Solent University
Coursework Assessment Brief

Assessment Details

Module Title:	Problem Solving Through Programming
Module Code:	QHO426
Module Leader:	Piotr Bednorz
Level:	4
Assessment Title:	Software Artefact with Documentation
Assessment Number:	2
Assessment Type:	Software Artefact + Documentation
Restrictions on Time/Word Count:	4 code files (+ additional classes for A-grade)
Consequence of not meeting time/word count limit:	It is essential that assignments keep within the time/file/word count limit stated above. Any work beyond the maximum time/file/word length permitted will be disregarded and not accounted for in the final grade.*
Individual/Group:	Individual
Assessment Weighting:	50%
Issue Date:	21st November 2023
Hand In Date:	9th February 2024
Planned Feedback Date:	4 weeks after deadline
Mode of Submission:	On-line Only FINAL submissions will be accepted. DRAFT submissions will not be considered an attempt and will not be marked.
Anonymous Marking	This assessment: Will be marked anonymously

Assessment Task

In this assessment, you will develop an application to analyse data from a publicly available dataset. The dataset will be provided to you as a CSV file, which accompanies this assessment brief.

Accompanying CSV file

The CSV file is called disneyland_reviews.csv and contains thousands of reviews about Disneyland parks around the world. The disneyland_reviews.csv file contains the following information, with each row in the file representing a single review:

Column	Description	Type
Review_ID	A unique identifier, contains no useful information	Int
Rating	The score given by the reviewer out of 5	Int
Year_Month	The year and month the review was made	String
Reviewer_Location	Where the reviewer is from	String
Branch	Which Disneyland Park is being reviewed	String

It is recommended that you familiarise yourself with the content of the data file before attempting the remainder of this assessment.

Accompanying Code Files

In addition to the CSV file, you are also provided with a zip file containing 4 python modules:
main.py
tui.py
process.py
visual.py

You are expected to confine your code to these four modules. Creating additional modules or unexpected code architectures will be seen as a lack of understanding.
The A grade task, however, will require you to use OOP and therefore develop classes. These should be placed in their own module as best practice requires. Therefore, any OOP classes you create are the only exception to this rule. All other code must be within the four provided modules.

The Software Artefact

This section outlines what you are expected to code within the provided modules. The coding tasks are separated into sections that will get progressively more involved. As you can expect, if you are looking to achieve a high grade, you will need to complete all the sections, whereas a D grade can be achieved even if some later sections have not been attempted.
Please note, completing all the tasks will not achieve a high grade on its own, you will also need to code to a good standard:

- Submissions in which most of the code is written in main.py, in a primarily linear fashion will not score highly.

- In contrast, submissions that logically break down their code into reusable functions/classes, place functions in the most logical of the provided modules, make very good use of parameters, return values and OOP and conform to PEP8 standards will score highly.

You are expected to use Git and GitHub as part of this assessment. Your GitHub repository must be private!

Please see the assessment criteria (further down in this document) for more information about how this assessment will be graded.

Below are the individual tasks, you should attempt each section one at a time, starting with 'section A' and working through in the correct order. You are strongly discouraged from attempting the sections in a different order (i.e., doing 'section C' before 'section B'). This will almost certainly lead to difficulties with the later tasks.

Please note. The program should run from **main.py**! The marker should not have to run any of the other module separately to see functionality. This means, if you use the other modules, then they will need to be imported in to main.py and implemented correctly.

----- Section A -----

We shall begin in main.py. You may create functions and separate them into the other modules – however, how you do this is strictly down to you.

1. First, when running the program, it should display the following (take note of the annotation):

```
-------------------------
Disneyland Review Analyser
-------------------------
```

The number of dashes should equal the number of characters in the title

2. Read in the data from the provided CSV file, said data should be stored in a list. The program should confirm to the user when it has finished reading in the dataset. It should also tell the user how many rows are in the dataset.

3. Output the following to the screen, this will act as a menu. The user should then be able to input their selection, which should be stored in a suitable variable.

```
Please enter the letter which corresponds with your desired menu choice:
    [A] View Data
    [B] Visualise Data
    [X] Exit
```

4. The program should confirm what the user has entered. If the user entered an invalid menu choice, then they should be informed of their mistake.

   Here is an example of what it would look like if the user entered 'A':

```
Please enter the letter which corresponds with your desired menu choice:
    [A] View Data
    [B] Visualise Data
    [X] Exit
A
You have chosen option A - View Data
```

5. The program should run continuously. Once the user has entered their choice, it should confirm this choice and then display the menu again, asking the user for their selection. Be careful how you implement this, the program does not need to display the title and load in the dataset again. The program should only end if the user indicates they wish to exit the program through the appropriate menu choice.

6. If the user selects 'A' as their menu choice, the program should display the following sub-menu and receive the user's choice:

```
Please enter one of the following options:
    [A] View Reviews by Park
    [B] Number of Reviews by Park and Reviewer Location
    [C] Average Score per year by Park
    [D] Average Score per Park by Reviewer Location
```

   If the user selects 'B' as their menu choice, the program should display the following sub-menu and receive the user's choice:

```
Please enter one of the following options:
        [A] Most Reviewed Parks
        [B] Average Scores
        [C] Park Ranking by Nationality
        [D] Most Popular Month by Park
```

----- Section B -----

7. We will now focus on the sub-menu that is displayed should the user choose 'A' at the main menu.
   The first sub-menu option will allow the user to see all the reviews for a specific park. If the user selects this option, then the program should respond by asking which park the user wishes to see the reviews for. The program should then display all reviews for said park.

8. The second sub-menu option will simply display the number of reviews a specific park has received from a given location. Both the park and the reviewer's location should be retrieved from the user.

9. If the user chooses the third sub-menu option, then the program will ask the user for a park and a year, it will then display the average rating for the given park in the given year.

The final option for this sub-menu will be developed in a later section.

----- Section C -----

We will now focus on the sub-menu that is displayed should the user choose 'B' at the main menu.

10. The first sub-menu option should display a pie chart showing the number of reviews each park has received.

11. The second sub-menu option should present the average review scores for each park as a single bar chart

12. The third sub-menu option will ask the user to enter a park. It will then display a bar chart that shows the top 10 locations that gave the highest average rating for that park.

13. The final sub-menu option will ask the user to enter a park. It will then display a bar chart that shows the average rating that park received for each month of the year. You do not need to worry about the year for this task, so May 2018 and May 2019 will both be simply counted as May.
    The bar chart should be ordered by month.

    The final output should be a bar chart with the months of the year (in order) along the X axis and the average rating on the Y axis.

----- Section D -----

14. Moving back to the submenu that is displayed if the user enters 'A' in the main menu:

    The final sub-menu option for the 'view data' menu is to display the average score per park by reviewer location.
    This, for every park, should output the average rating for every single location it has received a review from.
    The user should not have to enter any information for this task.

15. To achieve the highest grades, you will need to demonstrate Object Orientation. You will need to develop an 'export data' feature using OOP. The exporter should output aggregate information for each park including:
    Number of reviews.
    Number of positive reviews.
    Average review score
    Number of countries that have reviewed each park.
    The system should provide the option for the output to be in any of the following 3 formats: TXT, CSV or JSON.
    You should add an additional option (C) to the main menu, allowing the user to access this feature and select the format they wish to output the data to.
    You should include a rough class diagram for this in your supporting documentation (see below).

The Supporting Documentation

In addition to the software artefact, you will also need to submit either a word document or a PDF that contains the following (and no more!):

- A table that declares which tasks you feel you have completed, which were attempted but incomplete and which were not attempted.

- Copy of your code from all implemented modules, pasted as text. Simply add a header for each file name and paste the content of your .py files one after the other. Do not paste your code as pictures/screenshots.

- Evidence of your use of Git/GitHub. This should be in the form of two screenshots. The first should be a screenshot of your most recent commits, as displayed in the commit history page on GitHub. The second screenshot should show the oldest commits as listed in your commit history on GitHub. Your username should be clearly visible in the screenshots.
  You should include as many commits in the screenshots as you can fit.

- Evidence that you have tried to conform to PEP8 standards. This can be evidenced by a screenshot of PyCharm either confirming there are no errors or a screenshot of the errors that exists with a short statement confirming that you were unable to fix them.

- If you attempted the final task – a rough class diagram showing how you implemented OOP into the program.

Technologies

You must use Python and only **Python** for this assessment.
Only one third party library may be used, that being **MatPlotLib**.

Deliverables (What you should submit)

You will need to submit 2 files in total:

1. A **.zip** file containing the latest working version of your software project

2. Either a **.doc, .docx** or **.pdf** file containing the supporting documentation as outlined in the previous section. This should not be included in the zip file but instead submitted as a separate file.

- Please use your student number as the file names

Both files MUST be submitted as instructed.
Submissions that are missing the zip file or are corrupted will be automatically allocated an 'S' grade.
Submissions that are missing the supporting documentation, or the supporting documentation does not follow the points outlined in the above section, will be severely impacted!

Please read the FAQ section under the 'assessment' tab on the SOL page.

Assessment criteria

Grades	Criteria
S, F3, F2, F1	The submitted software artefact does not run, or a .zip file containing a working version of the program was not submitted to SOL, or the submitted file was corrupted/empty/unable to be opened. OR The marker is unable to test the program by simply running main.py. OR A lack of understanding is demonstrated, either through: • Not successfully completing the required tasks. • By submitting code that uses unapproved technologies/libraries which are not in line with those outlined in the 'technologies' section of the assessment brief. • By submitting unexpected architectures that are very different to what has been taught on the module. • By submitting code that is not confined to the provided modules. • Git/GitHub was not used or there is no evidence of its use submitted to SOL

	OR No supporting documentation has been submitted or the supporting documentation is off topic and does not follow the points requested in the 'supporting documentation' section of the assessment brief. OR The supporting documentation was submitted inside the .zip file and therefore did not generate a TurnItIn report. OR The criteria for a D grade (below) have not been met. OR There is evidence of plagiarism, collusion, or any other form of academic misconduct (Make sure your GitHub repository is private!).
D3, D2, D1	None of the S or F criteria have been met. AND The 'Section A' tasks have been completed. AND Supporting documentation has been submitted and is in line with what is asked for in the 'supporting documentation' section of the assessment brief. AND Code is largely linear or not organised into modules, however there should be some demonstration of functions, parameters and return values. AND There is a clear attempt to apply PEP8 standards, however some errors may exist.
C3, C2, C1	The D grade criteria has been met. AND The 'Section B' tasks have been completed and produce correct results.

	AND The code largely conforms to PEP8 standards; however, some obscure errors may exist. This is acknowledged in the supporting documentation. AND Functions have been proactively used and are logical. Some of the functions are organised into the provided modules. AND The code is mostly written to a good standard. The marker may encounter the odd mistake, but these are uncommon. The program does not crash through general usage/navigation. AND General consideration is shown. Outputs are mostly provided to the user in an easy-to-read manner. The system is generally user friendly and is forgiving of mistakes. For example, the menu system should accept both upper and lowercase letters. Users are informed of any wrong menu choices and can easily recover from this. AND Student has approached the project in a fairly professional manner. Git commits indicate a good approach and declarations in the supporting documentation are honest, with perhaps the odd mistake.
B3, B2, B1	The C grade criteria has been met. AND The 'Section C' tasks have been completed and produce correct results. AND The code fully conforms to PEP8 standards. AND Functions are widely used and proactively organised into logical modules. Parameters and return values are used logically and often.

	AND The code is written to a good standard. The program does not crash easily. AND A meticulous approach has clearly been taken. All requirements up to this point appear to be met. The program is very easy to use throughout and all outputs are nicely formatted and simple to read/understand. Charts are fully labelled and nicely presented. AND The students approach to the project has been professional. Git commits are frequent and accompanied by appropriate messages. Declarations made in in the supporting documentation are honest and accurate. Code is easy to read with appropriate naming throughout.
A4, A3, A2, A1	The B grade criteria has been met. AND The 'Section D' tasks have been completed and produce correct results. AND The code is written to an excellent standard that logically breaks down code into reusable functions/classes. Functions are organised in to the most logical of the provided modules. Parameter and return value usage is excellent! The highest marks will only have print and input statements in tui.py. Any other modules will not be responsible for outputting to the screen or getting user input. AND A very professional approach has been taken. The use of git is excellent, and the documentation is meticulous – documenting has been upfront, honest, and very accurate.

Learning Outcomes

This assessment will enable you to demonstrate in full or in part your fulfilment of the following learning outcomes identified in the Module Descriptor:

Living CV

As part of the University's Work Ready, Future Ready strategy, you will be expected to build a professional, Living CV as you successfully engage and pass each module of your degree.

The Living CV outputs evidenced on completion of this assessment are:
1. Solve problems using programming techniques
2. Develop python applications to PEP8 standards

Please add these to your CV via the Living CV builder platform on Solent Futures Online Solent Futures Online

Important Information

Solent University Academic Regulations 2023-24

Late Submissions

You are reminded that:

i. If this assessment is submitted late i.e. within 7 calendar days of the submission deadline, the mark will be capped at 40% if a pass mark is achieved;
ii. If this assessment is submitted later than 7 calendar days after the submission deadline, the work will be regarded as a non-submission and will be awarded a zero;
iii. If this assessment is being submitted as a referred piece of work, then it must be submitted by the deadline date; any Refer assessment submitted late will be regarded as a non-submission and will be awarded a zero.

Assessment regulations

Extenuating Circumstances

The University's Extenuating Circumstances (EC) procedure is in place if there are genuine short term exceptional circumstances that may prevent you submitting an assessment. You are able to self-certify for up to two assessment dates in any semester without supporting evidence for an extension of up to seven calendar days for coursework or to defer an exam to the resit period.

Alternatively, if you are not 'fit to study' (or you have used up your two self-certification opportunities), you can request:

- an extension to the submission deadline of 7 calendar days, or
- a request to submit the assessment at the next opportunity, i.e. the resit period (as a Defer without capping of the grade).

In both instances you must submit an EC application with relevant evidence. If accepted under the university regulations there will be no academic penalty for late submission or non-submission dependent on what is requested. You are reminded that EC covers only short-term issues (20 working days) and that if you experience longer term matters that impact on your learning then you must contact the Student Hub for advice.

Please find a link to the EC policy below:

Extenuating Circumstances

Academic Misconduct
Any submission must be your own work and, where facts or ideas have been used from other sources, these sources must be appropriately referenced. The University's Academic Regulations includes the definitions of all practices that will be deemed to constitute academic misconduct. You should check this link before submitting your work.

Procedures relating to student academic misconduct are given below:

Academic Misconduct

Ethics Policy
The work being carried out must be in compliance with the university Ethics Policy. Where there is an ethical issue, as specified within the Ethics Policy, then you will need an ethics release or ethics approval prior to the start of the project.

The Ethics Policy is contained within Section 2S of the Academic Handbook:

Ethics Policy

Grade marking
The University uses an alpha numeric grade scale for the marking of assessments. Unless you have been specifically informed otherwise your marked assignment will be awarded a letter/number grade. More detailed information on grade marking and the grade scale can be found on the portal and in the Student Handbook.

Grade Marking Scale

Guidance for online submission through Solent Online Learning (SOL)

Online Submission