Functions II

Warm-up

Last class, we practiced with functional decomposition and composition. Test your knowledge by answering the following questions:

Questions (15 min) Start time:

1. Design a program that prints a fish. Make sure to define appropriate utility functions for the distinctive features (such as the tail, body and head).

```
   vvv
    v
   vvv
  vvvvv
 vvvvvvv
vvvvvvvvv
 vvv vvv
  v   v
```

Model 1 Flow of Execution

As we saw last class, in addition to using Python's built-in functions (e.g., `print`, `abs`) and functions defined in other modules (e.g., `math.sqrt`), you can write your own functions.
Trace the following code:

```
1  def model_one():
2      word = input("Enter a word: ")
3      L = len(word)
4      ans = word * L
5      print(ans)
6
7  def main():
8      print("Starting main...")
9      model_one()
10     print("All done!")
11
12 main()
```

2. What is the output of the code above if the user enters `Hi`?

3. **STOP HERE and wait for further instructions**. The professor will trace the code using `Thonny`. As the professor steps through the program, pay attention to what is happening in the visualization.

a) Which button stops the execution of the code?

b) Which button starts the tracing of the code?

c) Which button allows the step by step tracing of the code?

d) Which button stops the tracing of the code and resumes the execution of the code?

4. What happens when a function is called?

5. What happens when a function finishes executing?

6. Draw the tracing diagram corresponding to the code here:

main's Frame	**model_one's Frame**

7. Notice that the variable `ans` is printed from within the `model_one` function. What happens if you try to `print(ans)` inside the main function?

8. In the space below, write a definition for a function called `str_to_list` that prompts the user to enter a word. The function should convert the string to a list and print the list.

Model 2 Passing Arguments

Instead of using `input` inside a function to get data, we can define a function to take a ***parameter*** (variable). When we call the function, we need to provide an ***argument*** (value). Let's trace the following code:

```
1  def model_two(word):
2      ans = word * len(word)
3      print(ans)
4
5  def main():
6      print("Starting main...")
7      w = input("Enter a word: ")
8      model_two(w)
9      print("All done!")
10
11 main()
```

9. Draw the tracing diagram corresponding to the code here:

main's Frame	model_one's Frame

10. Underline the parameter in the `model_two` function definition, then circle each use of the parameter inside the function.

11. Find the `model_two` function call in `main`, and underline the argument being passed by the function call.

12. When a variable is used as an argument, does the name of the variable need to be the same as the parameter variable name?

13. Assume that `s1 = "Hi"` and `s2 = "ya"`. In the function call `model_two(s1 + s2)`:

a) What is the argument for the function call?

b) Write the implied assignment statement that happens during the call.

c) What will be the value of parameter `word` when `model_two` begins executing?

d) Predict the output that will be produced by the function call.

14. Review the two implied assignment statements that you have written. What exactly gets "passed" when you call a function?

15. Change `model_two` so that, instead of multiplying `word` by the length of `word`, it will multiply by an integer passed as the second argument to the function. Write the new version of `model_two` in the space below. Use `times` for the name of the new integer parameter.

16. How does the call to `model_two` in `main` need to change so that it matches the new function definition? Give an example.

Model 3 Returning Values

Functions may optionally send a value back to the calling function using a `return` statement. Change the program in Python Tutor as follows:

```
1 def model_three(word):
2     ans = word * len(word)
3     return ans
4
5 def main():
6     print("Starting main...")
7     w = input("Enter a word: ")
```

```
8       result = model_three(w)
9       print(result)
10      print("All done!")
11
12  main()
```

17. Draw the tracing diagram corresponding to the code here:

main's Frame	model_one's Frame

18. Aside from the function name, how does line 8 in this exercise differ from line 8 in the previous exercise?

19. At what step number (in the simulation) has `model_three` completed its execution, but control has not yet returned to the `main` function?

20. In general, what value will be returned by `model_three`?

21. What changes in the frame for `main` at Line 8 during the execution of the program?

22. What value is returned by a function when there is no return statement?

23. Change line 8 so that `model_three` is still called but there is no assignment to `result`. What do you predict will happen in `main` after the `model_three` function call completes?

24. Why is a function that returns the value of a variable more useful than a function that simply prints the value of that variable?

25. Redesign the following program, by moving all the provided code statements in functions for respective distinctive features and to increase re-usability (and reduce repetition). Your resulting program must also have a `main` function that is the entry point in your program (that is, it gets executed first).

```
cube_len = int(input("Cube length? "))
box_w = int(input("Box width? "))
box_h = int(input("Box height? "))
box_l = int(input("Box length? "))

fit_w = box_w // cube_len
fit_h = box_h // cube_len
fit_l = box_l // cube_len

print(str(fit_w)+" Rubik's cubes will fit width-wise.")
print(str(fit_h)+" Rubik's cubes will fit height-wise.")
print(str(fit_l)+" Rubik's cubes will fit length-wise.")

res = fit_w * fit_h * fit_l
print(str(res)+" Rubik's cubes will fit in that container.")
```