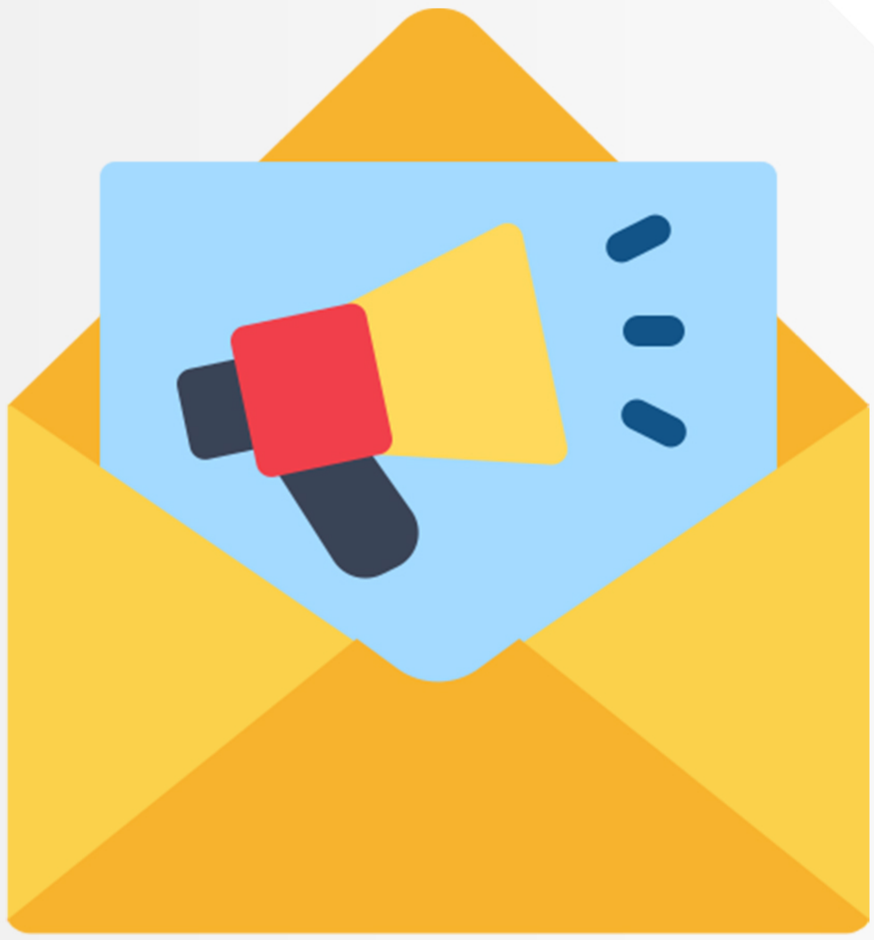
CODE CHALLENGE
CHAMPIONSHIP

Python

Junior

Handbook

Code Challenge Championship



Mission 1

Business Email Creator

Business Email Creator

Overview:

Creating emails for employees of a company could be an exhausting job if it's done manually. As a result, you will help a company generate email addresses for its employees. The goal is to collect employee information, including their IDs and names, and then generate email addresses for each employee using the provided domain name.

Mission:

You are tasked with developing a Python script that accomplishes the following:

1- Create an empty dictionary named "employees" to store employee information which will be their IDs as keys and their names as values.

2- Ask the user to enter the domain name of the company.

3- Continuously collect employee IDs and names until the user decides to enter "done" then add this information in the employees dictionary.

4- Create and print email addresses for each employee using the collected information and the domain name in that format:

"EmployeeNameEmployeeID@DomainName.com."

5- Finally, display the generated email addresses to the user.

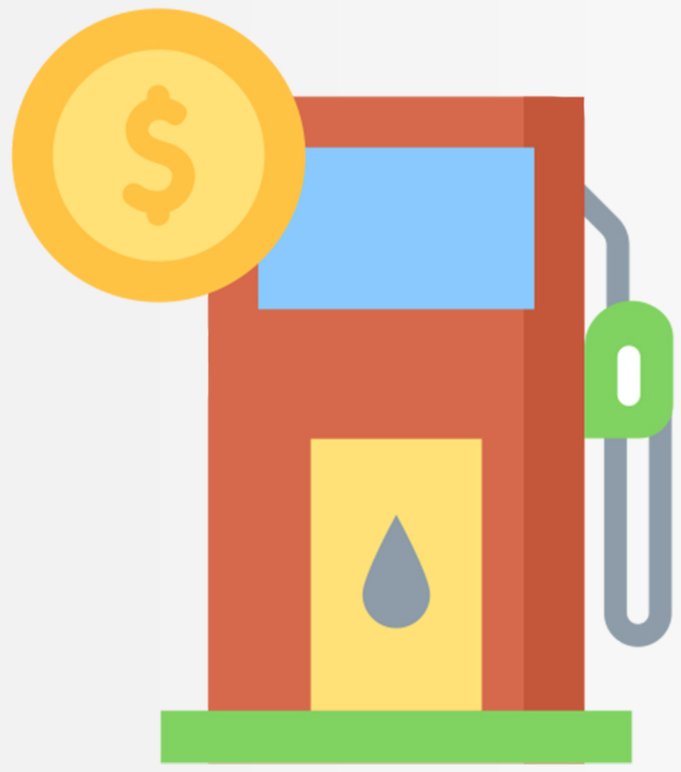
As a bonus part, handle cases where the user's input is missing as not entering an ID or a name and make a counter that will keep tracking the employees' number as shown in the expected output.

Useful Hints:

- ✓ Use the f-string to easily creating the business email for each employee using this format: f'{employee_name}{employee_id}@{domain_name}.com'
- ✓ Initialize the counter variable outside any while loop to avoid resetting its value after each iteration.
- ✓ Use the items() dictionary method to easily iterate over each key-value pair in the employees dictionary in order to create the employee's email.

Expected Output:

```
Hello, I will help you for making emails for your employees!
What is the domain name of the company? btech
Enter Employee 1 ID (or enter "done" to finish): 1041
Enter employee name: Tarek
Enter Employee 2 ID (or enter "done" to finish): 2903
Enter employee name: Ibrahim
Enter Employee 3 ID (or enter "done" to finish): 8002
Enter employee name: Adel
Enter Employee 4 ID (or enter "done" to finish): 3455
Enter employee name: Hadeer
Enter Employee 5 ID (or enter "done" to finish): 1298
Enter employee name: Alaa
Enter Employee 6 ID (or enter "done" to finish): done
Here are your employees emails!

Employee ID: 1041, Employee Name: Tarek, Email: Tarek1041@btech.com
Employee ID: 2903, Employee Name: Ibrahim, Email: Ibrahim2903@btech.com
Employee ID: 8002, Employee Name: Adel, Email: Adel8002@btech.com
Employee ID: 3455, Employee Name: Hadeer, Email: Hadeer3455@btech.com
Employee ID: 1298, Employee Name: Alaa, Email: Alaa1298@btech.com
```



Mission 2

Gas Station Prices

Gas Station Prices

Overview:

A gas station named "Sandmarton" decided to make a time-saving idea for clients who want to check how much will they pay before entering the gas station to enhance customers experience and avoid crowdedness. So, they want to display the fuel prices using a screen that helps customers calculate the cost of fuel based on their selection of fuel type and quantity.

Mission:

You will help them to create the script of that program's logic as the following:

1- Create a function named "fuel_calculator" that contains 2 parameters: the first is "fuel_type" of type string for take the fuel type as an input to that function and the second is "quantity" of type float for taking the fuel quantity in liters that the user needs.

2- The function will calculate the total price that the user should pay based on the following fuel types prices per 1 liter:

- **Fuel of benzene 80 → costs 8.25 EGP**
- **Fuel of benzene 92 → costs 9.75 EGP**
- **Fuel of benzene 95 → costs 10.50 EGP**
- **Fuel of diesel → costs 7.50 EGP**

3- After displaying the fuel options to choose from, you should continuously ask them to enter their choice then how many liters they need in order to calculate the total cost by the "fuel_calculator" function.

4- As long as the user doesn't enter "q" to quit the program, the program should be working and calculating the total price for each customer uses the program by entering the fuel type and the quantity in liters as mentioned earlier.

5- Finally, you should handle invalid cases entered by the user as entering an invalid fuel type or ignoring to enter an input or entering negative quantity for liters to the function as shown below in the expected output section.

As a bonus part, make a counter that counts the number of customers that will use the program until the program is exited by them and don't forget to display that number before the program ends.

Useful Hints:

- ✓ **You can create a dictionary that stores the fuel type as its keys and their prices as its values to easily access the fuel types' prices inside the function to calculate the total cost.**
- ✓ **To calculate the total cost, you should use the following formula: price of fuel type x number of liters.**
- ✓ **Initialize the counter variable outside the loop to avoid resetting its value after each iteration.**
- ✓ **Use the lower() or upper() string methods to handle any string input entered by the user.**

Expected Output:

An example of 2 users using the program to see the total cost they should pay if they benzene 80 for 35 liters and benzene 92 for 40 liters respectively.

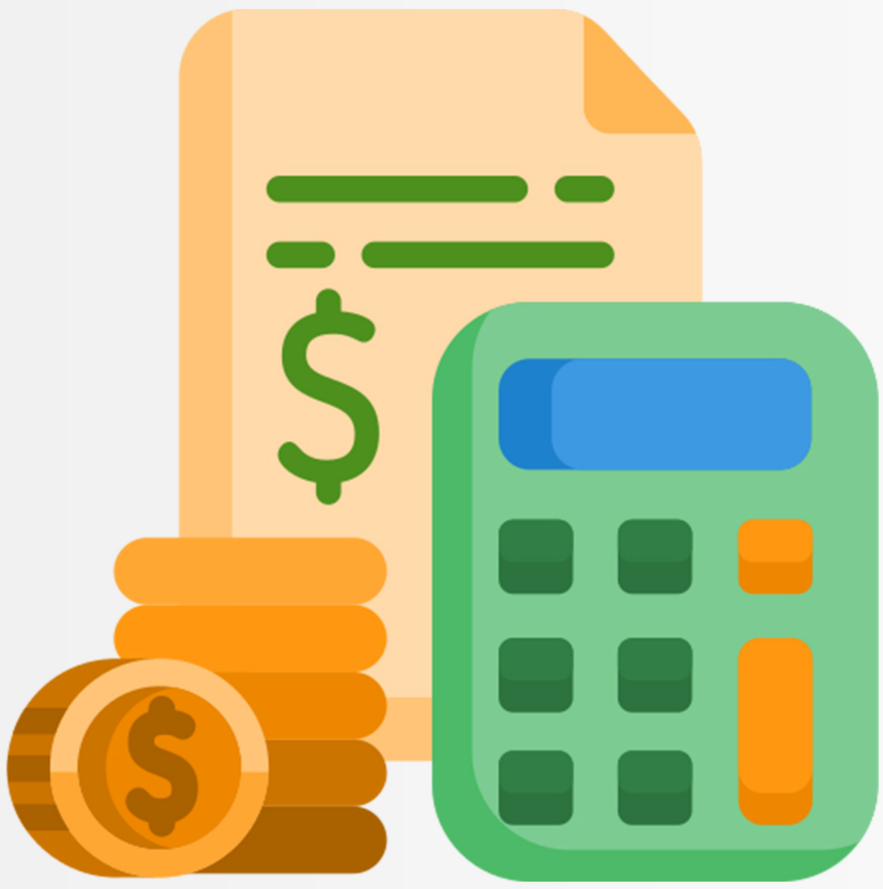
```
Welcome to Sandmarton Gas Station...
Check the prices before you pay!

Type 80 -> 8.25 EGP/liter
Type 92 -> 9.75 EGP/liter
Type 95 -> 10.5 EGP/liter
Type diesel -> 7.5 EGP/liter

Enter the type or q to quit: 80
Enter the number of liters: 35
For 35.0 liters of benzene 80 is 288.75 EGP

Enter the type or q to quit: 92
Enter the number of liters: 40
For 40.0 liters of benzene 92 is 390.0 EGP

Enter the type or q to quit: q
2 customers used the program before exiting..
Thanks for visiting us!
```

Handling uppercase inputs for a customer checked the total price of 84 liters of diesel which is 84 * 7.5 = 630 EGP

```
Welcome to Sandmarton Gas Station...
Check the prices before you pay!

Type 80 -> 8.25 EGP/liter
Type 92 -> 9.75 EGP/liter
Type 95 -> 10.5 EGP/liter
Type diesel -> 7.5 EGP/liter

Enter the type or q to quit: DIESEL
Enter the number of liters: 84
For 84.0 liters of diesel is 630.0 EGP

Enter the type or q to quit: Q
1 customers used the program before exiting..
Thanks for visiting us!
```

Handling invalid cases: leaving the input area empty, entering invalid fuel type, or negative quantity of liters.

```
Welcome to Sandmarton Gas Station...
Check the prices before you pay!

Type 80 -> 8.25 EGP/liter
Type 92 -> 9.75 EGP/liter
Type 95 -> 10.5 EGP/liter
Type diesel -> 7.5 EGP/liter

Enter the type or q to quit:
You should choose an option or q to quit!

Enter the type or q to quit: 90
Invalid fuel type, please try again!

Enter the type or q to quit: 95
Enter the number of liters: -50
Sorry, the number of liters should be at least 1 liter

Enter the type or q to quit: q
0 customers used the program before exiting..
Thanks for visiting us!
```



Mission 3

Library Fines Calculator

Library Fines Calculator

Overview:

A local library faces challenges in accurately calculating fines for overdue books as they use manual calculations, causing confusion and disputes among patrons. The Library Fine Calculator program simplifies the process, providing a user-friendly solution to swiftly calculate fines, easing administrative tasks and ensuring fair fines for patrons within seconds.

Mission:

You are tasked to develop a functional program that simplifies the fine calculation process, making it easier for library staff and patrons to determine accurate fines for overdue books as the following:

- Start with defining the fines. The library has different fine rates based on the type of book as the following: {'Regular': 20, 'Art': 10, 'History': 15, 'Scientific': 30, 'Children': 25}.
- Implement a function that named “calculate_fine” that will have 2 parameters: “book_type”, “days_overdue”. This function should calculate the fine rate by multiplying the number of days overdue by the fine rate of the book. In simpler terms, the fine is calculated by: fine rate of the book x days overdue
- After creating the function, you should show the types of books and their fines rates to the user then continuously asking them to enter the book type then the number of days overdue to pass them to the function you created that will calculate the total fine and show it for that user. The user should write ‘done’ to stop the program.

As a bonus part, handle uppercase and lowercase inputs and any invalid inputs like entering invalid book type or invalid days overdue as negative numbers or 0 days.

Useful Hints:

- ✓ Use a dictionary named fine_rates to store all book types as keys and their fine rates as values.
- ✓ Use the capitalize() string method to handle any uppercase or lowercase inputs entered by the user when they specify the book type.
- ✓ To format the output to two decimal places, you can use the f-string formatting, e.g., f'{total_fine:.2f}'

Expected Output:

Calculating a Children book Fine with 3 days Overdue

```
Welcome to the Library Fine Calculator. Here are the fines rates per day
The Regular book --> 20 EGP
The Art book --> 10 EGP
The History book --> 15 EGP
The Scientific book --> 30 EGP
The Children book --> 25 EGP

Enter the type of book or 'done' to exit: children
Enter the number of days overdue: 3
Fine for the Children book, 3 days overdue: 75.00 EGP

Enter the type of book or 'done' to exit: done
```

Calculating Art, and History books Fines with 5, and 9 days Overdue Respectively

```
Welcome to the Library Fine Calculator. Here are the fines rates per day
The Regular book --> 20 EGP
The Art book --> 10 EGP
The History book --> 15 EGP
The Scientific book --> 30 EGP
The Children book --> 25 EGP

Enter the type of book or 'done' to exit: ART
Enter the number of days overdue: 5
Fine for the Art book, 5 days overdue: 50.00 EGP

Enter the type of book or 'done' to exit: History
Enter the number of days overdue: 9
Fine for the History book, 9 days overdue: 135.00 EGP

Enter the type of book or 'done' to exit: DONE
```

Handling Invalid cases for bonus

```
Welcome to the Library Fine Calculator. Here are the fines rates per day
The Regular book --> 20 EGP
The Art book --> 10 EGP
The History book --> 15 EGP
The Scientific book --> 30 EGP
The Children book --> 25 EGP

Enter the type of book or 'done' to exit: Fiction
Invalid book type. Please enter a valid book type.

Enter the type of book or 'done' to exit: Scientific
Enter the number of days overdue: 0
Sorry, overdue days should be at least 1 day
Enter the number of days overdue: 4
Fine for the Scientific book, 4 days overdue: 120.00 EGP

Enter the type of book or 'done' to exit: done
```