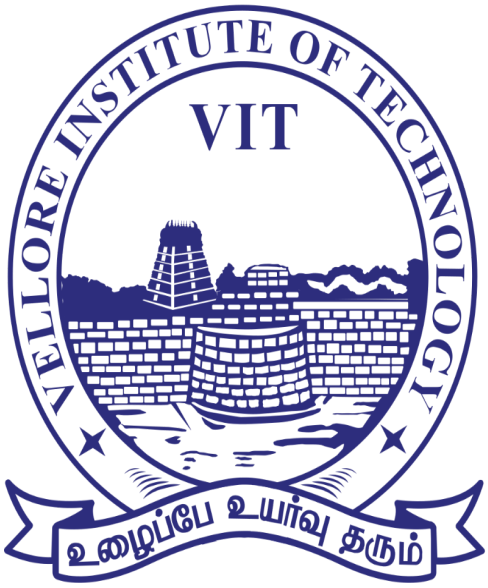

VIT®

BHOPAL

HABIT TRACKER USING PYTHON

Mini Project Report

Submitted By: Shivansh Sharma
Reg. No.: 25BCE10121
Under The Guidance Of:
Dr. Sandip Mal

Index

1. Problem Statement

The aim of this project is to develop a terminal-based Habit Tracking System using Python, which
allows users to add habits, log progress, calculate streaks, generate weekly and monthly statistics,
and maintain long-term consistency using a local
JSON-based database.

2. Objectives

- Create a habit management system using basic Python.
- Provide daily logging and date-based progress entry.
- Calculate streaks and longest streaks automatically.
- Generate weekly and monthly statistics using NumPy.
- Store all data permanently in a JSON file.
- Keep the system simple, clean and user-friendly.

4. Working & Explanation

The system works entirely through terminal inputs. Users can add habits with daily targets, log
values with dates, view detailed statistics, and reset or remove habits. NumPy is used for statistical
calculations such as mean, sum, min, and max values. JSON ensures long-term storage without
requiring any external database.

5. Full Source Code

```
import json, os, time, random
from datetime import date, datetime, timedelta
import numpy as np

DATA_FILE = "habit_data.json"

MOTIVATION = [
   "Okay nice, done for today.",
   "Good… at least you logged something.",
   "Keeping it moving, not bad.",
   "Alright, that counts too.",
   "Streak stays alive ig.",
   "Cool. Next."
]

def get_int(msg, minimum=None, maximum=None):
   while True:
       raw = input(msg).strip()
       try:
           x = int(raw)
           if minimum is not None and x < minimum:
               print("That's lower than allowed.")
               continue
           if maximum is not None and x > maximum:
               print("That's too high bro.")
               continue
           return x
       except:
           print("Put a number only… please.")

def get_text(msg):
   while True:
       s = input(msg).strip()
       if s:
           return s
```

```
        print("Don't leave it blank, it breaks stuff.")

def get_date(msg):
    while True:
        r = input(f"{msg} (YYYY-MM-DD) or press enter: ").strip()
        if r == "":
            return date.today()
        try:
            return datetime.strptime(r, "%Y-%m-%d").date()
        except:
            print("Wrong format. Try again.")

def load_data():
    if not os.path.exists(DATA_FILE):
        return {"habits": {}, "logs": []}
    return json.load(open(DATA_FILE))

def save_data(d):
    json.dump(d, open(DATA_FILE, "w"), indent=2, default=str)

def add_habit():
    d = load_data()
    name = get_text("Habit name: ")
    if name in d["habits"]:
        print("Already made this one.")
        return
    target = get_int("Daily target: ", minimum=1)
    d["habits"][name] = {"target": target}
    save_data(d)
    print("Added it.")

def remove_habit():
    d = load_data()
    name = get_text("Delete habit: ")
    if name not in d["habits"]:
        print("Nothing with that name.")
        return
    del d["habits"][name]
    d["logs"] = [x for x in d["logs"] if x["habit"] != name]
    save_data(d)
    print("Removed.")
```

```
def log_habit():
    d = load_data()
    name = get_text("Habit name: ")
    if name not in d["habits"]:
        print("No such habit exists.")
        return
    dt = get_date("Date")
    val = get_int("Value: ", minimum=0)
    d["logs"].append({
        "habit": name,
        "value": val,
        "date": dt.isoformat()
    })
    save_data(d)
    print(random.choice(MOTIVATION))

def habit_logs(name):
    d = load_data()
    return [x for x in d["logs"] if x["habit"] == name]

def totals_per_day(logs):
    res = {}
    for row in logs:
        d = datetime.strptime(row["date"], "%Y-%m-%d").date()
        res[d] = res.get(d, 0) + row["value"]
    return res

def current_streak(name):
    d = load_data()
    target = d["habits"][name]["target"]
    logs = habit_logs(name)
    per = totals_per_day(logs)
    streak = 0
    day = date.today()
    while per.get(day, 0) >= target:
        streak += 1
        day -= timedelta(days=1)
    return streak

def longest_streak(name):
    logs = habit_logs(name)
    if not logs:
```

```
        return 0
    d = load_data()
    target = d["habits"][name]["target"]
    per = totals_per_day(logs)
    days = sorted(per)
    longest = 0
    curr = 0
    prev = None
    for day in days:
        if per[day] >= target:
            if prev and (day - prev).days == 1:
                curr += 1
            else:
                curr = 1
            if curr > longest:
                longest = curr
        else:
            curr = 0
        prev = day
    return longest

def last_n_days(name, n):
    logs = habit_logs(name)
    per = totals_per_day(logs)
    today = date.today()
    vals = []
    for i in range(n):
        d = today - timedelta(days=(n-1-i))
        vals.append(per.get(d, 0))
    arr = np.array(vals)
    return {
        "counts": vals,
        "sum": int(arr.sum()),
        "mean": float(arr.mean()),
        "max": int(arr.max()),
        "min": int(arr.min())
    }

def view_stats():
    d = load_data()
    name = get_text("Habit name: ")
    if name not in d["habits"]:
```

```
        print("Not found.")
        return
    print("\n--- Habit Summary ---")
    print("Habit:", name)
    print("Daily target:", d["habits"][name]["target"])
    print("Current streak:", current_streak(name))
    print("Longest streak:", longest_streak(name))
    wk = last_n_days(name, 7)
    print("\nLast 7 days:", wk["counts"], "| Total:", wk["sum"])
    mn = last_n_days(name, 30)
    print("Last 30 days total:", mn["sum"], "| Mean:", round(mn["mean"], 2))
    if input("Show logs? (y/n): ").lower().strip() == "y":
        for row in sorted(habit_logs(name), key=lambda x: x["date"]):
            print(row["date"], "->", row["value"])

def show_all():
    d = load_data()
    if not d["habits"]:
        print("Nothing added yet.")
        return
    print("\n--- All Habits ---")
    for name in d["habits"]:
        logs = habit_logs(name)
        total = sum(l["value"] for l in logs)
        print(name, "| target:", d["habits"][name]["target"], "| logs:", len(logs), "|
total:", total)

def reset_habit():
    d = load_data()
    name = get_text("Reset habit: ")
    if name not in d["habits"]:
        print("No such habit.")
        return
    confirm = input("Type YES to wipe logs: ").strip()
    if confirm == "YES":
        d["logs"] = [x for x in d["logs"] if x["habit"] != name]
        save_data(d)
        print("Reset done.")
    else:
        print("Okay cancelled.")

def menu():
```

```
    while True:
        print("""
==== HABIT TRACKER ====
1 Add Habit
2 Remove Habit
3 Log Habit
4 View Stats
5 Show All Habits
6 Reset Logs
7 Exit
""")
        ch = input("Choice: ").strip()
        if ch == "1": add_habit()
        elif ch == "2": remove_habit()
        elif ch == "3": log_habit()
        elif ch == "4": view_stats()
        elif ch == "5": show_all()
        elif ch == "6": reset_habit()
        elif ch == "7":
            print("Alright, closing.")
            time.sleep(0.4)
            break
        else:
            print("Try again.")

if __name__ == "__main__":
    menu()
```

7. Conclusion

This project successfully demonstrates the use of Python for creating a real-world application. The
Habit Tracker covers data storage, input validation, statistics, streak calculations, and JSON-based
persistence. It is simple, lightweight, and useful for daily productivity.

8. References

• Python Official Documentation — https://docs.python.org
• NumPy Documentation — https://numpy.org
• ReportLab PDF Library — https://www.reportlab.com
• Stack Overflow (general debugging help)