

Programming Fundamentals (COSC2531)

Final Coding Challenge

Assessment Type	Individual assessment (no group work). Submit online via Canvas/Assignments/Final Coding Challenge. Marks are awarded per rubric (please see the rubric on Canvas). Clarifications/updates may be made via announcements. Questions can be raised via the Canvas discussion forum.
Due Date	End of Week 14 (exact time is shown in Canvas/Assignments/Final Coding Challenge) Deadline will not be advanced nor extended. Please check Canvas/Assignments/Final Coding Challenge for the most up to date information regarding the assignment. As this is a major assignment, a university standard late penalty of 10% (i.e., 3 marks) per each day applies for up to 5 days late, unless special consideration has been granted.
Weighting	30 marks out of 100

1. Overview

The main objective of this final project is to assess your capability of program design and implementation for solving a non-trivial problem. You are to solve the problem by designing a number of classes, methods, code snippets and associating them towards a common goal. If you have questions, please ask via the relevant Canvas discussion forums in a general manner; for example, you should replicate your problem in a different context in isolation before posting, and you must not post your code on the Canvas discussion forum.

2. Assessment Criteria

This assignment will determine your ability to:

- i. Follow coding, convention, and behavioural requirements provided in this document and in the course lessons;
- ii. Independently solve a problem by using programming concepts taught in this course;
- iii. Design an OO solution independently and write/debug in Python code;
- iv. Document code;
- v. Provide references where due;
- vi. Meet deadlines;
- vii. Seek clarification from your "supervisor" (instructor) when needed via the Canvas discussion forums; and
- viii. Create a program by recalling concepts taught in class, understand and apply concepts relevant to solution, analyse components of the problem, evaluate different approaches.

3. Learning Outcomes

This assignment is relevant to the following Learning Outcomes:

1. Analyse simple computing problems.
2. Devise suitable algorithmic solutions and code these algorithmic solutions in a computer programming language (i.e., Python).
3. Develop maintainable and reusable solutions using object-oriented paradigm.

4. Assessment Details

Please ensure that you have read Sections 1-3 of this document before going further.

Problem Overview: In this final coding challenge, you are asked to develop a Python program with the Object-Oriented Programming paradigm, named **my_record.py**, that can read data from files and perform some operations. You are required to implement the program following the below requirements. Note that we will give you some files for you to run with your developed program, BUT you should change the data in these files to test your program. **During the marking, we will use different data/files to test the behavior of your program.**

Requirements: Your code must meet the following **functionalities**, **code**, and **documentation** requirements. Your submission will be graded based on the **rubric** published on Canvas. Please ensure you read all the requirements and the rubric carefully before working on your assignment.

A - Functionalities Requirements:

There are **4 levels**, please ensure you only attempt one level after completing the previous level.

----- **PASS LEVEL (15 marks)** -----

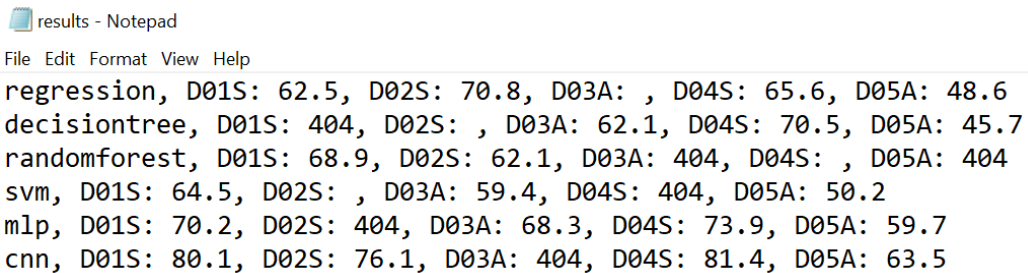
Your project is to implement the required functionalities in the Object-Oriented (OO) style with at least three classes: **Records**, **Dataset**, and **Algorithm**. You need to design appropriate static/instance variables, constructors, and static/instance methods in these classes. The class related info should be encapsulated inside the corresponding class.

At this level, your program can read data from the *result file* specified in the command line, which stores the results of various algorithms on some datasets (*result file*). Your program should create a list of *Dataset* objects, a list of *Algorithm* objects, and a variable (you can think carefully about which data type to use) to store the results of all algorithms on all the datasets. You should design the classes properly so that these actions can be encapsulated within the appropriate classes. Note that, at this level, we only know the IDs of the datasets and the names of the algorithms. These IDs and names are all unique.

In the main class, your program should create a *Records* object, call its method *read_results(result_file_name)* to load all the data from the result file, and then call the *display_results()* method to display the results of the algorithms on all the datasets in the required format (as specified in the following).

Below is an example of the file that stores the results of the algorithms in all the datasets – see the next page. The data fields in this result file are separated by commas, colons, and new lines. Each row contains the algorithm name, and the results of the algorithm on all the datasets. The format of each row is always *algorithm_name, dataset_1: result_1, dataset_2: result_2, ...*. When the result on a particular dataset does not exist (because the algorithm failed to work on that dataset), the data field

after the colon corresponding to that dataset will be empty. When the result on a particular dataset is not yet available (on-going, waiting for the final result), the data field after the colon corresponding to that dataset will be 404. For example, in our result file example, the algorithm *regression* has a result of 62.5 on the dataset D01S, 70.8 on the dataset D02S, fails to have a result on the dataset D03A, has a result of 65.6 on the dataset D04S, and has a result of 48.6 on the dataset D05A. The algorithm *decisiontree* is still waiting for the result on the dataset D01S, fails to have a result on the dataset D02S, has a result of 62.1 on D03A, 70.5 on D04S, and a result of 45.7 on the dataset D05A. You can assume there are no duplicate or redundant rows. And you can assume the format of the data in the file is always correct.



```

results - Notepad
File Edit Format View Help
regression, D01S: 62.5, D02S: 70.8, D03A: , D04S: 65.6, D05A: 48.6
decisiontree, D01S: 404, D02S: , D03A: 62.1, D04S: 70.5, D05A: 45.7
randomforest, D01S: 68.9, D02S: 62.1, D03A: 404, D04S: , D05A: 404
svm, D01S: 64.5, D02S: , D03A: 59.4, D04S: 404, D05A: 50.2
mlp, D01S: 70.2, D02S: 404, D03A: 68.3, D04S: 73.9, D05A: 59.7
cnn, D01S: 80.1, D02S: 76.1, D03A: 404, D04S: 81.4, D05A: 63.5
  
```

Your program should print a message indicating how to run the program if no result file is passed in as a command line argument. Otherwise, it can display a table showing the results of all algorithms for all the datasets, and two sentences showing the total number of algorithms and datasets, the number of nonexistent results, and the number of on-going results. The result of an algorithm for a dataset is displayed as a number with 1 digit after the decimal point. In the table, if the result of an algorithm does not exist for a particular dataset, then the data field at that location has an “XX” symbol. On the other hand, if the result of an algorithm on a dataset is not available yet (ongoing), then the data field at that location is shown as a double dash (--). Also, the data in the columns in the table are aligned, in particular, the algorithm names are aligned to the left whilst the results for all the datasets are aligned to the right.

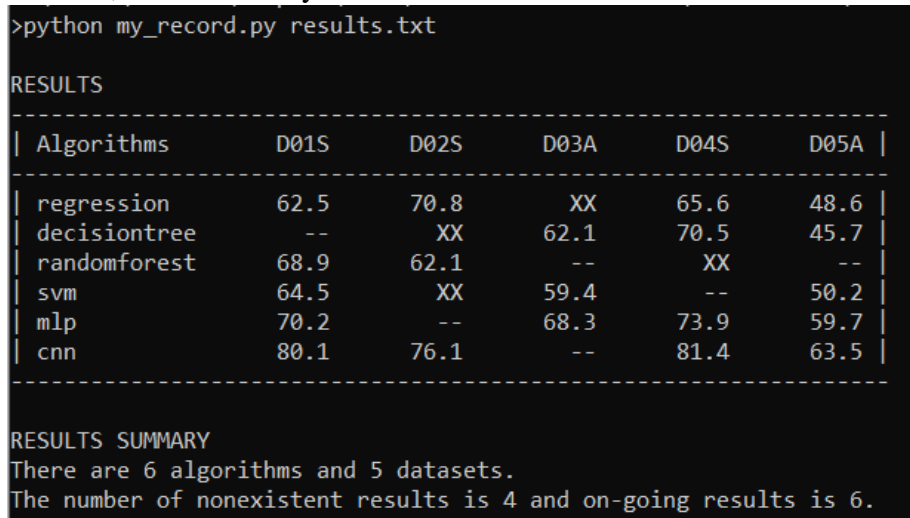
The printed messages corresponding to two scenarios need to be exactly as below:

1. This is when no result file is passed in as a command line argument.

```

>python my_record.py
[Usage:] python my_record.py <result file>
  
```

2. This is when a result file is passed in as command line arguments. Note that users can specify a different file name, not necessarily the name *results.txt*.



```

>python my_record.py results.txt

RESULTS
-----
| Algorithms      D01S      D02S      D03A      D04S      D05A |
-----
| regression      62.5      70.8      XX        65.6      48.6 |
| decisiontree    --        XX        62.1      70.5      45.7 |
| randomforest    68.9      62.1      --        XX        --   |
| svm             64.5      XX        59.4      --        50.2 |
| mlp            70.2      --        68.3      73.9      59.7 |
| cnn            80.1      76.1      --        81.4      63.5 |
-----

RESULTS SUMMARY
There are 6 algorithms and 5 datasets.
The number of nonexistent results is 4 and on-going results is 6.
  
```

----- CREDIT LEVEL (3 marks, you must only attempt this level after completing the PASS level) -----

At this level, your program can support more information about datasets. Now, apart from the ID, each dataset will have a name, complexity weight, the number of data points in the dataset, and the source where the dataset was published. All IDs, names, complexity weights, the number of data points, and the dataset published sources can be modified. There are two types of datasets: *Simple Dataset* and *Advanced Dataset*. Simple datasets are datasets with IDs ending with *S* whilst advanced datasets are datasets with IDs ending with *A*. Simple datasets all have the same complexity weight, which is 1 by default. Advanced datasets can have different complexity weights, but never be larger than 5. If this requirement regarding the complexity weight is violated for any dataset in the dataset file, the program should terminate and display the message indicating the issue and the corresponding dataset. You should define appropriate variables, getters, and setters for datasets.

Also, a dataset should have a method to compute some useful statistics, e.g., the average result, the range of the results, the number failed algorithms (it is your choice to define the necessary statistics to compute). You can define extra methods if necessary.

At this level, the dataset file will have more information. The dataset file includes the dataset IDs, the dataset names, the complexity weights, the number of data points, and the dataset published sources. You can assume there are no duplicate or redundant datasets. You can assume all the datasets available in this file are also available in the result file (in the PASS level), and vice versa.

```

datasets - Notepad
File Edit Format View Help
D01S, Iris, 1, 150, UCI
D02S, Wine, 1, 178, UCI
D03A, MNIST, 2, 70000, Personal
D04S, BreastCancer, 1, 569, UCI
D05A, CIFAR, 4, 60000, Personal

```

At this level, your program can now print the dataset information table on screen and save the table into a file name *reports.txt*. For example, given the above result file (in the PASS level) and the dataset file (in this level), the dataset information table should look like below. Note the content within the *reports.txt* file should also look the same.

In this level, users can pass two file names via the command line arguments: the result file and the dataset file. The program will then display **two tables**: the result table from the PASS level, and the dataset table from this level.

```

>python my_record.py results.txt datasets.txt
RESULTS
-----
| Algorithms      | D01S | D02S | D03A | D04S | D05A |
|-----|-----|-----|-----|-----|-----|
| regression      | 62.5 | 70.8 | XX   | 65.6 | 48.6 |
| decisiontree    | --   | XX   | 62.1 | 70.5 | 45.7 |
| randomforest    | 68.9 | 62.1 | --   | XX   | --   |
| svm             | 64.5 | XX   | 59.4 | --   | 50.2 |
| mlp             | 70.2 | --   | 68.3 | 73.9 | 59.7 |
| cnn             | 80.1 | 76.1 | --   | 81.4 | 63.5 |
|-----|-----|-----|-----|-----|-----|

RESULTS SUMMARY
There are 6 algorithms and 5 datasets.
The number of nonexistent results is 4 and on-going results is 6.

```

DATASET INFORMATION								
DatasetID	Name	Type	Weight	Ndata	Source	Average	Range	Nfail
D01S	Iris	S	1	150	UCI	69.2	62.5 - 80.1	0
D02S	Wine	S	1	178	UCI	69.7	62.1 - 76.1	2
D03A	MNIST	A	2	70000	Personal	63.3	59.4 - 68.3	1
D04S	BreastCancer	S	1	569	UCI	72.8	65.6 - 81.4	1
D05A	CIFAR	A	4	60000	Personal	53.5	45.7 - 63.5	0

DATASET SUMMARY
The most difficult dataset is CIFAR with an average result of 53.5.
The dataset with the most failures is Wine with the number of failures being 2.

Note users can specify different file names, not necessarily the names *results.txt*, and *datasets.txt*. The first file is always the result file, the second file is the dataset file.

In the Dataset table, the DatasetID column shows the IDs of the datasets, the Name column shows the dataset names, the Type column shows the types of the datasets (S for simple datasets, and A for advanced datasets), the Weight column shows the complexity weights of the datasets, the Ndata column shows the number of data points in the datasets, the Source column shows the dataset published sources. The Average column displays the average results of all the algorithms for each dataset (based on complete results only). The Range column displays the minimum and maximum results of all the algorithms for each dataset (based on complete results only). In both the Average and Range columns, the results are shown with 1 digit after the decimal point. The Nfail column shows the number of algorithms fails on each dataset (nonexistent results).

Apart from this table, your program should also display two sentences. The first sentence indicates the most difficult dataset. The most difficult dataset is the dataset with the lowest average result; if there are multiple datasets with the same lowest average results, you can choose either to display one or all datasets. The second sentence indicates the dataset with the most failures, i.e., the dataset that has the highest number of failed algorithms. Similarly, if there are multiple datasets with the same highest number of failures, you can choose to display one or all datasets.

Finally, it's worth emphasizing that with the program developed in this level, the behavior of command-line arguments is still kept as in the PASS level. Specifically, when the user does not pass any command-line argument, the program will display the usage message and when the user passes in a result filename, the program will display the results table. Based on this level, when the user passes in a result filename and a dataset filename, the program will display the results and dataset tables.

----- DI LEVEL (3 marks, you must only attempt this level after completing the CREDIT level) -----

At this level, your program can support two types of algorithms: *ML Algorithms* and *DL Algorithms*. All the algorithms need to be able to successfully run all the simple datasets (complete and ongoing results are counted as "successfully run"). ML algorithms need to be able to successfully run (complete and ongoing) on at least 1 advanced dataset whilst DL algorithms need to be able to successfully run (complete and ongoing) on at least 2 advanced datasets.

An algorithm should have a method to compute some useful statistics for the algorithm, e.g., the average results, the number of failed datasets, the failed datasets, the number of ongoing results (it is your choice to define the necessary statistics to compute). It should also have a method to check if an

algorithm satisfies the aforementioned requirement regarding the number of failed simple and advanced datasets for each algorithm. You can define extra methods for the algorithms if necessary.

At this level, the algorithm file will have more information. The file includes the algorithm names, the algorithm types, the years the algorithms were invented, the list of possible authors. You can assume there are no duplicate or redundant algorithms. You can assume all algorithms in the result file (in the PASS level) appeared in this file and vice versa. An example of the algorithm file is as follows.

```

algorithms - Notepad
File Edit Format View Help
regression, ML, 1885, John, Tom, Kate
decisiontree, ML, 1963, Mike, Tim
randomforest, ML, 2001, Bella, Ella
svm, ML, 1964, Van
mlp, DL, 1958, Jeff, Sarah
cnn, DL, 1989, Yann

```

At this level, users can pass three file names via the command line arguments: the result file, the dataset file, and the algorithm file. The program will then display **three tables**: the result table from the PASS level, the dataset table from the CREDIT level, and the algorithm table from the DI level. All these tables will also be stored in the text file *reports.txt* (from the CREDIT level).

Given the above result file (in the PASS level), the dataset file (in the CREDIT level), and the algorithm file (in this level), the result and dataset tables look the same as in the previous levels whilst the algorithm information table should look like below. Note the content within the *reports.txt* file should also look the same. Also, in the command line, users can specify different file names, not necessarily the names *results.txt*, *datasets.txt*, *algorithms.txt*. The first file is always the result file, the second file is the dataset file, and the last file is the algorithm file.

ALGORITHM INFORMATION							
Name	Type	Year	Authors	Average	Nfail	FailDataset	Nongoing
regression	ML	1885	John-Tom-Kate	61.9	1	D03A	0
decisiontree (!)	ML	1963	Mike-Tim	59.4	1	D02S	1
randomforest (!)	ML	2001	Bella-Ella	65.5	1	D04S	2
svm (!)	ML	1964	Van	58.0	1	D02S	1
mlp	DL	1958	Jeff-Sarah	68.0	0		1
cnn	DL	1989	Yann	75.3	0		1

ALGORITHM SUMMARY							
The best algorithm is cnn with an average result of 75.3.							
The algorithm with the least failure is cnn with the number of failures being 0.							

In the algorithm table, the Name column shows the algorithm names, the Type column shows the types of the algorithms (ML for ML algorithms and DL for DL algorithms), the Year column shows the invented years whilst the Author columns shows the list of authors separating by “-”. The Average column shows the average results of the algorithms (over all the complete results). The Nfail column shows the number of datasets the corresponding algorithm fails. The FailData column shows the names of the datasets where the algorithm fails. The Ngoing column shows the number of on-going results (results not yet available).

In the algorithm table, the data in the *Name* column is aligned to the left whilst the data in all other columns are aligned to the right. Besides, in the table, if an algorithm doesn’t satisfy the requirement of minimum successful simple and advanced datasets, then an exclamation mark (!) will be added next

to the algorithm's name. For example, in the table above, the algorithms *decisiontree* has an exclamation mark next to its name because it fails to run on the dataset D02S which is a simple dataset. Similar arguments can be applied to the algorithms *randomforest* and *svm*.

Apart from the algorithm information table, two additional sentences are required. The first sentence indicates the best algorithm (the algorithm with the highest average result) along with its average result. If there are multiple algorithms with the highest average results, you can choose to display one or all these algorithms. The second sentence indicates the algorithm with the least number of failures along with its number of failures. Similarly, if there are multiple algorithms with the least number of failures, you can choose to display one or all these algorithms.

Finally, it's worth emphasizing that with the program developed in this level, the behavior of command-line arguments is still kept as in the PASS and CREDIT levels. Specifically, when the user does not pass any command-line argument, the program will display the usage message. When the user passes in a result filename, the program will display the results table. When the user passes in a result filename and a dataset filename, the program will display the results and dataset tables. Based on this level, when the user passes in a result filename, a dataset filename, and an algorithm filename, the program will display the results, dataset, and algorithm tables.

----- HD LEVEL (6 marks, you must only attempt this level after completing the DI level) -----

At this level, your program can handle some variations in the files using built-in/custom exceptions:

1. When any of the file cannot be found, then your program should print a message indicating the names of the files that cannot be found and then quit gracefully. You can assume users always type the file names in the right order in the command line, e.g., the result file first, the dataset file second, and the algorithm file third.
2. The program will exit and indicate the corresponding error when one of these errors occurs:
 - a. The result file is empty;
 - b. Any of the results in the result file is not a valid number (except the empty data field);
 - c. The dataset ID doesn't start with the letter D, doesn't have 2 integer numbers in the middle, and doesn't end with either S or A.

The program will have some additional requirements (some might be challenging):

1. In this level, the algorithm information table now has one additional column, *Score*. The *Score* column is computed based on the scores the algorithms obtained if they come first, second, or third on a dataset (i.e., has highest, second highest, or third highest results). An algorithm obtains 3 pts if it comes first, 2 pts if it comes second, and 1 pts if it comes third. Algorithms come after the third place will not have any points. The total score of an algorithm is the sum of all the scores of all the datasets it finished. For example, in the table below, the algorithm *regression* has a score of 2 as it comes second on the dataset D02S. The algorithm *cnn* has a score of 12 as it comes first on 4 datasets: D01S, D02S, D04S, and D05A.

ALGORITHM INFORMATION								
Name	Type	Year	Authors	Average	Score	Nfail	FailDataset	Nongoing
regression	ML	1885	John-Tom-Kate	61.9	2	1	D03A	0
decisiontree (!)	ML	1963	Mike-Tim	59.4	3	1	D02S	1
randomforest (!)	ML	2001	Bella-Ella	65.5	2	1	D04S	2
svm (!)	ML	1964	Van	58.0	2	1	D02S	1
mlp	DL	1958	Jeff-Sarah	68.0	9	0		1
cnn	DL	1989	Yann	75.3	12	0		1

ALGORITHM SUMMARY

The best algorithm is cnn with an average result of 75.3.

The algorithm with the least failure is cnn with the number of failures being 0.

- In this level, the dataset information table is split into two tables: one for the simple datasets and one for the advanced datasets. The simple dataset table consists of all the information for the simple datasets whilst the advanced dataset table consists of all the information for the advanced datasets. All the formats and information of all the columns are same as with the previous one table. Each table is sorted (from high to low) based on the Average column. Note that this is also reflected in the *reports.txt* file.
- In this level, the algorithm information table is also split into two tables: one for the ML algorithms and one for the DL algorithms. The ML algorithm table consists of all the information for the ML algorithms whilst the DL algorithm table consists of all the information for the DL algorithms. All the formats and information of all the columns are same as with the previous one table. Each table is sorted (from high to low) based on the Score column. Note that this is also reflected in the *reports.txt* file.
- The *reports.txt* is accumulated, which means when the program runs, it will not overwrite the previous report, but instead, it places the new report on top of the file (i.e., the newest report is always at the top of the *reports.txt* file). In addition, the date and time when the report was generated (in the format *dd/mm/yyyy hh:mm:ss*, e.g. *01/10/2023 09:45:00*) are also saved in the text file for each report.

B - Code Requirements:

You must demonstrate your ability to program in Python by yourself, i.e., you should not use any Python packages/libraries that can do most of the coding for you. **The only Python libraries allowed in this assignment are sys, os, and datetime.** If other packages/libraries are used, you will get a 0 mark or a heavy penalty.

Your program at all levels should be fully OO, e.g., no variables, methods, or code snippets dangling outside a class. Your main program should simply create an object and run its methods to invoke methods from other classes to perform the required functionalities.

You should test/verify the program with different text files (not just run with our text files) to ensure your program satisfies all the required functionalities.

Your code needs to be formatted consistently. You must not include any unused/irrelevant code (even inside the comments). What you submitted must be considered the final product.

You should design your classes carefully. You may use a class diagram to assist with the design. **In the DI and HD levels, you are required to provide a detailed class diagram to show your class design.** An example of a class diagram is shown below on the next page – Figure 1 (note that this is a

class diagram of a different programming assignment). In the diagram, the variables and methods of each class are shown. Note that if your code is at the PASS or CREDIT level, you do not need to submit any diagram; a diagram at these levels would NOT result in any mark.

You could use tools like PowerPoint, Keynotes, or online tools like moqups.com to draw the diagram. **The diagram needs to be submitted in jpg, gif, png, or pdf format.**

Finally, note that in places where this specification may not tell you how exactly you should implement a certain feature, you need to use your judgment to choose and apply the most appropriate concepts from our course materials. You should follow answers given by your "client" (or "supervisor" or the teaching team) under Canvas/Discussions/Discussion on Final Coding Challenge.

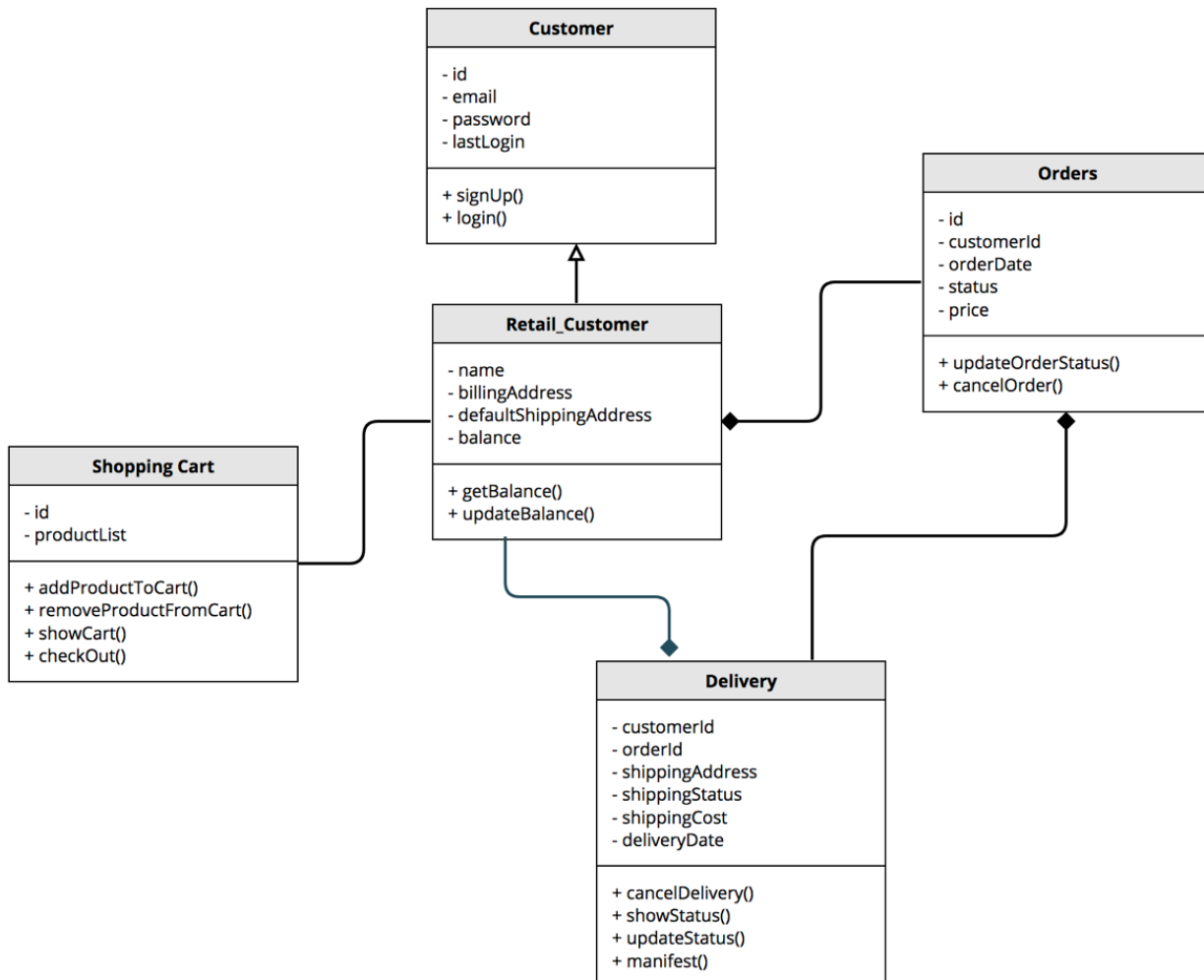


Figure 1. An example of a class diagram

C - Documentation Requirements:

You are required to write comments (documentation) as a part of your code. Writing documentation is a good habit in professional programming. It is particularly useful if the documentation is next to the code segment that it refers to. NOTE that you don't need to write an essay, i.e., you should keep the documentation succinct.

Your comments (documentation) should be in the same Python file. Please DO NOT write a separate file for comments (documentation).

At the beginning of your Python file, your code must contain the following information:

1. **Your name and student ID.**
2. **The highest level you have attempted.** This means you have completed all the requirements of the levels below. Mark will be only given at the lowest level of partial completion. For example, if you completed the PASS level, tried 50% of the CREDIT level, 30% of the DI level, 10% of the HD level, then your submission will be marked at the CREDIT level only (we will ignore the DI and HD levels, so please make sure you fully finish one level before moving to the next one).
3. **Any problems of your code and requirements that you have not met.** For example, scenarios that might cause the program to crash or behave abnormally, the requirements your program does not satisfy. Note that you do not need to handle or address errors that are not covered in the course.

Besides, the comments in this final coding challenge should serve the following purposes:

- Explain your code in a precise but succinct manner. It should include a brief analysis of your approaches instead of simply translating the Python code to English. For example, you can comment on why you introduce a particular function/method, why you choose to use a while loop instead of other loops, why you choose a particular data type to store the data information. These comments can be placed before the code blocks (e.g., functions/methods, loops, if) and important variable declarations that the comments refer to.
- Document some analysis/reflection as a part of your code. Here, you need to write some paragraphs (could be placed at the end or at the beginning of your code) to explain in detail your design process, e.g., how you came up with the design of the program, how you started writing the code after the design process, the challenges you met during the code development.
- Document the references, i.e., any sources of information (e.g., websites, tools) you used other than the course contents directly under Canvas/Modules, you must give acknowledgement of the sources, explaining how you use the sources in this assignment. More detailed information regarding the references can be found in Section 5.

D - Rubric:

Overall:

Level	Points
PASS level	15
CREDIT level	3
DI level	3
HD level	6
Others (code quality, modularity, comments/analysis)	3

More details of the rubric of this assessment can be found on Canvas ([here](#)). Students are required to look at the rubric to understand how the assessment will be graded.

4. Submission

As mentioned in the Code Requirements, **you must submit only one zip file with the name ProgFunFinal_<Your Student ID>.zip** via Canvas/Assignments/Final Coding Challenge. The zip file contains:

- The main Python code of your program, named **my_record.py**
- A diagram **in one of the formats: jpg, gif, png, pdf** (if you attempt the DI and HD levels)
- Other Python files written by you to be used by your main application.

It is your responsibility to correctly submit your file. Please verify that your submission is correctly submitted by downloading what you have submitted to see if the file includes the correct contents. The final zip file submitted is the one that will be marked. You do not need to submit the text files (result, course, student) we provide you with – we will mark based on our text files. **NOTE, your code must be able to run under command-line as in the specification. Specifically, it should be able to run under the following command lines:**

1. `python my_record.py`
2. `python my_record.py results.txt`
3. `python my_record.py results.txt datasets.txt`
4. `python my_record.py results.txt datasets.txt algorithms.txt`

If you attempt the PASS level, your program should be able to run the first and second command lines. If you attempt until the CREDIT level, your program should be able to run the first, second, and third command lines. If you attempt until the DI or HD level, your program should be able to run all the four command lines. **NOTE, the filenames specified in the command lines can be different, not necessarily results.txt, datasets.txt, algorithms.txt.** You can assume users always specify the correct order of file names, i.e., the result file first, the dataset file second, and the algorithm file third.

Late Submission

All assignments will be marked as if submitted on time. Late submissions of assignments without special consideration or extension will be automatically penalised at a rate of 10% of the total marks available per day (or part of a day) late. For example, if an assignment is worth 30 marks and it is submitted 1 day late, a penalty of 10% or 3 marks will apply. This will be deducted from the assessed mark. Assignments will not be accepted if more than five days late unless special consideration or an extension of time has been approved.

Special Consideration

If you are applying for extensions for your assessment within five working days after the original assessment date or due date has passed, or if you are seeking extension for more than seven days, you will have to apply for Special Consideration, unless there are special instructions on your Equitable Learning Plan.

In most cases you can apply for special consideration online [here](#). For more information on special consideration, visit the university website on special consideration [here](#).

5. Referencing Guidelines

What: This is an individual assignment, and all submitted contents must be your own. If you have used sources of information other than the contents directly under Canvas/Modules, **you must give acknowledgement of the sources, explaining in detail how you use the sources in this assignment, and give references using the IEEE referencing format.**

Where: You can add a code comment near the work (e.g., code block) to be referenced and include the detailed reference in the IEEE style.

How: To generate a valid IEEE style reference, please use the [citethisforme](#) tool if you're unfamiliar with this style.

6. Academic Integrity and Plagiarism (Standard Warning)

Academic integrity is about honest presentation of your academic work. It means acknowledging the work of others whilst developing your own insights, knowledge, and ideas. You should take extreme care that you have:

- Acknowledged words, data, diagrams, models, frameworks and/or ideas of others you have quoted (i.e., directly copied), summarized, paraphrased, discussed, or mentioned in your assessment through the appropriate referencing methods.
- Provided a reference list of the publication details so your readers can locate the source if necessary. This includes material taken from the internet sites.

If you do not acknowledge the sources of your material, you may be accused of plagiarism because you have passed off the work and ideas of another person without appropriate referencing, as if they were your own.

RMIT University treats plagiarism as a very serious offence constituting misconduct. Plagiarism covers a variety of inappropriate behaviors, including:

- Failure to properly document a source
- Copyright material from the internet or databases
- Collusion between students

For further information on our policies and procedures, please refer to the University website ([link](#)).

7. Assessment Declaration:

When you submit work electronically, you agree to the assessment declaration:

<https://www.rmit.edu.au/students/student-essentials/assessment-and-results/how-to-submit-your-assessments>