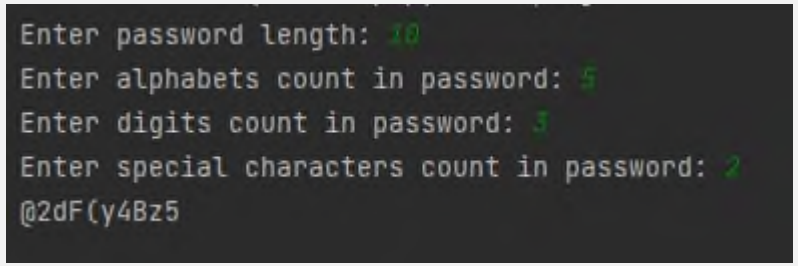
Python Programming
Mini Project
Password Generator.

Presented by Advaspire Team

What we are making

In this lesson, we are going to make a password generator. This generator is a bit different than the ones offered by Google or others as with this one, you can customized it in a way.



Setting Up

You can do this in one of the project you are doing as it don't need a new function to be defined. Just create a new file and name it PWGen.

Setting Up

At the top, lets start with importing random and string. The string here is a compilation of letters and digits that will be used for the password so make sure to import it.

```
import string
import random

## characters to generate password from
alphabets = list(string.ascii_letters)
digits = list(string.digits)
special_characters = list("!@#$%^&*()")
characters = list(string.ascii_letters + string.digits + "!@#$%^&*()")
```

Then, we need to set up the variables. All of the variables will be in a list. By putting list function at the start of the bracket, it will convert the data into a list so we don't need to manually insert it. The variables here will store all of the data from the string pip.

Define a PWGen

```
def generate_random_password():
    ## length of password from the user
    length = int(input("Enter password length: "))

    ## number of character types
    alphabets_count = int(input("Enter alphabets count in password: "))
    digits_count = int(input("Enter digits count in password: "))
    special_characters_count = int(input("Enter special characters count in password: "))

    characters_count = alphabets_count + digits_count + special_characters_count
```

To start, we are going to define a function for the generator so we can just call it whenever we need to use it. We will call it as generate_random_password. You can name it other names that you see fit.

Then, we need to set up some variable to get input from the users. Start with the length, as this will let you set the length of the password you want. The rest of the code here works the same way as its name intended to.

The character count will be used later on so it will be explained later.

Conditions

For this part, we are going to set a condition so that the character count is not more than the set length at the start of the code.

```
if characters_count > length:
    print("Characters total count is greater than the password length")
    return
```

If the character count is more than the length, it will print out to tell you. So there are no inconsistency with the password.

Randomizing the Passwords.

For the next 4 loop, it will do the randomizing for us. It will use the for loop and the random function.

Start with the password variable. It will be used to store all the randomized character from the loops after this.

```
## initializing the password
password = []

## picking random alphabets
for i in range(alphabets_count):
    password.append(random.choice(alphabets))

## picking random digits
for i in range(digits_count):
    password.append(random.choice(digits))

## picking random alphabets
for i in range(special_characters_count):
    password.append(random.choice(special_characters))
```

Random Cont.

```
## initializing the password
password = []

## picking random alphabets
for i in range(alphabets_count):
    password.append(random.choice(alphabets))

## picking random digits
for i in range(digits_count):
    password.append(random.choice(digits))

## picking random alphabets
for i in range(special_characters_count):
    password.append(random.choice(special_characters))
```

Lets start with the alphabets first. Copy the code here. The for loop will use the range function so the generated character wont be more than the set alphabet count. That is why the alphabets_count is in the bracket, for it to match the data.

Then inside the loop, we need to use the password variable at the start and then the append function. Because the variable is empty, we use the append function to add the chara generated by the loops to the password variable. The rest of the code has the same concept.

Basically, the password variable will store each chara that is chosen in the loop and combined together to form the random password.

Failsafe Mechanic

```
Enter password length: 10
Enter alphabets count in password: 5
Enter digits count in password: 3
Enter special characters count in password: 2
@2dF(y4Bz5
```

Now we need a failsafe mechanic in case the total character count is less than the password length. So to start, set the condition for character counts< length. Inside, type the random.shuffle. So this part will do like shuffling the generated password again which will then used in the next loop.

```
if characters_count < length:
    random.shuffle(characters)
    for i in range(length - characters_count):
        password.append(random.choice(characters))
```

In the for loop, we need to add the minus inside the bracket so they know how many chara needs to appended inside the generated password.

Finish

Once the chara generation is finished, add another line of code for random.shuffle(password). So it will shuffle all the letters and numbers generated previously to create a randomized password.

Then we need to code to convert the list to string and printing it out.
The method we use here will print the password as a string. If we were to use the str method, it will printout everything like inside a list.

```
    ## shuffling the resultant password
    random.shuffle(password)

    ## converting the list to string
    ## printing the list
    print("".join(password))

    ## invoking the function


generate_random_password()
```

Complete Game Code

```python
import string
import random

## characters to generate password from
alphabets = list(string.ascii_letters)
digits = list(string.digits)
special_characters = list("!@#$%^&*()")
characters = list(string.ascii_letters + string.digits + "!@#$%^&*()")

def generate_random_password():
    ## length of password from the user
    length = int(input("Enter password length: "))

    ## number of character types
    alphabets_count = int(input("Enter alphabets count in password: "))
    digits_count = int(input("Enter digits count in password: "))
    special_characters_count = int(input("Enter special characters count in password: "))

    characters_count = alphabets_count + digits_count + special_characters_count

    ## check the total length with characters sum count
    ## print not valid if the sum is greater than length
    if characters_count > length:
        print("Characters total count is greater than the password length")
        return

    ## initializing the password
    password = []
```

```python
    ## picking random alphabets
    for i in range(alphabets_count):
        password.append(random.choice(alphabets))

    ## picking random digits
    for i in range(digits_count):
        password.append(random.choice(digits))

    ## picking random alphabets
    for i in range(special_characters_count):
        password.append(random.choice(special_characters))

    ## if the total characters count is less than the password length
    ## add random characters to make it equal to the length
    if characters_count < length:
        random.shuffle(characters)
        for i in range(length - characters_count):
            password.append(random.choice(characters))

    ## shuffling the resultant password
    random.shuffle(password)

    ## converting the list to string
    ## printing the list
    print("".join(password))

    ## invoking the function


generate_random_password()
```

You can direct message your teacher and ask your question through Slack Robotene Community or arrange a One-to-One Consultation with your teacher.

Any Questions?

Thank you :)