Object-Oriented Programming in Python

Matriculation number:9212232
Name: Adam Mufazzal

```python
import json
from datetime import datetime

class HabitTracker:
    def __init__(self):
        self.allHabits = {"daily": {}, "weekly": {}}
        self.loadHabitData()

    def loadHabitData(self):
        try:
            with open("habit_data.json", "r") as jsonFile:
                data = json.load(jsonFile)
                self.allHabits.update(data)
        except FileNotFoundError:
            pass
```

Firstly we import the necessary libraries needed to run the program. This includes the json module that helps us to store users habits and handle JSON data. We also import the datetime module to get the current date from the users 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.

```
def addHabit(self, habitName, dailyWeekly):
    habitDict = self.allHabits[dailyWeekly]
    if habitName not in habitDict:
        habitDict[habitName] = []
        print(f"{habitName} has been added to the {dailyWeekly} list ")
    else:
        print(f"{habitName} is already present ")

def removeHabit(self, habitName, dailyWeekly):
    habitDict = self.allHabits[dailyWeekly]
    if habitName in habitDict:
        del habitDict[habitName]
        print(f"{habitName} has been removed from the {dailyWeekly} list ")
    else:
        print(f"{habitName} not found ")
```

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.

```
    def markHabit(self, habitName, date, dailyWeekly):
        habitDict = self.allHabits[dailyWeekly]
        if habitName in habitDict:
            if date not in habitDict[habitName]:
                habitDict[habitName].append(date)
                print(f"{habitName} has been marked for {date} ")
            else:
                print(f"{habitName} already marked for {date} ")
        else:
            print(f"{habitName} 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.

```
def calculate_longest_streak(self, habit_dates):
    #habit_dates.sort()

    current_streak = 0
    longest_streak = 0
    current_date = None

    for date_str in habit_dates:
        date = datetime.strptime(date_str, "%d/%m/%y")

        if current_date is None or (date - current_date).days == 1:
            current_streak += 1
        else:
            current_streak = 1

        if current_streak > longest_streak:
            longest_streak = current_streak

        current_date = date

    return longest_streak
```

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.

```python
# Function to calculate the longest streak for daily habits
    def calculate_daily_longest_streak(habit_data):
        longest_streaks = {}
        for habit, dates in habit_data.items():
            longest_streaks[habit] = habitTracker.calculate_longest_streak(dates)
        return longest_streaks

    # Function to calculate the longest streak for weekly habits
    def calculate_weekly_longest_streak(habit_data):
        longest_streaks = {}
        for habit, dates in habit_data.items():
            dates_flat = [date for week_dates in dates for date in week_dates]
            longest_streaks[habit] = habitTracker.calculate_longest_streak(dates_flat)
        return longest_streaks
```

Next we create a function called longestStreak() and pass two parameters. Then we create a variable called longestStreakSql and pass the required SQL values. Then we execute the changes. We run an if statement to see if data is present in the result variable. If it is the result is returned. If it is not the 0 is returned as a value.

```python
habitTracker = HabitTracker()

while True:
    print("\nHabit Tracker")
    print("1. Add a habit")
    print("2. Remove a habit")
    print("3. Mark a habit")
    print("4. Exit")

    choice = input("\nEnter your choice ")

    if choice == "1":
        dailyWeekly = input("Enter daily or weekly ")
        habitName = input("Enter your habit ")
        habitTracker.addHabit(habitName, dailyWeekly)

    elif choice == "2":
        dailyWeekly = input("Enter daily or weekly ")
        habitName = input("Remove your habit ")
        habitTracker.removeHabit(habitName, dailyWeekly)

    elif choice == "3":
        dailyWeekly = input("Enter daily or weekly ")
        habitName = input("Enter a habit you want to mark ")
        date = datetime.now().strftime("%d-%m-%Y")
        habitTracker.markHabit(habitName, date, dailyWeekly)

    elif choice == "4":
        with open("habit_data.json", "w") as jsonFile:
            json.dump(habitTracker.allHabits, jsonFile)
        break

    else:
        print("Enter 1, 2, 3, 4")
```

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 retrived and formatted in dd/mm/yy nad 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.

```
while True:
    print("\nHabit Tracker Data: ")
    print("1. List of all currently tracked habits")
    print("2. List of all habits with the same periodicity")
    print("3. Longest running streak of all habits")
    print("4. Longest running streak of a given habit")
    print("5. Exit")

    choice1 = input("Enter your choice ")

    if choice1 == "1":
        for dailyWeekly, habits in habitTracker.allHabits.items():
            for habit, dates in habits.items():
                print(f"{dailyWeekly.capitalize()} habit {habit}: {', '.join(dates)}")

    elif choice1 == "2":
        for dailyWeekly, habits in habitTracker.allHabits.items():
            for habit, dates in habits.items():
                print(f"{dailyWeekly.capitalize()} habit {habit}: {', '.join(dates)}")
```

Here we run a second while True loop where we print the whether we want to show all tracked habits, show all habits with the same periodicity, the longest running streak of all habits and the longest running streak of a given habit.
If the user chooses 1 a for loop is run to return all currently tracked habits.
If the user chooses 2 a for loop is run to return all habits with the same periodicity.

```
elif choice1 == "3":
    for habit, _ in habitTracker.allHabits["daily"].items():
        streak = habitTracker.calculate_longest_streak(habitTracker.allHabits["daily"][habit])
        print(f"Streak for {habit}: {streak}")
    for habit, _ in habitTracker.allHabits["weekly"].items():
        streak = habitTracker.calculate_longest_streak(habitTracker.allHabits["weekly"][habit])
        print(f"Streak for {habit}: {streak}")

elif choice1 == "4":
    habitName = input("Choose a habit ")
    streak = habitTracker.calculate_longest_streak(habitName)
    print(f"Longest streak for {habitName}: {streak}")

elif choice1 == "5":
    break

else:
    print("Enter 1, 2, 3, 4, 5")
```

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 or5.

```
Habit Tracker
1. Add a habit
2. Remove a habit
3. Mark a habit
4. Exit

Enter your choice 1
Enter daily or weekly daily
Enter your habit Brushing
Brushing has been added to the daily list

Habit Tracker
1. Add a habit
2. Remove a habit
3. Mark a habit
4. Exit

Enter your choice 1
Enter daily or weekly weekly
Enter your habit Jogging
Jogging has been added to the weekly list

Habit Tracker
1. Add a habit
2. Remove a habit
3. Mark a habit
4. Exit

Enter your choice 1
Enter daily or weekly weekly
Enter your habit Swimming
Swimming has been added to the weekly list
```

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 a existing habit or to mark it respectively.

```
Habit Tracker
1. Add a habit
2. Remove a habit
3. Mark a habit
4. Exit

Enter your choice 2
Enter daily or weekly weekly
Remove your habit Swimming
Swimming has been removed from the weekly list

Habit Tracker
1. Add a habit
2. Remove a habit
3. Mark a habit
4. Exit

Enter your choice 3
Enter daily or weekly daily
Enter a habit you want to mark Eating
Eating has been marked for 2023-12-05
```

```
Habit Tracker Data:
1. List of all currently tracked habits
2. List of all habits with the same periodicity
3. Longest running streak of all habits
4. Longest running streak of a given habit
5. Exit
Enter your choice 1
Brushing (daily): None
Eating (daily): 2023-12-05
Jogging (weekly): None

Habit Tracker Data:
1. List of all currently tracked habits
2. List of all habits with the same periodicity
3. Longest running streak of all habits
4. Longest running streak of a given habit
5. Exit
Enter your choice 2
Habits for daily Brushing: None
Habits for daily Eating: 2023-12-05
Habits for weekly Jogging: None
```

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.

```
Habit Tracker Data:
1. List of all currently tracked habits
2. List of all habits with the same periodicity
3. Longest running streak of all habits
4. Longest running streak of a given habit
5. Exit
Enter your choice 3
Streak for Brushing: 1
Streak for Eating: 1
Streak for Jogging: 1

Habit Tracker Data:
1. List of all currently tracked habits
2. List of all habits with the same periodicity
3. Longest running streak of all habits
4. Longest running streak of a given habit
5. Exit
Enter your choice 4
Choose a habit Brushing
Streak for Brushing: 1
```