MACHINE LEARNING

LAB CYCLE: 1

Submitted By:

Fathima Lana R T

I MSc CS (ML)

COS250903

# EXPERIMENT: 1.1

## PROBLEM:

Implement a program to take user input for age and check eligibility for voting using if-else.

## AIM:

To develop a Python program that accepts age as user input and checks voting eligibility using an if-else statement.

## ALGORITHM:

Aglorithm CheckEligibility(limit)

{

read(limit);

for i := 1 to limit do

{

read (age);

if (age<18) then

print(" you are eligible ");

i := i + 1;

else

print (" you are not eligible ");

        i := i + 1;

}

}

## RESULT:

The program successfully takes user input for age to check voting eligibility.

## SOURCE CODE:

## Check Voting Eligibility:

limit = int(input("Enter a limit "))

for i in range(limit):

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

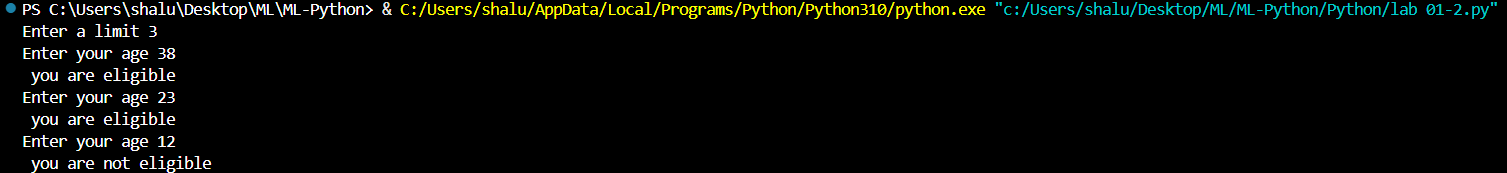
    if(age>18):

        print(" you are eligible ")

    else:

        print(" you are not eligible ")

## OUTPUT:



# EXPERIMENT: 1.2

## PROBLEM:

Implement a program to print the first 10 numbers using a for loop and a while loop.

## AIM:

To develop a Python program that prints the first 10 natural numbers using a for loop and while loop.

## ALGORITHM:

Algorithm PrintNumbers()

{

print (“While Loop”);

j := 1;

while j<=10 do

{

print (“j”);

j := j+1;

}

print (“For Loop”);

for i := 1 to 10 do

{

print (“i”);

}

## RESULT:

The program successfully displays the first 10 numbers using both for loop and while loop methods.

## SOURCE CODE:

print("while loop")

j=1

while(j<=10):

print(j ,end = " ")

j+=1

print("\n for loop")

for i in range (1,11):

print(i , end = " ")

## OUTPUT:

# EXPERIMENT: 2

## PROBLEM:

Write a Python function to compute factorial recursively and non-recursively.

## AIM:

To write a Python program to compute the factorial of a given number using both recursive and non-recursive functions.

## ALGORITHM:

Algorithm FactRecursion(num)

{

function fact(num);

{

if (num = 0 or num = 1) then

return 1;

else

return num \* fact (num - 1);

read (num);

print (" factorial ", fact (num));

}

}

Algorithm FactNonRecursion(num)

{

read (num);

fact := 1;

while num != 0 do

{

fact := fact\*num;

num := num-1;

}

print (“factorial”, fact)

}

## RESULT:

The program correctly computes the factorial of a number using non recursive and method.

## SOURCE CODE:

**//Recursion**

def fact (num):

    if num == 0 or num ==1:

        return 1

    else:

        return num \* fact(num -1)

num = int(input(" enter the no  "))

print(" factorial " + str (fact(num)))

**//Non - Recursion**

num=int(input("enter the no  "))

fact=1

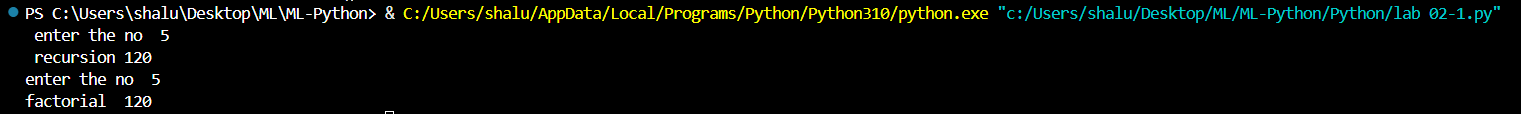
while(num!=0):

    fact=fact\*num

    num-=1

print("factorial  "+str(fact))

## OUTPUT:



# EXPERIMENT: 3

## PROBLEM:

Implement the following:

- Read from a text file.

- Write processed output to a new file.

## AIM:

To develop a Python program that:

- Reads data from a text file.

- Processes the contents and writes the output to a new text file.

## ALGORITHM:

Algorithm FileRead()

{

open “inp.txt” in write mode;

write “Helooo Hiiii” into file;

close inp.txt;

open “inp.txt” in read mode;

print (read contents of file);

close inp.txt;

}

Algorithm FileAppend()

{

open “inp.txt” in read mode;

open “out.txt” in write mode;

read contents of “inp.txt” and write into “out.txt”;

close “inp.txt”;

close “out.txt”;

open “out.txt” in append mode;

for i := 1 to 24 do

{

num := convert i to string

write num into “out.txt”

}

close “out.txt”;

}

## RESULT:

The program successfully reads data from a specified text file and writes the processed content into a new output file.

## SOURCE CODE:

### File Read:

file = open("out.txt","w")

file.write("Helooo Hiii")

file.close()

file = open("out.txt")

print(file.read())

file.close()

### File Append:

file=open("inp.txt","r")

file2=open("out.txt", "w")

file2.write(file.read() + "\n")

file.close()

file2.close()

file = open("out.txt", "a")

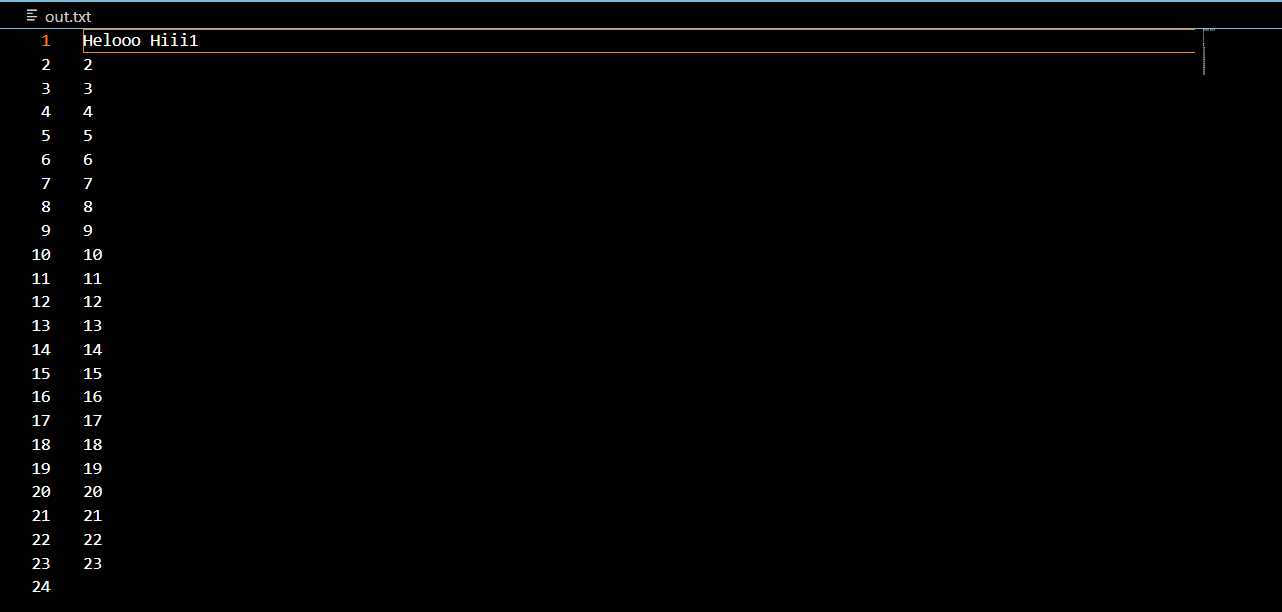
for i in range(1, 24):

    num=str(i)+"\n"

    file.write(num)

file.close()

## OUTPUT:



# EXPERIMENT: 4

## PROBLEM:

Implement a class Person with attributes name and age. Include a method to display the details.

## AIM:

To write a Python program to implement a class Person with attributes name and age, and to display the details of the person using a class method.

## ALGORITHM:

Algorithm ClassData(n,Datas)

{

define class Data with attributes:

name (string);

age (integer, default = 22);

read (n);

Datas := [];

while n != 0 do

{

p1 := Data();

p1.name := read (name);

p1.age := read (age);

append p1 to Datas;

n := n-1;

}

for i := 1 to length(Datas) do

{

print ("No: ", i);

print ("Name: ", Datas[i].name);

print ("Age: ", Datas[i].age);

}

}

## RESULT:

The program defines a Person class with name and age attributes and includes a method to display these details correctly.

## SOURCE CODE:

class Data:

    name = ""

    age = 22

n = int(input ("How many data do you want to store:"))

Datas = []

while n != 0:

    p1 = Data()

    p1.name = input("enter your name")

    p1.age = int(input("enter your age"))

    Datas.append(p1)

    n -= 1

for i,data in enumerate(Datas,1):

    print(f"\n No:{i} \n Name:{data.name} \n Age:{data.age}\n")

## OUTPUT:



# EXPERIMENT: 5

## PROBLEM:

Write a Python program to handle division by zero using try-except block.

## AIM:

To write a Python program that performs division of two numbers and handles the division by zero error using a try-except block.

## ALGORITHM:

Algorithm DivisionByZero()

{

read (num);

read (den);

try

{

result := num / den;

print("Result : ", num, "/", den, " = ", result);

}

catch (ZeroDivisionError)

{

print("Divison by 0 not possible");

}

}

## RESULT:

The program effectively handles a division-by-zero error using a try-except block, preventing the program from crashing

## SOURCE CODE:

num = int(input("Enter a numerator  "))

den = int(input("Enter a denominator  "))

try:

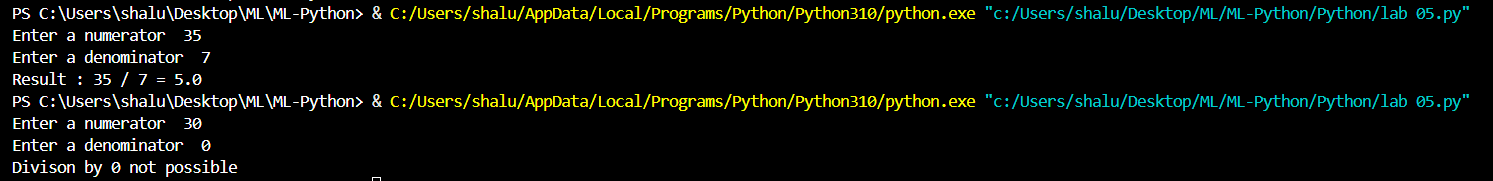
    result = num / den

    print(f"Result : {num} / {den} = {result} ")

except ZeroDivisionError:

    print("Divison by 0 not possible")

## OUTPUT:



# EXPERIMENT: 6

## PROBLEM:

Implement the following:

- Create a NumPy array.

- Perform element-wise addition, multiplication.

- Compute mean, variance, standard deviation.

## AIM:

To develop a Python program that creates a NumPy array, performs element-wise addition and multiplication, and computes the mean, variance, and standard deviation of the array elements.

## ALGORITHM:

Algorithm NumpyArray(arr1,arr2,add,mul,mean,var,sd)

{

Import numpy library;

read arr1;

read arr2;

print("Array 1:", arr1);

print("Array 2:", arr2);

add := arr1 + arr2;

print("Addition:", add);

mul := arr1 \* arr2;

print("Multiplication:", mul);

mean = mean(arr1);

var = var(arr1);

sd = std(arr1);

print("Mean of arr1:", mean);

print("Variance of arr1:", var);

print("Standard Deviation of arr1:", sd);

}

## RESULT:

The program successfully performs element-wise addition and multiplication on a NumPy array, and correctly computes its mean, variance, and standard deviation.

## SOURCE CODE:

import numpy as np

arr1 = np.array(list(map(int, input("\nEnter elements of Array 1 \n").split())))

arr2 = np.array(list(map(int, input("\nEnter elements of Array 2\n ").split())))

print("\nArray 1:", arr1)

print("\nArray 2:", arr2)

add = arr1 + arr2

print("\nAddition:", add)

mul = arr1 \* arr2

print("\nMultiplication:", mul)

mean = np.mean(arr1)

var = np.var(arr1)

sd = np.std(arr1)

print("\nMean of arr1:", mean)

print("\nVariance of arr1:", var)

print("\nStandard Deviation of arr1:", sd)

## OUTPUT:

# EXPERIMENT: 7

## PROBLEM:

Use NumPy to:

- Create matrices.

- Perform matrix multiplication, transpose, determinant, inverse (if exists).

## AIM:

To write a Python program using NumPy to create matrices and perform various matrix operations such as multiplication, transpose, determinant, and inverse (if it exists).

## ALGORITHM:

Algorithm MatrixOperations(rowsa,colsa,rowsb,colsb,A,B)

{

Import numpy library;

read (rowsa);

read (colsa);

read (rowsb);

read (colsb);

print("Enter elements of Matrix A ”,rowsa,”x”,colsa);

A := [];

for i := 1 to rowsa do

{

read (row);

append row to A;

}

A := convert A to numpy array;

for i := 1 to rowsb do

{

read (row);

append row to B;

}

B := convert B to numpy array;

print("Matrix A:", A);

print("Matrix B:", B);

if A.shape == B.shape then

D := A+B;

 print("A + B:", D);

else

    print("Matrix addition not possible (different dimensions)");

if colsa = rowsb then

C := A x B;

print("A x B:", C);

else

print("Matrix multiplication not possible (columns of A != rows of B)");

print("Transpose of A:", Transpose(A));

if rowsa = colsa then

detA := Determinant(A);

print("Determinant of A:", detA);

if detA != 0 then

invA := Inverse(A);

print("Inverse of A:", invA);

else

print("Matrix A is singular, no inverse exists");

else

print("Determinant and Inverse are not defined for non-square matrices");

}

## RESULT:

The program successfully uses NumPy to create matrices and perform essential matrix operations like multiplication, transpose, determinant, and inverse.

## SOURCE CODE:

import numpy as np

rowsa = int(input("Enter number of rows A: "))

colsa = int(input("Enter number of columns A: "))

rowsb = int(input("Enter number of rows B: "))

colsb = int(input("Enter number of columns B: "))

print(f"Enter elements of Matrix A ({rowsa}x{colsa}):")

A = []

for i in range(rowsa):

    row = list(map(int, input().split()))

    A.append(row)

A = np.array(A)

print(f"Enter elements of Matrix B ({rowsb}x{colsb}):")

B = []

for i in range(rowsb):

    row = list(map(int, input().split()))

    B.append(row)

B = np.array(B)

print("\nMatrix A:\n", A)

print("Matrix B:\n", B)

if A.shape == B.shape:

    D = A + B

    print("\nA + B:\n", D)

else:

    print("\nMatrix addition not possible (different dimensions)")

if A.shape[1] == B.shape[0]:

    C = np.dot(A, B)

    print("\nA x B:\n", C)

else:

    print("\nMatrix multiplication not possible (columns of A != rows of B)")

print("\nTranspose of A:\n", A.T)

if A.shape[0] == A.shape[1]:

    detA = np.linalg.det(A)

    print("\nDeterminant of A:", detA)

    if detA != 0:

        invA = np.linalg.inv(A)

        print("\nInverse of A:\n", invA)

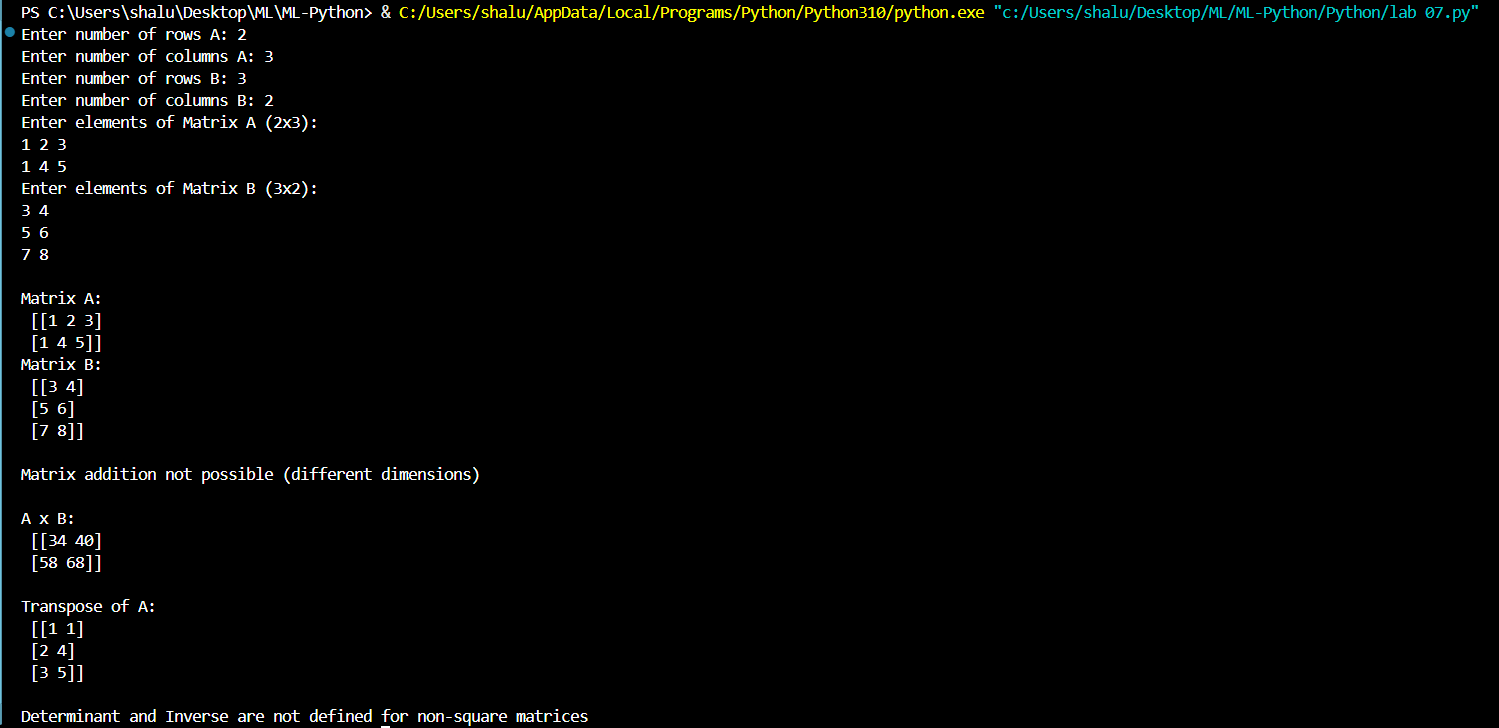
    else:

        print("\nMatrix A is singular, no inverse exists")

else:

    print("\nDeterminant and Inverse are not defined for non-square matrices")

## OUTPUT:



# EXPERIMENT: 8

## PROBLEM:

Load a CSV file using Pandas.

- Display head and tail of the dataset.

- Compute basic statistics: mean, median, mode.

- Handle missing values.

## AIM:

To develop a Python program using Pandas to:

- Load a dataset from a CSV file.

- Display the first few (head) and last few (tail) records of the dataset.

- Compute basic statistical measures such as mean, median, and mode.

- Identify and handle missing values in the dataset.

## ALGORITHM:

Algorithm DataLoading()

{

Import pandas library;

read CSV file into data;

print("Head :", head(data));

print("Tail :", tail(data));

mean\_sal := compute mean of Salary;

median\_sal := compute median of Salary;

mode\_sal := compute mode of Salary;

print("Mean of Salary :", mean\_sal);

print("Median of Salary :", median\_sal);

print("Mode of Salary :", mode\_sal);

}

## RESULT:

Using the Pandas library, the CSV dataset was loaded and analysed. Several key operations were performed to retrieve and prepare the data for further analysis.

## SOURCE CODE:

import pandas as pd

data = pd.read\_csv(r"C:\Users\shalu\Desktop\ML\ML-Python\Python\Salary\_dataset.csv")

print("Head :\n", data.head())

print("\n")

print("Tail :\n", data.tail())

print("\n")

mean\_sal=data['Salary'].mean()

median\_sal=data['Salary'].median()

mode\_sal=data['Salary'].mode()

print("Mean of Salary :\n",mean\_sal)

print("\n")

print("Median of Salary :\n",median\_sal)

print("\n")

print("Mode of Salary :\n",mode\_sal)

print("\n")

## OUTPUT:



# EXPERIMENT: 9

## PROBLEM:

Plot the graphs using matplotlib.

- Plot a simple line graph of y = x^2.

- Plot bar charts and histograms.

## AIM:

To develop a Python program using Matplotlib to plot graphs:

- A simple line graph of y = x^2.

- Bar charts and histograms.

## ALGORITHM:

Algorithm GraphPlots()

{

Import pandas, numpy, matplotlib.pyplot libraries;

data := read\_csv("Salary\_Dataset.csv");

fig, (x1, x2, x3) := subplots(1, 3, figsize=(18, 6));

x := linspace(-5, 5, 50);

y := x²;

plot(x1, x, x², color := "blue");

set title(x1, "y = x²");

set xlabel(x1, "x");

set ylabel(x1, "y");

group data by "Job Roles";

compute sum of "Salaries Reported" for each role;

sort values in descending order;

plot bar chart (roles, salaries) on subplot x2;

set title = "Number of Salaries Reported by Job Role";

set xlabel = "Job Role";

set ylabel = "Number of Salaries Reported";

rotate x-axis labels by 45°;

take "Salary" column from data;

plot histogram with 15 bins, purple color, edge color black on subplot x3;

set title = "Salary Distribution";

set xlabel = "Salary", ylabel = "Frequency";

enable y-axis gridlines;

tight\_layout();

show();

}

## RESULT:

The program successfully generates a line graph for y=x2, a bar chart, and a histogram of a given dataset using the Matplotlib library.

## SOURCE CODE:

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

data = pd.read\_csv(r"C:\Users\shalu\Desktop\ML\ML-Python\Python\Salary\_Dataset.csv")

fig, (x1, x2, x3) = plt.subplots(1, 3, figsize=(18, 6))

# Line graph: y = x²

x = np.linspace(-5, 5, 50)

x1.plot(x, x\*\*2, 'b-')

x1.set\_title('y = x²')

x1.set\_xlabel("x")

x1.set\_ylabel("y")

x1.grid(True)

# Bar chart: Number of Salaries Reported by Job Role

salaries\_reported\_by\_role = data.groupby("Job Roles")["Salaries Reported"].sum().sort\_values(ascending=False)

x2.bar(salaries\_reported\_by\_role.index, salaries\_reported\_by\_role.values, color="skyblue", edgecolor="black")

x2.set\_title("Number of Salaries Reported by Job Role")

x2.set\_xlabel("Job Role")

x2.set\_ylabel("Number of Salaries Reported")

x2.tick\_params(axis="x", rotation=45)

# Histogram: Salary distribution

x3.hist(data['Salary'], bins=15, alpha=0.7, color='purple', edgecolor="black")

x3.set\_title('Salary Distribution')

x3.set\_xlabel("Salary")

x3.set\_ylabel("Frequency")

x3.grid(axis="y", linestyle="--", alpha=0.7)

plt.tight\_layout()

plt.show()

## OUTPUT:

## 

# EXPERIMENT: 10

## PROBLEM:

Visualize:

- Plot a scatter plot of two features from the Iris dataset.

- Use color to indicate different classes.

## AIM:

To write a Python program to visualize the Iris dataset by plotting a scatter plot of two features, using different colors to represent the classes.

## ALGORITHM:

Algorithm VisualizeIris()

{

Import seaborn, pandas, matplotlib.pyplot libraries;

iris := read\_csv("iris.csv");

scatterplot(

x := "QUANTITYORDERED",

y := "SALES”

hue := "DEALSIZE",

data := iris,

palette := "tab10"

);

title := "Sales vs Quantity Ordered (by Deal Size)";

xlabel := "Quantity Ordered";

ylabel := "Sales ($)";

legend(title := "Deal Size");

tight\_layout();

show();

}

## RESULT:

To visualize relationships between features of the Iris dataset using scatter plots.

## SOURCE CODE:

import seaborn as sns

import pandas as pd

import matplotlib.pyplot as plt

file\_path = r'C:\Users\shalu\Desktop\ML\ML-Python\Python\iris2.csv'

df = pd.read\_csv(file\_path)

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

sns.scatterplot(

    x='QUANTITYORDERED',

    y='SALES',

    hue='DEALSIZE',

    data=df,

    palette='tab10',

    s=80,

    alpha=0.7

)

plt.title("Sales vs Quantity Ordered (by Deal Size)")

plt.xlabel("Quantity Ordered")

plt.ylabel("Sales ($)")

plt.legend(title='Deal Size', bbox\_to\_anchor=(1.05, 1), loc='upper left')

plt.tight\_layout()

plt.show()

## OUTPUT:

