Lab Week 6 Grading Rubric and Instructions

This lab is assigned for Week 6 of COM S 127: Introduction to Programming.

This lab is due by the end of the lab period seven (7) days after the one it is assigned in. See the syllabus for details.

Lab Objective

The purpose of this lab is to give students practice with modifying a piece existing code, injecting their own code into this existing code, and adding new functionality to this code to extend its functionality. This lab also allows students to practice working with strings, and to re-use their own code to solve smaller portions of larger problems.

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: 2-18-2023
- 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.

Reading:

- Read Runestone chapter 9, and show the TA the notes you took in your Engineering Notebook for this chapter once you are done.
    - NOTE: You do not need to complete any of the exercises at the end of the chapter. However, it would be helpful to you in the long term if you were to do so.

`tridecagonLSystem.py`

- Re-Read chapter 9.15 in the Runestone textbook, and make sure you really understand the code of the final example.
- Copy and paste the code of the final example, making sure to cite it properly in the top of the file.
- Modify the Runestone script in the following ways:
    - In the `drawLsystem()` function add in a new command, "`T`", which uses your `tridecagonTurtle()` to draw a new single tridecagon.
        - **NOTE**: You will likely have to modify your `tridecagonTurtle()` function to take in a turtle as an argument instead of using a turtle created in global space, or creating a brand new turtle inside the `tridecagonTurtle()` function itself. Look carefully at the `drawLsystem()` function and how it is called to see how to do this. There should

only be one (1) turtle created/ used in the final version - not several as you may have used in the past. This should be the turtle, called `t`, that is included in the Runestone code. You may copy/ paste your `tridecagonTurtle()` function into the `tridecagonLSystem.py` script, or you may call it from a module - either way is acceptable.
    - In the `drawLsystem()` function add in a new command, "`P`", which causes the turtle to be immediately placed in a new random location on the screen. There are a variety of potential solutions to this problem, so you may accomplish this however you wish - the turtle just needs to immediately move somewhere else on the screen when the "`P`" command is executed.
    - In the `applyRules()` function, change the rules to incorporate the new `T` and `P` commands. You output **must** utilize these new commands in some way
        - **NOTE**: You can change the original rule (`F-F++F-F`) to something else if you like.
- **NOTE**: The `main()` function prints out the list of instructions to the screen - the TAs should be able to see your new `T` and `P` commands in this printout.
- **NOTE**: You can feel free to change *any* of the default values in any of the function calls in the `main()` function. For example, maybe change the 'angle' from 60 to 75? 82? a totally random value each time?
    - In fact, you can feel free to change just about any aspect of the default Runestone script so long as it incorporates the two new commands. Use your imagination and have fun!
- Save your code, including your name, code creation date, lab number, and brief description of what your code does, to a file called `tridecagonLSystem.py`.

`reverseString.py` [thinkcspy 9.22]

- Use a `main()` function in the way demonstrated in class.
    - Ex: `if __name__ == "__main__":` etc.
- You will create two functions that check if a string is a palindrome or not.
    - **Version 1**:
        - Take a string as input from a user in the `main()` function and return that input to a variable.
        - Write a function called `reverseStringV1()` which takes the previously created variable as input.
        - This function should reverse the string using string slices and return the new string. This should be trivial.
            - Ex: "Hello" => "olleH"
            - **HINT**: Experiment with different values in the three 'zones' in the slicing operator: `string[START:STOP:STEP]`
        - Assign the output of your `reverseString()V1` function to a variable and print it out at the end of your `main()` function.
    - **Version 2**:
        - Take a string as input from a user in the `main()` function and return that input to a variable.
        - Write a function called `reverseStringV2()` which takes the previously created variable as input.
        - This function should reverse the string and return the new string **without using string slices**. Instead, build up a new string, character by character, and return that new string at the end of the function.
            - Ex: "Hello" => "olleH"
        - Assign the output of your `reverseString()V2` function to a variable and print it out

at the end of your `main()` function.
- Save your code, including your name, code creation date, lab number, and brief description of what your code does, to a file called `reverseString.py`.

`palindromeCheck.py` [thinkcspy 9.22]

- Use a `main()` function in the way demonstrated in class.
  - Ex: `if __name__ == "__main__":` etc.
- You will create two functions that check if a string is a palindrome or not.
  - **Version 1**:
    - Import your new `reverseString.py` module, created above, to gain access to its `reverseString()` function.
    - Take a string as input from a user in the `main()` function and return that input to a variable.
    - Write a function called `palindromeCheckV1()` which takes the previously created variable as input.
    - Inside `palindromeCheckV1()`, call one of your `reverseString()` functions from your `reverseString` module to help you determine if your original string is a palindrome or not. You can use this function to help you however you like.
  - **Version 2**:
    - Take a string as input from a user in the `main()` function and return that input to a variable.
    - Write a function called `palindromeCheckV2()` which takes the previously created variable as input.
    - Inside `palindromeCheckV2()`, iteratively compare the first character of your input string with the last character, the second character to the second to last character, etc., and use this information to determine of the input string is a palindrome or not.
- **NOTE**: A palindrome is a string that reads the same way forwards as backwards.
  - Ex: racecar, tacocat, etc.
- **NOTE**: Palindromes may or may not have a repeated 'middle' character.
  - Ex: abccba vs. abcba, etc.
- Save your code, including your name, code creation date, lab number, and brief description of what your code does, to a file called `palindromeCheck.py`.

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.

Lost Sacred Gems: The State of Indie Games in 2022 and Beyond

- Available: https://medium.com/super-jump/lost-sacred-gems-the-state-of-indie-games-in-2022-and-beyond-98332fd86d56

How to Become an Indie Game Developer in 2022

- Available: https://www.rokoko.com/insights/how-to-become-indie-game-developer

Generative AI to Become a $1.3 Trillion Market by 2032, Research Finds

- Available: https://www.bloomberg.com/company/press/generative-ai-to-become-a-1-3-trillion-market-by-2032-research-finds/

Landing a Job in Artificial Intelligence

- Available: https://www.computerscience.org/resources/landing-a-job-in-artificial-intelligence/

Files Provided

None

Example Script

myTest.py

```
# Matthew Holman                2-10-2023
# Lab Week 5 - A module to test another module's functions

import myFuncs

def getTwoInputs():
    a = int(input("Enter an integer: "))
    b = int(input("Enter an integer: "))
    return a, b

def main():
    running = True
    while running:
        choice = input("Choice?: [a]dd, [q]uit: ")
        if choice == "a":
            a, b = getTwoInputs()
            answer = myFuncs.add(a, b)
            print("The answer is: {0}".format(answer))
        elif choice == "q":
            print("Goodbye!")
            running = False
                else:
            print("ERROR: Please enter 'a' or 'q'")

if __name__ == "__main__":
    main()
```

myFuncs.py

```
# Matthew Holman                2-10-2023
# Lab Week 5 - A module to demonstrate functions

def add(a, b):
        return a + b
```

Example Output

Running: `python .\myTest.py`

```
Choice?: [a]dd, [q]uit: a
Enter an integer: 2
Enter an integer: 3
The answer is: 5
Choice?: [a]dd, [q]uit: asdf
ERROR: Please enter 'a' or 'q'
Choice? [a]dd, [q]uit: q
Goodbye!
```

Grading Items

- (**Attendance**) Did the student attend the lab meeting, or make arrangements to attend virtually via WebEx?: _____ / 1
- (**Reading**) Has the student read chapter 9 of the Runestone textbook and shown their notes in their Engineering Notebook to the TA?: _____ / 2
- (**`tridecagonLSystem.py`**) Has the student completed the task above, and saved their work to a file called `tridecagonLSystem.py`?: _____ / 3
- (**`reverseString.py`**) Has the student completed the task above, and saved their work to a file called `reverseString.py`?: _____ / 2
- (**`palindromeCheck.py`**) Has the student completed the task above, and saved their work to a file called `palindromeCheck.py`?: _____ / 2

TOTAL _____ / 10