Intermediate Python Programming

Description: Conditionals, Loops, String Manipulation and Basic Mathematical Operations

Instructions

- Please follow all instructions closely. There is no penalty for going “above and beyond” the assignment requirements. However, failure to meet all required parts can affect your grade. Please consult the rubric for each section.
- These programs can be completed as “raw” python “*.py” files in the editor of your choice, or by using Jupyter Notebook. Please ensure that all necessary “*.py” and “*.ipynb” files are included with your submission. Missing files will delay grading and possibly adversely affect your score.
- You may complete all parts in a single file or as individual files. If you submit multiple files, please ensure that the file names indicate their contents. Example: Lab1_Part1.py
- It is not necessary to include your name in the Python script filename. In fact, please do not do that unless specifically instructed to, as the filename is the Python module name. Instead, please include a comment at the top of each script file like this:

```
# Python II – Lab 1 – Bill Smith
```

- Extra credit is just that, extra credit. If you choose to submit extra credit ensure it is complete and does not break the program as that can adversely affect the base score.

Part I

Create a python script to determine if a user-entered string is a palindrome. A palindrome is text that reads the same forwards and backwards. Examples:

MOM

DAD

LEVEL

REDIVIDER

Requirements:

- The text to be scanned must be user-entered. Suggestion: input() function.
- You must use a loop to scan the characters
- Do not use a prebuilt “reverse” function.

Sample Run (User-entered data in RED):

```
Enter a string: REDIVIDER
Is 'REDIVIDER' a palindrome? True
```

Scoring:

Criteria	**Points**
Properly handles user input	5
Proper loop implementation (no overflows/all indices visited)	5
Successful palindrome detection	10
Appropriate output	5
TOTAL	**25**
EXTRA CREDIT	
Text comparison is case insensitive (A == a)	3
Handles sentence-long palindromes (whitespace ignored)	3
Example: Was it a car or a cat I saw	

Part II

Create a Python script that grades the strength of a password.

Requirements:

- You will prompt the user for a password. Do not worry about "hiding" the input text, this is just an academic exercise.
- The score is an integer value between 0 and 5, where 0 is a weak password 5 is a strong password.
- For each of the following conditions, add 1 to the score if true:
  - Password length is greater than or equal to 8
  - The password contains at least one upper-case letter
  - The password contains at least one lower-case letter
  - The password contains at least one numeric digit (0-9)
  - The password contains one of the following symbols: !@#$%^&*
- Use a loop to "visit" all of the characters (do not use regular expressions or other pattern matching libraries)
- Indicate the score as output

Sample Run (User-entered data in RED):

```
Enter a password: Cougar1!
Your password score is: 5
```

Scoring:

Criteria	Points
Properly handles user input	5
Proper loop implementation (no overflows/all indices visited)	5
Correct scoring for each criteria	20
Appropriate output	5
TOTAL	**35**
EXTRA CREDIT	
Give a bonus point for password length > 8	2
Give an additional point for password length >= 16	2

Part III

Create a python script to calculate compounded interest for given principal amount using a given interest rate and duration. For this assignment you may assume only good data will be entered, so you do not need to worry about "ABC" being entered for a numeric value.

Requirements:

- You will prompt the user for three pieces of information: principal amount (beginning balance), interest rate and term (in years).
- The principal amount will be parsed as a float and be between the values of 1 and 1,000,000. Two decimals may also be given. Suggested variable name: *principal*
- The percentage rate will be parsed as a float and be between the values of 0 and 100. Two decimals may also be given. Suggested variable name: *apr*
- The term will be parsed as an int and be between the values 1 and 30. Suggested variable name: *term*
- The interest calculation will be compounded annual interest. In other words, after each year, the yearly interest earned is calculated by multiplying the interest rate by the balance. This interest amount is then added back to the balance before the next year's interest is calculated. This is what causes the compounding effect.
- Use a for loop to iterate from 0 to term length
- Interest should be entered as 4.5 for 4.5%. Therefore, make sure to divide that number by 100.0 before multiplying with the balance, otherwise the interest calculation will be HUGE.
- The output must be in tabular form, with each year in the term clearly identified.
  - Hint: use "f" string formatting to get the tabular layout.
  - Example below is formatted in US units – use the appropriate formatting for your locale.
  - See the very end of the document for hints regarding the formatting of the number (try to get it on your own first!)

Sample Run (User-entered data in RED):

```
Enter the principal amount (ex: 1000.00): 100000
Enter interest rate percentage (ex: 4.5): 4.5
Enter term in years (ex: 10): 10
Year        Interest         Balance
==================================
   1  $     4,500.00  $  104,500.00
   2  $     4,702.50  $  109,202.50
   3  $     4,914.11  $  114,116.61
   4  $     5,135.25  $  119,251.86
   5  $     5,366.33  $  124,618.19
   6  $     5,607.82  $  130,226.01
   7  $     5,860.17  $  136,086.18
   8  $     6,123.88  $  142,210.06
   9  $     6,399.45  $  148,609.51
  10  $     6,687.43  $  155,296.94
```

Scoring:

Criteria	**Points**
Properly handles user input	5
Proper parsing of input data into variables	5
Proper loop creation (no overflows and all indices visited)	5
Correct interest calculation	10
Tabular layout for output	
Header	3
Value justification (numeric values right justified)	3
Values display appropriate number of decimal places (2)	3
Values have “thousands” separators	3
All values line up in vertical columns	3
TOTAL	**40**

Hints and Tips:

- You can check if a specified character is contained in another string with the following syntax:
```
if ch in "!@#$%^&*":
     # do something here
```
- The numeric formatting seen in the compounded interest example can be achieved with the following format specifiers in an f-string (assume "interest" is a float variable): `{interest:>12,.2f}`
  - Where:
    - *interest* is the variable to format
    - *>12* justifies the value to the right using 12 total characters
    - *,* uses a comma for the thousands separator
    - *.2f* formats as a floating point number with 2 total decimal places