Lab #3 Grading Rubric and Instructions

This is the third lab for the 'Introduction to Computer Programming' course.

Please see the syllabus for information about when the work in this lab is due.

Lab Objective

The purpose of this lab is for you to familiarize yourself with creating functions, and assigning the function output to a variable when the function returns. You will also get a bit more practice using loops and the 'turtle' library.

Instructions/ Deliverables

NOTE: These tasks can be completed in any order you like. See the **Grading Items** section below for the point distribution.

CITATION: Many of the exercises found here could possibly be seen as adaptations of exercises found in the online textbook “How to Think Like a Computer Scientist: Interactive Edition” By Jeffrey Elkner, Peter Wentworth, Allen B. Downey, Chris Meyers, and Dario Mitchell.

- **Available**: https://runestone.academy/ns/books/published/thinkcspy/index.html?mode=browsing
- **Accessed**: 2-2-2023
- The abbreviation 'thinkcspy' and the chapter/ section number will be used to indicate where similar exercises can be found. This citation will be placed next to the exercise title.
    - ex: [thinkcspy 2.13] indicates a similar exercise can be found in chapter 2, section 13.

CITATION: Some of the exercises found here are completely original to the instructor.

- The abbreviation 'MH' will be used to indicate these exercises. This citation will be placed next to the exercise title.
    - ex: [MH]

Lecture Notes/ Supplemental Readings:

- 'Check off' your notes in your Engineering Notebook for the following material with the TA/ Instructor. These notes should already be done before the start of the lab period.
    - Runestone chapter 6
        - **NOTE**: You do not need to complete any of the exercises at the end of the Runestone chapters. However, it would be helpful to you in the long term if you were to do so.
        - **NOTE**: You only need to take notes on the Runestone readings if you did *not* take lecture notes in class. The contents of these chapters are what was covered in the lectures.
    - `Debugging.pptx`
        - **Available**: `'Files' -> 'Supplimental_Slides_and_Files'` folder on Canvas
    - `Debugging_in_VS_Code.pptx`
        - **Available**: `'Files' -> 'Supplimental_Slides_and_Files'` folder on Canvas
    - What Does if **name** == "**main**" Do in Python? - by: Martin Breuss

        - **Available**: https://realpython.com/if-name-main-python/

`Code Conversion pt. 1`: [MH]

- Retrieve your code from lab week 3 - specifically `areaOfRectangle.py`, `rectanglePerimeter.py`, `areaOfCircle.py`, `circleCircumference.py`, `distanceSpeedTime.py`, `velocityAccelerationTime.py`, `calculateVoltage.py`, `calculateResistance.py`, `calculateCurrent.py`, `annualPercentageRate.py`, and `compoundAmount.py` - everything *except* `initials.py`.
    - Copy and paste each of these files into a new folder on your computer. These are the files you will work with on this exercise.
- Convert each one of these scripts to a new script that uses a function to calculate its output, instead of performing a calculation 'in line' with everything else in your code at the global scope.
    - The function in each script should be named according to the file name. For example, the `areaOfRectangle.py` script should have an `areaOfRectangle()` function, which takes the inputted values as parameters.
        - Be sure to include your citations *inside* these new functions.
- For each file, use a `main()` function, like the one seen in section 6.8 of the Runestone textbook.
    - Use the `if __name__ == "__main__":` version as seen in activity 6.8.2.
- Call `main()` inside the `if __name__ == "__main__":` statement in each file.
- Collect your input inside the `main()` function in each file.
- Call the new function you created for the file (ex: `areaOfRectangle(width, height)`) and pass your input value(s) into the new function. Make sure to Collect the return value from the function in a new variable back in `main()`.
    - Ex: `answer = areaOfRectangle(width, height)`
- Finally print out the results of the calculation.
    - Ex: `print("The answer is:", answer)`
- Please see the example code for `myFunctionExample.py` (in the 'Example Script' section) for details about what this file should look like.
- Save your code to files called `areaOfRectangle.py`, `rectanglePerimeter.py`, `areaOfCircle.py`, `circleCircumference.py`, `distanceSpeedTime.py`, `velocityAccelerationTime.py`, `calculateVoltage.py`, `calculateResistance.py`, `calculateCurrent.py`, `annualPercentageRate.py`, and `compoundAmount.py`. Each of these files should include your name, code creation date, lab number, and *new* brief description in *each* file. These will be the files you show the TA/ Instructor to 'check off.'

`randomProduct.py`: [thinkcspy 5.7]

- Take input from the user in the form of three integers - `a`, `b`, and `c`.
- Write a function that takes the three integers as input.
    - This function will be called `randomProduct()`, and it should use the `randrange()` function to calculate the product of `a` random numbers in a range between `b` through `c + 1` (non-inclusive).
        - For example, if `a = 5`, `b = 1`, and `c = 6`, then we would find the product of 5 random numbers between 1 and 6 inclusive, but we will be using 7 as the argument to the `randrange()` function (as `c = 6` and `6 + 1 = 7`, but the `randrange()` function is non-inclusive).
            - Meaning, if `a = 3`, `b = 2`, and `c = 5`, we might get a final calculation of: `3 * 5 * 4` where these three numbers (from `a`) are randomly generated (between `b` and `c + 1` non-inclusive).

        - HINT: This task will certainly involve using a 'for loop.'
        - HINT: You will need to import the `random` module at the top of your code for this to work.
        - HINT: You can assume that the user will provide proper input. Meaning that `b <= c`.
- For this file, use a `main()` function, like the one seen in section 6.8 of the Runestone textbook.
    - Use the `if __name__ == "__main__":` version as seen in activity 6.8.2.
- Call `main()` inside the `if __name__ == "__main__":` statement.
- Collect your input inside the `main()` function.
- Call the new function you created for the file and pass your input value(s) into the new function. Make sure to Collect the return value from the function in a new variable back in `main()`.
    - Ex: `answer = randomProduct(a, b, c)`
- Finally print out the results of the calculation.
    - Ex: `print("The answer is:", answer)`
- Please see the example code for `myFunctionExample.py` (in the 'Example Script' section) for details about what this file should look like.
- Save your code, including your name, code creation date, lab number, and brief description of what your code does, to a file called `randomProduct.py`.

`sqrtIter.py`: [thinkcspy 6.13]

- Review the following webpage, https://www.cuemath.com/algebra/square-root-of-2/ (accessed 2-2-2023) - particularly the section labeled "Square Root of 2 by Estimation and Approximation Method."
    - Notice the value of 'x' in the formula. This is the value we want to find the square root of.
    - Consider that we can extrapolate the information in "the iteration formula" to make a general-purpose function to calculate *any* square root - not just the square root of two (2).
    - NOTE: Be sure to cite the website above in your code.
- In your code, take two (2) integer input values from the user. One value for `x`, and the other called `iterations`.
    - `x` will store the number we want to calculate the square root of.
    - `iterations` will store how many times we want to do a 'for loop' on the formula.
- Create a function, called `sqrtIter()`, which takes both `x` and `iterations` as parameters. -This function will return the square root of `x`. - You will calculate the square root of `x` by running the "estimation and approximation method" shown on the website. - You will run this method in a 'for loop' the number of times specified by the 'iterations' variable.
    - HINT: We are calculating and returning `y`, as seen in the formula.
    - HINT: The 'initial guess' for 'y' can be stated in your code as `y = (x + 1) / 2`. This 'initial guess' is calculated from the formula `y = ((x / y_0) + y_0) / 2`, where the initial value for `y`, written here as `y_0` in the previous formula, has the value of one (1).
        - NOTE: This 'initial guess' computation should be the first thing that happens in your function - before your 'for loop.'
        - After calculating the 'initial guess,' You can then continually recalculate 'y' in your subsequent 'for loop.' You can do this using the formula `y = ((x / y) + y)/2`, as seen on the website.
            - This will work because `y` will have already been supplied with an initial value from the 'initial guess' the line above the 'for loop.'
            - If we did not have the 'initial guess,' and just tried to run the 'for loop' directly, we would have a 'variable undefined error.' This is because we would be attempting to assign something to our value for `y`, using `y` in the expression to assign to `y`, before `y` even has a value inside of it to begin with.

    - **HINT**: This task will certainly involve using a 'for loop.'
- For this file, use a `main()` function, like the one seen in section 6.8 of the Runestone textbook.
    - Use the `if __name__ == "__main__":` version as seen in activity 6.8.2.
- Call `main()` inside the `if __name__ == "__main__":` statement.
- Collect your input inside the `main()` function.
- Call the new function you created for the file and pass your input value(s) into the new function. Make sure to Collect the return value from the function in a new variable back in `main()`.
    - Ex: `answer = sqrtIter(x, iterations)`
- Finally print out the results of the calculation.
    - Ex: `print("The answer is:", answer)`
- Please see the example code for `myFunctionExample.py` (in the 'Example Script' section) for details about what this file should look like.
- Save your code, including your name, code creation date, lab number, and brief description of what your code does, to a file called `sqrtIter.py`.

`tridecagonTurtle.py`: [thinkcspy 6.13]

- Prompt the user to enter three numerical integer values - `s`, `x`, and `y`.
- Write a function, called `tridecagonTurtle()`, which takes `s`, `x`, `y`, and `t` as parameters, and use the turtle library to draw a shape called a **regular tridecagon** at the location and at the size specified by the user.
    - The `s` value specifies the length of one of the sides of the regular tridecagon.
    - The `x` and `y` values specify the `(x, y)` coordinates of the regular tridecagon's first point (called a vertex).
    - The `t` value is the turtle that will be drawing the regular tridecagon. You will create a turtle *outside* the `tridecagonTurtle()` function, and will then pass that turtle *into* the function as an argument by way of the `t` parameter in the function signature.
    - HINT: You may need to perform an internet search to learn what this shape is, and what angle occurs between each of its segments. You will need to cite this information in a comment. Be sure to include the URL, author (if available), date the article was written, and the date you accessed the webpage.
    - HINT: This task will certainly involve using a 'for loop.'
- For this file, use a `main()` function, like the one seen in section 6.8 of the Runestone textbook.
    - Use the `if __name__ == "__main__":` version as seen in activity 6.8.2.
- Call `main()` inside the `if __name__ == "__main__":` statement.
- Collect your input inside the `main()` function.
- Call the new function you created for the file.
    - Ex: `tridecagonTurtle(s, x, y, t)`
- Save your code, including your name, code creation date, lab number, and brief description of what your code does, to a file called `tridecagonTurtle.py`.

Debugging Demo: [MH]

- Choose one of the scripts you have created for this lab and run the VS Code debugger on it.
- Experiment with the debugger until you are confident you know how it works.
- Demonstrate the debugger to the TA/ Instructor by running the debugger on the script of your choice. Place a 'break point' on the topmost function signature and run the debugger until the script execution is complete.

Attendance:

- If you have completed all of your tasks for the lab, you may work on any of the 'Additional Resources for Study' found in the Canvas announcement of the same name.
    - **NOTE**: If you leave early, you will not receive the 'attendance points' for the lab.

Optional Readings

NOTE: These readings are not required. However, they may provide a bit of interest/ insight into the broader world of Computer Science. Please complete the rest of your lab tasks before doing these readings. You do not need to take notes on these in your Engineering Notebook.

How To Avoid These 7 Real Estate Scams - by: Jamie Johnson

- Available: https://www.rocketmortgage.com/learn/real-estate-scams

“Things I Wish I Knew” as a Computer Science (or related) Major - by: admin

- Available: https://wics.ics.uci.edu/things-i-wish-i-knew-as-a-computer-science-or-related-major/

What Jobs Use Python? A Comprehensive Guide - by: Simone Du Toit

- Available: https://careerkarma.com/blog/jobs-that-use-python/

The 10 Worst Companies in the Video Game Industry - by: ANDR01D

- Available: https://levelskip.com/industry/Worst-Video-Game-Companies

Files Provided

None

Example Script

myFunctionExample.py

```
# Matthew Holman                2-9-2024
# Lab Week 4 - An example of how to convert a previously made script.

def addTwo(x):
        # CITATION - URL: ?
        # CITATION - Author: ?
        # CITATION - Date Written/ Posted: ?
        # CITATION - Date Accessed: ?

        return x += 2

def main():
        val = int(input("Please enter an integer: "))
        answer = addTwo(val)
        print("Your value plus two is:", answer)

if __name__ == "__main__":
```

```
main()
```

Example Output

Running: `python myFunctionExample.py`

```
Please enter an integer: 5
Your value plus two is: 7
```

Grading Items

- (**Lecture Notes/ Supplemental Readings**) Has the student taken notes on the listed material, and shown their notes in their Engineering Notebook to the TA/ Instructor?: _____ / 20
- (**Code Conversion pt. 1**) Has the student completed the task above, and saved their work to files with the names specified in the exercise?: _____ / 20
- (**randomProduct.py**) Has the student completed the task above, and saved their work to a file called `randomProduct.py`?: _____ / 10
- (**sqrtIter.py**) Has the student completed the task above, and saved their work to a file called `sqrtIter.py`?: _____ / 10
- (**tridecagonTurtle.py**) Has the student completed the task above, and saved their work to a file called `tridecagonTurtle.py`?: _____ / 10
- (**Debugging Demo**) Did the student demonstrate their knowledge of the VS Code debugger?: _____ / 20
- (**Attendance**) Did the student attend the full lab meeting in person?: _____ / 10

TOTAL _____ / 100