

## Practical – 1

**Aim: Basic Python programs. [ NumPy, Panda, Matplotlib]**


### 1. Creating blank array with predefined data

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

standings = np.array([575, 285, 234, 206, 206, 205, 200, 175, 97, 62])
```


### 2. Slicing and Updating elements.

```
# ! Slicing
arr = standings[:4]
print("Data slicing: ", arr)
# ! Updating
standings[0] = 576
# ! Printing updated data
print("Updating data in standings: ", standings)
```

 `Data slicing: [575 285 234 206]`  
`Updating data in standings: [576 285 234 206 206 205 200 175 97 62]`


### 3. Slicing and Updating elements.

```
# ! Reshaping
newarr = standings.reshape(5, 2)
print(newarr)
```

 `[[576 285]`  
`[234 206]`  
`[206 205]`  
`[200 175]`  
`[ 97 62]]`

### 4. Looping in numpy


```
for i in newarr:
    print(i)
```

 `[576 285]`  
`[234 206]`  
`[206 205]`  
`[200 175]`  
`[97 62]`

### 5. Read csv file in numpy

```
from google.colab import drive
drive.mount('/content/drive')
```

```
data_set = pd.read_csv("/content/drive/MyDrive/temp/prac_1.csv")
```

 `Mounted at /content/drive`

## 6. Create a dataframe

```
df = pd.DataFrame(data_set)
print(df)
```

	Drivers	Standing
0	Carlos Sainz	200
1	George Russell	175
2	Max Verstappen	575
3	Pierre Gasly	62
4	Sergio Perez	285
5	Oscar Piastri	97
6	Fernando Alonso	206
7	Lewis Hamilton	234
8	Lando Norris	205
9	Charles Leclerc	206

## 7. Slicing in created dataframe

```
print(df.iloc[:4])
```

	Drivers	Standing
0	Carlos Sainz	200
1	George Russell	175
2	Max Verstappen	575
3	Pierre Gasly	62

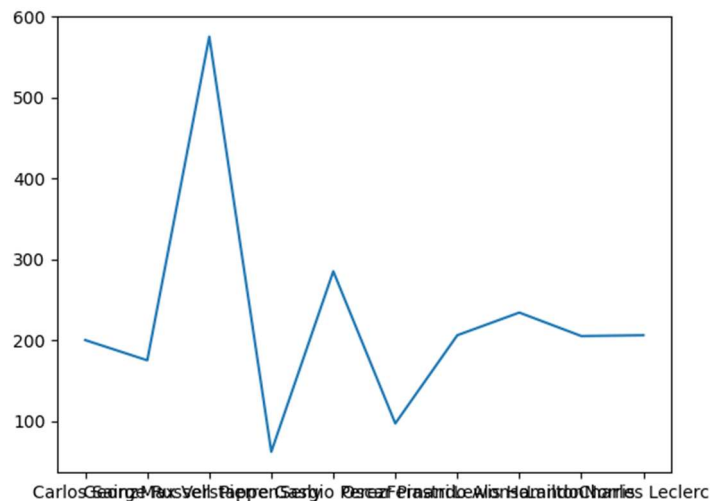
## 8. Column and Row manipulation

```
np.transpose(data_set)
```

	0	1	2	3	4	5	6	7	8	9
Drivers	Carlos Sainz	George Russell	Max Verstappen	Pierre Gasly	Sergio Perez	Oscar Piastri	Fernando Alonso	Lewis Hamilton	Lando Norris	Charles Leclerc
Standing	200	175	575	62	285	97	206	234	205	206

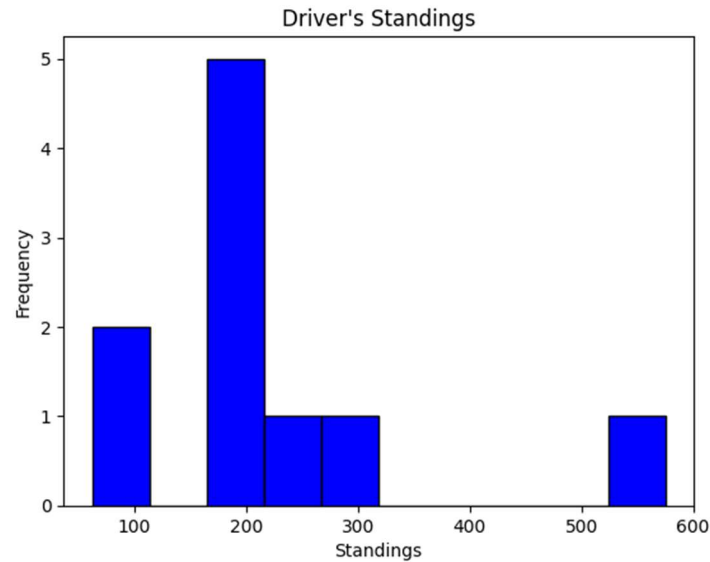
## 9. Importing matplotlib and make simple line chart

```
plt.plot(df["Drivers"], df["Standing"])
plt.show()
```



## 10. Make histogram

```
# ! Creating a histogram for standings
plt.hist(df["Standing"], bins=10, color='blue', edgecolor='black')
plt.title("Driver's Standings")
plt.xlabel("Standings")
plt.ylabel("Frequency")
plt.show()
```

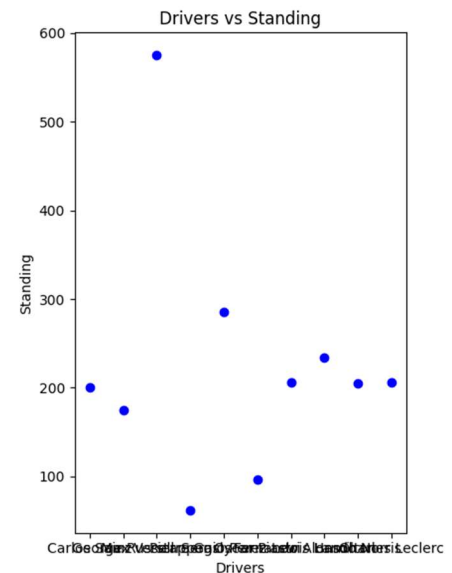


## 11. Plotting multivariate data

```
plt.figure(figsize=(12, 6))
```

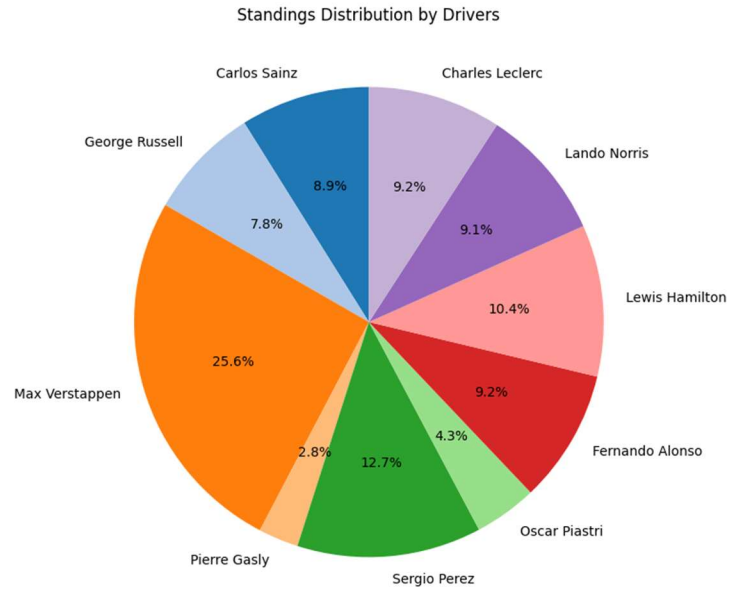
```
# ! Plot 1: Drivers vs Standing
plt.subplot(1, 3, 1)
plt.scatter(df["Drivers"], df["Standing"], color='blue')
plt.title("Drivers vs Standing")
plt.xlabel("Drivers")
plt.ylabel("Standing")
```

```
plt.tight_layout()
plt.show()
```



## 12. Plotting pie chart

```
plt.figure(figsize=(8, 8))
plt.pie(df["Standing"], labels=df["Drivers"], autopct='%1.1f%%', startangle=90,
        colors=plt.cm.tab20.colors)
plt.title("Standings Distribution by Drivers")
plt.show()
```



Faculty Signature: \_\_\_\_\_

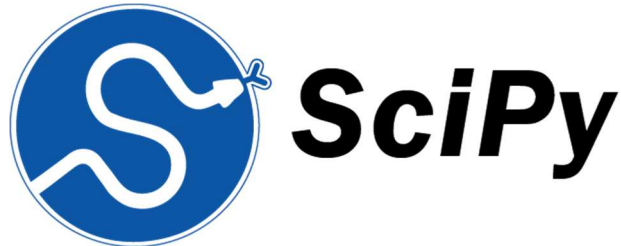
Date: \_\_\_\_\_

## Practical – 2

**Aim: Study of various Machine Learning libraries. [SciPy, Keras, SciKit-Learn, PyTorch, TensorFlow, Seaborn, Plotly]**

### 1. SciPy Library:

SciPy is an open-source Python library which is used to solve scientific and mathematical problems. It is built on the NumPy extension and allows the user to manipulate and visualize data with a wide range of high-level commands. As mentioned earlier, SciPy builds on NumPy and therefore if you import SciPy, there is no need to import NumPy.



- **Subpackages in SciPy:**

SciPy has a number of subpackages for various scientific computations which are shown in the following table:

Name	Description
cluster	Clustering algorithms
constants	Physical and mathematical constants
fftpack	Fast Fourier Transform routines
integrate	Integration and ordinary differential equation solvers
interpolate	Interpolation and smoothing splines
io	Input and Output
linalg	Linear algebra
ndimage	N-dimensional image processing
odr	Orthogonal distance regression
optimize	Optimization and root-finding routines
signal	Signal processing
sparse	Sparse matrices and associated routines
spatial	Spatial data structures and algorithms
special	Special functions
stats	Statistical distributions and functions

- **Features:**

- **Data visualization**

Includes functions for generating plot grids, contour plots, and scatter plots

- **Statistics**

Includes functions for performing statistical tests like t-tests, Wilcoxon rank sum tests, and linear regression analysis

- **Image processing**

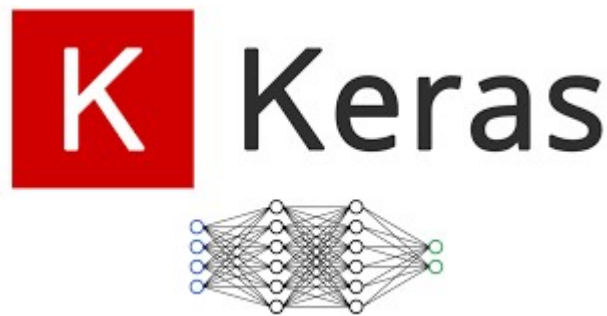
Includes functions for image manipulation and analysis, like color space transformation, texture analysis, and image segmentation

- **Sparse matrices**

Includes methods for operating with sparse matrices, which are matrices with the majority of their elements as zero.

## 2. Keras:

Keras is an open-source library that provides a Python interface for artificial neural networks. Keras was first independent software, then integrated into the TensorFlow library, and later supporting more. Keras 3 is a full rewrite of Keras [and can be used] as a low-level cross-framework language to develop custom components such as layers, models, or metrics that can be used in native workflows in JAX, TensorFlow, or PyTorch — with one codebase. Keras 3 will be the default Keras version for TensorFlow 2.16 onwards, but Keras 2 can still be used



- **What Keras does:**

- Keras allows users to quickly experiment with neural networks
- It provides a user-friendly API for defining and training deep learning models
- Keras can run on top of other frameworks like TensorFlow, CNTK, or Theano
- It supports CPUs and GPUs

- **Why Keras is useful:**

- Keras is open source and has a large community
- It's designed to be easy to use and experiment with
- It's flexible and modular, which makes it good for innovative research

- **Features:**

- User-friendly: Keras is easy to learn and use, with a simple API and pre-trained models.
- Flexible: Keras can be deployed on various devices and platforms.
- Fast: Keras offers fast debugging, prototyping, deployment, and research.
- Customizable: Keras allows users to easily customize and share models and their components.
- Integrates with other frameworks: Keras integrates with popular deep learning frameworks like TensorFlow and Theano.
- Dynamic input size: Keras allows users to export models with dynamic input size, which can handle varying image dimensions.
- Works with multiple back-ends: Keras supports multiple back-end engines, including TensorFlow, JAX, and PyTorch.

- Industry-strength: Keras can scale to large clusters of GPUs or an entire TPU pod.
- State-of-the-art research: Keras is used by many scientific organizations, including CERN, NASA, and NIH.
- Clear error messages: Keras provides clear and actionable error messages.

- **Subpackages**

Subpackages	Description
keras.layers	Predefined layers for creating neural networks, including Dense, Conv2D.
keras.models	Tools for defining and training sequential and functional API-based models.
keras.optimizers	Includes optimization algorithms like SGD, Adam, and RMSprop.
keras.losses	Provides loss functions such as Mean Squared Error and Cross Entropy.
keras.metrics	Built-in metrics for model evaluation like accuracy and precision.
keras.callbacks	Utilities for customizing training loops (e.g., EarlyStopping, ModelCheckpoint).
keras.preprocessing	Tools for preprocessing image, text, and sequence data.
keras.utils	Utility functions for tasks like one-hot encoding and model visualization.
keras.applications	Pretrained models for transfer learning and feature extraction.

### 3. SciKit-Learn:

scikit-learn (formerly scikits.learn and also known as sklearn) is a free and open-source machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support-vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. Scikit-learn is a NumFOCUS fiscally sponsored project.



- **Key features:**

- **Classification:**

- Identifying which category an object belongs to (e.g., spam detection, image recognition).

- **Regression:**

- Predicting a continuous numerical value (e.g., housing prices, stock prices).

- **Clustering:**

- Grouping similar data points together (e.g., customer segmentation, image segmentation).

- **Dimensionality reduction:**

- Simplifying data by reducing the number of features (e.g., feature selection, principal component analysis).

- **Model selection:**

- Choosing the best model for a given task (e.g., cross-validation, grid search).

- **Preprocessing:**

- Transforming data to make it suitable for machine learning algorithms (e.g., scaling, normalization).

- **Benefits:**

- **Ease of use:** Simple and consistent API for a wide range of algorithms.

- **Efficiency:** Optimized for performance, making it suitable for large datasets.

- **Versatility:** Wide range of algorithms and tools for various machine learning tasks.

- **Open source:** Free to use, modify, and distribute.

- **Community:** Large and active community providing support and documentation.



- **Subpackages**

Subpackages	Description
sklearn.linear_model	Linear models like Linear Regression, Ridge, Lasso, and Logistic Regression.
sklearn.ensemble	Ensemble methods like Random Forest, Gradient Boosting, and Bagging.
sklearn.cluster	Clustering algorithms such as K- Means, DBSCAN, and hierarchical clustering.
sklearn.svm	Support Vector Machines for classification and regression tasks.
sklearn.metrics	Metrics for evaluating models, including accuracy, precision, and ROC-AUC.
sklearn.preprocessing	Tools for scaling, normalization, and encoding data.
sklearn.model_selection	Cross-validation, train-test splitting, and hyperparameter tuning.
sklearn.decomposition	Dimensionality reduction methods like PCA and Truncated SVD.
sklearn.feature_extraction	Tools for extracting features from text or images.

#### 4. PyTorch:

PyTorch is a machine learning library based on the Torch library, used for applications such as computer vision and natural language processing, originally developed by Meta AI and now part of the Linux Foundation umbrella. It is one of the most popular deep learning frameworks, alongside others such as TensorFlow and PaddlePaddle, offering free and open-source software released under the modified BSD license. Although the Python interface is more polished and the primary focus of development, PyTorch also has a C++ interface.



- **Key features:**

- **Dynamic Computational Graph:**

Unlike some other frameworks, PyTorch builds the computational graph on-the-fly as you execute your code, making it more flexible and intuitive for debugging and experimentation.

- **GPU Acceleration:**

PyTorch efficiently utilizes GPUs for faster computations, which is crucial for training large deep learning models.

- **Pythonic:**  
PyTorch integrates well with Python's ecosystem, allowing you to use familiar Python libraries like NumPy, SciPy, and Pandas.
- **Strong Community:**  
PyTorch boasts a large and active community, providing support, tutorials, and a wealth of pre-trained models.
- **Production Ready:**  
PyTorch is not only suitable for research but also provides tools for deploying models in production environments.
- **Why use PyTorch?**
  - **Easy to learn:** PyTorch's syntax is similar to Python, making it easier for developers to learn.
  - **Flexible:** PyTorch supports dynamic computational graphs, which allows network behavior to change at runtime.
  - **Fast:** PyTorch can perform accelerated computations on GPUs and TPUs.
  - **Community support:** PyTorch has a large community of users who have built tools and libraries to extend the framework.
- **Subpackages**

Subpackages	Description
torch	Core module for tensors, mathematical operations, and basic utilities.
torch.nn	Provides tools for building neural networks, including layers and activations.
torch.optim	Contains optimization algorithms like SGD, Adam, and RMSprop.
torch.autograd	Enables automatic differentiation for gradient computation.
torch.utils	Utility functions for data handling, model saving, and loading.
torchvision	Supports image processing, pretrained models, and datasets for computer vision.
torchaudio	Provides tools for audio processing and datasets for speech recognition tasks.
torch	Core module for tensors, mathematical operations, and basic utilities.
torch.nn	Provides tools for building neural networks, including layers and activations.
torch.optim	Contains optimization algorithms like SGD, Adam, and RMSprop.
torchtext	Focused on natural language processing with text datasets and preprocessing.
torch.distributed	Enables distributed training across multiple GPUs and nodes.

## 5. TensorFlow:

TensorFlow is an open-source library that helps data scientists and engineers develop and deploy machine learning applications. It's used for tasks like image recognition, natural language processing, and handwriting recognition.



- **How it works**

- TensorFlow uses data flow graphs to describe machine learning algorithms.
- Nodes in the graph represent mathematical operations, while edges represent multidimensional data arrays (tensors).
- TensorFlow can train and run models on CPUs, GPUs, and TPUs.
- TensorFlow uses a high-level API, so complex coding isn't needed to prepare a neural network.

- **Key features:**

- Flexibility:

TensorFlow offers a flexible architecture, allowing you to build and experiment with diverse models, from simple linear regression to complex neural networks.

- Scalability:

You can train TensorFlow models on a single CPU, multiple CPUs, GPUs, or even distributed clusters, making it suitable for both small and large-scale projects.

- Eager Execution:

TensorFlow 2.0 introduced eager execution, which enables you to execute operations immediately and debug your code more easily.

- Keras Integration:

Keras, a high-level API for building neural networks, is tightly integrated with TensorFlow, simplifying the model development process.

- **Subpackages**

Subpackages	Description
tensorflow	Core package for tensor operations, graph construction, and execution.
tensorflow.keras	High-level API for building and training deep learning models.
tensorflow.data	Tools for creating efficient input pipelines for large datasets.
tensorflow.nn	Low-level neural network operations, such as activations and convolution.
tensorflow.train	Utilities for optimizing and saving models during training.
tensorflow.linalg	Linear algebra operations like matrix multiplication and decomposition.

tensorflow.math	Provides advanced mathematical operations beyond basic arithmetic.
tensorflow.compat	Ensures compatibility between TensorFlow versions.
tensorflow.tpu	Tools for optimizing training on Tensor Processing Units (TPUs).

## 6. Seaborn:

"Seaborn" refers to a Python library primarily used for data visualization, particularly for creating statistically informative and aesthetically pleasing graphs, built on top of the Matplotlib library; essentially, it provides a high-level interface for generating complex statistical plots with ease.



- **Key features:**

- **Flexibility:**

Seaborn provides a wide range of built-in plot types, allowing users to create visualizations such as scatter plots, bar plots, violin plots, and box plots with minimal code.

- **Ease of Use:**

Built on top of Matplotlib, Seaborn simplifies complex visualization tasks with higher-level interfaces and pre-set styles.

- **Beautiful Themes:**

Offers several aesthetic styles and color palettes to improve the look of plots, such as darkgrid, whitegrid, and dark.

- **Integration with Pandas:**

Works seamlessly with Pandas DataFrames, making it easy to visualize data directly from structured datasets.

- **Categorical Plots:**

Specialized for categorical data visualization, including catplot, stripplot, and swarmplot.

- **Subpackages**

Subpackage	Description
seaborn.relational	Functions like scatterplot and lineplot to visualize relationships between variables.
seaborn.categorical	Functions like boxplot, violinplot, and barplot for visualizing categorical data.
seaborn.distributions	Tools like histplot, kdeplot, and ecdfplot to explore data distributions.
seaborn.matrix	Functions like heatmap for visualizing matrices and correlations.

seaborn.axisgrid	FacetGrid and PairGrid for plotting subsets of data in a grid layout.
seaborn.palettes	Utilities for managing and customizing color palettes.
seaborn.regression	Functions like regplot and lmpplot for regression analysis visualizations.
seaborn.utils	Helper functions for adjusting plots and data manipulation.
seaborn.distributions	Tools like histplot, kdeplot, and ecdfplot to explore data distributions.
seaborn.matrix	Functions like heatmap for visualizing matrices and correlations.

## 7. Plotly:

Plotly is a free, open-source company that provides tools for data visualization, analytics, and statistics. It offers a variety of graphing libraries for different programming languages, including Python, R, and JavaScript.



- **Features**

- **Interactive Plots:** Allows zooming, panning, and other interactive features directly in the plots.
- **Wide Range of Charts:** Includes standard charts, 3D visualizations, geographic maps, and more.
- **Cross-Library Compatibility:** Works well with pandas, NumPy, and other data libraries.
- **Customizable Visualizations:** Offers detailed control over plot aesthetics and layouts.
- **Dash Integration:** Enables creation of full-fledged dashboards and web apps using Python.
- **Export Options:** Allows exporting plots in PNG, SVG, and other high-quality formats.
- **Web-Based Rendering:** Uses JavaScript-based rendering for dynamic, browser-compatible visuals.

- **Subpackages**

Subpackage	Description
plotly.graph_objects	Provides a low-level API for creating detailed and customizable visualizations.
plotly.express	High-level API for creating simple and quick plots with minimal code.

plotly.subplots	Tools for creating multi-panel visualizations with subplots.
plotly.figure_factory	Generates specialized plots like dendrograms, table plots, and heatmaps.
plotly.io	Handles input/output operations for saving and displaying visualizations.
plotly.colors	Provides utilities for working with color scales and palettes.
plotly.validators	Contains tools for validating plot inputs and configurations.
plotly.tools	Legacy module with helper functions for working with Plotly objects.

Faculty Signature: \_\_\_\_\_

Date: \_\_\_\_\_