OBJECT ORIENTED AND FUNCTIONAL PROGRAMMING WITH PYTHON

CEM OGUZ

32008124

Design & Steps

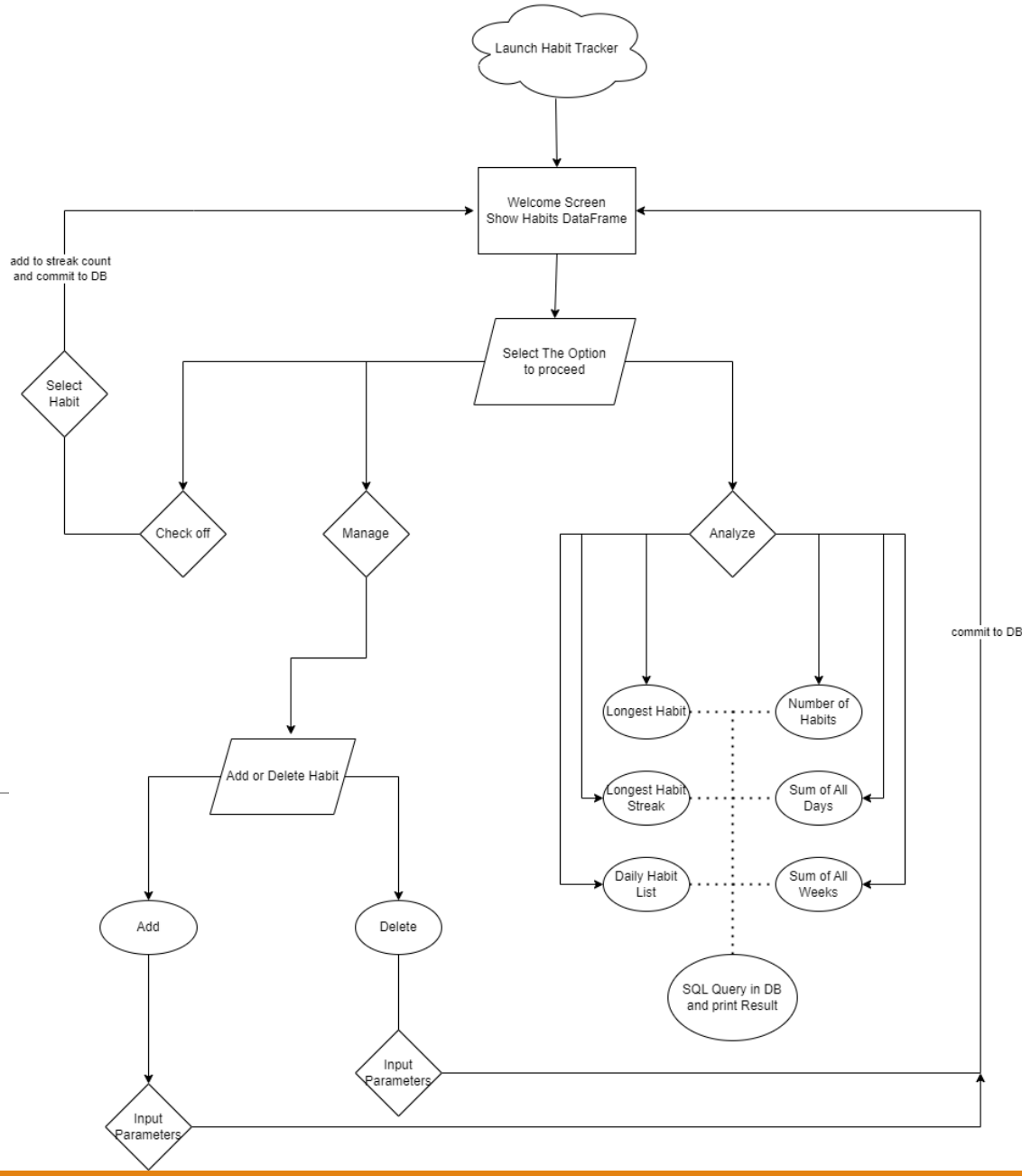
1- Display Data Frame

2- Ask the user the option to proceed

(Check off, Add, Delete, Analyse)

3- Perform the action & save changes

4- Go to Step 1



App Start - main.py

Running the main.py performs the codes on lines 98 and 99 as below. (which starts the app)

This mainMenu() function under the App class performs below (in order):

first displays the habit table with ShowData function

asks the user to make a selection. (as in the next slide)

```
91                  self.sql.deleteFromTable(f"{choice}")
92                  print(f"Successfully Deleted {choice} from your habit table"
93                  self.mainMenu()
94                  #-----------------------------------------------
95              if choiceHabits == "3":
96                  self.mainMenu()
97
98  if __name__ == "__main__":
99      App().mainMenu()  # Application starts running here.
```

Welcome Screen

When the app runs for the first time, below Data Frame is displayed.

habit	Description	Daily/Weekly	Date Started	Streak (Days)	Streak (Weeks)	Record	Last Updated
Code	code SQL for 2 hours	daily	2023-01-16	0	0	17	2023-04-01
Cook	cook turkish food	daily	2023-01-16	0	0	22	2023-04-02
Tennis	go tennis with friends	weekly	2023-01-16	7	1	14	2023-01-16
Grandma	visit grandma	weekly	2023-01-16	0	0	0	2023-04-02
Doctor	visit doctor for kids	weekly	2023-01-16	7	1	7	2023-01-16

```
/ WELCOME TO HABIT TRACKER \
| Please Select an Option  |
    (1) Check Off Habit
     (2) Manage Habits
     (3) Analyze Habits
         (4) Exit
```

SQL Class and main.py

When the user runs the main.py, init function creates an instance of SQL class (from SqliteHelper.py)

This creation is followed by running the init function inside the SQL class, which does the creation of the table in DB.

So, the application first fulfils the pre-condition of creating a table in DB.

Next slide shows further details

main.py SqliteHelper.py DataModel.py

```python
from DataFrame.DataModel import Data
from DataFrame.SqliteHelper import SQL


class App:
    """first class in main.py. Needs to be run for the application to start."""

    def __init__(self):
        self.sql = SQL()
```

SqliteHelper.py

SQL class connects and creates an empty table if it does not exist.

To do that, I import sqlite3 and use it to connect and create the table with SQL commands.
Name of the columns are: Habit, description, weekly_or_daily, date, streak_days, streak_weeks, record and last_update date .

main.py | SqliteHelper.py | DataModel.py

```python
import sqlite3 as sql
'''importing SQLite module'''


class SQL:
    """
    The main class for adding and deleting habits to habit table."""

    def __init__(self):
        self.conn = sql.connect("HabitTracker.db")
        self.cursor = self.conn.cursor()
        # We first create the table along with data types.
        self.cursor.execute("""CREATE TABLE IF NOT EXISTS habits (habit TEXT,desc TEXT,
        week_or_daily TEXT,date TEXT,streak_days INTEGER,streak_weeks INTEGER,record INTEGER,last_updated TEXT)""")
        self.conn.commit()
```

Classes

App Class (main.py)

Runs main menu, allows the user to check off, add and delete habits.

SQL Class (SqliteHelper.py)

Does the SQL part of the application. Like add/delete habits, adding habit streaks either daily or weekly, analyzing habits like longest habit, listing daily habits, number of habits, longest streak, sum of all days/weeks , and committing all of these transactions to DB.

Data Class (DataModel.py)

This class is to display and list the data we add or delete in the DB.

Visualization

habit	Description	Daily/Weekly	Date Started	Streak (Days)	Streak (Weeks)	Record	Last Updated
Code	code SQL for 2 hours	daily	2023-01-16	0	0	17	2023-04-01
Cook	cook turkish food	daily	2023-01-16	0	0	22	2023-04-02
Tennis	go tennis with friends	weekly	2023-01-16	7	1	14	2023-01-16
Grandma	visit grandma	weekly	2023-01-16	0	0	0	2023-04-02
Doctor	visit doctor for kids	weekly	2023-01-16	7	1	7	2023-01-16

I did not use unique identification numbers unlike most of the tables.
Instead , I used habit name itself as the index .

This part is coded in DataModel.py.
Pandas and tabulate modules are imported.

Adding & Displaying the Habits

Here, we simply add “workout” habit along with its other attributes shown in green.

```
                                / WELCOME TO HABIT TRACKER \
                                | Please Select an Option  |
                                    (1) Check Off Habit
                                     (2) Manage Habits
                                     (3) Analyze Habits
                                         (4) Exit

Your Choice: 2
                        What do you want to do on habit tracker?
                                    (1) Add a New Habit
                                     (2) Delete a Habit
                                   (3) Back to main menu

Your Choice: 1
Add Habits (you can go back by pressing 1 to continue, press any button to keep going):
Your habit name: workout
Enter a description for the habit: working out in my favorite gymcenter
Is the habit daily or weekly?: weekly
Start Date YYYY-MM-DD: 2023-01-01
WORKOUT HAS BEEN SUCCESSFULLY ADDED TO YOUR HABITS
```

Adding & Displaying Habits

When adding (or any other option) is completed, mainMenu() function runs to display the updated table.

In the table below, we see that workout is added and listed as 6th habit .

habit	Description	Daily/Weekly	Date Started	Streak (Days)	Streak (Weeks)	Record	Last Updated
Code	code SQL for 2 hours	daily	2023-01-16	0	0	17	2023-04-01
Cook	cook turkish food	daily	2023-01-16	0	0	22	2023-04-02
Tennis	go tennis with friends	weekly	2023-01-16	7	1	14	2023-01-16
Grandma	visit grandma	weekly	2023-01-16	0	0	0	2023-04-02
Doctor	visit doctor for kids	weekly	2023-01-16	7	1	7	2023-01-16
workout	working out in my favorite gymcenter	weekly	2023-01-01	0	0	0	2023-01-01

Analyzing Habits

Analyzing functions are under SqliteHelper.py with their names are on the left image.

And user interface is shown on the right image.

```
#BELOW FUNCTIONS ARE FOR ANALYZING.
    def findLongestHabit(self):...

    def findAllDailyHabits(self):...
    def findNumberOfHabits(self):...
    def longestHabitStreak(self):...
    def sumOfAllDays(self):...
    def sumOfAllWeeks(self):...
```

```
/ WELCOME TO HABIT TRACKER \
| Please Select an Option  |
    (1) Check Off Habit
     (2) Manage Habits
     (3) Analyze Habits
         (4) Exit

Your Choice: 3
(1) What's my longest habit?
(2) What's the list of my current daily habits?
(3) What's the number of habits
(4) What's my longest habit streak?
(5) What's the sum of all days in my tracker
(6) What's the sum of all weeks in my tracker

Please select a choice: 2
Your daily habits are:
['Code', 'Cook']
```

Streak

Adding streaks function is defined under addHabitStreak() function in SqliteHelper.py. It has some if blocks, for loops, else conditions etc. And some serious brain work is done under this function.

This part of coding is essential to comply with habit's previously added attributes such as :

A daily habit can be added a streak only if previous check-off was done no more than 1 day ago. Else, streak days count will not increase, and will be reset to 0.

Similar scenario applies for weekly habits too. A weekly habit can only be added a new streak only if last check off was done precisely 7 days ago. Else, streak weak count will not increase.

Deleting Habits

Deleting an habit is easy as deleting the entire habit row from the DB.

For this purpose user is simply prompted to enter the habit name, and that input is inserted into below SQL code.

In this sample, I deleted “workout” habit.

```
def deleteFromTable(self, habitName):
    '''deletes a habit from the table.'''
    self.cursor.execute(f"DELETE FROM habits WHERE habit='{habitName}'")
    self.conn.commit()
```

```
                    / WELCOME TO HABIT TRACKER \
                    | Please Select an Option  |
                        (1) Check Off Habit
                         (2) Manage Habits
                         (3) Analyze Habits
                             (4) Exit

Your Choice: 2
              What do you want to do on habit tracker?
                        (1) Add a New Habit
                         (2) Delete a Habit
                       (3) Back to main menu

Your Choice: 2
Which habit would you like to delete? (1 for main menu): workout
SUCCESSFULLY DELETED WORKOUT FROM YOUR HABIT TABLE
```

Records

Complying with the check off rules in previous slide will increase the streak days, and will be compared to record field at the same instance.

There is a separate block of code under addHabitStreak function shown as below.

```
#-----------------------------------------------------------------------------------------
        # This block checks if streak is larger than the record.
        self.cursor = self.conn.execute(f"SELECT record FROM habits WHERE habit = '{habitName}'")
        for row in self.cursor:
            if days > int(row[0]):
                self.cursor.execute(
                    f"UPDATE habits SET record = '{days}' WHERE habit = '{habitName}'")
                self.conn.commit()
        self.cursor.execute(f"UPDATE habits SET streak_days = '{days}' WHERE habit = '{habitName}'")
        self.conn.commit()
        #-------------------------------------------------------------------------
```