Object Oriented and functional programming with python

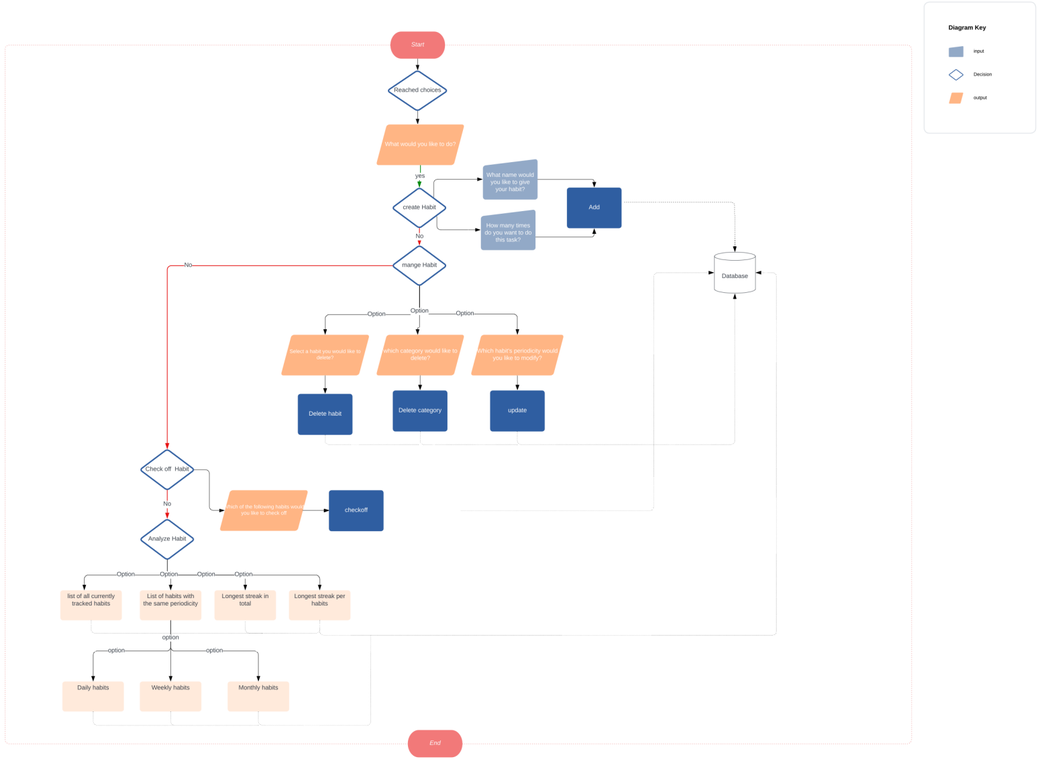
https://github.com/Oumaymabamoh/Habit_Tracker_Backend

Habit tracker backend

3 June 2023

The habit tracker app flowchart:

1. The program started
2. Ask the user to
    - create habit
    - Manage habit
    - checkoff habit
    - analyze habit
3. The program performs actions based on the user's choices.
4. The program will perform actions until the user wants to end the operation.
5. User can exit the program at anytime



Main class

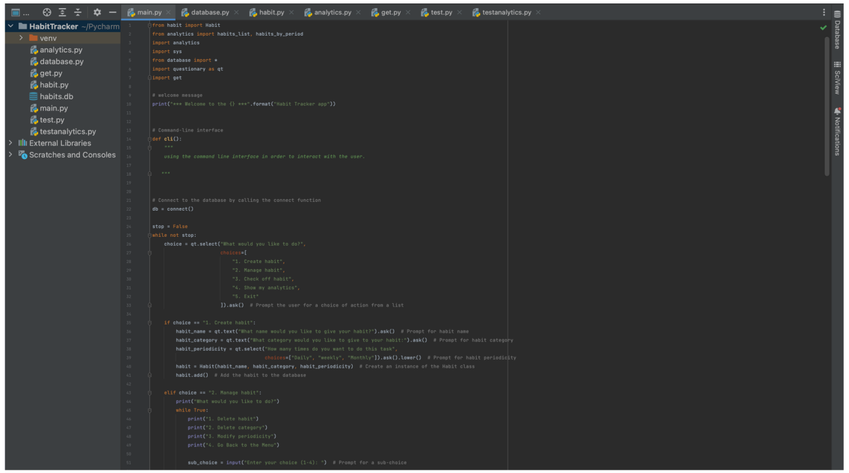
- db = connect(): This function establishes a connection to the database. It is responsible for connecting the application to a data storage system where habit-related data is stored.

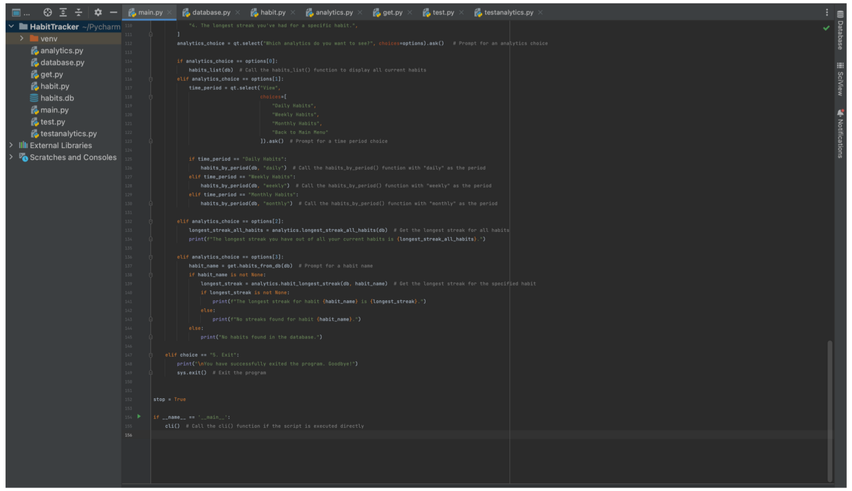
- stop = False: This variable is used as a flag to control the main loop of the application. It determines whether the application should continue running or exit based on user input.

- cli(): This function represents the command-line interface (CLI) of the application. It serves as the main interaction point with the user, displaying options and handling user input to perform various actions.

- Exit: This option within the CLI allows the user to exit the application. It terminates the program and displays a farewell message.

- sys.exit(): This function from the sys module is called when the user selects the "Exit" option. It terminates the program execution and exits the application.





Database Class

- SQLite3 is used as the database system.

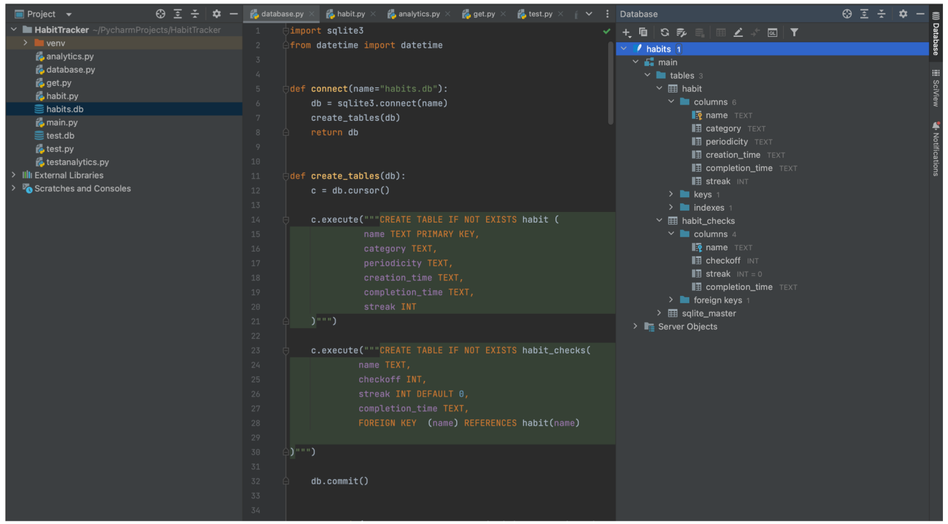
- connect(name="habits.db"): Establishes a connection to the SQLite database. It returns a database connection object that allows for data storage and retrieval. The optional parameter name specifies the name of the database file.

- create_tables(db): Creates the necessary tables in the database if they don't already exist. It uses a cursor object to execute SQL statements that create the "habit" and "habit_checks" tables. The tables are created using the CREATE TABLE IF NOT EXISTS statement, ensuring they are only created if they don't exist. The changes are committed using db.commit().

- db.cursor(): Creates a cursor object associated with the database connection. The cursor object allows for executing SQL statements and fetching data from the database.

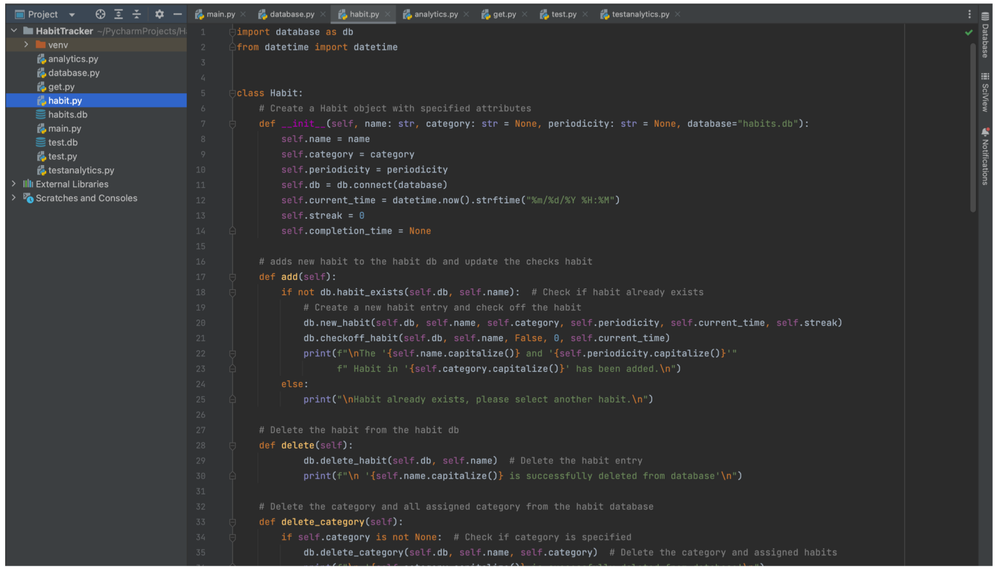
- c.execute(sql_statement, parameters): Executes an SQL statement using the cursor object. The sql_statement parameter contains the SQL statement to be executed, and the parameters parameter (optional) contains the values to be substituted in the SQL statement.

- db.commit(): Commits the changes made to the database. After executing SQL statements that modify the database, the commit() method is called to save those changes permanently.



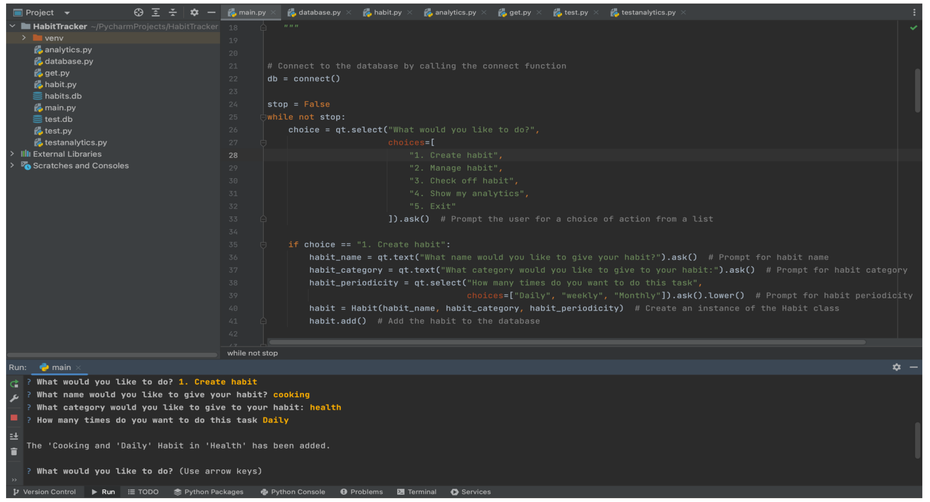
Habit class

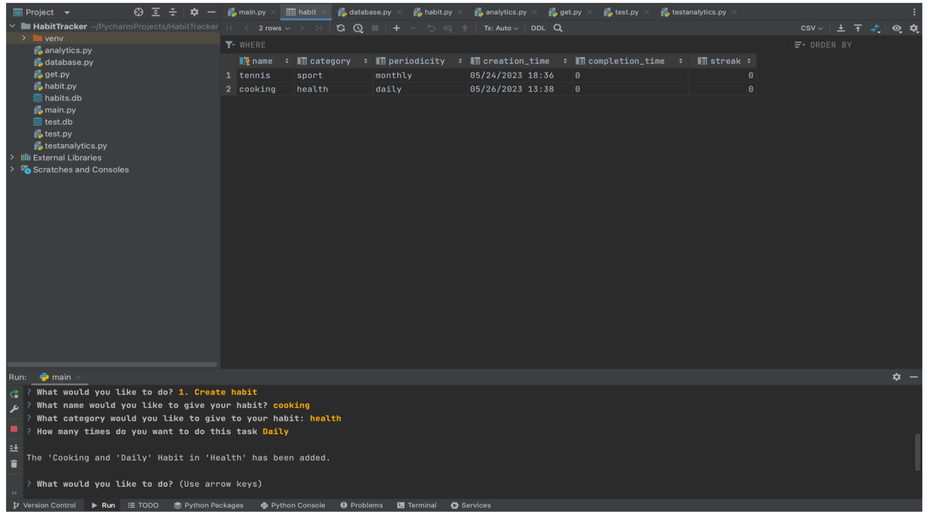
- Classes provide modular organization and code reusability.
- They encapsulate data and behavior, promoting abstraction and code maintainability.
- The Habit class encapsulates the attributes and behaviors related to
    - Adding habits
    - managing habits
    - tracking habit progress
- It interacts with the database module to perform database operations and provides an interface for manipulating and tracking individual habits.



Create habit

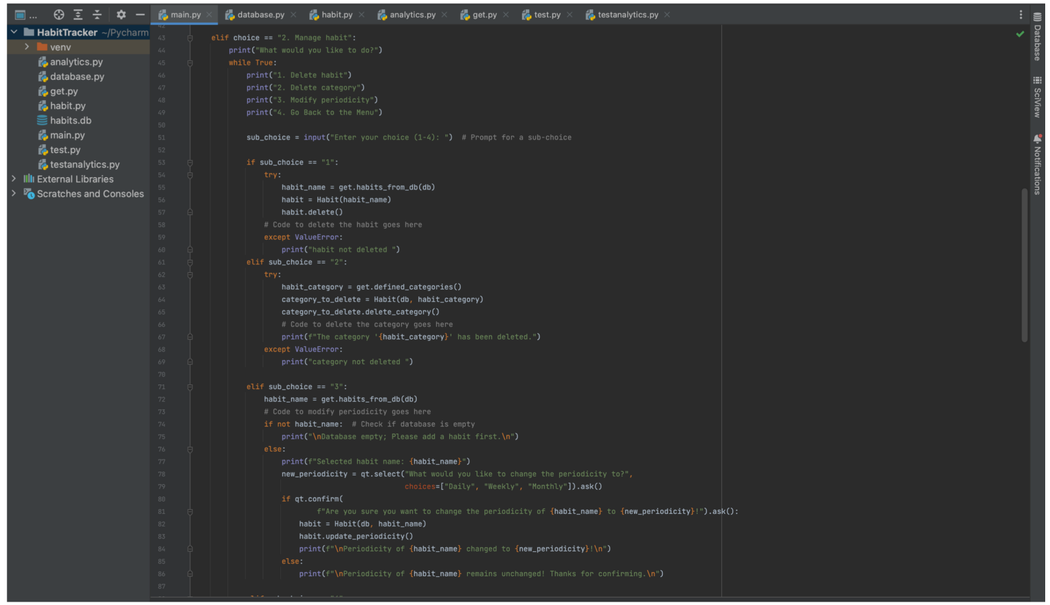
- Program start asking user to enter
  - The user is prompted to enter the habit name.
  - The user is prompted to enter the habit category.
  - The user is prompted to choose the habit's periodicity.
- User manages the program by answering questions in the Terminal
  - Questions in white
  - Answers in orange
- Once the entries are provided by user:
  - a successful message shows up and
  - the data is automatically saved in the database





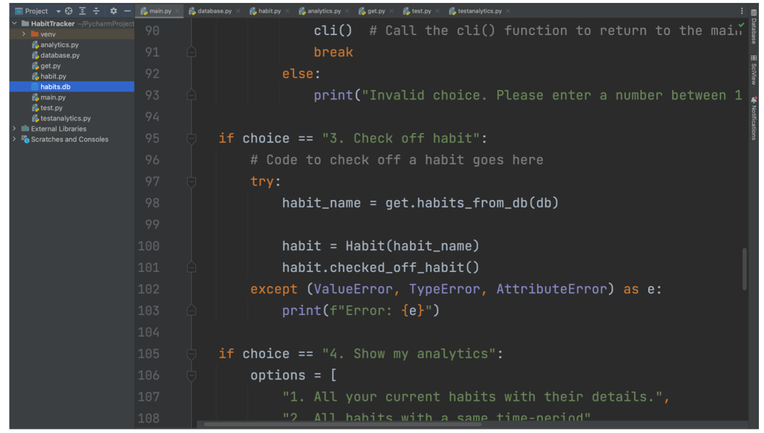
Manage habit

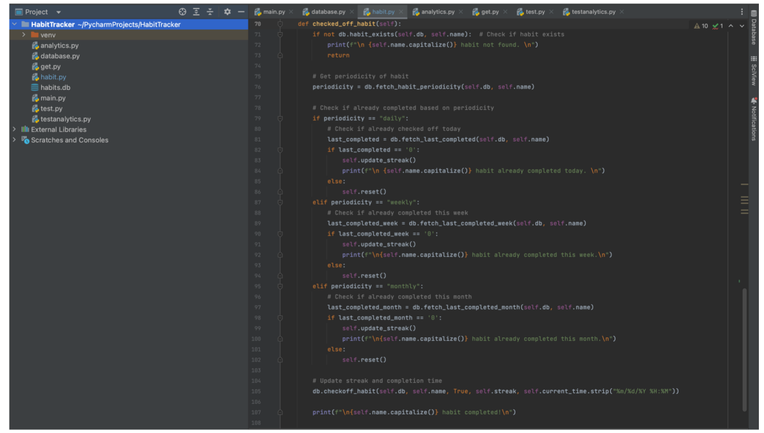
- Deleting a Habit:
  - Prompt the user to select the habit they want to delete.
  - Create a Habit object for the selected habit and call the delete() method to delete it from the database.
  - Inform the user if the habit was successfully deleted or if any error occurred.
- Deleting a Category:
  - Prompt the user to select the category they want to delete.
  - Create a Habit object with the selected category and call the delete_category() method to delete it from the database.
  - Inform the user if the category was successfully deleted or if any error occurred.
- Modifying Periodicity:
  - Prompt the user to select the habit they want to modify the periodicity for.
  - Display the current habit name and prompt the user to enter the new periodicity (Daily, Weekly, or Monthly).
  - Confirm with the user if they want to proceed with the change.
  - Create a Habit object with the selected habit and new periodicity, and call the update_periodicity() method to update the periodicity in the database.
  - Inform the user if the periodicity was successfully changed or if any error occurred.

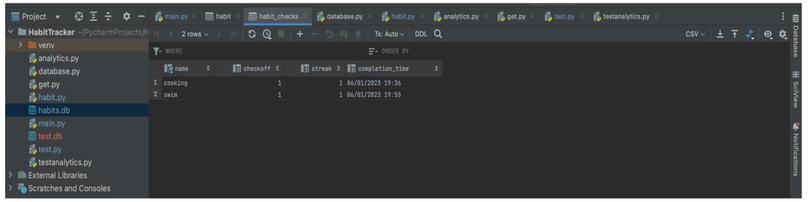


Check off habit

- The streak by 1 when a habit is checked off and resetting the streak to 0 when it is not complete
- The checked_off_habit method is called, which checks if the habit exists and retrieves its periodicity.
- Based on the periodicity, it checks if the habit has already been completed within the specified time period (daily, weekly, or monthly).
- If the habit has not been completed, the update_streak method is called to increment the streak count by 1 and update the streak value in the database.
- The checkoff_habit method is called to update the completion status, streak count, and completion time in the database.
- If the habit has already been completed within the time period, the reset method is called to reset the streak count to 0.



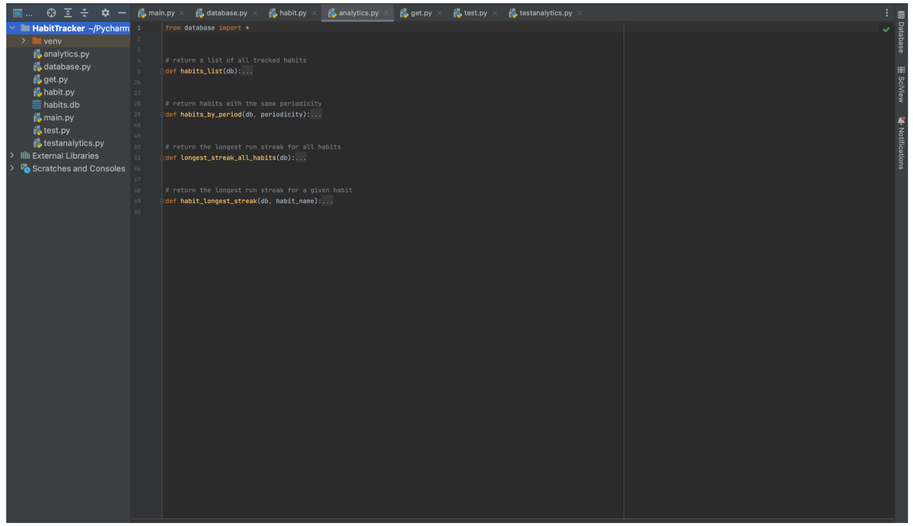




Habit analytics

The analysis module contains a concise overview of notable habit-related information:

- habits_list(db):
  - Retrieves all tracked habits from the database and displays them in a table format.
  - The table includes columns for the habit name, category, periodicity, and completion date/time.
- habits_by_period(db, periodicity):
  - Retrieves all habits with a specified periodicity from the database.
  - Displays the habits in a table format, including columns for the habit name, periodicity, streak, and completion time.
- longest_streak_all_habits(db):
  - Calculates and returns the longest streak among all habits.
  - Iterates over all habits in the database and finds the maximum streak value.
- habit_longest_streak(db, habit_name):
  - Retrieves the longest streak for a given habit.
  - Executes a SQL query to select the maximum streak value from the habit_checks table for the specified habit name.
  - Returns the longest streak value, or 0 if no streak is found or an error occurs.



Thank you