Fabien Bessez

COMP 211: Computer Science I

Homework 2a: Searching and Sorting

```
def f() -> None:
    print(y)
    return None

y = 8
f()
```

(A)

```
def f() -> None:
    print(y)
    return None

def g() -> None:
    y = 8
    f()
    return None

g()
```

(B)

```
def f() -> None:
    y = 8
    return None

y = 5
f()
print(y)
```

(C)

FIGURE 1. Programs for written problems.

Problem 1. Consider the Python program in Figure 1a. The way we have described function call evaluation in class, we should expect some sort of execution error, because when we execute the body of f, y will not be in the top binding table on the stack. However, this is not what happens in Python. Describe what happens (i.e., write this code and execute it), and then describe one way in which Python might be using binding tables that would account for Python's behavior.

When we execute the above code, it returns 8. We might have expected the function to return some sort of execution error because we assume y will not be in the top binding table on the stack. One way in which Python might be using binding tables that would account for this output is that it does not create a binding table for the f function until it is called. In other words, it will ignore the function definition for f, then assign the value 8 to y, then it will call the f() function with a value for y so then it prints y as it should.

Problem 2. Consider the Python program in Figure 1b. Based on your response to Problem 1, what do you expect to happen when you execute this program (answer this before writing this program and executing it)? What does happen? Describe one way in which Python might use binding tables that would account for Python's behavior.

I expect Python to output 8 when it runs the code shown in Figure 1b because it will ignore the function definitions, then it will go to check the g() function once it is called.

Inside the g() function, y is given a value and then the f() function is called. As we saw before, the f() function should print 8, and then that will be the final binding table.

Problem 3. Consider the Python program in Figure 1c. Based on your response to Problem 2, what do you expect to be printed (answer this before writing this program and executing it)? What is printed? Describe one way in which Python might use binding tables that would account for Python's behavior.

Based on my response to problem 2 on Figure 1b, I expect Figure 1c to return the value 8. However, when it is run, figure 1c code prints the value 5. A possible explanation for this is that it ignores the function definition, then assigns the value 5 to y. Then, it calls the f() function. The f function provides a local value for y (the local value is 8) within a binding table that is now placed on top of the stack that holds y = 5. Then once the f() function has finished executing, the top binding table that held y = 8 is popped off and the binding table that remains holds y = 5. Therefore, when it is asked to print(y), it prints 5.