Practical- 9

INSTITUTE OF TECHNOLOGY & MANAGEMENT UNIVERSE, VADODARA

ITM UNIVERSE

DATE: / /

CLASS:

Sheet No.....

* CSV :- comma Spearated values is a simple file formate used to store tabular data, such as a spreadsheet or data base.

→ CSV files store tabular data in plain text.

→ Each raw of a file is data record or observations. And each record consist of one or more field, separated by commas (,).

→ To work with python we have import csv module.

* Dialect :-

→ It is like a book which consist of three parameters.

→ To use dialect, first we have to register it using register_dialect().

→ Syntax:

csv.register_dialect("name", delimeter, skipinitialspace, quoting).

* reader():

→ To read a csv file in python, we use reader() of csv module.

→ Syntax:

csv.reader(csv_file, dialect="name")

Modes:
"r": To read an existing file
"w": To write or create a file if filepath don't exist.
"a": To append content in existing file.
"+": To give read and write permition.

DictReader():

→ Dict reader class allows, you to create object. But it maps the header and every observation/record in dictionary form.

→ Reading data in dictionary form is more easy way.

→ syntax :-

csv.DictReader("csv-file", fieldnames, dialect.)

Writing in csv :-

→ To write a csv file in python, we use the csv.writer().

→ It returns a object that converts the user's data into a delimited string.

→ This string can be used to write using writerow().

→ Syntax:

csv.writer("csv-file", dialect)

- writerow() :- It is used when we want only a single row at a time
- writerows() :- It is used to write multiple rows at a time.

DATE: / /
CLASS:
Sheet No.....

DictWriter():

→ This class returns a writer object which maps dictionaries onto output rows.

→ syntax:

csv.DictWriter("csv-file", fieldnames, dialect)

Practical-10

* Numpy module:

→ The Numpy module, short for "Numerical Python" is a fundamental package for scientific computing in python. It provides support for larger, multi-dimensional array and matrices, along with a collection of mathematical functions to operate on these arrays.

→ Purpose and functionalites of Numpy.

→ The main purpose of Numpy is to bring the capabilities of numerical computing to python.

- Multi-dimensional array object:- Numpy an ndarray object, which is a multi-dimensional array that can hold elements of the same data type
- Mathematical functions:- Numpy includes a wide range of mathematical functions for performing operations on arrays

→ In the numpy module there are several functions available to create matrices of different types and dimensions. Some of thies functions are listed bellow as follow:-

- np.array(): create a matrix from a list or nested lists.
- np.zeros(): create a matrix of zeros with the specified shap
- np.ones(): Create a matrix of ones with the specified shape.
- np.full(): Create a matrix filled with a specified value and s

- np.arange(): Create a matrix using np.arange() to generate a range of values
- np.linspace(): create an array of evenly spaced values over a specified interval:
- np.empty(): Create a new matrix of the specified shape and data type without initializing entries

  Note: The entries are randomly initialize
- np.identity(): Create a square identity matrix of the specified size.

* n-darray → 1d array :-

→ The Numpy module provides several functions that can be used to convert an n-dimensional array matrix into a 1-dimensional array matrix.

→ Below is a list of these conversion function :-

- flatten(): returns a copy of the array collapsed into one dimension.
- np.ravel(): does the same thing as flatten() but it doesn't return a copy of the array, instead it performs the operation the passed array itself.
- np.reshape(): Reshapes the array into the

DATE: / /
CLASS:
Sheet No.....

Slicing :-

Slicing refers to the extraction of a subset of elem from an array. When dealing with matrices, slicing allows you to access and manipulate specific parts of the matrix, such as rows, colums.

→ Slicing 1D Array:
To slice a Numpy array, you can use the syntax:

- array[start: stop: step]

- start: The starting index of the slice
- stop: The stopping index of the slice
- step: The step size or interval between elements.

Note: If start, stop, step is not provided, they defaut to the start of the array, the end of array, and 1, respectively.

→ Slicing 2D Array (Matrix):
This will create a sub-matrix with elements from the rows and colbums.

- selecting specific row or colums: You can use slicing to select specific row or colums. For eg. matrix[:, 1]

→ Slicing 3D Array (Matrix):

- slicing in 3D arrays are similar to those in 1D and 2D arrays, with an additional dimention to consid

The syntax for slicing in 3D array is:

- array[start:stop:step, start:stop:step, start:stop:step].
- Here each start:stop:step tuple corresponds to a different dimension (eg, depth, rows, colums.).

* statistical functions:

→ Numpy provides a wide range of statistical functions for performing various statistical computations on arrays and matrices. these fuunctions facilitate tasks such as calculating measures of central tendency, dispersion, and more:

→ list of statistical function in Numpy:

- Mean: np.mean(array, axis=None) - Computes the arithmetic mean along the specified axis
- Median: np.median(array, axis=None) - Computes the median along the specified axis.
- Standard Deviation: np.std(array, axis=None) - Computes the standard Deviation along the specified axis.
- Varience: np.var(array, axis=None) computes the varian along the specified axis.
- Percentile: np.percentile(array, q, axis=None) - compute the q-th percentile of the data along the specified axis.
- Minimum and Maximum: np.min(Array, axis=None) and np.max(array, axis=None) - computes the minimum and maximum values along the specified axis.

DATE: / /
CLASS:
Sheet No.....

Arithmatic functions:

→ Numpy proxides a diverse set of arithmetic functio that enable various mathematical operations on arra and matrices. These functions are fundamental fo numerical computations ~~and~~ in python:

→ list of Arithmetic function in Numpy:

- Addition: np.add(x1, x2) - Adds corresponding element of two arrays.
- Subtraction: np.subtract(x1, x2) - subtracts elements of the second array from the first array
- Multiplication: np.multiply(x1, x2) - Multiplies correspondi elements of two array
- Division - np.divide(x1, x2) - Divides elements of the first array by the corresponding eleme of the second arrays.
- Power: np.power(x, p) - Raises elements of the input array to the power of p.

Practical - 11

INSTITUTE OF TECHNOLOGY & MANAGEMENT UNIVERSE, VADODARA

ITM UNIVERSE

DATE: / /

CLASS:

Sheet No.....

Pandas module:

→ The pandas module in python is a powerful library provides data structure and data analysis tools. It is widely used for data manipulation cleaning, analysis, and visualization. The primary data structure in pandas are Series & and Dataframe

→ It provides methods to read csv file into a Dataframe, perform data manipulation, and write the modified data back to csv file formate.

Creating Dataframe and Series:

→ Creating Series:

- Using list or arrays: Pass a list or array of data to the pd.Series() constructor to create a series. Optionally you can also specify custom index labels.

→ Creating Dataframe:

- Using Dictionaries: Pass a dictionary to the pd.Dataframe() constructor where keys represent column names and values represent the data for each column. Optionally you can also specify custom index labels.

- Using other Data Structure: Dataframes can also be created from other data structures like numpy arrays or Series.

Functions in Pandas:-

→ The pandas module offers a wide range of analysis functions for data analysis. These functions are essential for gaining insights into the data, performing statistical analysis, and summarizing the information within a DataFrame. Below is a comprehensive list of analysis functions:

- head(n): Returns the first n rows of the DataFrame or Series, which is useful for quickly testing if your object has the right type of data in it.
- tail(n): Returns the last n rows of the DataFrame or Series, which is useful for verifying the data after sorting or appending operation.
- size: Returns the total number of elements in the DataFrame. It is an attribute, not a method.
- shape: Returns a tuple representing the dimensionality of the DataFrame, i.e, the number of rows and columns
- info(): Provides a concise summary of a DataFrame. This method prints information about a DataFrame including the index dtype and column dtypes, non-null values, and memory usage.
- describe(): Generates descriptive statistics that summarize the central tendency, dispersion, and shape of a dataset's distribution.