

Solent University Southampton
Faculty of Computing, Programming Technologies

Solar Record Management System

Author : 10152420
Module title : Problem Solving Through Programming
Module Code : QHO426
Tutor's name : Yaroslav Basha
Word Count :1028 words (+-10%)

Contents

<i>Description</i>	<i>page number</i>
List of Illustrations	
1. Introduction	1
2. Methodology	1
3. Results & Discussion	1
4. Conclusions	9
5. Reference list	10

List of Figures

<i>Description</i>	<i>page number</i>
1. First illustration	2
2. Second illustration	3
3. Third illustration	3
4. Fourth illustration	4
5. Fifth illustration	4
6. Sixth illustration	4
7. Seventh illustration	4
8. Eighth illustration	4
9. Ninth illustration	5
10. Tenth illustration	5
11. Eleventh illustration	5
12. Twelfth illustration	6
13. Thirteenth illustration	6
14. Fourteenth illustration	7
15. Fifteenth illustration	8
16. Sixteenth illustration	9
17. Seventeenth illustration	9

1. Introduction

The task here is to create a software application that can be used to display and process solar system data. The data will be present in a CSV file format and the data will cover non-planets and planets present in the solar system.

2. Methodology

There are a series of steps required to be taken to form the application. The primary step is going to be to show the welcome message stating, 'Solar Record Management System'. The menu containing the options: Load data, process data, visualize data, save data and exit will be displayed. These steps are then enforced as per user requirements.

3. Implementations, Results and Discussion

There are four part in this systems. Which are as follows:

- Data loaded from CSV file
- Data process
- Data visualize and animated
- Data Export as JSON format

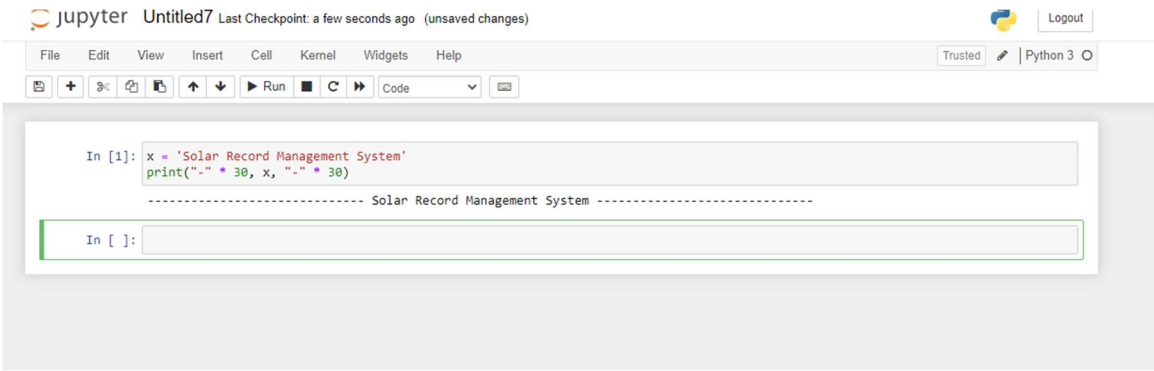
There are mainly three files to work with - 'tui.py', 'visual.py' and 'main.py' which consist of 30 task are as follow:

- Tui - It consist of 16 task to create welcome message , various menu options, user input and more.
- Visual - It consist of 4 task to display plots and animate pie and bar charts.
- Main - It consist of 10 task where contains the main logic for this solar system applications such as calling various functions from tui and visual, loading data from CSV file into a list and then finally all data export to json via OOP abstract method.

Implementation and outputs are as below:-

- 🚦 As per the task 1 displaying a welcome message, I have used single variable and multiply of 30 dashes before and after into the print function.

Figure 1



The screenshot shows a Jupyter Notebook window titled 'Untitled7'. The top bar includes the Jupyter logo, the title, and a status message 'Last Checkpoint: a few seconds ago (unsaved changes)'. Below the title bar is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', 'Widgets', and 'Help'. A toolbar with various icons is located below the menu bar. The main area contains a code cell with the following code:

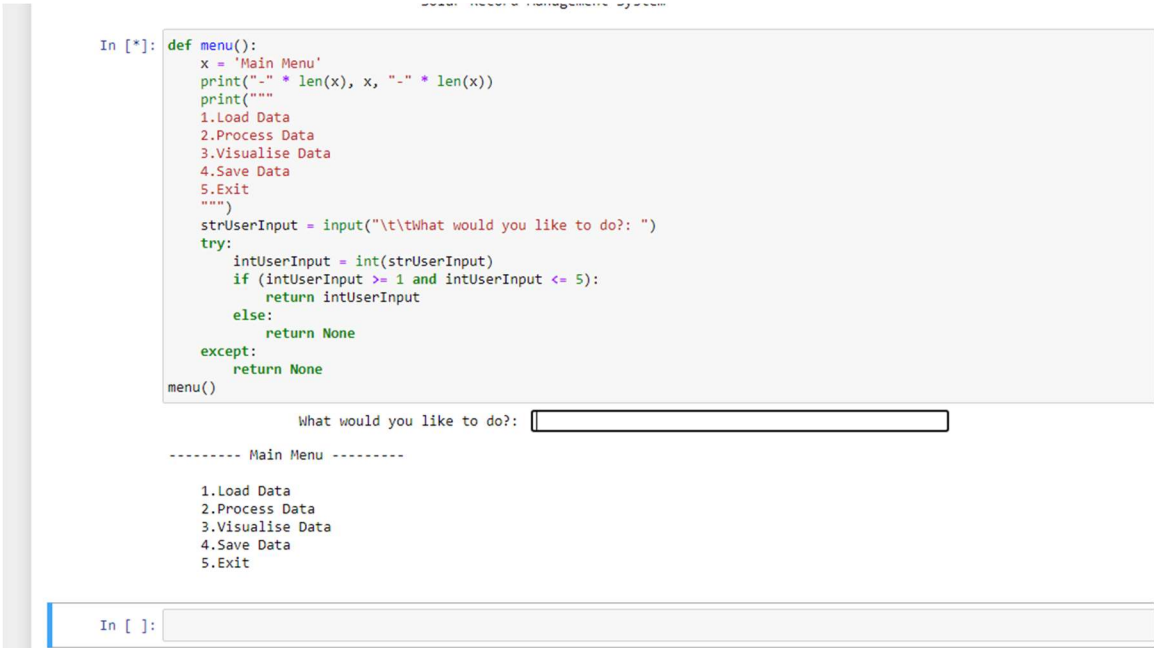
```
In [1]: x = 'Solar Record Management System'
print("-" * 30, x, "-" * 30)

----- Solar Record Management System -----
```

Below the code cell is an input prompt 'In []:' with an empty text box.

Tasks 2, 7, 15 and 16 are similar concepts, logics I have used to display the text-based menu options. Several times I have used different logics, methods for that which fulfil the task, but I am not satisfied. After lot of internet search, discuss with tutor, classmates and others, I have used finally the 'try and except block' to catch and handle the exceptions. As Python executes code following the try statement as a normal part of the program because of this exception error will crash the program if it is unhandled. In these tasks I have used to print of length of 'x' dashes before and after where 'x' is the variable, then print multiline messages, user input and finally under 'try and except block' to run as integer input value arrange is 1 to 5.

Figure 2



The screenshot shows a Jupyter Notebook window with a code cell containing the following code:

```
In [*]: def menu():
x = 'Main Menu'
print("-" * len(x), x, "-" * len(x))
print("""
1.Load Data
2.Process Data
3.Visualise Data
4.Save Data
5.Exit
""")
strUserInput = input("\t\tWhat would you like to do?: ")
try:
    intUserInput = int(strUserInput)
    if (intUserInput >= 1 and intUserInput <= 5):
        return intUserInput
    else:
        return None
except:
    return None
menu()
```

The output of the code is displayed below the code cell:

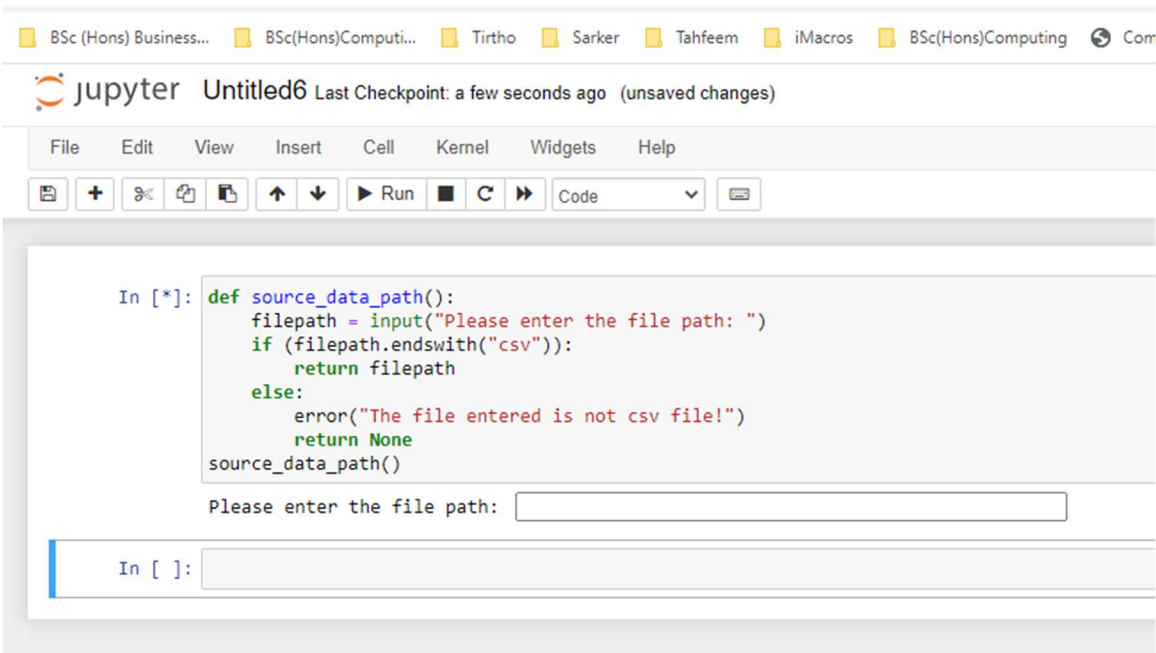
```
What would you like to do?: 
----- Main Menu -----
1.Load Data
2.Process Data
3.Visualise Data
4.Save Data
5.Exit
```

Below the output is an input prompt 'In []:' with an empty text box.

Tasks 3, 4 and 5 also same formatted print functions to used notifications word of starting, completing and to have an error messages.

Task 6 is interesting where the user to enter the file path for a data file. I have used here input function where user's enter file name with csv extension and to detection of the file name have a csv extension, I have applied here '.endwisth()' and then calling my previously created 'error()' function on task 5 to display error message if user entered wrong file path or name. Here I have not used error handling as this function calling from others place so I do not need to.

Figure 3



Rest of the ‘tui’ tasks I will put comments on my code.

Figure 4

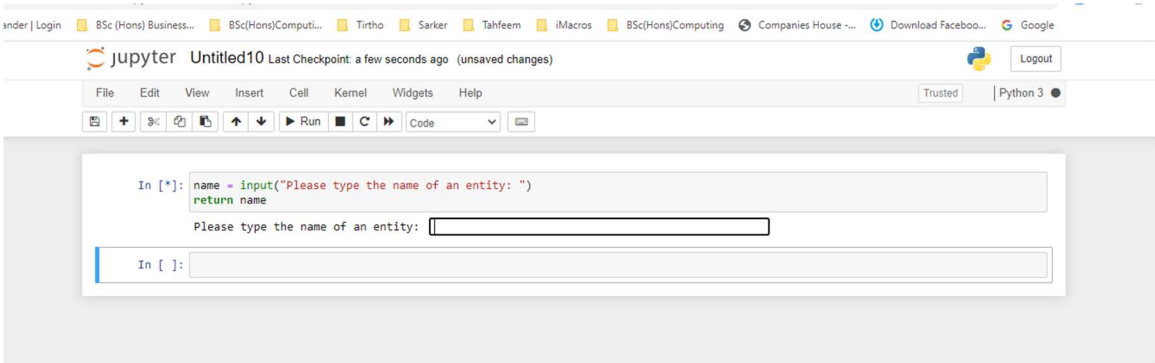


Figure 5



Figure 6



Figure 7

```
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
In [2]: def list_entities(entities, cols=[]):
        for entity in entities:
            if len(cols) > 0:
                list = [entities[z] for z in cols] # Using List comprehension
                #print(list)
            else:
                print(entities)
            return
        list_entities(['Earth', True, 9.8, 0, 0], [])

        ['Earth', True, 9.8, 0, 0]
```

Figure 8

```
] def list_categories(categories):
    for key, value in categories.items():
        print(key, ' : ', value)
    # The param 'categories' is a dictionary type called by main.py
```

Figure 9

```
In [1]: def gravity_range():
        ans = True
        while(ans):
            try:
                minGravity = float(input("lower limit:- "))
                maxGravity = float(input("upper limit:- "))
                ans = False
            except:
                continue
            if (minGravity > maxGravity):
                a = maxGravity
                maxGravity = minGravity
                minGravity = a # (www.youtube.com, n.d.)
            listGravityRange = [minGravity, maxGravity]
            tupleGravityRange = tuple(listGravityRange)
            return tupleGravityRange

        print(gravity_range())

        lower limit:- 0.001
        upper limit:- 9
        (0.001, 9.0)
```

Figure 10

```
In [1]: def orbits():
        entityNames = input('Enter entity names(e.g. Jupiter,Earth,Mars): ')
        entityNameList = entityNames.split(",")
        return entityNameList

        print(orbits())

        Enter entity names(e.g. Jupiter,Earth,Mars): Jupiter, Urenas, Neptune, Eart, Mars
        ['Jupiter', ' Urenas', ' Neptune', ' Eart', ' Mars']
```

🚀 I have done all task in ‘visual.py’ where required module imported including numpy. I have used some inbuilt numpy function that returns an ndarray object containing evenly spaced values within the length from categories. I have used zip function to add text inside the plot. I have loop over the plotlays by using enumerate() function. Below images confirms final output:

Figure 11

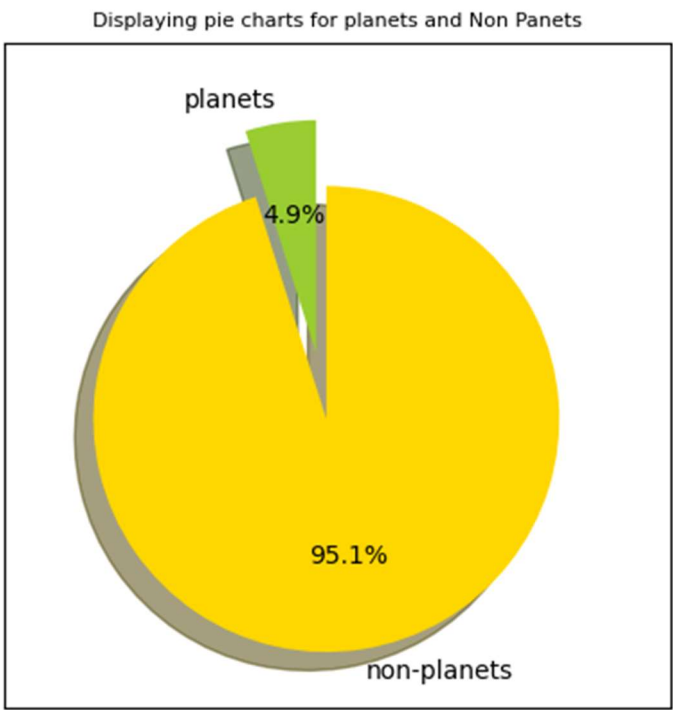


Figure 12

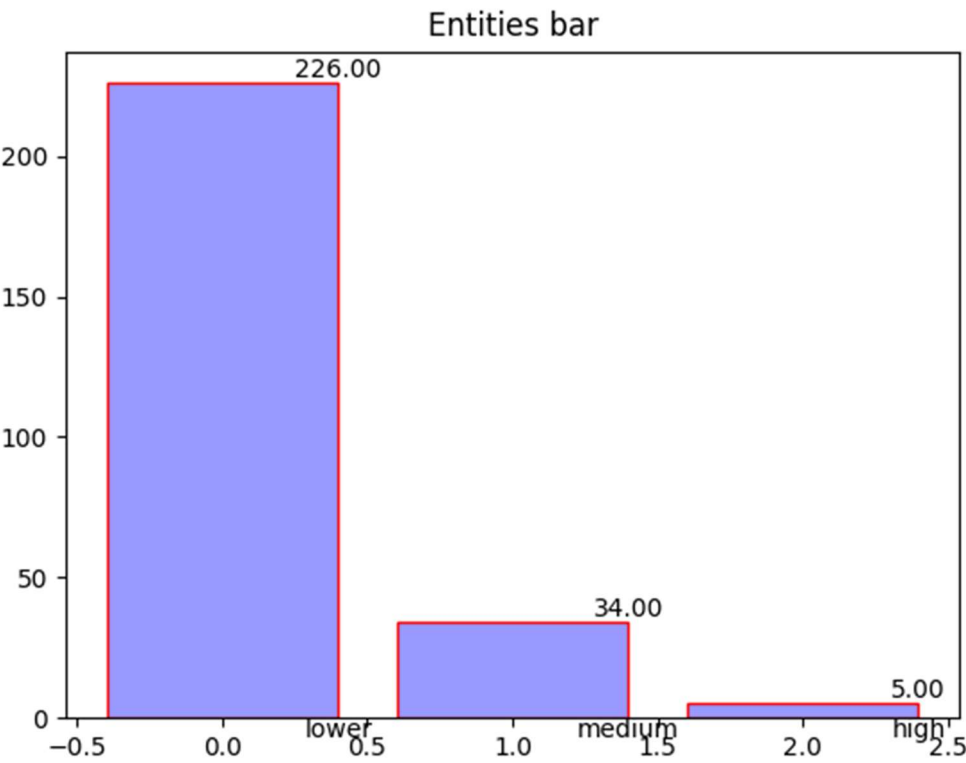


Figure 13

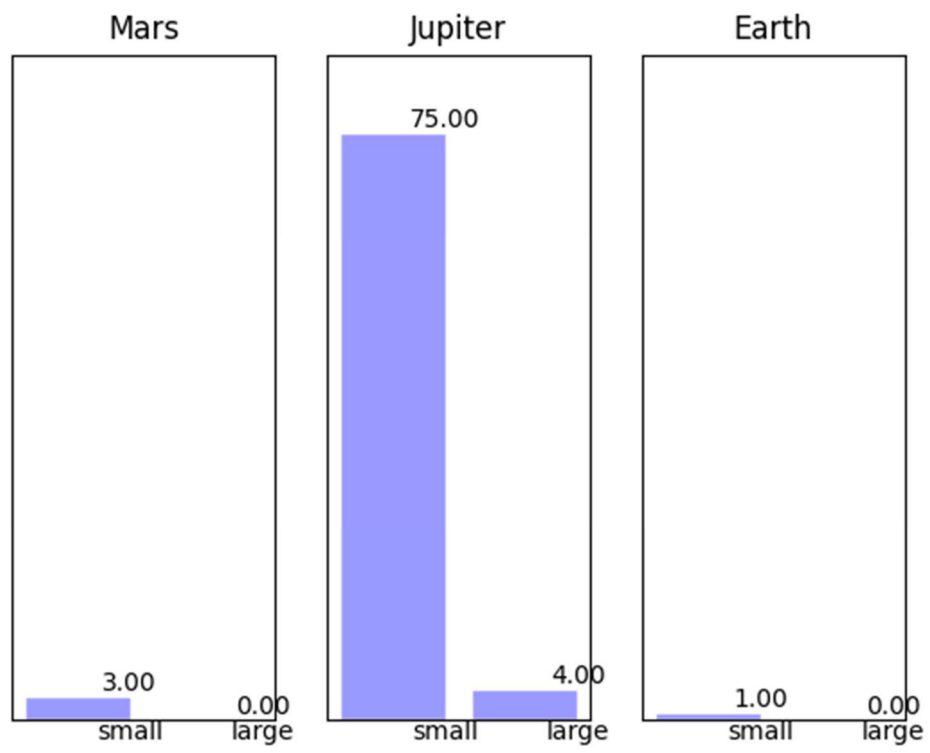
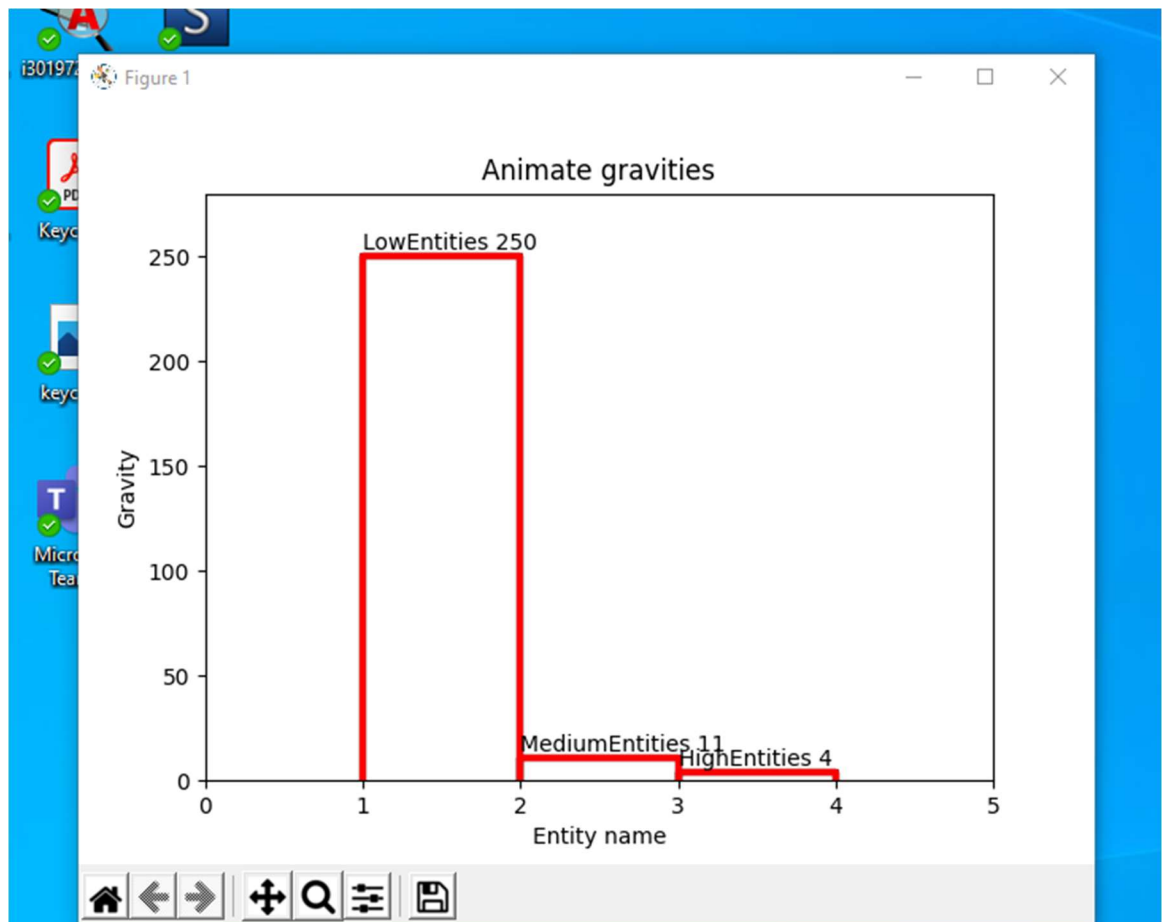


Figure 14



Finally, I done all the tasks for 'main.py' where task 17 I have imported csv, tui, visual, abstractmethod from ABC module and json, task 18 empty records list created, from tui I called welcome(), menu(), functions on task 19 and 20 accordingly. Here I go through line by line and called various functions where needed. In this section I have read data from CSV file and add it to the list, retrieve file path for CSV data file and handle with try and except block on task 21 and then go to task 22 and 23 to load and visualize data via module tui and visual where all the functions , pie chart, bar chart plotted. After successfully reach this stage, I easily done rest tasks to export data into json file format via abstract class. In this section I have commented all work I have done into the code so that anyone can manage this application in future for upgrade version. Here is the output below:

Figure 15

```
In [1]: from abc import ABC, abstractmethod
import json

records = []

class AbstractWriter(ABC):
    def __init__(self, recordsdata):
        self.recordsdata = recordsdata

    @abstractmethod
    def RecordsToJson(self):
        pass

class ConcreteWriter(AbstractWriter):
    def RecordsToJson(self):
        print('I am ok')

w = ConcreteWriter(records)
w.RecordsToJson()

I am ok

In [ ]:
```

Figure 16

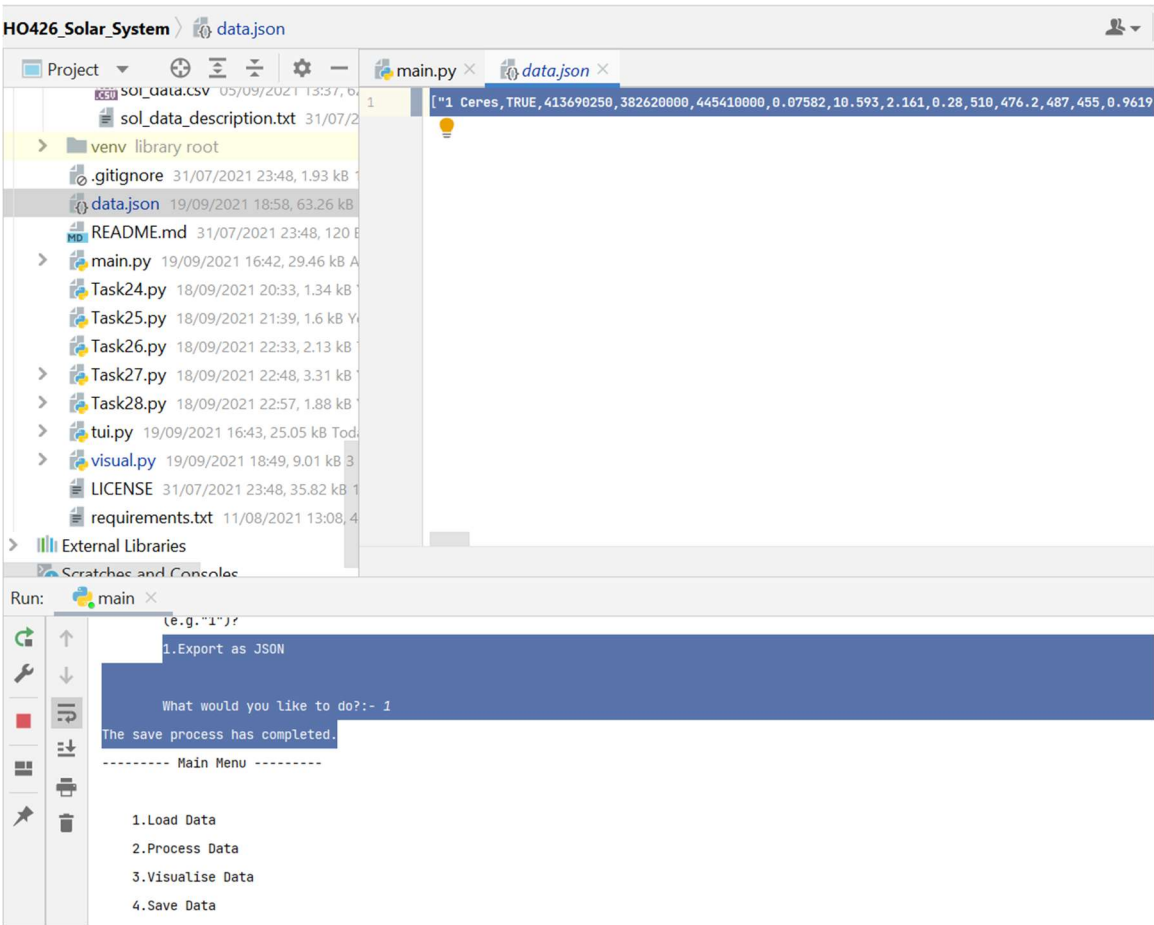
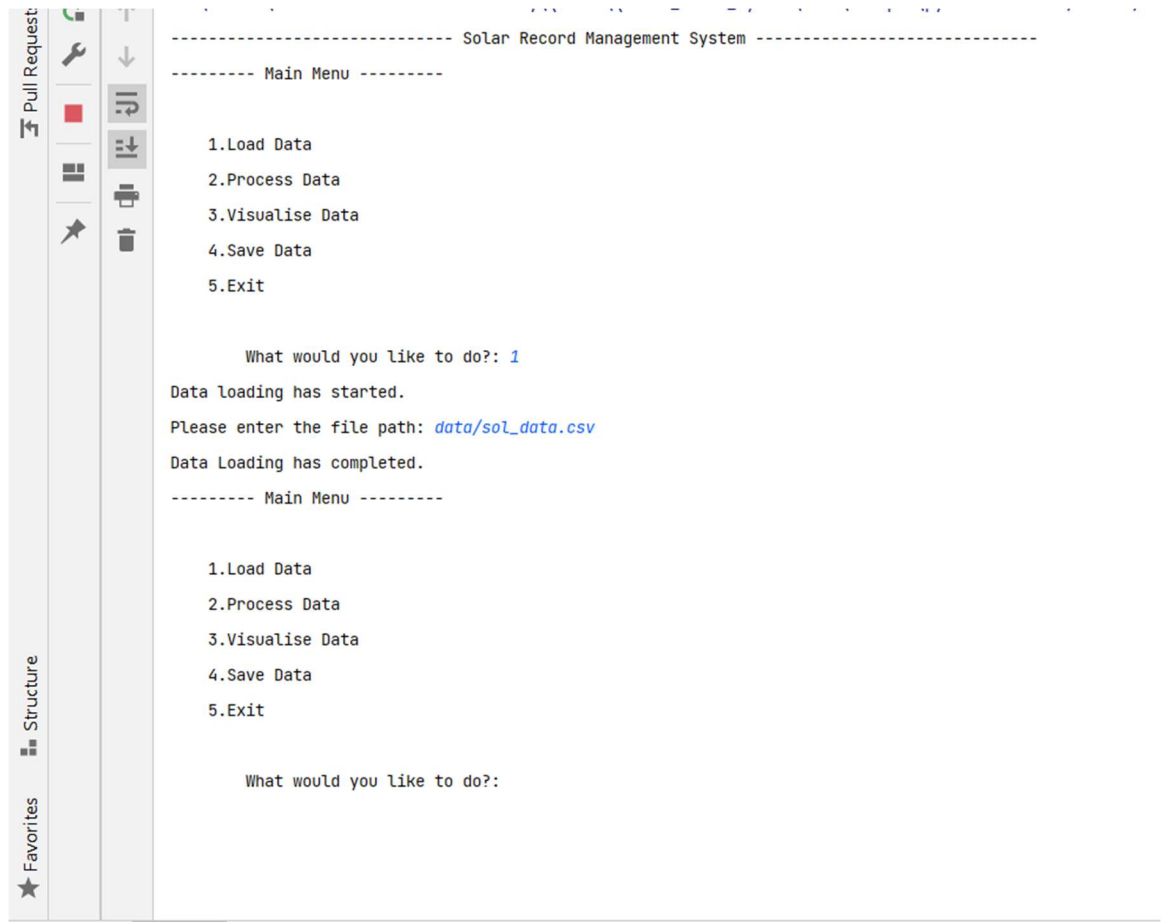


Figure 17



4. Conclusion

Finally solar system project successfully completed. At this stage I can now loaded CSV file data into a list to process and visualize and export data via json file format using OOP abstract method.

5. Reference list

- ✚ [www.guru99.com](https://www.guru99.com/python-check-if-file-or-directory-exists.html). (n.d.). *Python Check If File or Directory Exists*. [online] Available at: <https://www.guru99.com/python-check-if-file-exists.html>.
- ✚ [learn.solent.ac.uk](https://learn.solent.ac.uk/mod/page/view.php?id=2049756). (n.d.). *Solent Online Learning: Log in to the site*. [online] Available at: <https://learn.solent.ac.uk/mod/page/view.php?id=2049756> [Accessed 4 Sep. 2021].
- ✚ Solent University (n.d.). *Solent Online Learning: Log in to the site*. [online] learn.solent.ac.uk. Available at: <https://learn.solent.ac.uk/mod/page/view.php?id=2049728>.
- ✚ # GeeksforGeeks. (2019). *Python | Get a list as input from user*. [online] Available at: <https://www.geeksforgeeks.org/python-get-a-list-as-input-from-user/?ref=leftbar-rightbar> [Accessed 4 Sep. 2021].