Memory Diagrams: Step-by-Step Guide

Step 1: Label the 3 areas of the Stack Diagram

The following are the three major areas of Stack Diagrams:

1) **Print Output:** Print output keeps track of what is printed to the screen. Note: This *must* come from a call to input() or print()!
2) **Frames:** There will be at least one frame in your stack diagram: The Global Frame will have all of our variables created outside of functions, and functions themselves. You can also think of the Global Frame as being updated every time a function is defined, or a variable is assigned a value at the leftmost indentation (not indented at all). In addition to the Global Frame, each *function call* gets its own frame.
3) **Objects:** Each time an *object* (such as a function) is defined, its name goes in the global frame, and its value in the global frame is an arrow that points to the Objects section.

Step 2: Create a Global Frame

Label a new frame in the “Frames” section of the Stack Diagram with the title “Global Frame”. All of our programs have a global frame, as described above. The Global Frame is updated every time a function is defined, or a variable is assigned a value at the leftmost indentation (not indented at all).

Step 3: Update your Stack Diagram

Beginning at Line 1, look at all of the options for types of lines of code below. Use the appropriate section, A, B, C, D, E, F, or G, to update your Stack Diagram accordingly. Continue with this in the order of execution until the program is complete!

A) Print statement:

A print statement is a line that can be outside or inside of a function, and begins with *print* followed by some string value or expression inside a pair of parentheses.

Add whatever is printed in this line to the Print Output section of your Stack Diagram. Note that nothing is added or changed in the Frames or Objects sections!

B) Import Statements:

The easiest case: do nothing! For our purposes, we don’t reflect import statements in our Stack Diagrams. Continue on to the next line in the order of execution.

C) Variable Assignment:

A variable assignment begins with a variable name followed by the assignment operator.

First, **select the right Frame to add to:** If the line of code is not indented at all, then your variable will go in the Global Frame. Else, if the line of code is inside a function, then your variable will go in the Function Frame for the function call that is currently executing. Note: **only one** of these will be the case!

Next, inside the correct Frame chosen above, **add the variable name with a box next to it.** This box will hold the **value** assigned to it – not an expression, or whatever code is to the right of the assignment operator! Follow the instructions for the appropriate option below:

- **Option 1:** If to the right of the assignment operator is a value or expression, such as a mathematical expression or a string concatenation, then inside the box should be the *resulting* value after the expression is evaluated. However, we first must consider the *data type* of the value (Note: this only applies post-midterm)
    - If the data type is an *object* (i.e., Lists or Dictionaries), then the value is an arrow to the objects section of the Memory Diagram (similar to function definitions)
    - Else, if the data type is *not* an object (i.e., int, float, string, boolean), the value (or resulting value of an expression) goes directly in the box next to the variable name.

- **Option 2:** If to the right of the assignment operator is an *input* statement, leave the box empty for now and follow the instructions in Part D below. Then, put whatever value the user has input, which appears in the Print Output section, inside the box.

- **Option 3:** If to the right of the assignment operator is a function call, then leave the box empty for now and go to part F. Once that function frame is complete, the **return value** will be copied into this box you left empty.

D) Input statement:

An input statement is a line that can be outside or inside of a function, and begins with *input* followed by some string value or expression inside a pair of parentheses.

Add whatever text is inside the parentheses to the Print Output section of your Stack Diagram. Also in the Print Output section will be the *value that the user inputs*, immediately next to the text you printed still on the same line.

E) Function Definition:

A function definition is a line that is not indented at all, begins with the *def* keyword, and ends with a colon. In the Memory Diagram, these will update the **Global Frame** and **Objects** section.

For these lines, add the *function name* to the Global Frame, similar to how variable names are added, with a box to the right of it. The corresponding value in this box is an arrow that points to the Objects section, where we put a Function object with its name and parameters. Ex: *greeting(name)*.

Remember, after this line executes, continue to the next line in the program that is **not** indented!

F) Function Call:

Each time a function is called, a new Function Frame is added to the Frames section of the stack diagram.

Go to the function definition line for the function that was called, and inside the Function Frame, add all parameter names in the same way as variables, with their value as the argument passed in the function call.

Next, execute the lines inside the function, going to the appropriate steps above. Remember that function frames have their own separate variables: Variables created inside the function are added to the Function Frame, not the global frame!

G) Return Statement **OR** End of Function with No Return

When a function returns, or finishes executing the last line of the body without a return statement, the "Return value" is added to the function frame just like a variable, with the corresponding return value.

Recall that a function always returns a value, and if there is no return line then it defaults to *None.* Therefore, this step applies in two cases:

- The line of code begins with the keyword **return**
- The previous line executed was the last line in the body of a function, and there is no return statement

Also, once the function returns, be sure to **cross out this function frame** in your Memory Diagram.