Assignment II Specification
Print a string in a n x 7 grid

Goals and Topics

The assignment problem is straightforward. All necessary details have been supplied. The solution to the problem will use the programming concepts and strategies covered in Workshops 1-7. The subgoals are:

- Obtaining an advanced understanding of values and variables.
- Understanding program input and output, strings, functions, and expressions.
- Understanding simple strategies like iterations, conditionals, and validation plans.
- Translating simple design into Pseudocode and then Python code.
- The mechanics of editing, interpreting, building, running, and testing a program.
- Commenting source code.
- Becoming confident and comfortable with programming small problems.

Your Task

In this assignment, you will design and implement a string printing program for the client that allows users to input a valid string, then the program will print the string in a n x 7 grid.

Functional Requirements

The program should be implemented in *Python*. The client has specified the following requirements for the functionality of the program:

1. The program should be running without errors throughout two Phases: *Information Gathering* and *Information Presenting*.
2. *Information Gathering* is to gather the information from users such as a valid character (valid inputs: any single string only including characters “C”, “S”, “1”, “4” and “0”, one character can appear more than once in the string, case insensitive) and a break-word to break the iteration (valid inputs: “BREAK”, case sensitive).
3. If the user enters nothing or invalid input (e.g., “12”, “5.8” or “abc”), the program should alert an error message by a print function and then ask the user to re-enter. The process should iterate until a valid input is entered.

0	1	*	*	*	5
1	*				*
2	*				
3	*				
4	*				
5	*				*
6		*	*	*	

0	1	2	*	4	5
1		*	*		
2	*		*		
3			*		
4			*		
5			*		
6	*	*	*	*	*

0	1	*	*	*	5
1	*				*
2	*				
3	*	*	*	*	*
4					*
5	*				*
6		*	*	*	

0	1	2	3	*	5
1			*	*	
2		*		*	
3	*			*	
4	*	*	*	*	*
5				*	
6				*	

0	1	2	*	4	5
1		*		*	
2	*				*
3	*				*
4	*				*
5		*		*	
6			*		

0	1	*	*	*	5	6	7	*	*	*	11	12	13	*	*	*	17
1	*				*		*				*		*				*
2	*						*						*				
3	*						*	*	*	*	*		*				
4	*										*		*				
5	*				*		*				*		*				*
6		*	*	*				*	*	*				*	*	*	

0	1	2	*	4	5	6	7	8	9	*	11	12	13	14	*	16	17	18	19	20	*	22	23
1		*	*						*	*				*		*				*	*		
2	*		*					*		*			*				*		*		*		
3			*				*			*			*				*				*		
4			*				*	*	*	*	*		*				*				*		
5			*							*				*		*					*		
6	*	*	*	*	*					*					*				*	*	*	*	*

Figure 1: Part of valid string in n x 7 grid

4. *Information Presenting* is to present the outcome. Once receiving a new valid string from users, the program should print the string in a n x 7 grid (one string each time, no extra space between different characters. For example, each character is presented in a 6 x 7 grid. A string with two characters will be in a 12 x 7 grid as Figure 1).
5. The program will continue to ask the user to enter a string until the break-word "BREAK" is inputted and the program stops.
6. The string presented in the n x 7 grid should be the same as the character/string in Figure 1 (pay attention to the index value for each "*"). Do not show the table and index number. Just show the string using "*".
7. The code of printing the string should be organized efficiently with iterations and conditionals as the sample solution of Quiz 3 question b2. Just printing the string one line by one line using print(" *** ") is not acceptable.

Program Integration Test

You need to test the program for all functionality thoroughly before delivering the program to the client. After testing, please remove the testing code in the final version. The program should be running appropriately without any syntax or logic errors.

Non-Functional Requirements

- All code should appear in the Python code files.
- In the code document arrange your code in the following order:
  (a) Five functions for the 5 characters (the order must be "C", "S", "1", "4" and "0")
  (b) One or more functions for printing string with multiple characters ("CSC", "1401", "1404"(no need for a real string), and so on) (Challenging task).
  *Some students may be able to design a function including both (a) and (b). For assessment purposes, please separate them as per the requirement.
  (c) Main code
- Variable and function identifiers should follow appropriate conventions.
- Code is indented appropriately and grouped in blocks according to the common tasks they share.
- Appropriate comments should be added to all blocks of code. Do not simply translate the syntax into English for comments, instead, describe the purpose of the code. Docstring comments are required for each def function code.

Submission

What You Need to Submit – Three Files
For a complete submission, you need to submit three files as specified below. The assignment submission system will accept only the files with extensions specified in this section.

1. ***Pseudocode*** in a file saved in *.doc, .docx* or *.odt* format:

-The pseudocode should be completed before the Python code.
-The pseudocode should be succinct and accurate. Refer to workshop 7-course materials.
-The pseudocode should be presented logically and clearly.
Ideally, good Python pseudo-code enables different experienced Python programmers to complete the same program/solution independently without knowing the requirement of customers. The length of the pseudocode is better than 50% of that of real code for assignment purposes.

2. ***The program*** in a file saved with an. ipynb extension contains the source code implemented following the functional and non-functional requirements.
   The marker will only test your code in the Jupiter notebook. Please make sure your submitted Jupiter notebook file is **readable** by Anaconda installed as the "Setting up Python on your computer" section. Students cannot use any libraries or built-in functions which require an extra python package installed.
3. ***The program*** in a file saved with a *.doc/.docx/.odt* extension contains the exact same source code in the. ipynb file. You can just download the. ipynb as the *.py* file, then copy the code in the *.py* file and paste it to *the .doc/.docx/.odt* file. If you don't submit your code in both *.doc/.docx/.odt* and. Ipynb files, or the code in *.doc/.docx/.odt* and. *ipynb* files are different. You will get a penalty (up to 10 marks).

Marking Criteria

The assignment will be marked out of 20. Table 1 presents the marking criteria. If all criteria are satisfied, you will receive 20 marks. If not, all criteria are met, and part marks may be given. Check your own submission against these criteria before you submit it.

Table 1: Marking Criteria

ID	REQUIREMENTS You should submit **three files**. If you don't submit your code in both *.doc/.docx/.odt* and *.ipynb* files, or the code in *.doc/.docx/.odt* and *.ipynb* files are different. You will get a penalty (up to 10 marks).	Mark
Pseudocode (4 marks) * You may lose part of the Pseudocode marks corresponding to marks deducted in the Functional Requirements part.		
1	Pseudocode is presented in an acceptable format, and it is succinct and readable, without redundancy. The pseudocode for variables is well organized.	1
2	Pseudocode for functions and algorithms is presented logically and clearly.	1
3	Pseudocode for conditionals is presented logically and clearly.	1
4	Pseudocode for iterations is presented logically and clearly.	1
Functional Requirements (14 marks)		

5	User input for a single string is obtained appropriately. The validation plan validates a single string as required.	1
6	Input is terminated using the break-word "BREAK". It should be case-sensitive.	1
7	Character "C" is printed correctly as per the requirement. Docstring is presented appropriately. The code for printing the character "C" is organized efficiently with iterations and conditionals.	1
8	Character "S" is printed correctly as per the requirement. Docstring is presented appropriately. The code for printing the character "S" is organized efficiently with iterations and conditionals.	1
9	Character "1" is printed correctly as per the requirement. Docstring is presented appropriately. The code for printing the character "1" is organized efficiently with iterations and conditionals.	1
10	Character "4" is printed correctly as per the requirement. Docstring is presented appropriately. The code for printing the character "4" is organized efficiently with iterations and conditionals.	1
11	Character "0" is printed correctly as the requirement. Docstring is presented appropriately. The code for printing the character "0" is organized efficiently with iterations and conditionals.	1
12	The string "CSC" is printed correctly as per the requirement. Docstring is presented appropriately. The code for printing the string "CSC" is organized efficiently with iterations and conditionals.	1
13	String "1401" is printed correctly as per the requirement. Docstring is presented appropriately. The code for printing the string "1401" is organized efficiently with iterations and conditionals.	1
14	Valid inputs are case-insensitive. No extra space between the characters in the string. Docstring is presented appropriately for a multiple characters function to print string.	1
15	The code of printing any valid single string is organized efficiently with iterations, conditionals, and fruitful functions.	1
16	Any valid single string is printed correctly as the requirement. Hints: ID14-15are challenging tasks; you can complete ID 7-13 first if you don't know how to complete ID14-15. If you complete ID14-15 correctly, it means you meet the requirements of ID 7-13.	1
17	The program is implemented in Python. No syntax errors.	1
18	All the functions (except ID14 and 15) meet the requirements.	1
Non-functional Requirements (2 marks)		
19	The Identifiers of variables and functions are following professional conventions. Variables are used in calculation and expression instead of explicit values. At least five-line comments are added to describe the purpose of blocks of code. All the testing codes are removed from the submitted version.	2
	Total	20

Suggested Strategy

Plan to complete the assignment on time. Do not write all of the code in one sitting and expect that everything will be working smoothly like magic.

First step Read the assignment specification carefully; clarify anything unclear by putting a post on the assignment forum; think about how to do it, how to test it, and devise high-level algorithms for each independent part of the assignment. Begin to write the program (with comments), in incremental stages. Seek help on the assignment forum if needed.

Second step Re-read the specification carefully. Complete the "Pseudocode". Try to implement the function in one task a time, test it and make sure it works as expected before move to next task. Integrate all functions and finish initial coding.

Third step Fully test the program; have another review on the source code; re-read the specification (especially marking criteria) to make sure you have done exactly what is required.