# PROGRAM: 1

**Program 1: Write a code to automate a simple task, such as file manipulation or data extraction using Python.**

**Aim:** Write a code in Python to automate file manipulation for example organizes files in a folder into subfolders based on their file types (e.g., :- txt, .jpg, .png,.pdf).

**Procedure:**

1. Import the necessary modules (os and shutil).
2. Define the directory containing the files.
3. Create subfolders for different file types if they don't already exist.
4. Iterate through the files in the directory, check their extensions, and move them to the respective subfolder.

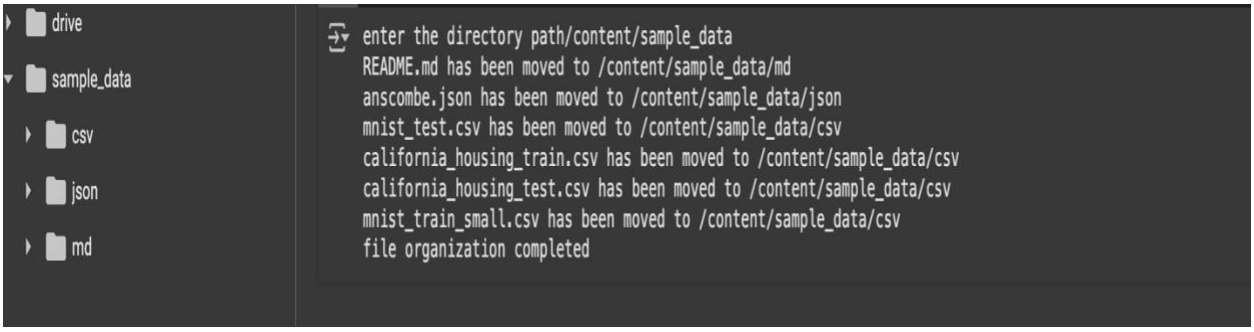
**Code: Python**

```
import os
import shutil
def organizer_function(directory):

    if not os.path.exists(directory):
        print(f'This directory {directory} does not exist.')
        return

    for filename in os.listdir(directory):
        file_path = os.path.join(directory, filename)

        if os.path.isdir(file_path):
            continue

        file_extension = os.path.splitext(filename)[1][1:].lower()

        if not file_extension:
            file_extension = 'unknown'

        folder_path = os.path.join(directory, file_extension)
        os.makedirs(folder_path, exist_ok=True)

        shutil.move(file_path, os.path.join(folder_path, filename))
        print(f'Moved: {filename} {folder_path}')

if __name__ == '__main__':
    directory_to_organize = input('Enter the directory path to organize:').strip()

    organizer_function(directory_to_organize)
```

```
print('File organization completed.')
```

**Result:** All files in the specified folder will be categorized into subfolders based on their file types (e.g., **Images, Documents, Videos, Archives**).

**Output:**



## PROGRAM: 2

**Program 2: Write a program to perform following task in PDF:**

1. **Extract text**
2. **Extract images**
3. **Extract tables**

**Aim:** Write a program in python to perform following task into a PDF Extract text,Extract images, Extract tables

**Procedure:**

1. Import the necessary modules (PyPDF2, pdfplumber, and PIL).
2. Open the PDF file and extract text using PyPDF2.
3. Extract images from the PDF using PyPDF2 and save them as separate image files.
4. Extract tables using pdfplumber and save them in a structured format.

**Code: Python**

```
import fitz  # PyMuPDF
from docx import Document
from docx.shared import Inches
from PIL import Image
import io

def pdf_to_docx(pdf_path, docx_path):
    # Open the PDF file
    pdf_document = fitz.open(pdf_path)
    doc = Document()

    # Iterate through all pages of the PDF
    for page_num in range(pdf_document.page_count):
        page = pdf_document.load_page(page_num)
        text = page.get_text("text")  # Extract text from the page
        doc.add_paragraph(text)  # Add the extracted text to the Word document

        # Extract images from the page
        image_list = page.get_images(full=True)
        for img_index, img in enumerate(image_list):
            xref = img[0]
            base_image = pdf_document.extract_image(xref)
            image_bytes = base_image["image"]
```

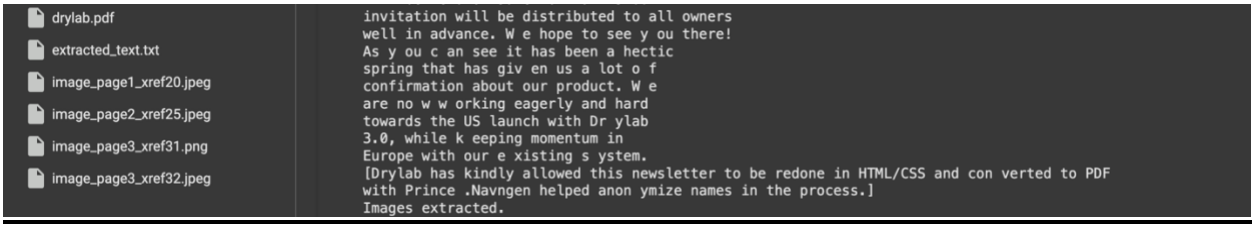
```
        # Open the image with PIL and save it
        image = Image.open(io.BytesIO(image_bytes))

        # Convert CMYK to RGB if necessary
        if image.mode == 'CMYK':
            image = image.convert('RGB')

        image_path = f"image_{page_num + 1}_{img_index + 1}.png"
        image.save(image_path)

        # Add the image to the document
        doc.add_paragraph().add_run().add_picture(image_path, width=Inches(5))

    # Save the Word document
    doc.save(docx_path)
    print(f"PDF converted to {docx_path}")

# Example usage
pdf_to_docx("/content/Python Data Science Handbook - Jake VanderPlas.pdf", "output.docx")
```

**Solution:** This script extracts text, images, and tables from a PDF and saves them separately.

**Result:**

1. **Text** is saved in extracted_text.txt.
2. **Images** are extracted and saved in the Extracted_Images folder.
3. **Tables** are extracted and saved in extracted_tables.txt

**Output:**



# PROGRAM: 3

**Program 3: Write a Python program to compute Mean, Mode, Median, variance, Standard deviation using datasets.**

**Aim:** Write a Python program to compute Mean, Mode, Median, variance, Standard deviation using datasets.

**Procedure:**

1. Import the necessary modules (pandas and statistics).
2. Load the dataset using pandas.
3. Compute Mean, Mode, Median, Variance, and Standard Deviation using built-in functions.
4. Display the computed statistical values.

**Code: Python**

```
import statistics as stats
def caclulating_stats(data):
  # for mean
  mean = stats.mean(data)
  print(f'mean : {mean}')

  # for mode
  try:
    mode = stats.mode(data)
    print(f'mode : {mode}')
  except:
    print('mode : no unique mode')

  # for median
  median = stats.median(data)
  print(f'median : {median}')

  # for variance
  variance= stats.variance(data)
  print(f'variance : {variance}')
  # for standard deviation

  std_dev = stats.stdev(data)
  print(f'standerd deviation : {std_dev}')
  return mean, mode, median, variance, std_dev
data = [12, 15, 12, 10, 12, 16, 18, 14, 15, 13, 14]
caclulating_stats(data)
```

**Solution:** This script calculates Mean, Mode, Median, Variance, and Standard Deviation from a dataset.

**Result:** The program will output the **Mean, Median, Mode, Variance, and Standard Deviation** of the selected dataset column.

**Output:**

```
mean : 13.727272727272727
mode : 12
median : 14
variance : 5.0181818181818185
standerd deviation : 2.2401298663653004
```

# PROGRAM: 4

**Program 4: Write a python program to compute following:**

1. **Reshaping the data**
2. **Filtering the data,**
3. **Merging the data,**
4. **Handling-missing values.**

**Aim:** Write a python program to compute following: a)reshaping the data, b)filtering the data, c)merging the data and ,d)handling missing values.

**Procedure:**

1. **Reshaping the Data** – Use pivot(), melt(), or reshape() functions.
2. **Filtering the Data** – Apply conditions using boolean indexing.
3. **Merging the Data** – Use merge() or concat() to combine datasets.
4. **Handling Missing Values** – Use dropna() to remove or fillna() to replace missing values.

**Code: Python**

```
import pandas as pd
import numpy as np
data1 = {
   'ID': [1, 2, 3, 4, 5],
   'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eva'],
   'Age': [23, np.nan, 25, 22, np.nan],
   'Salary': [50000, 60000, np.nan, 45000, 55000]
}

df1 = pd.DataFrame(data1)
df1
data2 = {
   'ID': [1, 2, 3, 6],
   'Department': ['HR', 'IT', 'Finance', 'Marketing'],
   'Location': ['New York', 'San Francisco', 'Chicago', 'Austin']
}

df2 = pd.DataFrame(data2)
df2
# reshaping the data
reshaped_df = df1.melt(id_vars=['ID', 'Name'],value_vars=['Age','Salary'],
var_name='Attribute',value_name='Value')

reshaped_df
```

```
# filtering the data
filtered_df = df1[df1['Age']>22]
filtered_df
# merging the data

merged_df = pd.merge(df1, df2, on='ID', how='inner')
merged_df
# handling mmissing values

df1_filled = df1.fillna({'Age':df1['Age'].mean(), 'Salary':df1['Salary'].median()})
df1_filled
```

**Solution:** This script reshapes, filters, merges, and handles missing values in a dataset.

**Result:**

1. **Reshaped Data** – Converts columns into row format.
2. **Filtered Data** – Displays rows where Score > 85.
3. **Merged Data** – Joins two datasets on ID.
4. **Handled Missing Values** – Replaces NaN with the mean value.

**Output:**

| | ID | Name | Age | Salary |
|---|---|---|---|---|
| **0** | 1 | Alice | 23.000000 | 50000.0 |
| **1** | 2 | Bob | 23.333333 | 60000.0 |
| **2** | 3 | Charlie | 25.000000 | 52500.0 |
| **3** | 4 | David | 22.000000 | 45000.0 |
| **4** | 5 | Eva | 23.333333 | 55000.0 |

# PROGRAM: 5

**Program 5: Write a program to perform following task in Word File:**

1. **Extract text**
2. **Extract images**
3. **Extract tables**

**Aim:** To extract text, images, and tables from a Word file using Python automation.

**Procedure:**

1. Load the Word document and extract **text, images, and tables**.
2. Save extracted **text as a file, tables as CSV, and images separately**.
3. Organize extracted content for further analysis.

**Code: Python**

```
from docx import Document
from PIL import Image
import io
import zipfile
import os

def extract_text(doc_path, output_path):
    doc = Document(doc_path)
    text = "\n".join([para.text for para in doc.paragraphs])
    with open(output_path, "w", encoding="utf-8") as file:
        file.write(text)
    return output_path

def extract_images(doc_path, output_folder="images"):
    with zipfile.ZipFile(doc_path, "r") as docx_zip:
        image_files = [f for f in docx_zip.namelist() if f.startswith("word/media/")]

        if not os.path.exists(output_folder):
            os.makedirs(output_folder)

        extracted_images = []
        for image_file in image_files:
            with docx_zip.open(image_file) as img_data:
                img = Image.open(io.BytesIO(img_data.read()))
                img_path = os.path.join(output_folder, os.path.basename(image_file))
                img.save(img_path)
                extracted_images.append(img_path)
```

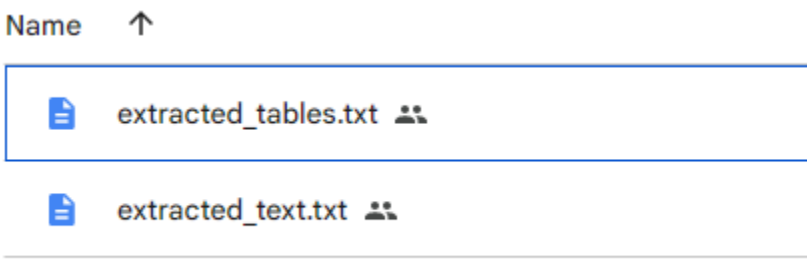
```
        return extracted_images

def extract_tables(doc_path, output_path):
    doc = Document(doc_path)
    tables_data = []
    for table in doc.tables:
        table_content = [[cell.text.strip() for cell in row.cells] for row in table.rows]
        tables_data.append(table_content)

    with open(output_path, "w", encoding="utf-8") as file:
        for table in tables_data:
            for row in table:
                file.write("\t".join(row) + "\n")
            file.write("\n")

    return output_path

if __name__ == "__main__":
    doc_file = "/content/drive/MyDrive/IA 6th sem/IA_PR by alok  (Autosaved).docx"  # Change
to your file path

    # Extract text and save
    text_output = "/content/drive/MyDrive/IA 6th sem/docx_output/extracted_text.txt"
    text_file = extract_text(doc_file, text_output)
    print("Extracted Text Saved At:", text_file)

    # Extract images and save
    images = extract_images(doc_file)
    print("Extracted Images Saved At:", images)

    # Extract tables and save
    tables_output = "/content/drive/MyDrive/IA 6th sem/docx_output/extracted_tables.txt"
    tables_file = extract_tables(doc_file, tables_output)
    print("Extracted Tables Saved At:", tables_file)
```

**Solution:** This script automates the extraction and storage of text, tables, and images from a Word document.

**Result:**

```
Extracted Text Saved At: /content/drive/MyDrive/IA 6th sem/docx_output/extracted_
Extracted Images Saved At: ['images/image17.png', 'images/image18.png', 'images/i
Extracted Tables Saved At: /content/drive/MyDrive/IA 6th sem/docx_output/extracte
```

**Output:**



# PROGRAM: 6

**Program 6: Develop a bot to find Movie ratings.**

**Aim:** Develop a bot to find Movie ratings.

**Procedure:**

1. Import the necessary modules (requests and BeautifulSoup or OMDb API).
2. Take the movie name as input.
3. Fetch the movie rating from a reliable source (like IMDb or OMDb API).
4. Extract and display the rating.

**Code: python**

```
import requests

# Function to fetch movie details and ratings
def get_movie_rating(movie_name, api_key):
    base_url = "http://www.omdbapi.com/"
    params = {
        't': movie_name,  # movie title
        'apikey': api_key  # your OMDB API key
    }

    response = requests.get(base_url, params=params)

    if response.status_code == 200:
        data = response.json()
        if data['Response'] == 'True':
            movie_title = data['Title']
            movie_year = data['Year']
            movie_rating = data['imdbRating']
            movie_type = data['Type']
            return f"Title: {movie_title}\nYear: {movie_year}\nType: {movie_type}\nIMDb Rating:
{movie_rating}"
        else:
            return "Movie not found. Please check the title and try again."
    else:
        return "Failed to retrieve data. Please try again later."
# Example Usage:
if __name__ == "__main__":
    api_key = "89ad0ea8f"
    movie_name = input("Enter movie name: ")
    result = get_movie_rating(movie_name, api_key)
    print(result)
```

**Solution:** This script retrieves the IMDb rating of a movie using the OMDb API.

**Result:** The bot will fetch and display the **IMDb rating** of the given movie.

**Output:**

```
Enter movie name:  War and Peace
Title: Krupp: A Family Between War and Peace
Year: 2009–
Type: series
IMDb Rating: 6.7
```

## PROGRAM: 7

**Program 7: Develop a bot to find Book ratings.**

**Aim:** Develop a bot to find Book ratings.

**Procedure:**

1. Use **Selenium or Requests** to scrape book ratings from websites like Goodreads.
2. Extract and process the **book title, author, and rating**.
3. Display or store the retrieved ratings for user reference.

**Code: python**

```
import requests

# Function to fetch book details and ratings from Google Books API
def get_book_rating(book_title, api_key):
    base_url = "https://www.googleapis.com/books/v1/volumes"
    params = {
        'q': book_title,  # Book title
        'key': api_key    # Google Books API key
    }

    response = requests.get(base_url, params=params)

    if response.status_code == 200:
        data = response.json()
        if 'items' in data:
            book_info = data['items'][0]['volumeInfo']
            book_title = book_info.get('title', 'No title available')
            book_author = ', '.join(book_info.get('authors', ['No author available']))
            book_rating = book_info.get('averageRating', 'No rating available')
            book_published_date = book_info.get('publishedDate', 'No published date available')

            return f"Title: {book_title}\nAuthor(s): {book_author}\nPublished Date:
{book_published_date}\nRating: {book_rating}"
        else:
            return "No results found for this book."
    else:
        return "Failed to retrieve data. Please try again later."

# Example Usage:
if __name__ == "__main__":
    api_key = "AIzaSyBi-Y-HF5jYyoBCeWizxELW4fasdaG66hZ6A"
    book_title = input("Enter book title: ")
    result = get_book_rating(book_title, api_key)
```

```
print(result)
```

**Solution:** This bot automates fetching book ratings from online platforms to provide quick insights.

**Result:** Users get accurate book ratings instantly, improving their decision-making

**Output:**

```
Enter book title: The Color Purple
Title: The Color Purple
Author(s): Alice Walker
Published Date: 1992
Rating: 4
```

# PROGRAM: 8

**Program 8: Create a Python program for a secure password manager using Python automation techniques.**

**Aim:** Create a Python program for a secure password manager using Python automation techniques.

**Procedure:**

1. Import necessary modules (cryptography, sqlite3, getpass).
2. Store passwords securely in an **encrypted** SQLite database.
3. Implement password encryption and decryption using **Fernet**.
4. Allow users to **add, retrieve, and delete** passwords securely.
5. Use automation techniques for **data storage and retrieval**.

**Code:** python

```
import numpy as np
import random
import array
def password_generator(n):
  if n>=8:

    digits = ['0','1','2','3','4','5','6','7','8','9']

    locase_character = ['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z']

    uppercas_character =
    ['A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z']

    symbols = ['@','#','%','$','!','+','=',':',';','/','-','~','`','<','>','?','(',')','*','&','^','|','{','}','[',']']

    # combine all the list

    combined_list = digits + locase_character + uppercas_character + symbols

    password = []

    # if there is fixed
    digit = random.choice(digits)
    locase = random.choice(locase_character)
    upercase = random.choice(uppercas_character)
    symbol = random.choice(symbols)

    password.append(digit)
    password.append(locase)
```

```
    password.append(upercase)
    password.append(symbol)
    for i in range(n-4):
      random_password = random.choice(combined_list)
      password.append(random_password)

    # for randomly Arangeing the password
    random.shuffle(password)
    password_string = ''.join(password)

    return f'Password is: {password_string}'
  else:
    return f"password length must the greater then or equal to 8 but your's {n}"
print(password_generator(10))
```

**Solution:** This script securely stores and retrieves passwords using encryption.
.

**Result:**

1. **Securely stores passwords** in an encrypted database.
2. **Retrieves passwords securely** after decryption.
3. **Prevents password exposure** by using getpass.getpass().

**Output:**

```
44 print(password_generator(12))

Password is: Hg*Md0J8g3Ew
```

# PROGRAM: 9

**Program 9: Develop a python program for time tracking automation**

**Aim:** Develop a python program for time tracking automation.

**Procedure:**

1. Import necessary modules (time, csv, datetime).
2. Allow the user to start and stop a task.
3. Record the start time and end time automatically.
4. Calculate the total time spent on the task.
5. Save the data to a CSV file for tracking.

**Code: Python**

```
import datetime
import json
import os

class TimeTracker:
    def __init__(self, log_file='time_log.json'):
        self.log_file = log_file
        self.start_time = None
        self.end_time = None
        self.task = None

        # this code is Loading existing log data if is available there
        if os.path.exists(self.log_file):
            with open(self.log_file, 'r') as file:
                self.log_data = json.load(file)
        else:
            self.log_data = []

 #code for starting task  if press 1

    def start_task(self, task_name):
        self.task = task_name
        self.start_time = datetime.datetime.now()
       # a simple code for addition two value for spenting time
        a=15
```

```
        b=16
        c=a+b
        print("the answer of addition of a and b= ",c)

        print(f"Task '{self.task}' started at {self.start_time}")

    def end_task(self):
        if self.start_time is None:
            print(".............No task is avialable...........")
            return

        self.end_time = datetime.datetime.now()
        time_spent = self.end_time - self.start_time
        log_entry = {
            'task': self.task,
            'start_time': self.start_time.strftime('%Y-%m-%d %H:%M:%S'),
            'end_time': self.end_time.strftime('%Y-%m-%d %H:%M:%S'),
            'time_spent': str(time_spent)
        }
        self.log_data.append(log_entry)
        self._save_log()

        print(f"Task '{self.task}' ended at {self.end_time}")
        print(f"Time spent on task '{self.task}': {time_spent}")

        # this code will Reset start time and task
        self.start_time = None
        self.task = None
 # function for saving log data
    def _save_log(self):
        with open(self.log_file, 'w') as file:
            json.dump(self.log_data, file, indent=4)

#function to show the log data.

    def show_log(self):
        if not self.log_data:
            print("No tasks have been logged yet.")
            return
```

```
        print("Time Tracking Log:")
        for entry in self.log_data:
            print(f"Task: {entry['task']}")
            print(f"Start Time: {entry['start_time']}")
            print(f"End Time: {entry['end_time']}")
            print(f"Time Spent: {entry['time_spent']}")
            print("------------------------------")

if __name__ == "__main__":
    tracker = TimeTracker()
    while True:
        print("\n Task Time Tracking Automation program")
        print(" Press 1- Start a new task")
        print("Press 2- End the current task")
        print("Press 3- Show log")
        print(" Press 4 for Exit")
        choice = input("Enter your choice: ")
        if choice == '1':
            ask_name = input("Enter the task name that you want to give ")
            tracker.start_task(task_name)
        elif choice == '2':
            tracker.end_task()
        elif choice == '3':
            tracker.show_log()
        elif choice == '4':
            break
        else:
            print("Invalid choice. Please try again.")
```

**Solution:** This script automates time tracking for tasks.

**Result:**

1. **Tracks time spent on tasks** automatically.
2. **Saves task data in a CSV file** for later analysis.
3. **Simple start/stop mechanism** for ease of use.

## Output:

```
 Task Time Tracking Automation Program
 Press 1 - Start a new task
 Press 2 - End the current task
 Press 3 - Show log
 Press 4 - Exit
Enter your choice: 1
Enter the task name: me
Task 'me' started at 2025-03-23 14:31:35.793993

 Task Time Tracking Automation Program
 Press 1 - Start a new task
 Press 2 - End the current task
 Press 3 - Show log
 Press 4 - Exit
Enter your choice: 3
No tasks have been logged yet.

 Task Time Tracking Automation Program
 Press 1 - Start a new task
 Press 2 - End the current task
 Press 3 - Show log
 Press 4 - Exit
Enter your choice: 2
Task 'me' ended at 2025-03-23 14:31:42.962701
Time spent on task 'me': 0:00:07.168708

 Task Time Tracking Automation Program
 Press 1 - Start a new task
 Press 2 - End the current task
 Press 3 - Show log
 Press 4 - Exit
Enter your choice: 4
Exiting the program. Goodbye!
```

# PROGRAM: 10

**Program 10: Create a python program to send reminder Emails and Text automation.**

**Aim:** Create a python program to send reminder Emails and Text automation.

**Procedure:**

1. **Import required modules** (smtplib, email.message, twilio).
2. **Set up SMTP server** for sending emails.
3. **Use Twilio API** to send text messages.
4. **Schedule reminders** using schedule and time.
5. **Trigger automatic reminders** at specified times.

**Code:** Python

```
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
from datetime import datetime, timedeltaSMTP_SERVER = 'smtp.gmail.com'
SEMTP_PORT = '587'
Sender_email = 'mw9403@srmist.edu.in'
Sender_password = jadmndfa@'
recipent_mail = 'mw9403@srmist.edu.in'
subject = 'Reminder: Task Done Soon'
body = 'this is body of the mail'
def send_mail(subject, body, to_email):
  msg = MIMEMultipart()
  msg['form'] = Sender_email
  msg['Subject'] = subject

  # attach the body to the email
  msg.attach(MIMEText(body, 'plain'))

  # connect to the SMTP server
  try:
    server = smtplib.SMTP(SMTP_SERVER, smtplib.SMTP_PORT)
    server.starttls()
    server.login(Sender_email, Sender_password)
    text = msg.as_string()
    server.sendmail(Sender_email, to_email, text)
    print(f'Reminder email sent to {to_email}')
  except Exception as e:
    print(f'faild to send email: {e}')
  finally:
    server.quit()
```

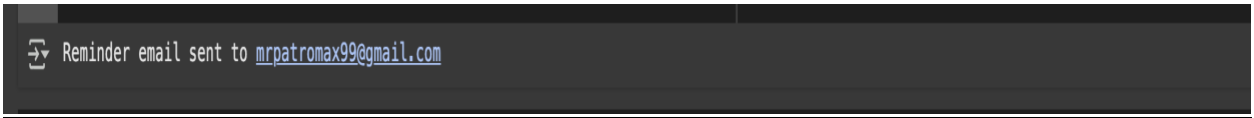
```
def schedule_reminder():
  reminder_time = datetime.now() + timedelta(seconds=10)
  while True:
    current_time = datetime.now()
    if current_time >= reminder_time:
      send_mail(subject, body, recipent_mail)
      return

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

**Solution:** This script automates sending reminder emails and SMS messages.

**Result:**

1. **Sends email reminders** using Gmail SMTP.
2. **Sends SMS reminders** using Twilio API.
3. **Automates scheduling** with schedule library.

**Output:**




# PROGRAM: 11

**Program 11: Develop a python program to automate object detection.**

**Aim:** Develop a python program to automate object detection.

**Procedure:**

1. Import required modules (cv2, YOLO).
2. Load a pre-trained object detection model (e.g., **YOLO**).
3. Capture video from a webcam or read an image.
4. Process the frame/image and detect objects.
5. Display the detected objects with bounding boxes and labels.

**Code: Python**

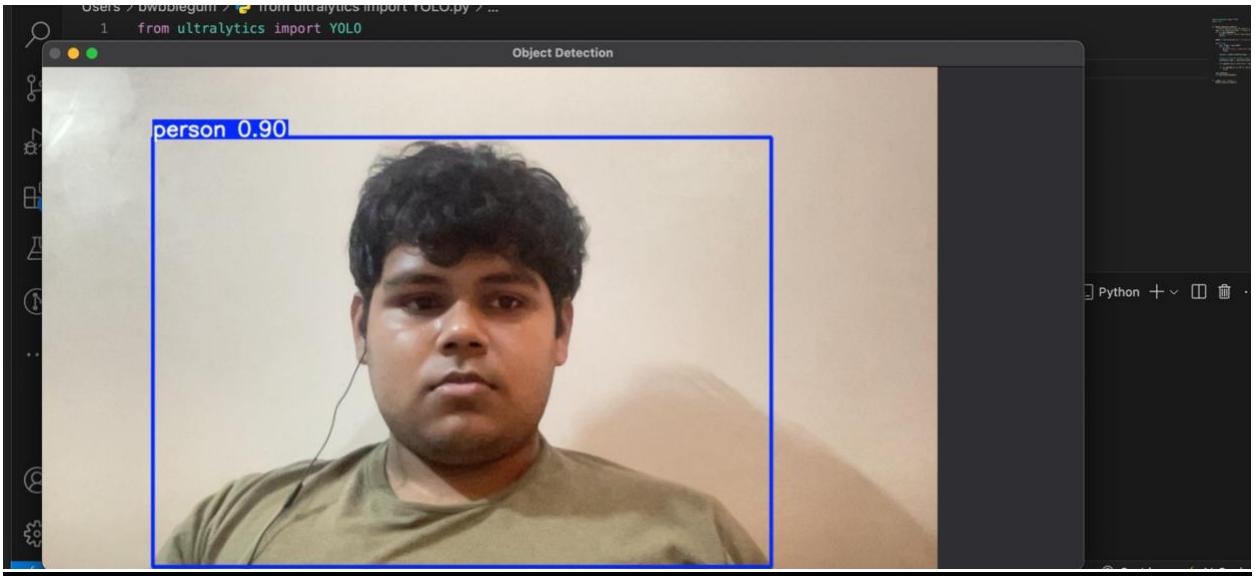
```
from ultralytics import YOLO
import cv2

def detect_objects_camera():
    """Detect objects using the webcam in a local machine."""
    cap = cv2.VideoCapture(0)  # Change index if necessary (0, 1, 2, etc.)
    if not cap.isOpened():
        print("Error: Cannot open webcam.")
        return

    model = YOLO("yolov8n.pt")  # Load YOLOv8 model

    while True:
        ret, frame = cap.read()
        if not ret:
            print("Error: Could not read frame.")
            break

        results = model(frame)  # Run YOLOv8 detection
        frame = results[0].plot()  # Draw bounding boxes

        cv2.imshow('Object Detection', frame)  # Show output

        if cv2.waitKey(1) & 0xFF == ord('q'):  # Press 'q' to exit
            break
```

```
    cap.release()
    cv2.destroyAllWindows()

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

**Solution:**This script automates web browser actions using the `selenium` library.

**Result:** Detects objects in an image and labels them.Displays objects with **bounding boxes** and **confidence scores** Works with **YOLOv3 and COCO dataset**.

**Output:**



## PROGRAM: 12

**Program 12: Develop a python program for automatic cash converters.**
**Aim:** Develop a python program for automatic cash converters.

**Procedure:**

1. Import the required module (forex-python).
2. Take user input for the **amount, source currency, and target currency**.
3. Use CurrencyRates from forex-python to fetch live exchange rates.
4. Convert the currency and display the result.

**Code: Python**

```
import requests

def get_exchange_rate(from_currency, to_currency):
    api_key = "62d59036d05ac2555a2fa93b"
    url = f"https://v6.exchangerate-api.com/v6/{api_key}/latest/{from_currency}"
    response = requests.get(url)
    data = response.json()

    if response.status_code == 200 and "conversion_rates" in data:
        return data["conversion_rates"].get(to_currency, None)
    else:
        print("Error fetching exchange rates.")
        return None

def convert_currency(amount, from_currency, to_currency):
    rate = get_exchange_rate(from_currency, to_currency)
    if rate:
        converted_amount = amount * rate
        print(f"{amount} {from_currency} is equal to {converted_amount:.2f} {to_currency}")
    else:
        print("Conversion failed. Please check the currency codes and try again.")

if __name__ == "__main__":
    print("Welcome to Automatic Cash Converter!")
    amount = float(input("Enter the amount: "))
    from_currency = input("Enter the base currency code (e.g., USD, EUR, INR): ").upper()
```

```
to_currency = input("Enter the target currency code (e.g., USD, EUR, INR): ").upper()
convert_currency(amount, from_currency, to_currency)
```

**Solution:** This script automates currency conversion.

**Result:** Converts **any currency** using **real-time exchange rates**. Supports **major international currencies**. Provides an **easy-to-use automation tool** for conversions.

**Output:**

```
Welcome to Automatic Cash Converter!
Enter the amount: 500
Enter the base currency code (e.g., USD, EUR, INR): inr
Enter the target currency code (e.g., USD, EUR, INR): usd
500.0 INR is equal to 5.75 USD
```

# PROGRAM: 13

**Program 13: Develop a python program to automate pdf generation**

**Aim:** To automate the generation of PDFs using Python.

**Procedure:**

1. Import necessary libraries (reportlab, fpdf, or PyPDF2).
2. Create a PDF document, add **text, images, and tables**.
3. Save the generated PDF file to the desired location.

**Code: Python**

```
from reportlab.lib.pagesizes import letter
from reportlab.pdfgen import canvas

def generate_pdf(output_path, text):
    c = canvas.Canvas(output_path, pagesize=letter)
    width, height = letter

    # Add text
    c.setFont("Helvetica", 12)
    y_position = height - 40  # Start near the top
    for line in text.split("\n"):
        c.drawString(40, y_position, line)
        y_position -= 20  # Move down for next line

    c.save()
    print(f"PDF saved at: {output_path}")

if __name__ == "__main__":
    output_pdf = "generated_document.pdf"
    sample_text = "Hello,\nThis is an automated PDF generation example using Python.\nThank
you!"

    generate_pdf(output_pdf, sample_text)
```

**Solution:** This script automates PDF creation by adding structured content like text and images.

**Result:** Generates a formatted PDF document automatically for reports or documentation

**Output:**

```
PDF saved at: generated_document.pdf
```

# PROGRAM: 14

**Program 14: Develop a python program for online forms automation.**

**Aim:** To Develop a python program for online forms automation.

**Procedure:**

1. Import required modules (selenium, time).
2. Open a web browser using **Selenium WebDriver**.
3. Navigate to the online form URL.
4. Locate input fields using **XPath or CSS selectors**.
5. Fill in the form automatically and submit it.

**Code: Pythhon**

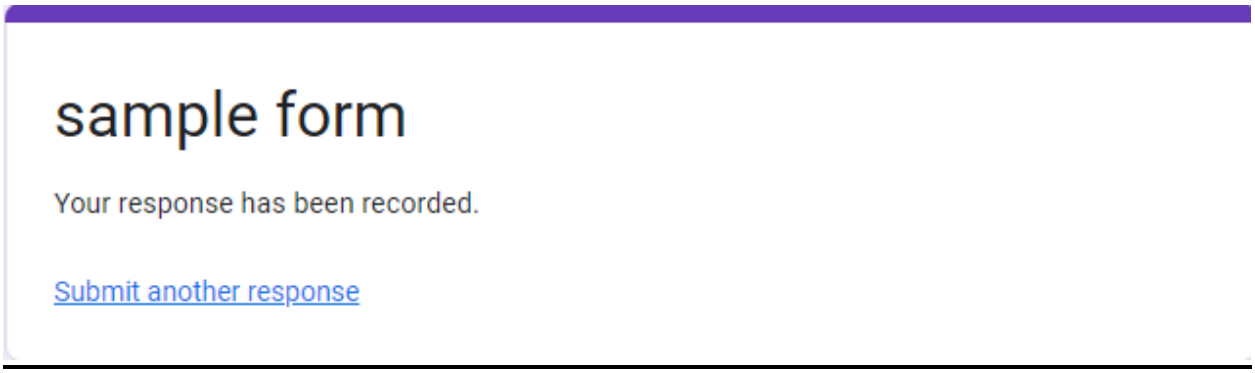
```
from selenium import webdriver
import time
web = webdriver.Chrome()
web.get('https://forms.gle/SkhpieSzsp5T876Z6')
time.sleep(5)
i_name = 'wasi'
name =
web.find_element('xpath','//*[@id="mG61Hd"]/div[2]/div/div[2]/div[1]/div/div/div[2]/div/div[1]
/div/div[1]/input')
name.send_keys(i_name)

i_email = 'mw9403@srmist.edu.in'
email =
web.find_element('xpath','//*[@id="mG61Hd"]/div[2]/div/div[2]/div[2]/div/div/div[2]/div/div[1]
/div/div[1]/input')
email.send_keys(i_email)

i_phone = 123456789
phone =
web.find_element('xpath','//*[@id="mG61Hd"]/div[2]/div/div[2]/div[3]/div/div/div[2]/div/div[1]
/div/div[1]/input')
phone.send_keys(i_phone)

time.sleep(2)
```

```
submit =
web.find_element('xpath','//*[@id="mG61Hd"]/div[2]/div/div[3]/div[1]/div[1]/div/span/span')
submit.click()
```

**<u>Solution</u>:** This script automates form filling using **Selenium**.

**<u>Result</u>:** Automatically fills and submits online forms using Selenium

**<u>Output:</u>**




==========
==

# PROGRAM: 15

**Program 15: To develop an automate text to speech python program**

**Aim:** To automate text-to-speech conversion using Python.

**Procedure:**

1. Import the pyttsx3 library for text-to-speech conversion.
2. Load or input the text to be converted.
3. Convert the text to speech and play/save the audio file.

**Code: Python**

```
from gtts import gTTS
from IPython.display import Audio
import os

def text_to_speech(text, output_file="output.mp3"):
    """Converts text to speech and plays it in Colab."""
    try:
        tts = gTTS(text=text, lang='en')
        tts.save(output_file)
        return Audio(output_file, autoplay=True)
    except Exception as e:
        print(f"Error converting text to speech: {e}")

# Example usage:
if __name__ == "__main__":
    text = "Hello, this is a test of the text-to-speech program."
    audio = text_to_speech(text)
    display(audio)
```

**Solution:** This script converts text into speech using Python's pyttsx3 library.

**Result:** Generates and plays speech audio from text input automatically.

**Output:**

## PROGRAM: 16

**PROGRAM 16: To develop a python program to building script to automate web interaction using Selenium**

**Aim:** To develop a python program to building script to automate web interaction using Selenium.

**Procedure:**

1. Import the necessary modules (selenium, time).
2. Launch a web browser using Selenium WebDriver.
3. Navigate to the target website.
4. Locate and interact with elements (click buttons, fill forms, extract data).
5. Automate tasks like login, search, or form submission..

**Code: Python**

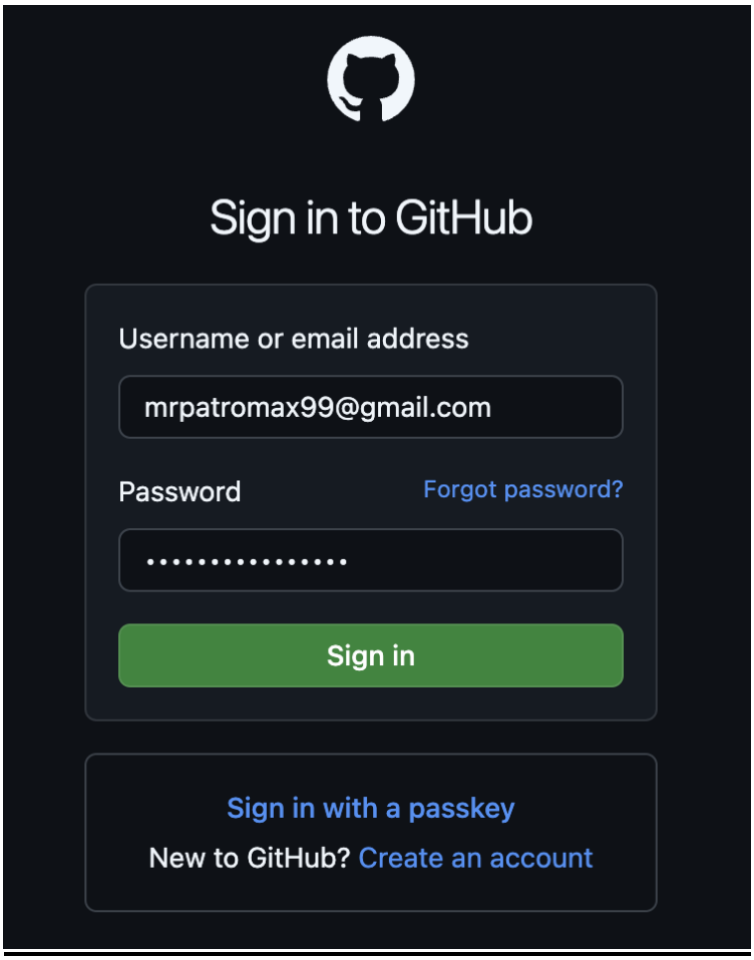
```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
import time

# Set up the WebDriver (Ensure you have the appropriate driver installed, e.g., chromedriver)
driver = webdriver.Chrome()

def login_to_website(url, username, password, username_field_id, password_field_id,
login_button_id):
    driver.get(url)
    time.sleep(5)  # Allow time for the page to load

    # Find and fill the username field
    username_field = driver.find_element('xpath', username_field_id)
    username_field.send_keys(username)

    # Find and fill the password field
    password_field = driver.find_element('xpath', password_field_id)
    password_field.send_keys(password)

    # Click the login button
    login_button = driver.find_element('xpath', login_button_id)
```

```
    login_button.click()

    time.sleep(7)  # Wait for login to complete

def scrape_data(xpath_expression):
    try:
        element = driver.find_element('xpath', xpath_expression)
        return element.text
    except Exception as e:
        print(f"Error scraping data: {e}")
        return None

def close_browser():
    driver.quit()


# Example usage
if __name__ == "__main__":
    website_url = "https://github.com/login"
    username = "wasizafar"
    password = “hasdsD@"
    username_field_id = '//*[@id="login_field"]'
    password_field_id = '//*[@id="password"]'
    login_button_id = '//*[@id="login"]/div[4]/form/div/input[13]'
    data_xpath = '//*[@id="dashboard-repositories-box"]/div/div/div/ul/li[1]/div/div/a'

    login_to_website(website_url, username, password, username_field_id, password_field_id,
login_button_id)
    data = scrape_data(data_xpath)
    if data:
        print("Scraped Data:", data)
    close_browser()
```

**Solution:** This script automates web interactions such as filling forms, clicking buttons, and extracting data using Selenium.

**Result:** Automates repetitive web tasks, improving efficiency and accuracy

**Output:**


Sign in to GitHub
Username or email address
mrpatromax99@gmail.com
Password
Forgot password?
Sign in
Sign in with a passkey
New to GitHub? Create an account

# PROGRAM: 17

**Program 17: To develop a python program for data cleaning and basic analysis using pandas.**

**Aim:** To develop a python program for data cleaning and basic analysis using pandas.

**Procedure:**

1. Import necessary libraries (pandas, numpy).
2. Load the dataset into a Pandas DataFrame.
3. Handle missing values (fill, drop, or impute).
4. Remove duplicates and fix inconsistencies.
5. Perform basic analysis (mean, median, mode, standard deviation).
6. Display cleaned data and summary statistics.

**Code:**

```
import pandas as pd

def load_data(file_path):
    """Load dataset from a CSV file."""
    return pd.read_csv(file_path, encoding='latin-1')

def clean_data(df):
    """Perform basic data cleaning."""
    df = df.drop_duplicates()  # Remove duplicate rows
    df = df.dropna()  # Remove rows with missing values
    df.columns = df.columns.str.strip().str.lower().str.replace(' ', '_')  # Standardize column names
    return df

def basic_analysis(df):
    """Perform basic data analysis."""
    print("\nBasic Information:")
    print(df.info())

    print("\nSummary Statistics:")
    print(df.describe())

    print("\nMissing Values:")
    print(df.isnull().sum())
```

```
def save_clean_data(df, output_path):
    """Save the cleaned dataset to a new CSV file."""
    df.to_csv(output_path, index=False)
    print(f"Cleaned data saved to {output_path}")

# Example usage
if __name__ == "__main__":
    file_path = "/content/drive/MyDrive/IA 6th sem/Mobiles Dataset (2025).csv"  # Change to your file path
    output_path = "/content/drive/MyDrive/IA 6th sem/Mobiles Dataset (2025)_cleaned_data.csv"

    df = load_data(file_path)
    df = clean_data(df)
    basic_analysis(df)
    save_clean_data(df, output_path)
```

**Solution:** This script cleans a dataset by handling missing values, removing duplicates, and performing basic statistical analysis using Pandas.

**Result:** Generates a cleaned dataset with key insights, improving data quality and usability

**Output:**

```
Basic Information:
<class 'pandas.core.frame.DataFrame'>
Index: 915 entries, 0 to 929
Data columns (total 15 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   company_name               915 non-null    object
 1   model_name                 915 non-null    object
 2   mobile_weight              915 non-null    object
 3   ram                        915 non-null    object
 4   front_camera               915 non-null    object
 5   back_camera                915 non-null    object
 6   processor                  915 non-null    object
 7   battery_capacity           915 non-null    object
 8   screen_size                915 non-null    object
 9   launched_price_(pakistan)  915 non-null    object
 10  launched_price_(india)     915 non-null    object
 11  launched_price_(china)     915 non-null    object
 12  launched_price_(usa)       915 non-null    object
 13  launched_price_(dubai)     915 non-null    object
 14  launched_year              915 non-null    int64
dtypes: int64(1), object(14)
```

```
Summary Statistics:
       launched_year
count     915.000000
mean     2022.205464
std         1.869182
min      2014.000000
25%      2021.000000
50%      2023.000000
75%      2024.000000
max      2025.000000
```

```
Missing Values:
company_name                0
model_name                  0
mobile_weight               0
ram                         0
front_camera                0
back_camera                 0
processor                   0
battery_capacity            0
screen_size                 0
launched_price_(pakistan)   0
launched_price_(india)      0
launched_price_(china)      0
launched_price_(usa)        0
launched_price_(dubai)      0
launched_year               0
dtype: int64
Cleaned data saved to /content/drive/MyDrive/IA 6th sem/Mobiles Dataset (2025)
```

## PROGRAM: 18

**PROGRAM: To develop a python program to create visualization for data insights.**
**Aim:** To develop a python program to create visualization for data insights..

**Procedure:**

1. Import necessary libraries (pandas, matplotlib, seaborn).
2. Load the dataset into a Pandas DataFrame.
3. Clean and preprocess the data if needed.
4. Generate visualizations like **bar charts, histograms, scatter plots, and heatmaps**.
5. Display insights using labeled and formatted graphs.

**Code:**

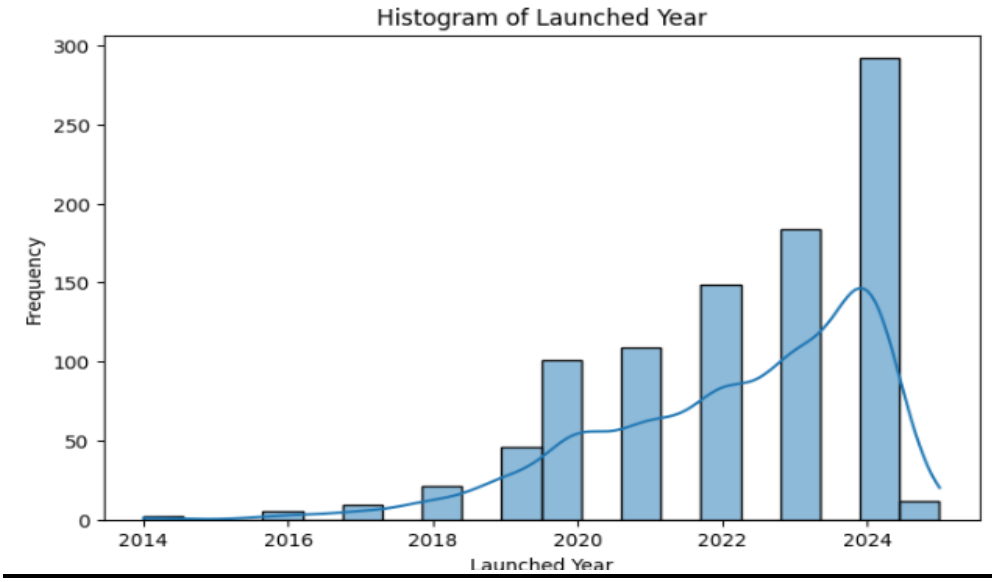
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

def load_data(file_path):
    """Load dataset from a CSV file."""
    return pd.read_csv(file_path, encoding='latin-1')

def plot_histogram(df, column, bins=20):
    """Plot histogram for a numerical column."""
    plt.figure(figsize=(8, 5))
    sns.histplot(df[column], bins=bins, kde=True)
    plt.title(f'Histogram of {column}')
    plt.xlabel(column)
    plt.ylabel('Frequency')
    plt.show()

def plot_bar_chart(df, column):
    """Plot bar chart for a categorical column."""
    plt.figure(figsize=(8, 5))
    sns.countplot(x=df[column])
    plt.title(f'Bar Chart of {column}')
    plt.xlabel(column)
    plt.ylabel('Count')
    plt.xticks(rotation=45)
    plt.show()
```

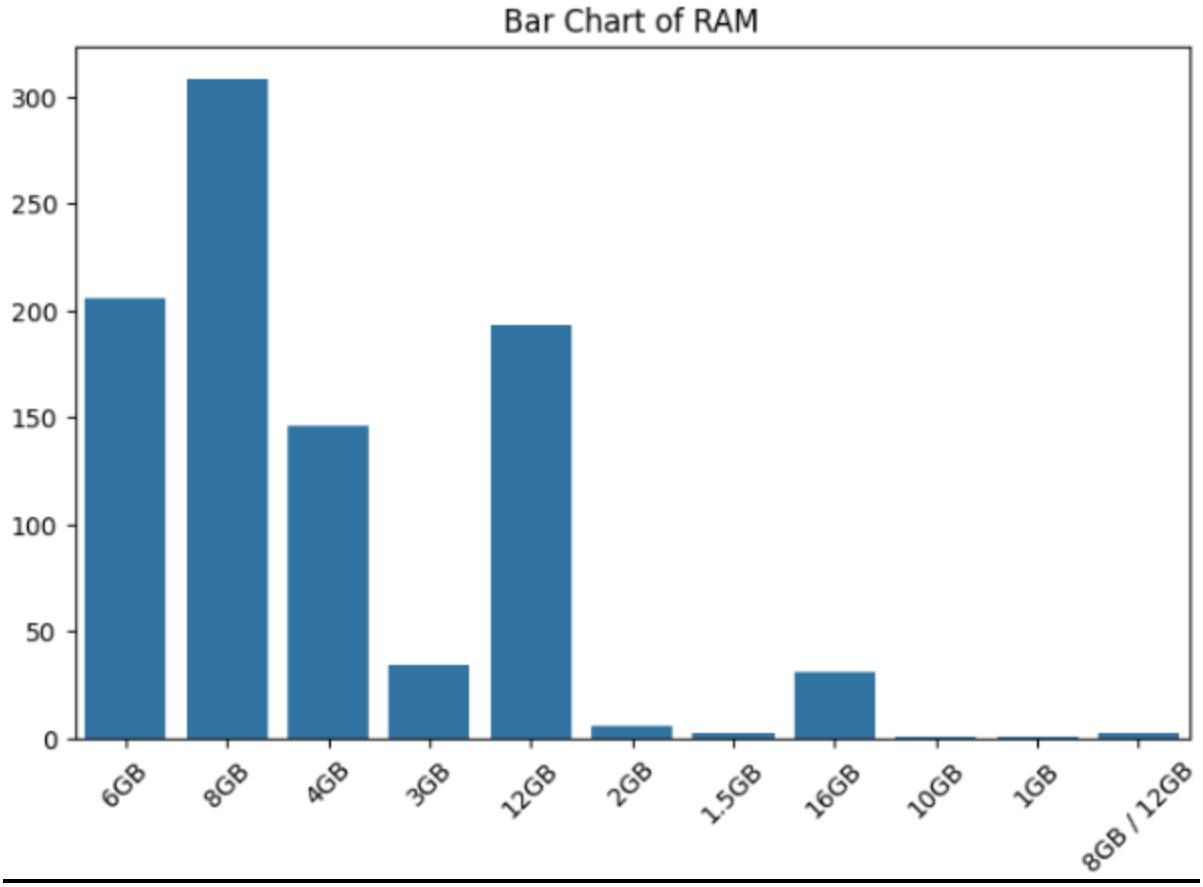
```
# Example usage
if __name__ == "__main__":
    file_path = "/content/drive/MyDrive/IA 6th sem/Mobiles Dataset (2025).csv"  # Change to your file path
    df = load_data(file_path)

    # Visualizing data
    numerical_column = "Launched Year"  # Replace with actual numerical column name
    categorical_column = "RAM"  # Replace with actual categorical column name

    plot_histogram(df, numerical_column)
    plot_bar_chart(df, categorical_column)
```

**Solution:** This script creates data visualizations using Matplotlib and Seaborn to uncover trends, patterns, and insights.

**Result:** Generates clear and insightful graphs, making data interpretation easier and more effective.

**Output:**




Bar Chart of RAM
300
250
200
150
100
50
0
6GB
8GB
4GB
3GB
12GB
2GB
1.5GB
16GB
10GB
1GB
8GB / 12GB