- Firstly, we import the necessary libraries needed to run the program. This includes the json module that helps us to store user's habits and handle JSON data. We also import the datetime module to get the current date from the user's system. Next, we create a HabitTracker class, under which all defined functions can be used. We start with the __init__ method and pass the parameter self. We initialize an instance variable of 'allHabits' as a dictinary. The 'daily' and 'weekly' are treated as keys. Then we call the 'loadHabitData' method to load existing habit data. We define the 'loadHabitData' method and pass the parameter self. It then goes through a try-catch exception, where it checkes if a file called 'habit_data.json' exists and gives it read permission. If the file does exist, it loads the existing data. If the file does not it ignores the exception.

- Next, we define a method called 'addHabit' and pass the parameters self, habitName and dailyWeekly. We then create a variable called 'habitDict'. We update the 'habitDict' to include all the habits listed in the Daily and Weekly list. If the 'habitName' is not present in the 'habitDict' it adds it and a message is printed to the console saying that the entered habit has been added to the Daily/Weekly list. If the 'habitName' is present in the 'habitDict' it prints a message to the console saying habit name is already present. Similary we also define another method called 'removeHabit' and pass the parameters self, habitName, and dailyWeekly. We then create a variable called 'habitDict'. We update the 'habitDict' to include all the habits listed in the Daily and Weekly list. If the 'habitName' is present in the 'habitDict' it adds it and a message is printed to the console saying that the entered habit has been removed from the Daily/Weekly list. If the 'habitName' is not present in the 'habitDict' it prints a message to the console saying the entered habit is not found.

- Then we define a method called 'markHabit' and pass the parmeters self, habitName, date and dailyWeekly. We then create a variable called 'habitDict'. We update the 'habitDict' to include all the habits listed in the Daily and Weekly list. If the 'habitName' is present in the 'habitDict', and if the 'date' is not present in the 'habitDict', the date is added for that specific habit and a message is printed to the console saying that the entered habit has been marked for the entered date. If the date is already present for the entered habit a message is printed to the console saying that the date has been already marked for the entered habit. If the habit is not found in the 'habitDict' it prints a message to the console saying the habit is not found.

- Then we define a new method called ‘calculate_longest_streak and pass the parameters self and habit_dates. We initialize the current_streak and longest_streak as 0. We also set the ‘current_date’ as none. Then we run a for loop and format the date in dd/mm/yy format and store the value in a variable ‘date’. We then check if the ‘current_date’ is None or if the ‘date’ minus the ‘current_date’ is 1. If it is then the streak is increase by 1. If it is not the ‘current_streak’ is set at 1. Also, if the ‘current_streak’ is greater than the ‘longest_streak’, the ‘longest_streak’ replaces the ‘current_streak’ and the ‘current_date’ is set as ‘date’. And also, the longest_streak is returned.

- We then create an instance of the ‘HabitTracker’ class called ‘habitTracker’. Next, we run a while True loop. This loop consists of 4 options that the user can choose. These are to Add a habit, Remove a habit, Mark a habit and to Exit from the loop. A variable ‘choice’ is created to keep track of the user’s choices.
  If the user chooses 1, he is asked to enter the frequency of the habit (Daily or Weekly) and the result is stored in ‘dailyWeekly’. Next, he is asked to enter the habit to add and the result is stored in ‘habitName’.
  If the user chooses 2, he is asked to enter the frequency of the habit (Daily or Weekly) and the result is stored in ‘dailyWeekly’. Next, he is asked to enter a habit to remove and result is stored in ‘habitName’.
  If the user chooses 3, he is asked to enter the frequency of the habit (Daily or Weekly) and the result is stored in ‘dailyWeekly’. Next, he is asked to enter a habit to mark and result is stored in ‘habitName’. Then the current system date is retrieved and formatted in dd/mm/yy and stored in ‘date’.
  If the user chooses 4, the ‘habit_data’ is updated and the while Loop is exited.
  If the user chooses any other numbers, then it prints a message to the console saying the enter 1,2,3 or 4.

- If the user chooses 3 the longest running streak of all habits is returned.
  If the user chooses 4 the longest running streak of a given habit is returned.
  If the user chooses 5 the loop is exited.
  If the user chooses any other numbers, then it prints a message to the console saying the enter 1,2,3,4 or 5.

- Here is the output access from the terminal for the first while True loop. Here when 1 is entered, the program asks for a periodicity and a habit. Once the user enters his values, the values are implemented in the database. Similarly for choice 2 and three, the same process takes place but instead to either remove an existing habit or to mark it respectively.

- Here is the output access from the terminal for the second while True loop. Here when 1 is entered, the program lists all tracked habits. Similarly for choice 2, the habits are listed based on periodicity. For choice 3, the longest running streak is listed. Here having 1 value of data is considered having a streak of 1 and 2 sets of data is considered having a streak of 2 and so on. Similarly for choice 4, the longest streak is fetched, but only for a particular habit asked from the user.