Day 8 Python Challenge

Today we have seen very varied but important things to write more efficient, dynamic and maintainable code. We learned how to install packages, how to divide our code into modules, how to handle possible errors, how to test our code, and we learn some more tools like decorators and generators.

Now it's time to put it all into action in a new program. Today's challenge is for you to create a software that works like a ticket vending machine in a drugstore. What's a ticket vending machine? It's a machine you can find at the entrance of many stores or banks. This machine asks you what procedure you've come to perform, and assigns you a turn number according to the area you want to go.

In our case, you are going to create it for a drugstore where there are three areas of attention: **perfumes**, **medicine** and **cosmetics**. Your program will ask the customer which of the areas they want to go, and it'll give them a shift number depending on which area they go to.

For example, if I choose cosmetics, it will give me the letter C (as in cosmetics), - (dash) 54. After this, it will ask us if we want to take another number to simulate that a new client is coming, and it will repeat the whole process.

Some things to keep in mind.

- Different customers will be taking different numbers for different areas (perfumes, medicine, cosmetics) in a different order, so the system must **keep track of how many numbers it has given for each of those areas** and produce the next number for each area as they ask for it. You probably already understand that we need to take advantage of generators and their efficiency to do this.
- The message where we tell the customer they're waiting number should have some additional text before and after the number. For example: "Your number is..." then the number itself with the letter at the beginning, and something like "Wait and someone will be with you shortly". In order for our code not to repeat itself, instead of putting this text in each of the functions that calculate the numbers, we can take advantage of the decorators flexibility to create that additional text only once and then wrap any of our functions with that unique text.
- Finally, you should take advantage of the fact that you now know how to split your program into different modules and separate the code into two parts: on the one hand, a module that can be called numbers.py, where you write all the generators and the

decorator, then a second module which we can call main.py, where you write the functions that manage the operation of the program (for example, instructions to choose an area and to decide if it will continue taking new clients or end the program).

Remember that you will need to import the content of numbers.py into main.py in order to have its functions available.

If you can manage to develop this code following these guidelines, you will have put into action everything you have learned today, and you will be reinforcing your knowledge in a great way.

It's time for loud music headphones, your favorite snack and a lot of concentration. It's time to program.