CS 171
Week 4 Lab

Overview

This lab consists of a collection of problems that you may solve in any order. To earn full credit, you must submit solutions to all problems at one time.

Important:

- You must be present in the lab session to earn credit
    - If you are recorded as absent but still complete the work, you will not earn credit. If you arrived after the CA/TA took attendance, be sure to check in so you are recorded as present/late.
    - If you miss the lab meeting, contact your instructor via email right away to request a make-up session.
- You may collaborate with at most one lab partner; Feel free to switch partners next week if you wish
- Each student must submit their own work to Gradescope for credit, even if you worked with a partner.
- At most 100 points are possible. To get a combined score, submit all of your solutions at once.
- Please only use features of the language that have been covered in reading and lectures.
    - Solutions that make use of language features not yet covered will not receive credit.
    - Students whose submissions repeatedly make use of additional skills not covered in the course so far may be suspected of violating academic integrity.
    - *Even if you know the language already, please work within the subset of the language covered up to this point in the course.*
- You may not use artificial intelligence tools to solve these problems.
- Your lab work is due at the end of the lab period. Late work will be accepted with a 30-point penalty if it is submitted within 12 hours of the end of your lab.

Helpful to Know:

- You may work these problems in any order
- Lab work will be graded by the auto-grader
- The last score earned will be recorded for your lab score
- To get a combined lab score, you must submit multiple files at the same time
- Lab assistants are here for your support. Ask them questions if you are stuck!

Materials You Are Allowed To Use:

- Lecture slides for Week 1 - 4
- Think Python Chapters 1 – 8

Problem A

Submission File: `lab04a.py`

ROT-13 (short for "rotate by 13 places") is a simple substitution cipher used to encode text. It works by shifting each letter of the alphabet 13 places forward. For example, 'A' becomes 'N', 'B' becomes 'O', and so on, wrapping around to the beginning of the alphabet if necessary. This means that applying ROT-13 twice will return the original text, making it a symmetric encryption method. It's often used for obscuring text in online forums or puzzles, as it provides a quick way to hide spoilers or sensitive information without requiring complex encryption algorithms. However, it's not secure for serious encryption purposes since the transformation is easily reversible.

Part 1

Write a function, `rotateChar(…)`, which receives a string containing a single character. If the character is an uppercase or lowercase character (A-Z or a-z), the function returns the character that is 13 characters away from it in the alphabet, looping around if necessary. If the character is not an uppercase or lowercase character, the function returns the character without modifying it.

Examples:

- `rotateChar('A')` returns `'N'`
- `rotateChar('@')` returns `'@'`
- `rotateChar('x')` returns `'k'`

Part 2

Write a function, `rotateString(…)`, which receives a string containing any number of characters. It should use your `rotateChar(…)` function to encode each character in the input string, returning the fully encoded string as a result.

Examples:

- `rotateString('Hello World!')` returns `'Uryyb Jbeyq!'`
- `rotateString('abc123@drexel.edu')` returns `'nop123@qerkry.rqh'`
- `rotateString('Guvf zrffntr jvyy frys-qrfgehpg')` returns `'This message will self-destruct'`

Important Note: Use only programming constructs and skills presented in **Lectures 1 – 4** and **Chapters 1 – 8**.

Problem B

Submission File: `lab04b.py`

Part 1

Write a function, `inOrder(…)`, which receives a string as a parameter, and returns True if the characters in the text appear in ascending order (in alphabetical order from low to high) or False otherwise.

Ex: If the text passed to the inOrder() function is "DREXEL", then the function returns False.
Ex: If the text passed to the inOrder() function is "ALMOST", then the function returns True.

Notes:

- *Use a while loop. DO NOT use sorted() or sort().*
- *You may assume that entries will be either ALL UPPER CASE or all lower case.*

Part 2

Use your function to write a program that reads a single line of text from the keyboard (no prompt required). The program prints "In ascending order" if the characters are sorted, or "Not in order" if the text is not sorted.

Problem C

Submission File: `lab04c.py`

Write a program that reads a loan balance, monthly payment amount, and monthly interest rate as float inputs and outputs the number of payments required until the loan is paid off. Interest is added to current balance *before* a payment is applied. For instance, if the current balance is $100.00 and the interest rate is 0.02, the new balance is $102.00 before a payment is applied. All values are floats.

Ex: If the input is:

```
1000.0
50.0
0.03
```

the output is:

```
31 payments
```

Ex: If the input is:

```
1000.0
1050.0
0.02
```

the output is:

```
1 payment
```

Your program should read input from the keyboard and print output to the console. While you may solve this problem using a function, you are not required to write any specific function for this problem.