Programming Fundamentals (COSC2531)
Final Coding Challenge (Assignment 3)

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	**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 game program with the Object-Oriented Programming paradigm, that is based on recently popular Japanese anime virtual pet game, named **pymon.py**, that can read data from files and perform operations and simulate interactions with users. You are required to implement the program following the below requirements. We will provide a sample dataset for you to run your program as a reference. However, you are required to modify the data, as the provided sample does not include all the validation cases needed to fully test your program. During the marking, we will use different data/files to test the behavior of your program.

Game objective: Pymon is an imaginary creature that you can adopt as a pet. In the game map, there will be two kinds of creatures: Pymon and animal. Only a Pymon can be captured and adopted as a pet. Initially, you control a Pymon to explore a map of interconnected locations with the mission to find other creatures to defeat in a race. The goal is to be able to find all the creatures located in various rooms before your Pymon runs out of energy. Once your Pymon defeats a creature, the defeated creature is captured, added to your pet list, and can assist you in completing your mission. If your Pymon got defeated by another creature, one energy point will be deducted. Once you run out of all energies, then you will lose your Pymon and it will wildly move to random location. If you hold an extra Pymon, this will become active. If this was your only and last Pymon, then it is game over. You will win the game once there is no longer any capturable creature on the map.

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 stages**, please ensure you only attempt one stage after completing the previous stage.

-- STAGE 1 (5 marks) ------------------------------------

Your project is to implement the required functionalities in the Object-Oriented (OO) style with at least four classes: ***Locations***, ***Pymon, Record, and Operation***. 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.

The goal of this level is to first construct a map that contains multiple interconnected locations.

Class Location

Location represents a placeholder for your Pymon to interact with other creatures and items. It will be listed on a map and interconnected with each other through a door. Every location has all directions “west”, “north”, “east”, and “south”, however not every direction will lead you to another room. Your Pymon can move to another location by going in one of these directions. Every location will be connected to at least 1 other location adjacent to it. If the location is connected on the west side, then the adjacent location will be connected to it on the east side. The suggested class implementation is as follows:

Location should have 5 attributes:

- Name. Short name of the location.
- Description. Short description of the location.
- Doors. A dictionary of connected locations in each direction “west”, “north”, “east”, “south”. Initially all doors will be assigned None as its initial value. None simply means there is no location currently exists in that direction.
- List of creatures. List of creatures that resides in it. Initially there will be None.
- List of items. List of items that it holds. Initially there will be None.

Location should also have:

- A constructor
- Relevant getter and setter methods
- Connect method: This method takes two parameters: 'direction' and 'location'. Its purpose is to assign a 'location' to an available 'direction'. It will add the other location to the current location’s list of connected locations. At the same time, it will add the current location to the other location’s list of connected locations. Example use case of this method:
    - school.connect(“west”, playground) connects playground on school’s west side and connect school on playground’s east side.
    - Alternatively, you could have a connect method that takes in the location name for each direction, e.g. connect_west(“playground”)

After the map is done, another objective of this level is to instantiate the game’s objects and to perform basic operations to interact with the game character called Pymon. You will have a Pymon named ‘Toromon’, which is a white and yellow Pymon with a square face to begin with.

Class Creature

Creature can be found throughout the map, and it has 3 basic attributes:

- Nickname: Each creature will have a unique name.
- Description: Each creature will have a brief description such as “large blue and white Pymon with yellow fangs.”
- Location: This will contain object reference of its current location.

A normal Creature cannot be challenged or adopted.

Class Pymon

Pymon is an inheritance of Creature that has an additional attribute:

- Energy: Initially there are 3/3 energy points. There are factors that will increase and decrease energy.

- Speed: Determines the typical hopping capacity per second, measured in meter per second (mps).

Besides constructor, relevant setter, and getter methods, the Pymon will have a 'move' ability, as explained below. More abilities will be added at a later stage.

- Move: This will take in a ‘direction’ parameter argument. The acceptable keywords are “west”, “north”, “east”, and “south”. If there is a connected location in that direction, then the Pymon will move to new location, otherwise it will not do anything and stay in the current location.

Class Record

In Stage 1, you are required to import two input files: locations.csv and creatures.csv. Then, you need to load the data to create and connect the relevant locations and assign the creatures to their respective locations.

Class Operation

The object instantiated from Class Operation would provide a menu option that links with a relevant method of another object.

At this level, the program would have the following menu options to operate Pymon.

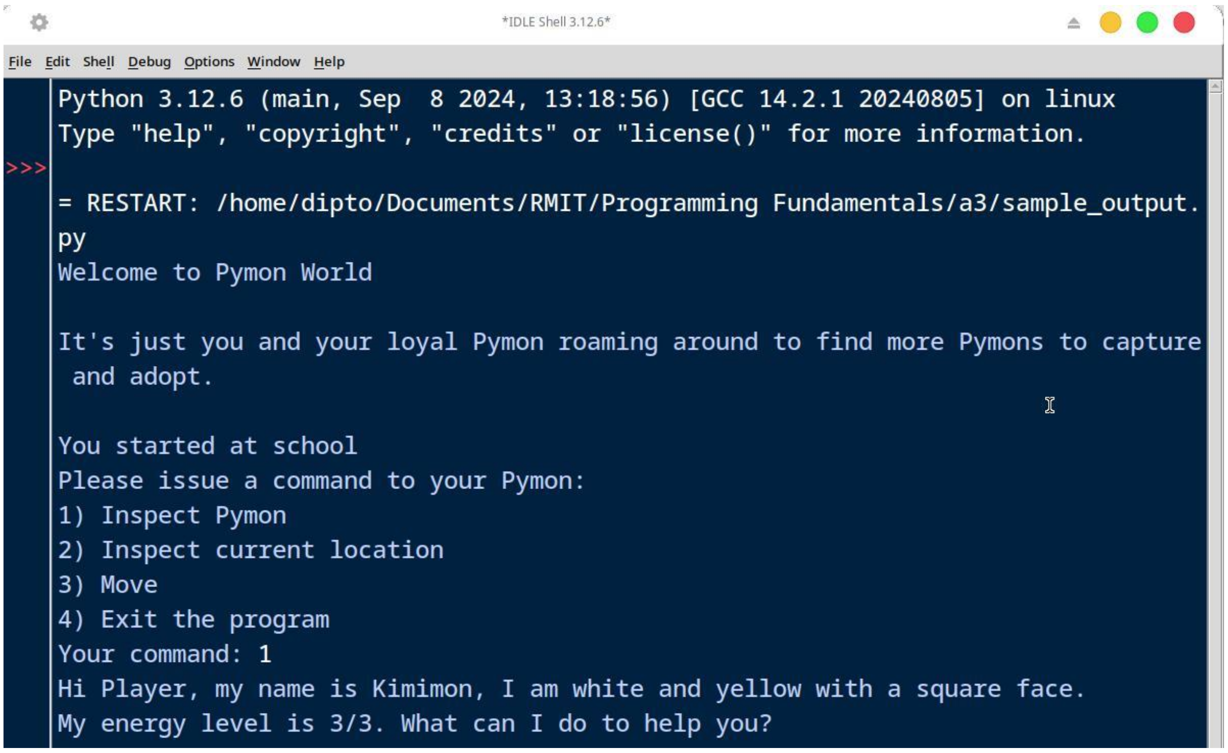
1. Inspect Pymon
   This is to review Pymon’s biodata, energy, and game statistics.
2. Inspect current location
   This provides details about your Pymon’s current location, including the location name, description, residing creatures, and available items. It only works when your Pymon is in that location. To view information about another location, you must first move your Pymon there.
3. Move
   This option commands Pymon to move from its current location to an adjacent location by entering a direction’s door namely “west”, “north”, “east”, and “south”. If there is no location connected through the door in that direction, then Pymon will remain in the current location.
4. Exit the program

Initial environment setup

At this level, the following initial setup will be required and tested.

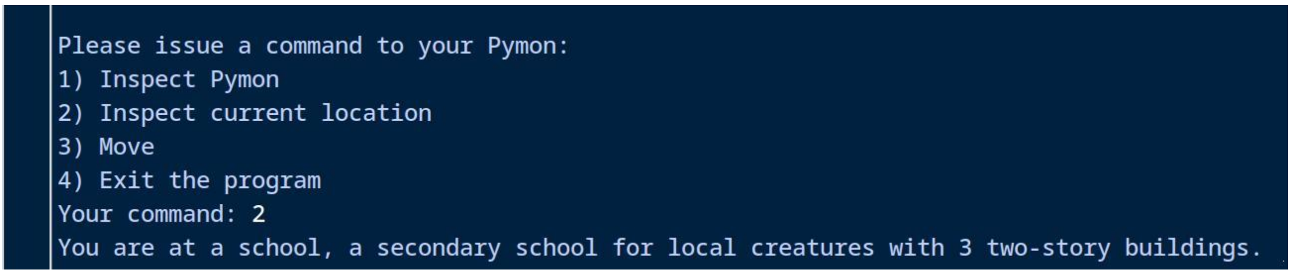
1. There will be 3 interconnected locations:
   a. Playground {west = School, north = Beach, east = None, south = None}
   b. Beach {west = None, north = None, east = None, south = Playground}
   c. School{west = None, north = None, east = Playground, south = None}
   There are more locations in the csv file, but at the P level you only need the above 3 locations which are manually connected by calling relevant connect methods.
2. Your Pymon will be spawned (placed) in the school initially.
3. The following creatures are spawned in the following locations:
   a. Playground contains “Kitimon”.
   b. Beach contains “Sheep”.
   c. School contains “Marimon”.
4. The rest of the creatures in the file will not be used at this stage.

When the game starts, the following output appears:



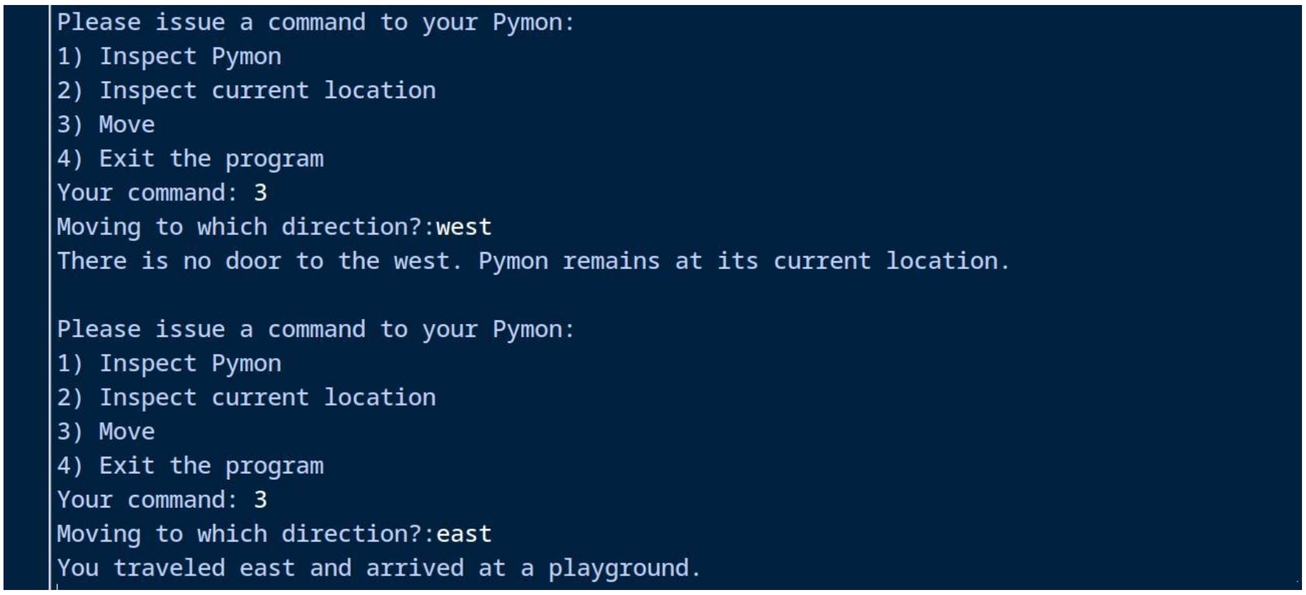
```
Python 3.12.6 (main, Sep  8 2024, 13:18:56) [GCC 14.2.1 20240805] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: /home/dipto/Documents/RMIT/Programming Fundamentals/a3/sample_output.
py
Welcome to Pymon World

It's just you and your loyal Pymon roaming around to find more Pymons to capture
 and adopt.

You started at school
Please issue a command to your Pymon:
1) Inspect Pymon
2) Inspect current location
3) Move
4) Exit the program
Your command: 1
Hi Player, my name is Kimimon, I am white and yellow with a square face.
My energy level is 3/3. What can I do to help you?
```

Sample output of the program at this level for inspecting current location is as follows:



```
Please issue a command to your Pymon:
1) Inspect Pymon
2) Inspect current location
3) Move
4) Exit the program
Your command: 2
You are at a school, a secondary school for local creatures with 3 two-story buildings.
```

Sample output of the program at this level for moving is as follows:



```
Please issue a command to your Pymon:
1) Inspect Pymon
2) Inspect current location
3) Move
4) Exit the program
Your command: 3
Moving to which direction?:west
There is no door to the west. Pymon remains at its current location.

Please issue a command to your Pymon:
1) Inspect Pymon
2) Inspect current location
3) Move
4) Exit the program
Your command: 3
Moving to which direction?:east
You traveled east and arrived at a playground.
```

---------------------- STAGE 2 (8 marks, you must only attempt this stage after completing the STAGE 1 ------------------------

At this level, you will implement 2 additional menu items:

1. Pick an item
   This option will command Pymon to attempt to pick up an item. Each location may contain items that Pymon can interact with. Some items can be picked and some cannot. Pymon can only pick an item if it is currently located in the same location as the item. When item is picked up, it will move from the location to Pymon inventory, that means the item will be removed from its original location.
2. View inventory
   This is to review items picked up along the journey. At later stage, each item in the inventory can be used in a certain way.

Item list

The following items may be found in a location. At the Credit level, there is only 1 item of each type.

1. Apple. Apple is Pymon's main food, which can be picked up and eaten to replenish Pymon's energy. Apple can only be used once.
2. Pogo stick. This can be picked up and when ridden, your Pymon will have extra speed in a race.
3. Binocular. With the binocular, Pymon can get an extra ability to quickly review of its surroundings including the location names in all directions. Additionally, a binocular also provides a foresight of a direction.
4. A tree. A tree does nothing and cannot be picked up. It is there for decoration purpose.

Initial Setup

Your game initial set up will now have the following items placed in the following locations:

1. Playground contains a 'tree' and a 'pogo stick'.
2. Beach contains 'apple'.
3. School contains 'binoculars'.

All locations should now be dynamically connected to each other as per the supplied *locations.csv* file. Each line in the file that contains *name, description, west, north, east, south* must be programmatically read and translated as an object that goes into a list of locations with proper connections.

Your Pymon starting location should be randomised from this level onwards.

Additionally, at this level, you will implement an additional menu item called "Challenge a creature", which sends the Pymon to enter a race against another Pymon that happened to be in the same location as yours. The Pymon that will reach the end first wins the race.

If your Pymon loses the race, its energy level will decrease by 1 point. Once your energy level reaches zero, you must relinquish your Pymon into the wild, and it will randomly move to another location. If you have another Pymon on your pet list, then this will become your current Pymon. All items in the inventory get transferred into your new Pymon's inventory. But if your one and only Pymon was taken away, then it is game over. If your Pymon wins the race against its opponent, you will capture the opposing Pymon and add it to your pet list.

To implement the above, Class Pymon needs to be modified to include the following additional method:

- Challenge to race: This ability takes a 'creature_name' parameter as an argument. If another Pymon is present in the location, your Pymon can challenge its opponent to race. While Pymon can be challenged, other types of creatures cannot.

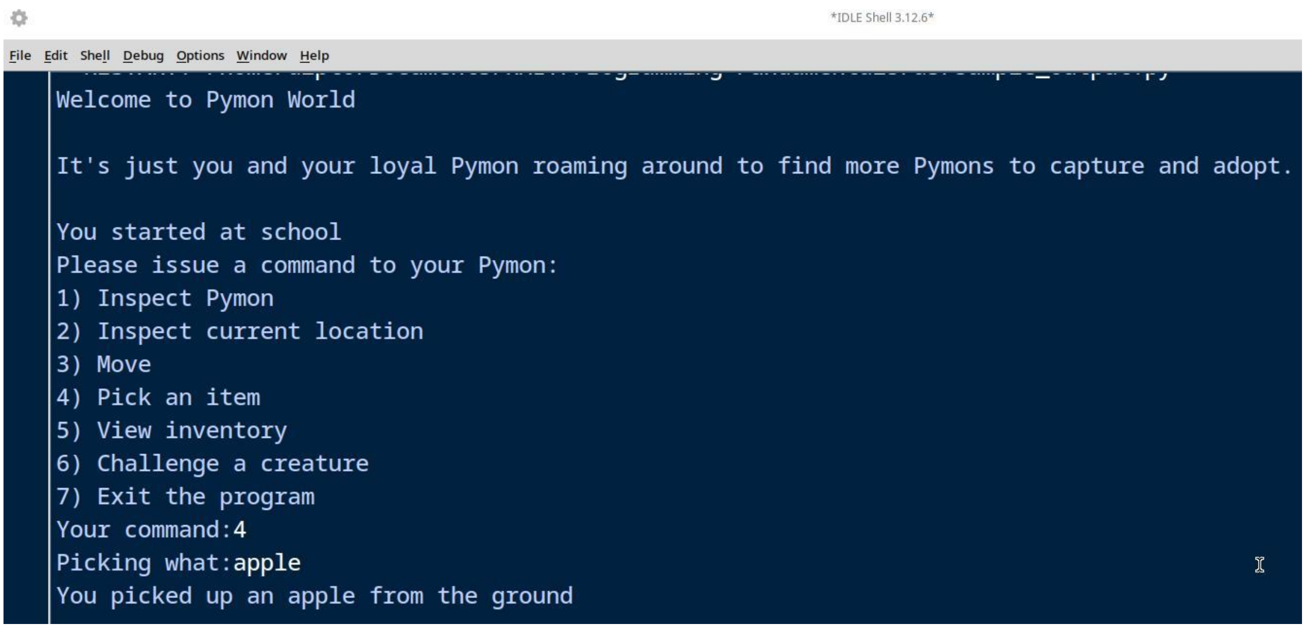
The menu will now look as follows:

1. **Inspect Pymon** (this menu item now has 2 sub items)
    1. ***Inspect current Pymon:*** View your current Pymon's name, description, and energy.

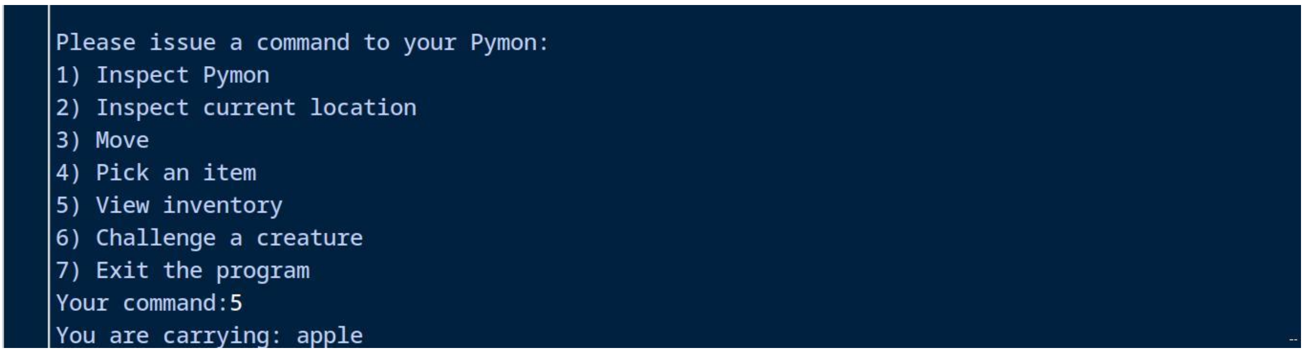
   2. ***List and select a benched Pymon to use.*** If another Pymon is available in the bench you can swap it with your current one. Items in the current inventory will stay with the Pymon who picked them up, until you lose this Pymon in a battle.

2. Inspect current location
3. Move
4. **Pick an item**
5. View inventory
6. **Challenge a creature:** This will send your Pymon into a **race mode** against another Pymon in the current location. If there is no other Pymon in the location, then the challenge is invalid and does not affect anything. Any attempt to challenge a creature other than a Pymon will result in humorous responses such as "the sheep just ignored you", "the chicken just laughed at you" or "the tree stands still".
7. Exit the program

Sample output of the program at this level for picking up an item is as follows:



```
Welcome to Pymon World

It's just you and your loyal Pymon roaming around to find more Pymons to capture and adopt.

You started at school
Please issue a command to your Pymon:
1) Inspect Pymon
2) Inspect current location
3) Move
4) Pick an item
5) View inventory
6) Challenge a creature
7) Exit the program
Your command:4
Picking what:apple
You picked up an apple from the ground
```



```
Please issue a command to your Pymon:
1) Inspect Pymon
2) Inspect current location
3) Move
4) Pick an item
5) View inventory
6) Challenge a creature
7) Exit the program
Your command:5
You are carrying: apple
```

The Race Mode

There could be 3 outcomes in the race:
Win: If your Pymon reaches the finish line before your opponent, then you win.
Lose: If your Pymon reaches the finish line after your opponent, then you lose.
Draw: In rare cases, both racers could reach the finish line at the exact same time, then it is a draw.

The race mode will simulate 2 Pymons race against each other to reach a 100-meter finish line by hopping forward. Speed is the main factor that will determine the winner of each round, measured in meters per second (mps). The speed is a typical hop gain should be treated as the base assumption of how fast can a Pymon move, in other words how far can a Pymon hop forward per second. Throughout the race, Pymon's speed would go up and down and is largely determined by a luck factor.

The following formula illustrates how luck factor would impact actual speed at a particular second:

*Actual speed at a particular second = typical speed +/- (randomised luck factor% * typical speed)*

where

Minimum luck factor: 20

Maximum luck factor: 50

An unlucky second

When a Pymon gets bad luck, its speed can reduce by a factor of 20-50% of its original speed in a particular second. For example, if this Pymon has 5mps and its luck factor is negative 20%, then its actual speed for that second is 4mps.

A lucky second

When a Pymon gets good luck, its speed can improve by a factor of 20-50% of its original speed in a particular second. For example, if this Python has 5mps and its luck factor is positive 30%, then its actual speed for that second is 6.5 mps.

To simulate the race, you can implement the following output:

```
Toromon (your Pymon) hopped 2.00 meters. Distance remaining for Toromon: 99.00
meters
Kitimon (opponent) hopped 10.00 meters. Distance remaining for Racer B: 90.00 meters
Toromon (your Pymon) hopped 2.00 meters. Distance remaining for Racer A: 98.00 meters
Kitimon (opponent) hopped 6.00 meters. Distance remaining for Racer B: 84.00 meters
Toromon (your Pymon) hopped 6.00 meters. Distance remaining for Racer A: 92.00 meters
Kitimon (opponent) hopped 7.00 meters. Distance remaining for Racer B: 77.00 meters
....
Kitimon (opponent) reached the finish line in 10 seconds! You lose!
```

Every second simulated is an actual second in real life. You should use the sleep() function from Python’s time module to create a delay between printing of each line. You may also use this function to create delays elsewhere to enhance your game flow.

-------------- STAGE 3 (5 marks, you must only attempt this level after completing the STAGE 2) --------------

This level further extends your program with extra features and complexities.

Pymon energy level can go up and down.

1. Energy level increases by one point if Pymon eats an edible item.
2. Energy level decreases by one point in every 2 moves. When Pymon moves 2 locations, its energy will decrease by 1 point. Once all energy points were depleted, your Pymon will escape into the wild and move to another random room. If this this was your one and only Pymon, then it is game over.

At this level Pymon can use items in the inventory to help its mission and maintain its energy level. Make the following modification in Class Pymon:

- Use item: This will take in an 'item' parameter argument to use an item in the inventory. An item must first be collected and added in inventory before it can be used.
  - If there is an edible item like an 'apple', then Pymon can eat it. Each item will increase energy by one point. Maximum energy is 3. If current energy level is already 3, then eating will not further increase energy.
  - A 'pogo stick' will give a temporary, unfair advantage in a race.
    - When pogo stick picked up it will be stored in your Pymon's inventory.
    - You must first use the pogo stick before entering a race challenge for it to affect the race.
    - When pogo stick has been applied, the actual speed after calculating the luck factor is multiplied by 2 in every second of the race.
    - For example:
      - Without a pogo, the speed will be normal: actual speed = typical speed +/- luck factor% * typical speed
      - With a pogo, the speed will be doubled: actual speed = (typical speed +/- luck factor % * typical speed) x 2
    - The pogo stick can only be used in one race. Regardless of the race outcome, after the race, the pogo will break and disappear from your inventory and will not be reused.
  - A 'binocular' will give Pymon farsighted vision of the current location and other connected location in a particular direction. When using binocular, Pymon can inspect the content of a particular location without travelling to that location by entering a location keyword. Here are the examples of output when a particular location keyword is entered.
    - "current". Example output "A sheep, another Pymon, and in the west is a cave".
    - "west". Example output "In the west, there seems to be a cave with an edible apple and another unknown creature nearby".
    - "north". Example output "This direction leads nowhere"
    - "east". Example output "This direction leads nowhere".
    - "south". Example output "This direction leads nowhere"

The menu will now look as follows:

1. Inspect Pymon
2. Inspect current location
3. Move
4. Pick an item

5. **View inventory** (this function now has an additional sub-command)
   a. ***Select item to use***
6. Challenge a creature
7. Generate stats
8. Exit the program

Additionally, your program should now be able to handle some variations in the files using built-in/custom exceptions:

1. InvalidDirectionException: This exception gets thrown when the selected direction does not contain any location.
2. InvalidInputFileFormat: This exception gets thrown when a CSV file has invalid content or in incorrect format.

-------------- STAGE 4 (5 marks, you must only attempt this level after completing the STAGE 3) ---------------

The menu will now look as follows:

1. Inspect Pymon
2. Inspect current location
3. Move
4. Pick an item
5. View inventory
6. Challenge a creature
7. **Generate stats**
8. Exit the program

The program now will have a function to display the save race stats in a file called race_stats.txt

1. Every race will be documented in a stats report which includes include number of wins, losses and draws, of every Pymon that you had control of. At the bottom of the battle list, it will show total wins, total draws and total loss. The following stats report will be displayed on the terminal screen and saved in race_stats.txt

   Pymon Nickname: "Toromon"

   Race 1, 25/10/2025 11:22AM Opponent: "Gumimon", Win

   Race 2, 26/10/2025 10:30PM Opponent: "Pumamon", Lose

   Total: W: 1 L: 1 D: 0

There are 3 data files that will provide the starting materials to run the game: creatures.csv, locations.csv and items.csv. Data from these 3 should be added to corresponding lists in a Record object. As the program gets more complex, it allows for custom files input to provide extension and extra variations to the game. For this, you may add an additional Record class to manage file input and output. NOTE, your code must be able to run under command-line as in the specification. Specifically, if the filenames of the locations, creatures, and items are locations.csv, creatures.csv, and items.txt, respectively, then your program should be able to run under the following command lines:

1. *python pymon_game.py*
2. *python pymon_game.py locations.csv*
3. *python pymon_game.py locations.csv creatures.csv*
4. *python pymon_game.py locations.csv creatures.csv items.csv*

After data is imported into the program, the program will place items and creatures at various locations randomly. At this stage, there could be more than one item in various locations. Your Pymon starting location will also be randomised.

The program will have some additional requirements:

1. Save game progress: This will save state of the game into a custom file like gamesave2025.csv
2. Load game progress: This will load state of the game from a custom file like gamesave2025.csv
3. Admin feature to add a custom location. This is to create a new location object and save it in locations.csv

   Sample csv line that represents a location is as follows:

School, a secondary school for local creatures with 3 two-story buildings., None, None, Playground, None

4. Admin feature to add a custom creature. This is to create a new creature object (could be both Pymon and other type of creature) and save it in creatures.csv. You may decide to have an inheritance structure to categorise a Creature into 2 categories: Pymon and Animals.
   Sample csv may contain *name*, *description*, and *adoptable* (yes/no) such as the following:
   Kitimon, large blue and white Pymon with yellow fangs, yes
   Sheep, small fluffy animal with interesting curly white fur, no
   Marimon, medium red and yellow Pymon with a cute round face, yes
5. Admin feature to randomise connections between locations. Once imported, the location now will ignore the connected locations defined in the CSV file. Instead, each side of the location will now be assigned a random location to increase the game's unpredictability. Similar to the initial setup not all sides would have a connecting door, i.e. will be assigned None.

B - Code Requirements:

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

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.

C - Documentation Requirements:

You are required to write comments (documentation) as 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 stage you have attempted.** This means you have completed all the requirements of the stages below. Mark will be only given at the lowest stage of partial completion. For example, if you completed the Stage 1, tried 50% of Stage 2, 30% of the Stage 3, 10% of the Stage 4, then your submission will be marked at the Stage 2 only (we will ignore the Stage 3 and 4, so please make sure you fully finish one stage 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.
4. **Reference list.**

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 the references, i.e., any sources of information (e.g., websites) 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 – Program Demo and Presentation Video

You need to create a program demo and presentation video. The recording should be done using MS Stream https://web.microsoftstream.com/ and its duration should not exceed 10 minutes. On MS Stream, your video should be created in "Screen Recording" mode that clearly shows you presenting with an open microphone and camera on your face, your desktop view that contains your game program running on terminal and the Python IDE that contains your code. The complete video should be downloaded and saved in .webm format and submitted in the same zip file as your Python code.

In your video you should:

- Demonstrate how your program runs from start to finish. You need to run all functionalities that you have developed.
- Present some analysis/reflection as a part of your code. Here, you need to talk 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.
- Highlight 1 part of the code that you are proudest of, e.g. a complex function or algorithm that you were able to achieve.
- **You are allowed to use Generative Artificial Intelligence tools (e.g., ChatGPT or Copilot) to help you when developing your code, however this must be fully acknowledged in your code documentation and in your presentation video, explaining:**
    - What prompt exactly that you use
    - Summary of the output or recommendation from the GenAI
    - Which part of the work was impacted or influenced by GenAI, e.g. any code segment, any method, algorithm that you have used GenAI recommendation on.

E - Rubric:

Overall:

Level	Points
Stage 1	5
Stage 2	8
Stage 3	5
Stage 4	5
Others: code quality, modularity, comments/analysis	2
Program demo and presentation video	5

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 **pymon_game_<Your Student ID>.py**
- Other Python files written by you to be used by your main application.
- All data files you have used as input, .csv files.
- 10-minute video file named **demo_<Your Student ID>.webm**

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 content. The final zip file submitted is the one that will be marked.

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 the 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 content 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 of 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