Assessment Brief

|  |  |
| --- | --- |
| **Module Title:** | Programming for Data Analysis, Processing, & Visualisation |
| **Module Code:** | B9DA100 |
| **Module Leader:** | Darren Redmond |
| **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 electricity, 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.

REPORT

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Supplier company** | **Customer name** | **Date billed** | **Amount** | **Debit/Credit** |

# Requirements and Reports/Queries

1. Provide a way for a user to **enter utility bill details** (USER INPUT): 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 (OPEN). 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.
3. 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 (USER).
  6. Provide a **report** to calculate the **average time between bills**. -> USER

# Notes

1. Please ensure that you build your code base using Test Driven Development (TEST MIN 3 FUNCTIONS), 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 (USER-PROOF).

# Marking Scheme

|  |  |
| --- | --- |
|  | Marks |
| Document code with sensible function and variable names, code can run. | 20 % |
| Unittest can run. | 20% |
| The 6 reports: 5 marks each. | 30% |
| Test manual: how does code run, eg: option 1, option 2… Exceptions. | 30 % |
| TOTAL | 100% |

# Extra features