American College of Thessaloniki

Computer Science 340: Artificial Intelligence

Instructor: Dr. Alexander Astaras

Student: Dusan Boljevic

Date: 18/05/2025

# Introduction

This project was developed as the term project for CS340 to demonstrate understanding and practical implementation of AI/ML techniques taught during the semester using Python. The objective is to apply AI and Machine Learning to a real-world dataset, showing the ability to prepare data, train a MLP (Multi Layer Processing) model, evaluate its performance and interpret the results. The project involved building a series of algorithms to do simple data manipulation on a dataset of MLB (Major League Baseball) players as well as a model trained to predict the results of a student in the “Student Dropout And Academic Success Rate” dataset. The model was trained to predict if a student is likely to graduate or drop out. The technology used for this project consisted of the Python programming language along with its libraries such as pandas and scikit-learn.

# Design

The project’s design is modular in order to ensure maintainability and clarity. The program is structured around a main menu that routes the user to two sub-menus containing the logic to perform the user-picked task. The project’s modular design is a good choice because it allows for scalability (new sub-menus and modules can easily be added).

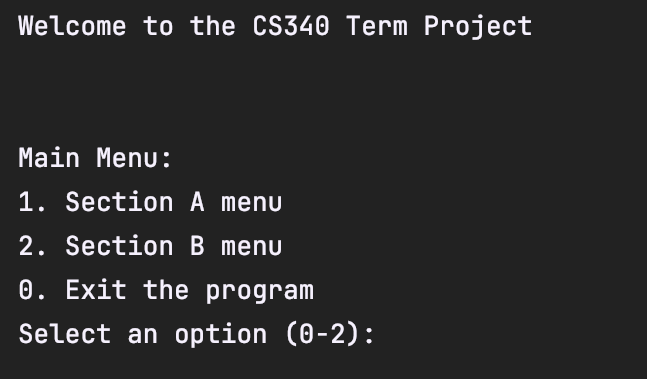
* **main.py:** Serves as an entry point, presenting the main menu to the user and allowing the user to open a sub-menu for either **Section A** or **Section B**.
* **menu\_logic.py:** Contains two functions called **a** and **b**, the two functions consist of the menu logic for each section and each menu choice calls functions for **section\_a.py** and **section\_b.py** respectively.
* **section\_a.py:** Contains all the needed algorithms for performing the tasks outlined for **Section A**, all the algorithms are in functions that are later called in **menu\_logic.py**.
* **section\_b.py:** Contains all the needed algorithms for performing the tasks outlined for **Section B** as well as the data preparation code, all the algorithms except the data preparation part are in functions that are later called in **menu\_logic.py**.

Reasoning behind design:

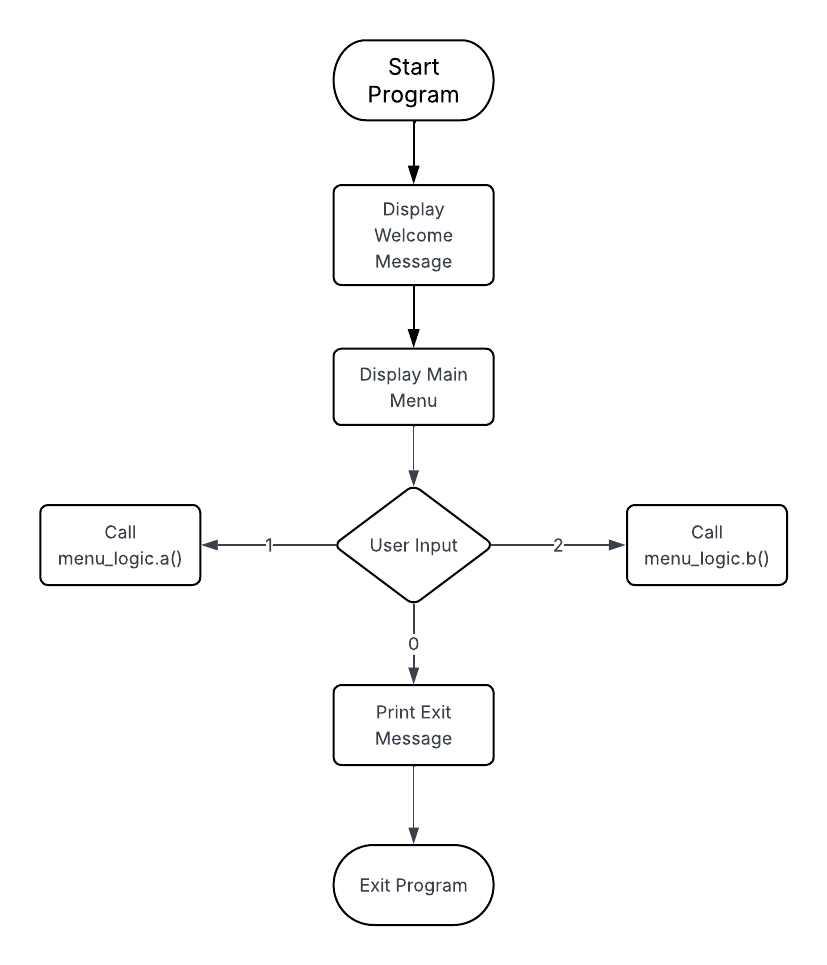
* Independent testing and development of each section.
* Code readability.
* Allows for hypothetical future additions.

Following are the UI and top-level flowchart for main.py:

*main.py output*



*main.py top-level flowchart*

**

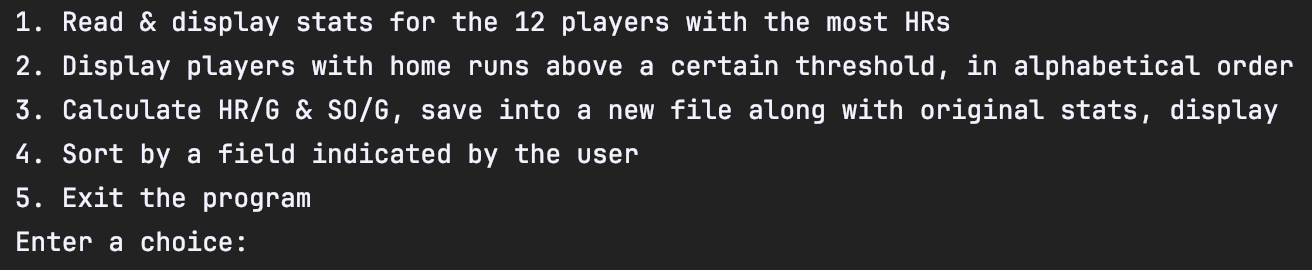
# 

# Section A

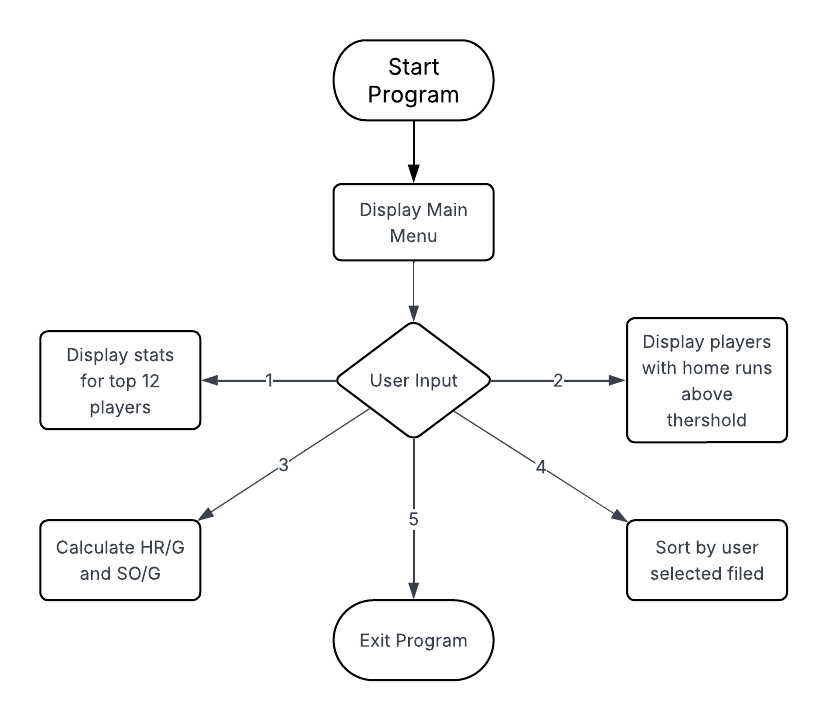
Section A of the CS340 term project was not particularly challenging in comparison with section B, at least when a useful library such as **pandas** is introduced. The code for this section had to do with very simple data manipulation with the data coming from a .csv file containing the stats of the top MLB players. Pandas made working with this data very simple and approachable allowing me, for example, to access variables like a player’s number of home runs by simply stating the index and column name. With this in mind, calculating new variables, filtering and custom sorting could all be done in very few lines of code. This section also didn’t call for any machine learning experimentation, although it can be implemented. If I were to implement an AI/ML model to this section it would probably be one that predicts the number of home runs each player will score next season according to historical data. Of course for such a model the dataset would also need to be larger than the one used, with more variables and more data.

Following are the UI and top-level flowchart for menu\_logic.a():

*menu\_logic.a() output*



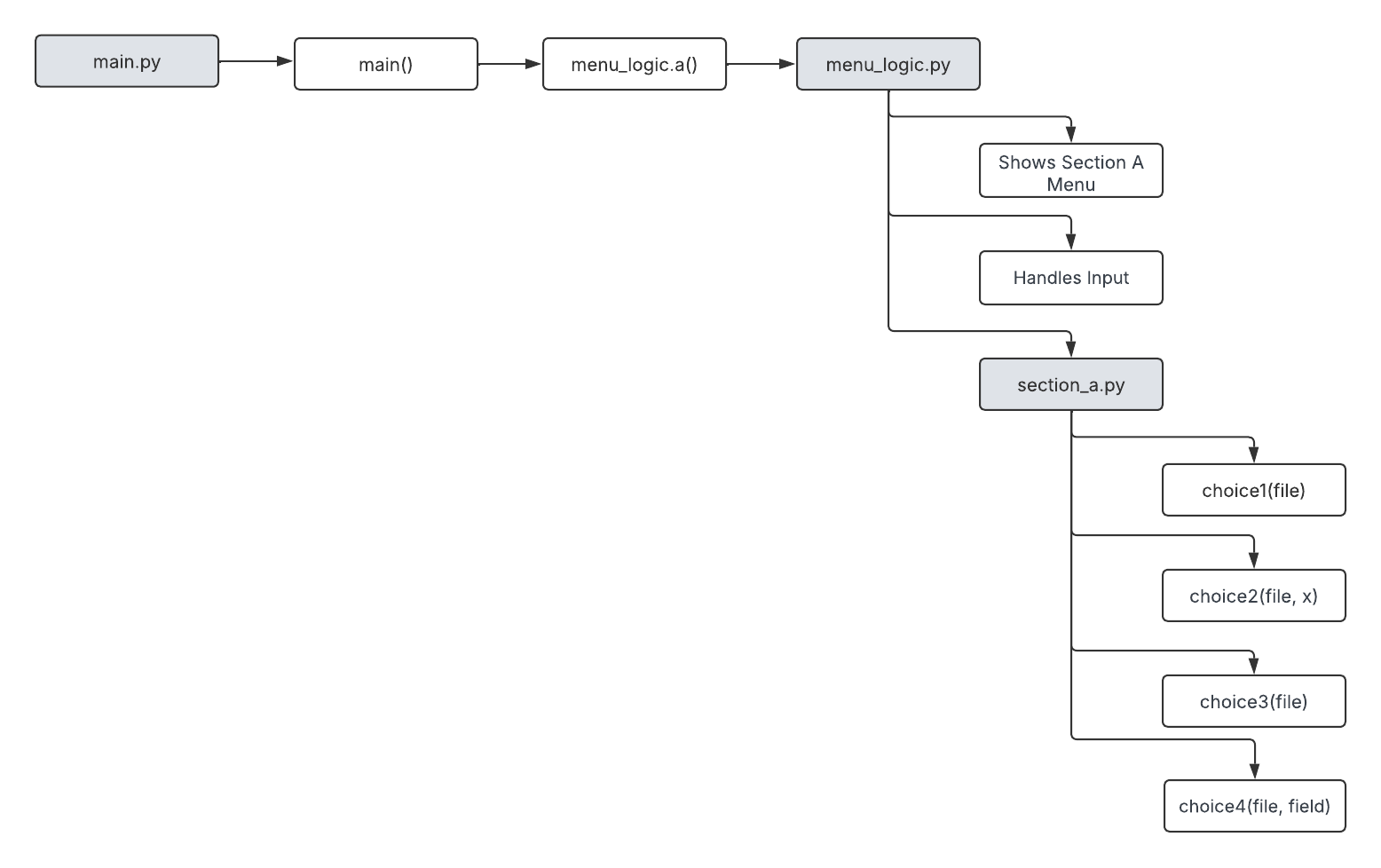
*menu\_logic.a() top-level flowchart*

**

The section\_a.py code itself is accessed using menu\_options.py, hence the flowchart and UI screenshot referring to menu\_options.py instead of section\_a.py. The section\_a.py code is a collection of functions corresponding to their menu choices.

Following is the function interdependency diagram for section\_a.py:

*Interdependency diagram for section A*

**

# Section B

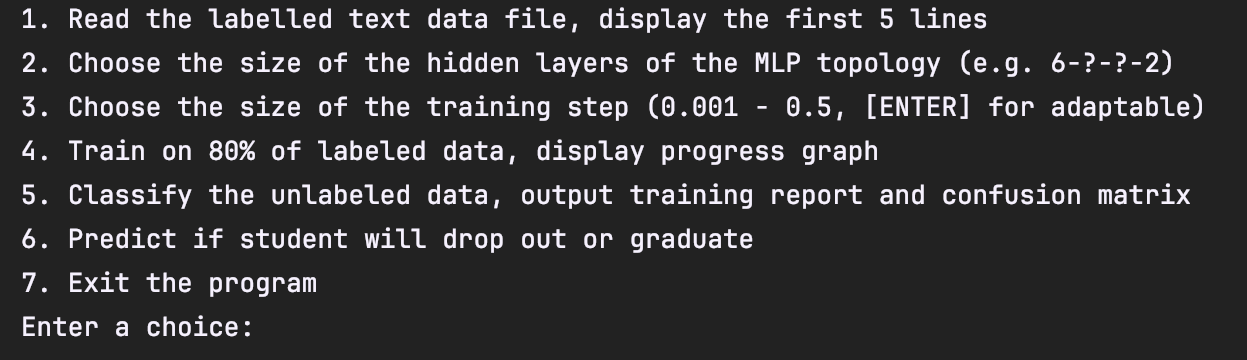
Section B of the CS340 term project was definitely more challenging than section A since it required extra steps such as preparing the dataset and setting up a base for the model which will later be trained based on user input (hidden layers, learning\_rate). My take on the project uses the “Predict Students' Dropout and Academic Success” dataset in order to train the model and predict if a student will graduate or drop out of school. This section required use of Python libraries such as scikit-learn for preparing and training the model, pandas for data cleaning and manipulation, and matplotlib for displaying the **training loss curve** in order to satisfy the project requirements. The approach to a successful AI/ML project is always to first prepare and clean the data before building the model. My first challenge arose in this stage, when I realised the dataset I am working with contained what I called “broken floats” (ex. 13.428.571). Having these “broken floats’ meant that I won’t be able to successfully train my model, leading me to make a function called **fix\_broken\_floats** that:

1. Looks for values that are stored as strings, because that's where the broken float values might be hiding,
2. Defines a pattern in order to extract a valid float value,
3. Fixes the “broken floats’ wherever it spots them,
4. Returns a dataframe with the fixed values.

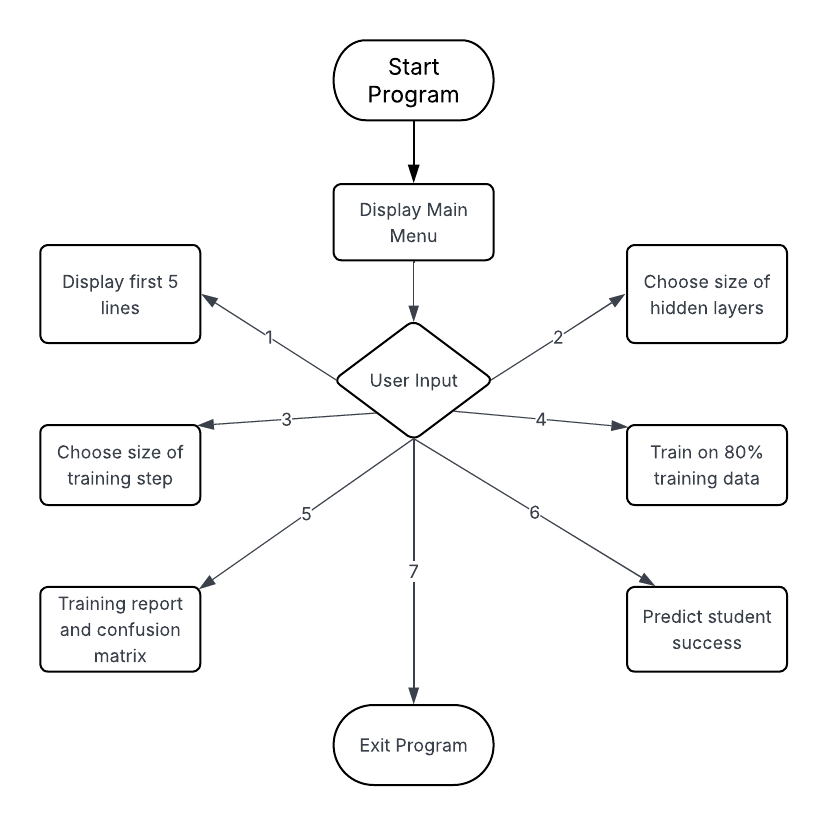
The fix\_broken\_floats function was then applied to the data frame followed by dropping non existent values and cleaning the data set to only contain the rows where the target is either ‘Graduate’ or ‘Dropout’. The values of the categorical variable **target** were then converted from ‘Graduate’ and ‘Dropout’ to 0 and 1. At this point, the data was almost ready, it just needed to be prepared for the train-test split. To do so I created two data frames (X and y) where the first one (X) contained everything from the cleaned dataset except for the **target** column, while the second one (y) contained only the values from the **target** column. Using scikit-learn the train-test split was then produced with the training size of 80%. I also took the liberty of creating some global variables in case the user chooses not to input specific instructions, the **hidden\_layers** global variable contained 1 hidden layer with 100 neurons by default and the learning rate was set to **adaptive** by default. At this point I was ready to start developing the outlined menu choices along with an extra choice I added which allows the user to input the index of a student and the model outputs a prediction (Graduate or Dropout) as well as the actual value in order to validate the responses and figure out how accurate the model is. Menu choices 2, 3 and 4 take advantage of the above mentioned global variables asking the user for input (choice2 and choice3) and changing the global variables’ values if the user chooses to do so, otherwise keeping the default values. The 4th choice is the one that actually trains the model using the global variables and the X\_train and y\_train data frames (derived from the train-test split); it also uses matplotlib to display the **loss curve**. The 5th choice was fairly simple to create, it first tells the user to train the model first (if not already trained with choice4) and uses **classification\_report** and **confusion\_matrix** functionsfrom the scikit-learn library to create and display the classification report and confusion matrix. Finally, the 6th menu choice which I added asks the user for input (number between 0 and size of data frame) and uses the model to predict the value of the **target** variable at the index input by the user. Of course the 6th menu choice also warns the user to train the model first if it is not already trained.

Following are the UI and top-level flowchart for menu\_logic.b():

*output for menu\_logic.b()*



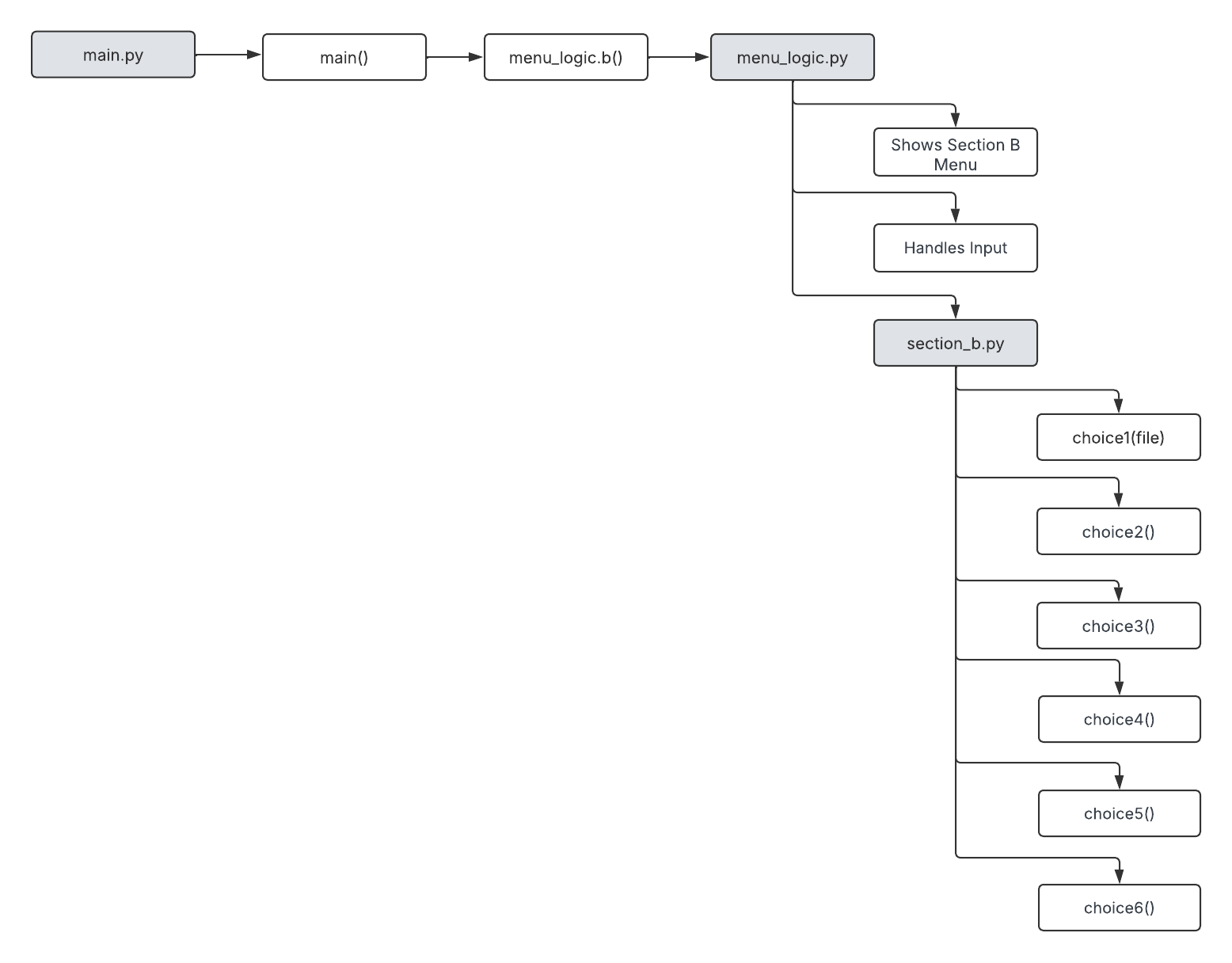
*top-level flowchart for menu\_logic.b()*

**

The section\_b.py code itself is accessed using menu\_options.py, hence the flowchart and UI screenshot referring to menu\_options.py instead of section\_b.py. The section\_b.py code is a collection of functions corresponding to their menu choices.

Following is the function interdependency diagram for section\_b.py:

*Interdependency diagram for section\_b.py*

**

# Conclusion

This project, especially section B, required a fair amount of knowledge of software engineering, data science, Python and its libraries of which there are many. The project could not be done without a deeper understanding of the dataset and machine learning principles. While Python libraries such as scikit-learn and pandas offer solutions that speed up the process of data preparation and training a model, those solutions don’t help much without understanding the theory behind them. Even though I have worked with Python in the machine learning context before, this was my first time using an MLP (Multi-Layer Processing) model. I also gained more experience in data preparation since the project required the use of a .txt file which brings more problems than simply importing the dataset using Python. Finally, I believe my approach is highly modular and ready for any future improvement thanks to the way I designed the program.