UNIT 5 INTRODUCTION TO PYTHON LIBRARIES

Python Libraries- Introduction to Libraries- Creating and Exploring Packages-Numpy, SciPy, matplotlib, Pandas, Scikit-learn- seaborn.

5.1 Python Libraries

- The Python Standard Library is a collection of exact syntax, token, and semantics of Python.
- Python has a huge collection of libraries.
- The Python Standard Library contains hundreds of modules for performing common tasks,
- Python's standard library is very extensive, offering a wide range of facilities.
- A Python library is a reusable chunk of code that you may want to include in your programs/ projects.
- Mostly it is written in C, and handles functionality like I/O and other core modules.
- Python supports very huge number of library files. Some examples are:
    - Numpy
    - SciPy
    - Pandas
    - Matplotlib
    - Scikit-Learn
    - Seaborn
    - Keras
    - PyTorch
    - TensorFlow

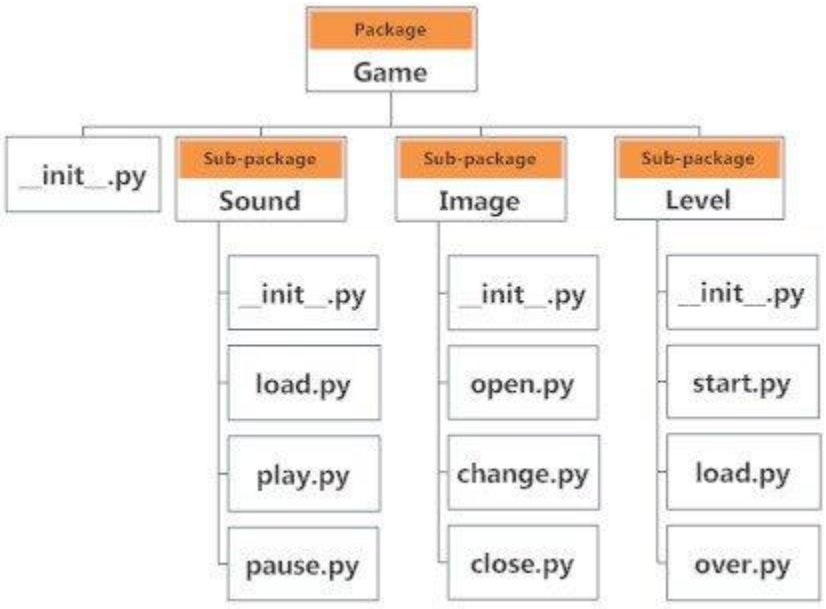
5.2 Introduction to Libraries

- **Library Definition:** The **library** is having a collection of related functionality of codes that allows you to perform many tasks without writing your code. It is a reusable chunk of code that we can use by importing it in our program.
- A **library** means "a bundle of code.".
- A library is a collection of pre-combined codes that can be used iteratively to reduce the time required to code.
- They are particularly useful for accessing the pre-written frequently used codes, instead of writing them from scratch every single time.
- The library contains built-in modules that provide access to system functionality such as file I/O.

- A library is a collection of modules.

5.3 Creating and Exploring Packages

- **Module Definition**: The module is a simple Python file that contains collections of functions and global variables and with having a .py extension file.
- **Package Definition:** The **package** is a simple directory having collections of modules. This directory contains Python modules and also having **__init__.py** file by which the interpreter interprets it as a Package.
- Module can contain multiple objects, such as classes, functions, etc.
- A package can contain one or more relevant modules.
- A package is actually a folder containing one or more module files.
- The package folder contains a special file called __init__.py, which stores the package's content.



- **import statement**:
  - ✓ A file is considered as a module in python. To use the module, you have to import it using the **import** keyword.
  - ✓ import modules from packages using the dot (.) operator.
  - ✓ import Game.Level.start
-

Syntax: import module_name	**Example:** import math
Syntax:	**Example:**

from module_name import member_name	from math import pi
Syntax: from module_name import *	**Example:** from math import *
Syntax: import module_name as alias_name	**Example:** import math as ma

- **Use the following steps to create a package in Python:**
    1. First, we create a directory and give it a package name, preferably related to its operation.
    2. Then create more modules in a directory, we put the classes and the required functions in a module.
    3. Finally we create an __init__.py file inside the directory, to let Python know that the directory is a package.
- **Example**:
    1. First create a directory (package) i.e name of the directory is Arithmetic
    2. Create first module with name of the module is addition.py and define one function i.e add() inside the addition.py file

```
def add(a,b):
    c=a+b
    print(f"Addition is :{c} ")
```

    3. Create second module with name of the module is subtraction.py and define one function i.e sub() inside the subtraction.py file

```
def sub(a,b):
    c=a-b
    print(f"Subtraction is :{c} ")
```

    4. Finally create the __init__.py file. This file will be placed inside Arithmetic directory and can be left blank.
    5. Call(import) Arithmetic package in a program i.e ruff.py to perform addition and subtraction of two numbers.

```
import Arithmetic.addition
import Arithmetic.subtraction
Arithmetic.addition.add(10,20)
Arithmetic.subtraction.sub(50,30)
```

Example:	**Output**:
1.Package name is **Arithmetic** 2.**addition.py**	Addition is :30 Subtraction is :20

```
def add(a,b):
    c=a+b
    print(f"Addition is :{c} ")
```

3.**subtraction.py**

```
def add(a,b):
    c=a+b
    print(f"Addition is :{c} ")
```

4.**ruff.py**

```
import Arithmetic.addition
import Arithmetic.subtraction
Arithmetic.addition.add(10,20)
Arithmetic.subtraction.sub(50,30)
```

5.4 Numpy

- NumPy (**Numerical Python**) is the fundamental package for numerical computation in Python
- It is the most popular machine learning library in Python.
- NumPy is a Python library used for working with arrays.
- It also has functions for working in domain of linear algebra, fourier transform, and matrices.
- It is an open source library.
- NumPy is one of the fundamental packages for Python providing support for large multidimensional arrays and matrices.
- NumPy relies on BLAS and LAPACK for efficient linear algebra computations.
- NumPy can also be used as an efficient multi-dimensional container of generic data.
- NumPy is one of the most used libraries for tasks involving modern scientific computations and evolving yet powerful domains like Data Science and Machine Learning.
- It supports for powerful N-dimensional array objects and built-in tools for performing intensive mathematical as well as scientific calculations.
- It is used for delivering high performance, interoperability with various computing platforms and hardware, and ease of use.

- NumPy is the fundamental package for scientific computing with Python, adding support for large, multidimensional arrays and matrices, along with a large library of high-level mathematical functions to operate on these arrays.

- **Features OfNumpy**
  - ✓ **Interactive:** Numpy is very interactive and easy to use.
  - ✓ **Mathematics:** Makes complex mathematical implementations very simple.
  - ✓ **Intuitive:** Makes coding real easy and grasping the concepts is easy.
  - ✓ **Lot of Interaction:** Widely used, hence a lot of open source contribution
- **Applications Of Numpy**
  - ✓ Extensively used in data analysis
  - ✓ Creates powerful N-dimensional array
  - ✓ Forms the base of other libraries, such as SciPy and scikit-learn
  - ✓ Replacement of MATLAB when used with SciPy and matplotlib

Example:	**Output:**
import numpy as np a=np.array([1,2,3,4,5]) b=np.array([[1,2,3],[4,5,6]]) print(a) print(b) print(type(a)) print(b.ndim) print(a[2]) print(a[1:3]) x = a.copy() x[0]=0 print(x) print(b.shape)	**[1 2 3 4 5]** **[[1 2 3]** **[4 5 6]]** **<class 'numpy.ndarray'>** **2** **3** **[2 3]** **[0 2 3 4 5]** **(2, 3)**

- import numpy library.
- NumPy package can be referred to as np instead of numpy.
- The indexes in NumPy arrays start with 0.
- **array()** - function to create arrays
- **type()** - the type of the object passed to it.
- **ndim** attribute - that returns an integer that tells us how many dimensions the array have.
- **copy() –** copy the content of one array into another.
- **shape attribute -** It returns an array has how many dimensions, and each dimension has how many elements.

5.5 SciPy

- SciPy stands for Scientific Python.
- It provides more utility functions for optimization, stats and signal processing.
- Like NumPy, SciPy is open source so we can use it freely.
- **SciPy** is an open-source library used for solving mathematical, scientific, engineering, and technical problems.
- It allows users to manipulate the data and visualize the data using a wide range of high-level Python commands.
- SciPy is built on the Python NumPy extention.
- SciPy library contains modules for optimization, linear algebra, integration, and statistics.
- SciPy is a library used by scientists, analysts, and engineers doing scientific computing and technical computing.
- It contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers, and other tasks common in science and engineering.

- **Features of Scipy**
  - ✓ Collection of algorithms and functions built on the NumPy extension of Python
  - ✓ High-level commands for data manipulation and visualization
  - ✓ Multidimensional image processing with the SciPy ndimage submodule
  - ✓ Includes built-in functions for solving differential equations
- **Applications of Scipy**
  - ✓ Multidimensional image operations
  - ✓ Solving differential equations and the Fourier transform
  - ✓ Optimization algorithms
  - ✓ Linear algebra
- **Subpackages in SciPy**

Package Name	Description
scipy.io	• File input/output
scipy.special	• Special Function
scipy.linalg	• Linear Algebra Operation
scipy.interpolate	• Interpolation
scipy.optimize	• Optimization and fit
scipy.stats	• Statistics and random numbers
scipy.integrate	• Numerical Integration
scipy.fftpack	• Fast Fourier transforms
scipy.signal	• Signal Processing

scipy.ndimage	• Image manipulation

- import the SciPy module using **fromscipy import *module*** statement.
- Ex : from scipy import constants - imported the *constants* module from SciPy

Example:	**Output:**
from scipy import special a = special.exp10(3) print(a) b = special.exp2(3) print(b) c = special.sindg(90) print(c) d = special.cosdg(45) print(d) d = special.tandg(45) print(d)	**1000.0** **8.0** **1.0** **0.7071067811865475** **1.0**

5.6 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.
- It is used to create a wide variety of visualizations, including line plots, histograms, bar charts, pie charts, scatter plots, tables, and many other styles.
- Matplotlib is a Python 2D plotting library that produces publication-quality figures in a variety of hard-copy formats and interactive cross-platform environments.
- This 2D plotting library of Python is very famous among data scientists for designing varieties of figures in multiple formats which is compatible across their respected platforms.
- **Features of Matplotlib:**
  - ✓ Usable as a MATLAB replacement, with the advantage of being free and open source
  - ✓ Supports dozens of backends and output types, which means you can use it regardless of which operating system you're using or which output format you wish to use

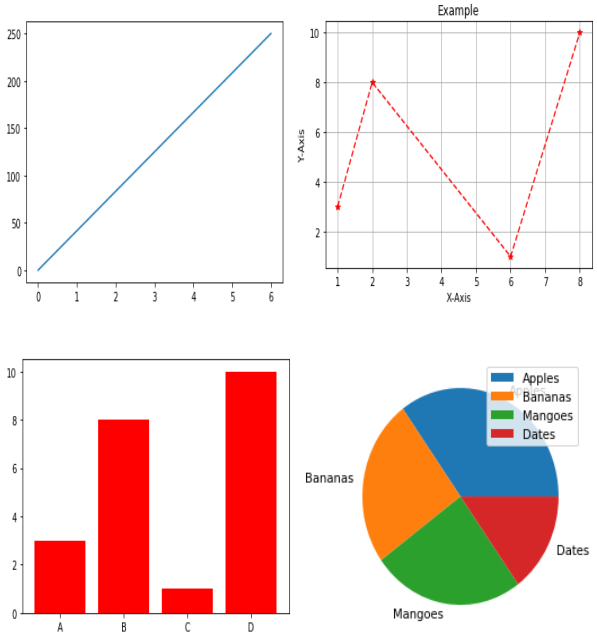
- ✓ Pandas itself can be used as wrappers around MATLAB API to drive MATLAB like a cleaner
- ✓ Low memory consumption and better runtime behavior

- **<u>Applications of Matplotlib</u>:**
  - ✓ Correlation analysis of variables
  - ✓ Visualize 95 percent confidence intervals of the models
  - ✓ Outlier detection using a scatter plot etc.
  - ✓ Visualize the distribution of data to gain instant insights

Example:

```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([0, 6])
y = np.array([0, 250])
plt.plot(x, y)
plt.show()

x = np.array([1, 2, 6, 8])
y = np.array([3, 8, 1, 10])
plt.plot(x, y,'*r--')
plt.title("Example")
plt.xlabel("X-Axis")
plt.ylabel("Y-Axis")
plt.grid()
plt.show()

x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])
plt.bar(x,y,color="red")
plt.show()

y = np.array([35, 25, 25, 15])
l = ["Apples", "Bananas", "Mangoes", "Dates"]
plt.pie(y,labels=l)
plt.legend()
plt.show()
```

Output:



- Most of the Matplotlib utilities lies under the pyplot submodule, and are usually imported under the plt alias.
- **plot()** function draws a line from point to point.
- keyword argument **marker** to emphasize each point with a specified marker
- **title()** function to set a title for the plot.
- **xlabel() and ylabel()** functions to set a label for the x- and y-axis
- **grid()** function to add grid lines to the plot.
- **bar()** function to draw bar graphs
- **pie()** function to draw pie charts

5.7 Pandas

- PANDAS referred as Python Data Analysis Library.
- PANDAS is another open source Python library for availing high-performance data structures and analysis tools.
- Pandas is a machine learning library in Python that provides data structures of high-level and a wide variety of tools for analysis
- It is an open source library.
- Ability to translate complex operations with data using one or two commands.
- Pandas have so many inbuilt methods for grouping, combining data, and filtering, as well as time-series functionality.
- Ability to group and sort data, select best suited output for the apply method, and provides support for performing custom types operations.
- Pandas enable the provision of easy data structure and quicker data analysis for Python
- It provides fast, expressive, and flexible data structures to easily work with structured (tabular, multidimensional, potentially heterogeneous) and time-series data.
- It is used for solving modern Data Science and Machine Learning problems.
- Reading and writing data into multiple formats like csv, excel, sqletc
- Pandas is the "*SQL of Python.*"
- It is used to handle two-dimensional data tables in Python.
- Load data into data frames, select columns, filter for specific values, group by values, run functions (sum, mean, median, min, max, etc.), merge dataframes and so on.
- It contains DataFrame as its main data structure.
- With DataFrame you can store and manage data from tables by performing manipulation over rows and columns.
- Also create multi-dimensional data-tables.
- It was created for data analysis, data cleaning, data handling and data discovery
- **Features Of Pandas**
    - ✓ An efficient DataFrame object for data manipulation

- ✓ Easy reshaping and pivoting of data sets
- ✓ Merging and joining of data sets
- ✓ Label-based data slicing, indexing, and subsetting
- ✓ working with time-series data
- ✓ Eloquent syntax and rich functionalities that gives you the freedom to deal with missing data
- ✓ Enables you to create your own function and run it across a series of data
- ✓ High-level abstraction
- ✓ Contains high-level data structures and manipulation tools
- ✓ Manipulating data will be easier.
- ✓ Support for various operations such as Re-indexing, Iteration, Sorting, Aggregations, Concatenations and Visualizations.

- **Applications Of Pandas**
  - ✓ It is the best data analysis tool available for solving real-world problems
  - ✓ General data wrangling and cleaning
  - ✓ It is the best data analysis tool available for solving real-world problems
  - ✓ ETL (extract, transform, load) jobs for data transformation and data storage, as it has excellent support for loading CSV files into its data frame format
  - ✓ Used in a variety of academic and commercial areas, including statistics, finance and neuroscience
  - ✓ Time-series-specific functionality, such as such as date range generation, moving window, linear regression and date shifting.

Example:	**Output:**
import pandas as pd df = pd.DataFrame({'States':['TN', 'AP', 'UP', 'MP'], 'Capitals':['CHENNAI', 'AMARAVATHI', 'LUCKNOW', 'BHOPAL']}) df.to_excel('data1.xlsx') df = pd.read_excel('data1.xlsx') print(df)	States Capitals 0 TN CHENNAI 1 AP AMARAVATHI 2 UP LUCKNOW 3 MP BHOPAL

Example: Read csv file	**Output:**
import pandas as pd df = pd.read_csv('stock.csv') print(df.to_string()) new_df = df.dropna() print(new_df.to_string()) x = df["maths"].mean() print(x)	Regno name science physics maths 0 111 aaa 89 78 56 1 222 bbb 54 87 64

- import pandas library alias name is pd.
- Data sets in Pandas are usually multi-dimensional tables, called DataFrames.
- loc attribute to return one or more specified row(s)
- dropna() - remove rows that contain empty cells.
- fillna() - replace empty cells with a value
- mean() - to calculate the mean value of the column
- median() - to calculate the median value of the column
- mode() - to calculate the mode value of the column
- drop_duplicates() – remove duplicates rows

5.8 scikit-Learn

- It is a Python library is associated with NumPy and SciPy.
- It is considered as one of the best libraries for working with complex data.
- It can be effectively used for a variety of applications which include classification, regression, clustering, model selection, naive Bayes', grade boosting, K-means, and preprocessing.
- Scikit-Learn, which simply defines itself as "Machine Learning in Python."
- It is designed to be interpolated into NumPy and SciPy.
- It is a simple tool for data analysis and mining-related tasks.
- It is open-source library files.
- It is being used for classification, regression and clustering to manage spam, image recognition, drug response, stock pricing, customer segmentation etc.
- It also allows dimensionality reduction, model selection and pre-processing.
- Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python.
- The sklearn library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction
- **Features Of Scikit-Learn**
    - ✓ **Cross-validation**: There are various methods to check the accuracy of supervised models on unseen data.
    - ✓ **Supervised Learning algorithms**: Almost all the popular supervised learning algorithms, like Linear Regression, Support Vector Machine (SVM), Decision Tree etc., are the part of scikit-learn.
    - ✓ **Unsupervised learning algorithms:** starting from clustering, factor analysis, principal component analysis to unsupervised neural networks.
    - ✓ **Feature extraction:** Useful for extracting features from images and text
    - ✓ **Clustering:** This model is used for grouping unlabeled data.

- ✓ **Simple and efficient tools for predictive data analysis.**
- ✓ **It helps in all aspects and algorithms of Machine Learning, even Deep Learning.**
- ✓ **It is easy to learn and use.**

- **<u>Applications of Scikit-Learn</u>**
  - ✓ clustering
  - ✓ classification
  - ✓ regression
  - ✓ model selection
  - ✓ dimensionality reduction
- **<u>Modelling process in Scikit-Learn</u>**
  - ✓ **Dataset Loading** - A collection of data is called dataset.
  - ✓ **Preprocessing the Data** - Before inputting that data to machine learning algorithms, we need to convert it into meaningful data. This process is called preprocessing the data.
  - ✓ **Splitting the dataset** - To check the accuracy of our model, we can split the dataset into two pieces-a training set and a testing set.
  - ✓ **Train the Model**- use our dataset to train some prediction-model.

Example:

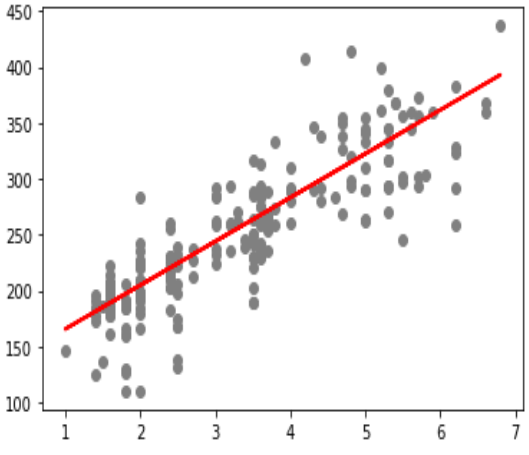
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import
train_test_split
from sklearn.linear_model import
LinearRegression

dataset=pd.read_csv("fuel.csv")
print(dataset.head())
print(dataset.shape)
X = dataset['ENGINESIZE'].values.reshape(-1,1)
y = dataset['CO2EMISSIONS'].values.reshape(-
1,1)
X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.2,
random_state=0)
regr = LinearRegression()
regr.fit(X_train, y_train)
print(regr.intercept_)
y_pred = regr.predict(X_test)
```

Output:

(1067, 13)

[126.18920155]

	Actual	Predicted
0	**356**	**342.055315**
1	**209**	**220.385324**
2	**230**	**220.385324**
3	**212**	**232.159839**
4	**168**	**224.310162**



df = pd.DataFrame({'Actual': y_test.flatten(), 'Predicted': y_pred.flatten()}) print(df) plt.scatter(X_test, y_test, color='gray') plt.plot(X_test, y_pred, color='red', linewidth=2) plt.show()	

5.9 seaborn

- Seaborn was designed to visualize the complex statistical models.
- It has the potential to deliver accurate graphs such as heat maps.
- Seaborn is a Python data visualization library based on matplotlib.
- It provides a high-level interface for drawing attractive and informative statistical graphics.
- **Seaborn** is a data visualization library built on top of matplotlib and closely integrated with pandas data structures in Python.
- Visualization is the central part of Seaborn which helps in exploration and understanding of data.
- Seaborn aims to make visualization a central part of exploring and understanding data.
- Visualizing Statistical Data Using Seaborn.
- Seaborn which has powerful libraries to visualize and explore your data.
- **Features Of Seaborn**
  - ✓ Dataset oriented API to determine the relationship between variables.
  - ✓ Automatic estimation and plotting of linear regression plots.
  - ✓ It supports high-level abstractions for multi-plot grids.
  - ✓ Visualizing univariate and bivariate distribution.
  - ✓ Provides a high-level interface to draw statistical graphics.
- **Applications of Seaborn**
  - ✓ Distribution Plots
  - ✓ Pie Chart
  - ✓ Bar Chart
  - ✓ Scatter Plots
  - ✓ Pair Plots
  - ✓ Heat maps

- Distplot stands for distribution plot, it takes as input an array and plots a curve corresponding to the distribution of points in the array.
- Import the Seaborn module with another name is sns
- load_dataset()-loading dataset
- **jointplot()** - Bivariate Distribution is used to determine the relation between two variables.
- catplot() - Time series can be plotted using catplot() i.e barchart

Example:	**Output:**
import pandas as pd import seaborn as sns from matplotlib import pyplot as plt dataset=pd.read_csv("fuel.csv") sns.distplot(dataset['CO2EMISSIONS'],hist=False) sns.jointplot(x = "ENGINESIZE",y = 'CO2EMISSIONS',data = dataset,kind='hex') sns.catplot("CO2EMISSIONS", data=dataset, color='green',kind='count')