

## **Assignment3**

### *Contents*

Learning Outcomes and Objectives .....	2
Learning Outcomes .....	2
Assignment Learning Objective .....	2
Assignment Graduation Attributes Indicators .....	3
Instructions: .....	4
Submission Instructions .....	4
Assignment Structure and Milestones.....	5
Milestone Breakdown.....	5
Submission Requirements .....	5
Part One: Milestone1 Hill Climbing, and Q1 (Reflex agent) .....	6
Milestone1 Deliverables and Group Work .....	10
The report should include:.....	10
Part Two: Milestone2 Q2 (minimax) and Q3 (alpha-beta) .....	11
Milestone2 Deliverables and Group Work .....	14
The report should include:.....	14
Part Three: Milestone3 Q4 (Expectimax).....	15
Milestone3 Deliverables and Group Work .....	17
The report should include:.....	17

### Assignment3

## Learning Outcomes and Objectives

This assignment strongly supports the course learning outcomes by engaging students in the design, implementation, and analysis of multiple AI search and decision-making algorithms within a controlled toy problem environment. Through **hill-climbing**, **minimax**, **alpha-beta pruning**, and **expectimax** agents, students apply a variety of search techniques to solve artificial intelligence problems, directly addressing **CLO1**. By implementing these algorithms in Python and integrating them into the Pacman framework, students achieve **CLO5** through hands-on programming experience. The **expectimax** component introduces probabilistic reasoning by modelling non-adversarial, stochastic ghost behaviour, supporting **CLO3**. Across all parts, students compare the strengths, weaknesses, and applicability of different AI approaches—local search, adversarial search, and probabilistic decision-making—based on problem characteristics and assumptions, fulfilling **CLO4**. Collectively, the assignment emphasizes not only correct implementation but also critical reflection on algorithm behaviour, limitations, and suitability, reinforcing core AI problem-solving skills.

### Learning Outcomes

- **CLO1:** *Apply various search algorithms to find solutions to AI toy problems*
- **CLO3:** *Apply Probabilistic reasoning to handle uncertainty*
- **CLO4:** *Compare the applicability of AI methods based on the type of problem being solved*
- **CLO5:** *Implement AI methods using one of AI languages to solve Artificial Intelligence problems.*

### Assignment Learning Objective

The learning objectives that can be achieved by solving the assignment tasks are as follows:

- 1) Apply local and adversarial search algorithms to solve AI toy problems.
- 2) Implement multi-agent search methods (**Hill-Climbing**, **Minimax**, **Alpha-Beta**, **Expectimax**) in Python.
- 3) Analyze the strengths and limitations of different AI search strategies.
- 4) Apply probabilistic reasoning to decision-making under uncertainty.
- 5) Compare AI methods based on problem characteristics and assumptions.
- 6) Evaluate algorithm behaviour using experimental results and reflections.

## **Assignment3**

### Assignment Graduation Attributes Indicators

This assignment supports multiple graduation attributes by engaging students in the application, implementation, and analysis of artificial intelligence search algorithms within a structured experimental environment. Students apply foundational concepts from specialized engineering sciences by implementing and reasoning about local search, adversarial search, and probabilistic decision-making algorithms, directly supporting **KB.4**. Through problem formulation, model development, and algorithm design for complex multi-agent environments, students address **PA.1** and **PA.2**, while systematic evaluation of algorithm behavior across different scenarios supports **PA.3**. The requirement to execute algorithms from multiple initial states, collect results, and analyze outcomes constitutes planned investigative activity aligned with **IN.1**, while interpreting these results and reflecting on algorithm performance supports **IN.2**. Throughout the assignment, students use Python, automated testing tools, and a simulation-based framework, demonstrating effective use of modern engineering tools and techniques in alignment with **ET.2**.

- KB.4 - Comprehend and apply first principles and concepts in specialized engineering sciences
- PA. 1 - Identify and formulate complex engineering problems
- PA. 2 - Develop models from first principles to solve complex engineering problems
- PA. 3 - Analyze complex engineering problems
- IN. 1 - Conduct planned activities (literature review, experiments, measurements, laboratories, etc.)
- IN. 2 - Interpret results and reaches valid conclusions regarding complex problems
- ET. 2 - Use appropriate techniques, resources and modern engineering tools in engineering activities

## **Assignment3**

### Instructions:

- The assignment can be completed **in groups (maximum of three members per group, recommended)**.
- All group members should work together, and they will receive the same mark.
  - Any partial grade will be given case-by-case.
- This Assignment is worth 6% of the total course grade, and it will be graded out of 100 marks and evaluated through your written submission, as well as the lab demo, as follows:
  - 100 marks (6% of the total course grade)
    - 60%: Blackboard submission
    - 40%: Lab demo during the lab session
- Please submit the submission file(s) through Blackboard. **Only one person from the group needs to submit; only the last submission will be graded.**
- During the lab demo, group members are **randomly** selected to explain the submitted solution.
- **Group members who do not present during the lab demo will lose the demo mark.**
- You must submit all your answers and screenshots inside pdf file.
- You must submit all of your Python code files inside an archive file (zip).

### Submission Instructions

1. **Create a New Document**
  - Open Microsoft Word (or an equivalent word processor) and **create a new blank document**.
  - **Do NOT use the original assignment instructions as a template for your answers. Your submission must be your own, organized work.**
2. **Write Your Answers in the New Document**
  - Clearly label each step or question and provide your answers in a structured, professional manner.
  - Use proper headings, subheadings, and numbering to keep your solution organized.
  - Ensure your writing reflects **professional engineering communication skills**—be concise, clear, and technically accurate.
3. **Do NOT Include Assignment Instructions**
  - The document you submit should **only contain your answers**. To the questions, steps or tasks in a very organized way.
    - Such as question1/Task1: (write the question/task text)
    - Then: Answer to Question1/Task1: (write the question/task answer)
  - ***Including the original instructions in your submission will result in a penalty.***
4. **Convert to PDF Before Submission**
  - After completing your answers in Word, **convert the document to PDF format**.
  - Verify that the PDF is complete, properly formatted, and readable before submitting.
5. **Final Check Before Submission**
  - Ensure your document is:
    - Well-organized and free of spelling/grammar errors.
    - Professionally formatted (consistent fonts, headings, and spacing).
    - Contains **only your solutions**, not the original instructions.

## Assignment3

# Assignment Structure and Milestones

Assignment3 is worth **6%** of your final course grade. This assignment is divided into two milestones to help you manage your workload effectively and complete the assignment.

## Milestone Breakdown

- **Milestone 1:** 2% of the final grade
- **Milestone 2:** 2% of the final grade
- **Milestone 3:** 2% of the final grade

Each milestone is graded **independently** and must be completed, demonstrated, and submitted separately. Also, it will be graded out of 100 and will follow the 60% and 40% instructions above. You **must complete Milestone 1 before moving on to Milestone 2**. And so on.

## Submission Requirements

There are three separate submission links (boxes) on Blackboard:

- One link/box for Milestone 1
- One link/box for Milestone 2
- One link/box for Milestone 3

You must submit each milestone to its corresponding box. **Submitting work to the wrong box may result in your work not being graded.**

For **each milestone**, you must follow this process:

1. **Complete the milestone**
2. **Demonstrate the milestone during your lab session**
3. **Submit the milestone on Blackboard to the correct link/box**

You must **fully complete and demonstrate Milestone 1 before submitting it**, and only then proceed to Milestone 2. And so on.

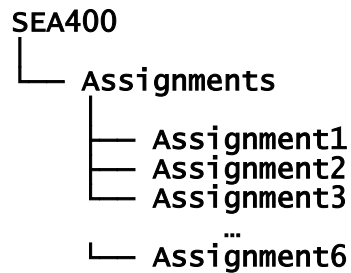
**Note:** Think of this assignment as one assignment divided into three smaller assignments. The milestone structure is intentional and is meant to:

- Help you start early
- Divide the work into manageable parts
- Improve your understanding and overall performance
- Do not leave both milestones until the last minute.

## Assignment3

# Part One: Milestone1 Hill Climbing, and Q1 (Reflex agent)

**Step 1.** At this stage, to stay organized and avoid confusion when working on assignments, it is important to create a dedicated workspace on your computer. Such as



You are given a 2D array representing the heights of hills in a landscape:



**Step 2.** What are the global and local **maxima**?

**Step 3.** Looking at the elevation map, what are some issues you would expect for a hill-climbing algorithm?

## Assignment3

**Step 4. Implement** a hill-climbing algorithm to find the highest peak starting from a given initial position. The algorithm should move to the neighbouring cell with the highest height until it reaches a peak (a cell with a height higher than its neighbours).

- i. Implement a function **hill\_climbing\_2d(heights, start\_pos)** that takes a 2D list of integers heights and a tuple **start\_pos** as input.
- ii. The function should return the position and height of the highest peak reached.
- iii. Assume that the landscape is represented as a 2D list of integers where each integer represents the height of a hill.
- iv. The function should move to the neighbouring cell with the highest height until it reaches a peak. If two neighbours have the same height, choose the first given this order of moves (**left, right, up**, and **down**). **Allow for 2 sideways moves.**

A **sideways move** is when you move to a neighbour with the **same height** as your current cell (neither higher nor lower). So you should still prefer moving to a **higher** neighbour whenever one exists but if no higher neighbour exists and there *is an equal-height neighbour*, you are allowed to move sideways **up to 2 times total** (across the whole run) After you have used **2 sideways moves**, you can not move sideways anymore: if there is no higher neighbour, you stop (you are effectively at a peak/stuck (shoulder)).

**Step 5. Trace a manual run before coding:** you must manually simulate the algorithm for the following start states.

- a. **A(3,6)**
- b. **B(6,2)**
- c. **C(8,7)**

Write the path as a sequence:

$A(3, 6) \rightarrow (r1, c1) \rightarrow (r2, c2) \rightarrow \dots$

$B(6, 2) \rightarrow (r1, c1) \rightarrow (r2, c2) \rightarrow \dots$

$C(8, 7) \rightarrow (r1, c1) \rightarrow (r2, c2) \rightarrow \dots$

**Step 6.** What is the solution found by a hill-climbing algorithm, assuming the following initial states:

- a. **A(3,6)**
- b. **B(6,2)**
- c. **C(8,7)**

Compare the generated solution (i.e., the path) to the manual solution and **take screenshots of the generated path for each of the initial states listed above.**

**Step 7.** Analyze the solutions and explain them. Are these what you expected?

**Step 8.** What are the issues faced by this algorithm? How can they be resolved?

**Step 9.** Why is the tie-breaking order necessary? What goes wrong if it's not deterministic?

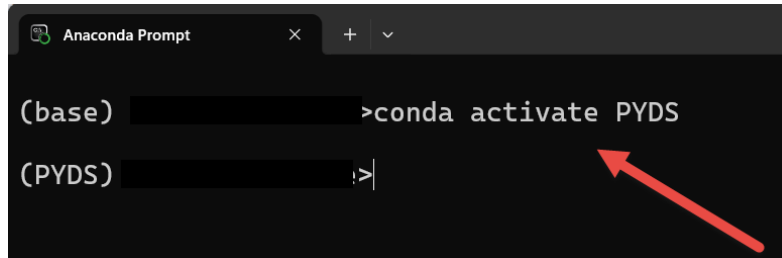
## Assignment3

**Step 10.** Download [multiagent.zip](#) from this [link](#)

- Move the [multiagent.zip](#) file to the directory you designated for assignment3 (see above)
- Unzip/extract the [multiagent.zip](#) file inside the assignment directory.

**Step 11.** At this point, you have two options for running the commands provided in the tutorial:

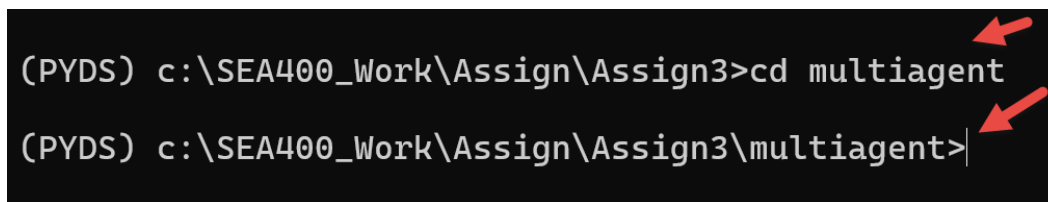
- Local Environment:** Use your own computer's terminal (Command Prompt, PowerShell, or Terminal on macOS/Linux).
- Google Colab:** If you prefer a cloud-based solution, you can use Google Colab.



```
Anaconda Prompt
(base) >conda activate PYDS
(PYDS) :>
```

Figure 1

**Step 12.** Open your terminal window and navigate to the [multiagent](#) directory



```
(PYDS) c:\SEA400_Work\Assign\Assign3>cd multiagent
(PYDS) c:\SEA400_Work\Assign\Assign3\multiagent>
```

Figure 2

**Step 13.** Follow the instructions to solve and test Question1 from the [link](#) (Project 2).

For each question, it is recommended to develop and write your algorithm (pseudo-code) that you plan to implement.

**Step 14.** After you write your pseudo-code on a piece of paper, it is time to implement your algorithm and add comments to each step (or add sufficient comments to your code).

**Step 15.** Improve the [ReflexAgent](#) in [multiAgents.py](#) to play respectably. The provided reflex agent code provides some helpful examples of methods that query the [GameState](#) for information. A capable reflex agent will have to consider both food locations and ghost locations to perform well

### **Assignment3**

**Step 16.** We will run your agent 10 times on the **openClassic** layout. You will receive 0 points if your agent times out or fails to win. You will receive 1 point if your agent wins at least 5 times, or 2 points if your agent wins all 10 games. You will receive an additional 1 point if your agent's average score is greater than 500, or 2 points if it is greater than 1000. You must run the command below to see if your implementation passes all the **autograder** conditions/test cases.

```
python autograder.py -q q1
```

To run it without graphics, use:

```
python autograder.py -q q1 --no-graphics
```

**Step 17.** If you do want to see any warning messages the use the following command

Windows:

```
python -w "ignore::SyntaxWarning" autograder.py -q q1
```

Linux (Terminal):

```
python -w "ignore::SyntaxWarning" autograder.py -q q1
```

**Step 18.** You are required to take screenshots to demonstrate the successful execution of your code for the above steps. These screenshots serve as proof of completion and proper identification.

**Step 19.** You need to submit your **multiAgents.py** file, which contains your implementation for Q1.

**Step 20.** In your own words, explain why a reflex agent evaluates **state-action pairs** instead of just states. How does this affect its short-term behaviour?

**Step 21.** Identify one situation where your agent makes a locally "reasonable" move that later leads to failure. What feature caused this decision?

## Assignment3

### Milestone1 Deliverables and Group Work

Create assignment report with the following name format

**group\_<number>\_assign\_<assignment number>\_MS1\_report.pdf**

For example, if **group26** created a report for **Assignment20**, then the report name should be

**group\_26\_assign\_20\_MS1\_report.pdf**

The report should include:

(a) Complete this declaration by adding your names:

We, ----- (mention your names), declare that the attached assignment is our own work in accordance with the Seneca Academic Policy. We have not copied any part of this assignment, manually or electronically, from any other source, including websites, unless specified as references. We have not distributed our work to other students.

(b) Specify what each member has done towards the completion of this work:

	Name	Task(s)
1		
2		
3		

(c) Read the assignment steps and questions carefully. If the assignment questions (or part of the question) are asked for output or response, then you should include the output images inside an assignment report as a PDF file and write a response to answer some of the assignment questions (or part of the question).

(d) As part of your assignment submission, **you are required to take screenshots to demonstrate the successful execution of your code**. These screenshots serve as proof of completion and proper identification.

- Use any document editor (e.g., Word, Google Docs) to compile all your screenshots into a single document.
- Make sure you follow the assignment instructions for the file naming convention

(e) Submit file (s)

- group\_26\_assign\_20\_MS1\_report.pdf**
- Python Source File (py):** Submit **one Python file** that contains your complete implementation of the hill-climbing algorithm
- multiAgents.py**

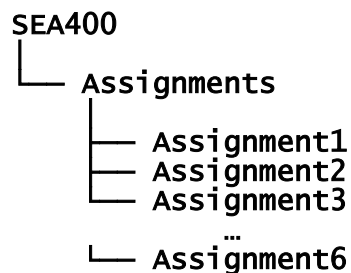
## Assignment3

# Part Two: Milestone2 Q2 (minimax) and Q3 (alpha-beta)

In Milestone2 of this assignment, you will implement **uninformed** and **heuristic search** algorithms. You will be using code from

- Project 2: <https://inst.eecs.berkeley.edu/~cs188/fa25/projects/proj2/>

**Step 22.** At this stage, to stay organized and avoid confusion when working on assignments, it is important to create a dedicated workspace on your computer. Such as



**Step 23.** Download **multiagent.zip** from this [link](#)

- Move the **multiagent.zip** file to the directory you designated for assignment3 (see above)
- Unzip/extract the **multiagent.zip** file inside the assignment directory.

**Step 24.** At this point, you have two options for running the commands provided in the tutorial:

- Local Environment:** Use your own computer's terminal (Command Prompt, PowerShell, or Terminal on macOS/Linux).
- Google Colab:** If you prefer a cloud-based solution, you can use Google Colab.

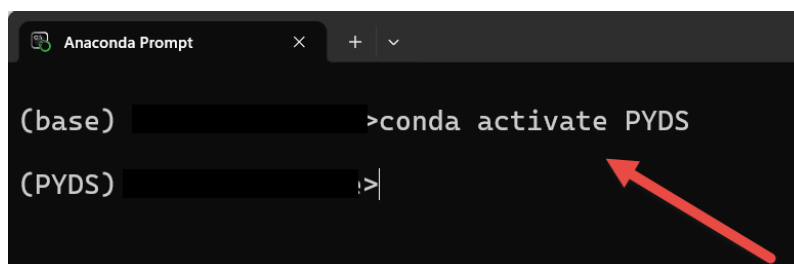
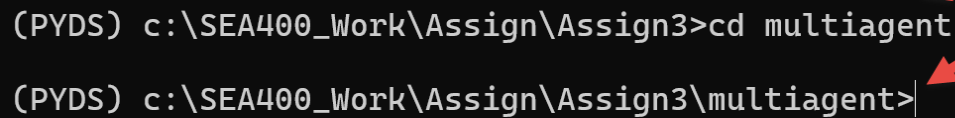


Figure 3

## Assignment3

**Step 25.** Open your terminal window and navigate to the **multiagent** directory

A terminal window with a black background and white text. The first line shows the command '(PYDS) c:\SEA400\_Work\Assign\Assign3>cd multiagent' with a red arrow pointing to the 'multiagent' directory name. The second line shows the prompt '(PYDS) c:\SEA400\_Work\Assign\Assign3\multiagent>' with a red arrow pointing to the prompt.

```
(PYDS) c:\SEA400_Work\Assign\Assign3>cd multiagent
(PYDS) c:\SEA400_Work\Assign\Assign3\multiagent>
```

Figure 4

**Step 26.** Follow the instructions to solve and test Question2 and Question3 from the [link](#) (Project 2).

For each question, it is recommended to develop and write your algorithm (pseudo-code) that you plan to implement.

**Step 27.** After you write your pseudo-code on a piece of paper, it is time to implement your algorithm and add comments to each step (or add sufficient comments to your code).

**Step 28.** You will write an adversarial search agent in the provided **MinimaxAgent** class stub in **multiAgents.py**. Your minimax agent should work with any number of ghosts.

**Step 29.** We will be checking your code to determine whether it explores the correct number of game states. You must run the command below to see if your implementation passes all the **autograder** test cases.

```
python autograder.py -q q2
```

To run it without graphics, use:

```
python autograder.py -q q2 --no-graphics
```

**Step 30.** If you do want to see any warning messages the use the following command

Windows:

```
python -w "ignore::SyntaxWarning" autograder.py -q q2
```

Linux (Terminal):

```
python -w "ignore::SyntaxWarning" autograder.py -q q2
```

## Assignment3

**Step 31.** You will write an adversarial search agent that uses alpha-beta pruning in the provided `AlphaBetaAgent` class stub in `multiAgents.py`.

**Step 32.** We will review your code to determine whether it has been implemented correctly. You must run the command below to see if your implementation passes all the **autograder** test cases.

```
python autograder.py -q q3
```

To run it without graphics, use:

```
python autograder.py -q q3 --no-graphics
```

**Step 33.** If you do want to see any warning messages the use the following command

Windows:

```
python -w "ignore::SyntaxWarning" autograder.py -q q3
```

Linux (Terminal):

```
python -w "ignore::SyntaxWarning" autograder.py -q q3
```

**Step 34.** You are required to take screenshots to demonstrate the successful execution of your code for the above steps. These screenshots serve as proof of completion and proper identification.

**Step 35.** You need to submit your `multiAgents.py` file, which contains your implementation for Q2 and Q3.

**Step 36.** What does it mean that there are “**multiple min layers**” for every max layer? Draw or describe the layer order for **2 ghosts** and  $\text{depth} = 1$ .

**Step 37.** In your own words, what problem does alpha–beta pruning solve compared to plain minimax?

**Step 38.** What do  $\alpha$  (**alpha**) and  $\beta$  (**beta**) represent intuitively during search? Describe them using Pacman vs ghost roles.

## Assignment3

### Milestone2 Deliverables and Group Work

Create assignment report with the following name format

**group\_<number>\_assign\_<assignment number>\_MS2\_report.pdf**

For example, if **group26** created a report for **Assignment20**, then the report name should be

**group\_26\_assign\_20\_MS2\_report.pdf**

The report should include:

(a) Complete this declaration by adding your names:

We, ----- (mention your names), declare that the attached assignment is our own work in accordance with the Seneca Academic Policy. We have not copied any part of this assignment, manually or electronically, from any other source, including websites, unless specified as references. We have not distributed our work to other students.

(b) Specify what each member has done towards the completion of this work:

	Name	Task(s)
1		
2		
3		

(c) Read the assignment steps and questions carefully. If the assignment questions (or part of the question) are asked for output or response, then you should include the output images inside an assignment report as a PDF file and write a response to answer some of the assignment questions (or part of the question).

(d) As part of your assignment submission, **you are required to take screenshots to demonstrate the successful execution of your code**. These screenshots serve as proof of completion and proper identification.

- Use any document editor (e.g., Word, Google Docs) to compile all your screenshots into a single document.
- Make sure you follow the assignment instructions for the file naming convention

(e) Submit file (s)

- group\_26\_assign\_20\_MS2\_report.pdf**
- multiAgents.py**

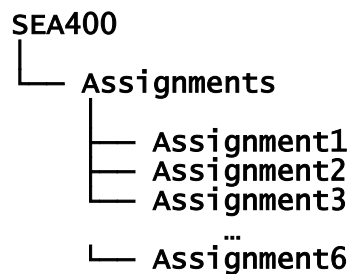
## Assignment3

# Part Three: Milestone3 Q4 (Expectimax)

In Milestone 3, you will implement the ExpectimaxAgent, which is useful for modelling the probabilistic behaviour of agents that may make suboptimal choices.

- Project 2: <https://inst.eecs.berkeley.edu/~cs188/fa25/projects/proj2/>

**Step 39.** At this stage, to stay organized and avoid confusion when working on assignments, it is important to create a dedicated workspace on your computer. Such as



**Step 40.** Download [multiagent.zip](#) from this [link](#)

- Move the [multiagent.zip](#) file to the directory you designated for assignment3 (see above)
- Unzip/extract the [multiagent.zip](#) file inside the assignment directory.

**Step 41.** At this point, you have two options for running the commands provided in the tutorial:

- Local Environment:** Use your own computer's terminal (Command Prompt, PowerShell, or Terminal on macOS/Linux).
- Google Colab:** If you prefer a cloud-based solution, you can use Google Colab.

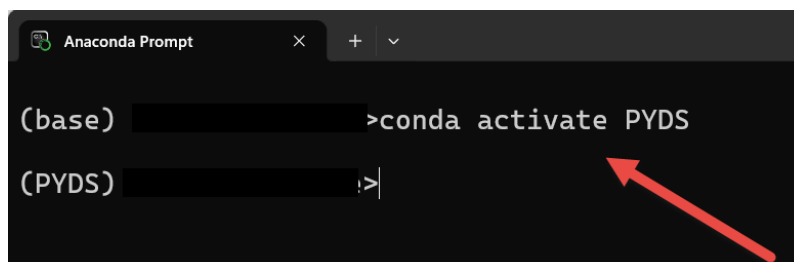
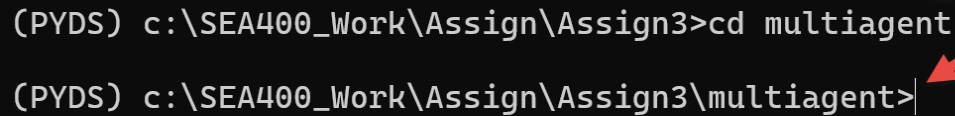


Figure 5

## Assignment3

**Step 42.** Open your terminal window and navigate to the **multiagent** directory

A terminal window with a black background and white text. The first line shows the command '(PYDS) c:\SEA400\_Work\Assign\Assign3>cd multiagent' with a red arrow pointing to the 'multiagent' directory name. The second line shows the prompt '(PYDS) c:\SEA400\_Work\Assign\Assign3\multiagent>' with a red arrow pointing to the prompt itself.

```
(PYDS) c:\SEA400_Work\Assign\Assign3>cd multiagent
(PYDS) c:\SEA400_Work\Assign\Assign3\multiagent>
```

Figure 6

**Step 43.** Follow the instructions to solve and test Question2 and Question3 from the [link](#) (Project 2).

For each question, it is recommended to develop and write your algorithm (pseudo-code) that you plan to implement.

**Step 44.** After you write your pseudo-code on a piece of paper, it is time to implement your algorithm and add comments to each step (or add sufficient comments to your code).

**Step 45.** You will write an adversarial search agent in the provided **ExpectimaxAgent** class stub in **multiAgents.py**. Your minimax agent should work with any number of ghosts.

**Step 46.** We will be checking your code to determine whether it explores the correct number of game states.

You must run the command below to see if your implementation passes all the **autograder** test cases.

```
python autograder.py -q q4
```

**Step 47.** If you do want to see any warning messages the use the following command

Windows:

```
python -w "ignore::SyntaxWarning" autograder.py -q q4
```

Linux (Terminal):

```
python -w "ignore::SyntaxWarning" autograder.py -q q4
```

**Step 48.** You are required to take screenshots to demonstrate the successful execution of your code for the above steps. These screenshots serve as proof of completion and proper identification.

**Step 49.** You need to submit your **multiAgents.py** file, which contains your implementation for Q4.

**Step 50.** In your own words, explain the key assumption difference between **Minimax/Alpha-Beta** and **Expectimax** regarding ghost behaviour.

**Step 51.** Why is **Expectimax** more appropriate than Minimax when ghosts choose *actions uniformly at random*?

## Assignment3

### Milestone3 Deliverables and Group Work

Create assignment report with the following name format

**group\_<number>\_assign\_<assignment number>\_MS3\_report.pdf**

For example, if **group26** created a report for **Assignment20**, then the report name should be

**group\_26\_assign\_20\_report.pdf**

The report should include:

(f) Complete this declaration by adding your names:

We, ----- (mention your names), declare that the attached assignment is our own work in accordance with the Seneca Academic Policy. We have not copied any part of this assignment, manually or electronically, from any other source, including websites, unless specified as references. We have not distributed our work to other students.

(g) Specify what each member has done towards the completion of this work:

	Name	Task(s)
1		
2		
3		

(h) Read the assignment steps and questions carefully. If the assignment questions (or part of the question) are asked for output or response, then you should include the output images inside an assignment report as a PDF file and write a response to answer some of the assignment questions (or part of the question).

(i) As part of your assignment submission, **you are required to take screenshots to demonstrate the successful execution of your code**. These screenshots serve as proof of completion and proper identification.

- Use any document editor (e.g., Word, Google Docs) to compile all your screenshots into a single document.
- Make sure you follow the assignment instructions for the file naming convention

(j) Submit file (s)

- group\_26\_assign\_20\_MS3\_report.pdf**
- multiAgents.py**