Question 1

Online security is one of the most crucial parts of our lives. The importance of security is increasing day by day as most things are going online. Passwords are a common feature of online security but many people use passwords that are easy to guess or passwords that are very common. Eight of the most common passwords used in 2020 are shown below in Fig 1. For this reason it is recommended that a secure password includes a combination of uppercase and lowercase letters as well as numbers and special characters.

Fig 1

(a) Open the program called **Question_1.** The source code is shown below. This program is designed to create and display a secure password.

The variable *password_length* is used to determine the length of the created password. This variable is initially set to **5.**

Three lists have been created that store the characters needed for generating a secure password.

Alphabets : Contains all lowercase and uppercase letters e.g a-z and A-Z
Digits: Contains numbers 0-9
Special characters: Contains the following special characters !@#$%^&*()

The *random.choice( )* method is used to randomly choose an element from the list '**alphabets'** which is then added to the password list for use in the final password.

```
# Name

import string
import random

## characters to generate password from
alphabets = list(string.ascii_letters)  # A list of all lower and upper case letters
digits = list(string.digits)            # A list of numbers 0 - 9
special_characters = list("!@#$%^&*()") # A list of all the pecial characters shown

password_length=5       # Hardcoded password length value
password=[]

for i in range (password_length):
    password.append(random.choice(alphabets))

print("".join(password)) # Joining the elements of the list password to create the password.
```

Source Code

(i) Insert a comment to say '*Initialising an empty list*' in an appropriate place in the program.

(ii) Modify the program so it uses a user inputed value for *password_length* instead of the existing hard coded value of 5. Your output will look similar to Fig 2.

```
How many characters will be in your password: 8
jamInrZY
```

Fig 2

(iii) Currently the program only gives the user a password made up of alphabetic characters as is shown in Fig 2.

Modify the program to ask the user to enter how many alphabetic characters, numeric characters and special characters they want to use in their password and append the characters to the existing password list. Your output will look similar to Fig 3.

```
How many characters will be in your password: 8

How many alphabetic characters will be in your password: 4
How many numeric characters will be in your password: 3
How many special characters will be in your password: 1

RNoH256(
```

Fig 3

(iv) Validate that the number of characters entered for the password in part (iii) matches the length of the password set by the user in part (ii). If the number of characters entered doesn't match the password length, the program should warn the user with an appropriate message and terminate the program. Your output will look similar to Fig 4

```
How many characters will be in your password: 8

How many alphabetic characters will be in your password: 3
How many numeric characters will be in your password: 2
How many special characters will be in your password: 2

You have not entered the correct number of characters. Please start again.
```

Fig 4

(v) The National Institute of Standards and Technology and many firms such as Microsoft and Google recommend that the minimum password length is 8 characters and a very strong password should be up to 20 characters. Modify the program so that a user must enter a password length of between 8 and 20 characters. The program should keep asking the user to enter a value for password_length until a valid number is given. Your output will look similar to Fig 5

```
How many characters will be in your password: 5

A valid password length is between 8 and 20. Please try again.
How many characters will be in your password: 24

A valid password length is between 8 and 20. Please try again.
How many characters will be in your password: 10

How many alphabetic characters will be in your password:
```

Fig 5

Question 2

A factor is any integer which divides exactly into anther integer so there is no remainder value. For example 3 is a factor of 12 because it divides exactly into 12. E.g 12/3 = 4 no remainder, whereas 5 is not a factor of 12 as there is a remainder of 2. You can use the modulus operator (%) in Python to check if there is any remainder value from a division calculation. This can be seen in Fig 6 and Output of Fig 6

```
print(6 % 3) # This divides 6 by 3, and returns a value of 0
             # as there is no remainder value in this calculation

print(6 % 5) # This divides 6 by 5, and returns a value of 1
             # as the remainder from this calculation is 1

print(8 % 5) # This divides 8 by 5, and returns a value of 3
             # as the remainder from this calculation is 3
```

Fig 6

```
6%3 gives a remainder of 0

6%5 gives a remainder of 1

8%5 gives a remainder of 3
```

Output of Fig 6

(a) Open the program called **Question_2.** The source code is shown below. This program tells the user if a value, *value_2,* is a factor or *value_1.*

```
value_1 = 18
value_2 = 3

if value_1 % value_2 == 0:
    print(f'{value_2} is a factor of {value_1}')
else:
    print(f'{value_2} is not a factor of {value_1}')
```

This program has two variables *value_1* and *value_2* and assigns them the values 18 and 3 respectively.

(i) Modify the program so the user can input values for *value_1* and *value_2.* Your output will look similar to Fig 7

```
Enter value 1: 25
Enter value 2: 5
5 is a factor of 25
```

Fig 7

(ii) Create a function called ***isfactor,*** that accepts *value_1* and *value_2* as input arguments and outputs to the user if *value_2* is a factor of *value_1. Your output will look similar to Fig 7.*

(iii) Create a second function called ***isfactor2*** in the same program that takes *value_1* and *value_2* as input arguments and outputs to the user all the factors of those two variables. Your output will look similar to Fig 8.

```
Enter value 1: 120
Enter value 2: 20

The factors of 120 are [1, 2, 3, 4, 5, 6, 8, 10, 12, 15, 20, 24, 30, 40, 60]

The factors of 20 are [1, 2, 4, 5, 10]
```

Fig 8

(iv) Create a third function called ***isfactor3*** that outputs to the user the highest common factor common to both *value_1* and *value_2.* Your output will look similar to Fig 9.

```
Enter value 1: 120
Enter value 2: 20

The highest common factor is 10
```

Fig 9

(v) Using conditional statements, modify the program so the user is given a choice of which function they want to use. Your output will look similar to Fig 10.

```
Enter value 1: 120
Enter value 2: 20

What operation would you like to carry out on your inputs:

        1.Is 20 a factor of 120
        2.Find out the factors of 120 and 20
        3.Find the highest common factor of 120 and 20

        Choice:2

The factors of 120 are [1, 2, 3, 4, 5, 6, 8, 10, 12, 15, 20, 24, 30, 40, 60]

The factors of 20 are [1, 2, 4, 5, 10]
```

Fig 10