Password Generator

This is the solution to the password generator project.

Instructions

Inside the file, you'll use three lists already saved to a corresponding variable: `letters`, `numbers`, and `symbols`.

```
letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l',
'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', 'A',
'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P',
'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']
numbers = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
symbols = ['!', '#', '$', '%', '&', '(', ')', '*', '+']
```

The program will ask:

```
How many letters would you like in your password?
```

```
How many symbols would you like?
```

```
How many numbers would you like?
```

The objective is to take the inputs from the user to these questions and then generate a random password.

Use your knowledge about Python lists and loops to complete the challenge.

Easy Version (Step 1)

Generate the password in sequence. If the user wants

- *4 letters*
- *2 symbols*
- *3 numbers*

then the password might look like this:

```
```

```
fgdx$*924
```

You can see that all the letters are together.

All the symbols are together and all the numbers follow each other as well.

Try to solve this problem first.

Hard Version (Step 2)

When you've completed the easy version, you're ready to tackle the hard version.

In the advanced version of this project the final password does not follow a pattern.

So the example above might look like this:

```
x$d24g*f9
```

And every time you generate a password, the positions of the symbols, numbers, and letters are different.