In [ ]:
```
Question1. Write a program for Password Management System
 File creation: Ask user to enter N user names and their passwords. Store use
and passwords into a file named “loginfile.txt”. Store each user and password
line.
 File Processing: Write a program that opens your “security.txt” file and rea
usernames and passwords from it. Store user names in one list and passwords in
another lists.
 Querying: ask user to enter user name and password for verification. If they
the values stored in the lists, print a message “Login Successful”. Otherwise
message “Login Failed, try again”.
```

In [5]:
```
def create_login_file():
    n = int(input("Enter the number of users: "))

    with open("loginfile.txt", "w") as file:
        for i in range(n):
            username = input(f"Enter username {i+1}: ")
            password = input(f"Enter password {i+1}: ")
            file.write(f"{username},{password}\n")

# Call the function to create the file
create_login_file()
```

```
Enter the number of users: 2
Enter username 1: joe
Enter password 1: kkk
Enter username 2: kiran
Enter password 2: fff
```

In [8]:
```
def read_login_file():
    usernames = []
    passwords = []

    with open("loginfile.txt", "r") as file:
        for line in file:
            username, password = line.strip().split(',')
            usernames.append(username)
            passwords.append(password)

    return usernames, passwords

# Call the function to read the file and get the lists
usernames_list, passwords_list = read_login_file()
```

In [ ]:
```

```

In [7]:
```
def login():
    entered_username = input("Enter your username: ")
    entered_password = input("Enter your password: ")

    if entered_username in usernames_list and entered_password == passwords_li
        print("Login Successful")
    else:
        print("Login Failed, try again")

# Call the login function to perform the login process
login()
```

```
Enter your username: joe
Enter your password: kkk
Login Successful
```

In [ ]:
```
Question2. Write a program for Student Performance Analysis
 Create a text file, 'marks.txt', with N marks as floating point numbers. Ope
read marks from it and compute and print the highest mark.
 If the user runs the program more than once you should not overwrite the pre
text file - simply append the marks to the end of the file.
 Modify the above program so that it also prints Top-3 highest marks (Note: y
need to use list concept)
 Modify the above program so that it also prints the Lowest-3 marks.
```

In [1]:
```
def write_marks_to_file():
    n = int(input("Enter the number of marks to add: "))

    with open("marks.txt", "a") as file:
        for i in range(n):
            mark = float(input(f"Enter mark {i+1}: "))
            file.write(f"{mark}\n")

# Call the function to add marks to the file
write_marks_to_file()
```

```
Enter the number of marks to add: 3
Enter mark 1: 50
Enter mark 2: 60
Enter mark 3: 70
```

In [2]:
```python
def compute_highest_mark():
    highest_mark = float("-inf")

    with open("marks.txt", "r") as file:
        for line in file:
            mark = float(line.strip())
            if mark > highest_mark:
                highest_mark = mark

    return highest_mark

# Call the function to compute and print the highest mark
highest_mark = compute_highest_mark()
print(f"The highest mark is: {highest_mark}")
```

```
The highest mark is: 70.0
```

In [3]:
```python
def compute_top_three_marks():
    top_three_marks = []

    with open("marks.txt", "r") as file:
        for line in file:
            mark = float(line.strip())
            top_three_marks.append(mark)

    top_three_marks.sort(reverse=True)
    return top_three_marks[:3]

# Call the function to compute and print the Top-3 highest marks
top_three_marks = compute_top_three_marks()
print("Top-3 highest marks are:")
for i, mark in enumerate(top_three_marks, start=1):
    print(f"{i}. {mark}")
```

```
Top-3 highest marks are:
1. 70.0
2. 60.0
3. 50.0
```

In [4]:
```
def compute_lowest_three_marks():
    lowest_three_marks = []

    with open("marks.txt", "r") as file:
        for line in file:
            mark = float(line.strip())
            lowest_three_marks.append(mark)

    lowest_three_marks.sort()
    return lowest_three_marks[:3]

# Call the function to compute and print the Lowest-3 marks
lowest_three_marks = compute_lowest_three_marks()
print("Lowest-3 marks are:")
for i, mark in enumerate(lowest_three_marks, start=1):
    print(f"{i}. {mark}")
```

```
Lowest-3 marks are:
1. 50.0
2. 60.0
3. 70.0
```

In [ ]:
```
Question3. Write a program for Stock Price Analysis
 File Creation: Continually prompt a user for stock name, followed by price v
5 days. Each row indicates stock name and daily prices of one stock. Store the
values in a text file called “stock-prices.txt”. Open the file in Append Mode.
message “Do you want to continue? “ and stop reading values accordingly. Then,
can close your file.
 File Processing: Now, open your file for processing. Print stock name, minim
price, maximum price and average price values.
 You can also print which day stock price was lowest in the week and which da
price was highest. So, modify your print statement to print stock name, minimu
price & day of minimum price, maximum price & day of maximum price and average
price values. (Hint: Use enumerate to get index values)
```

In [9]:
```
def create_stock_prices_file():
    with open("stock-prices.txt", "a") as file:
        while True:
            stock_name = input("Enter the stock name: ")
            prices = []
            for day in range(1, 6):
                price = float(input(f"Enter the price for day {day}: "))
                prices.append(price)
            file.write(f"{stock_name},{','.join(map(str, prices))}\n")

            continue_input = input("Do you want to continue? (yes/no): ")
            if continue_input.lower() != "yes":
                break

# Call the function to create the stock prices file
create_stock_prices_file()
```

```
Enter the stock name: gold
Enter the price for day 1: 50000
Enter the price for day 2: 60000
Enter the price for day 3: 70000
Enter the price for day 4: 60000
Enter the price for day 5: 750000
Do you want to continue? (yes/no): no
```

In [10]:
```
def process_stock_prices_file():
    with open("stock-prices.txt", "r") as file:
        for line in file:
            data = line.strip().split(',')
            stock_name = data[0]
            prices = list(map(float, data[1:]))

            min_price = min(prices)
            max_price = max(prices)
            avg_price = sum(prices) / len(prices)

            min_day = prices.index(min_price) + 1
            max_day = prices.index(max_price) + 1

            print(f"Stock Name: {stock_name}")
            print(f"Minimum Price: {min_price}, Day: {min_day}")
            print(f"Maximum Price: {max_price}, Day: {max_day}")
            print(f"Average Price: {avg_price:.2f}\n")

# Call the function to process the stock prices file
process_stock_prices_file()
```

```
Stock Name: gold
Minimum Price: 50000.0, Day: 1
Maximum Price: 750000.0, Day: 5
Average Price: 198000.00
```

In [ ]:
```
Question4. Write a program for File Explorer
 Display the contents of file
 Count the number of lines in a text file. (Use splitlines())
 Count the number of unique words in a file.
 Find frequency of words in a given file. (Hint: Use Counter object)
 Show a random line in a file. (Use Random object)
```

Type *Markdown* and LaTeX: α^2

In [1]:
```
import random
from collections import Counter

def display_file_contents(filename):
    try:
        with open(filename, "r") as file:
            contents = file.read()
            print("File Contents:")
            print(contents)
    except FileNotFoundError:
        print(f"Error: File '{filename}' not found.")

def count_lines(filename):
    try:
        with open(filename, "r") as file:
            lines = file.read().splitlines()
            num_lines = len(lines)
            print(f"Number of lines in the file: {num_lines}")
    except FileNotFoundError:
        print(f"Error: File '{filename}' not found.")

def count_unique_words(filename):
    try:
        with open(filename, "r") as file:
            words = file.read().split()
            num_unique_words = len(set(words))
            print(f"Number of unique words in the file: {num_unique_words}")
    except FileNotFoundError:
        print(f"Error: File '{filename}' not found.")

def find_word_frequency(filename):
    try:
        with open(filename, "r") as file:
            words = file.read().split()
            word_frequency = Counter(words)
            print("Word Frequency:")
            for word, frequency in word_frequency.items():
                print(f"{word}: {frequency}")
    except FileNotFoundError:
        print(f"Error: File '{filename}' not found.")

def show_random_line(filename):
    try:
        with open(filename, "r") as file:
            lines = file.read().splitlines()
            random_line = random.choice(lines)
            print("Random Line:")
            print(random_line)
    except FileNotFoundError:
        print(f"Error: File '{filename}' not found.")

def file_explorer():
    filename = input("Enter the filename: ")

    while True:
        print("\nMenu:")
        print("1. Display File Contents")
```

```
        print("2. Count Number of Lines")
        print("3. Count Number of Unique Words")
        print("4. Find Word Frequency")
        print("5. Show Random Line")
        print("6. Exit")

        choice = input("Enter your choice (1-6): ")

        if choice == "1":
            display_file_contents(filename)
        elif choice == "2":
            count_lines(filename)
        elif choice == "3":
            count_unique_words(filename)
        elif choice == "4":
            find_word_frequency(filename)
        elif choice == "5":
            show_random_line(filename)
        elif choice == "6":
            print("Exiting File Explorer.")
            break
        else:
            print("Invalid choice. Please enter a valid option.")

# Call the file_explorer function to start the program
file_explorer()
```

```
Enter the filename: samp.txt

Menu:
1. Display File Contents
2. Count Number of Lines
3. Count Number of Unique Words
4. Find Word Frequency
5. Show Random Line
6. Exit
Enter your choice (1-6): 1
File Contents:
What is data science used for?
Data science is used to study data in four main ways:

1. Descriptive analysis
Descriptive analysis examines data to gain insights into what happened or wha
t is happening in the data environment. It is characterized by data visualiza
tions such as pie charts, bar charts, line graphs, tables, or generated narra
tives. For example, a flight booking service may record data like the number
of tickets booked each day. Descriptive analysis will reveal booking spikes,
booking slumps, and high-performing months for this service.

2. Diagnostic analysis
Diagnostic analysis is a deep-dive or detailed data examination to understand
why something happened. It is characterized by techniques such as drill-down,
data discovery, data mining, and correlations. Multiple data operations and t
ransformations may be performed on a given data set to discover unique patter
ns in each of these techniques.For example, the flight service might drill do
wn on a particularly high-performing month to better understand the booking s
pike. This may lead to the discovery that many customers visit a particular c
ity to attend a monthly sporting event.

3. Predictive analysis
Predictive analysis uses historical data to make accurate forecasts about dat
a patterns that may occur in the future. It is characterized by techniques su
ch as machine learning, forecasting, pattern matching, and predictive modelin
g. In each of these techniques, computers are trained to reverse engineer cau
sality connections in the data.For example, the flight service team might use
data science to predict flight booking patterns for the coming year at the st
art of each year. The computer program or algorithm may look at past data and
predict booking spikes for certain destinations in May. Having anticipated th
eir customerâ€™s future travel requirements, the company could start targeted
advertising for those cities from February.

4. Prescriptive analysis
Prescriptive analytics takes predictive data to the next level. It not only p
redicts what is likely to happen but also suggests an optimum response to tha
t outcome. It can analyze the potential implications of different choices and
recommend the best course of action. It uses graph analysis, simulation, comp
lex event processing, neural networks, and recommendation engines from machin
e learning.

Back to the flight booking example, prescriptive analysis could look at histo
rical marketing campaigns to maximize the advantage of the upcoming booking s
pike. A data scientist could project booking outcomes for different levels of
marketing spend on various marketing channels. These data forecasts would giv
e the flight booking company greater confidence in their marketing decisions.
```

```
Menu:
1. Display File Contents
2. Count Number of Lines
3. Count Number of Unique Words
4. Find Word Frequency
5. Show Random Line
6. Exit
Enter your choice (1-6): 2
Number of lines in the file: 16

Menu:
1. Display File Contents
2. Count Number of Lines
3. Count Number of Unique Words
4. Find Word Frequency
5. Show Random Line
6. Exit
Enter your choice (1-6): 3
Number of unique words in the file: 232

Menu:
1. Display File Contents
2. Count Number of Lines
3. Count Number of Unique Words
4. Find Word Frequency
5. Show Random Line
6. Exit
Enter your choice (1-6): 5
Random Line:


Menu:
1. Display File Contents
2. Count Number of Lines
3. Count Number of Unique Words
4. Find Word Frequency
5. Show Random Line
6. Exit
Enter your choice (1-6): 6
Exiting File Explorer.
```

In [ ]:
```
Question5. [File Searcher]. Develop an application in Python to read through t
(“mbox-short.txt”) and when you find line that starts with “From”, you will sp
words using the split function. We are interested in who sent the message, whi
second word on the From line: From stephen.marquard@uct.ac.za Sat Jan 5 09:14:
You will parse the From line and print out the second word for each From line,
also count the number of From (not From:) lines and print out a count at the e
```

In [11]:
```
def read_email_data(mbox-short.txt):
    from_count = 0

    try:
        with open("mbox-short.txt", "r") as file:
            for line in file:
                if line.startswith("From "):  # Make sure to include the space
                    from_count += 1
                    words = line.strip().split()
                    sender = words[1]
                    print(sender)

        print(f"Total number of From lines: {from_count}")

    except FileNotFoundError:
        print(f"Error: File '{mbox-short.txt}' not found.")

# Call the function to read email data and extract sender information
read_email_data("mbox-short.txt")
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[11], line 19
     16         print(f"Error: File '{mbox-short.txt}' not found.")
     18 # Call the function to read email data and extract sender information
---> 19 read_email_data("mbox-short.txt")

TypeError: read_email_data() takes 0 positional arguments but 1 was given
```

In [ ]:
```
Question6. Write a program to read and write CSV files
 File Creation: Create MS Excel file (“student_marks.csv”) with 5 rows of stu
name, mark1, mark2, mark3, mark4. Use comma to separate each value in a row.
 File Display: Now, open your CSV file and display the file contents row by r
information at: https://docs.python.org/3/library/csv.html).
 File Writing: Now, open (“student_marks.csv”) for writing. Ask user to enter
followed by 4 marks for one new student and write them onto the file.
```

In [3]:
```python
import csv

def create_student_marks_file():
    data = [
        ["John Doe", 85, 78, 90, 88],
        ["Jane Smith", 92, 89, 78, 95],
        ["Mike Johnson", 78, 85, 80, 92],
        ["Sarah Lee", 89, 92, 85, 90],
        ["Chris Williams", 76, 80, 78, 84]
    ]

    with open("student_marks.csv", "w", newline="") as file:
        writer = csv.writer(file)
        writer.writerow(["Student Name", "Mark1", "Mark2", "Mark3", "Mark4"])
        writer.writerows(data)

# Call the function to create the CSV file
create_student_marks_file()
```

In [4]:
```python
def display_student_marks_file():
    with open("student_marks.csv", "r") as file:
        reader = csv.reader(file)
        for row in reader:
            print(",".join(row))

# Call the function to display the CSV file contents
display_student_marks_file()
```

```
Student Name,Mark1,Mark2,Mark3,Mark4
John Doe,85,78,90,88
Jane Smith,92,89,78,95
Mike Johnson,78,85,80,92
Sarah Lee,89,92,85,90
Chris Williams,76,80,78,84
```

In [6]:
```
def write_student_details():
    student_name = input("Enter student name: ")
    marks = []
    for i in range(4):
        mark = float(input(f"Enter mark {i + 1}: "))
        marks.append(mark)

    with open("student_marks.csv", "a", newline="") as file:
        writer = csv.writer(file)
        writer.writerow([student_name] + marks)

# Call the function to write new student details
write_student_details()
```

```
Enter student name: joel
Enter mark 1: 56
Enter mark 2: 67
Enter mark 3: 78
Enter mark 4: 89
```

In [ ]: