**Week 4 In-Class Exercises**

**(More on Conditions, Strings and for-Loops)**

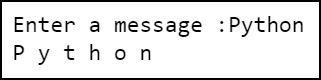
**Note:** Download the Jupyter notebook “Week 4 In-Class Exercises Starting Code (v1.0).ipynb” (which contains the starting code for the exercises below) from eLearn.

**Q1: Print a Message with Separators**

**Part I [ \* ]**

Prompt the user for a message. Display the message character by character but **separate every two consecutive characters by a space**.

Your code should produce the following output:

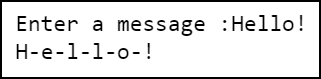
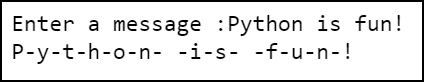
**Hint:**

* Use a for-loop to go through all the characters in the message one by one.
* Use print(s, end='') to print out s and keep the cursor in the same row.

**Part II [ \*\* ]**

Now write another piece of code that displays the message from the user character by character but separate every two consecutive characters **by a hyphen**.

Your code should produce the following output:

**Hint:** The last character should be handled differently from the other characters.

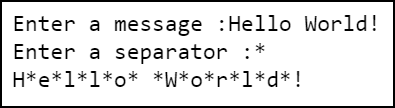
**Part III [ \*\* ]**

Now let us make the code above re-usable and more general.

Define a function called print\_message\_with\_separators that does the following:

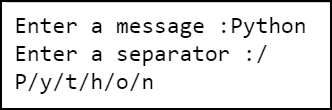
* The function takes in two parameters:
  + msg (of type str): a message string (e.g., 'Hello!')
  + sep (of type str): a separator string (e.g., '-')
* The function **prints out** msg character by character, but it inserts sep between every two consecutive characters of msg.
* The function **does not return any value**.

For example, when print\_message\_with\_separators("Hello World!", '\*') is called, we should see the following output:



We can see that the separator '\*' is inserted everywhere between two adjacent characters in the message.

When print\_message\_with\_separators("Python", '/') is called, we should see the following output:



**Q2: De Morgan’s Law [ \*\* ]**

Simplify the following expression.

not ( not ( a and not b ) or ( b or not a ) )

Simplified expression: .

Now check your answer by plugging in all combinations of possible values of a and b and evaluating the expression above as well as your simplified expression. Fill in your evaluations in the table below.

|  |  |  |  |
| --- | --- | --- | --- |
| **a** | **b** | **not (not (a and not b) or (b or not a))** | **Your simplified expression** |
| True | True |  |  |
| True | False |  |  |
| False | True |  |  |
| False | False |  |  |

**Q3: Display Numbers [ \*\* ]**

Define a function called display\_numbers. The function takes in two parameters: **m** and **n**. It is assumed that both **m** and **n** are integers, and **m** is less than or equal to **n**. The function displays all the integers between **m** and **n** (both inclusive) one by one, separated by spaces. However, the function handles the following numbers in a special way:

* If the number is a multiple of 3 but not a multiple of 5, the function displays a hyphen instead of the number.
* If the number is a multiple of 5 but not a multiple of 3, the function displays an asterisk instead of the number.
* If the number is a multiple of both 3 and 5, the function displays a # instead of the number.

For example, calling display\_numbers(4, 16) gives the following output to the screen:



**The following questions are to be done with Notepad++ and Anaconda Prompt.**

(Note: These questions can also be done in Jupyter Notebook. However, to ensure that you are also comfortable with running .py files in Anaconda Prompt, please save your solutions as .py files and test them in Anaconda Prompt.)

**Q4: Compute Sum [ \*\* ]**

Inside a file called sum\_utility.py, write a function called compute\_sum that takes in two parameters, **m** and **n**. The function returns the sum of all the integers between **m** and **n** (both inclusive).

You’re provided with a file sum\_test.py. Run this test file to test the function you have implemented in sum\_utility.py.

We should see the following output when sum\_test.py is executed:

|  |
| --- |
| The sum of integers between 4 and 10 is 49  The sum of integers between 50 and 100 is 3825 |

**Q5: Buying Books [ \*\* ]**

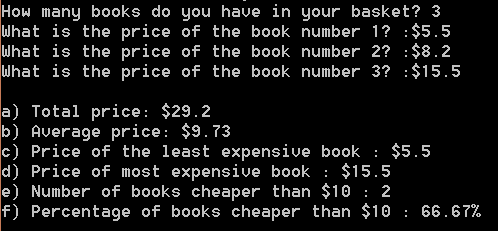
Write a simple program called books.py that helps a bookstore’s customer to gather some statistics of the books she has picked.

The program first asks the customer how many books she has in her basket. The program then prompts the customer for the price of each book. After that, the program does the following:

1. The program displays the total price of all the books.
2. The program displays the average price of all the books.
3. The program displays the price of the least expensive book
4. The program displays the price of the most expensive book.
5. The program displays how many books are cheaper than $10.00.
6. The program displays the percentage of books cheaper than $10.00.

**Note:** You can use the round() function to limit the number of digits after the decimal point to 2. For example, round(3.575, 2) returns 3.58 (as a float).

Your code should produce the following output:



**Q6: Palindrome [ \*\*\* ]**

A word is called a palindrome if it reads the same backwards as forwards. For example, “**madam**” and “**rotator**” are palindromes.

Write a program that prompts the user for a word and **use a for-loop** to check whether the word is a palindrome. (Hint: Compare the first character of the word with the last character of the word, the second character of the word with the last but one character of the word, and so on until you reach the middle of the word. See the picture below.)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| m | a | d | a | m |

Your code should produce the following output:

Graphical user interface, text, application, email

Description automatically generated  Graphical user interface, text, application, email

Description automatically generated

