Lab 01 Specification – A Hand-on Exercise to practice Computational Constructs
50 points

Due by: 03/17/2021 (2-weeks) 8:00 AM

Lab Goals

- Quick review of GitHub and git commands.
- Refresh your ability to write good code in Python.
- Think about how to solve a variety of programmatic challenges.

Learning Assignment

If you have not done so already, please read all of the relevant "GitHub Guides", available at the following website:
`https://guides.github.com/`
that explains how to use many of the features that GitHub provides. This reading will help you to understand how to use both GitHub and GitHub Classroom. To do well on this assignment, you should also read:

- **GT chapter 01**

Assignment Details

As announced earlier, we don't have a lab session next week, 03/10/2021, due to the college break. This is taken into consideration in our class scheduling. This lab should be completed within two weeks. Please make sure to push your code file(s) and commit by the deadline provided.

Now that we have discussed some basics of computational constructs and implemented a few Python programs together in the last few lectures, it is your turn. In this lab, you will practice a variety of programs to retain the knowledge of conditional and iterative constructs. This includes modifying one or more code files to implement a series of functionalities. At any duration during and/or after the lab, students are recommended to team up with the Professor and the TL(s) to clarify if there is any confusion related to the lab and/or class materials.

Students are recommended to get started with this part in the laboratory session, by discussing ideas and clarifying with the Professor and the Technical Leader(s). It is acceptable to discuss high-level ideas with your peers, while all the work should be done individually. Late submission is accepted for the part(s) in this section, based on the late policy outlined in the course syllabus.

It is required for all students to follow the honor code. Some important points from the class honor code are outlined below for your reference:

1. Students are not allowed to share code files and/or other implementation details. It is acceptable to have a healthy discussion with your peers. However, this discussion should be limited to sharing ideas only.
2. Submitting a copy of the other's program(s) is strictly not allowed. Please note that all work done during lab sessions will be an opportunity for students to learn, practice, and master the materials taught in this course. By doing the work individually, students maximize the learning and increase the chances to do well in other assessments such as skill test, exams, etc · · ·

Part 01 - A Simple Exercise to Implement Basic Computational Constructs (20 points)

Write a Python program to implement the `Currency Exchange Rate Converter` program, using a series of requirements outlined below.

1. For simplicity, let us make the `Currency Exchange Rate Converter` program work for four countries and the conversion table is shown below:

From Currency	To Currency	Amount
INR	USD	0.014
INR	CNY	0.097
INR	JPY	1.55
USD	INR	71.04
USD	CNY	6.87
USD	JPY	110.18
CNY	USD	0.15
CNY	INR	10.35
CNY	JPY	16.05
JPY	USD	0.0091
JPY	INR	0.64
JPY	CNY	0.062

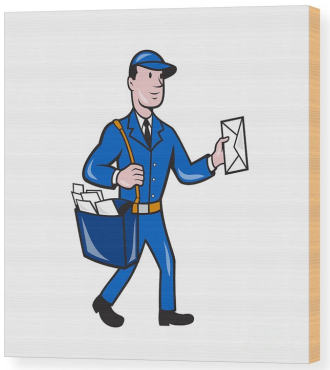
Table 1: Exchange Rate Table

2. The starter code is provided inside the lab repository in a file named, `currency.py`.
3. First, the program should greet the user with a welcome message. This is already implemented in the starter code file to get started and understand the other requirements outlined in this section.
4. After displaying the greeting message, the program should prompt the user to enter the from and to currency. To make this section easier to understand, this step is already implemented in the starter code as well.
5. Implement the exchange rate conversion rules based on the table provided in this section. The implementation should be constructed using if, else if, and else conditions.
6. For each of the conditions implemented in the previous step, implement a series of programming statements to formulate the rules for conversion from one currency to another.
7. Do some basic data validation procedure(s) to correctly identify invalid inputs. For example, if user-provided input for both currencies is `1`, then an invalid input message should be displayed to the user. In general, it is acceptable to assume that the user-provided input for the currency specification is always a number between 1 to 4 (inclusive).

8. A screenshot displayed on the next page shows the welcome greeting message, and the series of user prompts, in the starter code file. Note: your goal should be to display the right amount that the user would receive based on the logic implemented in the calculator.

```
amohan@amohanmacpro code % python3 currency.py
Welcome to currency exchange rate calculator ....
-------------------------------------------------
By using the following options to define the currency:
        1 for chinese yuan
        2 for japanese yen
        3 for indian rupee
        4 for united states dollar
Enter the currency that you have by using a number between 1 and 4:1
Enter the currency that you want by using a number between 1 and 4:2
Enter the amount to be converted:100
You will receive: 1605
Thank you for using our tool, goodbye!
amohan@amohanmacpro code %
```

Part 02 - A Postman App (20 points)

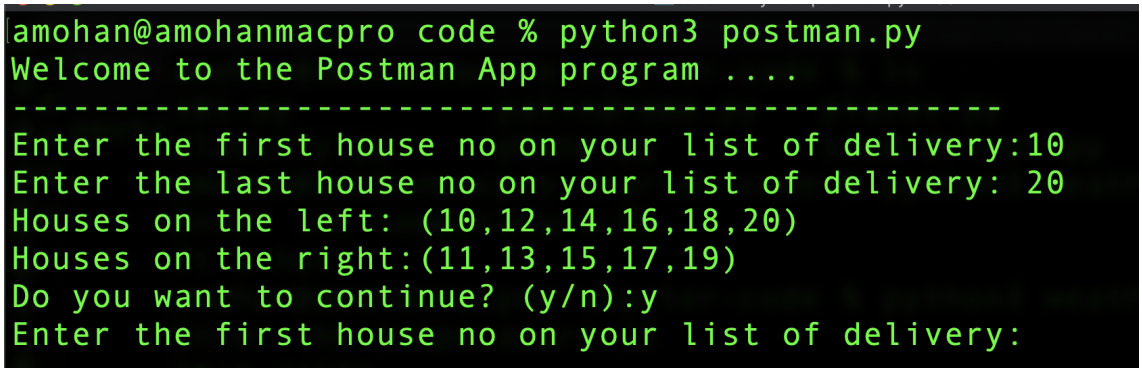


Write a Python program to implement a `Postman App` program, using a series of requirements outlined below.

1. The starter code is provided inside the lab repository in a file named, `postman.py`.
2. First, the program should greet the user with a welcome message. This is already implemented in the starter code file to get started and understand the other requirements outlined in this section.
3. Implement a repetitive block using a `while loop` to ask the postman to enter the first and last house nos in the postman's delivery list. To make this section easier to understand, this step is already implemented in the starter code as well.
4. Based on a simple thumb rule, the odd house no's are located on the right side of the street and the even house no's are located on the left side of the street. The goal of the program is to display the list of houses that are located on the left and right side of the street between the house nos provided **(inclusive)**.
5. So how do we identify if a given house number is odd or even? As discussed in the class discussion last time, there exists an operator known as modulo (%) in Python which allows one to easily detect the odd and even number in the program.
6. The program will repeat the steps to guide the postman by providing the list of houses on the left and right side of the street. The program will exit if the postman specifies there is no more guidance needed. This part is already implemented in the starter code.

7. The program should do some basic data validation procedure(s) to correctly identify invalid inputs. For example, if the postman provided inputs for the first house no is greater than the last house no, then an invalid input message should be displayed on the console. We assume that the first house no and end house no cannot be the same.

8. A screenshot displayed in next page shows the welcome greeting message, the user prompt, and a sample execution result of the program to detect the house location.



Part 03 - To Solve (5 points)

An important part of programming is solving problems by code tracing. By now, we had implemented multiple programming solutions that are connected to conditional and iterative blocks. Solve the problems provided in the `code-tracing.md` file. The programs to analyze, and apply the code tracing technique are provided in the `tracing` folder. The solution to the questions listed should be provided as instructed in the **code-tracing.md**.

Part 04 - To Think (5 points)

Another important part of programming is to develop thinking skills. By now, we had implemented one interesting idea connected to the life of a Postman. Think and come up with ideas to extend the program developed in the earlier parts of this section. The ideas should enrich and enhance what had been implemented already. Include a summary of one or more ideas in a file named **ideas.md**.

Part 05 - Honor Code

Make sure to **Sign** the following statement in the `honor-code.txt` file in your repository. To sign your name, simply replace Student Name with your name. The lab work will not be graded unless the honor code file is signed by you.

This work is mine unless otherwise cited - Student Name

Submission Details

For this assignment, please submit the following to your GitHub repository by using the link shared to you by the Professor:

1. Commented source code from the “currency.py” program.
2. Commented source code from the “postman.py” program.
3. A document containing the solution to the computational problems, in a file named `code-tracing.md`.
4. A document containing the ideas to enrich the functionality implemented in the Postman App named `ideas.md`.
5. A signed honor code file, named `honor-code.txt`.

6. To reiterate, it is highly important, for you to meet the honor code standards provided by the college. The honor code policy can be accessed through the course syllabus.

Grading Rubric

1. There will be full points awarded for the lab if all the requirements in the lab specification are correctly implemented. Partial credits will be awarded if deemed appropriate.

2. Failure to upload the lab assignment code to your git repo will lead you to receive no points given for the lab submission. In this case, there is no solid base to grade the work.

3. There will be no partial credit awarded if your code doesn't compile correctly. It is highly recommended to validate if the correct version of the code is being submitted before the due date and make sure to follow the honor code policy described in the syllabus. If it is a late submission, then it is the student's responsibility to let the professor know about it after the final submission in GitHub. In this way, an updated version of the student's submission will be used for grading. If the student did not communicate about the late submission, then automatically, the most updated version before the submission deadline will be used for grading purposes. If the student had not submitted any code, then, in this case, there are no points awarded to the student.

4. If you need any clarification on your lab grade, talk to the Professor. The lab grade may be changed if deemed appropriate.