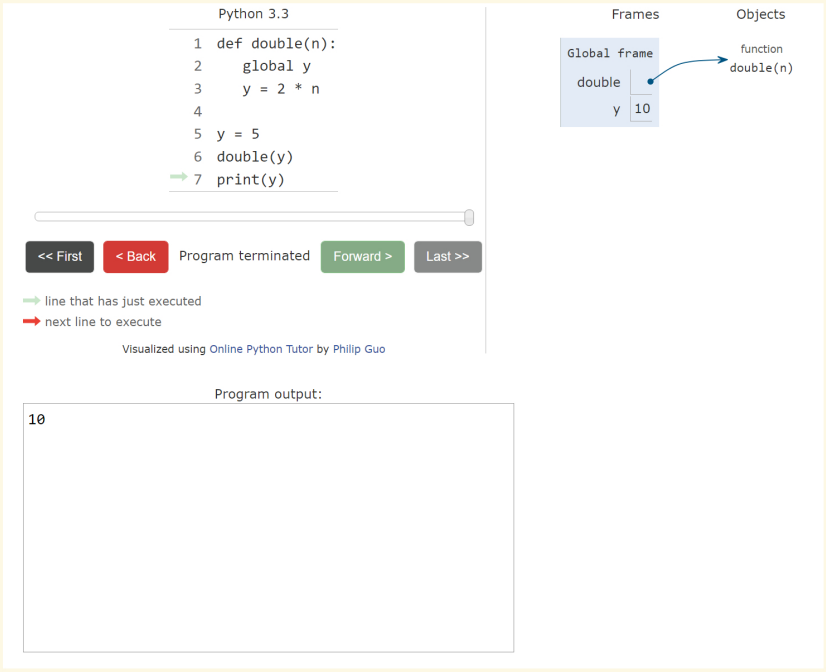
12.14. Side Effects

We say that the function `changeit` has a **side effect** on the list object that is passed to it. Global variables are another way to have side effects. For example, similar to examples you have seen above, we could make `double` have a side effect on the global variable y.

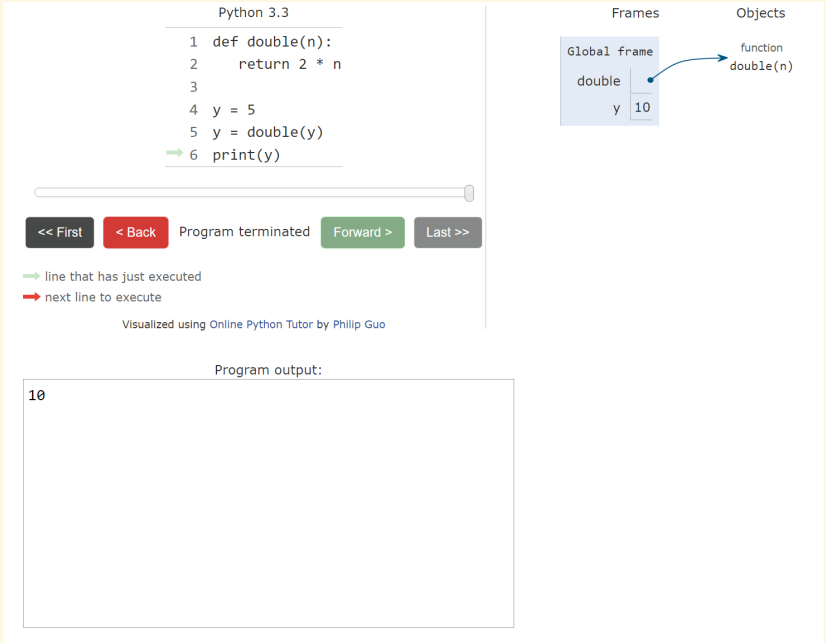


Activity: 1 -- CodeLens: (clens11_13_1)

Side effects are sometimes convenient. For example, it may be convenient to have a single dictionary that accumulates information, and pass it around to various functions that might add to it or modify it.

However, programs that have side effects can be very difficult to debug. When an object has a value that is not what you expected, it can be difficult to track down exactly where in the code it was set. Wherever it is practical to do so, it is best to avoid side effects. The way to avoid using side effects is to use return values instead.

Instead of modifying a global variable inside a function, pass the global variable's value in as a parameter, and set that global variable to be equal to a value returned from the function. For example, the following is a better version of the code above.



Activity: 2 -- CodeLens: (clens11_13_2)

You can use the same coding pattern to avoid confusing side effects with sharing of mutable objects. To do that, explicitly make a copy of an object and pass the copy in to the function. Then return the modified copy and reassign it to the original variable if you want to save the changes. The built-in `list` function, which takes a sequence as a parameter and returns a new list, works to copy an existing list. For dictionaries, you can similarly call the `dict` function, passing in a dictionary to get a copy of the dictionary back as a return value.

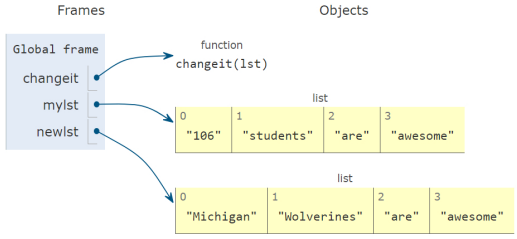
Python 3.3

```
1 def changeit(lst):
2     lst[0] = "Michigan"
3     lst[1] = "Wolverines"
4     return lst
5
6 mylst = ['106', 'students', 'are', 'awesome']
7 newlst = changeit(list(mylst))
8 print(mylst)
9 print(newlst)
```

<< First | < Back | Program terminated | Forward > | Last >>

line that has just executed
next line to execute

Visualized using Online Python Tutor by Philip Guo



Program output:

```
['106', 'students', 'are', 'awesome']
['Michigan', 'Wolverines', 'are', 'awesome']
```

Activity: 3 -- CodeLens: (clens11_13_3)

In general, any lasting effect that occurs in a function, not through its return value, is called a side effect. There are three ways to have side effects:

- Printing out a value. This doesn't change any objects or variable bindings, but it does have a potential lasting effect outside the function execution, because a person might see the output and be influenced by it.
- Changing the value of a mutable object.
- Changing the binding of a global variable.

You have attempted 4 of 3 activities on this page

12.13. Passing Mutable Objects">
'assing Mutable Objects">
12.15. Glossary">

| Back to top