ATSS's

Institute of Industrial and Computer Management and Research, Nigdi Pune

MCA Department

Academic Year: 2022-23

Practical Journal on Python Programming (SEM-II): IT 21 L

Submitted By:

Roll no:110

StudentName: Siddhesh k. Chavan

Seat No:

Date :

Course Outcomes:
Student will be able to

CO1: Implement python programming concepts for solving real life problems. (Apply)

CO2: Implement Advanced Internet Technologies (Apply)

ATSS's
Institute of Industrial and Computer Management and Research, Nigdi Pune
MCA Department

INDEX

Students Name: Siddhesh Kisan Chavan Roll No: 110

Sr. No	Program Title	Course Outcome	Page No.	Teacher's Sign with Date	Remarks
	Programs for understanding the data types, control flow statements, blocks and loop				
1.	Write Python Program to count the Total Number of Vowels, Consonants and Blanks in a String 2.Create Stack using list and implement necessary operation	**CO1**	7		
2.	Write a program to print following pattern A 4444 1 B C 333 0 1 C D E 22 1 0 1 D E F G 1 0 1 0 1 *	**CO1**	9		
	Programs for understanding functions, use of built in functions, user defined functions, Programs to use existing modules, packages and creating modules, packages				
3.	A list of numbers is said to be a hill if it consists of an ascending sequence followed by a descending sequence, where each of the sequences is of length at least two. Similarly, a list of numbers is said to be a valley hill if it consists of an descending sequence followed by an ascending sequence. You can assume that consecutive numbers in the input sequence are always different from each other. Write a Python function hillvalley(l) that takes a list l of integers and prints **1.** If sequence is in increase **"hill"** **2.** If sequence is in decreasing **"vally"** **3.** If sequence is in increaseing and decreasing **"hill Vally"**	**CO1**	11		
4.	Write a program which generates account number using random function	**CO1**	13		
5.	Write Python Program to Conduct a Linear Search for a Given Key Number in the List and Report Success or Failure	**CO1**	14		
6.	Write Python Program to Add Two Matrices	**CO1**	15		

	Programs for implementations of all object-oriented concepts like class, method, inheritance, polymorphism etc. (Real life examples must be covered for the implementation of objectoriented concepts)				
7.	Write Python Program to Simulate a Bank Account with Support for depositMoney, withdrawMoney and showBalance Operations	**CO1**	16		
8	Write a program to create point class with x,y,z coordinate and methods increment point, decrement point, add points , less than , greater than , equal to , check in which quadrant it lies,check whether the point is collinear and print point.		17		
	Programs for parsing of data, validations like Password, email, URL, PAN , Mobile number, IP address. Programs for Pattern finding should be covered.				
9.	Write regular expression for 1.To extract year, month and date from a string 2.To Extract only 3 digit number from string 3.To Extract all of the words and numbers from string 4.To Find out all of the words, which start with a vowel	**CO1**	19		
10.	Write a regular expression to extract to Validate IP Address in Python 1.Write regular expression to validate email address 2.Write regular expression to validate URL Regular Expression to Validate PAN Card Number in Python		20		
	Programs covering all the aspects of Exception handling, user defined exception, Multithreading should be covered. **Programs demonstrating the IO operations like reading from file, writing into file from different file types like data file, binary file, etc** **Programs to perform searching, adding, updating the content from the file**				
11.	A] An ABC company wants to perform the following tasks . 1. Copy the contents of 'FILE1.txt' to 'FILE2.txt' 2 Count the number of lines, characters, special symbols for a given file. 3 To remove the comments from python code. B] Write a python program using multithreading concept to perform above operations simultaneously. Accept the filenames from user. (HINT : Thread_obj=Thread(target=Func_A , args=(FILE1, FILE2)) where Thread_obj is thread object , Func_A is function to perform the specific task, FILE1, FILE2 are the file names passed in the form of tuple as parameter to function.	**CO1**	21		
12.	Write a demo Program for synchronization using RLock. Accept the two numbers from user and calculate factorial of both the numbers. (Hint : use Rlock(), aquire(), release() methods)	**CO1**	23		
13.	Write a program that reads the contents of the file and counts the occurrences of each letter. Prompt the user to enter the filename.	**CO1**	24		
	Program for performing CRUD operation with MongoDB and Python				

1. 2. **14.**	Perform the following operations using Python 1. Create collection 'emp' and insert five documents. 2. Write a Program for performing CRUD (Create , Read, Update, Delete) operation.	**CO1**	25		
	Basic programs with NumPy as Array, Searching and Sorting, date & time and String handling				
15 1. 2.	Create a 5x4 numpy array and find it's column-wise mean, max,min,sum Create two 2-d Numpy Arrays (Matrix A, Matrix B) and perform the following operation on matrix a. Addition (A+B) b. Multiplication (AxB) c. Scalar Multiplication (A x integer or integer x A) d. Transpose of Matrix 3. Write a Program to perform following using NumPy Arrays. a. Create a 5-by-5 array of random integers between 0 (inclusive) and 10 (exclusive) b. Create a sequence of equally gapped 5 numbers in the range 0 to 100 (both inclusive) c. Convert a 1-D array to a 3-D array d. Convert all the elements of a numpy array from float to integer datatype e. Stack two numpy arrays horizontally and vertically f. From two numpy arrays, extract the indexes in which the elements in the two arrays match (hint- np.where(a == b))	**CO1**	27		
	Programs for series and data frames should be covered. **Programs to demonstrate data pre-processing and data handling with data frame** **Program for data visualization should be covered.**				

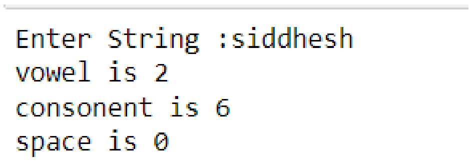
16.	A] Use **Automobile Dataset (Automobile_data.csv)** and perform following operations for data analysis. This Dataset has different characteristics of an auto such as body-style, wheel-base, engine- type, price, mileage, horsepower, etc. 1. From the given dataset print the first and last five rows 2. Find the most expensive car company name 3. Print All Toyota Cars details 4. Count total cars per company 5. Find each company's Highest price car 6. Find the average mileage of each car making company 7. Sort all cars by Price column 8. Apply the rank to the cars average-mileage. (highest average-mileage – rank 1, so on) 9. Concatenate two data frames using the following conditions 10. Create two data frames using the following two Dicts, Merge two data frames, and append the second data frame as a new column to the first data frame. a. Car_Price = {'Company': ['Toyota', 'Honda', 'BMV', 'Audi'], 'Price': [23845, 17995, 135925 , 71400]} b. Car_Horsepower = {'Company': ['Toyota', 'Honda', 'BMV', 'Audi'], 'horsepower': [141, 80, 182 , 160]} Refer : https://www.kaggle.com/datasets/toramky/automobile-dataset	**CO1**	31		
17.	B] Use company_sales_data.csv for this exercise. Read this file using Pandas or NumPy or using in-built matplotlib function. Read all product sales data and show it using a multiline plot. Display the number of units sold per month for each product using multiline plots. (i.e., Separate Plotline for each product).		35		

1.Write Python Program to
count the Total Number of Vowels, Consonants and Blanks in a String

Solution: Program

```
vowel=0
consonent=0
space=0
str=input('Enter String :')
for ch in str:
    if(ch=='a' or ch=='e' or ch=='i' or ch=='o' or ch=='u'):
        vowel+=1
    elif "a"<ch<"z":
        consonent+=1
    elif ch==" ":
        space+=1
print(f"vowel is {vowel}")
print(f"consonent is {consonent}")
print(f"space is {space}")
```

Output:
Screen Shot



```
Enter String :siddhesh
vowel is 2
consonent is 6
space is 0
```

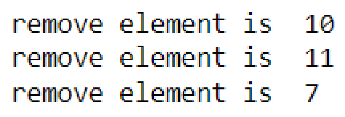
1.2 Create Stack using list and implement necessary operation

Solution: Program

```
class Stack:
    def __init__(self, size):
        self.value =[None]*size
        self.top= -1
        self.length = 0
        self.size = size
    def isEmpty(self):
        return self.length ==0
    def isFull(self):
        return self.length ==self.size
    def push(self, value):
        if self.isFull():  return "Stack is full"
        self.top+=1
        self.value[self.top] =value
        self.length +=1
    def pop(self):
        if self.isEmpty(): return "Stack is empty"
        removeNode =self.value[self.top]
        self.top-=1
        self.length -=1
        return removeNode
    def display(self):
        if self. isEmpty(): return "Stack is empty"
        for i in range(e, self.top+1):
            print(self.value[i])

s =Stack(5)
s.push(12)
s.push(2)
s.push(7)
s.push(11)
s.push(10)
print("remove element is ",s.pop())
print("remove element is ",s.pop())
print("remove element is ",s.pop())
```

Output:

Screen Shot



```
remove element is  10
remove element is  11
remove element is  7
```

2. Write a program to print following pattern

```
A        4444        1
B C      333        0 1
C D E    22        1 0 1
D E F G                 1   0 1 0 1
*                          *
* *                      * *
* * *                  * * *
* * * *              * * * *
* * * * *        * * * * *
```

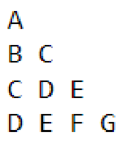
Solution: Program

1.

```
for i in range(65,69):
    z=i
    for j in range(65,i+1):
        print(chr(z),end=" ")
        z=z+1
    print()
```

Output:

Screen Shot



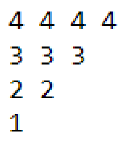
```
A
B C
C D E
D E F G
```

2.

```
rows=4
for i in range(rows,0,-1):
    num=i
    for j in range(0,i):
        print(num,end=' ')
    print("\r")
```

Output:

Screen Shot



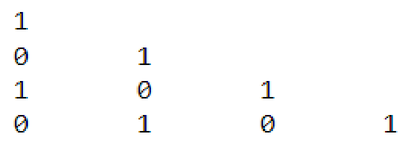
```
4 4 4 4
3 3 3
2 2
1
```

3.

```
n=4
for row in range(0,n):
    for col in range(0,row+1):
        if(((row+col)%2)==0):
            print("1",end="")
        else:
            print("0",end="")
        print("\t",end="")
    print("")
```

Output:

Screen Shot



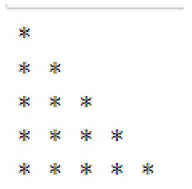
```
1
0       1
1       0       1
0       1       0       1
```

4.

```
rows=5
for i in range(1,rows+1):
    for j in range(1,i+1):
        print("*",end=' ')
    print('')
```

Output:

Screen Shot



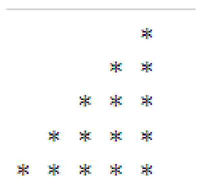
```
*
* *
* * *
* * * *
* * * * *
```

5.

```
rows=5
for i in range(1,rows+1):
    for j in range(1,rows+1):
        if(j<=rows-i):
            print(' ',end=' ')
        else:
            print('*',end=' ')
    print()
```

Output:

Screen Shot



```
        *
      * *
    * * *
  * * * *
* * * * *
```

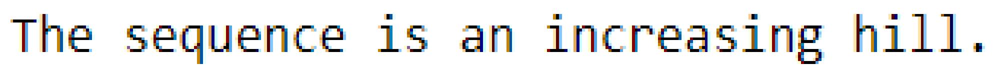
3. A list of numbers is said to be a hill if it consists of an ascending sequence followed by a descending sequence, where each of the sequences is of length at least two.
Similarly, a list of numbers is said to be a valley hill if it consists of an descending sequence followed by an ascendingsequence. You can assume that consecutive numbers in the input sequence are always different from each other.
Write a Python function hillvalley(l) that takes a list l ofintegers and prints
1.If sequence is in increase "hill"
2.If sequence is in decreasing "vally"
3.If sequence is in increaseing and decreasing "hill Vally"

Solution: Program

```
def hillvalley(l):
    n = len(l)

    if n < 4:
        print("Invalid sequence. Length should be at least 4.")
        return
    if l[0] < l[1]:
        increasing = True
    else:
        increasing = False
    if increasing:
        for i in range(1, n-1):
            if l[i] >= l[i+1]:
                increasing = False
                break
        if increasing:
            print("The sequence is an increasing hill.")
            return
    else:
        for i in range(1, n-1):
            if l[i] <= l[i+1]:
                increasing = True
                break
        if not increasing:
            print("The sequence is a decreasing valley.")
            return
    for i in range(1, n-1):
        if l[i] == l[i+1]:
            print("Invalid sequence. Consecutive numbers should be different.")
            return
    increasing = True
    for i in range(1, n-1):
        if increasing:
            if l[i] >= l[i+1]:
                increasing = False
        else:
            if l[i] <= l[i+1]:
                print("The sequence is an increasing-decreasing hill valley.")
                return
    print("The sequence is neither an increasing hill nor a decreasing valley.")
sequence = [1, 2, 3, 4, 5]
hillvalley(sequence)
```

Output:

Screen Shot



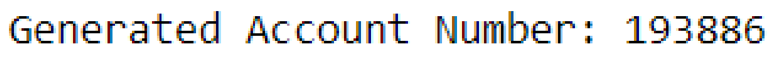
```
The sequence is an increasing hill.
```

4.Write a program which generates account number usingrandom function

Solution: Program

```
import random
def generate_account_number():
    account_number = random.randint(100000, 999999)
    return account_number
account_number = generate_account_number()
print("Generated Account Number:", account_number)
```

Output:
Screen Shot



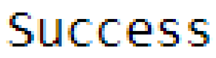
```
Generated Account Number: 193886
```

5.Write Python Program to Conduct a Linear Search or a Given Key Number in the List and Report Success or Failure

Solution: Program

```
list = [1,2,3,4,5]
def linearSea(list, key):
    if key>=len(list) or key<0: return "Failure"
    for i in range(len(list)):
        if i==key: return "Success"
print(linearSea(list, 4))
```

Output:
Screen Shot



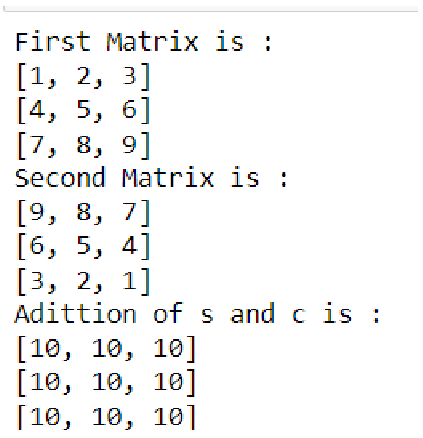
```
Success
```

6.Write Python Program to Add Two Matrices.

Solution: Program

```
s=[[1,2,3],[4,5,6],[7,8,9]]
c=[[9,8,7],[6,5,4],[3,2,1]]
ress=[[0,0,0],[0,0,0],[0,0,0]]
for i in range(len(s)):
    for j in range(len(s[0])):
        ress[i][j]=s[i][j]+c[i][j]
print("First Matrix is : ")
for i in s:
    print(i)
print("Second Matrix is : ")
for i in c:
    print(i)
print("Adittion of s and c is : ")
for i in ress:
    print(i)
```

Output:
Screen Shot



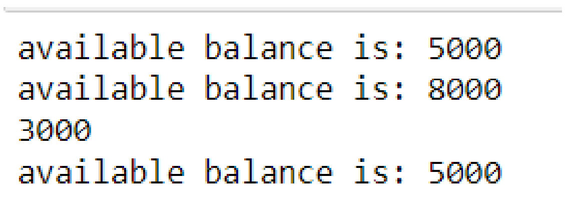
```
First Matrix is :
[1, 2, 3]
[4, 5, 6]
[7, 8, 9]
Second Matrix is :
[9, 8, 7]
[6, 5, 4]
[3, 2, 1]
Adittion of s and c is :
[10, 10, 10]
[10, 10, 10]
[10, 10, 10]
```

7. Write Python Program to Simulate a Bank Account with Support for depositMoney, withdrawMoney andshowBalance Operations

Solution: Program

```
class Bank:
    def __init__(self, amount):
        self.money = amount
    def depositMoney(self, amount):
        self.money+=amount
        print(f"available balance is: {self.money}")
    def withdrawMoney(self, amount):
        if self.money < amount:
            print(f"withdrawal amount is greater than deposited money ")
            return
        self.money-=amount
        return self.showBalance()
    def showBalance(self):
        print(f"current balance is {self.money}")
def Bank(name, amount):
    name = name
    amount = amount
    def display_amount():
        return f"available balance is: {amount}"
    def deposite(deposite_amount):
        nonlocal amount
        amount += deposite_amount
        return display_amount()
    def withdraw_amount(withraw_amount):
        nonlocal amount
        if withraw_amount > amount:
            return f"sorry can not do this transaction! your available balance is {amount}"
        amount -= withraw_amount
        return withraw_amount
    return {"display_amount":display_amount, "deposite":deposite, "withdraw_amount":withdraw_amount}
jack = Bank("Jack", 5000)
print(jack["display_amount"]())
print(jack["deposite"](3000))
print(jack["withdraw_amount"](3000))
print(jack["display_amount"]())
```

Output:
Screen Shot



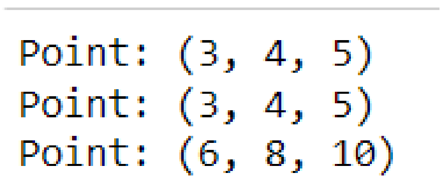
```
available balance is: 5000
available balance is: 8000
3000
available balance is: 5000
```

8. Write a program to create point class with x,y,zcoordinate and methods increment point, decrement point, add points , less than , greaterthan , equal to , check in which quadrant it
lies,check whether the point is collinear andprint point. Write a program to create point class with x,y,zcoordinate and methods increment point, decrement point, add points , less than , greaterthan , equal to , check in which quadrant it
lies,check whether the point is collinear andprint point.

Solution: Program

```
class Point:
    def __init__(self, x, y, z):
        self.x = x
        self.y = y
        self.z = z
    def increment_point(self, dx, dy, dz):
        self.x += dx
        self.y += dy
        self.z += dz
    def decrement_point(self, dx, dy, dz):
        self.x -= dx
        self.y -= dy
        self.z -= dz
    def add_points(self, other_point):
        new_x = self.x + other_point.x
        new_y = self.y + other_point.y
        new_z = self.z + other_point.z
        return Point(new_x, new_y, new_z)
    def __lt__(self, other_point):
        return self.x < other_point.x and self.y < other_point.y and self.z < other_point.z
    def __gt__(self, other_point):
        return self.x > other_point.x and self.y > other_point.y and self.z > other_point.z
    def __eq__(self, other_point):
        return self.x == other_point.x and self.y == other_point.y and self.z == other_point.z
    def get_quadrant(self):
        if self.x > 0 and self.y > 0 and self.z > 0:
            return "First Quadrant"
        elif self.x < 0 and self.y > 0 and self.z > 0:
            return "Second Quadrant"
        elif self.x < 0 and self.y < 0 and self.z > 0:
            return "Third Quadrant"
        elif self.x > 0 and self.y < 0 and self.z > 0:
            return "Fourth Quadrant"
        elif self.x > 0 and self.y > 0 and self.z < 0:
            return "Fifth Quadrant"
        elif self.x < 0 and self.y > 0 and self.z < 0:
            return "Sixth Quadrant"
        elif self.x < 0 and self.y < 0 and self.z < 0:
            return "Seventh Quadrant"
        elif self.x > 0 and self.y < 0 and self.z < 0:
            return "Eighth Quadrant"
        else:
            return "Origin"
    def is_collinear(self, other_point1, other_point2):
        slope1 = (other_point1.y - self.y) / (other_point1.x - self.x)
        slope2 = (other_point2.y - self.y) / (other_point2.x - self.x)
        return slope1 == slope2
    def print_point(self):
        print(f"Point: ({self.x}, {self.y}, {self.z})")
point1 = Point(1, 2, 3)
point2 = Point(4, 5, 6)
point3 = Point(7, 8, 9)
point1.increment_point(2, 2, 2)
point1.print_point()
point2.decrement_point(1, 1, 1)
point2.print_point()
result_point = point1.add_points(point2)
result_point.print_point()
```

Output:
Screen Shot



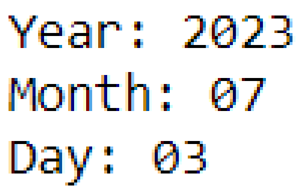
```
Point: (3, 4, 5)
Point: (3, 4, 5)
Point: (6, 8, 10)
```

9. Write regular expression for
1.To extract year, month and date from a string 2.To Extract only 3 digit number from string
3.To Extract all of the words and numbers from string 4.To Find out all of the words, which start with a vowel

Solution: Program

```
import re

string = "Today's date is 2023-07-03"
pattern = r'(\d{4})-(\d{2})-(\d{2})'

match = re.search(pattern, string)
if match:
    year = match.group(1)
    month = match.group(2)
    day = match.group(3)
    print("Year:", year)
    print("Month:", month)
    print("Day:", day)
```

Output:
Screen Shot



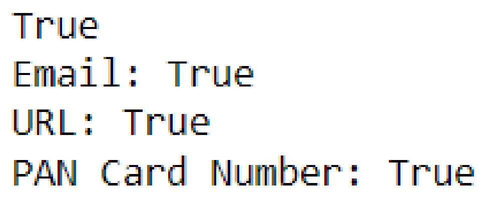
```
Year: 2023
Month: 07
Day: 03
```

10. Write a regular expression to extract to Validate IP Addressin Python
Write regular expression to validate email address2.Write regular expression to validate URL
Regular Expression to Validate PAN Card Number inPython

Solution: Program

```
import re

def is_valid_ip_address(ip_address):
    pattern = r'^(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$'
    return re.match(pattern, ip_address) is not None

ip = "192.168.0.1"
print(is_valid_ip_address(ip))


def is_valid_email(email):
    pattern = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'
    return re.match(pattern, email) is not None

def is_valid_url(url):
    pattern = r'^(http|https):\/\/[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,}(\/.*)?$'
    return re.match(pattern, url) is not None

def is_valid_pan_card_number(pan_number):
    pattern = r'^[A-Z]{5}[0-9]{4}[A-Z]$'
    return re.match(pattern, pan_number) is not None
email = "sid@gmail.com"
url = "https://www.youtube.com"
pan_card_number = "PQRSS1234A"

print("Email:", is_valid_email(email))
print("URL:", is_valid_url(url))
print("PAN Card Number:", is_valid_pan_card_number(pan_card_number))
```

Output:
Screen Shot



```
True
Email: True
URL: True
PAN Card Number: True
```

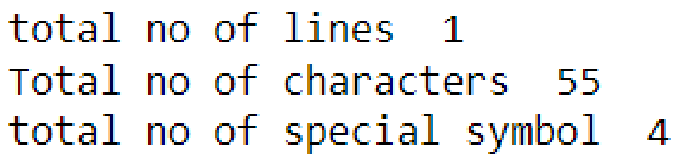
11. A] An ABC company wants to perform the following tasks .
Copy the contents of 'FILE1.txt' to 'FILE2.txt'
Count the number of lines, characters, special symbols for a given file.
To remove the comments from python code.
B] Write a python program using multithreading concept to perform above operations simultaneously. Accept the filenames from user.
(HINT : Thread_obj=Thread(target=Func_A , args=(FILE1, FILE2))
where Thread_obj is thread object , Func_A is function to perform the specific task, FILE1, FILE2 are the file names passed in the form of tuple as parameter to function.

Solution: Program

```
from threading import *
from string import punctuation

def copy():
    try:
        with open('file1.txt','r') as f:
            f1 = open('file2.txt','w')
            f1.writelines(f)
            f1.close()
    except:
        print("Something Went Wrong")

def count():
    special_symbol = set(punctuation)
    try:
        with open('file1.txt','r') as f:
            char_len=0
            symbol=0
            file = f.read()
            line_len = len(file.split('\n'))
            for line in file:
                if '\n' in line:
                    symbol+=1
                char_len+=len(line)
                for char in line:
                    if char in special_symbol:
                        symbol+=1
        print("total no of lines ",line_len)
        print("Total no of characters ",char_len)
        print("total no of special symbol ",symbol)
    except:
        print("Something Went Wrong")
```

```
t1 = Thread(target=copy)
t2 = Thread(target=count)
t1.start()
t2.start()
```

Output:
Screen Shot



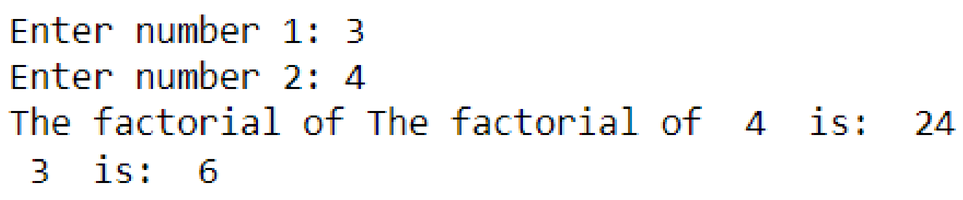
```
total no of lines  1
Total no of characters  55
total no of special symbol  4
```

12. Write a demo Program for synchronization using RLock. Accept the two numbers from user and calculate factorial of both the numbers.
(Hint : use Rlock(), aquire(), release() methods)

Solution: Program

```
import time
from threading import *

I=RLock()
def factorial(n):
   I.acquire()
   if n==0:
      result=1
   else:
      result=n*factorial(n-1)
   I.release()
   return result
def results(n):
   print("The factorial of ",n," is: ",factorial(n))
n1=int(input("Enter number 1: "))
n2=int(input("Enter number 2: "))
t1=Thread(target=results,args=(n1,))
t2=Thread(target=results,args=(n2,))
t1.start()
t2.start()
```

Output:
Screen Shot



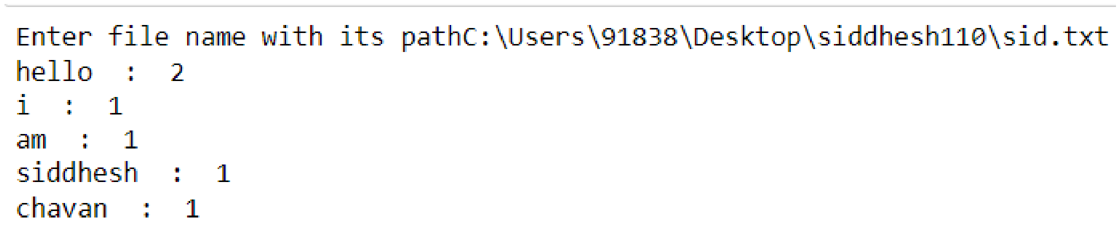
```
Enter number 1: 3
Enter number 2: 4
The factorial of The factorial of  4  is:  24
 3  is:  6
```

13. Write a program that reads the contents of the file and countsthe occurrences of each letter. Prompt the user to enter the filename.

Solution: Program

```
inp=input("Enter file name with its path")
text=open(inp,'r')
d=dict()
for line in text:
    line=line.strip()
    line=line.lower()
    words=line.split(" ")
    for word in words:
        if word in d:
            d[word]=d[word]+1
        else:
            d[word]=1
for key in list(d.keys()):
    print(key," : ",d[key])
```

Output:
Screen Shot



```
Enter file name with its pathC:\Users\91838\Desktop\siddhesh110\sid.txt
hello  :  2
i  :  1
am  :  1
siddhesh  :  1
chavan  :  1
```

14. Perform the following operations using Python

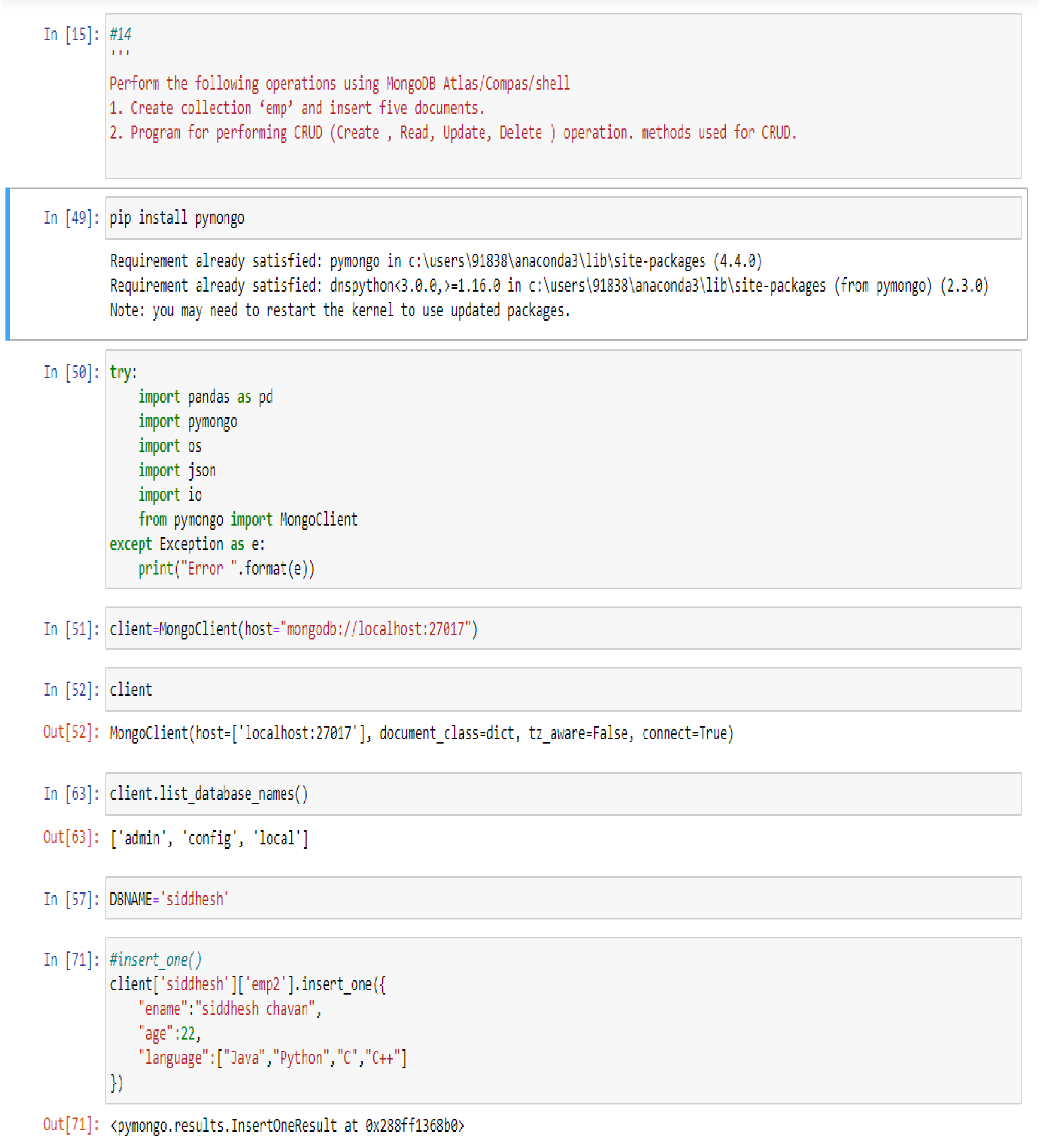
1.Create collection ‘emp’ and insert five documents.

2.Write a Program for performing CRUD (Create , Read,Update, Delete) operation.

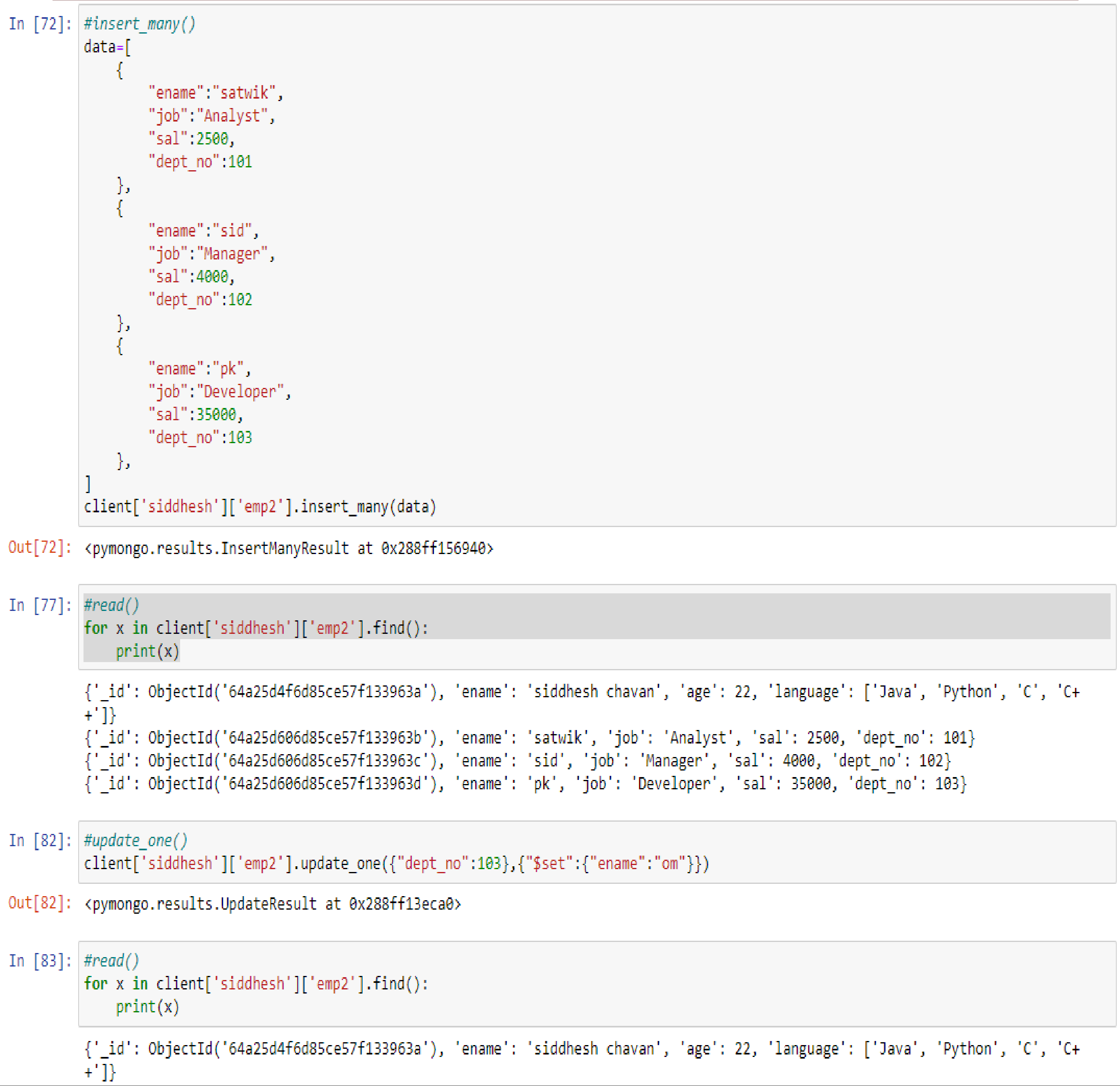
Solution: Program

Output:

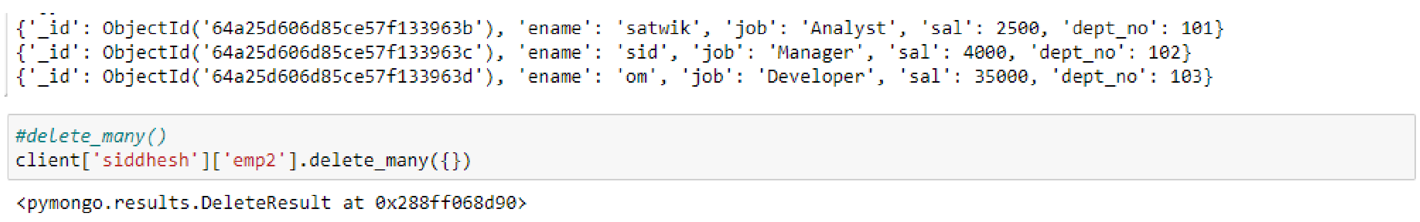
Screen Shot



```
In [15]: #14
         '''
         Perform the following operations using MongoDB Atlas/Compas/shell
         1. Create collection ‘emp’ and insert five documents.
         2. Program for performing CRUD (Create , Read, Update, Delete ) operation. methods used for CRUD.

In [49]: pip install pymongo

         Requirement already satisfied: pymongo in c:\users\91838\anaconda3\lib\site-packages (4.4.0)
         Requirement already satisfied: dnspython<3.0.0,>=1.16.0 in c:\users\91838\anaconda3\lib\site-packages (from pymongo) (2.3.0)
         Note: you may need to restart the kernel to use updated packages.

In [50]: try:
             import pandas as pd
             import pymongo
             import os
             import json
             import io
             from pymongo import MongoClient
         except Exception as e:
             print("Error ".format(e))

In [51]: client=MongoClient(host="mongodb://localhost:27017")

In [52]: client

Out[52]: MongoClient(host=['localhost:27017'], document_class=dict, tz_aware=False, connect=True)

In [63]: client.list_database_names()

Out[63]: ['admin', 'config', 'local']

In [57]: DBNAME='siddhesh'

In [71]: #insert_one()
         client['siddhesh']['emp2'].insert_one({
             "ename":"siddhesh chavan",
             "age":22,
             "language":["Java","Python","C","C++"]
         })

Out[71]: <pymongo.results.InsertOneResult at 0x288ff1368b0>
```



```
In [72]: #insert_many()
         data=[
             {
                 "ename":"satwik",
                 "job":"Analyst",
                 "sal":2500,
                 "dept_no":101
             },
             {
                 "ename":"sid",
                 "job":"Manager",
                 "sal":4000,
                 "dept_no":102
             },
             {
                 "ename":"pk",
                 "job":"Developer",
                 "sal":35000,
                 "dept_no":103
             },
         ]
         client['siddhesh']['emp2'].insert_many(data)

Out[72]: <pymongo.results.InsertManyResult at 0x288ff156940>

In [77]: #read()
         for x in client['siddhesh']['emp2'].find():
             print(x)

         {'_id': ObjectId('64a25d4f6d85ce57f133963a'), 'ename': 'siddhesh chavan', 'age': 22, 'language': ['Java', 'Python', 'C', 'C+
         +']}
         {'_id': ObjectId('64a25d606d85ce57f133963b'), 'ename': 'satwik', 'job': 'Analyst', 'sal': 2500, 'dept_no': 101}
         {'_id': ObjectId('64a25d606d85ce57f133963c'), 'ename': 'sid', 'job': 'Manager', 'sal': 4000, 'dept_no': 102}
         {'_id': ObjectId('64a25d606d85ce57f133963d'), 'ename': 'pk', 'job': 'Developer', 'sal': 35000, 'dept_no': 103}

In [82]: #update_one()
         client['siddhesh']['emp2'].update_one({"dept_no":103},{"$set":{"ename":"om"}})

Out[82]: <pymongo.results.UpdateResult at 0x288ff13eca0>

In [83]: #read()
         for x in client['siddhesh']['emp2'].find():
             print(x)

         {'_id': ObjectId('64a25d4f6d85ce57f133963a'), 'ename': 'siddhesh chavan', 'age': 22, 'language': ['Java', 'Python', 'C', 'C+
         +']}
```



```
{'_id': ObjectId('64a25d606d85ce57f133963b'), 'ename': 'satwik', 'job': 'Analyst', 'sal': 2500, 'dept_no': 101}
{'_id': ObjectId('64a25d606d85ce57f133963c'), 'ename': 'sid', 'job': 'Manager', 'sal': 4000, 'dept_no': 102}
{'_id': ObjectId('64a25d606d85ce57f133963d'), 'ename': 'om', 'job': 'Developer', 'sal': 35000, 'dept_no': 103}

#delete_many()
client['siddhesh']['emp2'].delete_many({})

<pymongo.results.DeleteResult at 0x288ff068d90>
```

15. Create a 5x4 numpy array and find it’s column-wise mean,max,min,sum

Create two 2-d Numpy Arrays (Matrix A, Matrix B) andperform the following operation on matrix

a. **Addition (A+B)**

b. **Multiplication (AxB)**

c. **Scalar Multiplication (A x integer or integer x A)**

d. **Transpose of Matrix**

3. **Write a Program to perform following using NumPy Arrays.**

a. **Create a 5-by-5 array of random integers between 0(inclusive) and 10 (exclusive)**

b. **Create a sequence of equally gapped 5 numbers in therange 0 to 100 (both inclusive)**

c. **Convert a 1-D array to a 3-D array**

d. **Convert all the elements of a numpy array from float tointeger datatype**

e. **Stack two numpy arrays horizontally and vertically**

From two numpy arrays, extract the indexes in which theelements in the two arrays match (hint- np.where(a == b))

Solution: Program

1.

```
import numpy as np
array = np.array([
    [2, 4, 6, 8],
    [10, 12, 14, 16],
    [18, 20, 22, 24],
    [26, 28, 30, 32],
    [34, 36, 38, 40]
])
mean = np.mean(array, axis=0)

# Calculate column-wise maximum
maximum = np.max(array, axis=0)

# Calculate column-wise minimum
minimum = np.min(array, axis=0)

# Calculate column-wise sum
```

```
sum_ = np.sum(array, axis=0)

print("Column-wise mean:")
print(mean)
print("\nColumn-wise maximum:")
print(maximum)
print("\nColumn-wise minimum:")
print(minimum)
print("\nColumn-wise sum:")
print(sum_)
```

2.

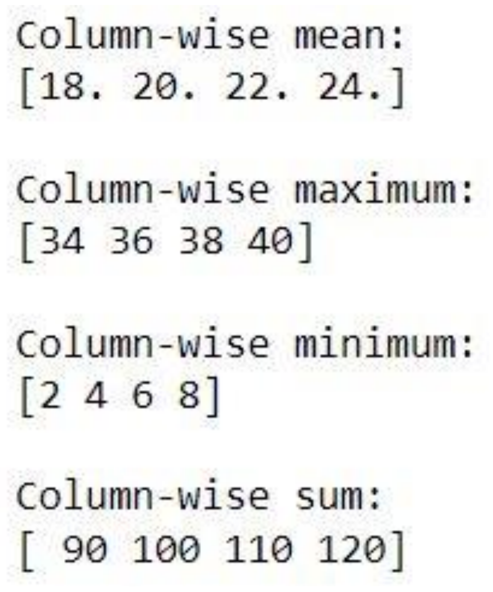
```
import numpy as np
a1=np.array([[[1,2,3],[8,5,7],[4,2,1]]])
a2=np.array([[[7,1,4],[4,2,8],[8,4,3]]])
#a.Adittion
print("Adittion is ",a1+a2)
#b.Multiplication
print("Multiplication is ",a1*a2)
#c.Scalar Multiplictaion
print("Scalar Multiplictaion is ",a1*5)
#a.Transpose
print("Transpose is ",a2.T)
```

3.

```
#a
arr1=np.random.randint(0,10,size=(5,5))
print(arr1)

#b
a=np.linspace(0,100,5)
print(a)

#c
a=np.array([1,2,3,4,5,6,7,8,9,10,11,12])
new=a.reshape(2,3,2)
print(new)

#d
float_array=np.array([1.44,2.33,3.55,4.99,6.21])
print("Float array is ")
print(float_array)
integer_array=float_array.astype(int)
print(integer_array)

#e
import numpy as np
a1=np.array([1,2,3])
```

```
a2=np.array([4,5,6])
print("first array is ",a1)
print("second array is ",a2)
hstacked_arr=np.hstack((a1,a2))
vstacked_arr=np.vstack((a1,a2))
print("horizontally stacked array is ",hstacked_arr)
print("vertically stacked array is ",vstacked_arr)

#f
import numpy as np
np.where([[True,False],[True,True]],[[1,2],[3,4]],[[5,6],[7,8]])
```

Output:
Screen Shot
1.



```
Column-wise mean:
[18. 20. 22. 24.]

Column-wise maximum:
[34 36 38 40]

Column-wise minimum:
[2 4 6 8]

Column-wise sum:
[ 90 100 110 120]
```

2.

```
Adittion is  [[[ 8  3  7]
  [12  7 15]
  [12  6  4]]]
Multiplication is  [[[ 7  2 12]
  [32 10 56]
  [32  8  3]]]
Scalar Multiplictaion is  [[[ 5 10 15]
  [40 25 35]
  [20 10  5]]]
Transpose is  [[[7]
  [4]
  [8]]

 [[1]
  [2]
  [4]]

 [[4]
  [8]
  [3]]]
```

3.

```
[[7 6 4 5 8]
 [7 6 8 6 1]
 [3 0 8 8 4]
 [9 5 1 2 1]
 [7 3 4 5 0]]
[  0.  25.  50.  75. 100.]
[[[ 1  2]
  [ 3  4]
  [ 5  6]]

 [[ 7  8]
  [ 9 10]
  [11 12]]]
Float array is
[1.44 2.33 3.55 4.99 6.21]
[1 2 3 4 6]
first array is  [1 2 3]
second array is  [4 5 6]
horizontally stacked array is  [1 2 3 4 5 6]
vertically stacked array is  [[1 2 3]
 [4 5 6]]
```

```
]: array([[1, 6],
          [3, 4]])
```

16.A] Use Automobile Dataset (Automobile_data.csv) and perform following operations for data analysis. This Dataset hasdifferent characteristics of an auto such as body-style, wheel- base, engine-type, price, mileage, horsepower, etc.

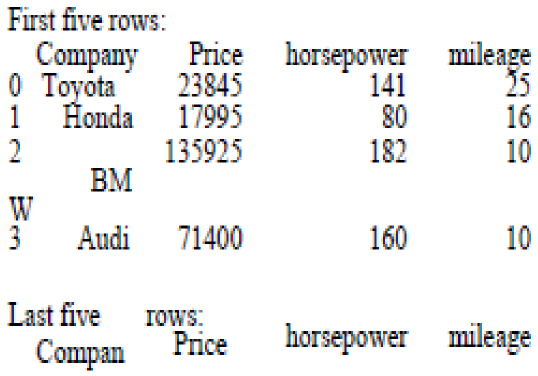
1.From the given dataset print the first and last five rows
2.Find the most expensive car company name
3.Print All Toyota Cars detailsCount total cars per company
4.Find each company's Highest price car
5.Find the average mileage of each car making company
6.Sort all cars by Price column
8. Apply the rank to the cars average-mileage. (highestaverage-mileage – rank 1, so on)
9.Concatenate two data frames using the following conditions
10.Create two data frames using the following two Dicts, Merge two data frames, and append the seconddata frame as a new column to the first data frame.
a. Car_Price = {'Company': ['Toyota', 'Honda', 'BMV','Audi'], 'Price': [23845, 17995, 135925 , 71400]}
b. Car_Horsepower = {'Company': ['Toyota', 'Honda','BMV', 'Audi'], 'horsepower': [141, 80, 182 , 160]}
Refer : https://www.kaggle.com/datasets/toramky/automobile-dataset

Solution: Program

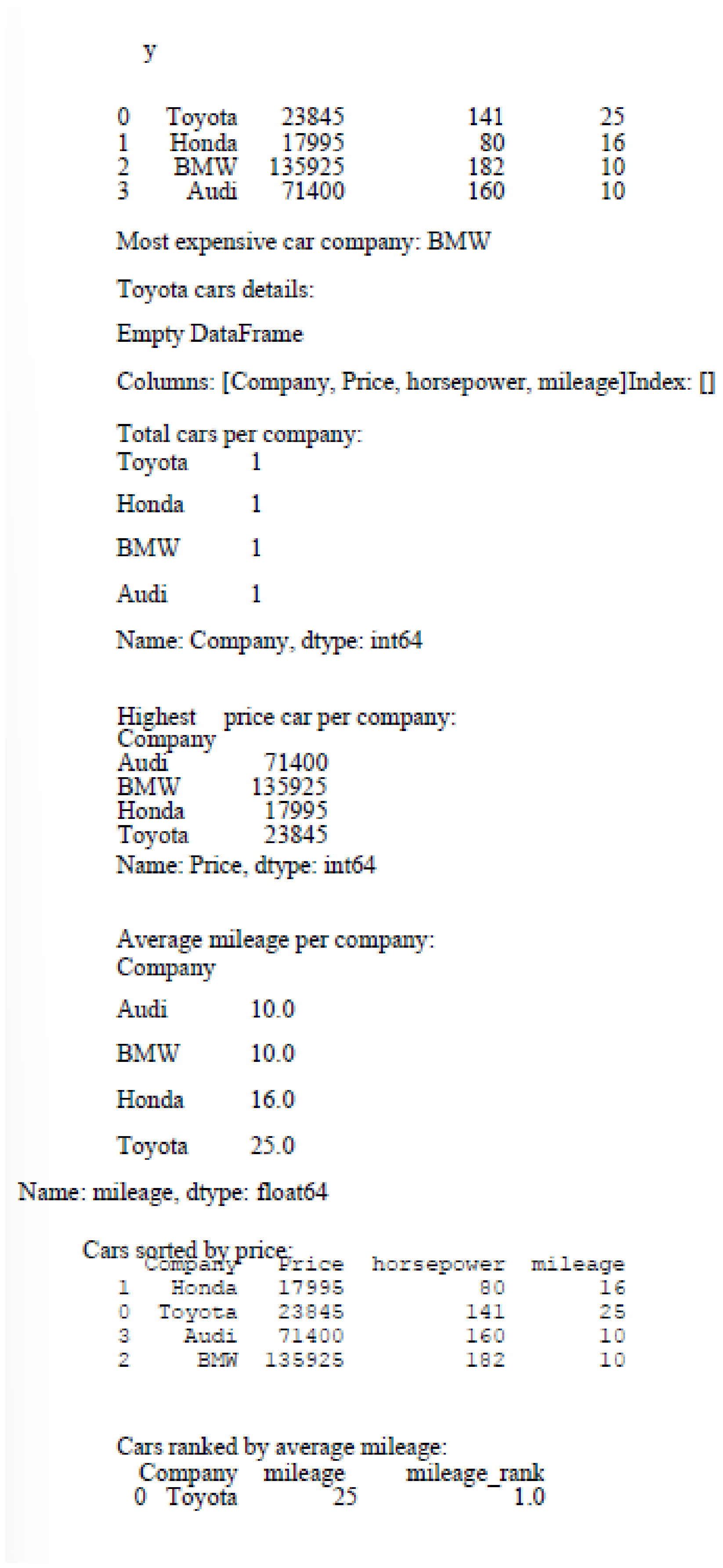
```
import pandas as pd
# Load the dataset
df = pd.read_csv('Automobile_data.csv')

# Print the first and last five rows
print("First five rows:")
print(df.head())
print("\nLast five rows:")
 print(df.tail())

# Find the most expensive car company name
most_expensive_company = df.loc[df['Price'].idxmax(), 'Company']
print("\nMost expensive car company:", most_expensive_company)

# Print all Toyota cars details
toyota_cars = df[df['Company'] == 'toyota']
print("\nToyota cars details:")
print(toyota_cars)

# Count total cars per company
total_cars_per_company = df['Company'].value_counts()
print("\nTotal cars per company:")
print(total_cars_per_company)

# Find each company's highest price car
```

```
highest_price_per_company = df.groupby('Company')['Price'].max()
print("\nHighest price car per company:")
print(highest_price_per_company)

# Find the average mileage of each car making company
average_mileage_per_company = df.groupby('Company')['mileage'].mean()
print("\nAverage mileage per company:")
print(average_mileage_per_company)

# Sort all cars by Price column
sorted_cars_by_price = df.sort_values('Price')
print("\nCars sorted by Price:")
print(sorted_cars_by_price)

# Apply rank to the cars' average mileage
df['mileage_rank'] = df['mileage'].rank(ascending=False)
 print("\nCars ranked by average mileage:")
print(df[['Company', 'mileage', 'mileage_rank']].sort_values('mileage_rank'))

# Concatenate two data frames based on conditions
car_price = {'Company': ['Toyota', 'Honda', 'BMW', 'Audi'], 'Price': [23845, 17995, 135925, 71400]}
car_horsepower = {'Company': ['Toyota', 'Honda', 'BMW', 'Audi'], 'horsepower': [141, 80, 182, 160]}
df1 = pd.DataFrame(car_price)
df2 = pd.DataFrame(car_horsepower)

concatenated_df = pd.concat([df1, df2['horsepower']], axis=1)
print("\nConcatenated data frames:")
print(concatenated_df)

# Merge two data frames and append the second data frame as a new column to the first data frame
car_price_df = pd.DataFrame(car_price)
car_horsepower_df = pd.DataFrame(car_horsepower)
merged_df = pd.merge(car_price_df, car_horsepower_df, on='Company')
print("\nMerged data frames:")
print(merged_df)
```

Output:
Screen Shot



```
First five rows:
   Company    Price   horsepower   mileage
0  Toyota     23845          141        25
1    Honda    17995           80        16
2            135925          182        10
       BM
W
3     Audi    71400          160        10

Last five   rows:
   Compan    Price    horsepower   mileage
```



```
    y

0    Toyota    23845         141         25
1     Honda    17995          80         16
2      BMW    135925         182         10
3      Audi    71400         160         10

Most expensive car company: BMW

Toyota cars details:

Empty DataFrame

Columns: [Company, Price, horsepower, mileage]Index: []

Total cars per company:
Toyota      1

Honda       1

BMW         1

Audi        1

Name: Company, dtype: int64


Highest   price car per company:
Company
Audi          71400
BMW          135925
Honda         17995
Toyota        23845
Name: Price, dtype: int64


Average mileage per company:
Company

Audi         10.0

BMW          10.0

Honda        16.0

Toyota       25.0

Name: mileage, dtype: float64

Cars sorted by price:
  Company   Price  horsepower  mileage
1   Honda   17995          80       16
0  Toyota   23845         141       25
3    Audi   71400         160       10
2     BMW  135925         182       10


Cars ranked by average mileage:
  Company  mileage      mileage_rank
0  Toyota       25               1.0
```

```
1     Honda          16              2.0
           2         10              3.5
         BM
          W
3       Audi         10              3.5

Concatenated data frames:
   Company     Price      horsepower
   0  Toyota   23845             141
1     Honda    17995              80
           2  135925             182
         BM
          W
3       Audi   71400             160

Merged data frames:
    Company    Price   horsepower
   0  Toyota   23845             141
1     Honda    17995              80
           2  135925             182
         BM
          W
3       Audi   71400             160
```

17. B] Use company_sales_data.csv for this exercise. Read thisfile using Pandas or NumPy or using in-built matplotlib function.

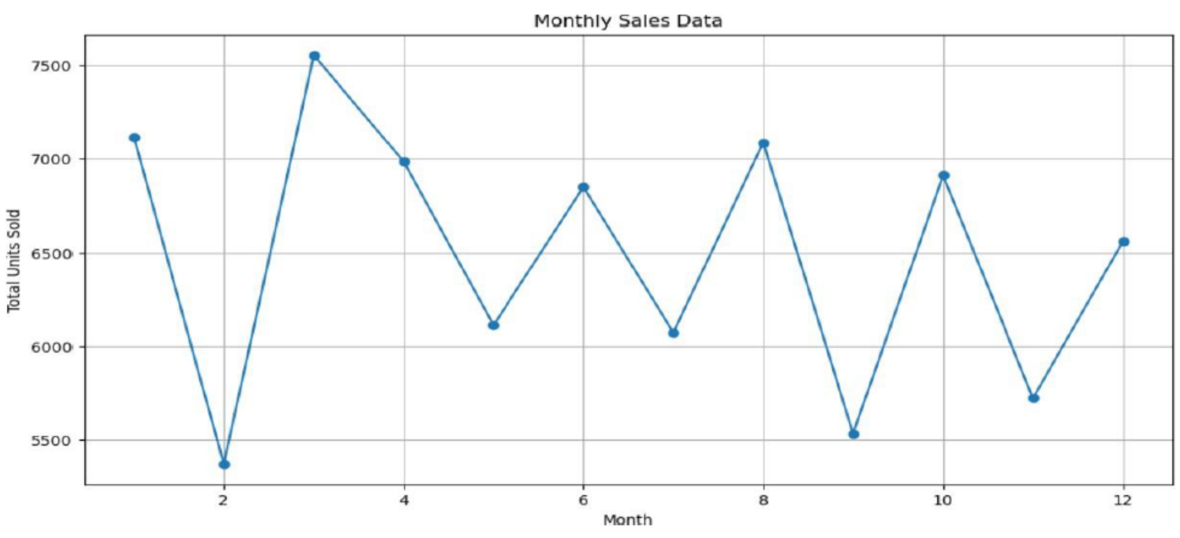
Read all product sales data and show it using a multiline plot. Display the number of units sold per month for each product using multiline plots. (i.e., Separate Plotline for each product).

Solution: Program

```
import pandas as pd
import matplotlib.pyplot as plt
# Read the CSV file
df = pd.read_csv('company_sales_data.csv')
# Display the data
print(df.head())
# Plotting the data
plt.figure(figsize=(12, 6))
plt.plot(df['month_number'], df['total_units'], marker='o')
plt.xlabel('Month')
plt.ylabel('Total Units Sold')
plt.title('Monthly Sales Data')
plt.grid(True)
plt.show()
```

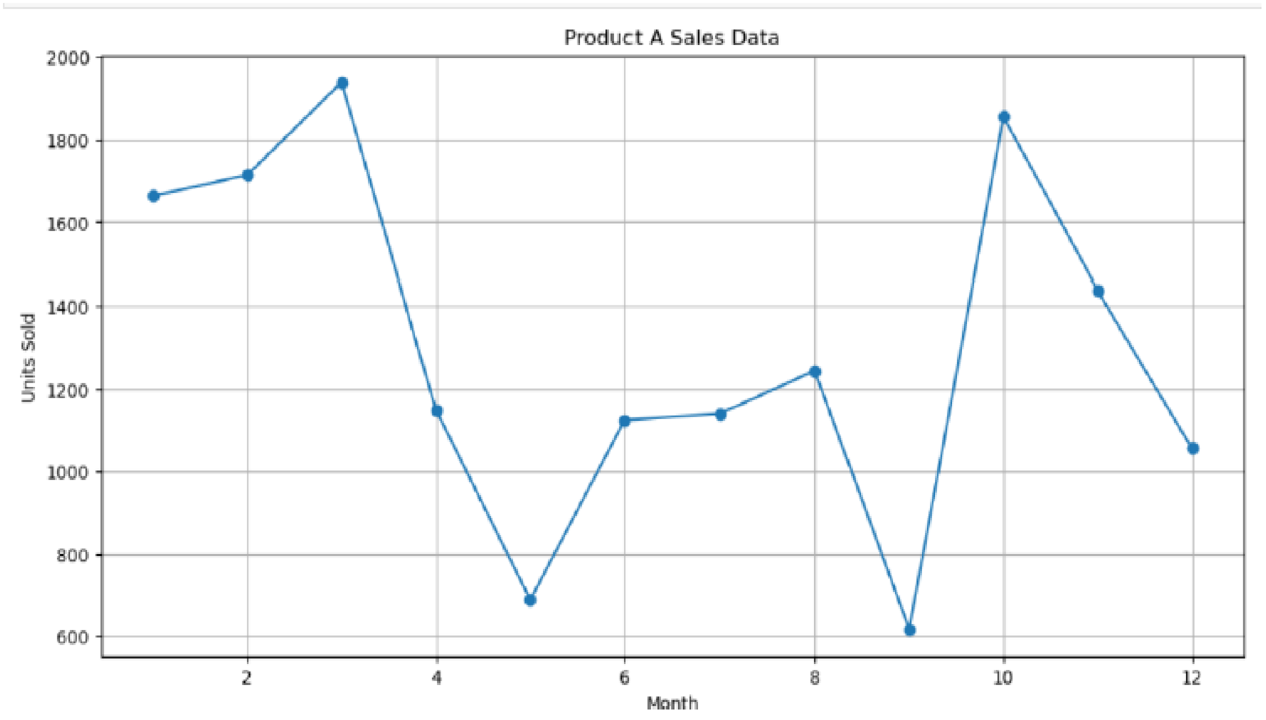
Output:
Screen Shot

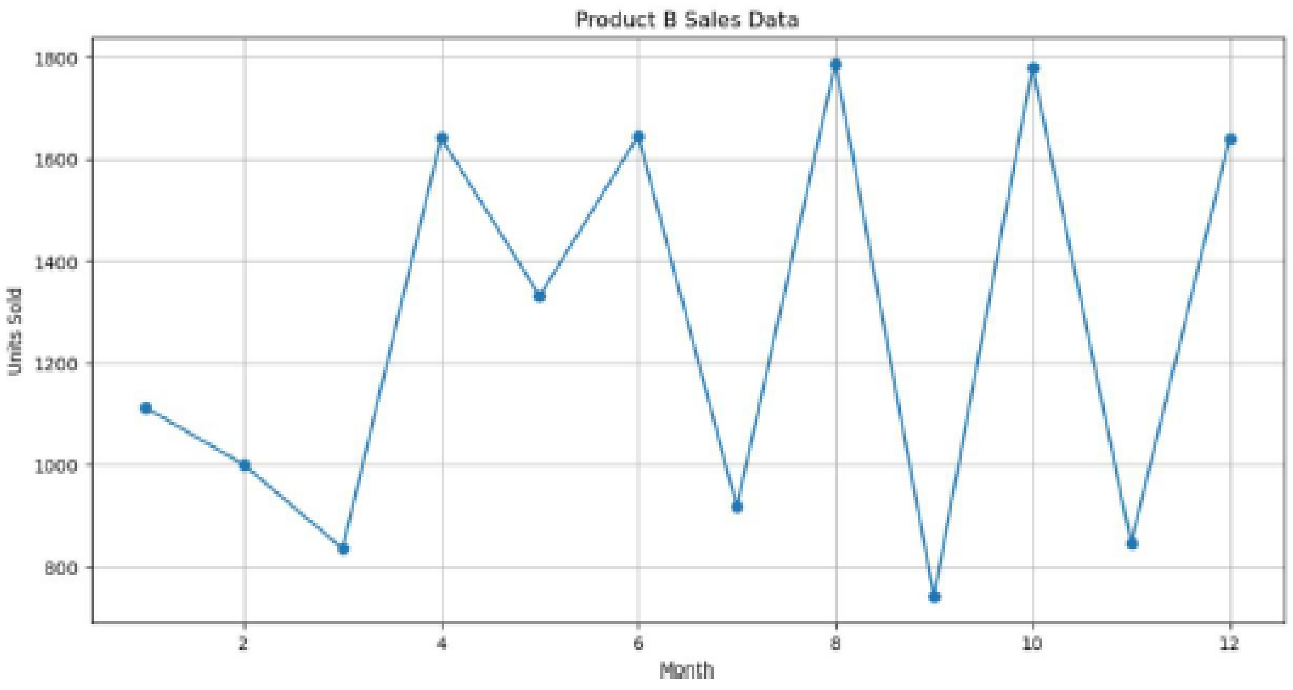
```
Product A   Product B   Product C   Product D   Product  E  total_units
1663        1113        1845        1793         701          7115
1714        1000        1157         530         970          5371

 1939         836        1855       1555        1369           7554
 1148        1641        1476        729        1992           6986
  688        1331        1716       1081        1295           6111
```

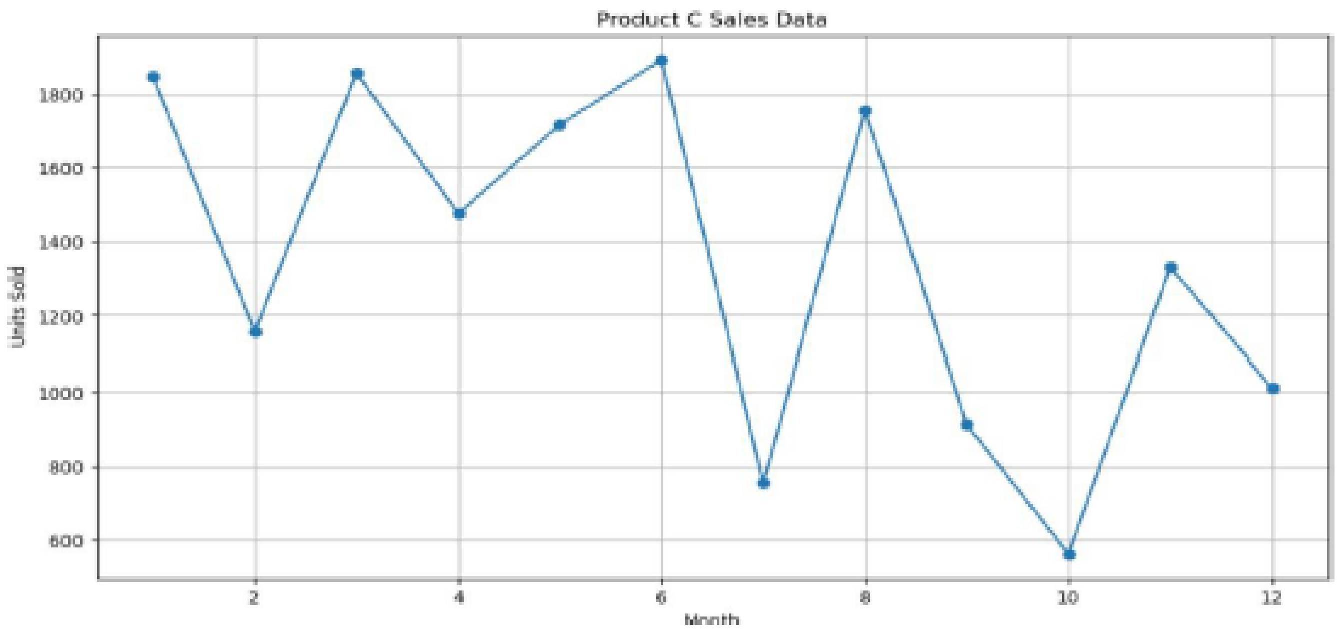


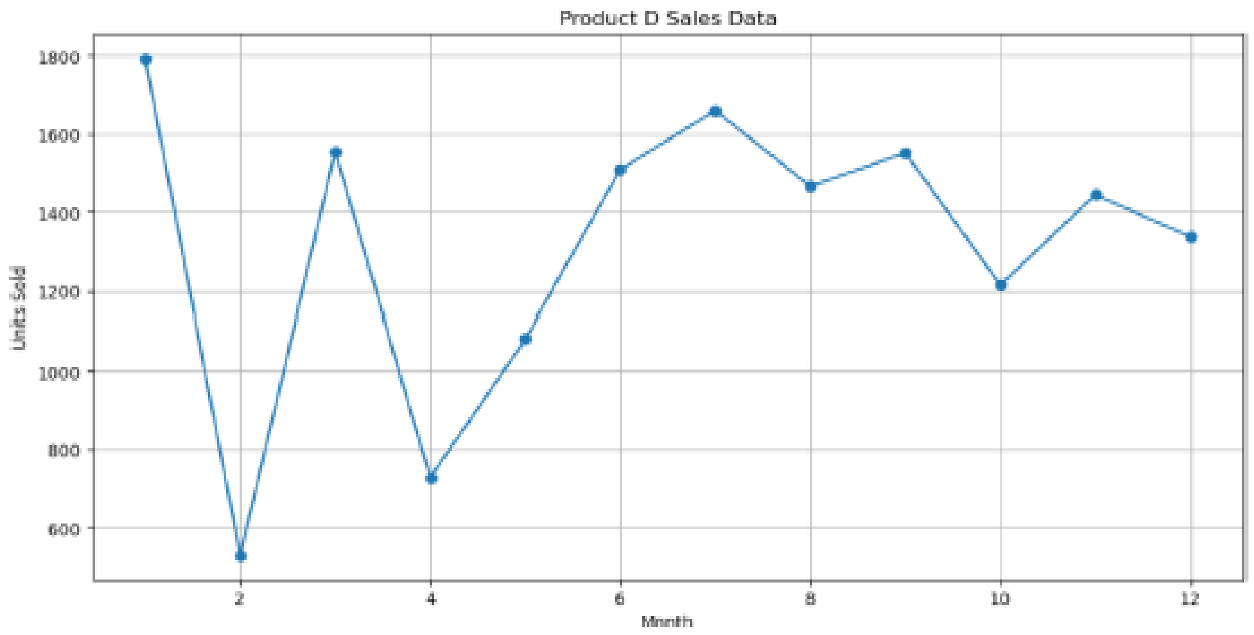
Display the number of units sold per month for each product using multiline plots. (i.e., Separate Plotline for each product).

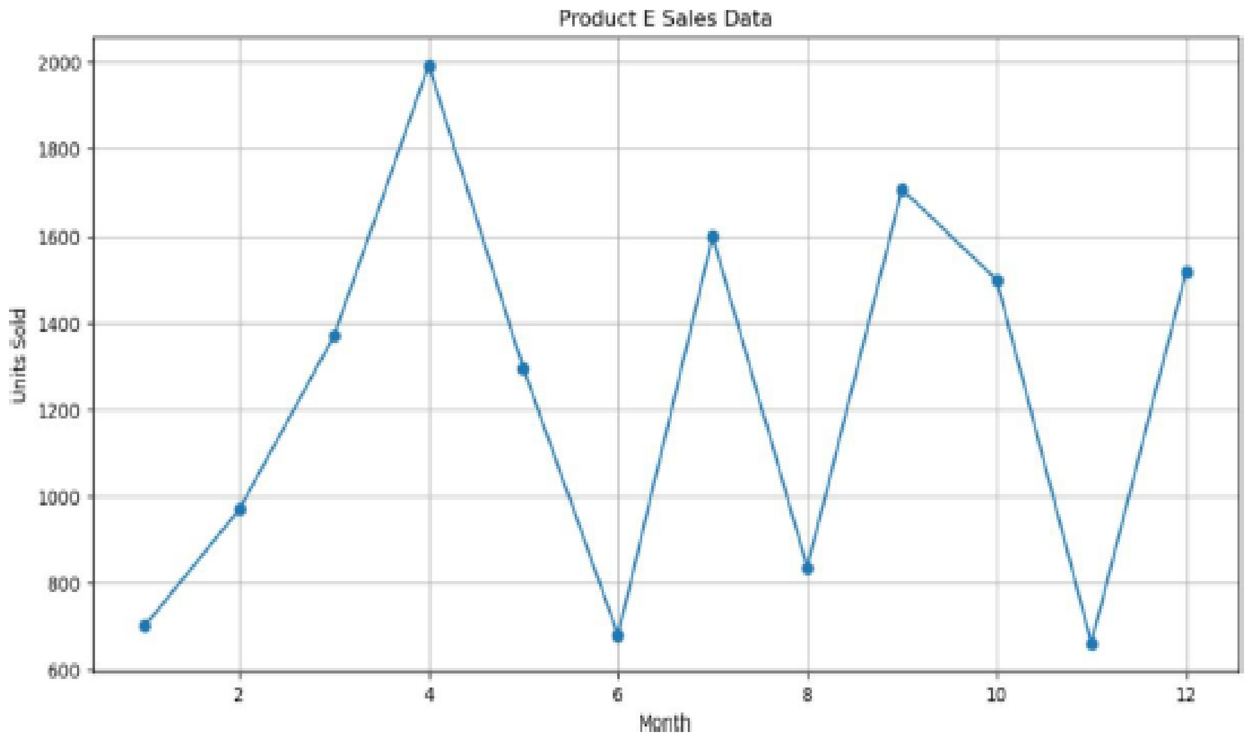
```
import pandas as pd
import matplotlib.pyplot as plt
# Read the dataset
df = pd.read_csv('company_sales_data.csv')
# Get the product names
products = df.columns[:-3]
# Create separate multiline plots for each
for product in products:
    plt.figure(figsize=(12, 6))
    plt.plot(df['month_number'], df[product],
        marker='o')
    plt.xlabel('Month')
    plt.ylabel('Units Sold')
    plt.title(f'{product} Sales Data')
    plt.grid(True)
    plt.show()
```




Product B Sales Data
Units Sold
Month
1800
1600
1400
1200
1000
800
2
4
6
8
10
12


Product C Sales Data
Units Sold
Month
1800
1600
1400
1200
1000
800
600
2
4
6
8
10
12


Product D Sales Data
Units Sold
Month
1800
1600
1400
1200
1000
800
600
2
4
6
8
10
12


Product E Sales Data
Units Sold
Month
2000
1800
1600
1400
1200
1000
800
600
2
4
6
8
10
12