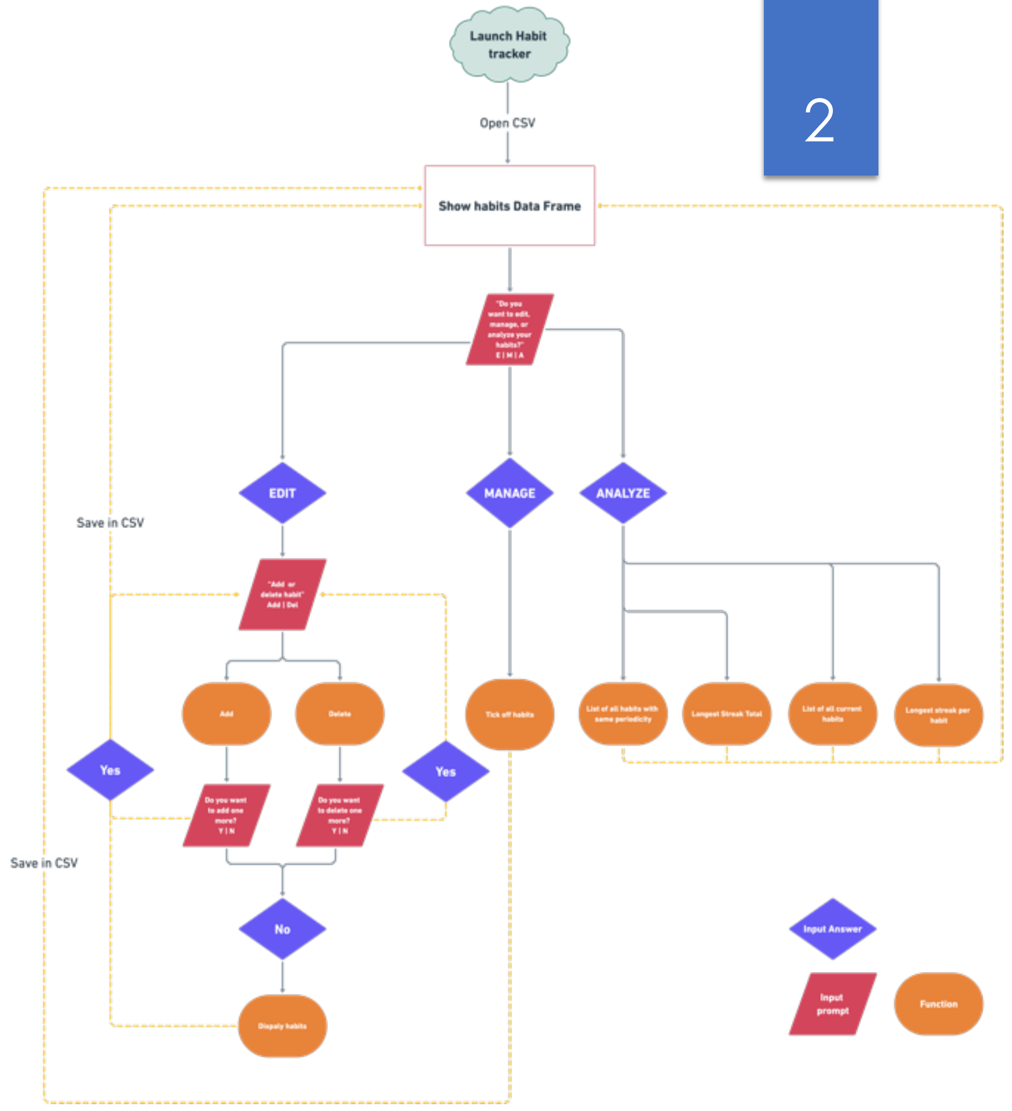
Object Oriented and Functional Programming with Python

BY EMRE ERTÜRK

17TH MARCH 2022

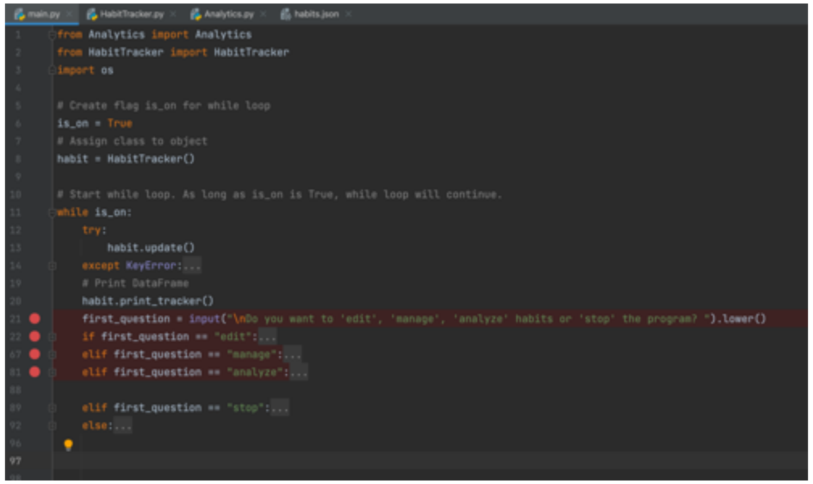
The design of the habit tracker app: **Step-by-step**

1. Shows Data Frame in the beginning
2. Asks user to edit, manage, or analyze habits
3. Performs actions in each path until user wants to go back or finish program

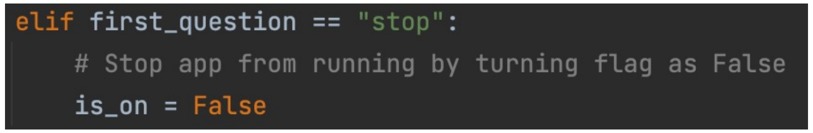

Launch Habit tracker
Open CSV
Show habits Data Frame
"Do you want to edit, manage, or analyze your habits?" E | M | A
EDIT
MANAGE
ANALYZE
Save in CSV
"Add or delete habit" Add | Del
Add
Delete
Tick off habits
List of all habits with same periodicity
Longest Streak Total
List of all current habits
Longest streak per habit
Yes
Yes
Do you want to add one more? Y | N
Do you want to delete one more? Y | N
Save in CSV
No
Dispaly habits
Input Answer
Input prompt
Function

Main code

- Design calls for while loop
  - Flag: is_on
  - While **is_on** is **True** the app will run
  - While loop terminates when user types **stop** (is_on == False)
    - Was missing in flowchart
- "Brain work" is done in classes **HabitTracker** and **Analytics**
- **Imports**: OS



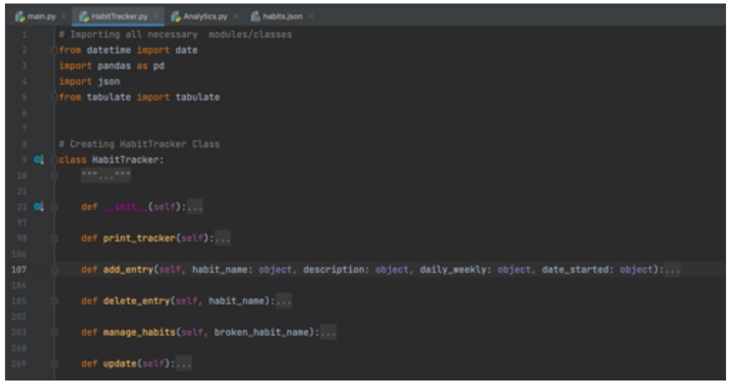
```
from Analytics import Analytics
from HabitTracker import HabitTracker
import os

# Create flag is_on for while loop
is_on = True
# Assign class to object
habit = HabitTracker()

# Start while loop. As long as is_on is True, while loop will continue.
while is_on:
    try:
        habit.update()
    except KeyError:...
    # Print DataFrame
    habit.print_tracker()
    first_question = input("\nDo you want to 'edit', 'manage', 'analyze' habits or 'stop' the program? ").lower()
    if first_question == "edit":...
    elif first_question == "manage":...
    elif first_question == "analyze":...

    elif first_question == "stop":...
    else:...
```



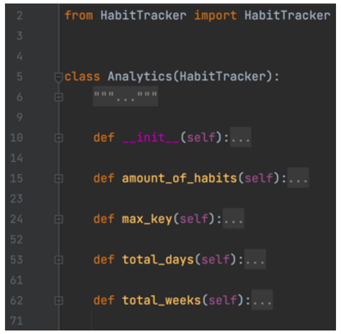
```
elif first_question == "stop":
    # Stop app from running by turning flag as False
    is_on = False
```

Classes: HabitTracker & Analytics

- Why classes?
  - Less repetitions of code
  - Easier to read
- **HabitTracker**:
  - To print table in Terminal for user
  - To compute all entries in DataFrame
  - Methods: print habit tracker data frame, add new entries, delete entries, manage habits, and update(refresh) dataframe
- **Analytics**:
  - Compute all extra Analytics of interest
  - Methods: compute amount of habits tracked, longest habit, total days tracked in days and weeks
- **Imports**: Datetime, Pandas, JSON, tabulate



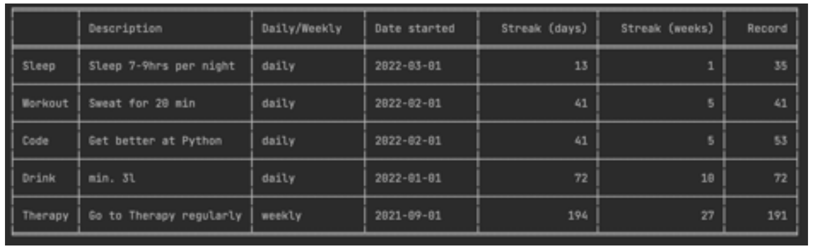
```
# Importing all necessary  modules/classes
from datetime import date
import pandas as pd
import json
from tabulate import tabulate


# Creating HabitTracker Class
class HabitTracker:
    """..."""

    def __init__(self):...

    def print_tracker(self):...

    def add_entry(self, habit_name: object, description: object, daily_weekly: object, date_started: object):...

    def delete_entry(self, habit_name):...

    def manage_habits(self, broken_habit_name):...

    def update(self):...
```



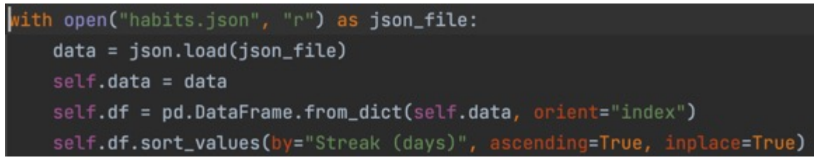
```
from HabitTracker import HabitTracker


class Analytics(HabitTracker):
    """..."""

    def __init__(self):...

    def amount_of_habits(self):...

    def max_key(self):...

    def total_days(self):...

    def total_weeks(self):...

```

Visualization

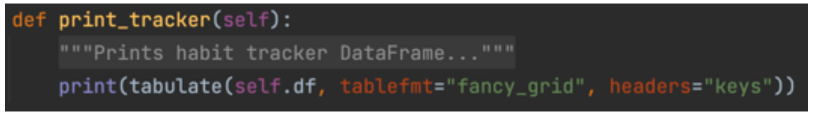
- The goal was to create a Data Frame object that includes all habits
- Data Frame characteristics
  - **Index**: habit
  - **Columns**: Descriptions, Daily or Weekly Tracking, Date Started, Streak in days, Streak in weeks, and Record in days
  - **Context Manager** to store data frame
  - **Pandas** to compute table
  - **Tabulate** to prettify the table
  - Sorted by number of 'Streaks (days)'
  - Printed by method in 'HabitTracker' class



	Description	Daily/Weekly	Date started	Streak (days)	Streak (weeks)	Record
Sleep	Sleep 7-9hrs per night	daily	2022-03-01	13	1	35
Workout	Sweat for 20 min	daily	2022-02-01	41	5	41
Code	Get better at Python	daily	2022-02-01	41	5	53
Drink	min. 3l	daily	2022-01-01	72	10	72
Therapy	Go to Therapy regularly	weekly	2021-09-01	194	27	191



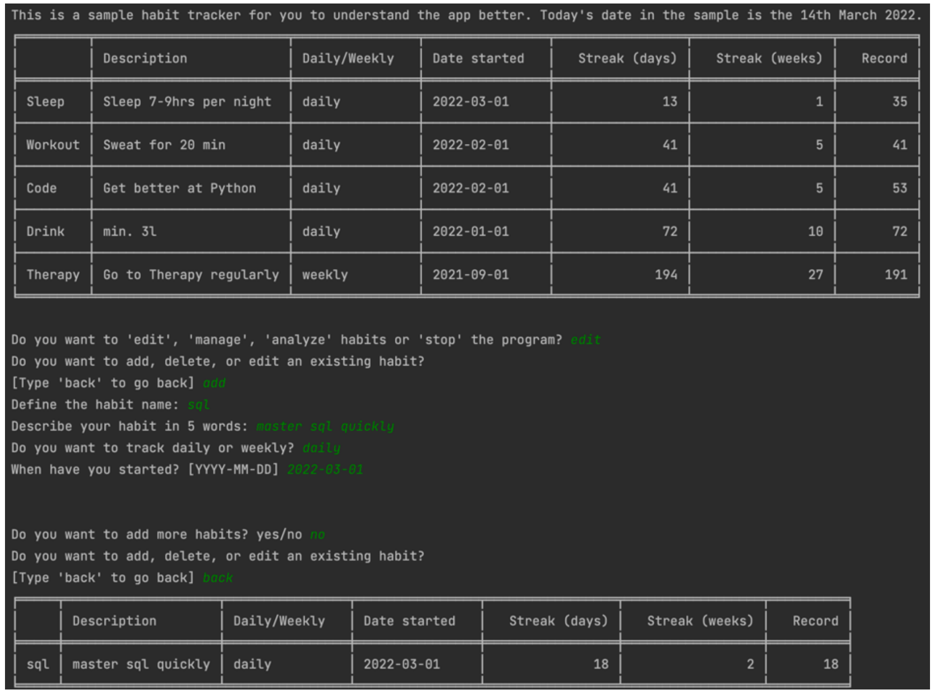
```python
with open("habits.json", "r") as json_file:
    data = json.load(json_file)
    self.data = data
    self.df = pd.DataFrame.from_dict(self.data, orient="index")
    self.df.sort_values(by="Streak (days)", ascending=True, inplace=True)
```



```python
def print_tracker(self):
    """Prints habit tracker DataFrame..."""
    print(tabulate(self.df, tablefmt="fancy_grid", headers="keys"))
```

Adding habits

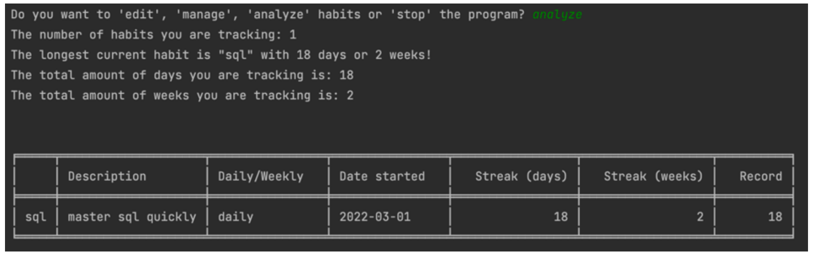
- User starts with sample habit tracker
- User manages program by answering questions in the Terminal.
  - Questions in white
  - Answers in green
- Once entry by user is made, sample habit tracker disappears and only users is visible.
- Most important analytics in Data Frame already



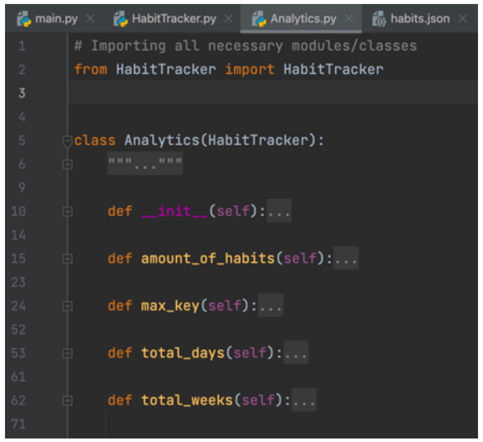
```
This is a sample habit tracker for you to understand the app better. Today's date in the sample is the 14th March 2022.
|         | Description            | Daily/Weekly | Date started | Streak (days) | Streak (weeks) | Record |
|---------|------------------------|--------------|--------------|---------------|----------------|--------|
| Sleep   | Sleep 7-9hrs per night | daily        | 2022-03-01   |            13 |              1 |     35 |
| Workout | Sweat for 20 min       | daily        | 2022-02-01   |            41 |              5 |     41 |
| Code    | Get better at Python   | daily        | 2022-02-01   |            41 |              5 |     53 |
| Drink   | min. 3l                | daily        | 2022-01-01   |            72 |             10 |     72 |
| Therapy | Go to Therapy regularly | weekly      | 2021-09-01   |           194 |             27 |    191 |

Do you want to 'edit', 'manage', 'analyze' habits or 'stop' the program? edit
Do you want to add, delete, or edit an existing habit?
[Type 'back' to go back] add
Define the habit name: sql
Describe your habit in 5 words: master sql quickly
Do you want to track daily or weekly? daily
When have you started? [YYYY-MM-DD] 2022-03-01


Do you want to add more habits? yes/no no
Do you want to add, delete, or edit an existing habit?
[Type 'back' to go back] back
|     | Description        | Daily/Weekly | Date started | Streak (days) | Streak (weeks) | Record |
|-----|--------------------|--------------|--------------|---------------|----------------|--------|
| sql | master sql quickly | daily        | 2022-03-01   |            18 |              2 |     18 |
```

Analyze habits

- Analytics Module a short summary of interesting habit data
    1. Number of habits tracked
    2. Longest current habit
    3. Total amount of days
    4. Total amount of weeks
- Computation via **Analytics** Child-Class
    - Based on **HabitTracker** class



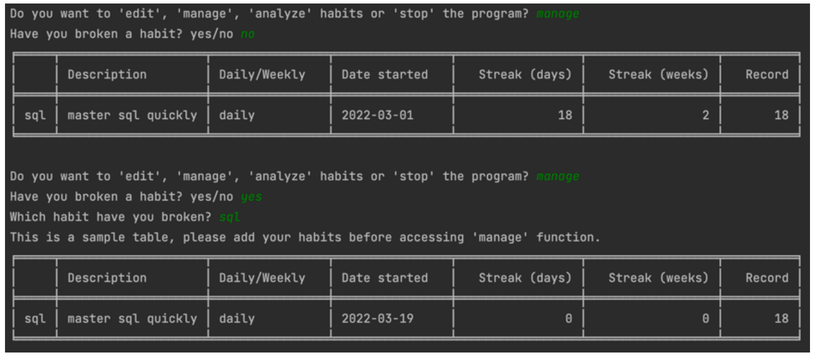
```
Do you want to 'edit', 'manage', 'analyze' habits or 'stop' the program? analyze
The number of habits you are tracking: 1
The longest current habit is "sql" with 18 days or 2 weeks!
The total amount of days you are tracking is: 18
The total amount of weeks you are tracking is: 2


|     | Description        | Daily/Weekly | Date started | Streak (days) | Streak (weeks) | Record |
|-----|--------------------|--------------|--------------|---------------|----------------|--------|
| sql | master sql quickly | daily        | 2022-03-01   |            18 |              2 |     18 |
```



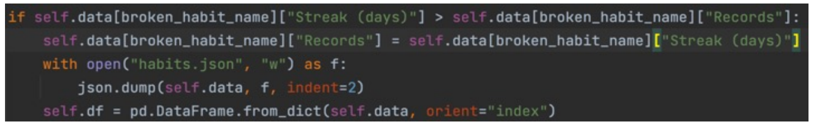
```python
# Importing all necessary modules/classes
from HabitTracker import HabitTracker


class Analytics(HabitTracker):
    """..."""

    def __init__(self):...

    def amount_of_habits(self):...

    def max_key(self):...

    def total_days(self):...

    def total_weeks(self):...
```

Manage habits

- If habit wasn't broken than nothing happens
- If habit was broken the **date** gets reset to today and the current **Streak** (days and weeks) is computed to 0
- Notice how the **Record** is kept at 18 while all the other gets reset
  - Record only changes when the Streak (days) exceeds the previous one



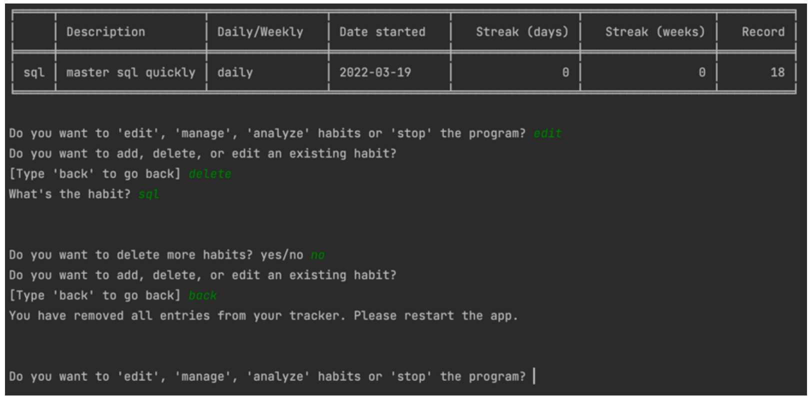
```
Do you want to 'edit', 'manage', 'analyze' habits or 'stop' the program? manage
Have you broken a habit? yes/no no

|     | Description        | Daily/Weekly | Date started |  Streak (days) |  Streak (weeks) |  Record |
|-----|--------------------|--------------|--------------|----------------|-----------------|---------|
| sql | master sql quickly | daily        | 2022-03-01   |             18 |               2 |      18 |

Do you want to 'edit', 'manage', 'analyze' habits or 'stop' the program? manage
Have you broken a habit? yes/no yes
Which habit have you broken? sql
This is a sample table, please add your habits before accessing 'manage' function.

|     | Description        | Daily/Weekly | Date started |  Streak (days) |  Streak (weeks) |  Record |
|-----|--------------------|--------------|--------------|----------------|-----------------|---------|
| sql | master sql quickly | daily        | 2022-03-19   |              0 |               0 |      18 |
```



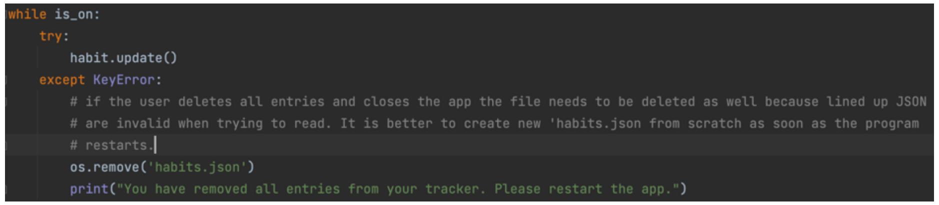
```python
if self.data[broken_habit_name]["Streak (days)"] > self.data[broken_habit_name]["Records"]:
    self.data[broken_habit_name]["Records"] = self.data[broken_habit_name]["Streak (days)"]
    with open("habits.json", "w") as f:
        json.dump(self.data, f, indent=2)
    self.df = pd.DataFrame.from_dict(self.data, orient="index")
```

Delete habits

- User can delete app by going in edit → delete → habit name
- If user deletes all habits, user is asked to restart the app other wise a KeyError will rise
  - At the very beginning of main loop the code catches that there are no habits in JSON
  - Normally that's where habit Data Frame refreshes
  - But if there are no entries in JSON the keys will no be found
  - That is why "habits.json" needs to be deleted entirely and the program restarted



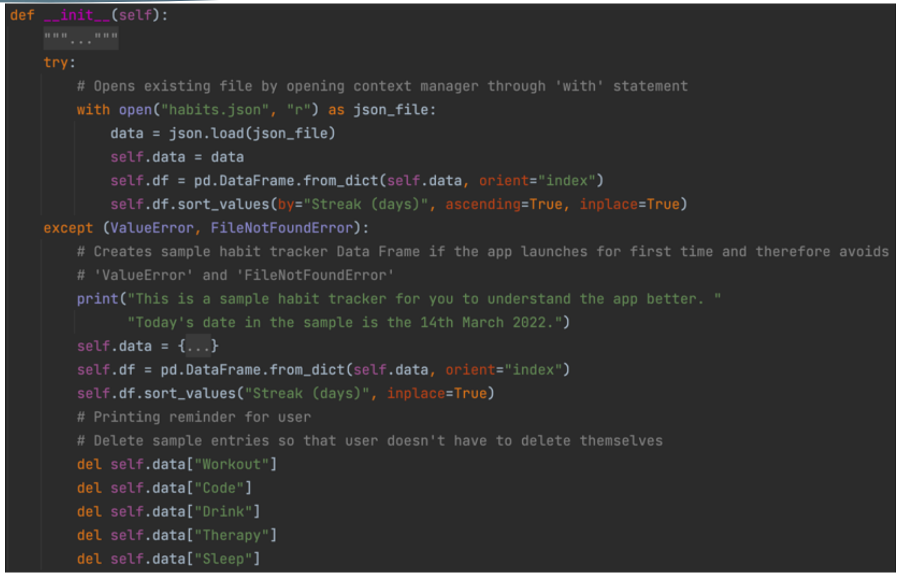
```
|     | Description        | Daily/Weekly | Date started |   Streak (days) |   Streak (weeks) |   Record |
|-----|--------------------|--------------|--------------|-----------------|------------------|----------|
| sql | master sql quickly | daily        | 2022-03-19   |               0 |                0 |       18 |

Do you want to 'edit', 'manage', 'analyze' habits or 'stop' the program? edit
Do you want to add, delete, or edit an existing habit?
[Type 'back' to go back] delete
What's the habit? sql


Do you want to delete more habits? yes/no no
Do you want to add, delete, or edit an existing habit?
[Type 'back' to go back] back
You have removed all entries from your tracker. Please restart the app.


Do you want to 'edit', 'manage', 'analyze' habits or 'stop' the program?
```



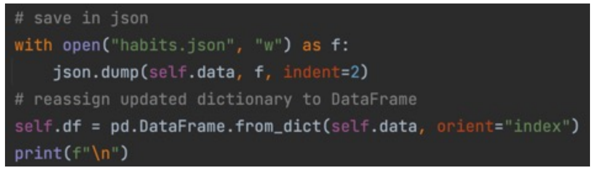
```python
while is_on:
    try:
        habit.update()
    except KeyError:
        # if the user deletes all entries and closes the app the file needs to be deleted as well because lined up JSON
        # are invalid when trying to read. It is better to create new 'habits.json from scratch as soon as the program
        # restarts.
        os.remove('habits.json')
        print("You have removed all entries from your tracker. Please restart the app.")
```

Saving habit data in file

- Initially CSV was intended for use but the shortcomings of CSV files for this project were too many
  - Switch to JSON files
- In HabitTracker class the JSON is loaded in try sections only if the user has used the app before
- If the user is newly opening the app or accidentally deleted all records (except errors), a temporary sample data frame is created
- New JSON is created when user adds a new habit



```python
def __init__(self):
    """..."""
    try:
        # Opens existing file by opening context manager through 'with' statement
        with open("habits.json", "r") as json_file:
            data = json.load(json_file)
            self.data = data
            self.df = pd.DataFrame.from_dict(self.data, orient="index")
            self.df.sort_values(by="Streak (days)", ascending=True, inplace=True)
    except (ValueError, FileNotFoundError):
        # Creates sample habit tracker Data Frame if the app launches for first time and therefore avoids
        # 'ValueError' and 'FileNotFoundError'
        print("This is a sample habit tracker for you to understand the app better. "
              "Today's date in the sample is the 14th March 2022.")
        self.data = {...}
        self.df = pd.DataFrame.from_dict(self.data, orient="index")
        self.df.sort_values("Streak (days)", inplace=True)
        # Printing reminder for user
        # Delete sample entries so that user doesn't have to delete themselves
        del self.data["Workout"]
        del self.data["Code"]
        del self.data["Drink"]
        del self.data["Therapy"]
        del self.data["Sleep"]
```



```python
# save in json
with open("habits.json", "w") as f:
    json.dump(self.data, f, indent=2)
# reassign updated dictionary to DataFrame
self.df = pd.DataFrame.from_dict(self.data, orient="index")
print(f"\n")
```

Changes: Phase 2 vs Phase 1

- Habit Tracker Data Frame includes most important Analytics including Records column
- The limitations of CSV files were too many
  - That’s why JSON file type was chosen
- Using two Classes turned out to be a good decision but a Parent-Child relationship wasn’t thought of before