CSE5APG
Week 4: Lab04

OBJECTIVES

- Implement Input, Process and Output steps.
- Learn how to use Repetitive Execution.
- Learn how to define and mainpluate Strings
- Learn how to define and use Files

---- Repetitive Execution ----

a. Use **for-loop** to print Pascal's triangle as a as follows.

```
0
111
22222
3333333
444444444
```

b. Use **for-loop** to print table that includes 6 rows and 6 columns where the dignal should be "X".

```
X 1 1 1 1 1
2 X 2 2 2 2
3 3 X 3 3 3
4 4 4 X 4 4
5 5 5 5 X 5
6 6 6 6 6 X
```

c. Use **for-loop** to print lower triangle (dignal) of 5x5 as a '*' pattern as follows:

```
*
* *
* * *
* * * *
* * * * *
```

d. The code below prints the numbers from 1 to 56. Rewrite the code using a **while-loop**.

```
for i in range(1,56):
    print(i)
```

e. Write a program that uses a **while-loop** to read a string from user and then print the characters of the string one-by-one on separate lines.

f. Write a program that asks the user for a weight and converts it from kilograms to pounds (pound= kilograms /0.45359237). If the user key in a weight below **0**, the program should tell the user that their entry is invalid and then ask the user to enter a new weight. The program should terminate if the user enter **-100**. [Hint: Use a while loop only, not an if statement].

---- Strings----

Question 1 – Review

a. How do you define a string (string literal) that has one or more single quotes?

   e.g. **Walk. Don't run.**

b. How do you define a string that has one or more single quotes and one or more double quotes?

   e.g. **The sign says, "Don't walk!"**

c. How do you define a string that contains two or more lines of text?

   e.g. **Walk.**
   Keep walking.
   Stop!

d. Why does this expression "number " + 9 give a run-time error? How to fix it?

e. How do you test if a string **s** contains a substring **sub**?

f. What string function allows you to count how many times a substring **sub** appears in a string **s**?

g. Given a string called **name**. How to test if the **name** starts with two underscores?

h. Given a non-empty string. How do you test that it contains only letters?

i. Given a string. How to remove all the leading and trailing whitespace characters?

j. Given a string **s**. How do you get a string the same as **s** but with all the lower-case letters converted to upper case?

Question 2 – The First, the Last and the Middle

Write a program that asks for a string, and then

- Display the first character
- Display the last character

- If the string has an odd number of characters, display the middle character. Otherwise, display the two middle ones (e.g. for **beauty**, display **au**).

Question 3 – Negative Indexes

Python allows negative indexes. Given a negative index **-n**, the equivalent positive index is **len(s)-n**. For example, the positive index corresponding to **-3** is **len(s)-3**.

Assume that a string contains two characters or more. Use a negative index to display

- The last character of **s**
- The second last character of **s**
- The substring of **s,** which excludes the first and the last character.

Question 4 – Displaying Phone Number

Write a program that asks the user for a string of a 10-digit phone number and prints it out in a readable format, as shown in the example below.

Input: **0394791234**

Output: **(03)9479-1234**

Hint: You can use the indexes in the table below in working out the solution:

string	0	3	9	4	7	9	1	2	3	4
index	0	1	2	3	4	5	6	7	8	9

Question 5 – Masking the Characters

Write a program that asks for a string **s** of length 4 or more. The program then displays a string that has the first two characters of **s** and the last character, with each of the characters in between replaced by a star *****.

Sample:

Input: **Strings**

Output: **St****s**

Question 6 – Swapping Dots and Commas

Write a program that asks for a string **s**. The program then displays the string **s** but with all the dots replaced by commas and vice versa.

Sample: Input: **123, 56.78**

output: **123.456, 78**

You can assume that string **s** does not contain substring <<dot>>

--- File Input-Output ---

Question 1- Review

a.

- What are the main steps we follow in writing text to a file?
- What are the functions/methods we use for these steps?
- Illustrate the writing process with a simple but complete example.

b. Suppose we open a file for writing. What happens if the file does not exist? What happens if the file already exists?

c.

- What are the main steps we follow in reading text from a file?
- What are the functions/methods we use for these steps?
- Illustrate the reading process with a simple example in which we read the entire contents of the file into a string.

d.

- How do we open a file to append text to?
- What happens if the file does not exist?
- What happens if the file already exists?

Question 2 - Copy Files

a. Write a program to copy a file to another file. Let the input file be **twinkle.txt** (provided), and the output file is **twinkle_copy1.txt**.

b. Write a program to copy a file to another file, with each line of text being preceded by the line number, followed by a colon. Start the line numbers at 1. Let the input file be **twinkle.txt** and the output file be **twinkle_copy2.txt**

Question 3 - Merge Files

Write a program to merge the two files. The files **names set1.txt** and **names set2.txt** are provided. Each file has 30 names, each of which is on a line of its own. The names in each file are sorted. The two files have names in common. The program is to merge the two files and save the merged list in the output file **names merged.txt**.

Question 4 - Finding Common Words

Write a program that reads two files and print out all the words they have in common. For this question, take a word as a sequence of characters separated by whitespace characters. For a quick test, you can take **names set1.txt** and **names set2.txt** as input files. Print out all the words they have in common.

Question 5 -Word Frequencies

Write a program to read a text file and print the frequencies (how many times this word occurs) of each word. To keep things simple, take all the words to be in lower case. We will take a word as a sequence of *letters* separated by *whitespace character* and the following punctuation marks: *comma, colon, semi-colon, period, question mark, exclamation mark*, and *quotation marks*. For a quick test, use the provided file **why.txt** for input.