

# Weeks 6-8 - Programming Project

## General information

In the previous Assignments, you worked on the concepts seen in the classroom through tailored and focused questions. Now it's the time to change that and start a more hands-on, longer project.

This time, the project consists of two different parts:

- **Part 1:** In the first part, you will work with a **dataset** and implement code that manipulates dictionaries, tuples, and arrays. You have more freedom in this first part and may choose to create/find your own dataset, tailoring your program to anything you find interesting.
- **Part 2:** In the second part of the project, you will implement more predefined functions related to recursion, sorting and plotting. Read below for more information on what you need to do.

**Remark.** In the next three weeks you will have time to work on your project. But don't forget to revise the new content given during the lectures!

## Part 1

Implement **four different functionalities** that operate on the dataset you choose. The functionalities should be encapsulated by (at least) four different functions.

### Example. Plant Watering Tracker project.

You are using a dataset that contains different plant names, and the amount of water/light they require. You build the following four functionalities in your program:

- `add_plant()`: that adds a plant entry to the database. To do that, your function uses another module that you implemented called `database.py` to manage loading from and saving in the database.
- `find_plant()`: that looks for information about a specific plant in the database
- `choose_plant()`: that given the light and sun of a place, return good candidate plants for that place.
- `choose_place()`: that given a plant and two different set of conditions (light and sun), returns the best place for that plant.

Those four functions should **necessarily** use one of the following data structures:

- Dictionary
- Tuple
- Array

**Remark.** All those data structures should be used in your project at least once. In addition to that, you are free to use any data structure that we've learned before or

CHAPTER

VIII

In this chapter:

[General Information](#)

[Part 1](#)

[Part 2](#)

[Additional recommendations](#)

[Project description](#)

[Project description template](#)

[Datasets](#)

[Submission instructions](#)

The **project** is individual, but you can discuss your **ideas** from part 1 with others.

**Dataset:** there are three datasets provided in the assignment from which you can choose (you can find their description at the end of this chapter). Or you can create your own datasets. If you come up with your own dataset, make sure to have enough samples on it ( 30), and variables of at least 2 different types (ex: string and numerical).

that you research yourself.

To help defining it a bit better, see the list of requirements bellow.

### Project requirements (Part 1).

Your project should contain the following files:

- `database.py`: a file that contains any function that deals with reading, writing or updating your database.
- `part1.py`: a file with the functions related to the first part of the project. This file requirements are the following:
  - A `main()` function that prompts the user with the possible choices of your program and is the function called upon running your code.
  - In total you should have at least **four** functions using dictionaries, arrays or tuples.
    - At least **one** of them should use arrays;
    - At least **one** of them should use tuples;
    - At least **one** of them should use dictionaries.
  - `project_description.pdf`. See more information below.

`main.py` should import the other files as modules and manage the interface with the user choices. Depending on what the user asks, the `main` function will call the appropriate function in `part1.py`, `database.py` or any other module of your project.

## Part 2

Download the file `part2.py`. You should implement the 4 functions below there (see a detailed description below):

- `sample_array(arr: ndarray(int/float), n: int)`: you should write the function `sample_array(arr, n)`, which returns a new array with `n` random samples from array `arr`.

**Tip!** This can be done using the `random` module. We did something similar before with the function `random_array`.

- `sort_array(arr: ndarray(int/float))`: should sort the given array `arr` using **one** of the following algorithms.
  - Bubblesort algorithm using Head-Tail technique. You can adapt your solution from problem 12.
  - Quicksort algorithm using Divide and Conquer.
  - Mergesort algorithm using Divide and Conquer.

**Remark.** Do **not** use built-in sorting functions. Your implementation has to use recursion

- `average_time(data: ndarray(int/float), n: int)`: receives the array `data` and an integer `n`. The function computes the average time (out of `n` repetitions) your computer takes to sort the array `data` using the function implemented in `sort_array()`.

**Tip!** Use the `perf_counter()` function from the `time` module to compute the time it takes between two statements.

- `estimate_complexity(data: ndarray(int/float), n_samples: ndarray(int))`: For each integer `n` in `n_samples`, the function should use `average_time()` to compute the mean time taken to sort the first `n` samples of the array `data`. Your function should

You can choose how many repetitions are used to compute the **average time** (but make sure that you use at least three).

return an array with the same shape of `n_samples` in which each element is the corresponding **average time**.

- `main()`: Your main function should use the functions `generate_rnd_array1()` and `generate_rnd_array2()` (already implemented in the `part2.py`) to create two random arrays. Then, your function should generate 3 different **plots**:

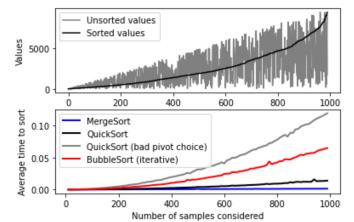
- the two generated arrays in their original order
- the two generated arrays sorted by your functions (ex: `sort_array()`).
- a plot showing the average time (y-axis of the plot) taken to sort each of the two arrays when considering different amount of samples (x-axis of the plot). You should use your implementation of `sort_arrays()` and `estimate_complexity`

### Project requirements (Part 2)

You project should contain the following files:

- `part2.py`: a file with the functions related to the second part of the project.
- `main()`
- `sample_array()`
- `sort_array()`
- `average_time()`
- `estimate_complexity()`
- `project_description.pdf` (Discussion section). See more information below.

Here is an example of a similar **plot** done for a specific random array:



In the top panel, the random (unsorted) data is shown (grey), with its respective sorted version (black). In the bottom panel the performance of three different algorithms are plotted for the dataset in the top panel. Note that some implementation details and the data statistics might influence the performance of some algorithms.

## Additional recommendations

- Your main function in part 1 should keep giving the user the option to execute one of the functionalities or to exit the program.
  - Your program should not have *hardcoded* paths and work in other computer/operating systems (ex: you can use relative paths in the `os` module if you need to navigate folders).
  - Comment your code with (at least) minimal descriptions of what the code is doing.
  - If you want, you can create additional files than the ones described here. As long as your code runs and works.
  - any module that does not require installation can be used (ex: comes with Spyder).
- Beyond that, you can use any of the following modules:

- |          |          |        |
|----------|----------|--------|
| - numpy  | - io     | - time |
| - scipy  | - sys    | - csv  |
| - pyplot | - random |        |

Beyond that, if you wish to use a module that does not fall into that category, ask your TA if that's possible.

## Project description

Beyond your codes, you should also submit a one/two page **project description** (as a pdf). This document should have the following information:

- A **description** of your project (ex: what is it about, what the user should do, how to use it, what the program does, etc)
- A **list of example uses** of arrays, tuples and dictionaries in the first part of your project

### Example.

- **dictionaries:**

In the function `add_plant` the program uses a dictionary to store information about the plant.

- **Discussion:** a short discussion on the results you get when you plot the run-time analysis in part 2. Include what you expected based on Big-O analysis of your code, and what you found for the two random arrays. Are they different? If so, elaborate why that is the case. You can include images of the plot.
- **Acknowledgements and references** of people that conceptually contributed to your project and **external source code** used, if any.
- **Authorship declaration** in the form of the following text:

*I declare that the work submitted here is from my authorship only. I havent used any generative AI to help with any code/text included in my work. I have given credit for the help I had conceptualizing my project. My work respects the university and course code of conduct*

See below an example of what your project description should look like (use that as a template).

If you use parts of code written by others in your project, that should be clearly recognizable in your code (use comments), and credited.

## Project description - Template

### Part 1 - Plant Watering Tracker: project and database

Brief project and dataset descriptions (1-2 paragraphs).

#### Project functionalities:

The user can chose among N options.

- Option 1 (**Dictionary**): calls the function `add_plant(plant_book: dict, plant_name: str, plant_data: tuple)` that modifies the dictionary `plant_book`, adding `plant_name` as a new key with value defined by the tuple `plant_data`.
- Option 2 (**Tuples and Dictionaries**): calls the function `find_plant(plant_name: str)` which ... (description of the function)...
- Option 3: ...
- Option 4: ...
- ...
- Option N: Exit.

### Part 2 - Discussion:

Discuss the result you have when you plot the run-time analysis. Include the following points in your discussion:

- What you expected based on Big-O analysis of your code
- What you found (include images)
- Comment your results. (Ex: if they are different/equal, why is that the case? What would you expect if you have implemented a different sorting algorithm? etc).

### Acknowledgements

I would like to thank the student X, the TA Y and the lecturer Z with the help on discussing the conceptual ideas for this project.

### Authorship declaration

*I certify that this assignment represents my own work. I have not used any unauthorized or unacknowledged assistance or sources in completing it including free or commercial systems or services offered on the internet or text generating systems embedded into software. I have given credit for the help I had conceptualizing my project. My work respects the university and course code of conduct.*

## Datasets

Below you can find the description of the three datasets, which you may use. If none of those datasets appeal to you, feel free to choose/make your own dataset! If you do that do not forget to examine the data before you work with it. Make sure it has no missing values, or otherwise make sure your code can account for those missing values. Last but not least, make sure the data is not too big, or too small, if in doubt ask your TA.

All provided **datasets** are in `.txt` format and as a first line have the titles of the columns (features).

### Movie Dataset

The first dataset we provide is an adventure movie dataset. A link to the full dataset can be found [here](#). We have cleaned and filtered the dataset. In the provided dataset, the entries are movies and the features are the following :

- name - Name of the Movie
- rating - Rating of the Movie
- duration - Duration of the Movie
- year - Year of movie release
- genre - Genre of the Movie
- gross\_income - Gross Income of the Movie

There are 696 entries and 6 features.

### Spotify Dataset

The second dataset is with the most streamed Spotify songs in 2023. A link to the full dataset can be found [here](#). The features we have selected are the following :

- track\_name - Name of the song
- artist(s)\_name - Name of the artist(s) of the song
- released\_year - Year when the song was released
- released\_month - Month when the song was released
- in\_spotify\_charts - Presence and rank of the song on Spotify charts
- mode - Mode of the song (major or minor)
- danceability\_%: - Percentage indicating how suitable the song is for dancing

There are 953 entries with 7 features each.

### Goodreads Dataset

The third dataset is a book one gathered from Goodreads. A link to the full dataset can be found [here](#). The features we have selected for you to work with are the following :

- title - The name under which the book was published.
- authors - Names of the authors of the book. Multiple authors are delimited with "-"
- average\_rating - The average rating of the book received in total
- language\_code - Helps understand what is the primary language of the book. For instance, eng is standard for English
- num\_pages - Number of pages the book contains.
- ratings\_count - Total number of ratings the book received
- publication\_date - Date when the book was first published

**choosing a dataset:** regardless of the dataset you choose to use (either provided by us, found, or made by you), it might be worth it to familiarize yourself with it. To do that, you can write a short description of it. For example, which of the features are numeric? which ones are categorical? are there any edge cases that should be considered in your project? ...

There are 894 entries with 7 features each.

## Submission Instructions - Final Project

### General submission:

Create a zip file called Programming\_Project containing (at least) the files:

- project\_description.pdf
- database.py
- part1.py
- part2.py

Any other file needed to run your project should be included in the zip folder. Moreover, your program should be able to run without error in other computers / operating systems.

### Example.

```
...>python part1.py
```

Submissions that fail those specifications may not be considered.