Python Fundamentals Project

In this project, you will use fundamental Python skills to create an interactive application. Each team will be assigned an application to create based on user requirements.

Project Workflow:

- Select a group name and a name for the application
- Each group will nominate a team leader responsible for creating and maintaining the project repository, assigning tasks, and monitoring progress.
- Create a **public** repository and include a **ReadMe.md** file.
- Adding Collaborators:
    - Go to settings.
    - In the **General** sidebar (left), under the **Access** section, click on **"Collaborators"**.
    - Under **"Manage Access"**, click on **"Add people"**
    - Add your team members. Each team member will receive an invitation to join the repository. Once accepted, you are officially ready to start.

Important Notes:

- Each member in the group should create a separate branch of the repository to work on.
- When your work on a branch is complete, send a pull request.
- The team leader will be responsible for compiling and maintaining the team's code.
- Ensure you have clear variable, function and class names. As well as clear comments and docstrings
- Your functions should be in a python script. You can start up your python application in a Jupyter Notebook during the demo. An example of the python script with functions is the myutils.py in the **modules and scripting github repository.**

Deliverables

- A Jupyter Notebook and a python (.py) file(s) for your given project, containing:
    - A Python application that accepts user input to interact with users. It should cover the User Stories and Technical Requirements
    - A Presentation slide deck that introduces the application and its features. As well as a link to the repository.

Python Fundamentals Project

Application Options

Application	Problem Statement
Option No. 1: **Recipe Manager**	Create a digital recipe book that allows users to store, retrieve, and manage your favorite recipes in a .csv file.
Option No. 2: **Plant Care Tracker**	Create an application for plant enthusiasts to track their houseplants' care schedule and maintenance history using a Python application.
Option No. 3: **Weather Tracker**	Weather enthusiasts want a simple application to track local weather observations and analyze weather patterns over time using a Python application that stores data logically.
Example **Habit Tracker**	Create a personal habit tracking system that helps users build and maintain only *positive* habits with a Python app!

Specific Requirements for Each Application

1. Recipe Manager

Problem Statement
Create a digital recipe book that allows users to store, retrieve, and manage your favorite recipes through a Python application that stores data in a .csv file.

User Stories

1. When I start up the application, I am given the following options:
    1. Add a new recipe to the collection

    2. Search for recipes by ingredient
    3. View all recipes
    4. View a random recipe suggestion
2. If I choose to add a new recipe, I am asked to provide:
    1. The recipe name
    2. A list of ingredients separated by commas
    3. Preparation time in minutes
    4. Cooking instructions
    5. Difficulty level (Easy, Medium, Hard)
3. If I choose to search by ingredient:
    1. The program asks for an ingredient to search for
    2. The program displays all recipes that contain that ingredient
4. If I choose to view all recipes:
    1. The program displays a list of all recipe names with their preparation times
5. If I choose to get a random recipe suggestion:
    1. The program randomly selects and displays a complete recipe from the collection

Technical Requirements

- Stores all recipes in a .csv file
- Loads previously created recipes when the user initializes the application

Stretch Goals - Mandatory

1. The program allows you to categorize recipes (Breakfast, Lunch, Dinner, Dessert)
2. The program can scale ingredient quantities based on desired number of servings
3. The program allows users to rate recipes and sort by rating
4. The program can generate a shopping list based on selected recipes
5. The program can track cooking history and suggest recipes you haven't made recently

2. Plant Care Tracker

Problem Statement

Create an application for plant enthusiasts to track their houseplants' care schedule and maintenance history using a Python application that stores data in a .csv file.

User Stories

1. When I start up the application, I am given the following options:
    1. Add a new plant to the collection
    2. Record a plant care activity
    3. View plants due for care
    4. Search plants by name or location
    5. View all plants
2. If I choose to add a new plant, I am asked to provide:
    1. Plant name/species
    2. Location in home
    3. Date acquired
    4. Watering frequency (in days)
    5. Sunlight needs (Low, Medium, High)
3. If I choose to record a care activity:
    1. The program asks which plant was cared for
    2. The program asks what activity was performed (Watering, Fertilizing, Repotting, Pruning)
    3. The program records the current date for this activity
4. If I choose to view plants due for care:
    1. The program shows a list of plants that need watering or other attention
5. If I choose to search plants:
    1. The program asks for a search term (name or location)
    2. The program displays matching plants with their care details

Technical Requirements

- Stores all plant information and care activities in a .csv file
- Loads previously added plants when the user initializes the application

Stretch Goals - Mandatory

1. The program can track plant growth measurements over time
2. The program can generate seasonal care reminders based on plant type
3. The program allows adding photos (file paths) to document plant progress
4. The program can adjust care schedules based on seasonal changes
5. The program can diagnose common plant problems based on symptoms

3. Weather Tracker

Problem Statement

Weather enthusiasts want a simple application to track local weather observations and analyze weather patterns over time using a Python application that stores data in a csv.

User Stories

1. When I start up the application, I am given the following options:
    1. Record a new weather observation
    2. View weather statistics
    3. Search observations by date
    4. View all observations
2. If I choose to record a new observation, I am asked to provide:
    1. The date in "MM-DD-YYYY" format
    2. The temperature in degrees Celsius
    3. Weather conditions (Sunny, Cloudy, Rainy, Snowy, etc.)
    4. Humidity percentage
    5. Wind speed in km/h
3. If I choose to view weather statistics:
    1. The program displays the average, minimum, and maximum temperatures
    2. The program displays the most commonly recorded weather condition
4. If I choose to search by date:
    1. The program asks for a date to search for
    2. The program displays all observations from that date
5. If I choose to view all observations:
    1. The program displays all recorded weather data in a formatted table

Technical Requirements

- Stores all weather observations in a .csv file
- Loads previously recorded observations when the user initializes the application

Stretch Goals - Mandatory

1. The program can display temperature trends using text-based graphs
2. The program allows filtering observations by month or season
3. The program can predict tomorrow's weather based on historical patterns
4. The program can compare current year data with previous years

5. The program can track and display record-breaking temperatures or conditions

Python Fundamentals Project

Example

Problem Statement

Create a personal habit tracking system that helps users build and maintain positive habits using a Python application that **stores data in a .csv file**.

User Stories

1. When I start up the application, I am given the following options:
    1. Add a new habit to track
    2. Log completion of a habit
    3. View habit streaks and statistics
    4. Edit or remove habits
    5. View all habits
2. If I choose to add a new habit, I am asked to provide:
    1. Habit name
    2. Frequency (Daily, Weekly, Monthly)
    3. Target goal (e.g., number of repetitions)
    4. Category (Health, Productivity, Learning, etc.)
    5. Start date
3. If I choose to log completion:
    1. The program shows active habits that can be logged today
    2. The program asks which habit was completed
    3. The program records the completion with timestamp
4. If I choose to view streaks and statistics:
    1. The program displays current streaks for each habit
    2. The program shows completion rates and best streaks
5. If I choose to edit or remove habits:
    1. The program asks which habit to modify
    2. The program allows updating habit details or deleting the habit

Technical Requirements

- Stores all habit information and completion logs in a .csv file
- Loads previously created habits when the user initializes the application

Stretch Goals - Mandatory

1. The program can visualize habit completion with calendar views

2. The program implements a reward system for maintaining streaks
3. The program sends mock notifications for habit reminders
4. The program analyzes optimal times of day based on past completions
5. The program can suggest habit pairings or "habit stacking" based on successful patterns