Name:_____ **Partner:** _____

Python Activity 22: Scope – Function Frame Model

The Function Frame model can help us understand how variables map to values.

Learning Objectives

Students will be able to:

Content:

- Use the **function frame model** to describe how **scope** works in python
- Identify how **call frames, global frames,** and **function frames** impact variables.

Process:

- Write code that properly assigns values to local and global variables.

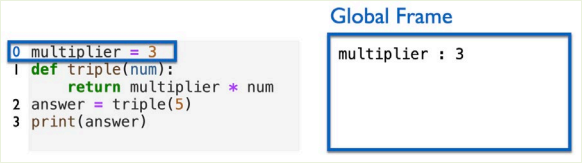
Prior Knowledge

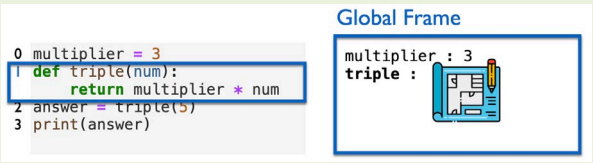
- Python concepts: assignment, functions, expressions, scope

Concept Model:

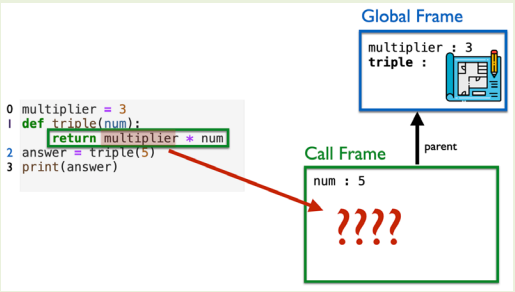
Observe your instructor describing how the **Function Frame Model** works. Below are provided a summary and a few snapshots of the illustrations:

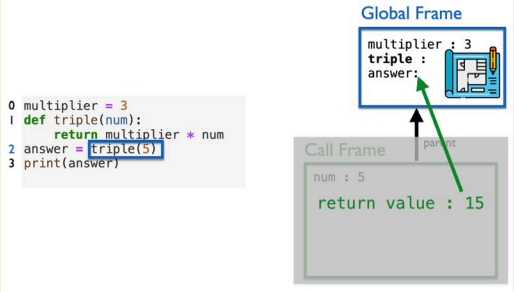
By default, python reads code one line at a time, starting from line 0. At first, when variables are assigned, their values are stored in the **global frame**. Function `def`initions are treated like a single line of code. A `def` statement does not call the function, it just defines it. Effectively, it assigns the name of the function to a blueprint for computing the function.





To execute an assignment statement, python first computes the value of its right-hand side. When a function is called, a new frame is created to record the variables used by that function. First the values of the argument variables are recorded in the **call (i.e., function) frame**. Then, the lines of the function are executed in order. To look up the value of a variable, first python looks in the call frame. If the variable isn't found in the call frame, then python looks in the **parent frame** (the frame we were in when the function was defined).





Ultimately, a **return value** is computed for the function call. The call frame is **destroyed** and the return value of the function call is assigned to the variable on the lefthand side of the assignment operator in the global frame.

Critical Thinking Questions:

1. Examine the following code below:

Code Example

```
0  def triple(num):
1      return multiplier * num
2  answer = triple(5)
3  multiplier = 3
4  print(answer)
```

a. What is recorded in the **global frame** after line 1 is initially seen by python? _____

b. What happens to the frames at line 2?

_____

c. What value is recorded for `multiplier` when `triple(..)` is called on line 2?

_____

d. What might happen when we run this code?

_____

2. Examine the following code below:

Code Example

```
multiplier = 3
def mystery(num):
    return multiplier * num
multiplier = 2
answer = mystery(5)
print(answer)
```

a. What is printed to the computer screen in the example to the left? _____

e. Why might that be?

_____

_____

3. Examine the following code below:

Code Example

```
list = 2468
list_str = list("whodoweappreciate")
print(list, list_str)
```

a. What is printed to the computer screen in the example to the left? _____

b. Why might that be?

_____

4. Examine the following code below:

Code Example

```
a = 3
b = 4
def square(a):
    return a * a
c = square(a) + square(b)
c = pow(c, 0.5)
print(c)
```

```
a = 3
b = 4
def square(a):
    return a * b
c = square(a) + square(b)
c = pow(c, 0.5)
print(c)
```

a. What is printed to the computer screen in the left example? _____

b. Why? _____

c. How do the left and right examples differ? _____

d. How will these two changes impact the output displayed to the computer?

_____