# Assignment1

## Contents

***Assignment1***

# Learning Outcomes and Objectives

This Assignment is designed as a **foundational learning experience** that prepares students to successfully achieve the course learning outcomes later in the term, particularly **CLO 5: Implement AI methods using one of the AI languages to solve Artificial Intelligence problems**. Through structured environment setup, Python fundamentals review, and object-oriented programming tasks, students develop the technical fluency required to implement AI algorithms in Python. The use of autograders, modular code, and function-based problem solving reinforces disciplined implementation and testing practices that are essential when working with AI methods such as search, inference, and probabilistic reasoning in subsequent labs. Additionally, exposure to development platforms such as local environments and Google Colab familiarizes students with industry-standard tools commonly used in AI and machine learning workflows.

## Learning Outcomes

- **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 lab tasks are as follows:

- Strengthening proficiency in Python fundamentals required for AI programming
- Develop skills in object-oriented programming using Python
- Practice setting up and managing development environments (local and cloud-based)
- Use automated testing tools to verify and debug program behavior
- Apply structured programming and abstraction techniques needed for AI method implementation
- Build foundational readiness for implementing Artificial Intelligence algorithms in later assignments

# ***Assignment1***

## Assignment Graduation Attributes Indicators

This assignment primarily supports the graduate attributes **IN.1 (Conduct planned activities)** and **ET.2 (Use appropriate techniques, resources, and modern engineering tools)** through structured assignment tasks, environment setup, Python programming, and automated testing. Students engage in hands-on implementation activities, interpret computational results, and validate outputs against expected behavior, contributing partially to **IN.2** and **PA.3**. While the assignment does not directly address complex engineering problem formulation or modeling from first principles (PA.1, PA.2), it provides essential foundational skills and tool proficiency that enable students to analyze and solve more complex Artificial Intelligence problems in subsequent coursework.

- 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

# ***Assignment1***

# Instructions:

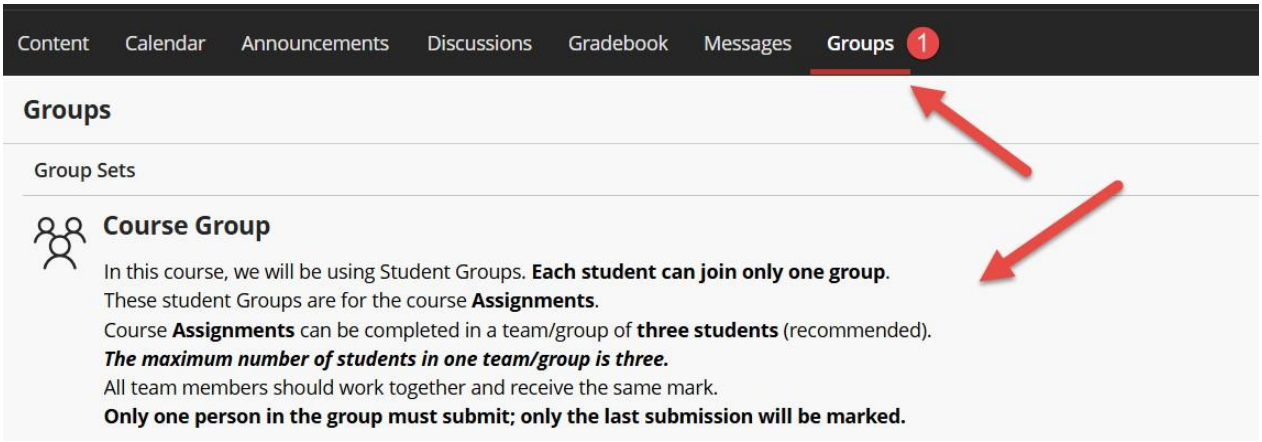
- The assignment can be completed **in groups (maximum of three members per group, recommended)**.
- All group members should work together, and they will receive the same mark.
    - Any partial grade will be given case-by-case.
- This workshop is worth 2% of the total course grade, and it will be graded out of 100 marks and evaluated through your written submission, as well as the lab demo, as follows:
    - 100 marks (2% 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).

***Assignment1***

# Part One: The Group/Team Contract

You need to join and form groups to work on the course workshops. Ideally, you will stick together for the rest of the term . Student Groups are integrated with Blackboard (BB). Log in to the course Backboard (BB) and click on **View Sign-up Sheet to join a Group**. These are self-enrolment groups.

**First**, click on the link "Groups," as shown in the image below.




**Second**, you will see a list of the course groups. By clicking on the Join button, you can join only one group.

# ***Assignment1***

Prepare and agree on a team contract to minimize future disagreements and conflicts. This contract must contain "Team Procedures," "Team Expectations," and "Consequences." **Note that the course instructor can void this contract.** See the following links for samples:

- Guidelines for writing team contracts- University of Arizona: https://math.arizona.edu/~sgfoster/115b/teamcontb.doc
- Team Contract- MIT: http://web.mit.edu/6.005/www/fa15/projects/abcplayer/team-contract/#team_contract

Submit a signed contract copy as **Group_<number>_Contract.docx.** (Adding your names at the end of the digital document is enough.)

***Assignment1***

# Part Two: Setting Up Anaconda Virtual Environment (Optional)

Follow the instructions on Blackboard (BB) for software installation if you have not already done so.

## Editor/IDE Tutorials:

- PyCharm Tutorials (html, html, html)
- Anaconda Tutorials (html, html, html)
- How to Use Windows Command Prompt to Run a Python File (html)

# Part Three: Google Colab: What is Colab?

- Colaboratory is a Google research project created to help disseminate data mining and machine learning education and research. It's a Jupyter Notebook environment that requires **no setup** and runs entirely in the cloud. (from Google Colab Notebooks page)
- It allows you to use virtual machines with a GPU (or TPU) to accelerate machine learning workloads for up to 12 hours at a time.
- It is **free to use!** You can visit Google Colaboratory directly.
- Watch Introduction to Colab to learn more, or just get started below!
- There is a paid option called Colab Pro, which gives access to faster GPUs, more RAM, more CPU cores, more disk space, and longer runtimes. ***Those won't be necessary for this course.***

***Assignment1***

# Part Four: Review Python Basics

**Step1.** Review Python Basics from this link:

a. If you are new to Python or need a refresher, we recommend going through the Python basics tutorial.

**Step2.** 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

```
SEA400
└── Assignments
    ├── Assignment1
    ├── Assignment2
    └── Assignment3
        ...
    └── Assignment6
```

**Step3.** Download **tutorial.zip** from this link

a. A zip archive located here: tutorial.zip
b. Move the **tutorial.zip** file to the directory you designated for assignment1 (see above)
c. Unzip/extract the **tutorial.zip** file inside the assignment1 directory.

**Step4.** At this point, you have two options for running the commands provided in the tutorial:

a. **Local Environment**: Use your own computer's terminal (Command Prompt, PowerShell, or Terminal on macOS/Linux). Make sure you have the required tools installed (e.g., Python, necessary libraries).

**b. Google Colab:** If you prefer a cloud-based solution, you can use Google Colab.

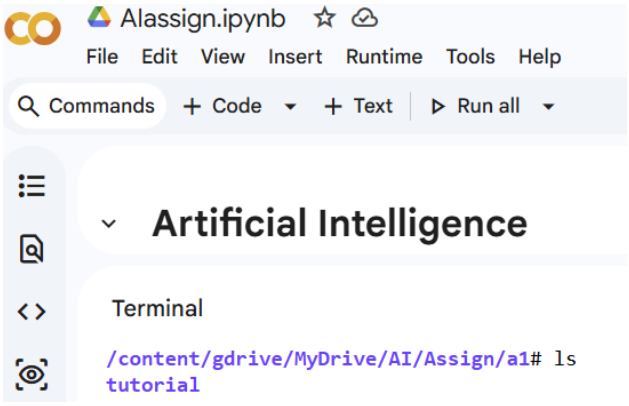



***Figure 1***

# Assignment1

**Step5.** Open your terminal window and navigate to the tutorial directory

```
/content/gdrive/MyDrive/AI/Assign/a1# cd tutorial/
/content/gdrive/MyDrive/AI/Assign/a1/tutorial#
```

***Figure 2***

**Step6.** Keep this terminal open and run code here. For example, you can check the programming environment by running the following command.

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

**Step7.** If you do want to see any warning messages the use the following command
Windows:

```
python -W "ignore::SyntaxWarning" autograder.py -q q
```

Linux (Terminal):

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

**Step8.** Follow the instructions to solve and test questions 1 to 3 from the link (Project 0).

**Step9.** Create a new document file, which you will convert to a PDF file later on before submitting it through Blackboard, and store the screenshot of your results from running these commands (you need to demo running these commands during the lab session)

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

```
Result:
```

## Assignment1

```
Starting on 1-14 at 19:37:15

Question q1
===========

Passed a = 1 and b = 1, returning a + b = 2
*** PASS: test_cases\q1\addition1.test
***     add(a,b) returns the sum of a and b
Passed a = 2 and b = 3, returning a + b = 5
*** PASS: test_cases\q1\addition2.test
***     add(a,b) returns the sum of a and b
Passed a = 10 and b = -2.1, returning a + b = 7.9
*** PASS: test_cases\q1\addition3.test
***     add(a,b) returns the sum of a and b

### Question q1: 1/1 ###


Finished at 19:37:15

Provisional grades
==================
Question q1: 1/1
------------------
Total: 1/1

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.
```

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

Result:

## Assignment1

```
Starting on 1-14 at 20:03:57

Question q2
===========

*** PASS: test_cases\q2\food_price1.test
***     buyLotsOfFruit correctly computes the cost of the order
*** PASS: test_cases\q2\food_price2.test
***     buyLotsOfFruit correctly computes the cost of the order
*** PASS: test_cases\q2\food_price3.test
***     buyLotsOfFruit correctly computes the cost of the order

### Question q2: 1/1 ###


Finished at 20:03:57

Provisional grades
==================
Question q2: 1/1
------------------
Total: 1/1

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.
```

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

Result:

## Assignment1

```
Starting on 1-14 at 20:21:37

Question q3
===========

Welcome to shop1 fruit shop
Welcome to shop2 fruit shop
*** PASS: test_cases\q3\select_shop1.test
***     shopSmart(order, shops) selects the cheapest shop
Welcome to shop1 fruit shop
Welcome to shop2 fruit shop
*** PASS: test_cases\q3\select_shop2.test
***     shopSmart(order, shops) selects the cheapest shop
Welcome to shop1 fruit shop
Welcome to shop2 fruit shop
Welcome to shop3 fruit shop
*** PASS: test_cases\q3\select_shop3.test
***     shopSmart(order, shops) selects the cheapest shop

### Question q3: 1/1 ###


Finished at 20:21:37

Provisional grades
==================
Question q3: 1/1
------------------
Total: 1/1

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.
```

```
python submission_autograder.py
```

Result:

# Assignment1

```
CS 188 Local Submission Autograder
Version 1.4.0

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

Setting up environment..................................................... DONE
Downloading autograder..................................................... DONE
Extracting autograder...................................................... DONE
Preparing student files:
  shopSmart.py..................... OK
  buyLotsOfFruit.py................ OK
  addition.py...................... OK
Running tests (this may take a while):
C:\Users\Admin\AppData\Local\Temp\cs188-r7dqdv0d\tutorial\autograder.py:18: DeprecationWarning: the imp module is depre
  import imp
  Question q1..................... 1/1
  Question q2..................... 1/1
  Question q3..................... 1/1
Generating submission token................................................ DONE

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

Final score: 3/3
Token file: tutorial.token

Please make sure that this score matches the result produced by autograder.py.
To submit your grade, upload the generated token file to Gradescope.

If you encounter any problems, notify the course staff via Piazza.

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

Process finished with exit code 0
```

# Part Five: Object-Oriented with Python

**Step10.** Download the start code for Assignment1 from Blackboard **Assign1.zip**.

**Step11.** Extract the file inside the Assignment1 folder.

**Step12.** Check the tutorial directory, open **shop.py**, read the file, and check the code.

**Step13.** Inspired by **shop.py**, modify **student.py** (given inside the **Assign1.zip** file) to complete the following for class ***Student***:

a. The initialization method takes the ***name*** of the student, as well as ***courseUnits***, a dictionary with keys as course strings and units for values e.g. `{'math':4.0, 'science': 3.0, 'art':2.0}` and sets the related class variables.

b. Method ***getCourseUnits(self, course)*** which looks up '**course'** and returns units of 'course', assuming 'course' is in the *courseUnits* dictionary or None otherwise

c. Method ***getGPA(self, gradeList)*** which given ***gradeList***, a list of (course, grade) tuples, calculates and returns GPA, only including the grades of courses that this student has registered for (those available in *courseUnits*). If none of the courses are in *courseUnits*, it will return *None*.

   For example, for the above student, if ***gradeList*** is `[('math',85.50), ('art',92.0)]`, it returns **87.67** (note the rounding), i.e. **(85.50 * 4.0 + 92.0 * 2.0) / (4.0 + 2.0)**

   And if ***gradeList*** is `[('science',80.00), ('geography',95.0])`, it returns **80.00**, since '**geography'** is not in this student's *courseList.*

# Assignment1

**Step14.** Inspired by the **shopSmart.py** *file* located inside the tutorial directory, modify **bestSemester.py** (given in this workshop) to complete the following:

**Step15.** Function ***highestGPA(semGradeList, astudent)*** which calculates the highest term (semester) GPA given a ***semGradeList***, a list of semester grade lists and a student.

Given the student and grade lists (don't change these), complete the print statements in **main**, so running this code results in the following output:

Expected output:

```
Student Jim registered
GPA for semester 1 for Jim is 83.0
Student not registered for sea100
GPA for semester 2 for Jim is 88.09
Student not registered for sea100
The highest achieved term GPA by this student is 88.09
```

# Part Six: Reflection

Reflecting on your approach and techniques will enhance your understanding of the concepts covered in this Assignment. This reflection will help solidify your learning and identify areas for further study. Ensure you have a firm grasp of the concepts and techniques applied in this lab. After this review, answer the following questions.

**Question 1.** What challenges (if any) did you encounter when setting up your local environment or Google Colab, and how did you resolve them?

I had difficulty accessing the folders added to Google Colab. I resolved this by searching for videos on YouTube.

**Question 2.** Which Python concepts from Project0 (e.g., variables, functions, conditionals, loops, tuples, dictionaries) were most important for completing this assignment?

Personally, I found the concepts of variables, functions, loops, and conditional statements to be the most important for completing this assignment.

**Question 3.** How did examining shopSmart.py help you understand how to approach **highestGPA**?

shopSmart.py helped me understand that:

Accessing collections using a loop, like shopSmart with the list of fruits, and the highest GPA loops through the list of semesters.

Also in shopSmart, I can call the helper method shop.getPriceOfOrder() to calculate the total price and perform a comparison. Therefore, for the highest GPA, I can also rely on the getGPA() method to calculate the score and include it in the comparison.

Finally, how to track the best results, like shopSmart using minCost to continuously track the lowest price. I applied this to the highest GPA but reversed the comparison. Through that, I created a variable called maxGPA to track the highest GPA value.

***Assignment1***

# Deliverables and Group Work

Create workshop report with the following name format

**group_<number>_assign_<assignment number>_report.pdf**

For example, if **group26** created a report for **workshop20,** then the report name should be

**group_26_assign_20_report.pdf**

## The workshop report should include:

(a) Complete this declaration by adding your names:

We, Dat Luan Lu and Maxim Orechnikov, 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 | Dat Luan Lu | Part 1 to part 6 |
| 2 | Maxim Orechinikov | Part 1 to part 6 |
| 3 | | |

(c) Read the assignment steps and questions carefully. If the assignment questions ( or part of the question) 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.
   a. Use any document editor (e.g., Word, Google Docs) to compile all your screenshots into a single document.
   b. Make sure you follow the assignment instructions for the file naming convention

(e) Submit file (s)
   **a. Group_<groupNO>_contract**.docx.
   **b. group_26_assign_20_report.pdf**
   **c. Group_<groupNO>_assign_<Assignment number>.zip**