Computer Science AP/X — CSCI-140/242
Secret Messages — Lab 1

8/2/2021

1 Requirements

You will individually implement a program, `transformer.py`, which encrypts/decrypts messages. The program will read in from the standard input and print the results in the standard output.

The program should execute as follows:

1. First prompt for the operation to perform. The options will be `E` for encryption (apply the transformation operations to the message), `D` for decryption (the given transformations have already been applied to generate the message), or `Q` for terminating the program.
2. If the user enters `Q`, the program finishes.
3. Otherwise, prompt for the message string.
4. Finally, prompt for the transformation operations string.
5. The program will send the output for the encryption/decryption process to the standard output.
6. Repeat step 1.

Recall from the in-lab writeup:

The format of the messages will be all upper-case letters (no spaces or punctuation). The format of the transformation strings will be as follows:

Table 1: Transformation operations

Operation	String form	Example	
S_i	`Si`	S_0	`S0`
S_i^k	`Si,k`	S_2^{-5}	`S2,-5`
R	`R`	R	`R`
R^i	`Ri`	R^{-3}	`R-3`
D_i	`Di`	D_2	`D2`
D_i^k	`Di,k`	D_2^3	`D2,3`
$T_{i,j}$	`Ti,j`	$T_{2,4}$	`T2,4`

If a series of transformations are to be supplied, they will be separated by semicolons. So, for example, if you were asked to encrypt the string HORSE given the transformation string `T2,4;S4;R` you would generate the string SHOES.

1.1 Sample Run

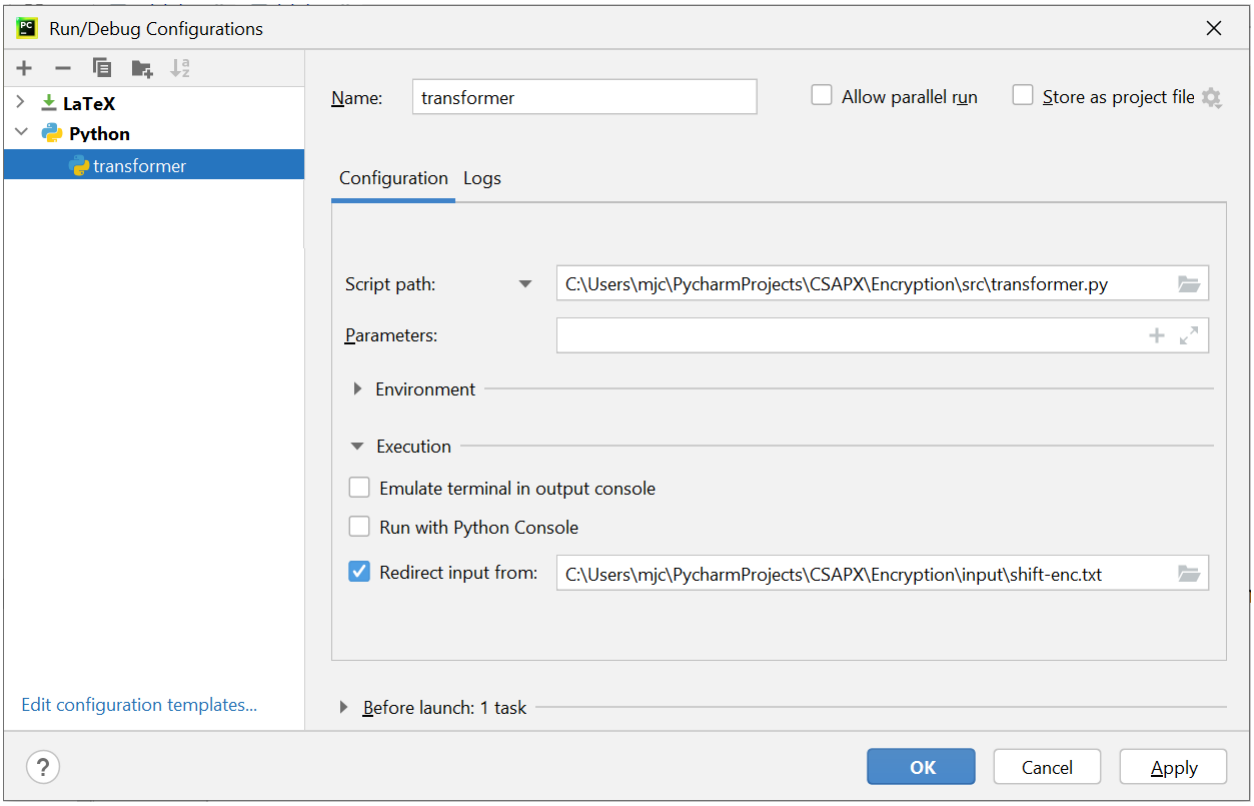
You have been provided with some sample files to verify the correctness of your solution. The files in the `input` folder contain the input required to run the program and the files in the `output` folder contain the output generated by your program when executed with its corresponding file from the `input` folder.

We have provided you with files to verify every operation individually and in combination with other operations. However, we encourage you to create your own test cases for a more thorough testing.

1.2 Simplifying Input

In order to save you the pain of having to enter into the console all the information the program requires to run, you can edit the run configuration to automatically redirect the input from a file. To do this, the run configuration pull down menu (left of the green run arrow) and select **Edit configurations**.

For the `transformer` run configuration, you can select the checkbox near the bottom for **Redirect input from**, and specify the full path to any file with the required input. For example, you can run your program redirecting the input from any file from the `input` folder.

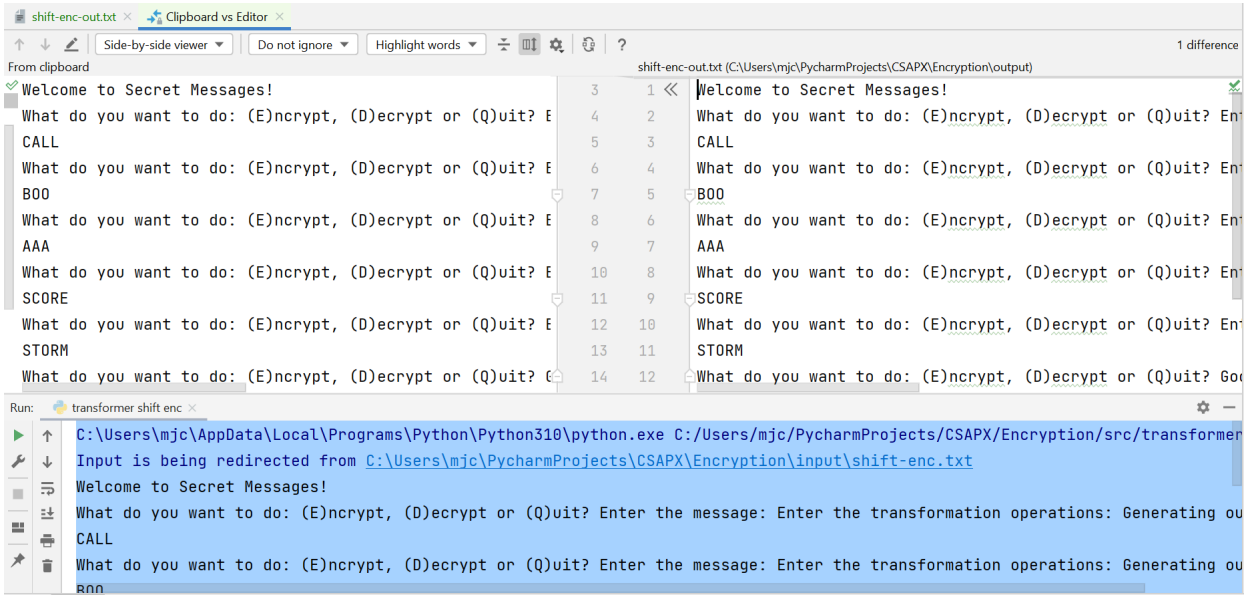


Now when your program executes an `input()` call it will read from the file versus asking you for input.

1.3 Comparing Output

The supplied outputs have a lot of text and can be very confusing to compare your output to. PyCharm provides a utility so you can compare a text file to the clipboard in a visual manner.

1. Open one of the output files in the Pycharm editor window.
2. Run the program with your desired input file.
3. Select all the text in the console and copy it to the clipboard.
4. In the solution output editor window right click and select **Compare with Clipboard**.
5. A new editor window will open highlighting the differences if there is any.



The ultimate goal here would be to make sure the output matches exactly.

2 Implementation Details

You are not allowed to use regular expressions for this lab. Everything should be done using `str` operations, e.g. slicing, concatenation, indexing, etc.

Additionally, you are not allowed to use the `global` keyword.

In order to receive full design points, you should be using functions to break down your program. For example, you should have one function that parses each transformation operations string into the sequence of transformations, and a function for every transformation. Look at your problem solving for more hints about how you can break this down to promote function reuse.

Your program must be properly documented to receive full style credit. The program should have a main docstring containing your name and a description of the assignment. Each function should have a properly formatted docstring with a description, arguments, return, etc.

You do not have to deal with erroneous input. It is assumed the user will provide valid input. The messages and operations will be legal, and are all formatted correctly.

3 Grading

The grade breakdown for this lab is as follows:

- Problem Solving: 15%
- In-Lab Activity: 10%
- Functionality: 55%
  - User input: 5%
  - Individual encryption operations: 20%
  - Overall encryption process: 15%
  - Decryption process: 15%
- Design: 10% - Your implementation uses functions to promote code reuse.
- Code Style and Documentation: 10%

4 Submission

Go to your project's `src` folder and zip it up. Rename the zip file to `lab1.zip`. Upload this zip to the MyCourses Assignment dropbox by the due date.

- To zip on Windows, right click on the `src` folder and select `Send to -> Compressed (zipped) folder`.
- To zip on MacOS, right click on the `src` folder and select `Compress "src"`.