Password Validator

During this semester you learned the Python topics of variables, conversions, arithmetic statements, logical constructs, looping constructs, functions, lists and string methods. This final project is the culmination of all those topics for you to demonstrate all that you have learned during the past 15 weeks.

Note: You can **NOT USE** Python's Regular Expression Operations known better as regex. You can only use what you learned in this class:
-variables
-conversions
-arithmetic statements
-logical constructs
-looping constructs
-functions
-lists
-string methods

Use Python String functions to build a program to do the following to perform Password Validation. Make sure you use the examples Professor Candido has provided and use the **Python String Functions** document found in Week 15. These are the required 12 coding steps:

1. Prompt the user for their First and Last name and store the input into a variable called **sName**.
2. Code a loop that will keep asking for a password **UNTIL** a valid password in entered.
3. Prompt the user for their desired password and store their response in a variable called **sPassword**.
4. Extract out the first initial from the first name and the first initial of the last name from the variable **sName** and put in a new variable called **sInitials**. For example if *Brian Candido* is in the **sName** variable then the **sInitials** should contain *BC*. You can assume the user will enter 1 first name and 1 last name.
5. Check to make sure the **sPassword** length is between 8 and 12 characters. If the password is not within the required length write out the message: *Password must be between 8 and 12 characters*.
6. Check to make sure the **sPassword** starts does not start with *Pass* or *pass*. If not write out the *Password can't start with Pass.*
7. Check to make sure the **sPassword** contains at least 1 uppercase letter A through Z. If not write out the message: *Password must contain at least 1 uppercase letter.*
8. Check to make sure the **sPassword** contains at least 1 lowercase letter a through z. If not write out the message: *Password must contain at least 1 lowercase letter.*
9. Check to make sure the **sPassword** contains at least 1 number between 0 and 9. If not write out the message: *Password must contain at least 1 number.*
10. Check to make sure the **sPassword** contains at least 1 of these special characters: ! @ # $ % ^. If not write out the message: *Password must contain at least 1 of these special characters: ! @ # $ % ^*
11. Check to make sure the **sPassword** does not contain the value of **sInitials** within the string. For example sPassword cannot contain BC or bc or any variation of BC so you will need to convert to either lowercase or uppercase than compare using Python string functions. If not write out the message: *Password must not contain user initials.*
12. No character can be present more than once. Write code to process each character in the password and keep track of how many occurrences are present. If any character (either uppercase or lowercase versions) is in the password more than 1 output the character and how many occurrences and print out the message: *These characters appear more than once:*
13. If **sPassword** passes all the above checks output this message: *Password is valid and OK to use.* And exit the loop.
14. Make sure your code has a **main()** function. It is your optional choice if you want additional functions.

Sample Output

Password is not valid:

```
Enter full name such as John Smith: Brian Candido
Enter new password: BC12345
Password must be between 8 and 12 characters
Password must contain at least 1 lowercase letter
Password must contain at least 1 of these special characters: ! @ # $ % ^
Password must not contain user initials.

Enter new password:
```

Password is not valid:

```
Enter new password: abc!123
Password must be between 8 and 12 characters
Password must contain at least 1 uppercase letter
Password must not contain user initials.

Enter new password:
```

Password is not valid:

```
Enter new password: PasSW0rd
Password can't start with Pass.
Password must contain at least 1 of these special characters: ! @ # $ % ^
These characters appear more than once:
s: 2 times

Enter new password:
```

Password is not valid:

```
Enter new password: Mkc!C06182006
Password must be between 8 and 12 characters
These characters appear more than once:
c: 2 times
0: 3 times
6: 2 times

Enter new password:
```

Password is valid:

```
Enter new password: Mkc!0618
Password is valid and OK to use.
```

Grading Rubric

Criteria	Meets (100%)	Somewhat (50%)	Not Present (0%)
Password validations steps 4 through 10 are	All String validation requirements stated in the instructions	Some of String validation requirements stated in the instructions Steps 3	String validation requirements stated in the instructions Steps 3

coded using Python String functions. 70 points	Steps 3 through 9 are present, functioning and coded efficiently.	through 9 are attempted and mostly functioning.	through 9 are not implemented.
Code to make sure the Password does not contain the user's initials (step 11). 10 points	The code to detect the user's initials is present and coded correctly.	The code to detect the user's initials is attempted and partially works.	The code to detect the user's initials is not present.
Code to check for repeating characters (step 12). 15 points	The code to detect repeating characters is present and coded correctly.	The code to detect repeating characters is attempted and partially works.	The code to detect repeating characters is not present.
Comments, Formatting and Variable Prefixing and main() function is coded and Password Input loop is coded. 5 points	Comments present and variable prefixing. Output was correctly formatted per the sample. main() is present. Input Loop is present.	Either Comments present or variable prefixing. Output was attempted but did not match the sample. main() is attempted. Input Loop is present attempted.	No Comments present and No variable prefixing. No formatting was attempted. main() is not present. No Input Loop is present.

Note: You can **NOT USE** Python's Regular Expression Operations known better as regex. You can only use what you learned in this class:
-variables
-conversions
-arithmetic statements
-logical constructs
-looping constructs
-functions
-lists
-string methods

Any code used that was not covered in this class will earn you a 0 for this Final Project.