|  |  |
| --- | --- |
| **Student** | **Juliana Salvadori** |
| **Student ID** | **10521647** |
| **Github** | <https://github.com/jusalvadori/B9DA100-JS/tree/master/billManagement> |
| **Files required** | bill\_management.py  bill\_management\_test.py  bill\_management\_visualization.py  bills.csv  bills-Copy.csv |
| **Description** | Go to [Description](#_Description) |

**Assessment Brief**

|  |  |
| --- | --- |
| Module Title: | Programming for Data Analysis, Processing, & Visualisation |
| Module Code: | B9DA100 |
| Module Leader: | Darren Redmond |
| Stage if relevant): |  |
| Assessment Title: | Utility Bill Management |
| Assessment Number if relevant): | 4 |
| Assessment Type: | Practical Programming |
| Individual/Group: | Individual |
| Assessment Weighting: | 30% |
| Issue Date: | 27th November 2019 |
| Hand In Date: | 22nd December 2019 |
| Planned Feedback Date: | 1 weeks post hand-in date. |
| Mode of Submission: | Online Via Moodle & Github |

**Students must submit their assignment on/before the given deadline on Moodle.**

**Email submissions will not be accepted.**

Python Programming Assignment

Dublin Bill Management Company is a start-up bill management company, i.e., they offer tracking for electricing, gas, and utility bills.

They have approached you to develop a command line interface python system that will record all of their customers bills, together with a menu to generate queries/reports that can be run against this data.

The following is some sample customer data that you should make use of (the full list is available as the file results.csv on Moodle):

Electric Ireland, John Smyth, 2017, 05, 12, 11.58, credit  
 Energia, Missy May, 2016, 12, 22, 122.52, debit  
 Vodafone, John Smyth, 2016, 11, 17, 20.00, debit  
 Energia, Susie Sue, 2016, 11, 03, 25.00, debit  
 Vodafone, Susie Sue, 2016, 11, 17, 5.00, credit

# Data Description

The file is a comma separated value (csv) file – each field is separated by a comma. The first field is the utility company, the second is the customer name, the third is the year, fourth is the month, fifth is the day, i.e. date (in YYYY, MM, DD format), the sixth field is the amount of the bill, while the last field is a flag indicating whether this is a credit or debit against the bill.

Write a python console application to capture customers and their utility bills.

You will need to be able to store utility bill details, create them and produce a number of reports (see below).

For this assignment, a new utility bill should store the supplier company, the name of the customer, the date on which the bill was raised, the amount, and an indicator to specify whether the bill is a debit or a credit.

# Requirements and Reports/Queries

1. Provide a way for a user to enter utility bill details: utility company, name of the customer, date of the bill, the amount, and a flag indicating whether the bill is debit or credit.
2. Start your code with the initial bills.csv above. However, you need to provide a mechanism for writing and reading these utility bills to a text file.

This option should be provided via a menu choice.

1. Provide a report that lists years, total credited and total debited, e.g., the output will look like the following:

Year Total Credited Total Debited

2016 €123.45 €678.90

2017 €543.21 €987.60

1. Provide a report that shows the most popular utility company. The most popular utility company is the one with the most bills against that provider.
2. Provide a report that shows the bills in date order.
3. Provide another report that displays the highest amount for a bill that is a credit, and one for a debit.
4. Provide a report to indicate how successful the company is. This should display the total number of bills.
5. Provide a report to calculate the average spent per period of time (month/year) that can be entered by the user.
6. Provide a report to calculate the average time between bills.

# Notes

1. Please ensure that you build your code base using Test Driven Development, TDD and the python unittest module should be used to complete your tests. So, remember: Red🡪Green🡪Refactor.
2. For writing and reading bills from and to a text file, it is entirely up to you how you do this.
3. Ensure you write your code in a defensive coding manner.

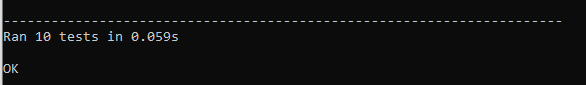
# **Description**

All the below steps were performed based on the original bills.csv file + any bill entered by the user during the execution time.

## Step 1 – Run unit tests

Run the bill\_management\_test.py class

The result should be as per below picture (Ran 10 tests OK)

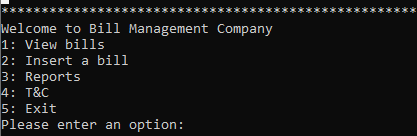


This class is going to read bills.csv file and apply some verifications including insert a new line to validate the method insert and in the end it will restore the original file (copying from bills-Copy.csv).

## Step 2 – Run application

Run the bill\_management.py class. It should start with the information from bills.csv file and in the end any new bill entered by the user should be written on the file along with the existing bills.

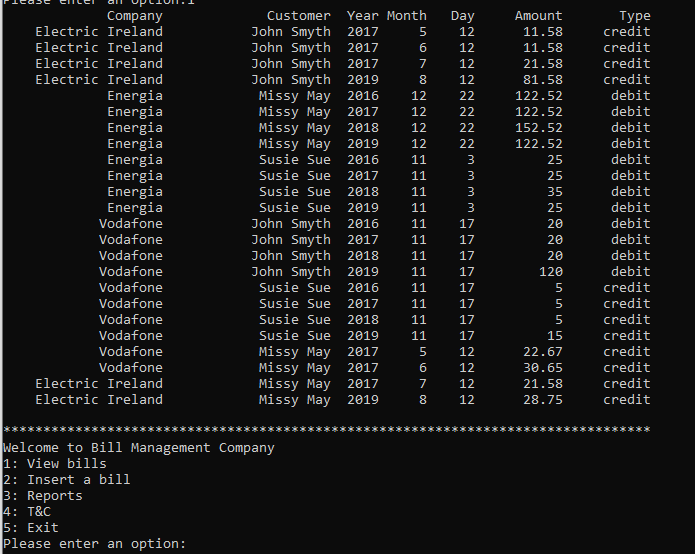
The result should be as per below picture (main menu options)



### Option 1: View bills

The result should be as per below picture.

It should list all the bills from the bills.csv file and at the show the main menu options again allowing the user to choose another option.



### Option 2: Insert a bill

The user should be asked for:

company name

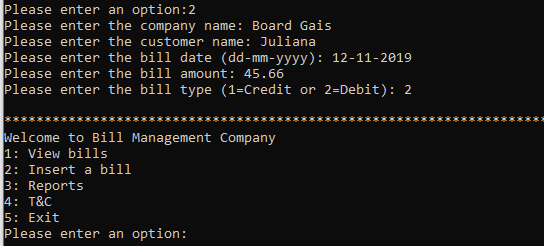
customer name

bill date

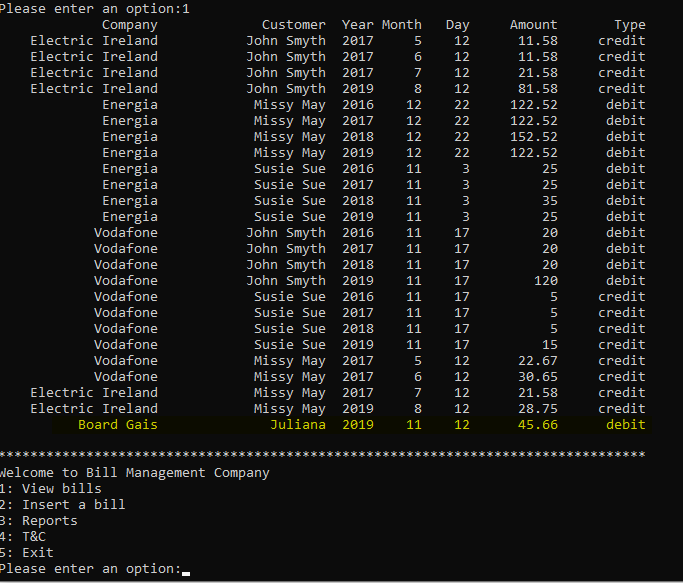
bill amount

bill type

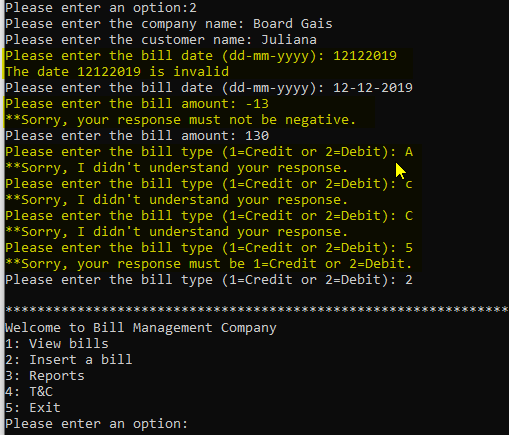
After pressing Enter in the last field (bill type), it should insert the new bill in the current list and return to the main menu options again allowing the user to choose another option.



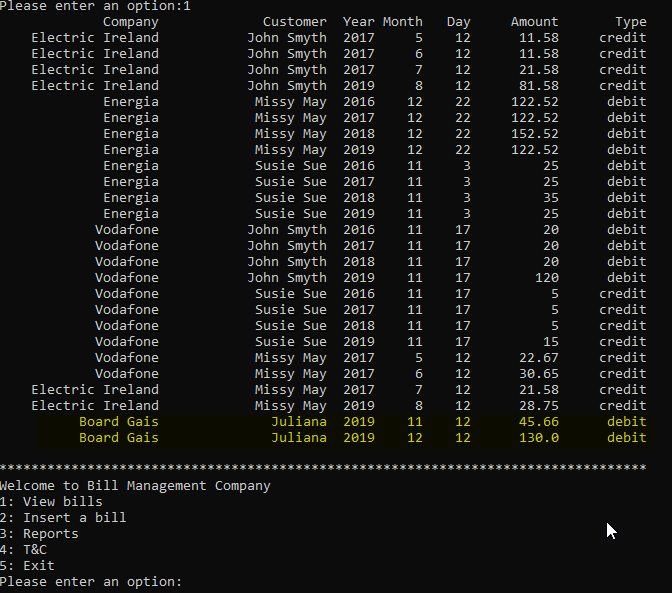
Choose Option 1 again and you should see the new bill information in the end of the list



Any invalid data input for bill date, bill amount and bill type should return an error message

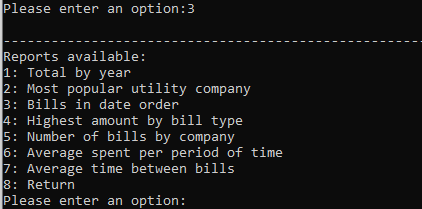


Choose Option 1 again and you should see one more line in the end of the list



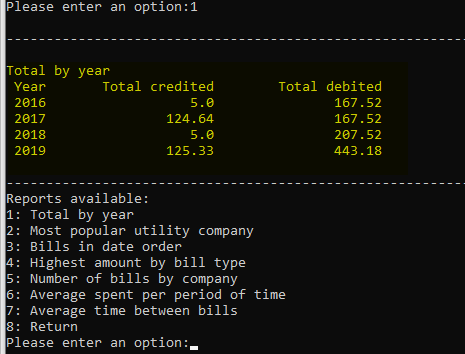
### Option 3: Reports

The reports menu should be display as per below picture:



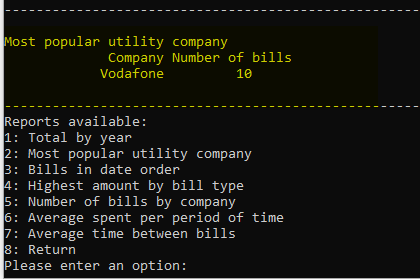
#### Option 1: Total by year

Displays a report that lists years, total credited and total debited in the below format and show the reports menu options again allowing the user to choose another option.



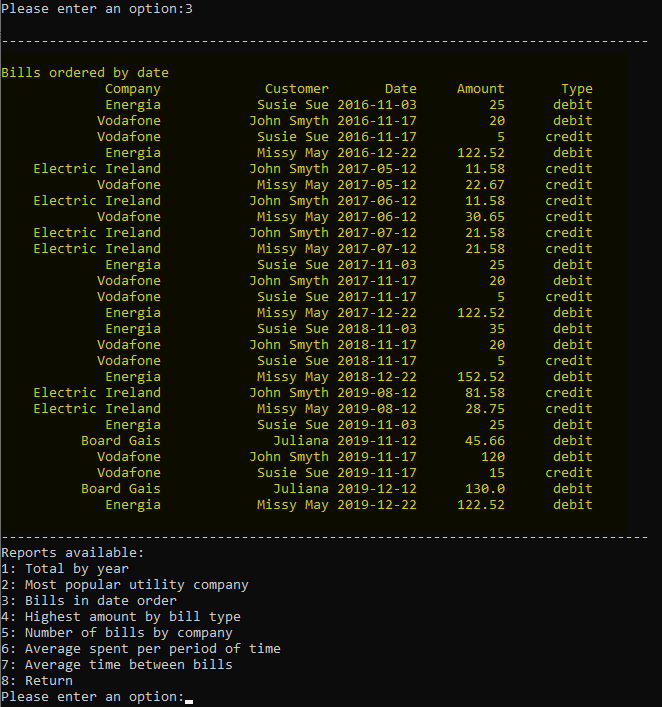
#### Option 2: Most popular utility company

Displays a report that shows the most popular utility company in the below format and show the reports menu options again allowing the user to choose another option.



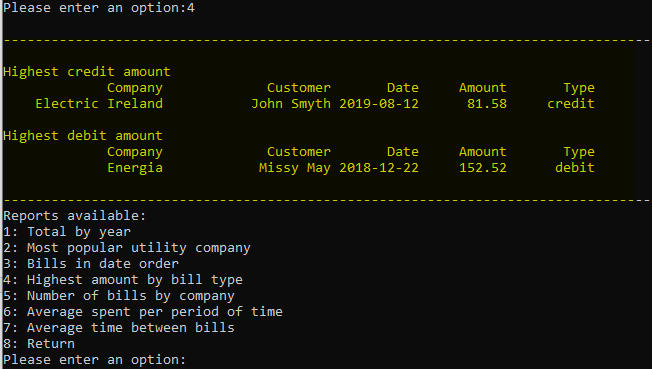
#### Option 3: Bills in date order

Displays a report that shows the bills in date order in the below format and show the reports menu options again allowing the user to choose another option.



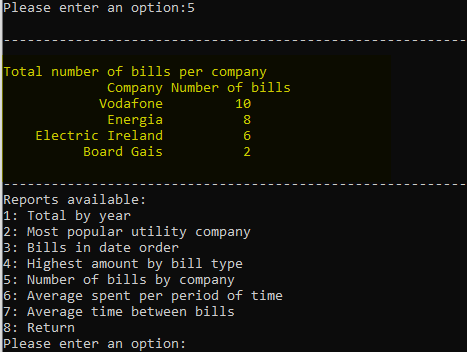
#### Option 4: Highest amount by bill type (debit/credit)

Displays a report that shows the highest amount for a bill that is a credit, and one for a debit in the below format and show the reports menu options again allowing the user to choose another option.



#### Option 5: Number of bills by company

Displays a report to show how successful the company is (the total number of bills) in the below format and show the reports menu options again allowing the user to choose another option.



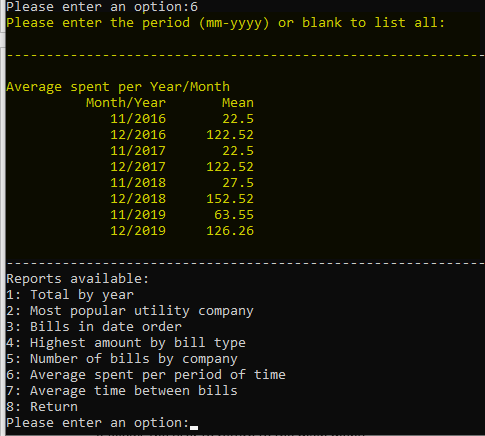
#### Option 6: Average spent per period of time

Displays a report to show the average spent per period of time (month/year), that can be entered by the user, in the below format and show the reports menu options again allowing the user to choose another option.

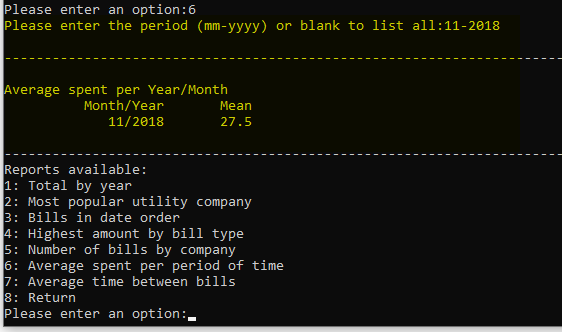
Firstly, the user will be asked for a period to be checked/listed.



If blank, it will be displayed the average spend for each month/year in the bills list.



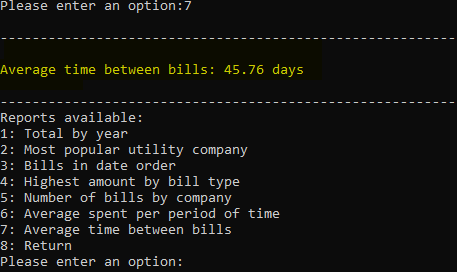
If the user enters month/year, it should list only the average for that month/year.



Note that for this average I am considering only accounts where type is debit.

#### Option 7: Average time between bills

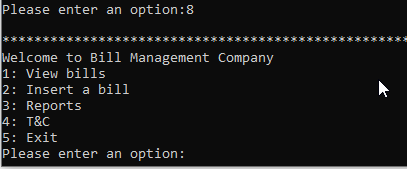
Displays a report to show the average time between bills in the below format and show the reports menu options again allowing the user to choose another option.



Note that for this average I am considering all bills in the list.

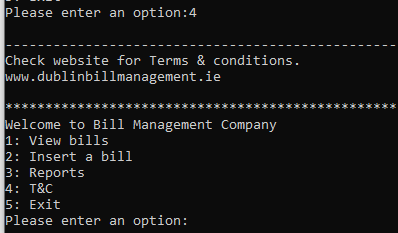
#### Option 8: Return

It allows the user to return to the main menu.



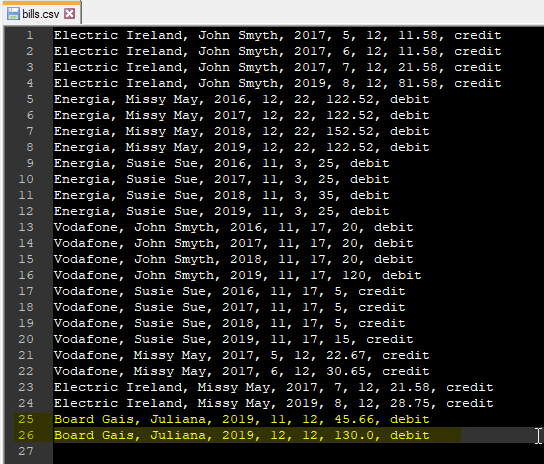
### Option 4: T&C

It should display the below message



### Option 5: Exit

Use this option at any time, when the main menu is being displayed, to exit the application. Any new bill entered by the user will be written on the bills.csv file along with the existing bills.



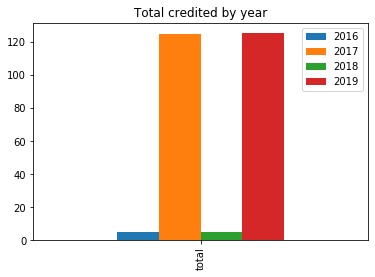
## Step 3 - Visualization

Run the bill\_management\_visualization.py class.

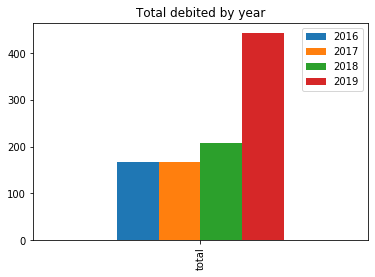
Note that this class uses methods from bill\_management class, so both needs to be in the same work directory.

It should list the below graphs:

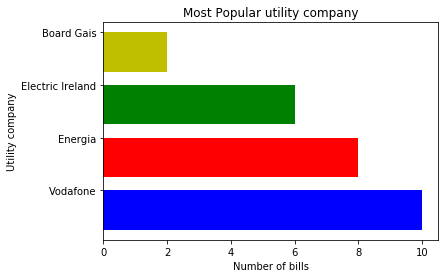
#### Total credited by year



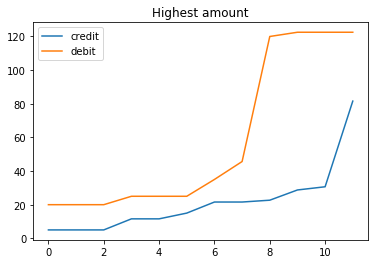
#### Total debited by year



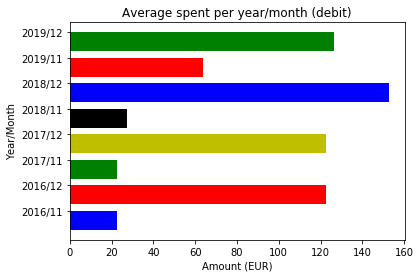
#### Most popular utility company



#### Highest amount for a bill that is a credit and for bill that is a debit



#### Average spend per year/month (debit bills)



#### Amount spent per consumer

