

**MODULAR PROGRAMME**

**COURSEWORK ASSESSMENT SPECIFICATION**

**Module Details**

|  |  |  |
| --- | --- | --- |
| **Module Code** UFCFVQ-15-M | **Run** 22JAN/1 | **Module Title**  Programming for Data Science |
| **Module Leader**  Dave Wyatt | **Module Coordinator** | **Module Tutors** Dave Wyatt; Jan Van Lent; Elisa Covato; Ahsan Kazmi |
| **Component and Element Number**  A | | **Weighting: (% of the Module's assessment)**  **100%** |
| **Element Description** PRACTICAL COURSEWORK | | **Total Assignment time** 20 hours + 4 hours in-class time |

**Dates**

|  |  |
| --- | --- |
| **Date Issued to Students**  28th March 2023 | **Date to be Returned to Students**  7th June 2023 |
| **Submission Place**  **GitLab** | **Submission Date**  9th May 2023 |
| **Submission Time** **2.00 pm** |

**Deliverables**

|  |
| --- |
| This assessed work has three deliverables:   * **Git Repository Link**, to be submitted via the link <https://forms.office.com/e/5D1pYSkKc0> * **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 involves three separate components:

|  |  |
| --- | --- |
| Component | Assignment % |
| Git Repository Link | 5 |
| Python Programming | 55 |
| Python Programming with Libraries | 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 Task B. Please read this document carefully and ensure 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>.

# Git Repository Link (Task A)

* 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 commit all development work including .csv data files used for each task.
* 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 that occur after the coursework deadline will be treated as LATE.

## A.1. Requirements

* The repository should use the following naming convention: ***00973398***\_***di2wyatt***\_fvq\_spring\_2023, where the first 8-digit number is your student ID (***00973398*** in this example), and the second part is your username (***di2wyatt*** in this example). Please remove the hyphen (-) from your username. The **fvq\_spring\_2023** refers to the module code (UFC**FVQ**-15-M), and module run information (Spring 2023). 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 2023 run.
* You MUST set your repository’s initial visibility to ***Private***. You will be instructed when to change visibility back to ***Public*** at the appropriate time via a module announcement.

## 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\_di2wyatt\_fvq\_spring\_2023***](https://gitlab.uwe.ac.uk/di2-wyatt/00973398_di2wyatt_fvq_spring_2023)

## A.3. Submission

* Your Repository URL should be submitted at the following link: <https://forms.office.com/e/5D1pYSkKc0>.
* You should submit this URL as soon as you have completed this task. Please DO NOT wait until you have completed the remainder of the coursework before you do this. Marks will be deducted for late submission.

## 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.
* Marks will be deducted for incorrect naming convention and/or late submission of the link.

# B. Python Programming (Task B)

* This task focuses on using Python to calculate a set of Spearman’s Rank Correlation Coefficients 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 or NumPy.
* To print the Spearman’s Rank Correlation Coefficient for a given pair of Python Lists, it would be very easy to use the ***spearmanr()*** function provided in the SciPy library. However, this programming task is designed to assess your coding abilities and by preventing you from using this 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. Try typing “calculate Spearman’s Rank Correlation Coefficient by hand” into your favourite search engine.
* There is a single data file available for use in this programming task. The file contains a record of US police killings 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. Please reflect on your experiences rather than simply describing what you did. The report should:
  + include an explanation of how you approached the task.
  + identify any strengths/weaknesses of the approach used.
  + consider how the approach used could be improved.
  + 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 geometric mean | The function should take a Python List as a parameter and return its geometric mean. Use the following list to test your function: 64, 9, 90, 28, 46, 95, 34, 28, 86, 62, 14, 77, 99, 80, 99, 56, 79, 37, 74, 6, 67, 32, 5, 94, 53, 62, 19, 44, 16, 74, 92, 60, 74, 80, 10, 43, 51, 41, 91, 41, 27, 40, 48, 27, 13, 41, 13, 28, 17, 64. | **2** |
| FR2 | Develop a function to read a single specified column of data from a CSV file | The function should accept 3 parameters: the data file name, a column number and conversion indicator. 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). The conversion indicator can take two values: *True* or *False*. An indication of *True* means that the column values should be converted from strings to numbers. The function should return 2 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. | **6** |
| FR3 | 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 accept two parameters: the data file name and a list of conversion indicators (1 for each column in the file). It should make use of the function developed in FR2 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 example illustration is given in Appendix 2. | **6** |
| FR4 | Develop a function to calculate the Spearman’s Rank Correlation Coefficient for 2 lists of data | This function should calculate the Spearman’s Rank Correlation Coefficient for 2 lists of data. You should make use of the function developed in FR1. The function should take 2 lists of data (of equal length) as parameters. The function should return the calculated coefficient value. | **10** |
| FR5 | Develop a function to generate a set of Spearman’s Rank Correlation Coefficients for a given data file | This function should make use the function developed in FR4 to generate a Spearman’s Rank Correlation Coefficient for every pair of columns in the data read into memory in FR3. The function should return a list of tuples, each tuple containing the 2 column names and associated correlation coefficient value. An illustration of this is given in Appendix 3. | **10** |
| FR6 | Develop a function to print a custom table | This function should output the Spearman’s Rank Correlation Coefficient for a subset of the column pairs generated in FR5. The function should take 3 parameters: list of correlation coefficient tuples, border character to use and which columns to include. High marks will be given for good use of padding in table cells to improve readability. An example illustration is given in Appendix 4. | **8** |

## B.2. Deliverables

* There is single deliverable for this task:
  + A Jupyter Notebook file (in .ipynb format) containing a complete solution to Task B.
    - You must use the template provided [[1]](#footnote-1).
  + 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 are awarded for each completed requirement **according to the level of completion**.
  + 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. Python Programming with Libraries (Task C)

* This programming task focuses on using NumPy/SciPy, Pandas, and Matplotlib/Seaborn to combine, clean and analyse two datasets related to student performance.
* Two data files have been provided for this task. These files provide some real data from the Open University.
  + The ***task2a.csv*** data file contains background information about 26746 students including gender, age, disability status and score.
  + The ***task2b.csv*** data file contains information about the number of click events made by 26074 students using the University’s Virtual Learning Environment (VLE) system.
  + The files are 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 must write a reflective report which focuses on the process taken to develop a solution to the task. Please reflect on your experiences rather than simply describing what you did. The report should:
  + include an explanation of how you approached the task.
  + identify any strengths/weaknesses of the approach used.
  + consider how the approach used could be improved.
  + suggest alternative approaches that could have been taken instead of the one you used.

## C.1. Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Requirement | Description | Marks Available |
| FR7 | 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** |
| FR8 | Clean the merged data | Remove all rows from the merged data that contain a missing value in any column. Remove the following unnecessary column: ***region***, ***final\_result*** and ***highest\_education*** | **3** |
| FR9 | Filter out unnecessary rows | Remove all rows where ***click\_event*** is greater than ***20000*** and any duplicate rows based on ***id\_student*** | **2** |
| FR10 | Investigate the effects of engagement on attainment | Use an appropriate visualisation tool (such as Matplotlib or Seaborn) to investigate if there is any relation between the engagement (click events) and the level of attainment (score). You must include an explanation of your findings to achieve good marks for this requirement. | **6** |
| FR11 | Test the hypothesis that engagement has some effect on levels of attainment | Using an appropriate Python library, test if there is any statistically significant relation between engagement and attainment. You must include an explanation of your findings to achieve good marks for this requirement. | **4** |
| FR12 | Investigate the effects of gender on levels of attainment | Use an appropriate visualisation tool (such as Matplotlib or Seaborn) to investigate if there is any effect on levels of attainment due to gender. You must include an explanation of your findings to achieve good marks for this requirement. | **6** |
| FR13 | Test if there is any difference between the attainment of male and female students | Using an appropriate Python library, test if there is any statistically significant difference between male and female students. You must include an explanation of your findings to achieve good marks for this requirement. | **4** |

## C.2. Deliverables

* There is single deliverable for this task:
  + A Jupyter Notebook file (in .ipynb format) containing a complete solution to Task C
    - You must use 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.

## C.3. Submission

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

## C.4. Grading Criteria

* Marks are allocated as follows:
  + up to 27 marks for the Python code solution
    - Marks are awarded for each completed requirement **according to the level of completion**.
  + 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.

# 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 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 FR3 should look something like the illustration below. However, you must ensure that your function can work for any CSV file with a similar structure (such as a file with 5 columns and 100 rows or with 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 List of Tuples produced in FR5 should look something like the illustration below. The full data output is too large to include here and so only some of the data has been included to help illustrate what is required. Remember that different CSV data files will result in different data being stored. 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.

**[**

**("Pregnancies", "Glucose", 0.3712),**

**("Pregnancies", "BloodPressure", 0.3195),**

**("Pregnancies", "SkinThickness", -0.7058),**

**("Pregnancies", "Insulin", -0.697),**

**("Pregnancies", "BMI", -0.1080),**

**("Pregnancies", "DiabetesPedigreeFunction", 0.0609),**

**("Pregnancies", "Age", 0.3412),**

**("Pregnancies", "Outcome", 0.0),**

**("Glucose", "Pregnancies", 0.3712),**

**("Glucose", "BloodPressure", 0.1818),**

**("Glucose", "SkinThickness", 0.0419),**

**("Glucose", "Insulin", 0.0067),**

**("Glucose", "BMI", 0.011),**

**("Glucose", "DiabetesPedigreeFunction", 0.3132),**

**("Glucose", "Age", 0.7208),**

**("Glucose", "Outcome", 0.5361),**

**+++++++ More data would be included here ++++++++**

**("Outcome", "Pregnancies", 0),**

**("Outcome", "Glucose", 0.5361),**

**("Outcome", "BloodPressure", -0.1034),**

**("Outcome", "SkinThickness", 0.4041),**

**("Outcome", "Insulin", 0.3022),**

**("Outcome", "BMI", 0.1237),**

**("Outcome", "DiabetesPedigreeFunction", 0.3712),**

**("Outcome", "Age", 0.413)**

**]**

# Appendix 4 – Output table for Statistics

Using the output from the function produced in FR5, 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 $**

**$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$**

**$ Glucose $ - $ 0.1818 $ 0.0110 $ 0.7208 $**

**$ BloodPressure $ 0.1818 $ - $ -0.2782 $ 0.5117 $**

**$ BMI $ 0.0110 $ -0.2782 $ - $ -0.1541 $**

**$ Age $ 0.7208 $ 0.5117 $ -0.1541 $ - $**

**$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$**

1. There is a Jupyter Notebook template available in the Assignment folder on Blackboard - ***UFCFVQ-15-M\_Python\_Programming\_Template.ipynb*** [↑](#footnote-ref-1)
2. There is a Jupyter Notebook template available in the Assignment folder on Blackboard - ***UFCFVQ-15-M\_Python\_Programming\_With\_Libraries\_Template.ipynb*** [↑](#footnote-ref-2)