Python Slips for Lab Exam-2022-23

1. Write a python code to create two sets. Find out the Union, intersection, difference and symmetric difference between these two sets.

s1={10,20,30,40,50}

s2={11,20,22,30,33}

print(s1.union(s2))

print(s1.intersection(s2))

print(s1.difference(s2))

print(s1.symmetric\_difference(s2))

---------------------------------------------------------------------------------------------------------

1. Write a Python program to sort (ascending and descending) a dictionary by key and then by value.

dict1={"India":"Asia","France":"Europe","Libya":"Africa","Japan":"Asia","UK":"Europe",

"Chad":"Africa"}

print(dict(sorted(dict1.items())))

---------------------------------------------------------------------------------------------------------

1. Write a program to count the numbers of characters in the given string and store them in a dictionary data structure

def cnt(d, s):

for i in s:

d.update({i:s.count(i)})

return d

s="Object Oriented"

d={}

print(cnt(d,s))

---------------------------------------------------------------------------------------------------------

1. With a given integral number n, write a program to generate a dictionary that contains (i, i\*i) such that is an integral number between 1 and n (both included) and then the program should print the dictionary.  
   Suppose the following input is supplied to the program: 8  
   Then, the output should be: {1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64}

d={}

for i in range(int(input("Enter Total numbers:"))):

n = int(input("Enter Number:"))

d.update({n:n\*n})

print(d)

---------------------------------------------------------------------------------------------------------

1. Write a Python program to find a pair of elements (indices of the two numbers) from a given array whose sum equals a specific target number.  
   Input: numbers= [10,20,10,40,50,60,70], target=50  
   Output: 3, 4

def sum(l,t):

for i in range(len(l)):

for j in range(i+1,len(l)):

if l[i] + l[j] ==t:

return i,j

l=[10,20,10,40,50,60,70]

t=50

print(sum(l,t))

---------------------------------------------------------------------------------------------------------

1. A Government wants to provide student loans to students in their country. But in order for a student to be eligible to get a loan, he/She must be in the age range 17 to 21, and must have a minimum of 80% score in academics. Write a program to accept a name, age, and marks of student and display if he/she is eligible for the loan or not.

name=input("Enter name:")

age=int(input("Enter age:"))

marks=int(input("Enter marks:"))

if age >=17 and age <21 and marks >=80:

print("You are eligible for the loan")

else:

print("You are not eligible for the loan")

---------------------------------------------------------------------------------------------------------

1. Create class called 'library' with data attributes like Acc-Number, publisher, title and Author, The methods of class should include  
   i) Read()-Acc-Number,title,Author,publisher  
   ii)Compute()-to accept the number of day late, calculate and display the fine fine charged at the rate of Rupees 5/- per day  
   iii)Display the data

class Library:

def read(self,acc\_number,title,author,publisher):

self.acc\_number = acc\_number

self.title = title

self.author = author

self.publisher = publisher

def compute(self):

late=int(input("Enter number of late days: "))

return late\*5

def display(self):

print("Acc\_number: ",self.acc\_number,"\nTitle: "+self.title+"\nAuthor: "

+self.author+"\nPublisher: "+self.publisher+"\nFine: ",self.compute())

library = Library()

library.read(123423,"Head-First-Java","Bert and Kathy","Head-First")

library.display()

---------------------------------------------------------------------------------------------------------

1. Write a class called “Investment” with fields called ‘principal’ and ‘interest’. The constructor should set the values of those fields. There should be a method called value\_after that returns the value of the investment after n years. The formula for this is p(1 + i) n , where p is the principal, and i is the interest rate.

class Investment:

def \_\_init\_\_(self,principal,interest):

self.principal = principal

self.interest = interest

def value\_after(self,n):

return self.principal\*(1+self.interest)\*n

investment = Investment(100,7)

n=int(input("Enter number of years:"))

print("Interest after ",n," year's: ",investment.value\_after(n))

---------------------------------------------------------------------------------------------------------

1. Define a class which has at least two methods:  
   getString: to get a string from console input  
   printString: to print the string in upper case.

class Str:

def getString(self):

return input("Enter String: ")

def printString(self,str):

print("In UpperCase: ",str.upper())

s = Str()

val = s.getString()

s.printString(val)

---------------------------------------------------------------------------------------------------------

1. Write a Python class named Circle constructed by a radius and two methods which will compute the area and the perimeter of a circle.

class Circle:

def \_\_init\_\_(self,radius):

self.radius = radius

def area(self):

return 3.14\*self.radius\*self.radius

def parameter(self):

return 2\*3.14\*self.radius

circle = Circle(int(input("Enter radius: ")))

area = circle.area()

print("Area of circle: ",area)

parameter =circle.parameter()

print("Parameter of circle: %.2f"%parameter)

---------------------------------------------------------------------------------------------------------

1. Define a class named Shape and its subclass Square. The Square class has an init function which takes a length as argument. Both classes have a area function which can print the area of the shape where Shape's area is 0 by default.

class Shape:

def area(self):

return 0

class Square(Shape):

def \_\_init\_\_(self,length):

self.length = length

def area(self):

return self.length\*self.length

length = int(input("Enter length: "))

square = Square(length)

print("Area of Square: ",square.area())

square = Shape()

print("Area of Shape: ",square.area())

---------------------------------------------------------------------------------------------------------

1. Write a Python program to check the validity of a password (input from users).

Validation :

At least 1 letter between [a-z] and 1 letter between [A-Z].

At least 1 number between [0-9].

At least 1 character from [$#@].

Minimum length 6 characters.

Maximum length 16 characters.

import re

def check\_password(password):

if not len(password)>=6 & len(password)<=16:

return False

elif not re.compile(r"[a-z]+").search(password):

return False

elif not re.compile(r"[A-Z]+").search(password):

return False

elif not re.compile(r"[0-9]+").search(password):

return False

elif not re.compile(r"[$#@]+").search(password):

return False

return True

password = input("Enter Password: ")

if check\_password(password):

print("Password is valid")

else:

print("Password is not valid")

---------------------------------------------------------------------------------------------------------

1. Write a Python code to validate Email ID.

import re

email = input("Enter email:")

pattern = re.compile(r"[a-zA-Z0-9\_.+-]+@[a-zA-Z]+\.[a-zA-Z]+")

match = bool(pattern.search(email))

if match:

print("Email is valid!!!")

else:

print("Email is invalid!!!")

---------------------------------------------------------------------------------------------------------

1. Write a Regular Expression to represent all 10 digit mobile numbers.   
   Rules:   
   1. Every number should contains exactly 10 digits.   
   2. The first digit should be 7 or 8 or 9   
   Write a Python Program to check whether the given number is valid mobile number or not?

import re

def check\_phno(phno):

if len(phno)!=10:

return False

if not bool(re.compile(r"^(7|8|9)").match(phno)):

return False

return True

phno = input("Enter Phone Number:")

if check\_phno(phno):

print("Valid Phone Number")

else:

print("Invalid Phone Number")

---------------------------------------------------------------------------------------------------------

1. Write a program to raise an exception if the user enters name of an employee which is not in predifined list of Employee (note: create a list of employee)

class InvalidEmployeeException(Exception):

pass

employees=["Nikhil","Om","Raghav","Arav","Jon"]

employee = input("Enter name of employee:")

try:

if employee in employees:

print("Valid employee name!!!")

else:

raise InvalidEmployeeException

except:

print("InvalidEmployeeException: Invalid employee name!!!")

---------------------------------------------------------------------------------------------------------

1. Write a programm for synchronization of Threads using RLOCK. Accept the two numbers from user and calculate factorial of both Numbers simultaneously

import threading

def fact(number, lock):

result = 1

for i in range(1, number + 1):

result \*= i

with lock:

print("Factorial of",number,"is:",result)

num1 = int(input("Enter first number: "))

num2 = int(input("Enter second number: "))

lock = threading.RLock()

thread1 = threading.Thread(target=fact, args=(num1, lock))

thread2 = threading.Thread(target=fact, args=(num2, lock))

thread1.start()

thread2.start()

thread1.join()

thread2.join()

---------------------------------------------------------------------------------------------------------

1. Write a multithreading program where one thread prints square of a number and another thread prints factorial of a number. Also display the total time taken for the execution

import threading

import time

def square(number):

result = number \* number

print("Square of",number,"is:",result)

def factorial(number):

result = 1

for i in range(1, number + 1):

result \*= i

print("Factorial of",number,"is:",result)

number = int(input("Enter a number: "))

thread1 = threading.Thread(target=square, args=(number,))

thread2 = threading.Thread(target=factorial, args=(number,))

start\_time = time.time()

thread1.start()

thread2.start()

thread1.join()

thread2.join()

end\_time = time.time()

total\_time = end\_time - start\_time

print(f"Total time taken:",total\_time,"seconds")

---------------------------------------------------------------------------------------------------------

1. Write a python program to check the given number is prime or not. Handle Suitable Exception

def is\_prime(num):

for i in range(2,num):

if num % i == 0:

return False

break

return True

try:

num = int(input("Enter Number:"))

if is\_prime(num):

print(num,"is a prime number")

else:

print(num,"is not a prime number")

except ValueError as ve:

print("Invalid Input!!!")

---------------------------------------------------------------------------------------------------------

1. Develop a program to Print the number of line,words,and characters present in the given file. Accept the file name from user. Handle necessary exceptions

file = input("Enter File Name:")

try:

with open(file,'r') as f:

lines = 0

words = 0

char = 0

for data in f:

lines = lines + 1

char = char + len(data)

words = words + len(data.split())

print("No of lines:",lines,"\nNo of characters:",char,"\nNo of words:",words)

except FileNotFoundError as fnfe:

print("FileNotFoundError:",file,"not found!!!")

---------------------------------------------------------------------------------------------------------

1. Python Program to Append the Contents of One File to Another File

file1='data.txt'

file2='demo.txt'

fa= open(file2,'a')

with open(file1,'r') as fr:

for data in fr:

fa.write(data)

print("Contents of",file1,"is appended to",file2)

---------------------------------------------------------------------------------------------------------

1. Develop a program to remove the comment character from all the lines in a given file. Accept file name from user

file = input("Enter file name:")

with open(file,'r') as f:

cont =""

for data in f:

if '#' in data:

for char in data:

if char !='#':

cont=cont+char

else:

cont = cont+data

with open(file,'w') as f:

f.write(cont)

print("Comment character's removed from",file)

---------------------------------------------------------------------------------------------------------

1. Write a python program to read "employee.txt" file and display the alternate employee records.

name = input("Enter employee name: ")

address = input("Enter employee address: ")

with open('employee.txt','a') as f:

data = "Name: "+name+"\tAddress: "+address+"\n"

f.write(data)

print("Name: "+name+" Address: "+address+" saved in employee.txt")

---------------------------------------------------------------------------------------------------------

1. Write a Python program to combine each line from first file with the corresponding line in second file.

with open('file1.txt','r') as f1, open('file2.txt','r') as f2:

for data in f1:

print(data[0:len(data)-1]+" "+f2.readline())

---------------------------------------------------------------------------------------------------------

1. Write a Python program to perform following operation on MongoDB database  
   i) Create collection "Book" with fields  
   Book-name, Book-code, Book-Author, Book-Price,Book-publication-year  
   ii) Insert 5 documents  
   iii)Find the books whose price between 500-800.  
   iv) Update price of book "Python programming" as 1000

from pymongo import MongoClient

client = MongoClient('mongodb://localhost:27017/')

db = client['mydatabase']

collection = db['Book']

book\_fields = {

"Book-name": "",

"Book-code": "",

"Book-Author": "",

"Book-Price": 0,

"Book-publication-year": ""

}

collection.insert\_one(book\_fields)

books = [

{

"Book-name": "AIT",

"Book-code": "B001",

"Book-Author": "Author 1",

"Book-Price": 900,

"Book-publication-year": "2020"

},

{

"Book-name": "ADBMS",

"Book-code": "B002",

"Book-Author": "Author 2",

"Book-Price": 400,

"Book-publication-year": "2018"

},

{

"Book-name": "OT",

"Book-code": "B003",

"Book-Author": "Author 3",

"Book-Price": 900,

"Book-publication-year": "2022"

},

{

"Book-name": "SPM",

"Book-code": "B004",

"Book-Author": "Author 4",

"Book-Price": 600,

"Book-publication-year": "2019"

},

{

"Book-name": "Python programming",

"Book-code": "B005",

"Book-Author": "Author 5",

"Book-Price": 750,

"Book-publication-year": "2021"

}

]

collection.insert\_many(books)

query = {"Book-Price": {"$gte": 500, "$lte": 800}}

projection = {"\_id": 0} # Exclude the "\_id" field

result = collection.find(query, projection)

print("Books with price between 500 and 800:")

for book in result:

print(book)

filter\_query = {"Book-name": "Python programming"}

update\_query = {"$set": {"Book-Price": 1000}}

collection.update\_one(filter\_query, update\_query)

projection = {"\_id": 0} # Exclude the "\_id" field

result = collection.find({}, projection).sort("Book-publication-year")

print("Books in the order of publication year:")

for book in result:

print(book)

---------------------------------------------------------------------------------------------------------

1. Write a python program to connect with MongoDB Database. Create collection “restaurants” (Assume suitable structure). Query the “restaurants” collection to  
   a. display the fields restaurant\_id, name, establishment\_year and cuisine for all the documents in the collection restaurant  
   b. find the restaurants who achieved a score more than 90.  
   c. arrange the name of the restaurants in ascending order  
   d. Update restaurant score for the establishment year 2019

from pymongo import MongoClient

client = MongoClient('mongodb://localhost:27017/')

db = client['mydatabase']

collection = db['restaurants']

restaurants\_data = [

{

"restaurant\_id": 1,

"name": "Navmi",

"establishment\_year": 2010,

"cuisine": "Italian",

"score": 85

},

{

"restaurant\_id": 2,

"name": "Kesar",

"establishment\_year": 2012,

"cuisine": "Chinese",

"score": 95

},

{

"restaurant\_id": 3,

"name": "Amarai",

"establishment\_year": 2015,

"cuisine": "Mexican",

"score": 92

},

{

"restaurant\_id": 4,

"name": "Orchid",

"establishment\_year": 2019,

"cuisine": "Indian",

"score": 88

},

{

"restaurant\_id": 5,

"name": "Emrald",

"establishment\_year": 2021,

"cuisine": "Japanese",

"score": 91

}

]

collection.insert\_many(restaurants\_data)

projection = {"\_id": 0} # Exclude the "\_id" field

result = collection.find({}, projection)

print("All documents in the 'restaurants' collection:")

for doc in result:

print(doc)

projection = {"\_id": 0, "restaurant\_id": 1, "name": 1, "establishment\_year": 1, "cuisine": 1}

result = collection.find({}, projection)

print("\nFields restaurant\_id, name, establishment\_year, and cuisine for all documents:")

for doc in result:

print(doc)

query = {"score": {"$gt": 90}}

result = collection.find(query, projection)

print("\nRestaurants with a score more than 90:")

for doc in result:

print(doc)

result = collection.find({}, projection).sort("name", 1)

print("\nRestaurants sorted by name in ascending order:")

for doc in result:

print(doc)

filter\_query = {"establishment\_year": 2019}

update\_query = {"$set": {"score": 95}}

collection.update\_many(filter\_query, update\_query)

print("\nUpdated restaurant score for the establishment year 2019.")

projection = {"\_id": 0, "restaurant\_id": 1, "name": 1, "establishment\_year": 1, "cuisine": 1, "score": 1}

result = collection.find(filter\_query, projection)

print("Restaurants with the establishment year 2019 and their updated scores:")

for doc in result:

print(doc)

---------------------------------------------------------------------------------------------------------

1. Write a python program to connect with MongoDB Database. Create collection “movies” (title, writer, year, actors, director...consider this fields in movies collection). Query the “movies” collection to  
   a. get all documents where actors include "Amitabh Bachchan"  
   b. get all movies released before the year 2000 or after 2010  
   c. Update some documents by adding some extra fields  
   d. Delete movie “movie\_name”

from pymongo import MongoClient

client = MongoClient('mongodb://localhost:27017/')

db = client['mydatabase']

collection = db['movies']

movies\_data = [

{

"title": "Coolie No. 1",

"writer": "David Dhawan",

"year": 1995,

"actors": ["Govinda", "Karishma Kapoor"],

"director": "David Dhawan"

},

{

"title": "Mela",

"writer": "Dharmesh Darshan",

"year": 2000,

"actors": ["Aamir Khan", "Twinkle Khanna"],

"director": "Raj Kapoor"

},

{

"title": "Jalwa",

"writer": "Pankaj Parashar",

"year": 1987,

"actors": ["Amitabh Bachchan", "Naseeruddin Shah"],

"director": "Pankaj Parashar"

},

{

"title": "Piku",

"writer": "Shoojit Sircar",

"year": 2015,

"actors": ["Amitabh Bachchan", "Irfan Khan"],

"director": "Shoojit Sircar"

},

{

"title": "Sanak",

"writer": "Kanishk Varma",

"year": 2022,

"actors": ["Vidyut Jammwal", "Rukmini Maitra"],

"director": "Raj Kapoor"

}

]

collection.insert\_many(movies\_data)

projection = {"\_id": 0} # Exclude the "\_id" field

result = collection.find({}, projection)

print("All documents in the 'movies' collection:")

for doc in result:

print(doc)

query = {"director": "Raj Kapoor"}

result = collection.find(query, projection)

print("\nDocuments with director set to 'Raj Kapoor':")

for doc in result:

print(doc)

query = {"actors": "Amitabh Bachchan"}

result = collection.find(query, projection)

print("\nDocuments where actors include 'Amitabh Bachchan':")

for doc in result:

print(doc)

query = {"year": {"$gte": 1990, "$lte": 1999}}

result = collection.find(query, projection)

print("\nMovies released in the 90s:")

for doc in result:

print(doc)

query = {"$or": [{"year": {"$lt": 2000}}, {"year": {"$gt": 2010}}]}

result = collection.find(query, projection)

print("\nMovies released before the year 2000 or after 2010:")

for doc in result:

print(doc)

filter\_query = {"title": "Saaho"}

update\_query = {"$set": {"genre": "Drama", "rating": 4.5}}

collection.update\_many(filter\_query, update\_query)

delete\_query = {"title": "Sanak"}

collection.delete\_one(delete\_query)

print("\nUpdated documents with extra fields and deleted movie 'Movie Sanak'.")

---------------------------------------------------------------------------------------------------------

1. Write a python program using numpy to do the following operations  
   i) create aaray which contains multiples of 2 till 200  
   ii)Find the number of occurrences of a sequence in a NumPy array  
   iii)Count the number of elements along a given axis

import numpy as np

array\_multiples\_of\_2 = np.arange(2, 201, 2)

print("Array of multiples of 2 till 200:")

print(array\_multiples\_of\_2)

sequence = np.array([2, 4, 6])

occurrences = np.count\_nonzero(np.correlate(array\_multiples\_of\_2, sequence, mode='valid'))

print("\nNumber of occurrences of the sequence [2, 4, 6]:", occurrences)

matrix = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

elements\_count = np.size(matrix, axis=None)

print("\nNumber of elements in the matrix:", elements\_count)

---------------------------------------------------------------------------------------------------------

1. Write a python program using numpy to do the following operations  
   i) create a 2\*2 matrix and find the transpose of it  
   ii) create two matrices and find the multiplication of matrices.

import numpy as np

matrix\_2x2 = np.array([[1, 2], [3, 4]])

transpose\_matrix = np.transpose(matrix\_2x2)

print("\nOriginal matrix:")

print(matrix\_2x2)

print("\nTranspose of the matrix:")

print(transpose\_matrix)

matrix\_A = np.array([[1, 2], [3, 4]])

matrix\_B = np.array([[5, 6], [7, 8]])

matrix\_product = np.dot(matrix\_A, matrix\_B)

print("\nMatrix A:")

print(matrix\_A)

print("\nMatrix B:")

print(matrix\_B)

print("\nMatrix product of A and B:")

print(matrix\_product)

---------------------------------------------------------------------------------------------------------

1. Write a python program to do the following operations on Numpy Array  
   i) Find the k smallest values of a NumPy  
   ii) Compute the nth percentile of the NumPy array  
   iii) Calculate the sum of all columns in a 2D NumPy array  
   iv)Calculate average values of two given NumPy arrays

import numpy as np

arr = np.array([9, 3, 5, 1, 8, 2, 6, 4, 7])

k = 3

k\_smallest\_values = np.partition(arr, k-1)[:k]

print("The", k, "smallest values:", k\_smallest\_values)

arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9])

n = 75

nth\_percentile = np.percentile(arr, n)

print("The", n, "percentile:", nth\_percentile)

matrix = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

column\_sums = np.sum(matrix, axis=0)

print("Sum of columns:", column\_sums)

arr1 = np.array([1, 2, 3, 4, 5])

arr2 = np.array([6, 7, 8, 9, 10])

average\_arr1 = np.average(arr1)

average\_arr2 = np.average(arr2)

print("Average of arr1:", average\_arr1)

print("Average of arr2:", average\_arr2)

---------------------------------------------------------------------------------------------------------

1. Write a python program to do the following operations on Numpy Array  
   i )Create a 5X2 integer array from a range between 100 to 200 such that the difference between each element is 10  
   ii) Return array of odd rows and even columns from below numpy array  
   sampleArray = numpy.array([[3 ,6, 9, 12], [15 ,18, 21, 24],   
   [27 ,30, 33, 36], [39 ,42, 45, 48], [51 ,54, 57, 60]])

import numpy as np

array\_5x2 = np.arange(100, 200, 10).reshape(5, 2)

print("5x2 array:")

print(array\_5x2)

sample\_array = np.array([[3, 6, 9, 12], [15, 18, 21, 24], [27, 30, 33, 36], [39, 42, 45, 48], [51, 54, 57, 60]])

odd\_rows\_even\_cols = sample\_array[::2, 1::2]

print("\nArray of odd rows and even columns:")

print(odd\_rows\_even\_cols)

---------------------------------------------------------------------------------------------------------

1. Write a Python program to get the number of days of a given month and year

import numpy as np

def get\_number\_of\_days(year, month):

start\_date = np.datetime64(f'{year}-{month:02d}')

end\_date = start\_date + np.timedelta64(1, 'M') - np.timedelta64(1, 'D')

return int((end\_date - start\_date) / np.timedelta64(1, 'D')) + 1

year = int(input("Enter the year: "))

month = int(input("Enter the month (1-12): "))

number\_of\_days = get\_number\_of\_days(year, month)

print("Number of days in the given month and year:", number\_of\_days)

---------------------------------------------------------------------------------------------------------

1. Write a Python program to convert two date difference in days, hours, minutes, seconds

import numpy as np

def get\_time\_difference(start\_date, end\_date):

time\_difference = np.timedelta64(end\_date - start\_date, 's')

days = time\_difference.astype('timedelta64[D]')

hours = (time\_difference.astype('timedelta64[h]') % np.timedelta64(1, 'D')).astype(int)

minutes = (time\_difference.astype('timedelta64[m]') % np.timedelta64(1, 'h')).astype(int)

seconds = (time\_difference.astype('timedelta64[s]') % np.timedelta64(1, 'm')).astype(int)

return days, hours, minutes, seconds

start\_date = np.datetime64('2023-06-01')

end\_date = np.datetime64('2023-06-09 12:30:00')

days, hours, minutes, seconds = get\_time\_difference(start\_date, end\_date)

print("Time difference: {} days, {} hours, {} minutes, {} seconds".format(days, hours, minutes, seconds))

---------------------------------------------------------------------------------------------------------

1. Write a Python program to do the following operations:  
   Data set: Iris.csv  
   a) Loading data from CSV file  
   b) Compute the basic statistics of given data - shape, no. of columns, mean  
   c) Visualize data using Scatter plot

import pandas as pd

import matplotlib.pyplot as plt

# a) Loading data from CSV file

data = pd.read\_csv("Iris2.csv")

# b) Compute the basic statistics of given data

shape = data.shape

num\_columns = len(data.columns)

mean = data.mean(numeric\_only=True, skipna=True)

print("Shape:", shape)

print("Number of columns:", num\_columns)

print("Mean:\n", mean)

colors = {"setosa": "red", "versicolor": "green", "virginica": "blue"}

for species, group in species\_groups:

plt.scatter(group["SepalLengthCm"], group["SepalWidthCm"], c=colors[species], label=species)

plt.xlabel("Sepal Length (cm)")

plt.ylabel("Sepal Width (cm)")

plt.legend()

plt.title("Scatter plot of Iris dataset")

plt.show()

---------------------------------------------------------------------------------------------------------

1. Write a python code for handling missing data using fillna(),dropna() method on Iris.csv file[fill the data using mean/median/mode value]

import pandas as pd

df = pd.read\_csv('Iris.csv')

print("DataFrame before handling missing data:")

print(df.head())

df\_filled\_mean = df.fillna(df.mean())

df\_filled\_median = df.fillna(df.median())

df\_filled\_mode = df.fillna(df.mode().iloc[0])

df\_dropped = df.dropna()

print("\nDataFrame after filling missing values with mean:")

print(df\_filled\_mean.head())

print("\nDataFrame after filling missing values with median:")

print(df\_filled\_median.head())

print("\nDataFrame after filling missing values with mode:")

print(df\_filled\_mode.head())

---------------------------------------------------------------------------------------------------------

1. Create a DataFrame from a dictionary with the following data:

'Name': ['John', 'Emma', 'Peter', 'Olivia']

'Age': [25, 30, 28, 35]

'City': ['New York', 'London', 'Paris', 'Sydney']

1. Display the created DataFrame.
2. Access the 'Name' column of the DataFrame.
3. Access the second row of the DataFrame.
4. Add a new column 'Gender' with values ['M', 'F', 'M', 'F'] to the DataFrame.
5. Update the 'City' value of the third row to 'Tokyo'.\

---------------------------------------------------------------------------------------------------------

1. Use Bollywood.csv file to answer this questions:

a) How many records are present in the dataset? Print the metadata information of the dataset.

b)How many movies got released in each genre? Which genre had highest number of releases? Sort number of releases in each genre in descending order.

c) Which month of the year, maximum, number movie releases are seen?(Note: Extract a new column called month from releaseDate column)

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

data = pd.read\_csv("bollywood.csv")

num\_records = len(data)

print("Number of records in the dataset:", num\_records)

print("\nMetadata information:")

print(data.info())

genre\_counts = data['Genre'].value\_counts()

print("\nNumber of movies released in each genre:")

print(genre\_counts)

print("\nGenre with the highest number of releases:", genre\_counts.idxmax())

print("\nNumber of releases in each genre (descending order):")

print(genre\_counts.sort\_values(ascending=False))

data['ReleaseDate'] = pd.to\_datetime(data['ReleaseDate'])

data['Month'] = data['ReleaseDate'].dt.month

month\_counts = data['Month'].value\_counts()

max\_month = month\_counts.idxmax()

print("\nMonth with the maximum number of movie releases:", max\_month)

---------------------------------------------------------------------------------------------------------

1. Use Bollywood.csv file to answer this questions:

a) Which are the top 10 movies with maximum return on investment (ROI)? Calculate Return on investment (ROI) as (BoxOfficeCollection –Budget) / Budget.

b) Draw a histogram and a distribution plot to find out the distribution of movie budget. Interpret the plot to conclude if the most movies are high or low budgeted movies.

c) Which Genre of movies typically sees more youtube likes? Draw boxplots for each genre of movies to compare.

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

data = pd.read\_csv("bollywood.csv")

# e) Top 10 movies with maximum return on investment (ROI)

data['ROI'] = (data['BoxOfficeCollection'] - data['Budget']) / data['Budget']

top\_10\_movies\_roi = data.nlargest(10, 'ROI')

print("\nTop 10 movies with maximum ROI:")

print(top\_10\_movies\_roi[['MovieName', 'ROI']])

# g) Histogram and distribution plot of movie budget

plt.figure(figsize=(10, 6))

sns.histplot(data['Budget'], bins=20, kde=True)

plt.xlabel('Budget')

plt.ylabel('Count')

plt.title('Distribution of Movie Budget')

plt.show()

# h) Boxplots of youtube likes by genre

plt.figure(figsize=(10, 6))

sns.boxplot(data=data, x='Genre', y='YoutubeLikes')

plt.xlabel('Genre')

plt.ylabel('Youtube Likes')

plt.title('Youtube Likes by Genre')

plt.xticks(rotation=45)

plt.show()

---------------------------------------------------------------------------------------------------------

1. Use company\_sales\_data.csv file to answer this questions:  
   a. Read Total profit of all months and show it using a line plot  
   b. Get total profit of all months and show line plot with the following Style properties:  
    i. Line Style dotted and Line-color should be red  
    ii. Show legend at the lower right location.  
    iii. X label name = Month Number  
    iv. Y label name = Sold units number

import pandas as pd

import matplotlib.pyplot as plt

data = pd.read\_csv('company\_sales\_data.csv')

months = data['month\_number']

profit = data['total\_profit']

plt.plot(months, profit)

plt.xlabel('Month Number')

plt.ylabel('Total Profit')

plt.title('Total Profit of All Months')

plt.show()

plt.plot(months, profit, linestyle='dotted', color='red', marker='o', markerfacecolor='red', linewidth=3)

plt.xlabel('Month Number')

plt.ylabel('Sold units number')

plt.title('Total Profit of All Months')

plt.legend(['Total Profit'], loc='lower right')

plt.show()

---------------------------------------------------------------------------------------------------------

1. Use company\_sales\_data.csv file to answer this questions:  
   a. Read all product sales data and show it using a multiline plot  
   b. Read face cream and facewash product sales data and show it using the bar chart  
   c. Calculate total sale data for last year for each product and show it using a Pie chart

import pandas as pd

import matplotlib.pyplot as plt

data = pd.read\_csv('company\_sales\_data.csv')

products = data.columns[1:-1] # Exclude 'month\_number' and 'total\_profit' columns

for product in products:

plt.plot(months, data[product])

plt.xlabel('Month Number')

plt.ylabel('Sold units number')

plt.title('Product Sales Data')

plt.legend(products)

plt.show()

face\_cream\_sales = data['facecream']

facewash\_sales = data['facewash']

plt.bar(months, face\_cream\_sales, label='Face Cream')

plt.bar(months, facewash\_sales, label='Facewash', alpha=0.7)

plt.xlabel('Month Number')

plt.ylabel('Sold units number')

plt.title('Face Cream and Facewash Sales Data')

plt.legend()

plt.show()

last\_year\_data = data[data['month\_number'] > 8]

total\_sales = last\_year\_data[products].sum()

plt.pie(total\_sales, labels=products, autopct='%1.1f%%')

plt.title('Total Sales for Last Year')

plt.show()

---------------------------------------------------------------------------------------------------------

1. Create a NumPy array of shape (5, 5) with random integers ranging from 1 to 100. Perform the following operations:
2. Find the sum of all elements in the array.
3. Calculate the mean of each row in the array.
4. Compute the standard deviation of each column in the array.

import numpy as np

arr = np.random.randint(1, 101, size=(5, 5))

print("Array:")

print(arr)

total\_sum = np.sum(arr)

print("\nSum of all elements:", total\_sum)

row\_means = np.mean(arr, axis=1)

print("\nMean of each row:")

print(row\_means)

column\_std = np.std(arr, axis=0)

print("\nStandard deviation of each column:")

print(column\_std)