial idea, and its why we call it a loop. The program will LOOP until each element of the list has been assigned to the iteration variable.

The Scope of a For Loop

```
numbers = [1, 5, 2]  
message = "The number is"
```

```
for number in numbers:  
    print(message, number)
```

```
print("The last number was", number)
```

All three
variables are
available inside!

... And afterwards!

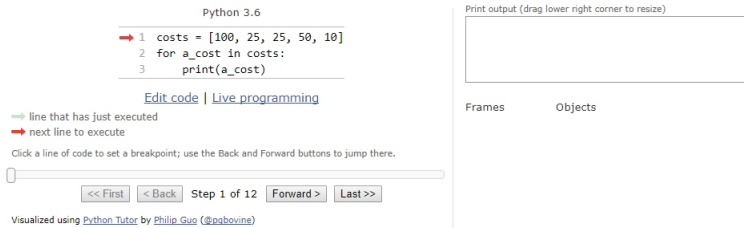
Unlike functions, For loops do not have their own scope.

Variables defined outside of a For loop can be used inside the loop body.

The iteration variable can be used after the loop has finished.

Nothing special has to be done with regards to scope with loops.

Tracing a For Loop



Let's look at a loop in practice.

Notice how the value of the iteration variable changes each time we go through the loop again, and the cursor jumps back up to the start of the loop.

This behavior is crucial!

(You can view this yourself: <https://goo.gl/yebYBw>)