JamCoders: Week 1

Lecture 4A:

- More Functions
- Scope

Scope

Scope

Variable Scope refers to the “visibility” of variables.

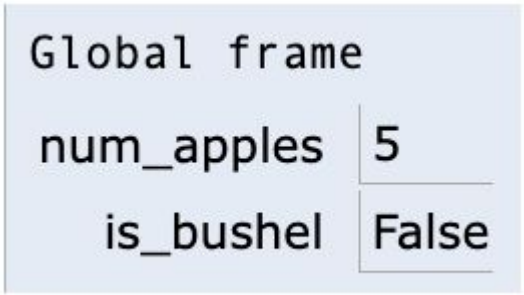
It answers the question “can I reference this variable here”

Scope

All of the code we have written before has always written to and read from the *global* frame.

Therefore, we only had one frame to look at.

```
num_apples = 5
is_bushel = num_apples > 10
if is_bushel:
    print("That's a bushel!")
else:
    print("That's a handful...")
```



visualizer link

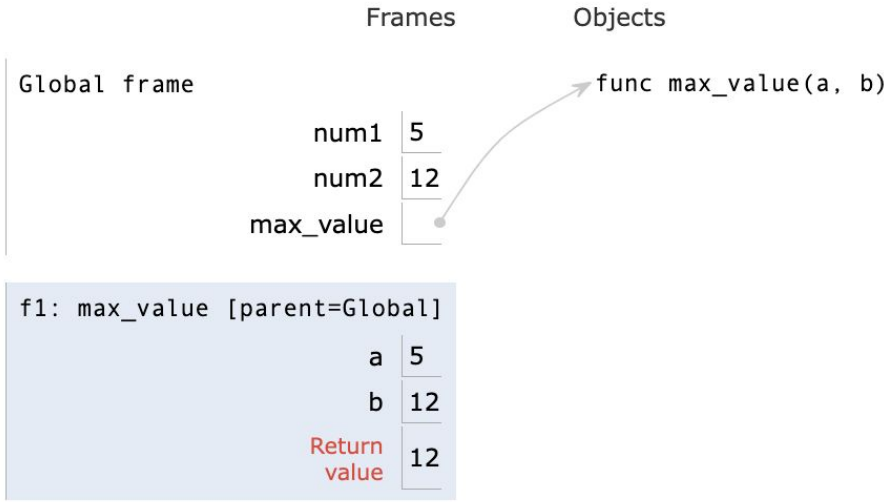
Scope

However, when a function is executed, the function gets *a new frame* for each time it is executed, called the **local frame**.

```
num1 = 5
num2 = 12

def max_value(a, b):
  if a > b:
    return a
  return b

print(max_value(num1, num2))
```



visualizer link

Any variable assignments in the function gets made in the **local frame**. Referencing a variable always looks for a match first in the local frame, and only then in the parent Global frame.

Scope

Tricky case:

```
num1 = 5
num2 = 12

def max_value(num1, num2):
  if num1 > num2:
    return num1
  return num2

print(max_value(45, 32))
```

Think through what happens when this code is executed? Which values are used for num1 and num2 in the function body?

visualizer link

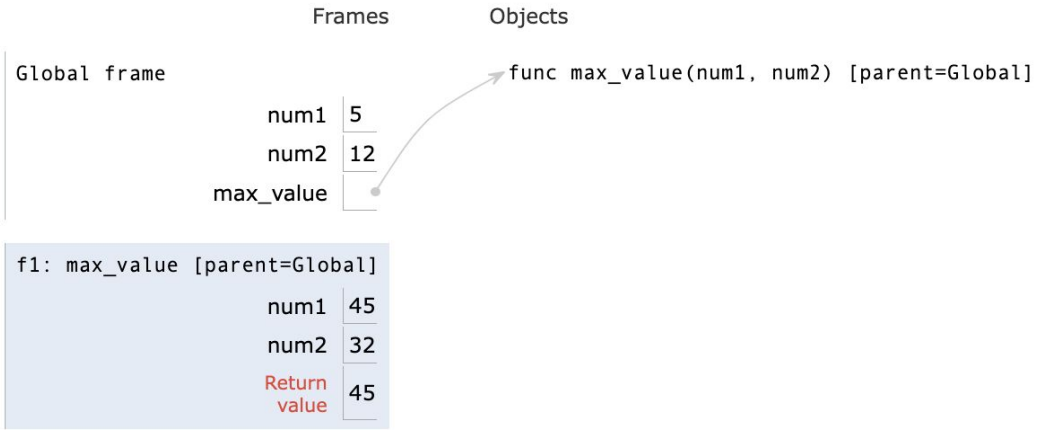
Scope

Tricky case:

```
num1 = 5
num2 = 12

def max_value(num1, num2):
  if num1 > num2:
    return num1
  return num2

print(max_value(45, 32))
```



visualizer link

Scope

Another tricky case:

```
def add_up(a, b, c):
  total = a + b + c

add_up(10, 5, 6)
print(total)
```

Think through what happens when this code is executed? Which values are used for num1 and num2 in the function body?

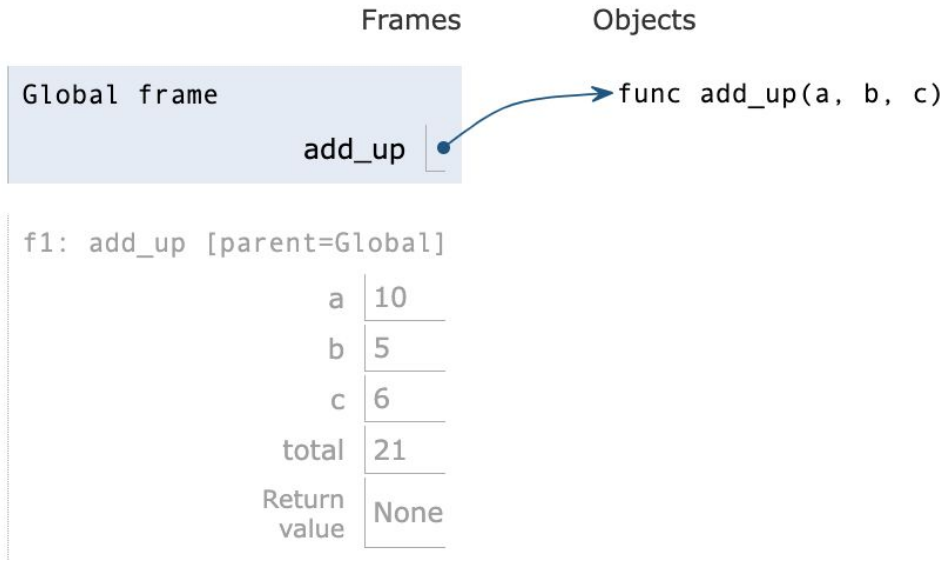
visualizer link

Scope

Another tricky case:

```
def add_up(a, b, c):
  total = a + b + c

add_up(10, 5, 6)
print(total)
```



NameError: name 'total' is not defined

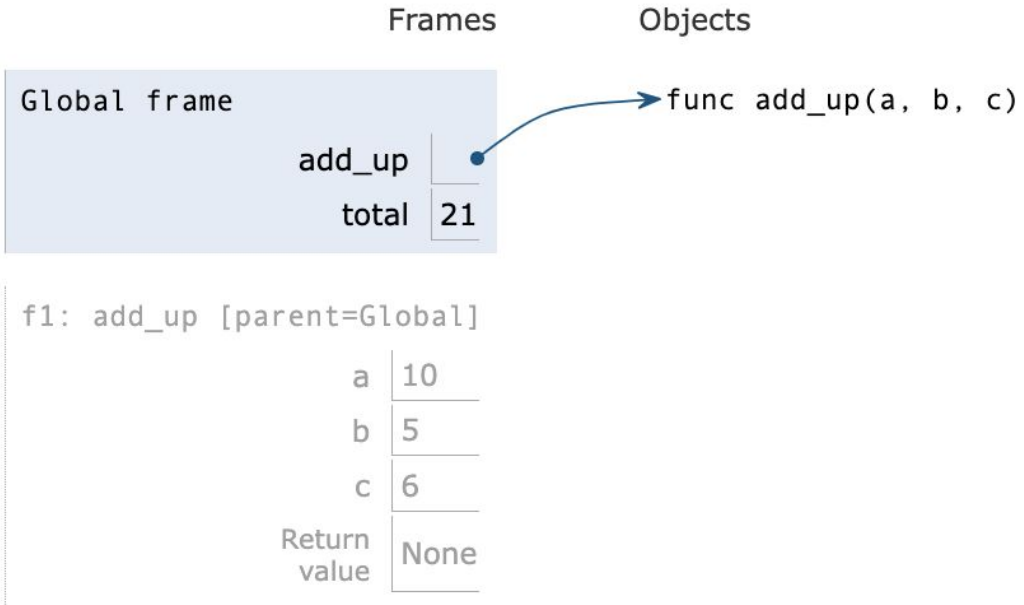
visualizer link

Scope

- Use the `global` keyword inside of your function to specify which variables should be written to the global frame.

```
def add_up(a, b, c):
  global total
  total = a + b + c

add_up(10, 5, 6)
print(total)
```



No error when we go to print `total` because it is written to the global frame!

visualizer link

Local Frames

One last detail…

Local frames get discarded after the function is done running and a value is returned. So anything you assign in just a local frame disappears after the function is done running unless you *return* it or you write to a global variable.

Live Coding Demo