ISCG5421
Programming Principles and Practice

Assignment

Semester 1, 2020

School of Computing, Electrical and Applied Technology

Due Date: See the table on page 2

Total Marks: 100
Course Weighting: 50%

Learning outcomes covered in this assignment

1. Implement the designs by writing well-structured programs that follow enforced programme language conventions and programming standards.
2. Identify the fundamental data requirements of an intermediate-level program.
3. Apply the logic structures of the language.
4. Select and use intermediate-level data structures.
5. Write a complete program whilst adhering to available coding standards.

Assignment instructions:

- You will work individually on this assignment.
- Select an idea from appendix 1. For the selected idea you will develop a small intermediate-level Python program. You can choose your own idea which is subject to lecturer approval.
- You will identify the fundamental data requirements of an intermediate-level program, select and use intermediate-level data structures, and implement your designs by writing a complete and well-structured program that adheres to programming language standards and conventions.
- There are five tasks to this assignment
  Task 1: Setup
  Task 2: Design
  Task 3: Implementation
  Task 4: Video demo
  Task 5: In-class presentation
- You will create a developer journal and record everything you do for the tasks. Store it in a text file with your code in the GitHub repository. There are **10 marks** for this journal. See appendix 2 for sample developer journal.
- Each task has different submission date. See the table on page 2.

Submission dates and time:

Tasks	Submission dates	Submission time
Task 1: Setup	6th April 2020	8 pm
Task 2: Design	6th April 2020	8 pm
Task 3: Implementation	31st May 2020	8 pm
Task 4: Video demo	31st May 2020	8 pm
Task 5: In-class presentation	1st June 2020	In-Class

Assignment Submission:

- Clone your repository into a directory, then zip and submit the entire directory containing all previous versions to Moodle. Don't delete the GitHub repository. Your lecturer will be using this to check progress and help throughout the semester.
- For the video demo, upload it to YouTube as Unlisted (not Private), then add a link to it in your README.md file.
- Filename: *5421_Assignment_S1_2020_Kris_Pritchard_(ID1234567).zip*

Task 1: Setup [5 Marks]

You need to do the following setup before creating your program.

Create a repository on GitHub for your program. [1 mark]

Add a detailed README describing with at least three paragraphs what your program will do or what problem it solves. Use additional diagrams where appropriate. [2 marks]

Create at least 10 descriptive tasks on GitHub that break down what you will need to do, and add rough time estimates to each task. Estimates should be no longer than 8 hours, so break them into smaller tasks. [2 marks]

Task 2: Design [15 Marks]

Based on your setup in task 1 you now need to design your program. Your design must include the following:

Identify and create class diagrams for at least 5 classes, then add those diagrams to the README. [4 marks]

Create a file that uses if __name__ == '__main__': which just prints 'Hello' for your program entry point. [1 mark]

Create at least two additional Python files that will act as modules for each of the classes and their associated functions. [1 mark]

Add detailed comments to each of the modules/files describing the purpose of each file. [3 marks]

Add placeholder classes and functions with comments describing their purpose. Use your class diagrams to help with this. [3 marks]

Draw and add to GitHub at least 3 simple MSPaint style diagrams showing example screens of how your application will look. [3 marks]

Task 3: Implementation [55 Marks]

You will implement your designs by writing a well-structured program that follow enforced program language principles and conventions. You must include the following:

Select and use at least two data structures from the list given below. Add reason(s) why you have selected these data structures in your programs comments. [5 marks]

- tuple
- list
- dictionary
- set
- stack
- queue

The program implementation must:

- Implement classes for the diagrams you designed in Task 2.
  You can change these diagrams as you go, but need to keep the diagrams and code in sync with each other. [5 marks]

- Have at least 10 working unit tests that have descriptive names or comments explaining what they test. [5 marks]

- Use docstrings and comments which state the purpose of your code, on every module, class, function, and block of code. [5 marks]

- Use pylint to ensure that your program adheres to Python conventions. [4 marks]

- Contains a requirements.txt file with any external packages required. [1 mark]

- Export data to a file using either multithreading or multiprocessing and time.sleep(5) to simulate the application taking 5 seconds to perform a task. [5 marks]

- Create 5 custom exceptions and raise/handle each of them with relevant logging. [5 marks]

- Use all 5 logging levels for logging messages (debug, info, warning, error, critical) at different points in your program. [5 marks]

- Either use the **requests** module to get data from an external API, or use the **socket** module to allow multiple clients to communicate with each other. [5 marks]

- Use programming principles learned in this course to minimise code duplication and maximise code reuse. [1 mark]

- Use the **datetime** module to display the current date and time somewhere when the application is started. [1 mark]

- Avoid using global variables where possible, using objects to hold application state. [1 mark]
- Show usage of async/await keywords in at least one location. [2 marks]
- Program runs without unexpected bugs or behaviour. [5 marks]

Task 4: Video Demo [10 Marks]

Create a 60-90 second YouTube video demonstrating your apps features, then add the link to the video to your README.

Task 5: In-class presentation [5 marks]

Give a 2-3 minute in-class presentation of your app and talk about key challenges encountered while building it, and how you overcame them.

Marking Schedule

Task	Marking Criteria	Marks	Given	Comments
1	Created a GitHub Repository. Detailed README file with description Created 10 or more descriptive tasks on GitHub.	1 2 2		
2	5 class diagrams are identified and created. Student has added diagrams to README Uses a __name__ == '__main__' program entry point. Has at least 2 additional modules. Modules/Files contain detailed comments describing their purpose. Placeholder classes/functions from class diagrams have detailed comments describing their purpose. Has at least 3 simple diagrams of screens the app will have.	4 1 1 3 3 3		
3	Selected and used at least two data structures. Implemented classes based on class diagrams and keeps them in sync with each other. Has at least 10 working and well-named unit tests. Has docstrings and comments on all files/classes/functions/etc. Pylint detects no problems with code.	5 5 5 5 4		

	Has a requirements.txt file for external packages. Exports data to a file using either multithreading or multiprocessing with time.sleep(5). Has created at least 5 custom exceptions. Uses all 5 logging levels extensively. Program communicates via the Internet (requests, sockets, or another module). Uses programming principles. Uses **datetime** module. Contains no unnecessary global variables. Uses async/await. Program runs correctly without unexpected bugs or behaviour.	1 5 5 5 5 1 1 1 2 5		
4	Created a 60-90 second YouTube video demoing the app with narration.	10		
5	Give 2-3-minute presentation of program and talk about challenges, things learned, etc.	5		
	Has a detailed developer journal with at least 30 days of entries, and at least 3 entries per day. Push this to GitHub daily.	10		
Total Marks		**100**		

Late Submission of Assignments:

Assignments submitted after the due date and time without having received an extension through Affected Performance Consideration (APC) will be penalised according to the following:

- 10% of marks deducted if submitted within 24hrs of the deadline,
- 20% of marks deducted if submitted after 24hrs and up to 48hrs of the deadline,
- 30% of marks deducted if submitted after 48hrs and up to 72hrs of the deadline,
- No grade will be given for an assignment that is submitted more than 72hrs after the deadline.

Assistance to other Students:

Students themselves can be an excellent resource to assist the learning of fellow students, but there are issues that arise in assessments that relate to the type and amount of assistance given by students to other students. It is important to recognise what types of assistance are beneficial to another's learning and also what types of assistance are unacceptable in an assessment.

Beneficial Assistance:

- Study Groups
- Discussion
- Sharing Reading Material
- Reading the available online and library resources

Unacceptable Assistance:

- Working together on one copy of the assessment and submitting it as own work
- Giving another student your work
- Copying someone else's work, this includes work done by someone not on the course
- Changing or correcting another student's work
- Copying from books, the Internet etc. and submitting it as own work; anything taken directly from another source must be acknowledged correctly; show the source alongside the quotation
- Don't copy code from a website or video tutorial and pretend you made it or slightly change it. This will be an instant fail (0%).

Appendix 1

Sample Project Ideas:

1. Grocery Store Inventory System
2. Movie Rental Inventory System
3. Payroll System
4. Bank Accounts System
5. Multiplayer Retro Video Game
6. GUI based Accounting Application
7. Web based Chat Application

If you choose to do something different you need to get an approval from your lecturer.

Appendix 2

Developer Journal Example:

===
March 23rd 2020 – Started @ 2:30PM
Going to work on Task #5, to try and implement saving user data to text files.
Estimate: 30 minutes

Got a bug, file isn't being created correctly.
5:15PM Still stuck on bug. Switching to Task #7 to add a button.
5:45PM Added a button, closed Task #7.

7PM: Fixed bug with file not being created. I was accidentally saving it to the wrong directory.
Future Suggestion: Make sure to always check which directory I save files to.
Actual time taken: ~4 hours.

7:30PM: Need to create an Invoice class which has save() method.
Estimate: 1hr

Stopped @ 9pm.

===
March 24th 2020 – 10AM
Continuing on Task #6 from yesterday – Creating class for Invoice

10:30AM: Finished class for Invoice (Task #6).
Actual time taken: ~2 hours.

10:45AM: Debug Issue #12 – Game collisions aren't working and object disappears off screen.
Estimate: 4 hours

1PM: While debugging game collisions I realized that I forgot to enable logging. I enabled logging and instantly saw why collisions weren't working. I wasn't checking if a collision occurred before moving the object.
Future Suggestion: When creating classes, start with a list of pseudo-code comments of things I need to remember to check. Also don't forget to enable logging when debugging.
Actual time taken: ~2 hours

2PM: Refactored / rewrote the do_stuff() function to be smaller and more clear.

4PM: Found a StackOverflow post for a possible solution to Task #9:
https://stackoverflow.com/questions/1077347/hello-world-in-python

5PM: Solution didn't work. Out of time for this feature so will cut it and work on a different task.
Etc.. Write in your journal and work on your code daily.