University of Bristol
SEMTM0028 2025/26
Software Development: Programming and Algorithms (SDPA)

Coursework (100%)

This unit assessment asks you to apply the skills and tools that you've learned throughout the unit, on a selection of different tasks. General guidelines are as follows:

- **Deadline**: See Blackboard
- **Rules**: This is an Individual based assessment. Do not share your code with anybody. You are welcome to discuss questions with other students, but don't share the answers. The experience of solving the problems in this project will prepare you for real problems in data science (and life).
- **Support**: If you're ever feeling overwhelmed or don't know how to make progress email the unit director for help.
- **Academic Integrity**: academic misconduct (for example, plagiarism, collusion, contract cheating) is unacceptable and will be dealt with in accordance with university policies. Please not that submitting artificial intelligence answers as your own is a form of contract cheating. More information is available here: https://www.bristol.ac.uk/students/support/academic-advice/academic-integrity/.
- **Late submission**: school policy penalises late submissions. Any submission after the deadline is subject to a penalty.
- **Advice**: develop your answers incrementally. To perform a complicated task, break it up into steps, perform each step on a different line, give a new name to each result, and check that each intermediate result is what you expect.

1 Coursework General Instructions

1.1 Documentation and commenting your code

Your code must be documented appropriately using **docstrings** and **comments**. Document as you go, not all at the end. One strategy is to write a brief comment about the 'one thing' that the function is supposed to do when you declare it, then refer to that comment while implementing it: this helps maintain focus when implementing the function and helps stop yourself from making it do more than the 'one' thing it is supposed to do. Each class and each method should have at least one docstring.

For this project you will need to create a **readme.md** text file in your ~/project folder that explains each part of the project. Specifications are below:

- **Part 1**: the readme.md file should include a description of your code design, classes, methods, and other key details. Your readme.md file should be at least several paragraphs in length and should explain what your project is, what each of the files you wrote for the project contains and does, and if you debated certain design choices, explain why you made them. Ensure you allocate sufficient time and energy to writing a **readme.md** that documents your code thoroughly.

- **Part 2**: documentation for Part 2 is in Jupyter notebook using markdown cells. Any additional libraries or external sources needed to run the code correctly must be explained in the **readme.md** file.

If you are unfamiliar with Markdown syntax, you might find GitHub's Basic Writing and Formatting Syntax helpful:
https://docs.github.com/en/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax .

1.2 Version Control

You must use version control to keep track of your progress during implementation using Git (we recommend GitHub). You should perform regular commits as you implement features and fix errors. Your commit comments should reflect the context of the changes that were made in each commit - a fellow developer can always diff the contents to see exactly what changed but that does not provide the context. You should try to ensure each commit comment contains a short subject line. Ensure that the repository is **private** (i.e. cannot be seen by anyone other than yourself). Failure to do so will be considered **plagiarism**.

1.3 Submitting your coursework

Your submission will comprise your entire version control repository for the three parts. Your repository must contain all source files required to run your program. You submission must include all your code for each part: Part 1 and Part 2.

- **Part 1**: includes different Python scripts to run your code.
- **Part 2**: Jupyter Notebook file that include code and explanation in the markdown for each step. The collected data saved in Step 1 must be submitted too.
- **Markdown (readme.md) file**: to explain any required instructions for running your code (please refer to 'code documentation and commenting section').
- If you have used any additional code for Part 2, beyond standard Python packages, then you will need to include an additional subdirectory containing additional code required to run your scripts. The authorship of any such code should be clearly stated in the markdown (**readme.md**).

For submitting your work on Blackboard, you will need to submit one zip file where the filename includes your UoB name and unit code separated by an underscore (for example 'jp16127_SEMTM0024'). Your zip file should contain the latest version of your Git repository. Note: your zip file **must be submitted to the Blackboard submission point**. Submissions sent via email or other means will not be accepted. sent via email or other means will not be accepted.

1.4 Grading

Your grade will be assessed on both correctness and style.

- **Correctness**: how well your program works according to specifications (Part 1), and how good is the quality of your analysis (Part 2).
- **Style**: the quality of your code and documentation.
- **Git:** Engagement with Git is expected as part of developing professional practice/vocational skills (ILO4) and will be part of the overall marking assessment.

Your code should be well-written and well-commented. You are encouraged to format your code per Python style guidelines (https://peps.python.org/pep-0008/). It should be clear enough for another programmer, such as the course staff, to understand and modify it if needed. Quality code is expected to meet our style requirements:

- Every function written is commented, in your own words, and uses a docstring that describes its behaviour, parameters, returns, and highlights any special cases.
- There is a comment at the top of each code file you write with your name, section, and a brief description of what that program does.

2 Part 1: Software Development (50%)

This part will assess whether you can program competently in Python, using object-oriented techniques (ILO 1). You will have to design, implement, test, and debug a text-based flower shop simulation. You must use an object-oriented approach with appropriate relationships between classes.

This simulation must be text-based and will not have a graphical user interface; hence it is not allowed to use turtle, pygame, tkinter, matplotlib, or any other API or external library for implementation. Only Python standard libraries can be used in Part 1 (https://docs.python.org/3/library).

2.1 Background

A flower shop creates different bouquets for customers to purchase. The flower shop needs to have employees (florists) to arrange the bouquets and maintain the greenhouse. It also needs to have the right flowers to create the appropriate bouquets and purchase the right supplies to maintain the plants. Each month the flower shop owner must decide whether to add/remove florists, how many bouquets of each type to sell, pay expenses, and then choose which vendor to restock supplies. The goal of the flower shop is to make a profit and avoid going bankrupt.

2.2 Task Overview

In this part, you are required to write a text-based (no GUI is permitted) Python program using object-oriented programming to simulate the monthly actions of the shop owner. Your program should do the following:

1. The shop offers the following types of bouquet, each requiring a specific quantity of materials.

Bouquet	Greenery	Roses	Daisies	Time to Prepare (in minutes)
Fern-tastic	4	0	2	20 minutes
Be-Leaf in Yourself	2	1	3	30 minutes
You Rose to the Occasion	2	4	2	45 minutes

The shop rents a greenhouse, where the supplies are stored. The following table shows the maximum quantity that the shop's greenhouse can store in any given month. The table shows the losses that occur each month due to depreciation (e.g. flowers dying off). When determining the quantity to depreciate, be sure to take the ceiling (i.e. round up to the nearest integer). The table also shows the greenhouse costs per month.

For example, if the shop if the shop has 100 bunches of daisies that did not get used during the quarter, only 85 bunches will still be available for the next month, and the flower shop will have incurred greenhouse costs of £80.

Supply	Greenhouse Capacity	Depreciation	Greenhouse Costs
Roses	200 bunches	40%/month	£1.50 / bunch
Daisies	250 bunches	15%/month	£0.80 / bunch
Greenery	400 bunches	5%/month	£0.20 / bunch

The shop pays a fixed monthly rent of £800 and keeps a monthly amount of cash. The shop starts with a cash balance of £7,500.

2. Regardless of how much demand there is, each florist gets paid per month and provides 80 hours of bouquet making labour per month. Each florist has a name and works at a fixed rate of £15.50/hour. You can only add/remove florists each month (i.e. even if there is no demand, you must pay florists for the entire month). Based on the confines of the shop, the maximum number of florists that can be employed is 4. Due to local business requirements, the shop must employ at least one florist. Assume the owner requires no salary and cannot sell any bouquets.

3. There are two vendors that sell the supplies as follows. Assume the vendors can deliver the supplies immediately, and that there is no limit to the amount that can be purchased.

Vendor Name	Roses	Daisies	Greenery
Evergreen Essentials	£2.80 / bunch	£1.50 / bunch	£0.95/ bunch

FloraGrow Distributors	£1.60 / bunch	£1.20 / bunch	£1.80 / bunch

The customer demand by bouquet type per month is shown in the following table

Bouquet Type	**Demand**	**Price**
Fern-tastic	175	£18.50
Be-Leaf in Yourself	100	£17.75
You Rose to the Occasion	250	£32.50

4. At the beginning of the program, prompt the user to enter the number of months to run the simulation. Default will be to run the simulation for six months. Assume the shop's greenhouse is full when the simulation starts.
5. Allow the owner to choose how many florists to add/remove and the amount of bouquets to sell for each bouquet type (these must be positive integers). When removing an employee, you are free to decide which employee is removed but be sure to indicate which one was removed.
6. For each month:
   (a) Prompt the owner to add/remove florists at the beginning of the month.
   (b) Prompt the owner for how much of each bouquet type to sell, ensuring that they do not exceed the demand, supplies, and labour constraints.
   (c) Given these decisions, assume the month runs accordingly. Based on how much was sold, update the shop's cash.
   (d) Pay the florists from cash.
   (e) Pay greenhouse costs from cash.
   (f) Apply the depreciation to the greenhouse.
   (g) Display the status of the shop to include the number of florists, the florist names (any specialities), and number of shop supplies in the greenhouse.
   (h) Purchase ingredients from the supplier so that the greenhouse is fully replenished.
   (i) If there is not enough cash on hand to pay expenses, then the shop goes bankrupt and the simulation ends.

The simulation terminates after the months specified, or the shop goes bankrupt.

2.3 Task Specifications

- **Create your classes (30% of Part 1)**. Your program should have at least three classes, each in their own file, to include:
  - **FlowerShop** class contains attributes and associated functions about the supplies, cash status, and number of florists.

- **Run your code (30% of Part 1)**. Create the main script (main.py) to run the classes you implemented. The main script must not define any classes. The classes should be imported into the main script. All constraints must be met. The simulation ends at the maximum number of months, or when the shop goes bankrupt.
- **Extend your program (40% of Part 1)**. Make sure the basic requirements are implemented correctly before extending your program. One example of an extension would be to modify your program so that a florist can be specialised in making one bouquet. They can still make other bouquets, but they can make the specialised bouquet in half the time. When hiring a florist, prompt for whether the florist has a speciality, and if so, prompt for the bouquet type they specialise in. Update so that the bouquet demand and florist speciality is considered when meeting customer demand. Demand for a bouquet type is first met by specialised florists followed by regular florists.
- **Errors and exceptions handling (included in the grading for each step)**. Your code will be taking user input, and will need to handle potential errors. Most errors occur in Null values and non-integer inputs. Make sure you handle these errors. Your program should continue to run when given invalid input. Examples of errors to be handled (please note this is not an exhaustive list):
    - No input
    - Non-integer or positive input
    - A florist already exists (no duplicate names)

2.4 Example Run

Below is an example of how to interact and what you should expect from the basic version of you text-based simulation, be sure to also add the extension. You may use different messages and output, just be sure that it is user friendly and meets the requirements specified above.

```
----------------------------------------------------------------


Welcome to the FlowerShop Simulator!


----------------------------------------------------------------


How many months would you like to run the game for?: 2
Month: 1

Before the month starts, there are some owner actions for you to carry
out. First, review the number of staff, then decide how many bouquets to
sell.

Current number of florists: 0
      How many florists would you like to hire? : 1
Please input florist name (one at a time): Gaby
Current staff:
      ['Gaby']
```

```
How much of each bouquet would you like to sell?

Ferntastic: 400
This exceeds the demand for this bouquet.
Ferntastic: 40
Be-Leaf in Yourself: 10
You Rose to the Occasion: 10

----------------------------------------
Month in progress...

----------------------------------------

End of month calculations:

Cash Balance, Month Start: £7500
    Income: £ 1242.5
    Outgoings:
        Employee costs: £ 1240.0
        Greenhouse costs: £ 361.0
        Rent: £ 800

Current shop status:

    Current staff: ['Gaby']

    Greenhouse quantity:
        Roses: 90.0
        Daisy: 102.0
        Greenery: 190.0

The greenhouse has spare capacity and needs to be restocked...

Do you want to purchase roses from Evergreen Essentials (0), or FloraGrow
Distributors (1)?
Press (i) if you would like to see price information from either supplier.
Input: 0

Do you want to purchase daisies from Evergreen Essentials (0), or
FloraGrow Distributors (1)?
Press (i) if you would like to see price information from either supplier.
Input: 0

Do you want to purchase greenery from Evergreen Essentials (0), or
FloraGrow Distributors (1)?
Press (i) if you would like to see price information from either supplier.
Input: 0

    + Flower restock costs: £729.5
    End of month Cash Balance: £6341.5

Month: 2

Before the month starts, there are some owner actions for you to carry
out. First, review the number of staff, then decide how many bouquets to
sell.
```

```
Current number of florists: 1
      How many florists would you like to hire? : 1
Please input florist name (one at a time): James
Current staff:
      ['Gaby', 'James']

How much of each bouquet would you like to sell?
Ferntastic: 40
Be-Leaf in Yourself: 20
You Rose to the Occasion: 30

----------------------------------------

Month in progress...

----------------------------------------

End of month calculations:

Cash Balance, Month Start: £6341.5
    Income: £ 2070.0
    Outgoings:
        Employee costs: £ 2480.0
        Greenhouse costs: £ 158.0
        Rent: £ 800

Current shop status:

    Current staff: ['Gaby', 'James']

    Greenhouse quantity:
        Roses: 36.0
        Daisy: 42.5
        Greenery: 133.0

The greenhouse has spare capacity and needs to be restocked...

Do you want to purchase roses from Evergreen Essentials (0), or FloraGrow
Distributors (1)?
Press (i) if you would like to see price information from either supplier.
Input: 0

Do you want to purchase daisies from Evergreen Essentials (0), or
FloraGrow Distributors (1)?
Press (i) if you would like to see price information from either supplier.
Input: 0

Do you want to purchase greenery from Evergreen Essentials (0), or
FloraGrow Distributors (1)?
Press (i) if you would like to see price information from either supplier.
Input: 0

    + Flower restock costs: £1024.1
    End of month Cash Balance: £4973.5
```

```
**********************************************************************

Congratulations! You have completed the simulation!
```

3 Part 2: Data Analytics (50%)

Jupyter notebook must be used for this part. Your answer for each step must include both your code and your explanation and answers to questions in the markdown cells. Please refer to the information given for markdown cells in each step. Effectively identify, deploy, and usefully integrate pre-existing packages of library code (ILO3). Analyse and interpret data science lifecycle, and understand how to implement each step by appraising, selecting, and applying appropriate tools and techniques (ILO4).

3.1 Step 1: Crawl a real-world dataset (10% of Part 2)

Scrape an external (remotely hosted) dataset to extract and analyse. Here are a couple of example sources (but you are not limited to using these – feel free to explore):

- https://fbref.com/en/
- https://www.reddit.com

Your dataset can be either extracted from APIs and/or via web scraping. **Reading data from non-external (i.e. local) files is not permitted for this task**. Extracted data must contain **at least 5 columns and 150 rows**. Save your extracted data in a CSV format.

Note: 'pandas.read table' is not allowed for this task. You may use data from multiple sources and merge them as long as **at least one of them is crawled from APIs/web scraping that satisfies the requirement**.

Explain in the markdown cells: *where does the data come from? What are the variables of interest? How was the data scraped/collected?*

3.2 Step 2: Perform data preparation and cleaning (20% of Part 2)

- Load the dataset into a data frame using Pandas
- Handle missing data, if any
- Handle any outliers or inconsistencies in the data, if any
- Perform any additional steps to enrich your data (parsing dates, creating additional columns/features etc)

Explain in markdown cells: *steps to prepare, clean your data, or extract new features*.

3.3 Step 3: Perform exploratory analysis (30% of Part 2)

Explore your data, examples are as follows:

- Compute the mean, sum, range, and other interesting statistics for numeric columns
- Explore distributions of numeric columns using histograms etc.
- Explore the relationship between columns using scatter plots, bar charts, etc

Explain in markdown cells: *outline in detail your entire analysis. Explain your insights clearly.* You should include a brief description of the intent/interpretation of each plot.

3.4 Step 4: Ask a question about your data (30% of Part 2)

- Ask **one interesting question** about your dataset. Your question must be complex enough to have sub-questions. The quality of your question is assessed based on the complexity and how interesting it is. Trivial and basic questions will not give you full marks.

- Answer the question and sub-questions by computing the results using, for example, numpy, pandas, scikit-learn, or by plotting graphs using matplotlib. Perform grouping/aggregation wherever necessary. You can use hypothesis testing and basic modelling (such as regression models) to answer the proposed questions.

3.5 Step 5: Summarise and write a conclusion (10% of Part 2)

Explain in markdown cells:

- Write a summary of what you've learned from the analysis
- Share ideas for future work on the same topic using other relevant datasets/sources

4 Marking Criteria

4.1 Distinction (70% and above)

The submission demonstrates an excellent command of software design, development, and data analysis principles. In Part 1, classes are well structured, extensible, and fully functional; code executes correctly and efficiently, and the extensions show creativity and strong problem-solving ability. Robust error and exception handling, thoughtful validation, and elegant code style are all evident. Code is excellently documented with both docstrings and inline comments. In Part 2, the analytical work is rigorous and insightful, demonstrating a deep understanding of data handling, method selection, and interpretation of results. The overall work reflects independence, originality, and technical mastery.

4.2 Merit (60–69%)

Work in this range is well-executed and demonstrates solid understanding across both parts of the coursework. In Part 1, the code meets most requirements, is logically structured, and generally well styled, though minor inefficiencies or design inconsistencies may be present. Code is generally well-documented throughout. Extensions are sound but may lack innovation or depth. In Part 2, the analysis is coherent and accurate, showing clear understanding but limited critical insight or exploration. Testing and error handling are present but not comprehensive. The work is of a high standard overall, with minor weaknesses preventing distinction-level performance.

4.3 Pass (50–59%)

A pass-level submission demonstrates basic competence in implementing and understanding the main tasks. In Part 1, the programme generally runs and meets the core requirements, but the structure may be simplistic or partially incomplete, with weaknesses in style, documentation, or handling of errors. The extended functionality is attempted but lacks refinement or robustness. In Part 2, the data analysis is descriptive rather than analytical, showing limited engagement with method or interpretation. Version control may be minimal or inconsistently applied. While the work meets minimum MSc requirements, it lacks polish, precision, or depth.

4.4 Fail (Below 50%)

Work in this band does not demonstrate sufficient understanding or achievement of the learning outcomes. In Part 1, the code may not run, key elements may be missing, or classes are poorly designed or incorrect. There is little or no evidence of testing, validation, or exception handling, and the overall implementation shows weak technical competence. In Part 2, the analysis is incomplete, incorrect, or lacks any meaningful interpretation. Poor or absent version control, incomplete submission, or evidence of misunderstanding core concepts would all fall within this category. The work fails to meet postgraduate expectations for independence, rigour, and correctness