PRACTICE SHEET

What you will learn:

File Handling & Automation, API Integration, Graphs, Data Visualization, Linux & Shell Commands, Markdown & HTML Conversion, Web Development (Flask), QR Code Generation, CSV Handling

---

[EASY] Q1) Suppose, user x has used a text editing software to type some text. After saving the article as WORDS.TXT, she realised that she had wrongly typed alphabet J in place of alphabet I everywhere in the article. Write a function definition for JTOI() in Python that would display the corrected version of the entire content of the file WORDS.TXT with all the alphabets "J" to be displayed as an alphabet "I" on screen. Then it writes this corrected content back to the WORDS.TXT file.

Example:

If x has stored the following content in the file WORDS.TXT:

WELL, THJS JS A WORD BY JTSELF. YOU COULD STRETCH THJS TO BE A SENTENCE

The function JTOI() should display the following content:

WELL, THIS IS A WORD BY ITSELF. YOU COULD STRETCH THIS TO BE A SENTENCE

[MEDIUM] Q2) You are given a file named links.txt that contains a list of **bidirectional edges** in an undirected graph. Each line of the file contains two node labels separated by a comma, representing a connection (edge) between two nodes.

Example contents of links.txt:

a,b

b,c

a,d

This represents the following undirected graph:

a -- b -- c

\

d

Write a Python program that:

1. Reads the links.txt file.
2. Constructs an undirected graph using the file data.
3. Calculates the **degree** of each node (i.e., how many neighbors it has).
4. Ranks and displays the nodes in **descending order of their degree**.
5. If **two nodes have the same degree**, they should be ranked **alphabetically**.

[MEDIUM] Q3) Write a program which creates a directory “mydir” in the current folder you are. Then move inside “mydir” from the program itself. Create a file “a.txt” there. Then, write the following three lines to the a.txt file:

First line

second line

third line

Close the file. Rename the file to “myfile.txt”. Open “myfile.txt” and replace the second line with the line: "sorry! The content of this line has been changed! " without touching (reading/writing) the first line.

[HARD] Q4) You are a rookie programmer recruited by a futuristic gaming corporation - MetaFight Studios. They've built a simulation game similar to Free Fire. A major bug has crashed the vest system of all players, and now it's your job to **rescue the saved game files** and **visualize their armor status**.

The **player data is stored in a text file**, but due to the bug, some entries are **corrupted or missing**. You must write a Python program to **read, validate, correct**, and **visualize** this data.

Your Mission:

1) You are given a file **players.txt** with data in the following format (each line represents one player). Your program must:

- Read this file.
- Handle errors (like missing or incorrect data).

- Use a **default vest level = 1** and **default health = 100** if data is missing or wrong.
- Calculate **vest HP** based on the following:

Level 1 = 25 HP

Level 2 = 50 HP

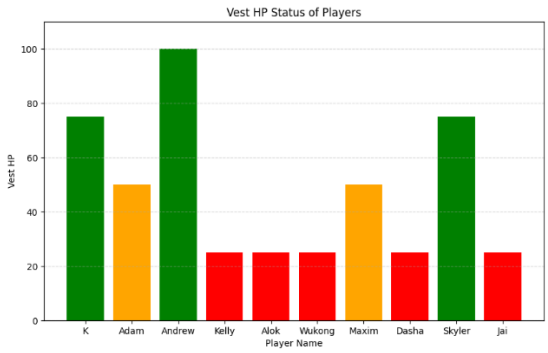
Level 3 = 75 HP

Level 4 = 100 HP

2) Store the **cleaned and corrected data** into a new file called cleaned_players.txt.

3) Then, **visualize** the player names and their corresponding **vest HP** using a **bar graph** (Matplotlib or Turtle).

The corrupted input file is given as **players.csv**



[MEDIUM] Q5) Write a Python program that searches for a file, obtains its size and reports the size in bytes/KB/MB/GB/TB as appropriate.

[MEDIUM] Q6) Write a Python program that reports the time of creation, time of last access and time of last modification for a given file.

Hint: Functions getctime(), getmtime() and getatime() return the creation, modification, and access times for the given file.
These times are returned as the number of seconds since the epoch.
Epoch is considered to be 1st Jan 1970, 00:00:00.

ctime() function of the time module converts the time expressed in seconds since epoch into a string representing local time.

[EASY] Q7) Which command is used to create an empty file in Linux?

A. new
B. vi
C. mkdir
D. touch

[EASY] Q8) Which command shows hidden files in a directory?

A. ls -l
B. ls -h
C. ls -a
D. ls -x

[EASY] Q9) What does rm -r myfolder/ do?

A. Renames the folder
B. Recursively deletes the folder
C. Creates a backup
D. Removes hidden files only

[EASY] Q10) You are the system admin of **IIT Bhilai**, and your task is to organize folders and files for three departments: CSE, ECE, and MECH. Each department should have the following structure:

/IIT

└── CSE

| └── timetable.txt

└── ECE

| └── timetable.txt

└── MECH

└── timetable.txt

Then, move the **CSE timetable** to a shared folder called Common, rename it as cse_tt.txt, and finally, delete the MECH folder entirely.

Write the Linux commands step by step to:

1. Create the /IIT directory and subdirectories.
2. Add a file named timetable.txt in each department folder.
3. Create a Common folder and move + rename the CSE timetable.
4. Delete the MECH folder completely.

Expected Linux Commands:

mkdir -p IIT/CSE IIT/ECE IIT/MECH

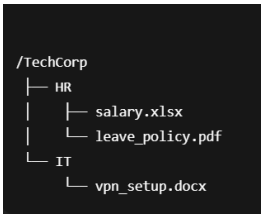
touch IIT/CSE/timetable.txt IIT/ECE/timetable.txt IIT/MECH/timetable.txt

mkdir IIT/Common

mv IIT/CSE/timetable.txt IIT/Common/cse_tt.txt

rm -r IIT/MECH

[MEDIUM] Q11) You are working at **TechCorp Pvt Ltd**, and you've been asked to organize the HR and IT documents. Initially, create the following structure:



Then perform the following tasks:

1. Copy vpn_setup.docx to HR folder.
2. Rename leave_policy.pdf to hr_policy.pdf.
3. List all files under /TechCorp/HR with detailed info.
4. Print the absolute path where you are right now.
5. Delete salary.xlsx.

Write the Linux commands for the above steps.

[HARD] Q12) **Markdown → HTML Converter**

You are given a file named notes.md that contains text written in a simplified form of Markdown. Your task is to **convert the Markdown content into HTML**, applying only the following formatting rules:

Conversion Rules:

Markdown Syntax	**HTML Output**
# Heading	Heading
bold text	bold text
[link text](url)	link text

Write a Python program that:

1. Reads the content of notes.md.
2. Converts all headings, bold texts, and markdown-style links to HTML using string manipulation.
3. Saves the output as output.html.
4. The final HTML file should be well-structured with <html>, <head>, and <body> tags.

[HARD] Q13) **JSON Contact Book Manager**

Design a **Python-based Contact Book** that stores contact details using the **JSON file format**, which is widely used for data storage in real-world APIs and applications.

Features to Implement: Your program should allow the user to:

1. **Add a new contact** with the following fields:
    - Name
    - Phone Number
    - Email
2. **View all contacts**, sorted **alphabetically by name**.

3. **Exit the program**, saving all contacts persistently in a file called contacts.json. If the file already exists, load its content. If not, start with an empty contact book.

Hints:

- Use json.load() to read existing contacts from contacts.json at the start.
- Store contacts in a Python dictionary like:

```
{
  "Alice": {"phone": "1234567890", "email": "alice@example.com"},
  "Bob": {"phone": "9876543210", "email": "bob@example.com"}
}
```

- To sort contacts alphabetically, use: for name in sorted(contacts):

Sample Output:

--- Contact Book Menu ---

1. Add new contact

2. View all contacts

3. Exit

Enter your choice: 1

Enter name: Alice

Enter phone: 9876543210

Enter email: alice@example.com

Contact added!

--- Contact Book Menu ---

1. Add new contact

2. View all contacts

3. Exit

Enter your choice: 2

Contact List (sorted):

Alice - 9876543210 - alice@example.com

[HARD] Q14) Write a Python program that:

1. Asks the user to enter the name of a **city**.
2. Fetches the **current weather data** for that city using the **OpenWeatherMap API**.
3. Displays:
    - Temperature in Celsius
    - Weather condition (e.g., "clear sky", "rain")
    - Wind speed

Setup:

1. Create a **free account** on https://openweathermap.org/
2. Get your **API key** from the dashboard.
3. Replace YOUR_API_KEY_HERE in the code with your actual key.

Hints:

- Use requests.get() to call: https://api.openweathermap.org/data/2.5/weather?q={city}&appid={API_KEY}&units=metric
- Use .json() method to convert the response into a Python dictionary.
- Use: data['main']['temp'], data['weather'][0]['description'], data['wind']['speed']

Sample Output:

Enter city: Delhi

City: Delhi

Temperature: 34.56°C

Condition: haze

Wind Speed: 2.1 m/s

[MEDIUM] Q15) A hacker-turned-entrepreneur named **Cipher Singh** is building a secure way to share contact info at tech events. Instead of sharing paper or typing long emails or URLs, Cipher wants to **generate a QR code** that can be scanned to access an email or a website instantly.

He needs your help to automate this using **Python**. You're his tech intern.Build a program that:

1. Asks the user to enter a **valid email address** or a **website URL**.
2. Generates a QR code containing that data.
3. Saves the QR code image as output.png in the current folder.
4. Displays a confirmation message on the screen like:
   QR Code saved as output.png

Hints:

- Install the module (if not already):

pip install qrcode[pil]

- Use:

import qrcode

img = qrcode.make("your data here")

img.save("output.png")

Sample Output:

=== QR Code Generator ===

Enter your email or URL: https://github.com/ciphersingh

QR Code saved as output.png

[HARD] Q16) You're helping your friend **Nira**, who often forgets her thoughts while working. She asked you to create a **tiny personal web notepad** where she can:

1. Type quick thoughts on a web page
2. Submit them via a form
3. Store them in a file (notes.txt)
4. Visit another page to **see all her notes**

As a Python developer, you'll build this using **Flask**, a lightweight web framework.

Build a Flask web app that has two pages:

1. / – Note Entry Page:

- A simple web form with a <textarea> and a **submit** button.
- On submission, the note should be **appended to notes.txt**.

2. /view – View Notes Page:

- Shows all notes (line by line) saved in notes.txt.

Folder Structure:

/project-folder

└── app.py

└── templates/

  └── index.html

  └── view.html

Hints:

- Use Flask, render_template, and request.form
- Create a templates folder for HTML files
- Use POST method to submit the form
- Use Python file handling (with open(...)) to read/write to notes.txt

Starter Code: app.py file content

```
from flask import Flask, render_template, request, redirect
```

```
app = Flask(__name__)
@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        note = request.form['note']
        if note.strip():
            with open('notes.txt', 'a') as f:
                f.write(note.strip() + '\n')
        return redirect('/')
    return render_template('index.html')
@app.route('/view')
def view_notes():
    try:
        with open('notes.txt', 'r') as f:
            notes = f.readlines()
    except FileNotFoundError:
        notes = []
    return render_template('view.html', notes=notes)
if __name__ == '__main__':
    app.run(debug=True)
```

To Run the App:

pip install flask

python app.py

Then open: http://127.0.0.1:5000/ in your browser.

[MEDIUM] Q17) Your friend Kalash is a first-year college student who's always low on cash. Every month, he promises to **track his expenses**, but ends up spending without realizing how much he's blowing through.

One day, Kalash asks **you** to create a simple **Python-based expense tracker** that he can use from his laptop - nothing fancy, just something that:

1. Records what he bought and how much he paid
2. Calculates total money spent
3. Tells him the average expense
4. Warns him of any big expenses (over ₹1000)

You decide to save the data in a **CSV file** called expenses.csv so that his records stay safe even if the program closes.

Build a simple command-line Python program that:

- Lets Kalash **add new expenses** (item name + amount)
- Saves all expenses in a CSV file (expenses.csv)
- When requested, **shows a summary** of:
    - Total money spent
    - Average expense per item
    - List of **expensive items** (₹1000+)

Sample:

=== Kalash's Budget Tracker ===

1. Add Expense

2. Show Summary

3. Exit

Enter choice: 1

Enter item name: College Books

Enter amount (in ₹): 1750

Expense added!

=== Kalash's Budget Tracker ===

1. Add Expense

2. Show Summary

3. Exit

Enter choice: 2

Total Spent: ₹3450

Average Expense: ₹1150.0

Expenses over ₹1000:

- College Books: ₹1750

- Shoes: ₹2200