Module 6: Function

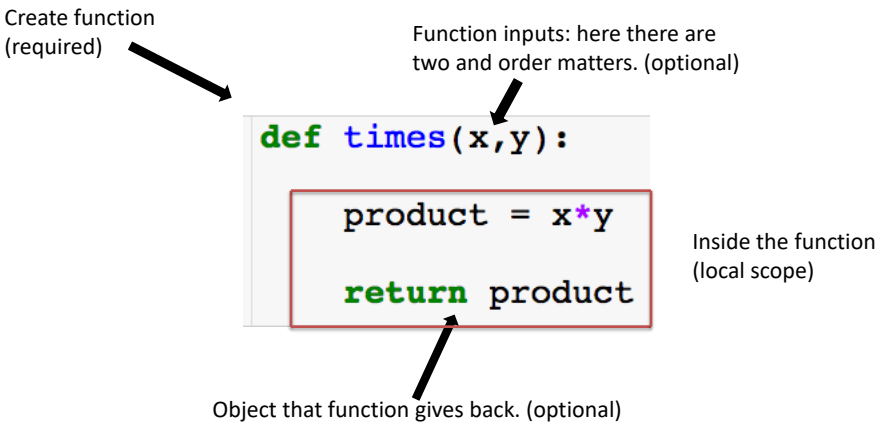
1

Summary

- Functions:
  - Def
  - Return
  - execution
- Scope:
  - Local scope vs global scope
- Sequence:
  - Putting functions together
  - Nested functions
  - Multiple returns

2

Functions



The code inside the function is never executed unless you call the function!

3

Scope

- Python scopes are the places where variables are defined and looked up.
- Local Scope
  - Variables created with a function, i.e., inside of a def.
  - Variables inside of a def will not clash with variables outside even if they have the same name.
- Global Scope
  - Variable created outside of a function.

When outside of a function, python **only sees variables in the global scope**.

4

Scope

- Python scopes are the places where variables are defined and looked up.
- Local Scope
  - Variables created with a function, i.e., inside of a def.
  - Variables inside of a def will not clash with variables outside even if they have the same name.
- Global Scope
  - Variable created outside of a function.

When inside of a function, python **first searches the local scope and then searches the global.**

5

Stringing Functions Together

```
def Add(x,y):

    return x+y
```

```
def Check_Mult_Two(x):

    if x %2==0:

        print("Yes, %d is a mulitple of 2" %x)

    else:

        print("No, %d is not mulitple of 2" %x)
```

```
num_one = 10
num_two = 15

sum_one_two = Add(num_one,num_two)
Check_Mult_Two(sum_one_two)
```

```
No, 25 is not mulitple of 2
```

6

Nested Functions

```
def Count_Vowels(name):
    total=0
    vowels = ['a','e','i','o','u']
    for i in name.lower():
        if i in vowels:
            total+=1
    return total
```

Count number of vowels in name

```
def Percent_Vowels(name):

    num_vowels = Count_Vowels(name)
    percent = num_vowels/len(name)
    return percent
```

Computes percentage of vowels in name

```
Percent_Vowels("jake")
```

7

Back to Scope

```
def ChangeElement(Q):
    Q[0]=4
    return Q

L=[1,2,3]
print(L)
M = ChangeElement(L)
print(L)
```

```
[1, 2, 3]
[4, 2, 3]
```

Local scope:

$Q = [4, 2, 3]$

Global scope:

$L = [4, 3, 2]$
$M = [4, 2, 3]$

8

Back to Scope

```
def ChangeElement(Q):
    Q[0]=4
    return Q

L=[1,2,3]
print(L)
M = ChangeElement(L)
print(L)
```

```
[1, 2, 3]
[4, 2, 3]
```

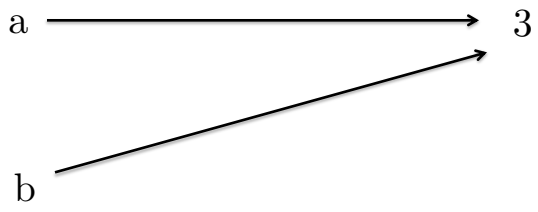
Q: Why did a global variable get changed when I changed a local variables?

A: We have to better understand how variables are assigned in Python. We'll come back to this example later.

9

Variable Assignment

```
a=3
b=a
b=5
```



10

Variable Assignment

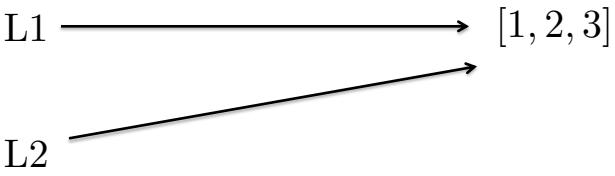
```
a=3
b=a
b=5
```

Anytime you set a variable equal to a new object, its pointer will now point to the new object.

11

Variable Assignment

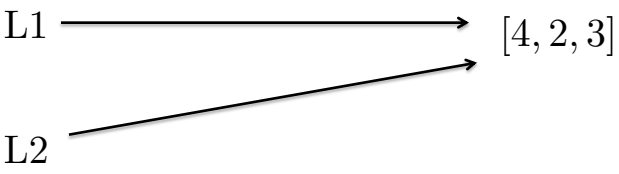
```
L1=[1,2,3]
L2=L1
L1[0]=4
```



12

Variable Assignment

```
L1=[1,2,3]
L2=L1
L1[0]=4
```



13

Example Where This Matters

```
def GetMedianIndex(l):
    l.sort()
    index = int(len(l) / 2)
    return index


l = [4,3,1,4,2,6,7]
middle = GetMedianIndex(l)
l
```

Local scope:

Global scope:

14

Copy

```
import copy
L1=[1,2,3]
L2=copy.copy(L1)
L1[0]=4
```

akin to $L2 = [1, 2, 3]$

L1 ⟶ $[1, 2, 3]$

L2 ⟶ $[1, 2, 3]$

15

Copy

```
import copy
L1=[1,2,3]
L2=copy.copy(L1)
L1[0]=4
```

L1 ⟶ $[4, 2, 3]$

L2 ⟶ $[1, 2, 3]$

16