Python-Module-5-Important-Topics

        - os.walk()
        - os.getcwd()
    - 6. What are the important characteristics of CSV file format.
        - What is CSV?
        - Characteristics of CSV File format
    - 7. Write the output of the following python code:
    - 8. What is the difference between loc and iloc in pandas DataFrame. Give a suitable example
        - Difference between loc and iloc
        - Example
            - Using loc
            - Using iloc
    - 9. Explain the attributes of an ndarray object.
- Python-Module-5-University-Questions-Part-B
    - 1. Explain how the matrix multiplications are done using numpy arrays.
    - 2. How to plot two or more lines on a same plot with suitable legends, labels and title.
    - 3. Consider a CSV file ‘employee.csv’ with the following columns(name, gender, startdate ,salary, team)
    - 4. Write Python program to write the data given below to a CSV file
    - 5. Consider the following two-dimensional array named arr2d
        - arr2d[:2]
        - arr2d[:2, 1:]
        - arr2d[1, :2]
        - arr2d[:2, 1:] = 0
    - 6. Write a Python program to add two matrices and also find the transpose of the resultant matrix.
    - 7. Given a file “auto.csv” of automobile data with the fields index, company,body-style, wheel- base, length, engine-type, num-of-cylinders, horsepower,average-mileage, and price, write Python codes using Pandas to
    - 8. Given the sales information of a company as CSV file with the following fields monthnumber, facecream, facewash, toothpaste, bathingsoap, shampoo, moisturizer, totalunits, totalprofit. Write Python codes to visualize the data as follows
    - 9. Write a code segment that prints the names of all of the items in the currentworking

directory.

- 10. Write a python program to create two numpy arrays of random integers between 0 and 20 of shape (3, 3) and perform matrix addition, multiplication and transpose of the product matrix.
- 11. Write Python program to write the data given below to a CSV file named student.csv
- 12. Consider the above student.csv file with fields Name, Branch, Year, CGPA .
- 13. Consider a CSV file 'weather.csv' with the following columns (date,temperature, humidity, windSpeed, precipitationType, place, weather {Rainy,Cloudy, Sunny}).

1. Os Module

OS Module in python provides functions for interacting with the OS

Creating a Directory

- We can create a new directory using `mkdir()` function from OS Module

```
import os
os.mkdir("d:\\tempdir")
```

Changing current working directory

- This is done using `chdir`

```
import os
os.chdir("d:\\tempdir")
```

How to know what is my current working directory?

- You can use `getcw()` to get the current working directory

Removing a directory

- The `rmdir()` function removes the specified directory

```
import os
os.rmdir('d:\\samplefolder')
```

List files and subdirectories

- The listdir() function returns the list of all files and directories in the specified directory
- If we dont specify any directory, then list of files an directories in the current working directory will be returned

2. Sys Module

- The sys module provides functions and variables used to manipulate different parts of the python runtime environment

sys.argv

- **Returns a list of command line arguments passed to a python script.** The item at index 0 in this list is always the name of the script. The rest of the arguments are stored at their subsequent indices

sys.exit

- This causes the script to exit back to either the Python console or the command prompt.
- This is used to safely exit from the program in the case of generation of an exception

sys.maxsize

- Returns the largest integer a variable can take

sys.path

- This is an environment variable that is a search path for all Python modules

sys.version

- This attribute displays a string containing the version number of the current python interpreter

3. Numpy

> **Note**
>
> You can try these numpy examples yourselves at
> https://www.w3schools.com/python/numpy/trypython.asp?filename=demo_numpy_editor

- Numpy is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed
- Using NumPy, a developer can perform the following operations
  - Mathematical and logical operations on arrays
  - Fourier transforms and routines for shape manipulation
  - Operations related to linear algebra. NumPy has in-

Difference between python list and a NumPy array

- NumPy gives you an enormous range of fast and efficient ways of creating arrays and manipulating numerical data inside them
- While a python list can contain different data types within a single list, all the elements in a NumPy array should be homogeneous
- The mathematical operations that are meant to be performed on arrays would be extremely inefficient if arrays werent homogeneous

ndarray Object

- The most important object defined in NumPy is an N-dimensional array time called ndarray.

Features of ndarray

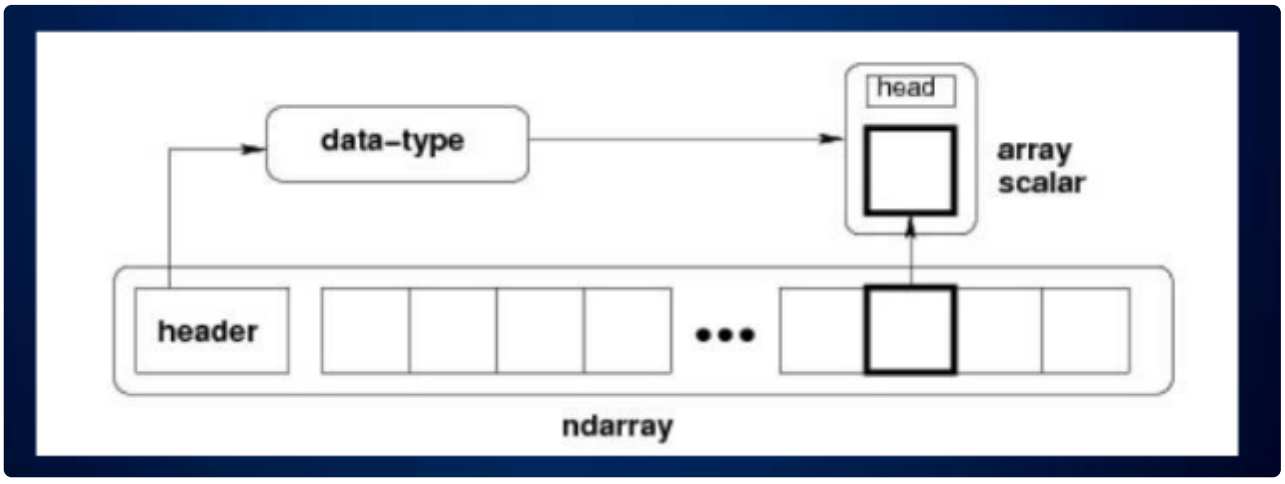
- **N-dimensional Array:** The `ndarray` is like a grid or table of numbers. This grid can have any number of dimensions. For example, a 1-dimensional array (like a list), a 2-dimensional array (like a table or matrix), or even higher dimensions.
- **Collection of Same Type:** All the items (elements) in this array are of the same type, like all integers or all floats. This makes operations on the array very fast.
- **Zero-based Indexing:** You can access each item in the array using an index that starts from 0. For example, `array[0]` gives you the first element.

- **Consistent Memory Size:** Each item in the array takes up the same amount of memory space. This is different from regular Python lists, where each element can be of different sizes.
- **Data-type Object (dtype):** Each element in the ndarray is an object of data-type object (called dype)
- **Slicing and Array Scalars:** When you extract a part of the array (called slicing), the extracted items are represented as Python objects of specific types known as **array scalars**.

Simple example

```
import numpy as np

# Creating an ndarray
array = np.array([1, 2, 3, 4])

# Accessing the first element (zero-based index)
print(array[0])  # Output: 1

# Checking the data type
print(array.dtype)  # Output: int64 (or int32 depending on your system)

# Slicing the array
slice = array[1:3]
print(slice)  # Output: [2 3]
```

Also check this diagram on how ndarray, dtype and array scalar type are related



- Here you can see a large rectangle on the bottom

    - This is ndarray
    - This ndarray contains a header
        - Header contains the datatype for the entire array
- There is smaller squares inside the ndarray
    - These small squares are array scalars, which are inside the ndarray

Example - Creating Arrays

Program

```
import numpy as np
a = np.array([1,2,3,4])
print("Value of a is\n",a,"\n") # Gives output [1 2 3 4]

b = np.array([(1,2,3),(4,5,6)], dtype = float)
print("Value of b is\n",b,"\n")

c = np.array([(1,2,3),(4,5,6),(7,8,9)])
print("Value of c is\n",c,"\n")
```

Output

```
Value of a is
 [1 2 3 4]

Value of b is
 [[1. 2. 3.]
 [4. 5. 6.]]

Value of c is
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
```

ndarray Object - Parameters

- **ndarray.ndim**
    - ndim represented the number of dimensions(axes) of the ndarray

    - a = np.array([1,2,3,4])
        - This is a 1D array, so ndim gives 1
    - b = np.array([(1,2,3),(4,5,6)])
        - This is a 2D array, so ndim gives b
- **ndarray.shape**
    - shape is a tuple of integers representing the size of ndarray in each dimension
    - If the array is 3x3
    - ndarray.shape gives (3,3)
- **ndarray.size**
    - Total number of elements
    - Product of elements in shape
    - if shape is (3,3) then size is 3x3 = 9
- **ndarray.dtype**
    - Data type of elements of numpy array
- **ndarray.itemsize**
    - Returns size of each element of a numpy array

Example - ndarray Object Parameters

Program

```
import numpy as np
a = np.array([[[1,2,3],[4,3,5]],[[3,6,7],[2,1,0]]])
print("The dimension of array a is:",a.ndim)
print("The size of array a is:",a.shape)
print("The total no of elements in array a is:",a.size)
print("The datatype of elements in array a is:",a.dtype)
print("The size of each element in array a is:",a.itemsize)
```

Output

```
The dimension of array a is: 3
The size of array a is: (2, 2, 3)
The total no of elements in array a is: 12
The datatype of elements in array a is: int64
The size of each element in array a is: 8
```

Arithmetic operations with NumPy Array

- The arithmetic operations with NumPy arrays perform element wise operations. This means the operators are applied only between corresponding elements
- Arithmetic operations are possible only if the array has the same structure and dimensions

Basic Operations with Scalars

Program

```
import numpy as np
a = np.array([1,2,3,4,5])
b = a + 1
print(b)
c = 2**a
print(c)
```

Output

```
[2 3 4 5 6]
[ 2  4  8 16 32]
```

- In the first one, Each element is added by 1
- For the second one, 2^e where e is each element

Arithmetic Operators in numpy

Program

```
import numpy as np
a = np.array([7,3,4,5,1])
b = np.array([3,4,5,6,7])

print(a+b)
print(np.add(a,b))
print("----------")

print(a-b)
print(np.subtract(a,b))
print("----------")
```

```
print(a*b)
print(np.multiply(a,b))
print("-----------")

print(a/b)
print(np.divide(a,b))
print("-----------")

print(np.remainder(a,b))
print("-----------")

print(np.mod(a,b))
print("-----------")

print(np.power(a,b))
print("-----------")

print(np.reciprocal(a,b))
print("-----------")
```

Output

```
[10  7  9 11  8]
[10  7  9 11  8]
----------
[ 4 -1 -1 -1 -6]
[ 4 -1 -1 -1 -6]
----------
[21 12 20 30  7]
[21 12 20 30  7]
-----------
[2.33333333 0.75       0.8        0.83333333 0.14285714]
[2.33333333 0.75       0.8        0.83333333 0.14285714]
-----------
[1 3 4 5 1]
-----------
[1 3 4 5 1]
-----------
```

```
[  343    81  1024 15625     1]
-----------
[0 0 0 0 1]
-----------
```

- Here we are using add,subtract, multiply and divide etc
- we can use either the symbol or name
    - either + or add can be used

Trignometry operations with NumPy Array

- **np.sin()**
- **np.cos()**
- **np.tan()**

Comparison in NumPy

- We can use == operator to check if they are equal
- numpy.greater(x1,x2)
- numpy.greater_equal(x1,x2)
- numpy.less(x1,x2)
- numpy.less_equal(x1,x2)

4. Pandas

> **Note**
>
> You can try these pandas examples yourselves at https://www.w3schools.com/python/pandas/trypandas.asp?filename=demo_pandas_editor

What is Pandas?

- Pandas is an open-source library that is built on top of NumPy library. It is a Python package that offers various data structures and operations for manipulating numerical data and time series.

- It is mainly popular for importing and analyzing data much easier. Pandas is fast and it has high-performance and productivity for users.

Advantages

- Fast and efficient for manipulating and analyzing data.
- Data from different file objects can be loaded.
- Easy handling of missing data (represented as NaN)

Series

- Pandas Series is a one-dimensional labeled array capable of holding data of any type (integer, string, float, python objects, etc.)
- The axis labels are collectively called index. Pandas Series is nothing but a column in an excel sheet.
- We can form a simple series using an array of data

Example of Series

Program-1 (Simple Series)

```
import pandas as pd
obj = pd.Series([3,5,-8,7,9])
print(obj)
```

Output

```
0    3
1    5
2   -8
3    7
4    9
dtype: int64
```

- Here the index values are 0,1,2,3,4 for the Values in the series

Program-2 (Series with custom index)

- We can also put custom index values

```
import pandas as pd
obj = pd.Series([3,5,-8,7,9],index=['d','b','a','c','e'])

print(obj)
```

Output

```
d    3
b    5
a   -8
c    7
e    9
dtype: int64
```

Dataframe

What is Dataframe?

- **Pandas DataFrame** is two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns)
  - A **Pandas DataFrame** is like a table of data.
  - It has rows and columns.
  - You can change its size by adding or removing rows and columns.
  - It can hold different types of data in different columns (e.g., numbers, text).
- Pandas DataFrame consists of three principal components, the data, rows, and columns.

Basic operations of Dataframe

- Creating a DataFrame
- Dealing with Rows and Columns
- Indexing and Selecting Data
- Working with Missing Data
- Iterating over rows and columns

Creating a DataFrame

Example-1

```
import pandas as pd
lst = ['mec','minor','stud','eee','bio']
df = pd.DataFrame(lst)
```

Output

```
       0
0    mec
1  minor
2   stud
3    eee
4    bio
```

- Here the values are occupying column 0
- Lets see how to do it for multiple columns

  Example-2

```
import pandas as pd
lst = {
    'Column 0': ['mec', 'minor', 'stud', 'eee', 'bio'],
    'Column 1': ['data1', 'data2', 'data3', 'data4', 'data5']
}
df = pd.DataFrame(lst)
```

Output

```
  Column 0 Column 1
0      mec    data1
1    minor    data2
2     stud    data3
3      eee    data4
4      bio    data5
```

Example-3

- Lets make a table with name and age

```
import pandas as pd
# initialise data of lists.
data = {'Name':['Tom', 'nick', 'krish', 'jack'], 'Age':[20, 21, 19, 18]}
# Create DataFrame
df = pd.DataFrame(data)
# Print the output.
print(df)
```

Output

```
    Name  Age
0    Tom   20
1   nick   21
2  krish   19
3   jack   18
```

5. Matplotlib

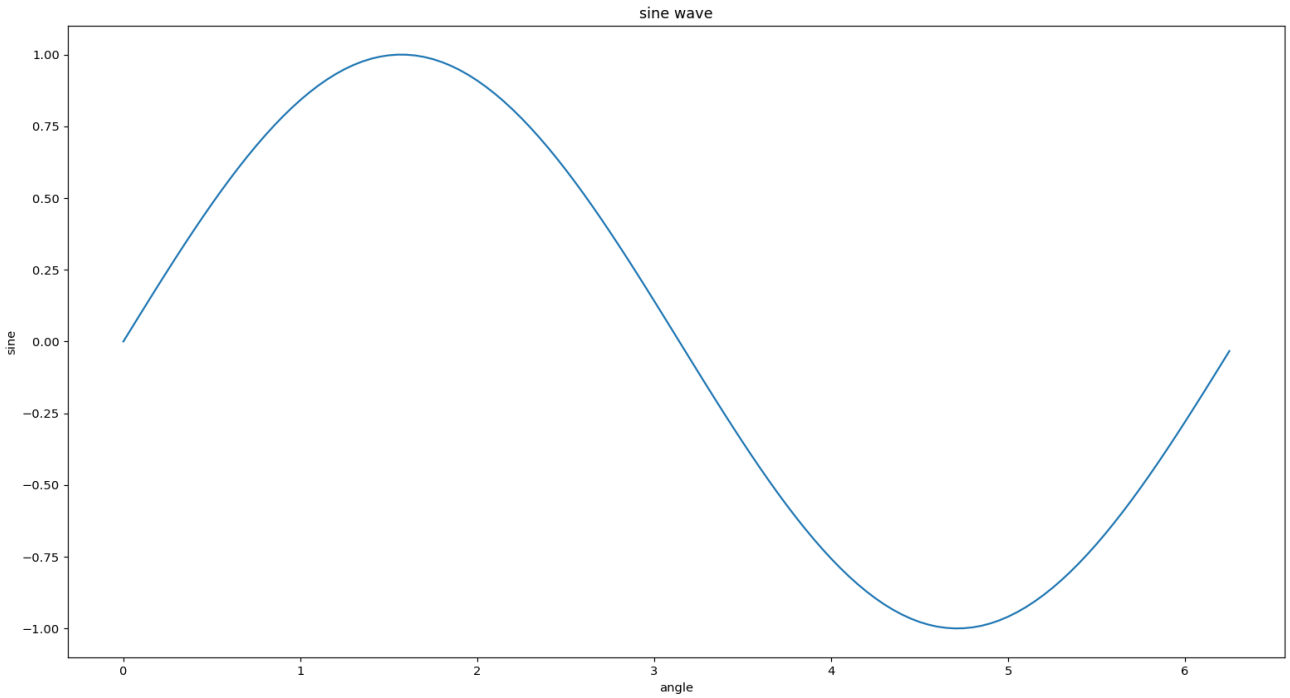
What is Matplotlib?

- Matplotlib is one of the most popular Python packages used for data visualization
- It is a cross-platform library for making 2D plots from data in arrays. Matplotlib is written in Python and makes use of NumPy.

Example - 1 -Sin Wave

```
from matplotlib import pyplot as plt
import numpy as np
import math
x = np.arange(0,math.pi*2,0.05)
y = np.sin(x)
plt.plot(x,y)
plt.xlabel("angle")
plt.ylabel("sine")
plt.title("sine wave")
plt.show()
```

Output



- 1. To begin with, the Pyplot module from Matplotlib package is imported
  - import matplotlib.pyplot as plt
- 2. Next we need an array of numbers to plot.
  - `import numpy as np`
  - `import math`
  - `x = np.arange(0, math.pi * 2, 0.05)`
    - **np.arange:** This is a function from the NumPy library that generates an array of evenly spaced values within a given range.
    - **0:** The starting value of the range (inclusive).
    - **math.pi * 2:** The end value of the range (exclusive). This calculates 2π2π, which is approximately 6.2832.
    - **0.05:** The step size, which determines the spacing between values in the array.
- 3. The ndarray object serves as values on x axis of the graph. The corresponding sine values of angles in x to be displayed on y axis are obtained by the following statement
  - `y = np.sin(x)`
- 4. The values from two arrays are plotted using the plot() function.
  - plt.plot(x,y)

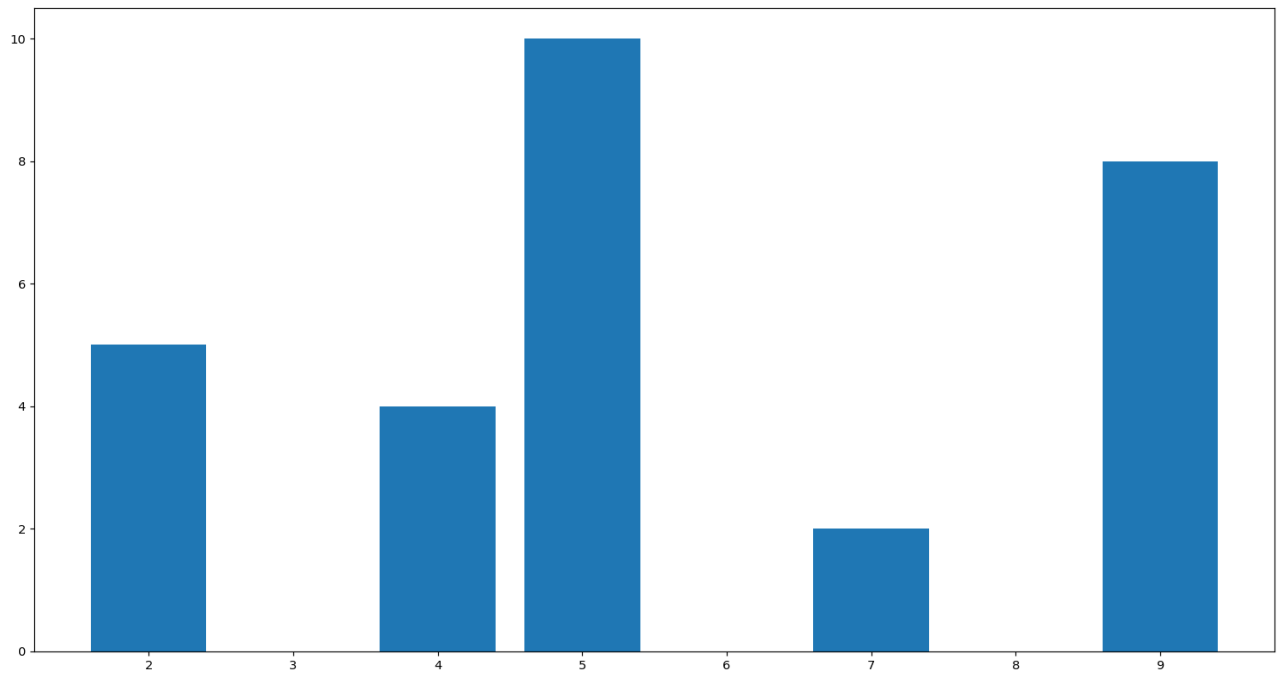
- 5. You can set the plot title, and labels for x and y axes.You can set the plot title, and labels for x and y axes.
  - plt.xlabel("angle")
  - plt.ylabel("sine")
  - plt.title('sine wave')
- 6. The Plot viewer window is invoked by the show() function
  - plt.show()

Example - 2 - Creating a bar plot

Program

```
from matplotlib import pyplot as plt
x = [5, 2, 9, 4, 7]
y = [10, 5, 8, 4, 2]
# Function to plot the bar
plt.bar(x,y)
# function to show the plot
plt.show()
```

Output



Pylab

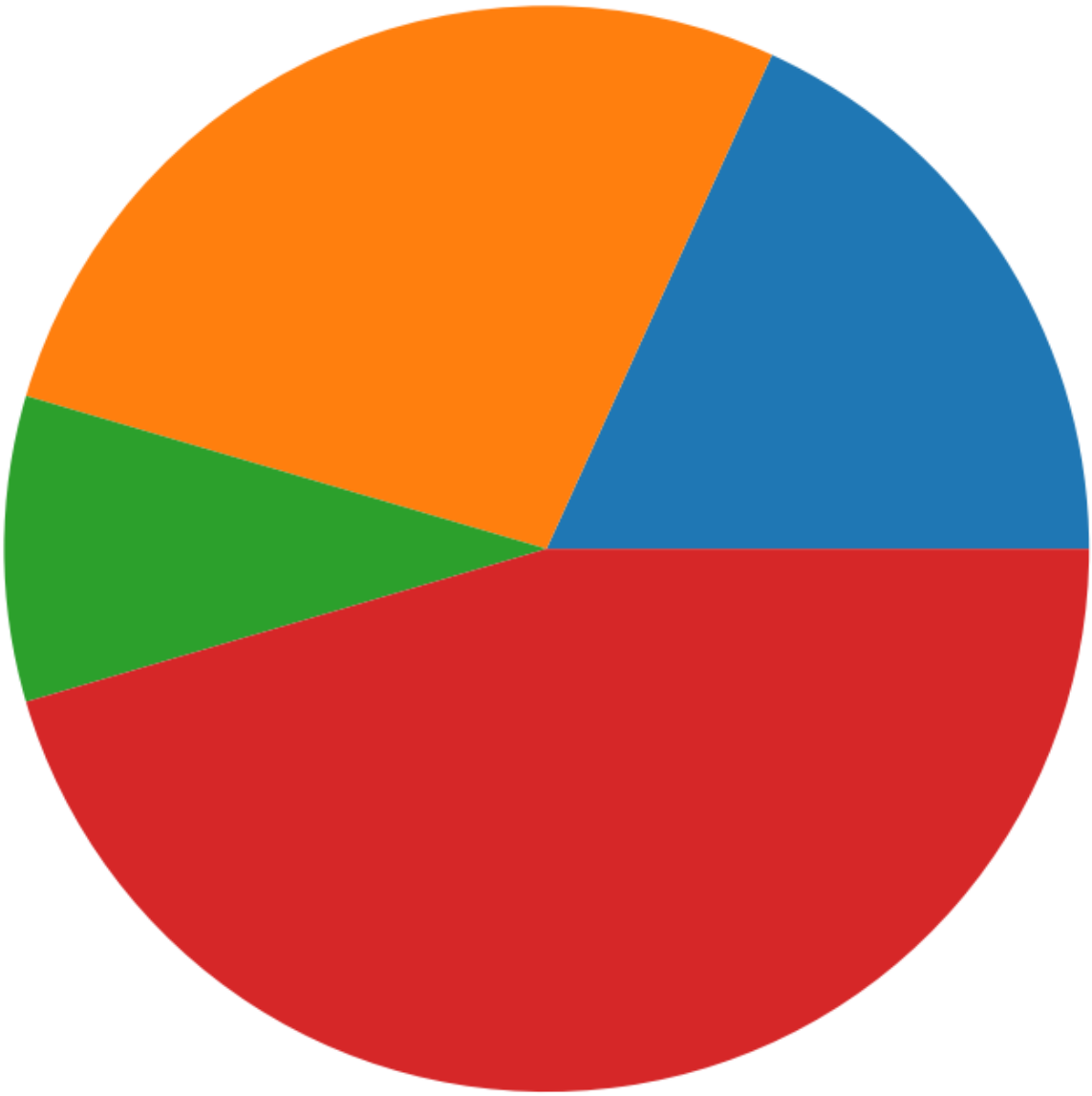
PyLab is a convenience module that bulk imports matplotlib.pyplot (for plotting) and NumPy (for Mathematics and working with arrays) in a single name space.

Example - 3 - Creating a Pieplot

Program

```
data=[20,30,10,50]
from pylab import *
pie(data)
show()
```

Output



Python-Module-5-University-Questions-Part-A

1. How do you assign a random number to a variable in Python?

- To assign a random number to a variable in Python, you can use the `random` module, which provides various methods for generating random numbers.

```
import random

random_num = random.random()
print(random_num)
```

2. What is the use of os module in python?

The `os` module in Python provides functions to interact with the operating system. It allows you to Create, delete, rename, and list files and directories.

3. Write a Python code that checks to see, if a file with the given pathname exists on the disk, before attempting to open a file for input

```
import os

# Function to check if the file exists and then open it
def open_file_if_exists(filepath):
    if os.path.isfile(filepath):
        with open(filepath, 'r') as file:
            content = file.read()
            print("File content:\n", content)
    else:
        print("File does not exist.")

# Example usage
filepath = 'example.txt'
open_file_if_exists(filepath)
```

*4. What is Flask in Python?

- Flask is a lightweight and flexible web framework for Python. It’s designed to make getting started with web development quick and easy, while still being powerful enough to build complex web applications.

5. Explain the os and os.path modules in Python with examples. Also, discuss the walk( ) and getcwd( ) methods of the os module

What is os and os.path?

- The `os` module in Python provides a way to interact with the operating system, offering various functions for file and directory manipulation, process management etc.
- The `os.path` module, which is part of `os`, provides functions for manipulating file and directory paths.

Example of os

```
import os
os.mkdir('new_directory')  # Create a new directory
os.rename('old_name.txt', 'new_name.txt')  # Rename a file
os.remove('file_to_delete.txt')  # Remove a file
os.rmdir('directory_to_delete')  # Remove a directory
```

Example of os.path

``

```
path = os.path.join('directory', 'file.txt')  # Join paths
exists = os.path.exists('file.txt')  # Check if file exists
is_dir = os.path.isdir('directory')  # Check if it is a directory
```

os.walk()

- The `os.walk()` method generates the file names in a directory tree by walking either top-down or bottom-up through the directory.

```
import os

for dirpath, dirnames, filenames in os.walk('path/to/directory'):
    print(f"Directory: {dirpath}")
    for dirname in dirnames:
        print(f"Subdirectory: {dirname}")
    for filename in filenames:
        print(f"File: {filename}")
```

os.getcwd()

- The `os.getcwd()` method returns the current working directory.

```
import os

current_directory = os.getcwd()
print(f"Current Working Directory: {current_directory}")
```

6. What are the important characteristics of CSV file format.

What is CSV?

- CSV is a data format that has fields/columns separated by the comma character and records/rows terminated by newlines
- Example of a csv file

```
ID,Name,Age,City
1,John Doe,28,New York
2,Jane Smith,34,Los Angeles
3,Emily Jones,22,Chicago
4,Michael Brown,45,Houston
5,Sarah Davis,29,Miami
```

- Here first line is the column name
- The remaining lines are the rows

Characteristics of CSV File format

- One line for each record
- Comma separated fields
- Space-characters adjacent to commas are ignored

7. Write the output of the following python code:

```
import numpy as np
arr1 = np.arange(6).reshape((3, 2))
arr2 = np.arange(6).reshape((3,2))
arr3 = arr1 + arr2[0].reshape((1, 2))
print(arr3)
```

1. **Importing numpy:**
    1. import numpy as np
2. **Creating the first array (`arr1`):**
    1. arr1 = np.arange(6).reshape((3, 2))
    2. `np.arange(6)` generates an array with values `[0, 1, 2, 3, 4, 5]`
    3. `.reshape((3, 2))` reshapes this array into a 3x2 array:
    4. So the arr1 variable will have 3 rows and 2 columns

```
arr1 = [[0, 1],
        [2, 3],
        [4, 5]]
```

3. **Creating the second array (`arr2`):**
    1. arr2 = np.arange(6).reshape((3, 2))

```
arr2 = [[0, 1],
        [2, 3],
```

```
        [4, 5]]
```

4. **Adding the first row of `arr2` to `arr1`:**
   1. arr3 = arr1 + arr2[0].reshape((1, 2))
   2. `arr2[0]` selects the first row of `arr2`, which is `[0, 1]`.
   3. `.reshape((1, 2))` reshapes this into a 1x2 array:
   4. so arr2[0] will become 0, 1

   Adding this 1x2 array to each row of `arr1` results in element-wise addition:

```
arr3 = [[0 + 0, 1 + 1],
        [2 + 0, 3 + 1],
        [4 + 0, 5 + 1]]
     = [[0, 2],
        [2, 4],
        [4, 6]]
```

5. **Printing the resulting array (`arr3`):**
   1. print(arr3)
6. The output of the code is:

```
[[0 2]
 [2 4]
 [4 6]]
```

8. What is the difference between loc and iloc in pandas DataFrame. Give a suitable example

Difference between loc and iloc

- In Pandas, `loc` and `iloc` are both methods used for indexing and selecting data in a DataFrame
- `loc`: It is used to select data by label. When you use `loc`, you specify rows and columns based on their labels. This means you refer to the index labels and column names to retrieve data

- `iloc`: It is used to select data by integer location. When you use `iloc`, you specify rows and columns based on their integer positions, starting from 0. This means you refer to the numerical index positions to retrieve data.

Example

Im creating a dataframe

```
import pandas as pd

# Creating a sample DataFrame
data = {'A': [1, 2, 3, 4, 5],
        'B': ['a', 'b', 'c', 'd', 'e']}
df = pd.DataFrame(data, index=['row1', 'row2', 'row3', 'row4', 'row5'])
```

The dataframe will look like this

```
      A  B
row1  1  a
row2  2  b
row3  3  c
row4  4  d
row5  5  e
```

- Here row1,row2,etc are the index, We cane use this index to access values inside the table

Using loc

- Suppose i want to print the content inside `row2`
- In this case i need to use `loc`

```
print("Using loc:")
print(df.loc['row2'])  # Selecting row with label 'row2'
print(df.loc['row2', 'B'])  # Selecting value at row 'row2' and column 'B'
```

- This will give this output

```
Printing content of row2
A    2
B    b
Name: row2, dtype: object
Printing content inside row2 and column B
b
```

Using iloc

- Suppose i want to access the 0th row and 1st row, without using the index name
- For that we will be using `iloc`

```
print("Printing content of 0th row")
print(df.iloc[0])  # Selecting row at integer position 1 (second row)

print("Printing content of 0th row and 0th column")
print(df.iloc[0,0])  # Selecting row at integer position 1 (second row)

print("Printing content of 1st row")
print(df.iloc[1])  # Selecting row at integer position 1 (second row)

print("Printing content of 1st row and 1st column")
print(df.iloc[1, 1])  # Selecting value at row 1 and column 1 (second row,
second column)
```

```
Printing content of 0th row
A    1
B    a
Name: row1, dtype: object
Printing content of 0th row and 0th column
1
Printing content of 1st row
A    2
B    b
Name: row2, dtype: object
```

```
Printing content of 1st row and 1st column
b
```

9. Explain the attributes of an ndarray object.

- **ndarray.ndim**
  - ndim represented the number of dimensions(axes) of the ndarray
  - a = np.array([1,2,3,4])
    - This is a 1D array, so ndim gives 1
  - b = np.array([(1,2,3),(4,5,6)])
    - This is a 2D array, so ndim gives b
- **ndarray.shape**
  - shape is a tuple of integers representing the size of ndarray in each dimension
  - If the array is 3x3
  - ndarray.shape gives (3,3)
- **ndarray.size**
  - Total number of elements
  - Product of elements in shape
  - if shape is (3,3) then size is 3x3 = 9
- **ndarray.dtype**
  - Data type of elements of numpy array
- **ndarray.itemsize**
  - Returns size of each element of a numpy array

Python-Module-5-University-Questions-Part-B

1. Explain how the matrix multiplications are done using numpy arrays.

- Matrix multiplication, also called the matrix dot product.

- The rule for matrix multiplication is as follows:
    - The number of columns (n) in the first matrix (A) must equal the number of rows (m) in the second matrix (B).

```
from numpy import array
# define first matrix
A = array([[1, 2],[3, 4],[5, 6]]
print(A)
# define second matrix
B = array([[1, 2],[3, 4]])
print(B)
# multiply matrices
C = A.dot(B)
print(C)
```

- Here Number of columns of A = 2
- Number of rows of B = 2

The output of the program is

```
[[1 2]
 [3 4]
 [5 6]]
[[1 2]
 [3 4]]
[[ 7 10]
 [15 22]
 [23 34]]
```

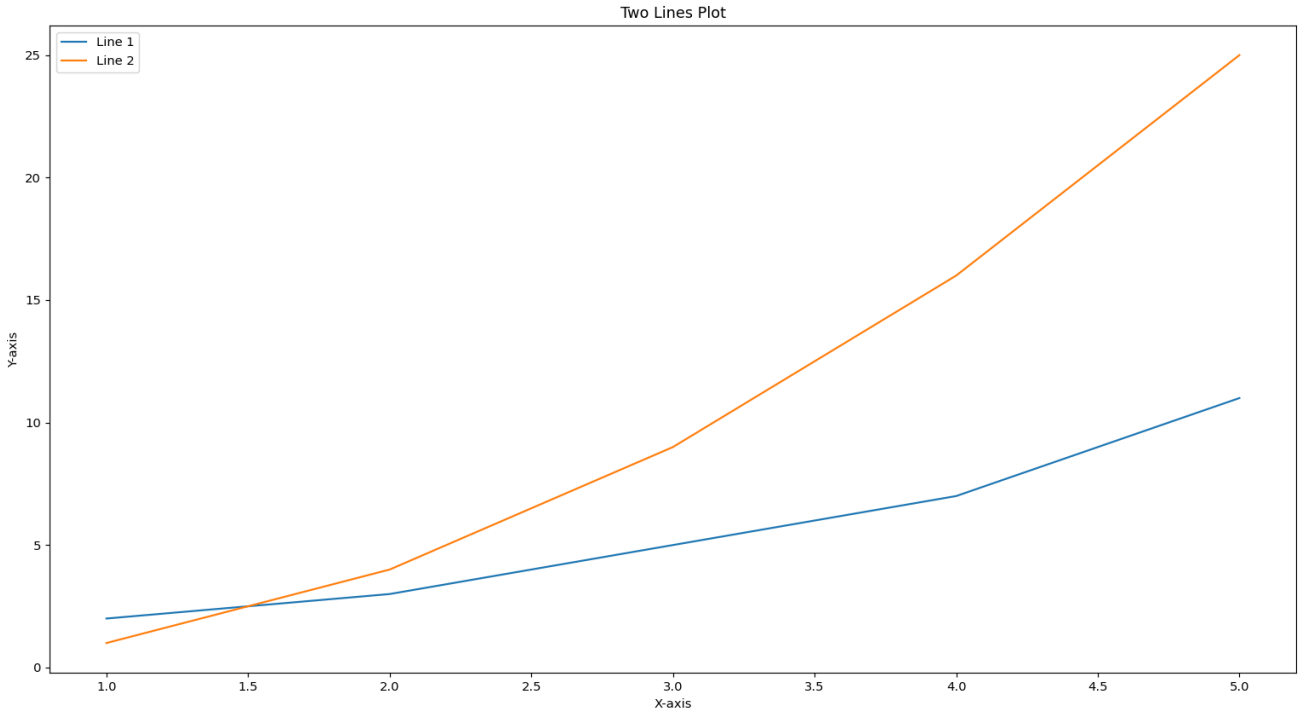
2. How to plot two or more lines on a same plot with suitable legends, labels and title.

```
from matplotlib import pyplot as plt


# Create data for the lines
```

```
x = [1, 2, 3, 4, 5]
y1 = [2, 3, 5, 7, 11]
y2 = [1, 4, 9, 16, 25]

# Plot each line with labels
plt.plot(x, y1, label='Line 1')
plt.plot(x, y2, label='Line 2')


plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Two Lines Plot')

# Add legend
plt.legend()

# Show the plot
plt.show()
```

- Here We plot 2 lines using plt.plot
- We set the label using plt.xlabel, plt.ylabel
- We set the title using plt.title
- Legend is set using plt.legend(), and the label attribute in plt.plot()

The output is



3. Consider a CSV file ‘employee.csv’ with the following columns(name, gender, start_date ,salary, team)

- **Write commands to do the following using panda library.**
  - **1. print first 7 records from employees file**
  - **2. print all employee names in alphabetical order**
  - **3. find the name of the employee with highest salary**
  - **4. list the names of male employees**
  - **5. Display to which all teams employees belong**

1. **First we import pandas**
    1. import pandas as pd
2. **Now lets read the employee.csv file**
    1. For that we will use `pd.readcsv`
    2. `data = pd.read_csv("employee.csv") # Read the CSV file`
3. **The question 1 says to print first 7 records**
    1. For that we will use .head
    2. `data.head(7)`

4. **Question 2 says to print all the employee names in alphabetical order**
    3. `print(data.sort_values(by="name")["name"]) # Sort by name and print the "name" column`
5. **Question 3 says to FInd the name of the employee with the highest salary**
    1. highest_paid_employee = data.loc[data["salary"].idxmax(), "name"]
    2. It uses the .idxmax() function
    3. `.idxmax()` returns the index label (row number) where the maximum value in the "salary" column is located.
    4. So here row = row number with highest salary
    5. Column = name
6. Question 4 says to list the name of male employees
    1. `male_employees = data[data["gender"] == "M"]["Name"]`
7. Question 5 says to display the teams
    1. `unique_teams = data["team"].unique()`

The entire code will look like this

```
import pandas as pd

# Sample CSV file (replace 'employee.csv' with your actual file path)
data = pd.read_csv("employee.csv")

# 1. Print first 7 records
print("First 7 records:")
print(data.head(7))

# 2. Print employee names in alphabetical order
print("\nEmployee names (alphabetical):")
print(data.sort_values(by="name")["name"])

# 3. Find employee with highest salary
highest_paid_employee = data.loc[data["salary"].idxmax(), "name"]
print("\nEmployee with highest salary:", highest_paid_employee)

# 4. List names of male employees
male_employees = data[data["gender"] == "M"]["name"]
print("\nMale employees:", male_employees.tolist())
```

```
# 5. List all teams
unique_teams = data["team"].unique()
print("\nTeams:", unique_teams.tolist())
```

employee.csv file

```
name,gender,start_date,salary,team
Alice,F,2023-01-01,50000,Marketing
Bob,M,2022-05-15,72000,Engineering
Charlie,M,2024-02-10,48000,Sales
David,M,2021-12-25,65000,Marketing
Emily,F,2023-07-09,38000,Finance
Frank,M,2022-09-22,80000,Engineering
Grace,F,2024-03-14,42000,Sales
```

Output

```
First 7 records:
      name gender  start_date  salary         team
0    Alice      F  2023-01-01   50000    Marketing
1      Bob      M  2022-05-15   72000  Engineering
2  Charlie      M  2024-02-10   48000        Sales
3    David      M  2021-12-25   65000    Marketing
4    Emily      F  2023-07-09   38000      Finance
5    Frank      M  2022-09-22   80000  Engineering
6    Grace      F  2024-03-14   42000        Sales

Employee names (alphabetical):
0      Alice
1        Bob
2    Charlie
3      David
4      Emily
5      Frank
6      Grace
Name: name, dtype: object
```

```
Employee with highest salary: Frank

Male employees: ['Bob', 'Charlie', 'David', 'Frank']

Teams: ['Marketing', 'Engineering', 'Sales', 'Finance']
```

*4. Write Python program to write the data given below to a CSV file**

*Reg_no Name Sub_Mark1 Sub_Mark2 Sub_Mark3
10001 Jack 76 88 76
10002 John 77 84 79
10003 Alex 74 79 81

```python
import csv

data = [
    ["Reg_no", "Name", "Sub_Mark1", "Sub_Mark2", "Sub_Mark3"],
    [10001, "Jack", 76, 88, 76],
    [10002, "John", 77, 84, 79],
    [10003, "Alex", 74, 79, 81],
]

with open("student_data.csv", "w", newline="") as csvfile:
    writer = csv.writer(csvfile)
    writer.writerows(data)

print("Data written to student_data.csv successfully!")
```

- The `csv.writer` function creates a writer object that helps write data to the CSV file.
- **Writes data to the CSV file:** The `writer.writerows` method writes all rows from the `data` list to the CSV file.

5. Consider the following two-dimensional array named arr2d

```
[[1, 2, 3],
[4, 5, 6],
[7, 8, 9]]
```

Write the output of following Python Numpy expressions:

1. **arr2d[:2]**
2. **arr2d[:2, 1:]**
3. **arr2d[1, :2]**
4. **arr2d[:2, 1:] = 0**

arr2d[:2]

```
[[1 2 3] [4 5 6]]
```

- Slicing with `[:2]` selects all rows from the beginning (`0`) up to, but not including, index `2`.
- So, it extracts the first two rows (index 0 and 1) of the original array and returns a new 2D array containing those rows.

arr2d[:2, 1:]

```
[[2 3] [5 6]]
```

- Slicing with `[:2, 1:]` selects elements based on both rows and columns.
  - `[:2]` selects rows as explained in example 1.
  - `, 1:]` selects columns starting from index `1` (the second column) up to the end for each row included in the first selection.
- Therefore, it extracts elements from the second column (index 1) onwards for the first two rows (0 and 1) and returns a new 2D array with those elements.

arr2d[1, :2]

```
[4 5]
```

- Slicing with `[1, :2]` selects elements from a specific row and columns.
  - `[1]` selects the second row (index 1) of the array.
  - `, :2` selects columns from the beginning (`0`) up to, but not including, index `2` (i.e., the first two columns).
- This extracts elements from the first two columns (0 and 1) of the second row (index 1) and returns a new 1D array (row vector) containing those elements.

arr2d[:2, 1:] = 0

```
[[1 0 0] [4 0 0] [7 8 9]]
```

- `[:2, 1:]` selects elements from the second column onwards for the first two rows.
- Assigning `0` to this selection replaces those elements with zeros, effectively modifying the original `arr2d` array.

6. Write a Python program to add two matrices and also find the transpose of the resultant matrix.

```python
import numpy as np

# Define two matrices
matrix1 = np.array([[1, 2, 3], [4, 5, 6]])
matrix2 = np.array([[7, 8, 9], [10, 11, 12]])

# Add the matrices
sum_matrix = matrix1 + matrix2
print("Sum of matrices:\n", sum_matrix)

# Find the transpose of the resultant matrix
transpose_sum = sum_matrix.T
print("\nTranspose of the sum:\n", transpose_sum)
```

Output

```
Sum of matrices:
 [[ 8 10 12]
 [14 16 18]]

Transpose of the sum:
 [[ 8 14]
 [10 16]
 [12 18]]
```

7. Given a file “auto.csv” of automobile data with the fields index, company,body-style, wheel- base, length, engine-type, num-of-cylinders, horsepower,average-mileage, and price, write Python codes using Pandas to

1. **Print total cars of all companies**
2. **Find the average mileage of all companies**
3. **Find the highest priced car of all companies**

Question 1

```python
# 1. Print total cars of all companies
data = pd.read_csv("auto.csv")
total_cars = len(data)
print(f"Total Cars: {total_cars}")
```

Question 2

```python
average_mileage = data["average-mileage"].mean()
print(f"Average Mileage: {average_mileage} MPG")
```

Question 3

```
highest_priced_car = data[data["price"] == data["price"].max()]
print("Highest Priced Car:") print(highest_priced_car)
```

Example- auto.csv

```
index,company,body-style,wheel-base,length,engine-type,num-of-
cylinders,horsepower,average-mileage,price
1,Chevrolet,Wagon,98.6,190.9,2.5L,4,98,25,8145
2,Chevrolet,Minivan,97.0,186.6,3.0L,4,161,19,10095
3,Dodge,Sedan,96.8,190.0,2.3L,4,100,21,7875
4,Dodge,Sedan,96.8,190.0,2.0L,4,130,23,8845
5,Plymouth,Minivan,95.2,187.1,2.4L,4,100,18,8845
6,Ford,Sedan,97.5,190.0,2.0L,4,120,21,7195
7,Ford,Sedan,97.5,190.0,2.0L,4,100,25,7595
8,Ford,Wagon,98.8,195.0,2.3L,4,120,20,8575
9,Mercury,Sedan,97.5,190.0,2.0L,4,120,21,7195
10,Mercury,Sedan,97.5,190.0,2.0L,4,100,25,7595
```

Output

```
Total Cars: 10
Average Mileage: 21.80 MPG
Highest Priced Car:
   index    company body-style  wheel-base  length engine-type  num-of-
cylinders  horsepower  average-mileage  price
1      2  Chevrolet    Minivan        97.0   186.6        3.0L
4         161               19  10095
```

8. Given the sales information of a company as CSV file with the following fields month_number, facecream, facewash, toothpaste, bathingsoap, shampoo, moisturizer, total_units, total_profit. Write Python codes to visualize the data as follows

1. **Toothpaste sales data of each month and show it using a scatter plot**
2. **Face cream and face wash product sales data and show it using the bar chart**

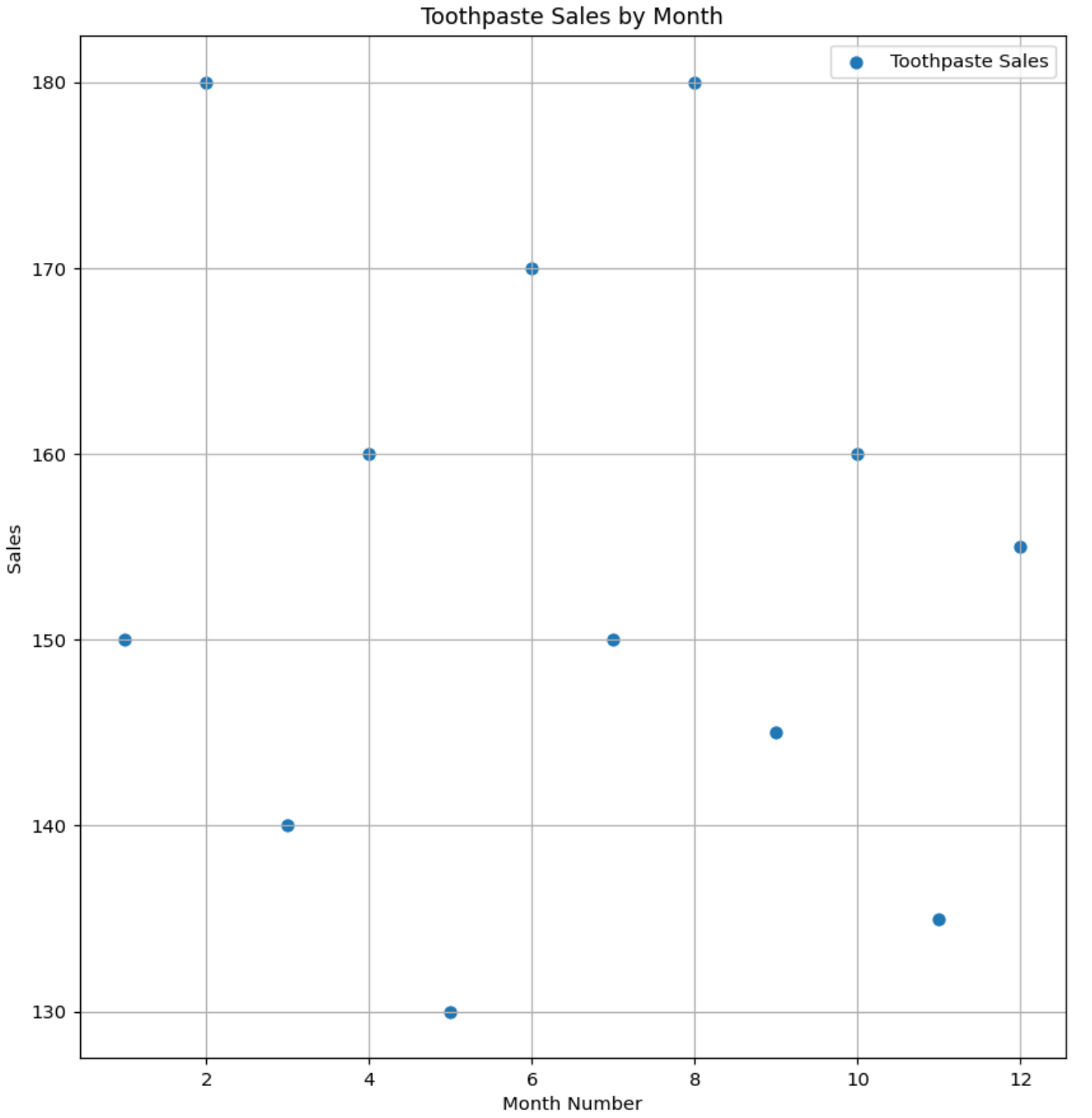
```
import pandas as pd
import matplotlib.pyplot as plt

# Read the CSV file
data = pd.read_csv("sales_data.csv")

# 1. Toothpaste sales scatter plot
plt.figure(figsize=(8, 6))  # Adjust figure size as needed
plt.scatter(data["month_number"], data["toothpaste"], label="Toothpaste
Sales")
plt.xlabel("Month Number")
plt.ylabel("Sales")
plt.title("Toothpaste Sales by Month")
plt.grid(True)
plt.legend()
plt.show()

# 2. Face cream and face wash bar chart
plt.figure(figsize=(8, 6))  # Adjust figure size as needed
face_cream_sales = data["facecream"]
face_wash_sales = data["facewash"]
product_labels = ["Face Cream", "Face Wash"]
plt.bar(product_labels, [face_cream_sales.sum(), face_wash_sales.sum()])
plt.xlabel("Product")
plt.ylabel("Sales")
plt.title("Face Cream vs Face Wash Sales")
plt.grid(axis="y")  # Grid on y-axis only
plt.show()
```

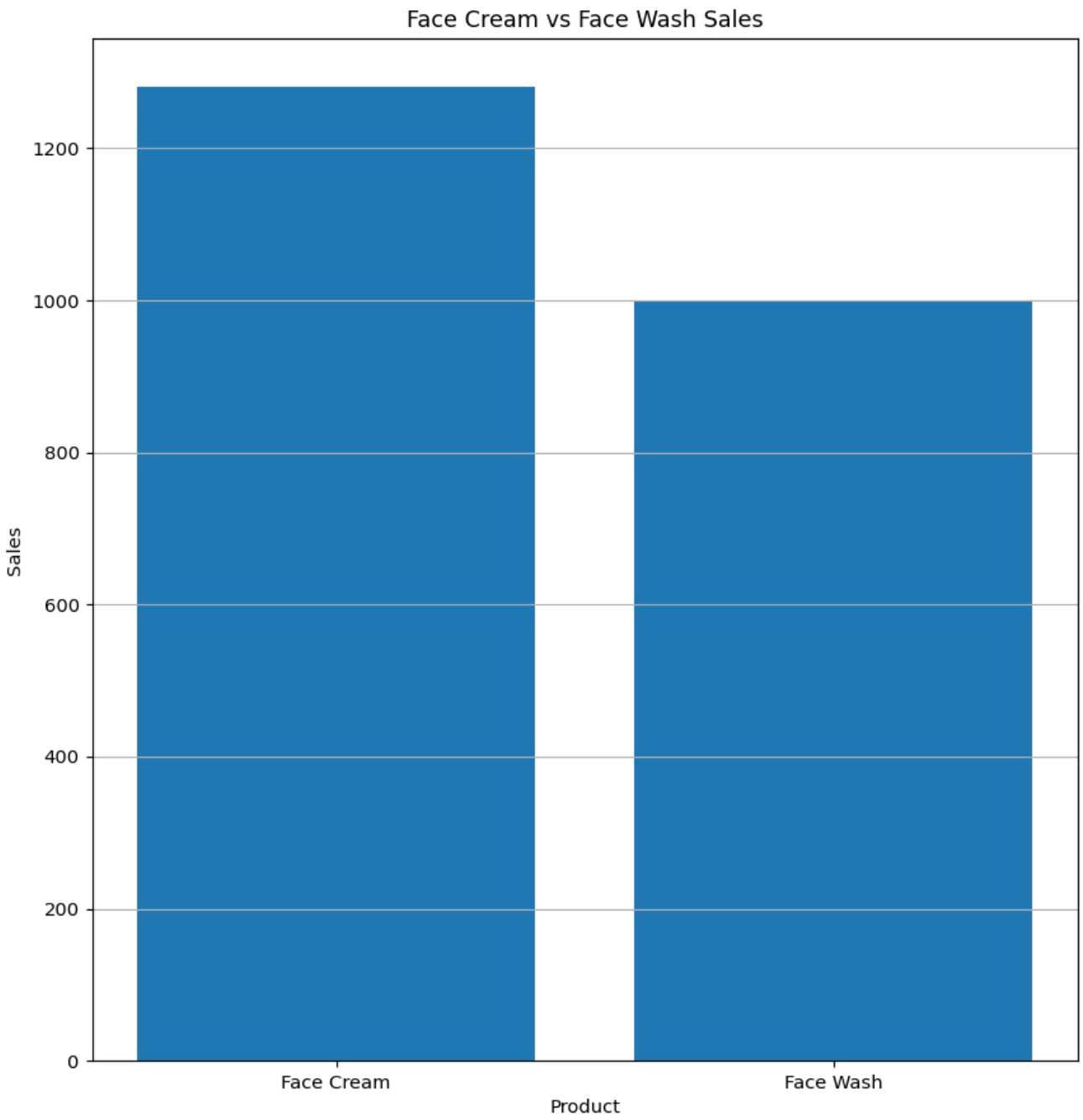
sales_data.csv

```
month_number,facecream,facewash,toothpaste,bathingsoap,shampoo,moisturizer,t
otal_units,total_profit
1,100,80,150,120,90,70,510,2000
2,120,90,180,130,100,80,600,2500
3,90,70,140,110,80,60,450,1800
4,110,85,160,125,95,75,550,2200
```

```
5,80,65,130,100,70,55,400,1600
6,130,100,170,140,110,85,635,2700
7,100,80,150,120,90,70,510,2000
8,140,110,180,150,120,90,690,3000
9,95,75,145,115,85,65,480,1900
10,120,90,160,130,100,80,600,2400
11,85,70,135,105,75,60,430,1700
12,110,85,155,120,90,75,535,2100
```

Output





9. Write a code segment that prints the names of all of the items in the currentworking directory.

```
import os
for item in os.listdir():
        print(item)
```

10. Write a python program to create two numpy arrays of random integers between 0 and 20 of shape (3, 3) and perform matrix addition, multiplication and transpose of the product matrix.

```
import numpy as np

# Create two NumPy arrays of random integers between 0 and 20 of shape (3,
3)
array1 = np.random.randint(0, 21, size=(3, 3))
array2 = np.random.randint(0, 21, size=(3, 3))

# Print the original arrays
print("Array 1:\n", array1)
print("Array 2:\n", array2)

# Perform matrix addition
sum_matrix = np.add(array1, array2)

# Print the sum matrix
print("\nSum of matrices:\n", sum_matrix)

# Perform matrix multiplication
product_matrix = np.dot(array1, array2)

# Print the product matrix
print("\nProduct of matrices:\n", product_matrix)

# Perform transpose of the product matrix
transposed_product = product_matrix.T

# Print the transposed product matrix
print("\nTranspose of the product matrix:\n", transposed_product)
```

11. Write Python program to write the data given below to a CSV file named student.csv

fields = ['Name', 'Branch', 'Year', 'CGPA']
rows = [['Nikhil', 'CSE', '2', '8.0'],
['Sanchit', 'CSE', '2', '9.1'],
['Aditya', 'IT', '2', '9.3'],
['Sagar', 'IT', '1', '9.5']]

```
import csv

# Define data fields and rows
fields = ['Name', 'Branch', 'Year', 'CGPA']
rows = [
    ['Nikhil', 'CSE', '2', '8.0'],
    ['Sanchit', 'CSE', '2', '9.1'],
    ['Aditya', 'IT', '2', '9.3'],
    ['Sagar', 'IT', '1', '9.5'],
]

# Open the CSV file in write mode
with open('student.csv', 'w', newline='') as csvfile:
  # Create a CSV writer object
  writer = csv.writer(csvfile)

  # Write the header row
  writer.writerow(fields)

  # Write each data row
  writer.writerows(rows)

print("Student data written to student.csv successfully!")
```

12. Consider the above student.csv file with fields Name, Branch, Year, CGPA .

Write python code using pandas to

1. **To find the average CGPA of the students**
2. **To display the details of all students having CGPA > 9**
3. **To display the details of all CSE students with CGPA > 9**
4. **To display the details of student with maximum CGPA**
5. **To display average CGPA of each branch**

```python
import pandas as pd

# Read the CSV data into a DataFrame
data = pd.read_csv('student.csv')

# 1) Average CGPA of all students
avg_cgpa = data['CGPA'].mean()
print("Average CGPA:", avg_cgpa)

# 2) Students with CGPA > 9
high_cgpa_students = data[data['CGPA'] > 9]
print("\nStudents with CGPA > 9:\n", high_cgpa_students)

# 3) CSE students with CGPA > 9
cse_high_cgpa = data[(data['Branch'] == 'CSE') & (data['CGPA'] > 9)]
print("\nCSE Students with CGPA > 9:\n", cse_high_cgpa)

# 4) Student with maximum CGPA
max_cgpa_student = data.loc[data['CGPA'].idxmax()]
print("\nStudent with maximum CGPA:\n", max_cgpa_student)

# 5) Average CGPA of each branch
avg_cgpa_branch = data.groupby('Branch')['CGPA'].mean()
print("\nAverage CGPA of each branch:\n", avg_cgpa_branch)
```

student.csv

```
Name,Branch,Year,CGPA
Nikhil,CSE,2,8.0
Sanchit,CSE,2,9.1
Aditya,IT,2,9.3
Sagar,IT,1,9.5
```

Output

```
Average CGPA: 8.975000000000001

Students with CGPA > 9:
       Name Branch  Year  CGPA
1  Sanchit    CSE     2   9.1
2   Aditya     IT     2   9.3
3    Sagar     IT     1   9.5

CSE Students with CGPA > 9:
       Name Branch  Year  CGPA
1  Sanchit    CSE     2   9.1

Student with maximum CGPA:
 Name      Sagar
Branch       IT
Year          1
CGPA        9.5
Name: 3, dtype: object

Average CGPA of each branch:
 Branch
CSE    8.55
IT     9.40
Name: CGPA, dtype: float64
```

13. Consider a CSV file ‘weather.csv’ with the following columns (date,temperature, humidity, windSpeed, precipitationType, place, weather {Rainy,Cloudy, Sunny}).

Write commands to do the following using Pandas library.

1. **Print first 10 rows of weather data.**
2. **Find the maximum and minimum temperature**

3. **List the places with temperature less than 28oC.**
4. **List the places with weather = “Cloudy”**
5. **Sort and display each weather and its frequency**
6. **Create a bar plot to visualize temperature of each day**

```
import pandas as pd
import matplotlib.pyplot as plt

# Read the CSV data into a DataFrame
data = pd.read_csv('weather.csv')

# 1. Print first 10 rows
print("First 10 rows:\n", data.head(10))

# 2. Maximum and minimum temperature
max_temp = data['temperature'].max()
min_temp = data['temperature'].min()
print("\nMaximum temperature:", max_temp, "°C")
print("Minimum temperature:", min_temp, "°C")

# 3. Places with temperature less than 28°C
cold_places = data[data['temperature'] < 28]
print("\nPlaces with temperature less than 28°C:\n",
cold_places['place'].unique())

# 4. Places with weather = "Cloudy"
cloudy_places = data[data['weather'] == "Cloudy"]
print("\nPlaces with weather = 'Cloudy':\n",
cloudy_places['place'].unique())

# 5. Sort and display weather frequency
weather_counts = data['weather'].value_counts()
print("\nWeather frequency:\n", weather_counts)

# 6. Bar plot for temperature
plt.bar(data['date'], data['temperature'], color='skyblue')
plt.xlabel('Date')
plt.ylabel('Temperature (°C)')
plt.title('Daily Temperature')
plt.show()
```

weather.csv

```
First 10 rows:
          date  temperature  humidity  windSpeed precipitationType
place weather
0  2024-06-02         30.5        65         10        Light Rain   New York
Rainy
1  2024-06-02         25.8        72          8               NaN     London
Cloudy
2  2024-06-01         28.2        58         12               NaN      Paris
Sunny
3  2024-06-02         22.1        80          5     Light Drizzle     Berlin
Cloudy
4  2024-06-01         33.7        48         15               NaN      Tokyo
Sunny
5  2024-06-02         29.9        70          9     Moderate Rain  Singapore
Rainy
6  2024-06-01         21.5        62         11               NaN     Moscow
Cloudy
7  2024-06-02         18.7        85          7        Heavy Rain     Ottawa
Rainy
8  2024-06-01         31.2        55         14               NaN    Beijing
Sunny
9  2024-06-02         27.4        75          6        Light Rain       Rome
Rainy

Maximum temperature: 33.7 °C
Minimum temperature: 18.7 °C

Places with temperature less than 28°C:
 ['London' 'Berlin' 'Moscow' 'Ottawa' 'Rome']

Places with weather = 'Cloudy':
 ['London' 'Berlin' 'Moscow']

Weather frequency:
 weather
```

```
Rainy     4
Cloudy    3
Sunny     3
Name: count, dtype: int64
```

Output