**Que 1.**

lst=[1.5,'Python', -5, 4, 0.8, -3.2, 'C++']

lst.append(0.8)

lst1=[4,1.5,6,0.8]

lst=lst+lst1

lst.insert(1,'daivik')

lst.remove('C++')

lst.pop(4)

lst.index(-5)  *#to display*

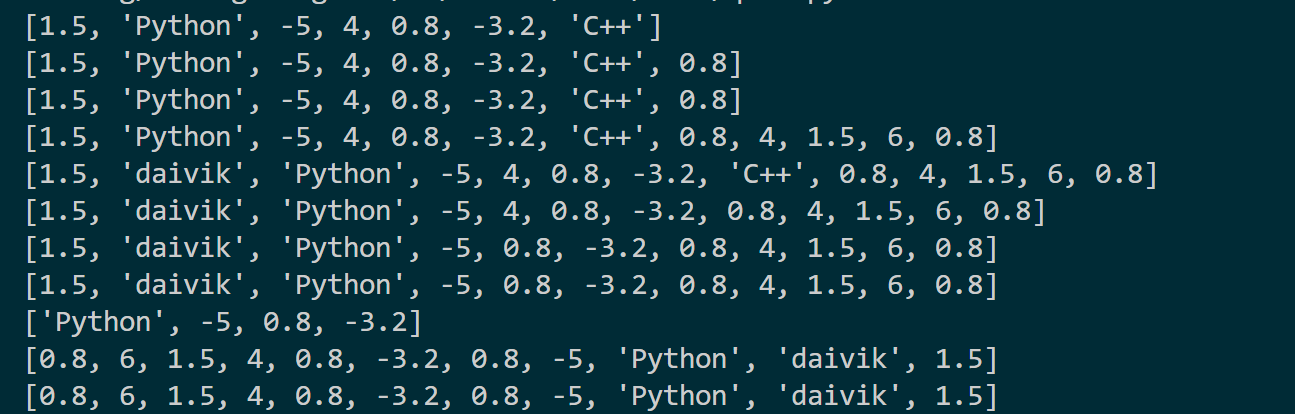
lst.count(4)

print(lst[2:6])

lst.reverse()

lst2=lst.copy()

print(lst2)



**que 2.**

**def** swapWithTemp(x,y):

    print(**f**"Before Swaping:\nx={x}\ny={y}")

    z=x

    x=y

    y=z

    print(**f**"After Swaping:\nx={x}\ny={y}")

**def** swapWithOutTemp(x,y):

    print(**f**"Before Swaping:\nx={x}\ny={y}")

    x+=y

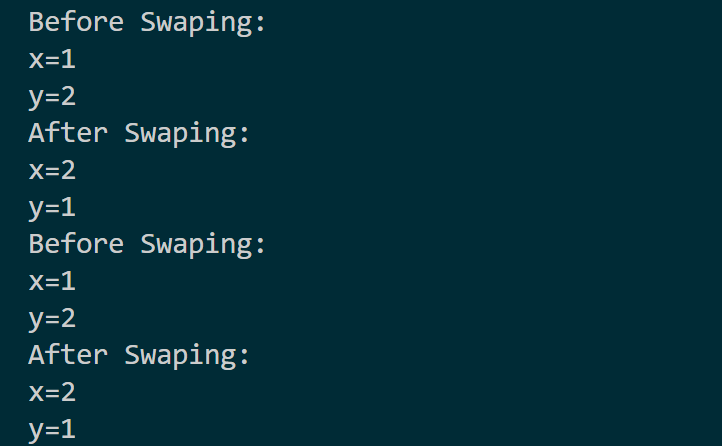
    y=x-y

    x=x-y

    print(**f**"After Swaping:\nx={x}\ny={y}")

swapWithTemp(1,2)

swapWithOutTemp(1,2)



**que3.**

**def** multiFunct(a,b,c):

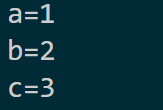
    return a,b,c

**def** main():

    a,b,c=multiFunct(1,2,3)

    print(**f**"a={a}\nb={b}\nc={c}")

main()



**que4.**

'''

Inventory Management: In a retail store, use a list to keep track of product stock

levels, and implement functions to add, remove, and update items in the

inventory.

'''

items = {

    "Rice": 5,

    "Wheat": 10,

    "Biscuit": 12,

}

**def** add\_item(item, quantity):

    items[item] = quantity

**def** remove\_item(item):

    if item in items:

        del items[item]

    else:

        print(**f**"{item} not found in the inventory.")

**def** update\_item(item, new\_quantity):

    if item in items:

        items[item] = new\_quantity

    else:

        print(**f**"{item} not found in the inventory.")

**def** main():

    while True:

        print(**f**"Choose an option:\n1: Add item\n2: Remove item\n3: Update item\n4: Exit\nCurrently the items are {items}")

        choice = input("Enter your choice: ")

        if choice == '1':

            item = input("Enter item to add: ")

            quantity = int(input("Enter item quantity to add: "))

            add\_item(item, quantity)

            print(**f**"{item} added to inventory.")

        elif choice == '2':

            item = input("Enter item to remove: ")

            remove\_item(item)

            print(**f**"{item} removed from inventory.")

        elif choice == '3':

            item = input("Enter item to update: ")

            new\_quantity = int(input("Enter new item quantity: "))

            update\_item(item, new\_quantity)

            print(**f**"{item} quantity updated in the inventory.")

        elif choice == '4':

            break

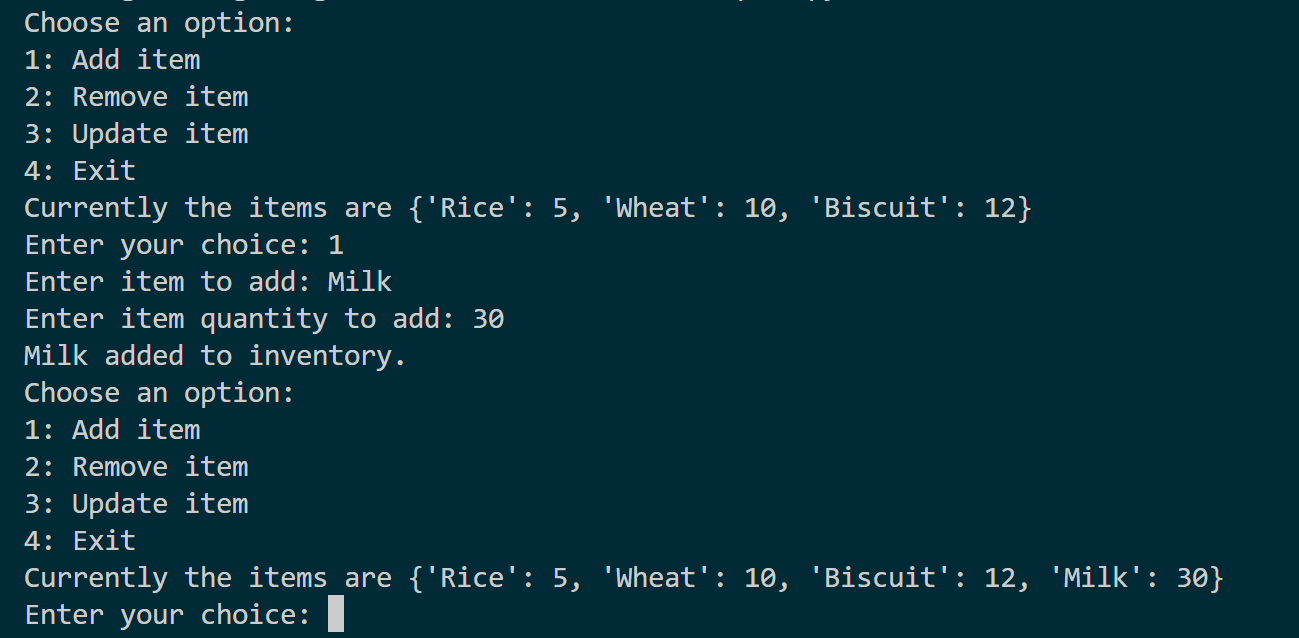
        else:

            print("Not a valid option. Please choose 1, 2, 3, or 4.")

    print("Inventory Management System is closed.")

if \_\_name\_\_ == "\_\_main\_\_":

    main()



**que5**

'''

To-Do List Application: Create a to-do list application where users can add,

remove, and update tasks using a list.

'''

tasks = []

**def** add(task):

    tasks.append(task)

    print(**f**"Task '{task}' added to the to-do list.")

**def** remove(task):

    if task in tasks:

        tasks.remove(task)

        print(**f**"Task '{task}' removed from the to-do list.")

    else:

        print(**f**"Task '{task}' not found in the to-do list.")

**def** update(old\_task, new\_task):

    if old\_task in tasks:

        index = tasks.index(old\_task)

        tasks[index] = new\_task

        print(**f**"Task '{old\_task}' updated to '{new\_task}'.")

    else:

        print(**f**"Task '{old\_task}' not found in the to-do list.")

**def** display():

    if tasks:

        print("To-Do List:")

        for i, task in enumerate(tasks, start=1):

            print(**f**"{i}. {task}")

    else:

        print("To-Do List is empty.")

while True:

    print("\nTo-Do List Application")

    print("1. Add Task")

    print("2. Remove Task")

    print("3. Update Task")

    print("4. Display Tasks")

    print("5. Exit")

    choice = input("Enter your choice (1/2/3/4/5): ")

    if choice == '1':

        task = input("Enter the task to add: ")

        add(task)

    elif choice == '2':

        task = input("Enter the task to remove: ")

        remove(task)

    elif choice == '3':

        old\_task = input("Enter the task to update: ")

        new\_task = input("Enter the new task: ")

        update(old\_task, new\_task)

    elif choice == '4':

        display()

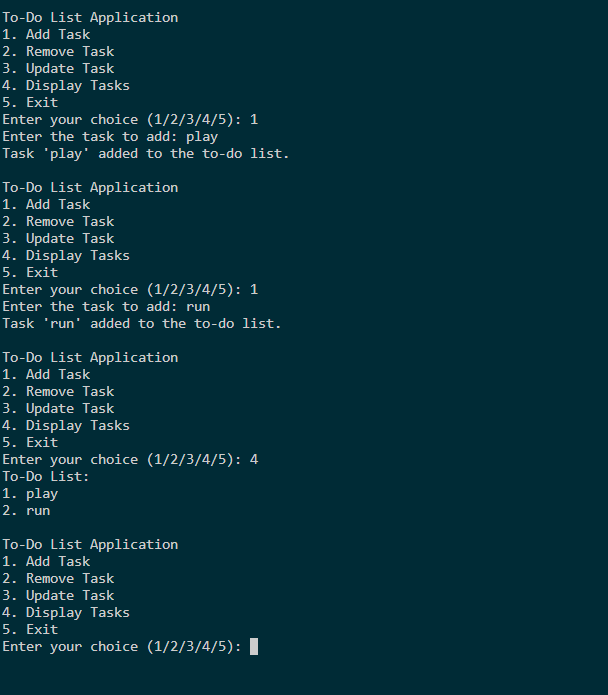
    elif choice == '5':

        print("To-Do List Application is closed.")

        break

    else:

        print("Invalid choice. Please select a valid option (1/2/3/4/5).")



**que6.**

data\_list = []

**def** add\_data():

    while True:

        input\_data = input("Enter data in this format => post\_no, number of Likes, number of comments, number of shares\nOr type 'Exit' to finish: ")

        if input\_data.lower() == 'exit':

            break

        post\_no, likes, comments, shares = map(int, input\_data.split(','))

        total\_interactions = likes + comments + shares

        data\_list.append([post\_no, likes, comments, shares, total\_interactions])

add\_data()

if data\_list:

    total\_interactions = sum(item[4] for item in data\_list)

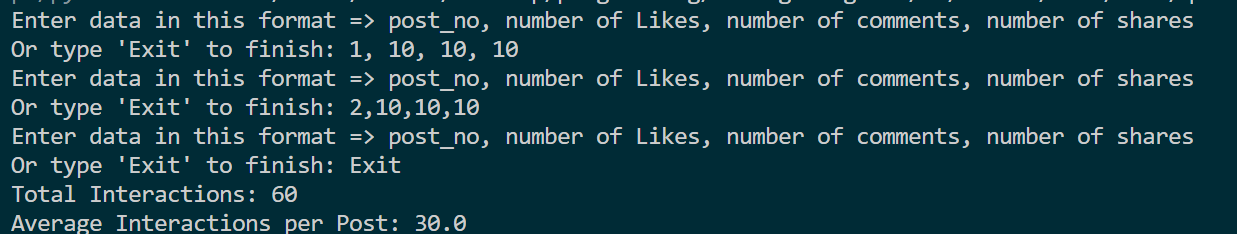
    average\_interactions = total\_interactions / len(data\_list)

    print(**f**"Total Interactions: {total\_interactions}")

    print(**f**"Average Interactions per Post: {average\_interactions}")

else:

    print("No data entered.")



**que7.**

student\_data = []

**def** add\_student(name, score):

    student\_data.append((name, score))

    print(**f**"Added: {name} - Test Score: {score}")

**def** calculate\_average\_score():

    if not student\_data:

        print("No student data available.")

    else:

        total\_score = sum(score for \_, score in student\_data)

        average\_score = total\_score / len(student\_data)

        print(**f**"Average Test Score: {average\_score**:.2f**}")

while True:

    print("\nStudent Grade Management System")

    print("1. Add Student")

    print("2. Calculate Average Score")

    print("3. Exit")

    choice = input("Enter your choice (1/2/3): ")

    if choice == '1':

        name = input("Enter student's name: ")

        score = float(input("Enter test score: "))

        add\_student(name, score)

    elif choice == '2':

        calculate\_average\_score()

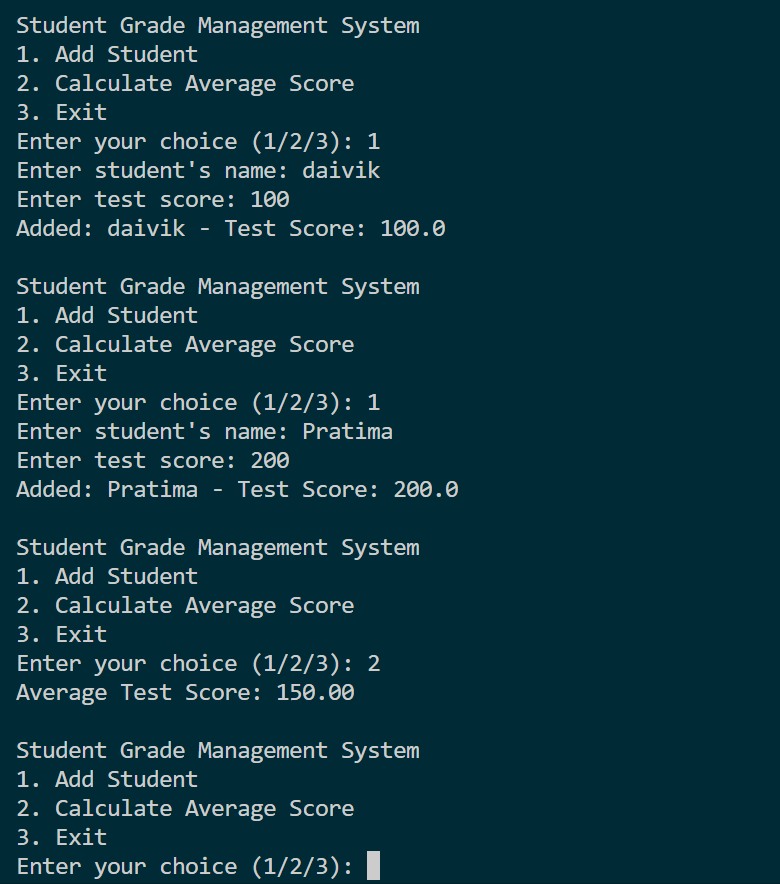
    elif choice == '3':

        print("Student Grade Management System is closed.")

        break

    else:

        print("Invalid choice. Please select a valid option (1/2/3).")



**que8.**

attendee\_list = []

**def** register\_attendee(name):

    attendee\_list.append(name)

    print(**f**"Registered: {name}")

**def** check\_in\_attendee(name):

    if name in attendee\_list:

        print(**f**"Checked in: {name}")

    else:

        print(**f**"{name} not found in the attendee list.")

**def** generate\_attendee\_list():

    if attendee\_list:

        print("Attendee List:")

        for i, attendee in enumerate(attendee\_list, start=1):

            print(**f**"{i}. {attendee}")

    else:

        print("No attendees registered.")

while True:

    print("\nEvent Registration System")

    print("1. Register Attendee")

    print("2. Check in Attendee")

    print("3. Generate Attendee List")

    print("4. Exit")

    choice = input("Enter your choice (1/2/3/4): ")

    if choice == '1':

        name = input("Enter attendee's name: ")

        register\_attendee(name)

    elif choice == '2':

        name = input("Enter attendee's name to check in: ")

        check\_in\_attendee(name)

    elif choice == '3':

        generate\_attendee\_list()

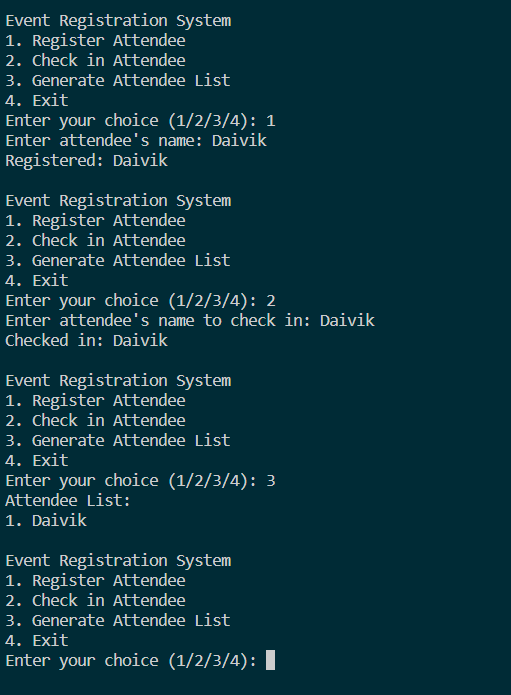
    elif choice == '4':

        print("Event Registration System is closed.")

        break

    else:

        print("Invalid choice. Please select a valid option (1/2/3/4).")



**que9.**

dictionary = {

    "Humans": "many",

    "Brave": "few",

    "key": 1

}

print(dictionary)

if "Brave" in dictionary:

    print(dictionary.get("Brave"))

else:

    print(-1)

dictionary.update({"Hello": "World"})

print(dictionary)

dictionary.pop("Hello")

print(dictionary)

dict\_copy = dictionary.copy()

print(dict\_copy)

*# Correction: Use len(dict\_copy) instead of dict\_copy.len()*

print(len(dict\_copy))

*# Deleting both dictionaries*

del dictionary

del dict\_copy

