```
# -*- coding: utf-8 -*-
"""live assignment.ipynb

Automatically generated by
Colab.

Original file is located at

https://colab.research.google.com/drive/1jxH9CpdrUrDvlTVjqaLsNfjUk2oqZNgG

1. Demonstrate three
different methods for creating identical 2D arrays in NumPyƒ Provide the code for each

method and the final output after each method.
"""

import pandas as pd

# using
np.array
import numpy as np
a = np.array([[1,3], [2,4]])
print('2D array using
np.array:')
print(a)

# using np.zeros (Return a new array of given shape and type, filled with
zeros.)
a_zeros = np.zeros((5,6))
print('2D array using np.zeros')
print(a_zeros)

# using
np.ones  (Return a new array of given shape and type, filled with ones.)
a_ones =
np.ones((5,6))
print('2D array usining np.ones')
print(a_ones)

"""2. Using the
Numpy function, generate an array of 100 evenly spaced numbers between 1 and 10 and Reshape
that 1D array into a 2D array."""

array_1d = np.linspace(1, 10, 100)
array_2d
=array_1d.reshape((10, 10))

print('1D array of evenly spaced
numbers:')
print(array_1d)

print('\nReshaped 2D array:')
print(array_2d)

"""3.
Explain the following terms:

 The difference in np.array, np.asarray and np.asanyarray.

  The
difference between Deep copy and shallow copy.
"""

# 1. np.array

#    a. this
function creates a new array from any object exposing the array interface, or from any nested
sequence.
#    b. By default, np.array makes a copy of the input data.
#    c. It always
returns a new array even if the input is already an ndarray.

data = [11, 2, 3]
arr =
np.array(data)

# 2. np.asarray
# a. This function converts the input to an ndarray, but unline
'np.array', it will not make a copy of the input data if it is already an ndarrY.
# b. This
function is useful when you want to ensure that an object is an ndarray without incurring the
overhead of copying the data..

data1 = np.array([1, 2, 3])
arr = np.asarray(data1)

# 3..
np.asanyarray:

# a. Similar to np.asarray, ths function converts the input to an ndarray, but
it passes through subclass of ndarray..
# b. If the input s already a subclasss of ndarray,
like a masked array or matrix, np.aanyarray will return unchanged..

data = np.array([1, 2,
3])
arr = np.asanyarray(data)

data

data1

"""4. The Difference Between Deep
Copy and Shallow Copy

Deep Copy:

a. A deep copy of an object creates a new object and
recursively copies all objects found in the original.

b. Changes to the new object will not
affect the original object because they do not share the same memory
locations.
"""

import copy
data = [1, [2, 3], 4]
deep_copy =
copy.deepcopy(data)
deep_copy[1][0] = 99 # this change will not reflect in the original
list
print(data)

"""Shallow Copy:

a. A shallow copy of an object creates a new
object, but inserts references into it to the objects found in the original.

b. Changes to
mutable objects within the copied object will affect the original object because they reference
the same memory location.
"""

data = [1, [2,  3], 44]
shallow_copy =
copy.copy(data)

shallow_copy[1][0] = 99 # this change will reflect in the original
list
print(data)

random_array = np.random.uniform(0.5, 20, (3, 3))
rounded_array =
np.round(random_array, 2)
print("Random
Array:")
print(random_array)
print("\nRounded
Array:")
```

```
print(rounded_array)

"""5. Create a NumPy array with random
integers between 1 and 10 of shape (5,6)).. After creating the array
perform the following
operations:

 a)Extract all even integers from array.

 b)Extract all odd integers from
array
"""

random_array = np.random.randint(1, 11, (5, 6))
even_integers =
random_array[random_array % 2 == 0] #for even integer
odd_integers = random_array[random_array
% 2 != 0] # for odd integer
print("Random
Array:")
print(random_array)
print("\nEven
Integers:")
print(even_integers)
print("\nOdd
Integers:")
print(odd_integers)

"""6. Create a 3D NumPy array of shape (3,
3, 3) containing random integers between 1 and 10. Perform the following operations:

 a) Find
the indices of the maximum values along each depth level (third axis).

 b) Perform
element-wise multiplication of between both array
"""

# Create the first 3D
array of shape (3, 3, 3) with random integers between 1 and 10
array1 = np.random.randint(1,
11, (3, 3, 3))
print("Array1:")
print(array1)

# Find the indices of the maximum
values along each depth level (axis=2)
max_indices = np.argmax(array1,
axis=2)
print("\nIndices of the maximum values along each depth level (third
axis):")
print(max_indices)

# Create the second 3D array of shape (3, 3, 3) with random
integers between 1 and 10
array2 = np.random.randint(1, 11, (3, 3,
3))
print("\nArray2:")
print(array2)

# Perform element-wise multiplication between
array1 and array2
result = np.multiply(array1, array2)
print("\nElement-wise
multiplication of Array1 and Array2:")
print(result)

"""7. Clean and
transform the 'Phone' column in the sample dataset to remove non-numeric characters and convert
it to a numeric data type. Also display the table attributes and data types of each
column."""

import pandas as pd

data = {
    'Name': ['prashu', 'ramu',
'ajay'],
    'Phone': ['(123) 456-7890', '987-654-3210', '555.666.7777']
}

df =
pd.DataFrame(data)
print("Original DataFrame:")
print(df)


df['Phone'] =
df['Phone'].str.replace(r'\D', '', regex=True)

# Convert to numeric data type
df['Phone'] =
pd.to_numeric(df['Phone'])

print("\nCleaned
DataFrame:")
print(df)


print("\nDataFrame
Info:")
print(df.info())

"""8.  Perform the following tasks using people
dataset:

 a) Read the 'data..csv' file using pandas, skipping the first 50 rows.

 b) Only
read the columns: 'Last Name', ‘Gender’,’Email’,‘Phone’ and ‘Salary’ from the
file.

 c) Display the first 10 rows of the filtered dataset.

 d) Extract the ‘Salary’'
column as a Series and display its last 5 values.

"""

# a) Read the
'data..csv' file using pandas, skipping the first 50 rows.
file_path ='People Data.csv'
data =
pd.read_csv(file_path, skiprows=50)
data

# b) Only read the columns: 'Last Name',
‘Gender’,’Email’,‘Phone’ and ‘Salary’ from the file.
file_path ='People
Data.csv'
people_data = pd.read_csv(file_path, usecols=['Last Name', 'Gender', 'Email',
'Phone', 'Salary'])
people_data

# c) Display the first 10 rows of the filtered
dataset.
filtered_first_10_rows = people_data.head(10)
print("First 10 rows of the
filtered dataset:")
print(filtered_first_10_rows)

# d) Extract the ‘Salary’' column
as a Series and display its last 5 values.
salary_series =
people_data['Salary']
last_5_salaries = salary_series.tail(5)
print("\nLast 5 values of
the 'Salary' column:")
print(last_5_salaries)

""" 9.  Filter and select
```

```
rows from the People_Dataset, where the “Last Name' column contains the name 'Duke','Gender'
column contains the word Female and ‘salary’ should be less than
85000."""

filtered_rows = people_data[
    (people_data['Last
Name'].str.contains('Duke', na=False)) &
    (people_data['Gender'].str.contains('Female',
na=False)) &
    (people_data['Salary'] < 85000)
]

print("\nFiltered rows where
'Last Name' contains 'Duke', 'Gender' contains 'Female', and 'Salary' is less than
85000:")
print(filtered_rows)

"""10. Create a 7*5 Dataframe in Pandas
using a series generated from 35. random integers between 1 to
6?"""

random_integers = np.random.randint(1, 7, size=35)
df =
pd.DataFrame(random_integers.reshape(7, 5), columns=[f'Col{i}' for i in range(1,
6)])
print(df)

"""11. Create two different Series, each of length 50, with the
following criteria:

a) The first Series should contain random numbers ranging from 10 to
50.

b) The second Series should contain random numbers ranging from 100 to 1000.

c) Create a
DataFrame by joining these Series by column, and, change the names of the columns to 'col1',
'col2',
etc
"""

series1 = pd.Series(np.random.randint(10, 51, size=50)) #first
Series with random numbers ranging from 10 to 50

series2 = pd.Series(np.random.randint(100,
1001, size=50)) #second Series with random numbers ranging from 100 to 1000


df =
pd.DataFrame({'col1': series1, 'col2': series2})

print(df)

"""12. Perform the
following operations using people data set:

    a) Delete the 'Email', 'Phone', and 'Date of
birth' columns from the dataset.

    b) Delete the rows containing any missing values.

    d)
Print the final output also
"""

df = pd.read_csv("People
Data.csv")
columns_to_drop = ['Email', 'Phone', 'Date of birth'] # Deleting
Columns:
df.drop(columns=columns_to_drop, inplace=True)
df.dropna(inplace=True) # deleting the
row containing missing value

print(df)
```

```
"""14. Create two NumPy arrays, x and
y, each containing 100 random float values between 0 and 1. Perform the
following tasks using
Matplotlib and NumPy:

  a) Create a scatter plot using x and y, setting the color of the
points to red and the marker style to 'o'.

  b) Add a horizontal line at y = 0.5 using a
dashed line style and label it as 'y = 0.5'.

  c) Add a vertical line at x = 0.5 using a
dotted line style and label it as 'x = 0.5'.

  d) Label the x-axis as 'X-axis' and the y-axis
as 'Y-axis'.

  e) Set the title of the plot as 'Advanced Scatter Plot of Random Values'.

  f)
Display a legend for the scatter pplot, the horizontal line, and the vertical
line
"""

import matplotlib.pyplot as plt

x = np.random.rand(100)
y =
np.random.rand(100)

plt.scatter(x, y, color='red', marker='o', label='Data
Points')

plt.axhline(y=0.5, color='blue', linestyle='--', label='y = 0.5') #Add a horizontal
line at y = 0.5

plt.axvline(x=0.5, color='green', linestyle=':', label='x = 0.5') #Add a
vertical line at x = 0.5

plt.xlabel('X-axis')
plt.ylabel('Y-axis')

plt.title('Advanced
Scatter Plot of Random Values')
plt.legend()
plt.show()

"""14. Create a
time-series dataset in a Pandas DataFrame with columns: 'Date', 'Temperature', 'Humidity'
and
Perform the following tasks using Matplotlib:

  a)  Plot the 'Temperature' and 'Humidity'
on the same plot with different y-axes (left y-axis for 'Temperature' and
right y-axis for
'Humidity').

  b) Label the x-axis as 'Date'.

  c) Set the title of the plot as 'Temperature
and Humidity Over Time'.
"""

np.random.seed(0)
date_range =
pd.date_range(start='2023-01-01', periods=30, freq='D')
temperature =
np.random.uniform(low=-10, high=30, size=30)
humidity = np.random.uniform(low=20, high=80,
size=30)

data = {
    'Date': date_range,
    'Temperature': temperature,
    'Humidity':
```

```
humidity
}

df = pd.DataFrame(data)

fig, ax1 =
plt.subplots()

ax1.set_xlabel('Date')
ax1.set_ylabel('Temperature',
color='tab:red')
ax1.plot(df['Date'], df['Temperature'], color='tab:red',
label='Temperature')
ax1.tick_params(axis='y', labelcolor='tab:red')

ax2 = ax1.twinx()  #
instantiate a second axes that shares the same x-axis
ax2.set_ylabel('Humidity',
color='tab:blue')  # we already handled the x-label with ax1
ax2.plot(df['Date'],
df['Humidity'], color='tab:blue', label='Humidity')
ax2.tick_params(axis='y',
labelcolor='tab:blue')

plt.title('Temperature and Humidity Over Time')

fig.tight_layout()  #
to make sure the labels don't overlap
plt.show()

"""15. Create a NumPy array
data containing 1000 samples from a normal distribution. Perform the following
tasks using
Matplotlib:

  a) Plot a histogram of the data with 30 bins.

  b) Overlay a line plot
representing the normal distribution's probability density function (PDF).

  c) Label the
x-axis as 'Value' and the y-axis as 'Frequency/Probability'.

  d) Set the title of the plot as
'Histogram with PDF Overlay
"""

from scipy.stats import
norm

np.random.seed(0)
data = np.random.normal(loc=0, scale=1, size=1000)

plt.hist(data,
bins=30, density=True, alpha=0.6, color='g', label='Histogram')

xmin, xmax = plt.xlim()
x =
np.linspace(xmin, xmax, 100)
p = norm.pdf(x, loc=0, scale=1)
plt.plot(x, p, 'k', linewidth=2,
label='PDF')

plt.xlabel('Value')
plt.ylabel('Frequency/Probability')

plt.title('Histogram
with PDF Overlay') # Set the title of the
plot


plt.legend()

plt.show()
```

```
"""16. Set the title of the plot as 'Histogram
with PDF Overlay"""

np.random.seed(0)
data = np.random.normal(loc=0, scale=1,
size=1000)

plt.hist(data, bins=30, density=True, alpha=0.6, color='g',
label='Histogram')

xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
p = norm.pdf(x,
loc=0, scale=1)
plt.plot(x, p, 'k', linewidth=2,
label='PDF')

plt.xlabel('Value')
plt.ylabel('Frequency/Probability')

plt.title('Histogram
with PDF Overlay') # Set the title of the
plot


plt.legend()

plt.show()

"""17. Create a Seaborn scatter plot of two
random arrays, color points based on their position relative to the origin (quadrants), add a
legend, label the axes, and set the title as 'Quadrant-wise Scatter
Plot'."""

import seaborn as sns

np.random.seed(0)
x = np.random.randn(100)
y =
np.random.randn(100)

def determine_quadrant(x, y):
    if x > 0 and y > 0:

return 'Q1'
    elif x < 0 and y > 0:
        return 'Q2'
    elif x < 0 and y <
0:
        return 'Q3'
    elif x > 0 and y < 0:
        return 'Q4'

data =
pd.DataFrame({'x': x, 'y': y})
data['quadrant'] = data.apply(lambda row:
determine_quadrant(row['x'], row['y']), axis=1)

plt.figure(figsize=(10, 6))
scatter_plot =
sns.scatterplot(data=data, x='x', y='y', hue='quadrant',
palette='viridis')

plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Quadrant-wise Scatter
Plot')

plt.legend(title='Quadrant')

plt.show()

"""17. With Bokeh, plot a line
chart of a sine wave function, add grid lines, label the axes, and set the title as 'Sine Wave
Function'."""
```

```
from bokeh.plotting import figure, show, output_notebook
from
bokeh.io import push_notebook

x = np.linspace(0, 4 * np.pi, 100)
y = np.sin(x)

p =
figure(title="Sine Wave Function", x_axis_label='X-axis',
y_axis_label='Y-axis')

p.line(x, y, legend_label="Sine Wave",
line_width=2)

p.xgrid.grid_line_color = 'gray'
p.ygrid.grid_line_color =
'gray'

output_notebook()
show(p, notebook_handle=True)

"""18. Using Bokeh,
ggenerate a bar chart of randomly generated categorical data, color bars based on their values,
add hover tooltips to display exact values, label the axes, and set the title as 'Random
Categorical Bar Chart'8"""

# Import necessary libraries
from bokeh.plotting
import figure, show, output_notebook
from bokeh.models import ColumnDataSource,
HoverTool
import random

# Generate random categorical data and their corresponding
values
categories = ['Category A', 'Category B', 'Category C', 'Category D', 'Category
E']
values = [random.randint(1, 10) for _ in range(len(categories))]

# Create a
ColumnDataSource
source = ColumnDataSource(data=dict(categories=categories, values=values))

#
Create a figure
p = figure(x_range=categories, height=350, title="Random Categorical Bar
Chart",
           toolbar_location=None, tools="")

# Add vertical
bars
p.vbar(x='categories', top='values', width=0.9, source=source,
       line_color='white',
fill_color='colors')

# Add hover tooltips to display exact values
hover =
HoverTool()
hover.tooltips = [("Value", "@values")]
p.add_tools(hover)

#
Label the axes
p.xaxis.axis_label = "Categories"
p.yaxis.axis_label =
"Values"

# Set the title
p.title.align = 'center'

# Generate random colors for
bars
colors = ['#' + ''.join([random.choice('0123456789ABCDEF') for j in range(6)]) for i in
range(len(categories))]
```

```
source.data['colors'] = colors

# Show the
plot
show(p)

"""20. Using Plotly, create a basic line plot of a randomly
generated dataset, label the axes, and set the title as 'Simple Line
Plot'"""

import plotly.graph_objects as go

x = np.linspace(0, 10, 100)
y =
np.random.rand(100)

fig = go.Figure()

fig.add_trace(go.Scatter(x=x, y=y, mode='lines',
name='Random Data'))

fig.update_layout(
    title='Simple Line Plot',

xaxis=dict(title='X-axis'),
    yaxis=dict(title='Y-axis')
)
fig.show()

"""21.
Using Plotly, create an interactive pie chart of randomly generated data, add labels and
percentages, set the title as 'Interactive Pie Chart'"""

categories =
['Category A', 'Category B', 'Category C', 'Category D']
values = np.random.randint(1, 100,
size=len(categories))

fig = go.Figure(data=[go.Pie(labels=categories, values=values,
textinfo='label+percent')])
fig.update_layout(title='Interactive Pie Chart')

fig.show()
```