Python Lab Programs

# Exercise 1 Installation

# Install packages using pip  
pip install numpy scipy pandas matplotlib scikit-learn jupyter  
# To start jupyter  
jupyter notebook

# Exercise 2 Numpy Arrays

import numpy as np  
  
# a. Create and display 1D array  
arr1 = np.array([1, 2, 3])  
print("1D Array:", arr1)  
  
# b. Matrix addition, subtraction, multiplication  
A = np.array([[1, 2], [3, 4]])  
B = np.array([[5, 6], [7, 8]])  
print("Addition:  
", A + B)  
print("Subtraction:  
", A - B)  
print("Multiplication:  
", np.dot(A, B))  
  
# c. Slicing a matrix  
print("Sliced:  
", A[:, 1])  
  
# d. Transpose  
print("Transpose:  
", A.T)

# Exercise 3 Numpy Statistics

import numpy as np  
  
arr = np.array([1, 2, 3, 4, 5])  
print("Max:", np.max(arr))  
print("Min:", np.min(arr))  
print("Sum:", np.sum(arr))  
print("Mean:", np.mean(arr))  
print("Standard Deviation:", np.std(arr))

# Exercise 4 Numpy File Io

import numpy as np  
  
arr = np.array([10, 20, 30])  
np.save("saved\_array.npy", arr)  
  
loaded = np.load("saved\_array.npy")  
print("Loaded array:", loaded)

# Exercise 5 Scipy Integration

from scipy import integrate  
import numpy as np  
  
# Single  
res1, \_ = integrate.quad(lambda x: x\*\*2, 0, 1)  
print("Single:", res1)  
  
# Double  
res2, \_ = integrate.dblquad(lambda x, y: x\*y, 0, 1, lambda x: 0, lambda x: 2)  
print("Double:", res2)  
  
# Triple  
res3, \_ = integrate.tplquad(lambda x, y, z: x\*y\*z, 0, 1, lambda x: 0, lambda x: 1, lambda x, y: 0, lambda x, y: 1)  
print("Triple:", res3)

# Exercise 6 Scipy Interpolation

from scipy import interpolate  
import numpy as np  
  
# 1D interpolation  
x = np.linspace(0, 10, 10)  
y = np.sin(x)  
f = interpolate.interp1d(x, y)  
print("Interpolated:", f(5.5))  
  
# Multivariate  
from scipy.interpolate import griddata  
points = np.array([[0, 0], [1, 0], [0, 1], [1, 1]])  
values = np.array([0, 1, 1, 0])  
grid\_x, grid\_y = np.mgrid[0:1:5j, 0:1:5j]  
result = griddata(points, values, (grid\_x, grid\_y), method='linear')  
print("Grid:  
", result)

# Exercise 7 Linear Equations

from scipy.linalg import solve  
import numpy as np  
  
A = np.array([[3, 1], [1, 2]])  
b = np.array([9, 8])  
x = solve(A, b)  
print("Solution:", x)

# Exercise 8 Pandas Basics

import pandas as pd  
  
# Series  
s = pd.Series([1, 2, 3])  
print("Series:", s)  
  
# DataFrame  
data = {'Name': ['Alice', 'Bob'], 'Age': [25, 30]}  
df = pd.DataFrame(data)  
print("DataFrame:  
", df)

# Exercise 9 Data Cleansing

import pandas as pd  
import numpy as np  
  
df = pd.DataFrame({'A': [1, np.nan, 3], 'B': [4, 5, np.nan]})  
print("Original:  
", df)  
  
# Check missing  
print("Is Null:  
", df.isnull())  
  
# Fill missing  
print("Filled:  
", df.fillna(0))  
  
# Drop missing  
print("Dropped:  
", df.dropna())

# Exercise 10 Csv Processing

import pandas as pd  
  
df = pd.read\_csv('data.csv') # Assume data.csv is available  
print("Full:  
", df)  
print("Specific Rows:  
", df.iloc[0:2])  
print("Specific Columns:  
", df[['Name']])  
print("Specific Row+Col:  
", df.loc[0, 'Name'])

# Exercise 11 Matplotlib Plots

import matplotlib.pyplot as plt  
import numpy as np  
  
# Line chart  
x = np.linspace(0, 10, 100)  
y = np.sin(x)  
plt.plot(x, y)  
plt.xlabel('x')  
plt.ylabel('sin(x)')  
plt.title('Line Chart')  
plt.savefig('line\_chart.png')  
plt.close()  
  
# Box plot  
plt.boxplot(np.random.randn(100))  
plt.savefig('boxplot.png')  
plt.close()  
  
# Scatter plot  
plt.scatter(np.random.rand(10), np.random.rand(10))  
plt.savefig('scatter.png')  
plt.close()  
  
# Bubble chart  
x = np.random.rand(10)  
y = np.random.rand(10)  
size = np.random.rand(10) \* 300  
plt.scatter(x, y, s=size, alpha=0.5)  
plt.savefig('bubble.png')  
plt.close()  
  
# 3D plot  
from mpl\_toolkits.mplot3d import Axes3D  
fig = plt.figure()  
ax = fig.add\_subplot(111, projection='3d')  
x = y = np.linspace(-5, 5, 100)  
X, Y = np.meshgrid(x, y)  
Z = np.sin(np.sqrt(X\*\*2 + Y\*\*2))  
ax.plot\_surface(X, Y, Z, cmap='viridis')  
plt.savefig('3dplot.png')  
plt.close()

# Exercise 12 Decision Tree

from sklearn.datasets import load\_iris  
from sklearn.tree import DecisionTreeClassifier  
  
iris = load\_iris()  
X, y = iris.data, iris.target  
  
model = DecisionTreeClassifier()  
model.fit(X, y)  
  
print("Prediction:", model.predict([X[0]]))

# Exercise 13 Knn

from sklearn.datasets import load\_iris  
from sklearn.neighbors import KNeighborsClassifier  
  
iris = load\_iris()  
X, y = iris.data, iris.target  
  
model = KNeighborsClassifier(n\_neighbors=3)  
model.fit(X, y)  
  
print("Prediction:", model.predict([X[0]]))

# Exercise 14 Networking

import socket  
  
# a. Local machine name and IP  
print("Hostname:", socket.gethostname())  
print("Local IP:", socket.gethostbyname(socket.gethostname()))  
  
# b. Remote machine IP  
print("Google IP:", socket.gethostbyname('www.google.com'))  
  
# c. IPv4 conversion  
ip = '192.168.1.1'  
packed\_ip = socket.inet\_aton(ip)  
print("Packed IP:", packed\_ip)  
  
# d. Service name  
print("Service on port 80:", socket.getservbyport(80, 'tcp'))  
  
# e. Byte order conversion  
print("Host to network (16-bit):", socket.htons(1234))  
print("Network to host (32-bit):", socket.ntohl(12345678))

# Exercise 15 Echo Server Client

# Server  
# Run this in one terminal  
import socket  
  
server\_socket = socket.socket()  
server\_socket.bind(('localhost', 12345))  
server\_socket.listen(1)  
print("Server listening...")  
conn, addr = server\_socket.accept()  
print("Connected by", addr)  
data = conn.recv(1024)  
conn.sendall(data)  
conn.close()  
  
# Client  
# Run this in another terminal  
import socket  
  
client\_socket = socket.socket()  
client\_socket.connect(('localhost', 12345))  
client\_socket.sendall(b'Hello, Server!')  
data = client\_socket.recv(1024)  
print("Received:", data.decode())  
client\_socket.close()