Mini Project: Random Password Generator

Security on the Internet is very important. One of the things that you as an Internet user can do is use strong passwords. Strongs passwords are typically long sequences of random characters, and you should have a different password for each account on the Web.

Create a console Python program that will help generate random passwords based on some simple settings.

1. Main menu

The main menu should show the name of the program and the options. It should then accept input from the user.

```
-- Password generator --
Choose option:
1: generate password
2: exit the program
Your choice:
```

1. If the user provides (1), begin the password generation procedure (described below). After the procedure has been completed, show the main menu again.

2. If the user provides (2), show a farewell message and finish the program:

```
Bye!
```

3. If the user provides anything else, show them a warning and show the whole main menu again.

```
Please enter a correct value
```

2. Password generation

There are three groups of characters that can be used in the passwords:

letters: a-z
digits: 0-9
special characters: !@#$%&*^|()_+

The most basic password type contains lowercase letters only. However, the user should also be able to include uppercase letters, digits and/or special characters. For instance, if the user decides to include uppercase letters and special characters (without digits), each character in the generated password should be randomly chosen as a (a) lowercase character, (b) uppercase character, or (c) special character. The user should also be able to specify the length of the password.

Using the prompts shown below, ask the user for the length of password, whether uppercase letters should also be considered, whether digits should be considered, and whether special characters should be considered. Different combinations are possible: for example, the user may ask for a password that contains lower and uppercase letters as well as special characters, but contains no digits.

```
Provide password length:
Use uppercase letters? (y/n):
Use digits? (y/n):
Use special characters? (y/n):

Generated password:
```

3. Other details

If there is anything else about the program not specified in this instruction, you are free to solve it as you considered appropriate. Remember that in programming there are always different ways you can solve the same problem. :)

4. Sample program output

Here is an example of complete program output (alongside user input) from start to end

```
-- Password generator --
Choose option:
1: generate password
2: exit the program
Your choice: 4
Please enter a correct value

-- Password generator --
Choose option:
1: generate password
```

```
2: exit the program
Your choice: 1
Provide password length: 10
Use uppercase letters? (y/n): y
Use digits? (y/n): n
Use special characters? (y/n): n

Generated password: hTekKDjVmK

-- Password generator --
Choose option:
1: generate password
2: exit the program
Your choice: 1
Provide password length: 15
Use uppercase letters? (y/n): n
Use digits? (y/n): n
Use special characters? (y/n): y

Generated password: j)nddk*ip&*trn!

-- Password generator --
Choose option:
1: generate password
2: exit the program
Your choice: 2
Bye!
```

5. Hints & tips

Below you will find some hints from me. Remember they are only suggestions: you can use a different approach and still write a correct program.

1. Create three strings (or three lists) with the characters from each group that may appear in the password. The first string (or list) should contain all letters from a to z, the second string (list) should contain all digits 0-9, and the third one – all the special characters mentioned in the instruction.

2. Use the **random** module that you already know. You can use the **choice()** function to pick a random character from a string:

```
alpha = "abcdefghijklmnopqrstuvwxyz"
letter = choice(alpha)
```

Think about how you can add digits and special characters to the solution.

3. To include uppercase letters in the password generation process, you can use a function named **randint** from the **random** module. You will easily find its documentation online. Instruct **randint** to randomly return 0 or 1. If you get 0, use a lowercase letter as-is. If you get 1, use the **upper()** function on the lowercase letter to make it uppercase.

4. Use the technique described in points 1-3 to generate a password character by character until you get the desired length.

If you are still not sure how to write the program, remember that you can always take a look at the sample solution that I've prepared for you. It is available as a downloadable resource in the same lecture.