Lab Week 7 Grading Rubric and Instructions

This lab is assigned for Week 7 of COM S 1270: Introduction to Computer Programming.

This lab is due by the end of the lab period six (6) days after the one it is assigned in. Please see the syllabus for details.

Lab Objective

The purpose of this lab is to give you more in-depth practice working with strings.

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: 2-26-2023
- 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]

Reading:

- 'Check off' your notes in your Engineering Notebook for Runestone chapter 9 with the TA. This should already be done before the start of the lab period.
    - **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.

`reverseString.py`: [thinkcspy 9.22]

- Use a `main()` function in the way demonstrated in class.
    - Ex: `if __name__ == "__main__":` etc.
- When a string is reversed, its characters print out in the opposite order of their original configuration.
    - Ex: `APPLE CAT!` becomes `!TAC ELPPA`
- You will create several functions that reverse a string.
    - **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 `reverseStringV1()` 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, use the built in `.reversed()` function. Please note that this built-in function will return a list of characters, so you will need to join all of those characters together to form the new string.
            - Ex: "Hello" => "olleH"
        - Assign the output of your `reverseStringV2()` function to a variable and print it out at the end of your `main()` function.
    - **Version 3**:
        - Take a string as input from a user in the `main()` function and return that input to a variable.
        - Write a function called `reverseStringV3()` which takes the previously created variable as input.
        - This function should reverse the string and return the new string **without using string slices or the built in `reversed()` function**. Instead, use a `for` loop to build up the string character by character. Make use of the `range()` function to access the individual elements of the original string by their index.
            - Ex: "Hello" => "olleH"
        - Assign the output of your `reverseStringV3()` function to a variable and print it out at the end of your `main()` function.
    - **Version 4**:
        - Take a string as input from a user in the `main()` function and return that input to a variable.
        - Write a function called `reverseStringV4()` which takes the previously created variable as input.
        - This function should reverse the string and return the new string **without using string slices or the built in `reversed()` function**. Instead, use a `for` loop to build up the string character by character. Do *not* make use of the `range()` function, but rather access each element of the original string directly.
            - Ex: "Hello" => "olleH"
        - Assign the output of your `reverseStringV4()` function to a variable and print it out at the end of your `main()` function.
    - **Version 5**:
        - Take a string as input from a user in the `main()` function and return that input to a variable.

        - Write a function called `reverseStringV5()` which takes the previously created variable as input.
        - This function should reverse the string and return the new string **without using string slices or the built in `reversed()` function**. Instead, use a `while` loop to build up the string character by character.
            - Ex: "Hello" => "olleH"
        - Assign the output of your `reverseStringV5()` 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.
- A string is a palindrome if its characters read forwards are the same as the characters read backwards.
    - Ex: `TACOCAT` *is* a palindrome, but `PIZZACAT` is not.
- You will create several 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 `reverseStringVX()` functions.
        - 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 `reverseStringVX()` 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.
        - Assign the output of your `palindromeCheckV1()` 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 `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.
        - Assign the output of your `palindromeCheckV2()` function to a variable and print it out at the end of your `main()` function.
- **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`.

`findSubString.py`: [MH]

- Use a `main()` function in the way demonstrated in class.
    - Ex: `if __name__ == "__main__":` etc.

- A string contains a given substring if a subset of its characters exactly match the configuration of the substring. In a typical implementation, a function that searches for substrings returns the index of the first character of the first instance of the substring in the original string. If the substring is not found, a typical substring search function returns -1.
    - Ex: If the string is `APPLE CAT!` and the substring is `CAT`, then the function returns `6` because this is the index of the first character.
    - Ex: If the string is `APPLE CAT!` and the substring is `PIZZA`, then the function returns `-1` because the original string does not contain the substring.
- You will create several functions that check if a string contains a certain substring.
    - **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 `findSubStringV1()` which takes the previously created variable as input.
        - Inside `findSubStringV1()`, use the built in string `.find()` method to find the index of the substring. Return the index of the substring, or -1 if the substring is not inside the original string.
        - Assign the output of your `findSubStringV1()` 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 `findSubStringV2()` which takes the previously created variable as input.
        - Inside `findSubStringV2()`, use the built in string `.index()` method to find the index of the substring. You will need to find a way to stop the code from crashing if the substring is not found. Return the index of the substring, or -1 if the substring is not inside the original string.
            - **NOTE**: Do not use the `try`/ `except` keywords to solve the crashing issue.
        - Assign the output of your `findSubStringV2()` function to a variable and print it out at the end of your `main()` function.
    - **Version 3**:
        - Take a string as input from a user in the `main()` function and return that input to a variable.
        - Write a function called `findSubStringV3()` which takes the previously created variable as input.
        - Inside `findSubStringV3()`, iteratively compare the characters of the original string with the characters of the substring until you find a match between the two. Return the index of the substring, or -1 if the substring is not inside the original string.
        - Assign the output of your `findSubStringV3()` 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 `findSubString.py`.

`tridecagonLSystem.py`: [thinkcspy 9.22]

- Re-read chapter 9.15 in the Runestone textbook, and make sure you really understand the code of the final example (9.15.2).
- Copy and paste the code of the final example (9.15.2), making sure to cite it properly in the top of the file.

- Import your `tridecagonTurtle` module from a previous lab so that you can access its `tridecagonTurtle()` function.
- Modify the Runestone script in the following ways:
    - In the `drawLsystem()` function add in a new command, "`H`", which uses your `tridecagonTurtle()` function from previous labs to draw a new single tridecagon.
        - **NOTE**: Look carefully at the `drawLsystem()` function and how it is called to see how to complete this task. There should only be one (1) turtle created/ used in the final version. This should be the turtle, called `t`, that is included in the Runestone code.
    - 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. Your 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!
- **NOTE**: Unless you go through all of the original code very carefully, and fully understand how it works, this will be a very difficult task. Indeed, if you ask the instructor or a TA for help on this task, you will immediately be asked how the original code works. If you cannot adequately explain it, then that is what you will need to work on first until you can explain it.
- 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`.

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.

Machine learning, explained - by: Sara Brown - April 21, 2021

- Available: https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained

Evolutionary computation - by: engati.com

- Available: https://www.engati.com/glossary/evolutionary-computation

How To Get a Job in Computer Science (With Job Salaries) - by: Indeed Editorial Team - March 16, 2023

- Available: https://www.indeed.com/career-advice/finding-a-job/job-in-computer-science

Making Video Games Is Not a Dream Job - by: wired.com - May 14, 2021

- Available: https://www.wired.com/2021/05/geeks-guide-jason-schreier/

Files Provided

None

Example Script

None

Example Output

None

Grading Items

- (**Reading**) Has the student read chapter 9 of the Runestone textbook and shown their notes in their Engineering Notebook to the TA?: _____ / 10
- (**reverseString.py**) Has the student completed the task above, and saved their work to a file called `reverseString.py`?: _____ / 20
- (**palindromeCheck.py**) Has the student completed the task above, and saved their work to a file called `palindromeCheck.py`?: _____ / 10
- (**findSubString.py**) Has the student completed the task above, and saved their work to a file called `findSubString.py`?: _____ / 10
- (**tridecagonLSystem.py**) Has the student completed the task above, and saved their work to a file called `tridecagonLSystem.py`?: _____ / 30
- (**Attendance**) Did the student attend the full lab meeting in person, or did they attend the full lab meeting virtually via WebEx?: _____ / 20

TOTAL _____ / 100