Code Tracing

What is code tracing?

If you remember from class (both java and python) we would go through a chunk of code, and keep track of variable values as they changed, to find out what our final value or output would look like. We were essentially tracing the code to make sure it worked as intended. Formally, manual code tracing is when the programmer interprets the results of each line of code and keeps track by hand of the effect of each statement. This type of code tracing is often faster than using a debugger to step through (trace) the execution of a program. Debuggers automate the code tracing, and you will learn about them in a later lesson. While manually tracing code is more error prone than using a debugger tool, it is a skill that all good programmers possess. This is because knowing how to manually trace a small block of code correctly does not require a programmer to run an entire program over to see the effect of a small edit in the code. Below is an example.

```plaintext
1    SUM=0
2    NUM=2
3    WHILE (NUM<100)
4         SUM = SUM + NUM
5         NUM=NUM*5
6    PRINT (SUM)
```

Let’s trace line by line, and keep track of the two variables, SUM and NUM.

To begin:

We see SUM=0 on line 1 and NUM=2 on line 2.

Next on line 3, we see a conational while loop. It tell us to keep doing what is inside of the loop until NUM is equal to or bigger than 100. Another way to say it is continue while NUM is less than 100.

Let go inside the loop, but first, is NUM less than 100? Yes, NUM=2.

Line 4 says to add SUM and NUM together. SUM=0 and NUM=2, 0+2=2, and that is the new value for SUM. SUM=2 now.

Line 5 says multiply NUM by 5, and restore that value back into NUM. NUM=2, so we do 2*5=10, so now NUM=10.

We are at the end of the loop so we need to check if we need to run the loop again. Let’s Check; is NUM less than 100? Yes, NUM=10, so we jump back into the loop.

At this point, SUM=2, and NUM=10.

We go to line 4, add SUM and NUM together again and restore that value in SUM. SUM+NUM is 210=12, so SUM=12 now.

*If you keep track... ⦿ 2 12
*This is the thrid value for sum.

Line 5, NUM*5 is 10*5=50. So now NUM=50.

Let’s check line 3 again to see if we jump back in. IS NUM less than 100? Yes, NUM=50, so we do it again.

Line 4, SUM+NUM is 12+5=62, so SUM=62.

Line 5, NUM*5=50*5=250. So NUM=250 now.

Again, we check line 3, is NUM less than 100? NO!

Since the condition of NUM<100 is not true, we do not re-enter the loop, and proceed to line 6.

*Keeping track SUM = 0 2 12 62

*of variables NUM = 0 10 50 250

Line 6 simply says to print SUM, so the end output to the screen would be:

   62

For the following assignment, I want you to trace through the code, making sure you keep track of the values of the variables as they change, as I did in the example above where you see **keeping track of variables. Once you trace through the code while keeping track of the variables, if there is a final output to the screen (I have used python’s print() command, if you are Java, it works the same as System.out.println)

**ASSIGNMENT:** Trace the program blocks below, keeping track of variables, and state the any actual output to the screen.

1.

    i = 0
    total = 0
    while total < 10 :
        i = i + 1
        total = total + i
    print(i, total)
2. 
   \[
   x = 2 \\
   \text{while } (x < 200): \\
   \quad x = 2 * x \\
   \quad \text{print } (x)
   \]

3. 
   \[
   \text{count} = 0 \\
   \text{sum} = 0 \\
   \text{num} = 5 \\
   \text{while } (\text{count} < 10) \\
   \quad \text{sum} = \text{sum} + \text{num} \\
   \quad \text{num} = \text{num} + 5 \\
   \quad \text{count} = \text{count} + 1 \\
   \text{if } (\text{num} > 100): \\
   \quad \text{print } (\text{“Bigger than 100”}) \\
   \text{else:} \\
   \quad \text{print}(\text{“Smaller than 100”})
   \]

4. The code chunk below has a problem with it. As you trace through the code, you should be able to identify the problem innate to the code. What is the problem?

   \[
   \text{num} = 100 \\
   \text{while } (\text{num} > 0): \\
   \quad \text{num} = \text{num} / 2 \\
   \text{print } (\text{num})
   \]