CSC120H Lab 2

1 Objectives

- Use the Function Design Recipe
- Write functions in a Python script file.

2 Driver and navigator

Just like last time, you **must** complete this lab with a partner, and you and your partner will take on distinct roles:

driver: The person typing at the keyboard.

navigator: The person watching for mistakes, and thinking ahead.

The rest of these instructions call you `s1` and `s2`. Pick which one is which. `s1` should log in, start up Wing, and be the first driver. (If you forgot how to start Wing, check out the screenshots on the course webpage from last week.)

3 Function Design Recipe

This week in lecture, we learned about the recipe for writing our own functions. Here is a quick reminder of the steps.

1. **Examples:** pick a meaningful function name; show 2 or 3 function calls with expected return values.
2. **Header:** write the first line of the function definition; pick meaningful parameter name(s). Include a **type contract** (the parameter and return types).
3. **Description:** a description of the function; say what is returned and mention parameters by name.
4. **Body:** write the body of the function.
5. **Test:** execute the function calls and check whether the values returned match what was expected.

4 Writing Type Contracts

For each of the functions below, we have completed step 1, and most of the header in step 2. Complete the **Type Contract** by **writing directly** on this handout (the driver should write). Do not complete steps 3-6 of the function design recipe.
Explain to your partner why the type contracts for the three functions are different.

```
def function1(n:         ) ->            :
"""
>>> function1(2)
4
>>> function1(3)
5
"""

def function2(x:         ) ->            :
"""
>>> function2(2)
4.0
>>> function2(3)
5.0
"""

def function3(y:         ) ->            :
"""
>>> function3(2.0)
4.0
>>> function3(3.0)
5.0
"""
```

5 Writing Descriptions

Switch driver and navigator.
For each of the functions below, we have completed steps 1 - 2. Complete step 3: add a **Description**. Do not complete steps 4-5. Remember: A good description for a function that returns a value starts with the word `Return` and mentions all of the parameters by name. It does **not** explain the details of *how* the function does its work.
Use the names of the functions and the examples to help you understand what the function does, and write your description accordingly.

```
def number_of_days(num_weeks: int) -> int:
    """




    >>> number_of_days(1)
    7
    >>> number_of_days(3)
    21
    '''
```

```
def bill_split(total: float, num_people: int) -> float:
    '''




    >>> bill_split(10.0, 2)
    5.0
    >>> bill_split(10.0, 4)
    2.5
    '''
```

6 Calculating Area

Switch driver and navigator.
Create a **new Python script file** in Wing. Save it right away in your home directory as `lab2_part6.py`. You will write your code in this file for now.
You will be calculating the area of a circle. Recall that the area of a circle is defined as πr^2, where r is the radius.
We are going to simplify things by assuming $\pi = 3$.

1. **In the Wing editor** (not in the Shell), create a variable named `radius1`, and assign it the value 2. Now create a new variable called `area1` under it and assign it the value of multiplying 3 by (`radius1`)^2. (The '3' is the value of pi).

2. After the variable assignment for `area1` in your Python script, create a variable named `radius2`, and assign it the value 3. Now create a new variable called `area2` and assign it the value of multiplying 3 by (`radius2`)^2.

3. After the variable assignment for `area2` in your Python script, create a variable named `radius3`, and assign it the value 4. Now create a new variable called `area3` and assign it the value of multiplying 3 by (`radius3`)^2.

4. Run your Python script by clicking the green run button on the top of Wing. Notice how nothing appears in the shell at the moment.

5. In your Python script, add some calls to the `print` function to print out the values of the three area variables. For example: `print(area1)`
   You should have three calls to `print`.

6. Re-run your Python script with the added `print` calls. Check to make sure the shell now prints out those values.

7. Notice that after assigning all of those variables, your Python script is getting a little repetitive. Explain to your partner a good way to assign the area variables without having to create variables for each radius and writing out the entire area calculation each time (Hint: we've been talking about it all week!)

You will now use the Function Design Recipe to write a function that calculates the area of a circle given a specific radius. Create a new Python script called `lab2_functions.py` which you will put your function definitions into.
continued...

You will define a function in your new script called `circle_area` which takes a float argument called `radius` and will return a float containing the area of the circle.

1. Perform steps 1-3 of the Function Design Recipe. Write some examples in the docstring, put the header above them with the correct type contract, and add a proper description. Refer to the examples in the Function Design Recipe lab handout you were given for more guidance.
2. Write the function body with a proper return statement at the end (step 4 of the recipe).
3. Time to test your function! Instead of just randomly putting in values, make some test cases and write down what you expect the return value of the function will be when you run it. Put a few simple and complicated cases you want to test, including edge cases like `radius = 0`. Don't fill in the 'Actual Return value' column until the next step.

Argument `radius` value	Expected Return value of function	Actual Return value
`2`	$3 * 2^2 = 12$	
`3`		

4. Now let's see if your function works as expected. Under and outside your functions, add some print statements with calls to your `circle_area` function. For example:

   ```
   print(circle_area(2))
   ```

   You should have many print statements, one after the other. Make sure they are **not indented** inside your functions.

   Run your Python script to run the print statements. Fill in the 'Actual Return value' column in the table above. If you find that you have a different result than what you expected for any test case, try to figure out what went wrong. Ask your TA for any assistance.

   When you are done this part, show your TA your work before moving on to the next part.

7 Functions with multiple arguments: Calculating Volume

Switch driver and navigator.
Now, you will use your `circle_area` function as part of the function body for another function.
The volume of a cylinder is defined as $(\pi r^2)h$, where r is the radius of the base, and h is the height of the cylinder. We will again **assume $\pi = 3$**.

1. In the expression for the volume, $(\pi r^2)h$, which part looks similar to the previous function you wrote?: _____. What is different?: _____.
2. Open your `lab2_functions.py` script.
3. Under your `circle_area` function, define a function called `cylinder_volume.py` which takes **two** float parameters called `radius` and `height`, and returns a float.
4. Perform the first three steps of the function design recipe (examples, header, description). Don't pick anything too complicated for the examples.
5. Write the function body with a proper return statement at the end. **You must call your circle_area function** inside your new function. This time, also try to use an **intermediate variable** (don't shove all your code into the return statement).

6. Time to test your function! Just like before, make some test cases and write down what you expect the return value of the function will be when you run it. Put a few simple and complicated cases you want to test, including edge cases like `radius = 0` and `height = 0`. Don't fill in the 'Actual Return value' column until the next step.

`radius` value	`height` value	Expected Return value of function	Actual Return value
`1`	`2`	$(3 * 1^2) * 2 = 3 * 2 = 6$	
`2`	`1`		

7. Now let's see if your function works as expected. Under your functions, add some print statements with calls to your `cylinder_volume` function. For example:

   ```
   print(cylinder_volume(1, 2))
   ```

   You should have many print statements, one after the other.

   Run your python script to execute the print statements. Fill in the 'Actual Return value' column in the table above. If you find that you have a different result than what you expected for any test case, try to figure out what went wrong. Ask your TA for any assistance.

   When you are done this part, show your TA your work.

8 If you have time: Loan Calculator

Attempt this section if you have lab time remaining. If you do not complete it in lab, work on it at home.

In this section you will write a loan calculator function using the following formula for Compound Interest:

Compound Interest Accrued $= A * ((1 + i/n)^{nt} - 1)$

where A is the principal or initial amount of the loan, i is the nominal interest rate per year, n is the number of times interest is compounded per year and t is the number of years.

Create a file called `loan_function.py` and write the following function **using the Function Design Recipe**. All the information that you need is in the following table. You have to specify the parameter names; make sure you pick good names for the parameters based on what they represent in the equation for compound interest (**don't use the letters A, i, n ,t**)!

Function Name	Description
`compound_interest(...parameters...)`	• Returns the compounded interest accrued on the loan. For example, if the loan is \$2000, the nominal interest rate is 5%, the interest compounds quarterly (i.e., 4 times per year), and the loan is help over a period of 3 years, then it will accrue \$321.509 in interest. • The first parameter is the principal or value of the loan. • The second parameter is the nominal interest rate. (Note that 5% interest would be input as 0.05.) • The third parameter is the number of times the interest is compounded per year. • The fourth parameter is the number of years.

Create a new Python Script to test your functions with print statments (give it an appropriate name).
For testing, try the following argument values:

Let $A = 1000; i = 0.06; n = 12; t = 3$. How much compound interest is accrued?
Let $A = 500; i = 0.13; n = 4; t = 5$. How much compound interest is accrued?

Call your TA over to make sure you have the correct results.