Flow — Functions
Introduction to Programming for Engineers

Find one or two people to work with. Go over the code together and try to reach consensus about what is happening, and then answer the questions that follow.

Part 1

Code

Here is an interaction between a programmer and the Python REPL. Read through it and discuss with your team. Try to come to a consensus what is happening.

The >>> symbol is the prompt asking for user input.

def foo(x, y):
    return x + y

def foo2(x, y):
    return x + y + a

def foo3(x, y):
    print(x + y)
    a = 20
    return x + y + a

def foo4(x, y=0):
    global a
    a = 20
    return x + y + a

a = 10
>>> def foo(x, y):
...     return x + y
... >>> foo(10, 20)
30
>>> def foo2(x, y):
...     return x + y + a
... >>> foo2(10, 20)
40
>>> def foo3(x, y):
...     print(x + y)
...     a = 20
...     return x + y + a
... >>> foo3(4, 5)
9
29
>>> a
10
>>> def foo4(x, y=0):
...     global a
...     a = 20
...     return x + y + a
... >>> foo4(6, 7)
33
>>> a
20
>>> def foo4(x):
...     return x + y + a
... >>> foo4(6)
26
Questions

1. On line 2 we see variables x and y. These are called parameters. On line 5 the values 10 and 20 are called arguments. From what you can see, what is the difference between a parameter and an argument?

2. Variable a is called a global variable. How do global variables seem to work with the rest of the program? Consider what happens in foo2, foo3, and foo4.

3. Line 13 causes output to appear on line 18, and line 15 causes output to appear on line 19. What is the difference between printing a result and returning a result?

4. Line 22 introduces some new syntax. What does it do? What would happen if you got rid of the =0 part and kept the rest of the code as-is?
Part 2

```python
>>> def foo(x):
...     print("Entering foo")
...     x = x + 1
...     print("Returning from foo")
...     return x
...

>>> def bar(y):
...     print("Entering bar")
...     y = foo(y) + 1
...     print("Returning from bar")
...     return y
...

>>> def baz(z):
...     print("Entering baz")
...     z = bar(z) + 10
...     print("Returning from baz")
...     return z
...

>>> baz(20)  # what will happen now?

Questions

1. What will be printed out when we call baz(20)?

2. Challenge: create a function foo that keeps track of how many times it has been called. E.g.,

```python
>>> foo(10)
Foo has been called 1 time.
11

>>> foo(10)
Foo has been called 2 times.
12
```

It will add the number of calls to its argument for the return value. Note the proper pluralization of the word time as well!