Solent University
Coursework Assessment Brief

Assessment Details

Module Title:	Programming for Problem Solving
Module Code:	COM731
Module Leader:	Jarutas Andritsch
Level:	7
Assessment Title:	The Software Project
Assessment Number:	AE2
Assessment Type:	Software Artefact and documentation
Restrictions on Time/Word Count:	Documentation 2000 words (+/-10%) (excluding Table of contents, Table of Figures, Index of Tables)
Consequence of not meeting time/word count limit:	It is essential that assignments keep within the word count limit stated above. Any work beyond the maximum time/word length permitted will be disregarded and not accounted for in the final grade.
Individual/Group:	Individual
Assessment Weighting:	60%
Issue Date:	23rd September 2024
Hand In Date:	6th January 2025
Planned Feedback Date:	Within 4 working weeks after the hand-in
Mode of Submission:	Online via Solent Online Learning (SOL): - 1 copy of a zip file containing the source codes: main program as Jupyter notebook (.ipynb), own-defined module as .py) and dataset for your software artefact - 1 copy of a software document in PDF format. This should not be included in the zip file but instead submitted as a separate file. Note: The software code and documentation are considered one unit. Make sure to submit both files together. If document file is missing, it may result in fail or a capped pass mark if you meet the passing criteria. Submitting only the document will result in fail. Only FINAL submissions will be accepted. DRAFT submissions will not be considered an attempt and will not be marked.
Anonymous Marking	This assessment **is exempt from anonymous marking.**

Assessment Task

You are required to develop a software application that addresses the problem scenario using Python and the tools specified in this assessment brief. You must document software implementation which provide a technical, concise and critical discussion of your solution. You should discuss the technical details of how your software solution was implemented, providing appropriate justifications.

Introduction

Lung cancer ranks among the top causes of deaths globally, highlighting the need for research, prevention, and improved treatments. Analysing lung cancer data helps identify risks, enhance diagnoses, and develop new therapies.

In this assessment, you will process, manage, and analyse health data. You'll be working on lung cancer data in this task. The dataset, lung_cancer_data.csv, includes 38 columns, with each row representing a complete patient record without any missing data. The dataset contains synthetic values to protect the privacy and sensitivity of real patient information, while the collected data (columns) closely resemble real-world clinical scenarios. It's recommended to familiarise yourself with the dataset's contents before continuing with the assessment.

Requirements

The requirements for the system are as follows:

a) The system enables users to get data from a CSV file using basic Python features (like control structures and file processing) and the **csv module**. Users need to input the file path or filename for file access. In task A, the software loads data from the CSV file into memory using the csv module's **reader() function** to load the data into memory and store it in a list. Do not convert the list to dataframe format for this task. The tasks to be performed from the loaded data are outlined below.
   - a1. Retrieve demographic information: age, gender, smoking history, and ethnicity based on the patient ID.
   - a2. Retrieve medical history details including family history of lung cancer, comorbidities of diabetes disease, comorbidities of kidney disease, and the haemoglobin level associated with a certain ethnicity.
   - a3. Retrieve treatment details including age, tumor size, tumor location, and tumor stage of patients who have survived more than 100 months on a certain treatment.
   - a4. Retrieve information from your chosen columns and apply a specific condition that relates to patient. Please select at least three columns and one condition that differs from previous requirements.

b) The system enables users to analyse data through the use of the **pandas module**. In task B, the system loads data from a CSV file into memory using the **read_csv() function**, utilising the file path or filename obtained from task A without prompting the user again. With the loaded dataframe, proceed to perform the specified tasks:
   - b1. Identify the top 3 treatments for a certain ethnicity where patients have survived more than 100 months.
   - b2. Analyse the average white blood cell counts for certain treatments based on a certain ethnicity.
   - b3. Analyse the average number of smoking packs for patients in each treatment group, with a blood pressure (pulse) over 90 and a tumor size smaller than 15.0 mm, based by tumor location.
   - b4. Analyse the data to derive meaningful insights based on your unique selection, distinct from the previous requirements.

c) The system enables users to visualise data using the **matplotlib module**. You're permitted to only use functions from the matplotlib library that were taught in class to create charts or graphs. Do not use any functions from the pandas module or any chart types not covered in class for this task. Use the dataframe from task B to complete the following tasks:
   - c1. Create a chart to illustrate the proportion of cancer treatments among a certain ethnicity as specified by the user.
   - c2. Create a chart to show the trend of average smoking packs consumption across different cancer stages for each ethnicity within a single chart.
   - c3. Create a chart that visually compares the average of all blood pressure types across different treatment types within a single chart.

c4. Create a visualisation of your selection to showcase information related to patients, treatment, or conditions that can reveal trends, behaviours, or patterns, ensuring it is distinct from previous requirements.

Software documentation

The software documentation provides a comprehensive technical discussion of the software implementation, supported by evidence related to legal, social, ethical, and professional aspects of software development. This documentation should cover the following key topics:

- Overview: the aim and objectives of the project and brief discussion of the dataset
- Self-reported requirement completion
- Project Implementation: Project Structure and self-created module/functions (technically explain how the module/function implemented)

Use of AI in this Assessment

Generative AI is permitted at Solent University under specific conditions and must continue to follow the university's rules around Academic Misconduct and the AI and Academic Integrity policy. In this assessment, you are allowed to use these tools solely for syntax guidance in programming. The use of AI tools for problem-solving and development process are not permitted.

Here's an example prompt that can be used for seeking syntax guidance. Any prompts with a similar approach suggesting a sample for a specific syntax or explanation are acceptable.
Prompts for syntax guidance:
"Could you provide an example of how to use a for loop in Python?"
"I'm trying to understand how to define a function with multiple parameters in Python. Can you explain?"
"What is the correct approach to format a multiline string in Python?"

Prompting to generate code or asking for solutions to specific problems is not permitted. Any prompt asking GenAI to write code, or anything similar in meaning, is not allowed. Here's an example prompt that can't use:
"Can you write a Python code implementing a task based on this requirement?"
"Could you generate a Python program that creates the visual chart described in this requirement?"
"Write a Python function to retrieve information from a file."

The application of technology should always involve human oversight and control. It is crucial for you to thoroughly review and edit the results, considering that AI may produce authoritative sounding yet incorrect or incomplete output.

In the report, you must disclose the use of generative AI and AI-assisted technologies in the programming and/or writing process. This disclosure should be placed at the end of the report, following the GitHub Repository Evidence section. Create a new section titled 'Declaration of Generative AI and AI-assisted technologies in the Programming and/or Writing Process.'

Include the following statement:

"During the implementation of this work, I, [STUDENT_NAME], utilised [NAME OF TOOL/SERVICE] to [SPECIFY WHAT CONTENT WAS/WERE GENERATED AND WHAT CHANGE YOU MADE].

After using this tool/service, I thoroughly reviewed and edited the content as necessary, taking full responsibility for the final content of the work."

Note: For the specification - Please ensure you specify each piece of content generated using a generative AI tool and indicate the specific alterations made to make it work for this assessment.

Disclosing the use of generative AI and AI-assisted technologies for syntax aids will not impact the grade. Your assessment evaluation will be conducted according to the criteria outlined in the assessment brief. It is considered good practice for ensuring transparency in the work. This practice is in place to prevent any potential misinterpretation, where the use of these tools might be misconstrued as claiming someone else's work as the student's own.

Failure to disclosure the utilisation of Generative AI and AI-assisted technologies tool will lead to a potential reduction in your overall grade. This reduction could limit it to a passing grade if the assessment meets the pass mark or result in failure if the quality does not meet the passing criteria.

AI and Academic Integrity Policy

Expectations

The assessment must be completed individually. You must not share, in part or whole, your assessment with another party other than the module tutor and for the purpose of submission to the university. You must ensure that the University's academic misconduct guidelines are followed in their entirety.

It is expected that you will develop a software application that meets the stated requirements. You have been provided with a CSV file that contains data. Your application will need to appropriately load the data contained in this file, process the loaded data, analyse the loaded data, and visualise suitable information from these loaded data. You should appropriately test your implemented functionality.

You are required to evidence your work throughout your assessment. You should create a suitable **private Git repository**. Please ensure you **regularly commit** your implementation to your repository **with clear and descriptive commit messages** as you work on your solution. As part of the project and demonstration for this assessment, you will be asked to document and show your project's repository. You should ensure that your submission complies with academic misconduct guidelines, is your own work and any external sources have been appropriately referenced. **Failure to provide a Git commit history or unsuitable commit history will result in a potential cap on your overall mark, limiting it to a passing grade.**
Note: If you have any special requirement or disability, please discuss this with your tutor before the submission date.

Environment

You are required to use the following tools:

- Jupyter Notebook as your development environment
- Python3.10 or above as the standard python library
- Additionally, the following libraries/modules/function may be imported and utilised (if needed):
    - os – to retrieve or check file paths
    - random – to generate random numbers
    - numpy – to perform efficient mathematical operations on numerical arrays
- The COM731 environment from the class session which installed pandas, matplotlib, Jinja2, and tabulate modules
- Git Tools and GitHub for version control

No other python libraries or modules should be used other than the specified.

Assessment criteria

Learning Outcomes	UPPER FIRST A1 – A2 Exceed expectations in many aspects		FIRST A3 – A4 Substantially exceeds expectations		UPPER SECOND B1 – B3 (High) Meet learning outcomes and exceeds expectations in several aspects			LOWER SECOND C1 - C3 (Good) Meet learning outcomes and sometimes exceeds expectations			THIRD D1 - D3 (Competent) Meet learning outcomes			FAIL F1 – F3 (Incomplete/Poor) Fails to meet learning outcomes		
SOLENT MARK	100	92	83	74	68	65	62	58	55	52	48	45	42	35	20	15
Design computer programs in a logical and structured way using appropriate techniques and principles	The text-based user interface is exceptionally well-structured as a standalone module, seamlessly integrated into the main program and other modules for displaying results. Usability is a central focus, with user interactions driven by meticulously designed structures and functions. It presents results flawlessly with clear, intuitive prompts, and offers insightful error messages when necessary, ensuring an unparalleled smooth and user-friendly experience.		The text-based user interface is expertly developed as a standalone module, effectively integrated into the main program and other modules. Usability is a key priority, featuring well-organised structures and functions for user interactions. It consistently presents results with clear prompts and provides helpful error messages, contributing to a highly smooth and user-friendly experience.		The text-based user interface is competently crafted as a standalone module, successfully integrated into the main program and other modules. Usability considerations are evident, with well-organised structures and functions for user interactions. It generally presents results with clarity and offers helpful error messages, resulting in a mostly smooth experience.			The text-based user interface is competently developed as a standalone module, although integration into the main program and other modules could be improved. Usability aspects are somewhat addressed, with modestly organised structures and functions for user interactions. It typically presents results and offers basic error messages, resulting in an acceptable but not fully polished user experience.			The system utilises a basic user interaction message, usability considerations are minimal keeping all the code within a single file. It presents results plainly in a straightforward layout or format.			Little or no user interface, little or no user interaction message has been implemented. All attempt or majority coding use static data or initialise data to meet specific requirements.		Usability aspects are largely overlooked, resulting in poorly structured structures and functions for user interactions. Results may lack clarity, and error messages are often unhelpful, leading to a frustrating and suboptimal user experience.

Develop computer programs aligned to appropriate programming standards and code conventions	The software artefacts unequivocally demonstrate flawless realisation of all specific requirements, achieving a perfect 100% correctness and containing no errors. The code is exceptionally well-modularised, expertly combining user-defined and built-in functions that operate precisely as intended and adhere to established best practices. Every self-created requirement is advanced, intricate, and has many aspects. Each requirement is highly complex, showing a sophisticated understanding and execution. Clear and meticulously	The software artefacts exhibit an excellent level of proficiency, confidently addressing all specific requirements (90-100%) with excellent correctness, only minor errors. The code demonstrates excellent modularity, effectively incorporating both user-defined and built-in functions, aligned with best practices. Every self-created requirement is consistently advanced and complex. They are highly intricate. This reflects a sophisticated understanding with a detailed approach. Comments are consistently clear	The software artefacts showcase a high level of competence, effectively implementing a significant portion of the specific requirements (80-90%) with high correctness, only occasional errors. The code is highly modular, effectively utilising both user-defined and built-in functions in adherence to best practices. Most self-created requirements are complex. They are advanced showing a solid understanding and practical focus. Comments are consistently clear and well-structured, enhancing overall readability. Coding standards are diligently followed,	The software artefacts display proficiency in addressing specific requirements (60-80%), maintaining correctness, minor errors. The code exhibits modularity with user-defined functions, but they are not well integrated into the main program or are inefficiently implemented. Some self-created requirements are minimal complexity. Despite having advanced features, they are simplified in some ways. Comments are generally clear, contributing to code readability. Coding standards are consistently followed, including proper indentation and consistent variable/function naming.	The software artefacts competently execute a majority of the specific requirements (50-60%), with satisfactory correctness, occasional errors. Code modularity is evident, but the main program still processes in a sequential order, with a preference for user-defined functions. Self-created requirements follow basic principles, much like fixed requirements. Comments, while existent, may require improvement in terms of clarity and consistency. Coding standards are only occasionally followed, affecting the code	The software artefacts exhibit limited proficiency in implementing select specific requirements (30-50%), with noticeable correctness issues and errors or only the software documentation is submitted. Code modularity is limited, or no module implemented, predominantly relying on built-in functions or just plain python statement. All code is written in sequential order using static, predefined values. Self-created requirements seem simple or randomly chosen, possibly to display results	The software artefacts demonstrate limited implementation of requirements (less than 30%), with substantial correctness challenges and frequent errors. Code lacks modularity, primarily relying on built-in functions. little or no proof that self-created requirements involve intentional effort beyond copying fixed requirements and adjusting variables, showing a basic replication without much progress. Comments, if available, may lack clarity and consistency, making it

	structured comments are seamlessly integrated throughout the code, greatly enhancing overall readability. This clarity is further enhanced through steadfast adherence to coding rules and conventions, including impeccable indentation and consistent variable/function naming.	and well-structured, contributing to excellent code readability. Coding standards are consistently upheld, with impeccable indentation and uniform variable/function naming.	with good indentation and uniform variable/function naming.		readability to some extent.	resembling fixed requirements. They are basic in complexity. Comments, if present, may lack clarity and consistency or no comment provided, hindering code understanding. Coding standards are inconsistently followed, diminishing code readability.	difficult to understand the code. Coding standards are largely ignored, severely impacting code readability.
Utilise suitable tools to design, implement, test and evaluate solutions	Extensive error exception handling and validation have been used in the code.	Well error exception handling and validation have been used in the code.	There is a good evidence of error exception handling and validation.	There is some error handling or error exception, but input validation may be limited.	There is a few evidence of error handling or error exception, little or none input validation.	There is little or no evidence of error handling or error exception, no input validation.	No evidence of attempting required threshold.

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. I can solve real-world problems by getting and analysing large amounts of data.
2. I can confidently write Python code to obtain, manipulate, and analyse real-world dataset.
3. I am experienced in using environment tool such as Jupyter notebook to design, implement, test and evaluate solutions.
4. I can conduct written and verbal presentations to share insights to audiences of varying levels of technical sophistication.

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 2024-25

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