

**MODULAR PROGRAMME**

**COURSEWORK ASSESSMENT SPECIFICATION**

**Module Details**

|  |  |  |
| --- | --- | --- |
| **Module Code** UFCFVQ-15-M | **Run** 21JAN/1 | **Module Title**  Programming for Data Science |
| **Module Leader**  Dave Wyatt | **Module Coordinator** | **Module Tutors** Dave Wyatt, Jan Van Lent, Mahmoud Elbattah and Ahsan Kazmi |
| **Component and Element Number**  A | | **Weighting: (% of the Module's assessment)**  **100%** |
| **Element Description** PRACTICAL COURSEWORK | | **Total Assignment time**  20 hours plus 4 hours in-class time |

**Dates**

|  |  |
| --- | --- |
| **Date Issued to Students**  **28th March 2022** | **Date to be Returned to Students**  **13th June 2022** |
| **Submission Place**  **GitLab** | **Submission Date**  **12th May 2022** |
| **Submission Time** **2.00 pm** |

**Deliverables**

|  |
| --- |
| This assessed work has three deliverables:   * **Git Repository Link**, to be submitted via <https://forms.office.com/r/XVpbG0Up3A> * **Two Programming Tasks**, to be completed as instructed in the assignment specification and submitted via GitLab |

**Module Leader Signature**

|  |
| --- |
| DAVE WYATT |

Programming for Data Science -   
Assignment Specification

Portfolio of Programming Exercises and Development Process Report

# Information

This single coursework assignment will run during the Spring Term, and involves three separate components:

|  |  |
| --- | --- |
| Component | Assignment % |
| Git Repository Link | 5 |
| Programming Task 1 | 55 |
| Programming Task 2 | 40 |
| Total | **100** |

Clear instructions and coding requirements are given below. This document also contains an Appendix section with additional illustrations of what is required for Programming Task 1. Please read this document carefully and ensure that you have included all requested elements when submitting your work.

PLEASE NOTE: All submissions **MUST** be your own work.

* Students are **NOT** permitted to collude or submit anyone else’s work as their own (including any partial solutions found on the web).
* An analysis of submissions will be made across the cohort to identify any evidence of collusion and/or plagiarism during the marking phase.
* The following link provides more information about assessment offences and any associated penalties: <https://www.uwe.ac.uk/study/academic-information/assessments/assessment-offences>.

# A. Git Repository Link

* A GitLab repository should be created using the [gitlab.uwe.ac.uk](https://gitlab.uwe.ac.uk/) system according to the naming convention detailed in A.1 below
* The repository should be used to submit all coursework development work.
* The initial repository commit should be an appropriately edited README.md file (including a student ID, student name and MSc programme information)
* NOTE: any commits to this repository occurring after the deadline will be treated as LATE.

## A.1. Requirements

* The repository should use the following naming convention: ***00973398***\_***wyatt***\_fvq\_spring\_2022, where the first 8-digit number is your student ID (***00973398*** in this example), and the second part is your surname (***wyatt*** in this example). Please remove all spaces and hyphens (-) from your surname. The **fvq\_spring\_2022** refers to the module code (UFC**FVQ**-15-M), and module run information (Spring 2022). This must be included in your repository name to ensure we are able to identify your work as belonging to the Programming for Data Science module Spring 2022. Your repository MUST be created using your own current GitLab account. Any other GitLab or GitHub account will NOT be accepted as a valid submission.

## A.2. Deliverables

* The repository URL is the main deliverable for this part of the coursework, e.g., [***https://gitlab.uwe.ac.uk/di2-wyatt/00973398\_wyatt\_fvq\_spring\_2022***](https://gitlab.uwe.ac.uk/di2-wyatt/00973398_wyatt_fvq_spring_2022)***.***

## A.3. Submission

* The Repository URL should be submitted at the following link: <https://forms.office.com/r/XVpbG0Up3A>

## A.4. Grading Criteria

* Marks are allocated as follows:
  + up to 5 marks will be awarded to any student who has edited the README.md file and committed it with an **appropriate commit message**

# B. Programming Task 1

* This programming task focuses on using Python to calculate a set of standard statistical measures for a given dataset using built-in functions and data structures ONLY.
  + For Task 1, you **MUST NOT** import any Python library functions. This means you cannot use Python modules such as *math* or libraries such as Pandas, NumPy or SciPy.
* To print a set of summary statistics for a given data file, it would be very easy to use the ***describe()*** function provided in the Pandas library. However, this programming task is designed to assess your coding abilities and by preventing you from using this function (or any other statistical summary function) you are forced to gain a deeper understanding of how to complete that task. To do this, you will need to develop your own algorithm.
* There is a single data file available for use in this programming task. The file contains a record of US police criminal incidents for the year 2015.
  + The data file is called ***task1.csv***. This CSV file includes a header row with multiple named data values. This file is available in the Assignments section on Blackboard
* Students are expected to follow appropriate coding standards such as code commenting, consistent identifier naming, code readability, and appropriate use of data structures.
* You are expected to identify the strengths/weaknesses of your approach. For this programming task, you are expected to write a reflective report which focuses on the process taken to develop a solution to the task. The report should:
  + include an explanation of how you approached the task
  + include any pseudo code or other algorithmic aid used to help complete the task
  + identify any strengths/weaknesses of the approach used
  + suggest alternative approaches that could have been taken instead of the one you used

## B.1. Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Requirement | Description | Marks Available |
| FR1 | Develop a function to find the arithmetic mean | The function should take a Python List as a parameter and return its arithmetic mean. You should use the following list to test your function: 29, 17, 28, 6, 14, 7, 4, 27, 21, 15, 10, 16, 24, 26, 3, 11, 13, 8, 23, 9, 0, 22, 12, 2, 18, 19, 5, 1, 20, 25. | **3** |
| FR2 | Develop a function to find the standard deviation[[1]](#footnote-1) | The function should take a Python List as a parameter and return its standard deviation. You should use the same list as in FR1 to test your function. | **4** |
| FR3 | Develop a function to find the median | The function should take a Python List as a parameter and return the median value of the data in the List. You should use the same list as in FR1 to test your function. | **3** |
| FR4 | Develop a function to find the skewness | The function should take a Python List as a parameter and return skewness of the data in the List. You should use the same list as in FR1 to test your function. | **4** |
| FR5 | Develop a function to read a single specified column of data from a CSV file | The function should accept two parameters: the data file name and a column number. The column number specifies which of the columns to read. It can range between 0 and n-1 (where n is the number of columns in the file). The function should return two values: a List containing all the specified column’s data values and the column name. You should use the ***task1.csv*** data file to test your function but your function should also work for other CSV files. An illustration of this is given in Appendix 1. | **7** |
| FR6 | Develop a function to read CSV data from a file into memory | The ***task1.csv*** data file contains multiple columns of data values. This function should make use of the function developed in FR5 to read all the columns of data from the data file and add them to a Dictionary data structure. The Dictionary should contain one entry for each column in the CSV data file. An illustration of this is given in Appendix 2. | **7** |
| FR7 | Develop a function to generate a set of statistics for a given data file | This function should make use of all the functions developed in FR1-FR4 to generate a set of statistics for each column in the data read into memory in FR6. The function should return a dictionary containing all calculated statistics for each column plus an extra entry showing the name of each statistic. An illustration of this is given in Appendix 3. | **7** |
| FR8 | Develop a function to print a custom table | This function should be able to output all the statistics generated in FR7. The function should also offer customisable options including which border character to use and which columns to output. High marks will be given for good use of padding in the table cells to improve readability. An illustration of this is given in Appendix 4. | **7** |

## B.2. Deliverables

* There is single deliverable for this task:
  + A Jupyter Notebook file (in .ipynb format) containing a complete solution to Programming Task 1 (based on the template provided[[2]](#footnote-2)).
  + The Jupyter Notebook should also include a Development Process Report written using Markdown reflecting on the process taken to develop a solution to this task
    - The report should not exceed 500 words.

## B.3. Submission

* The Jupyter Notebook file (incl. Development Process report) should be submitted electronically by including it in the student’s coursework GitLab repository.

## B.4. Grading Criteria

* Marks are allocated as follows:
  + up to 42 marks for the Python code solution
    - Marks will be awarded for each completed requirement **according to the level of completion** (commonly **0%**, **25%, 50%**, **75%** or **100%**).
  + up to 5 marks for adherence to good coding standards.
  + up to 8 marks for the Development Process Report
    - Marks will be awarded for appropriate use of technical language, critical reflection on development process and quality of engagement with the reflective process

# C. Programming Task 2

* This programming task focuses on using NumPy/SciPy, Pandas, and Matplotlib/Seaborn to combine, clean and analyse two datasets related to bike sharing in London between 2015 and 2017.
* Two data files have been provided for this task.
  + The ***task2a.csv*** data file contains the number of bike shares per hour between January 2015 and January 2017.
  + The ***task2b.csv*** data file contains the temperature, “feels like” temperature, humidity, wind speed for every hour between 2015 and 2017.
  + The files are available in the Assignment section on Blackboard
* Students are expected to follow appropriate coding standards such as code commenting, consistent identifier naming, code readability, and appropriate use of data structures.
* You are expected to identify the strengths/weaknesses of your approach. For this programming task, you are expected to write a reflective report which focuses on the process taken to develop a solution to the task. The report should:
  + include an explanation of how you approached the task
  + include any pseudo code or other algorithmic aid used to help complete the task
  + identify any strengths/weaknesses of the approach used
  + suggest alternative approaches that could have been taken instead of the one you used

## C.1. Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Requirement | Description | Marks Available |
| FR9 | Read CSV data from two files and merge it into a single Data Frame | For this task you should use the ***task2a.csv*** and ***task2b.csv*** data files | **2** |
| FR10 | Clean the merged data | Remove all rows from the merged data whose bikes shares value is less than 20. Remove the following unnecessary column: ***season\_code*** | **2** |
| FR11 | Investigate the distribution of bike shares by time of the day | Use an appropriate visualisation tool (such as Matplotlib or Seaborn) to investigate the distribution of bike shares by time of the day. You must include an explanation of your findings to achieve good marks for this requirement. | **6** |
| FR12 | Compare the distribution of bike shares by time of the week | Use an appropriate visualisation tool (such as Matplotlib or Seaborn) to compare the distribution of bike shares by time of the week. Emphasis should be given to the differences between weekends and weekdays. You must include an explanation of your findings to achieve good marks for this requirement. | **6** |
| FR13 | Investigate the effects of “feels like” temperature on the number of bike shares | Use an appropriate visualisation tool (such as Matplotlib or Seaborn) to investigate the correlation between the “feels like” temperature and the number of bike shares. You must include an explanation of your findings to achieve good marks for this requirement. | **7** |
| FR14 | Test the hypothesis that “feels like” temperature influences the number of bike shares | Using an appropriate Python library, test if there is any significant correlation between the “feels like” temperature and the number of bike shares. You must include an explanation of your findings to achieve good marks for this requirement. | **5** |

## C.2. Deliverables

* There is single deliverable for this task:
  + A Jupyter Notebook file (in .ipynb format) containing a complete solution to Programming Task 2 (based on the template provided[[3]](#footnote-3)).
  + The Jupyter Notebook should also include a Development Process Report written using Markdown reflecting on the process taken to develop a solution to this task
    - The report should not exceed 500 words.

## C.3. Submission

* The Jupyter Notebook file (incl. Development Process report) should be submitted electronically via the student’s coursework GitLab repository.

## C.4. Grading Criteria

* Marks are allocated as follows:
  + up to 28 marks for the Python code solution
    - Marks will be awarded for each completed requirement **according to the level of completion** (commonly **0%**, **25%, 50%**, **75%** or **100%**).
  + up to 4 marks for adherence to good coding standards.
  + up to 8 marks for the Development Process Report
    - Marks will be awarded for appropriate use of technical language, critical reflection on development process and quality of engagement with the reflective process

# Appendix 1 – Example Column Extraction

For the following illustration, you should assume that the column number parameter is equal to ***1*** for the data file. There are 9 columns in this file and so column number can range between 0 and 8. For this data, the function created in FR5 would return two values: ***“Glucose”*** and ***[148,85,183,89,137,116,78,115,197,125,110,168,139]***

# Table Description automatically generated with medium confidence

# Appendix 2 – In-Memory Data Structure

Using the file illustrated in Appendix 1, the Dictionary produced in FR6 should look something like the illustration below. However, you must ensure that your function can work for any CSV file with a similar structure (e.g., one with 5 columns and 100 rows or 20 columns and 1000 rows).

**{**

**"Pregnancies" : [6,1,8,1,0,5,3,10,2,8,4,10,10],**

**"Glucose" : [148,85,183,89,137,116,78,115,197,125,110,168,139],**

**"BloodPressure" : [72,66,64,66,40,74,50,0,70,96,92,74,80],**

**"SkinThickness" : [35,29,0,23,35,0,32,0,45,0,0,0,0],**

**"Insulin" : [0,0,0,94,168,0,88,0,543,0,0,0,0],**

**"BMI" : [33.6,26.6,23.3,28.1,43.1,25.6,31,35.3,30.5,0,37.6,38,27.1],**

**"DiabetesPedigreeFunction" : [0.627,0.351,0.672,0.167,2.288,0.201, 0.248,0.134,0.158,0.232,0.191,0.537,1.441],**

**"Age" : [50,31,32,21,33,30,26,29,53,54,30,34,57],**

**"Outcome" : [1,0,1,0,1,0,1,0,1,1,0,1,0]**

**}**

# Appendix 3 – Statistical data based on In-Memory Data Structure

Using the in-memory data structure illustrated in Appendix 2, the Dictionary produced in FR7 should look something like the illustration below. Remember that different CSV data files will result in different data being stored in the Dictionary. The data file you have been provided with does not include any of the data shown below. Don’t be tempted to simply copy the result below into your Jupyter Notebook.

**{**

**"Stats" : ["Mean","Stdev","Median","Skewness"],**

**"Pregnancies" : [5.23,3.70,5, 0.05],**

**"Glucose" : [130,37.03,125,0.38],**

**"BloodPressure" : [64.92,24.54,70,-1.58],**

**"SkinThickness" : [15.31,17.84,0,0.42],**

**"Insulin" : [68.69,152.29,0,2.92],**

**"BMI" : [29.22,10.47,30.5,-1.80],**

**"DiabetesPedigreeFunction": [0.56,0.63,0.25,2.15],**

**"Age" : [36.92,12.04,32,0.71],**

**"Outcome" : [0.54,0.52,1,-0.18]**

**}**

# Appendix 4 – Output table for Statistics

Using the output from the function produced in FR7, the following table outputs a subset of the available columns (as defined by the function parameter) using the border character **\*** and padding within the cells to ensure the table is readable:

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\* Glucose \* BloodPressure \* BMI \* Age \***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\* Mean \* 130 \* 64.92 \* 29.22 \* 36.92 \***

**\* Stdev \* 37.03 \* 24.54 \* 10.47 \* 12.04 \***

**\* Median \* 125 \* 70 \* 30.5 \* 32 \***

**\* Skewness \* 0.38 \* -1.58 \* -1.80 \* 0.71 \***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

1. Note that there are different types of standard deviation. Clearly indicate which standard deviation has been implemented in the code comments [↑](#footnote-ref-1)
2. There is a Jupyter Notebook template available in the Assignment folder on Blackboard - ***UFCFVQ-15-M \_Programming\_Task\_1\_Template.ipynb*** [↑](#footnote-ref-2)
3. There is a Jupyter Notebook template available in the Assignment folder on Blackboard - ***UFCFVQ-15-M\_Programming\_Task\_2\_Template.ipynb*** [↑](#footnote-ref-3)