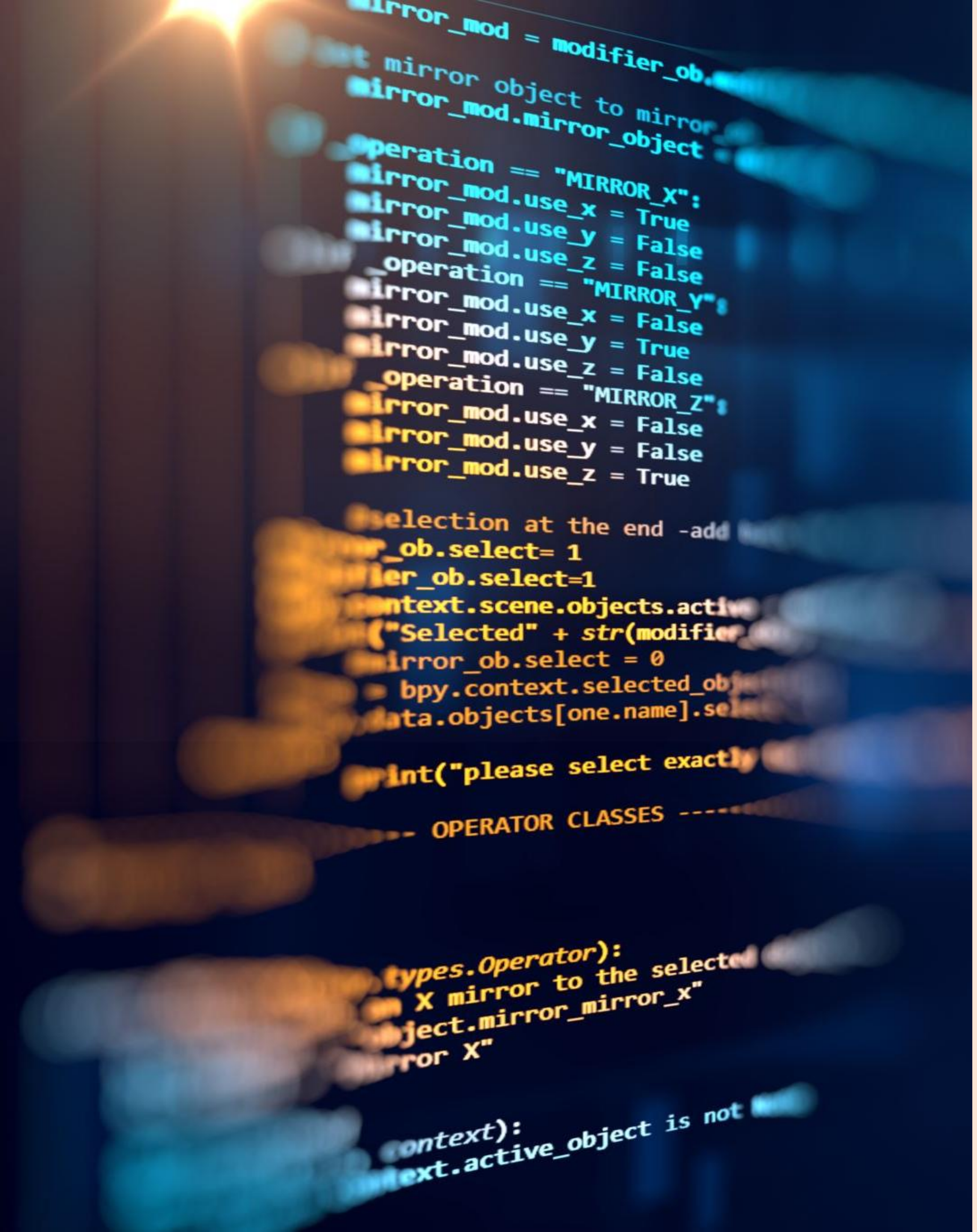
PROJECT: Habits Tracker Application with Python

Development phase

Name: Tourad Baba

Date: 2023-11-13

GitHub Link:
https://github.com/TouradBaba/Habit_Tracker



Habit Tracking App: Implementation

In Phase 2, we delve into the practical implementation of our Habit Tracking App. This presentation provides insights into the frameworks, tools, and components chosen for development. The code is thoroughly commented, ensuring clarity, and user-friendly features allow seamless habit creation, management, and analysis.

Introduction

Conception Phase Recap: The conception phase involved a detailed design process, focusing on clarity and user experience. The design diagram served as a visual guide, ensuring a systematic approach to development.

Frameworks and Tools: Our development stack includes Python 3.11 for robust coding, SQLite for efficient database management, questionary for implementing the CLI and Unittest for comprehensive testing. These choices align with the goal of creating a scalable and resilient application.

Tools and Frameworks: Overview

Python 3.11 is the primary programming language for our application, providing versatility and a rich ecosystem of libraries.

SQLite: SQLite serves as the relational database management system. Its lightweight nature and simplicity make it ideal for our habit tracking app.

Using questionary for implementing the CLI and building a user-friendly interface. That give user access to all tools and functions.

Unittest: Unittest is our testing framework, ensuring the reliability and functionality of our code through automated testing.

Database Implementation

Database Module (db):

Manages the application's interaction with the SQLite database.

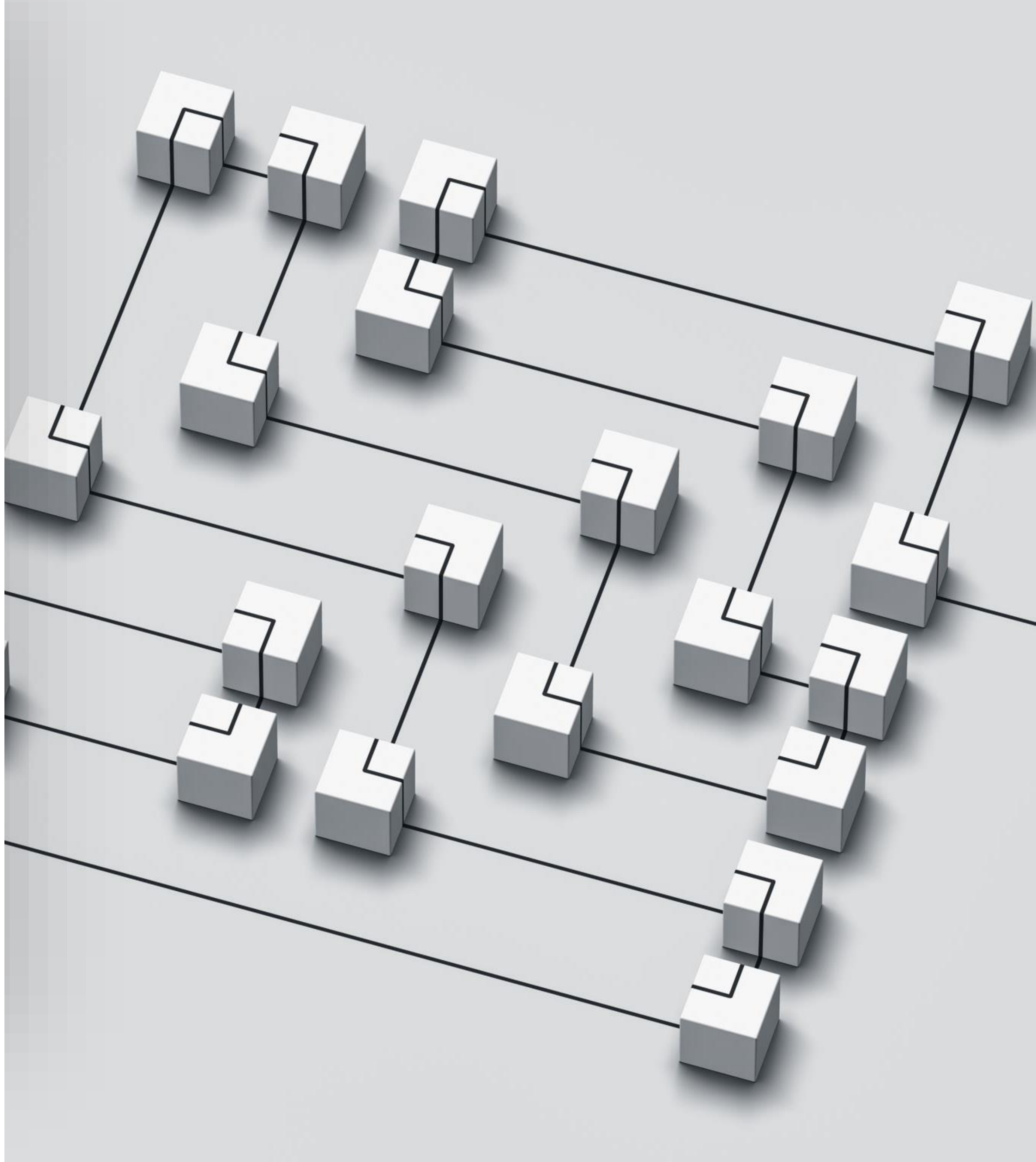
Tables:

-Habits Table:

Stores habit information (name, description, periodicity, etc.).

-Habit Logs Table:

Records habit completion details.



Database Implementation (Connections)

Connection to Habit Class:

Usage in Habit Class:

The Habit class utilizes the SQLite database for persistent storage of habit-related information.

Connection established through the db module to interact with the database seamlessly.

Data Operations:

Addition: New habits are added to the database using the add_habit function.

Removal: Existing habits are removed through the delete_habit function.

Update: Habit details are modified using the update_habit function.

Streak and habit completion: by using increment_streak, reset_streak,update_habit_progress and mark_as_completed functions.

Connection to Analytical Module:

Analytical Operations:

The analytical module interacts with the database to perform data analysis and generate insights.

Queries are executed to fetch relevant information, such as completion rates and streaks.

Examples:

Longest Streak Overall: Queries the database for the maximum streak across all habits.

Completion Rates: Calculates success rates by querying habit logs and creation times.

Habit Class - Implementation Details

- Habit Class
  - Purpose:
    - Represents a habit with attributes such as name, description, periodicity, and category.
    - Manages habit-related data in the SQLite database.
  - Methods:
    - add(self):
      - Adds a new habit to the database if it doesn't already exist.
      - Provides feedback on the success or failure of the operation.
    - remove(self)
      - Removes an existing habit from the database.
      - Verifies habit existence before deletion.
    - update(self, new_name, new_description, new_periodicity, new_category)
      - Modifies the details of an existing habit.
      - Ensures the habit exists before updating.

Habit Class - Implementation Details (Contd.)

- Methods (Contd.):
  - increase_streak(self)
    - Increments the streak count of a habit.
    - Retrieves the current streak from the database and updates it.
  - clear_streak(self)
    - Resets the streak count of a habit to zero.
    - Updates the database accordingly.
  - modify_streak(self, new_streak)
    - Modifies the streak count of a habit based on user input.
    - Handles input validation to ensure the input is a valid integer.
  - complete_habit(self)
    - Marks a habit as completed in the database.
    - Invokes the increase_streak method upon completion.
- User Interaction:
  - These methods align with user actions, providing a convenient interface for habit management.
  - Examples include adding, removing, updating, and completing habits.

Analytical Module - Implementation Details

- Analytical Module
  - Purpose:
    - Provides functions for analyzing habits and their statistics.
    - Key functionalities include tracking habits, calculating completion rates, and identifying areas for improvement.
  - Functions:
    - get_all_tracked_habits(db_conn)
      - Retrieves the names of all tracked habits from the database.
      - Facilitates a comprehensive view of available habits.
    - habits_by_periodicity(db_conn, periodicity)
      - Return habits by their periodicity (daily, weekly, monthly)
    - calculate_completion_rate(db_conn)
      - Determines the completion rate of each habit based on its periodicity.
      - Considers daily, weekly, and monthly habits, providing nuanced insights.
    - find_longest_streak_overall(db_conn)
      - Identifies the longest streak among all habits.
      - Offers an overarching view of the most consistent habits.

Analytical Module - Implementation Details (Contd.)

- Functions (Contd.):
  - find_longest_streak_for_habit(db_conn, habit_name)
    - Locates the longest streak for a specific habit.
    - Useful for pinpointing exceptional performance in a particular habit.
  - identify_habits_needing_improvement(db_conn, completion_rate_threshold=0.7)
    - Pinpoints habits with completion rates below a specified threshold.
    - Aids users in identifying areas for improvement.
- Integration:
  - These functions seamlessly integrate into the habit analysis feature.
  - Enable users to gain valuable insights into their habits and areas for enhancement.

CLI (Command Line Interface)

- Tool: The habit tracking app employs the questionary library to create an efficient and user-friendly Command Line Interface (CLI).
- Purpose:
  - Allows users to interact with the habit tracking app through the command line.
  - Facilitates easy and intuitive management of habits.
- Key Features:
  - Connecting to the Database:
    - Establishes a connection to the SQLite database.
    - Ensures seamless communication with the app's data.
- Menu Options:
  - Show All Habits: Displays a list of all tracked habits.
  - Add Habit: Enables users to add new habits, with options for custom or predefined habits.
  - Remove Habit: Provides options to remove single habits or all habits.
  - Update Habit: Allows users to modify habit details.
  - Habit Streak: Manages habit streak actions like completion, streak increase, clearing, and modification.
  - Habit Analysis: Offers insights into habit statistics and performance, This menu give access to the analytical module functions.
  - Exit: Closes the CLI and exits the application.
- Integration with Habit Class and Analytical Module:

  Interacts seamlessly with the Habit class and Analytical Module to perform actions on habits and analyze user data.

Predefined Habits implementation

- Overview:
    - The habit tracking app introduces a set of predefined habits to streamline the habit creation process.
    - Users have the option to choose from a curated list of 5 habits, comprising 2 daily, 2 weekly, and 1 monthly habit.
- Integration with CLI:
    - Implemented within the CLI framework, predefined habits enhance the user experience by offering ready-to-use options.
    - When a user selects the "Add Habit" option in the CLI, they are presented with the choice to either enter a custom habit or choose from predefined ones
- Streamlined Habit Creation:
    - Users opting for predefined habits experience a streamlined creation process, avoiding the need to input habit details manually.
    - Each predefined habit comes with preset periodicity, category, and description, simplifying the habit setup.

Testing

- Habit Class Testing:
  - Objective: Verify the correctness of methods in the Habit class.
  - Coverage:
    - Addition, removal, and updating of habits.
    - Streak management functions (increase, clear, modify).
    - Habit completion tracking.
- Database Operations Testing:
  - Objective: Validate the integrity and functionality of database-related operations.
  - Coverage:
    - All methods: Insertion, deletion, and modification of habits in the database…
- Analytical Module Testing:
  - Objective: Ensure accurate habit analysis and performance calculation.
  - Coverage:
    - All methods: Listing habits based on periodicity, Longest streak calculations, Completion rate computations….

- Testing Framework:
  - Tool: Unittest.
  - Reasoning: A comprehensive testing framework providing a structured approach to test development and execution.
- Test Results:
  - Habit Class: All methods passed tests, confirming proper habit management and streak handling.
  - Database Operations: Database interactions were validated, ensuring accurate habit data storage.
  - Analytical Module: Analytical functions exhibited correct habit analysis and performance assessment.