Introduction to Libraries in Python

Python is renowned for its extensive collection of libraries that extend its capabilities beyond the basic functions of the language. These libraries are precompiled codes, which can be used in various programs for specific operations, saving time and effort for developers.

What is a Library?

In the context of programming, a library is akin to a collection of books in a traditional library. It's a set of pre-written code modules that serve a specific functionality. These modules are reusable and can be integrated into a programmer's code, enhancing the development process and functionality of the software.

Python Standard Library

The Python Standard Library is a powerful toolkit included with every Python installation. It contains over 200 core modules that provide access to system functionality and are essential for programming tasks ranging from file I/O to data serialization.

Popular Python Libraries

Here are some of the most popular Python libraries that have become staples in various domains:

- Matplotlib

  A plotting library for creating static, interactive, and animated visualizations in Python. It's highly useful for data visualization and graphical plotting.

- NumPy:

  A library for numerical computing, providing support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays

- Pandas:

  An open-source library providing high-performance, easy-to-use data structures, and data analysis tools. It's particularly well-suited for structured data operations and manipulations.

- OSGeo:

  `OSGeo`, the Open Source Geospatial Foundation, supports and promotes the collaborative development of open geospatial technologies and data. The OSGeo library in Python is a collection of tools for processing geospatial data, supporting the likes of GDAL/OGR, Proj.4, GEOS, etc

- Seaborn:

  `Seaborn` is a statistical data visualization library in Python. It's based on matplotlib and provides a high-level interface for drawing attractive and informative statistical graphics

- Scikit-learn

`Scikit-learn` is a powerful Python library for machine learning. It provides simple and efficient tools for data mining and data analysis. It's built on `NumPy`, `SciPy`, and `matplotlib`, and it's open source, commercially usable - BSD licensed:

- TensorFlow:

  Developed by **Google**, this open-source library is used for high-level computations and is instrumental in machine learning and deep learning algorithms

- Xarray:

  `Xarray` is an open-source project and Python library that makes working with labelled multi-dimensional arrays simple, efficient, and fun! Xarray introduces labels in the form of dimensions, coordinates, and attributes on top of raw NumPy-like arrays, which makes data more self-describing and interoperable

Using pip to Install Python Libraries

`pip` is the library installer for Python. You can use it to install Libraries from the Python library Index (PyPI) and other indexes.

How to Install a Libraries with pip

To install a library, open your command-line interface and enter:

```
pip install package-name
```

```
pip install xarray
```

```
Requirement already satisfied: xarray in /usr/local/lib/python3.10/dist-packages (2023.7.0)
Requirement already satisfied: numpy>=1.21 in /usr/local/lib/python3.10/dist-packages (from xarray)
Requirement already satisfied: pandas>=1.4 in /usr/local/lib/python3.10/dist-packages (from xarray)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.10/dist-packages (from xarr
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (fr
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from panda
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-date
```

How to List Installed Python Packages

```
pip list
```

```
tninc                          8.2.3
threadpoolctl                  3.5.0
tifffile                       2024.4.24
tinycss2                       1.3.0
tokenizers                     0.19.1
toml                           0.10.2
tomli                          2.0.1
toolz                          0.12.1
torch                          2.2.1+cu121
torchaudio                     2.2.1+cu121
torchdata                      0.7.1
torchsummary                   1.5.1
torchtext                      0.17.1
torchvision                    0.17.1+cu121
tornado                        6.3.3
tqdm                           4.66.2
traitlets                      5.7.1
traittypes                     0.2.1
transformers                   4.40.1
triton                         2.2.0
tweepy                         4.14.0
typer                          0.9.4
types-pytz                     2024.1.0.20240417
types-setuptools               69.5.0.20240423
typing_extensions              4.11.0
tzdata                         2024.1
tzlocal                        5.2
uc-micro-py                    1.0.3
uritemplate                    4.1.1
urllib3                        2.0.7
vega-datasets                  0.9.0
wadllib                        1.3.6
wasabi                         1.1.2
wcwidth                        0.2.13
weasel                         0.3.4
webcolors                      1.13
webencodings                   0.5.1
websocket-client               1.8.0
Werkzeug                       3.0.2
wheel                          0.43.0
widgetsnbextension             3.6.6
wordcloud                      1.9.3
wrapt                          1.14.1
xarray                         2023.7.0
xarray-einstats                0.7.0
xgboost                        2.0.3
xlrd                           2.0.1
xyzservices                    2024.4.0
yarl                           1.9.4
yellowbrick                    1.5
yfinance                       0.2.38
zict                           3.0.0
zipp                           3.18.1
```

Using Libraries in Python

After installation, you can use the Library in your Python scripts as follows:

```
import numpy as np
```

This line imports the NumPy library and aliases it as np, allowing you to use its functions and modules with the np prefix.

```
from osgeo import gdal
```

Upgrading a Library

To upgrade an existing Library to the latest version, use:

```ba pip install --upgrade package-namesh

```
pip install --upgrade xarray
```

```
Requirement already satisfied: xarray in /usr/local/lib/python3.10/dist-packages (2023.7.0)
Collecting xarray
  Downloading xarray-2024.3.0-py3-none-any.whl (1.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.1/1.1 MB 9.9 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.23 in /usr/local/lib/python3.10/dist-packages (from xarray)
Requirement already satisfied: packaging>=22 in /usr/local/lib/python3.10/dist-packages (from xarray
Requirement already satisfied: pandas>=1.5 in /usr/local/lib/python3.10/dist-packages (from xarray)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (fr
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from panda
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-date
Installing collected packages: xarray
  Attempting uninstall: xarray
    Found existing installation: xarray 2023.7.0
    Uninstalling xarray-2023.7.0:
      Successfully uninstalled xarray-2023.7.0
Successfully installed xarray-2024.3.0
```

Uninstalling a Library

If you need to remove a Library, use:

```
pip uninstall package-name
```

```
pip uninstall xarray
```

```
Found existing installation: xarray 2024.3.0
Uninstalling xarray-2024.3.0:
  Would remove:
    /usr/local/lib/python3.10/dist-packages/xarray-2024.3.0.dist-info/*
    /usr/local/lib/python3.10/dist-packages/xarray/*
Proceed (Y/n)? Y
  Successfully uninstalled xarray-2024.3.0
```

DataFrame
```

A DataFrame is a data structure that organizes data into a 2-dimensional table of rows and columns, much like a spreadsheet. DataFrames are one of the most common data structures used in modern data analytics because they are a flexible and intuitive way of storing and working with data.

Importing DataFrame

```
import pandas as pd

df= pd.read_csv('Weather_Data.csv')

df
```

	Date	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGust
0	2/1/2008	19.5	22.4	15.6	6.2	0.0	W	
1	2/2/2008	19.5	25.6	6.0	3.4	2.7	W	
2	2/3/2008	21.6	24.5	6.6	2.4	0.1	W	
3	2/4/2008	20.2	22.8	18.8	2.2	0.0	W	
4	2/5/2008	19.7	25.7	77.4	4.8	0.0	W	
...	...	...	...	...	...	...	...	
3266	6/21/2017	8.6	19.6	0.0	2.0	7.8	SSE	
3267	6/22/2017	9.3	19.2	0.0	2.0	9.2	W	
3268	6/23/2017	9.4	17.7	0.0	2.4	2.7	W	
3269	6/24/2017	10.1	19.3	0.0	1.4	9.3	W	
3270	6/25/2017	7.6	19.3	0.0	3.4	9.4	W	

3271 rows × 22 columns

df.head()

```
df.head()
```

	Date	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpee
0	2/1/2008	19.5	22.4	15.6	6.2	0.0	W	4
1	2/2/2008	19.5	25.6	6.0	3.4	2.7	W	4
2	2/3/2008	21.6	24.5	6.6	2.4	0.1	W	4
3	2/4/2008	20.2	22.8	18.8	2.2	0.0	W	4
4	2/5/2008	19.7	25.7	77.4	4.8	0.0	W	4

5 rows × 22 columns

```
df.head(10)
```

	Date	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpe
0	2/1/2008	19.5	22.4	15.6	6.2	0.0	W	
1	2/2/2008	19.5	25.6	6.0	3.4	2.7	W	
2	2/3/2008	21.6	24.5	6.6	2.4	0.1	W	
3	2/4/2008	20.2	22.8	18.8	2.2	0.0	W	
4	2/5/2008	19.7	25.7	77.4	4.8	0.0	W	
5	2/6/2008	20.2	27.2	1.6	2.6	8.6	W	
6	2/7/2008	18.6	26.3	6.2	5.2	5.2	W	
7	2/8/2008	17.2	22.3	27.6	5.8	2.1	W	
8	2/9/2008	16.4	20.8	12.6	4.8	3.0	W	
9	2/10/2008	14.6	24.2	8.8	4.4	10.1	W	

10 rows × 22 columns

The `df.head()` function in Python is used with a DataFrame object, typically from the `pandas` library. It returns the first **n** rows of the DataFrame for quick inspection. By default, if no parameter is specified, `df.head()` will return the first **5** rows. You can also specify the number of rows you want to display by passing an integer as an argument, like `df.head(10)` to display the first 10 rows.

df.tail()

```
df.tail()
```

	Date	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGust
3266	6/21/2017	8.6	19.6	0.0	2.0	7.8	SSE	
3267	6/22/2017	9.3	19.2	0.0	2.0	9.2	W	
3268	6/23/2017	9.4	17.7	0.0	2.4	2.7	W	
3269	6/24/2017	10.1	19.3	0.0	1.4	9.3	W	
3270	6/25/2017	7.6	19.3	0.0	3.4	9.4	W	

5 rows × 22 columns

```
df.tail(10)
```

	Date	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGust
3261	6/16/2017	11.9	17.3	0.0	1.6	0.2	WNW	
3262	6/17/2017	13.2	19.1	0.0	1.0	0.2	SSW	
3263	6/18/2017	11.3	18.0	1.8	2.0	6.3	S	
3264	6/19/2017	11.2	18.3	0.4	2.2	1.9	SSW	
3265	6/20/2017	11.3	20.0	4.4	2.2	5.8	W	
3266	6/21/2017	8.6	19.6	0.0	2.0	7.8	SSE	
3267	6/22/2017	9.3	19.2	0.0	2.0	9.2	W	
3268	6/23/2017	9.4	17.7	0.0	2.4	2.7	W	
3269	6/24/2017	10.1	19.3	0.0	1.4	9.3	W	
3270	6/25/2017	7.6	19.3	0.0	3.4	9.4	W	

10 rows × 22 columns

The `df.tail()` function in Python, when used with a DataFrame object from the `pandas` library, returns the last **n** rows of the DataFrame. This is useful for quickly verifying data, especially after sorting or appending rows. By default, if no parameter is specified, `df.tail()` will return the last **5** rows. You can specify the number of rows you want to display by passing an integer as an argument, like `df.tail(10)` to display the last 10 rows

df.info()

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3271 entries, 0 to 3270
Data columns (total 22 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Date           3271 non-null   object
```

```
 1   MinTemp        3271 non-null   float64
 2   MaxTemp        3271 non-null   float64
 3   Rainfall       3271 non-null   float64
 4   Evaporation    3271 non-null   float64
 5   Sunshine       3271 non-null   float64
 6   WindGustDir    3271 non-null   object
 7   WindGustSpeed  3271 non-null   int64
 8   WindDir9am     3271 non-null   object
 9   WindDir3pm     3271 non-null   object
 10  WindSpeed9am   3271 non-null   int64
 11  WindSpeed3pm   3271 non-null   int64
 12  Humidity9am    3271 non-null   int64
 13  Humidity3pm    3271 non-null   int64
 14  Pressure9am    3271 non-null   float64
 15  Pressure3pm    3271 non-null   float64
 16  Cloud9am       3271 non-null   int64
 17  Cloud3pm       3271 non-null   int64
 18  Temp9am        3271 non-null   float64
 19  Temp3pm        3271 non-null   float64
 20  RainToday      3271 non-null   object
 21  RainTomorrow   3271 non-null   object
dtypes: float64(9), int64(7), object(6)
memory usage: 562.3+ KB
```

The `df.info()` method in Python, when used with a DataFrame object from the `pandas` library, prints a concise summary of the DataFrame. This summary includes information such as the index dtype, column dtypes, non-null values, and memory usage. It's a valuable method for getting a quick overview of the DataFrame, especially to understand its structure, the data types of each column, and to identify columns with missing values

df.columns

```
df.columns
```

```
Index(['Date', 'MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine',
       'WindGustDir', 'WindGustSpeed', 'WindDir9am', 'WindDir3pm',
       'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm',
       'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am',
       'Temp3pm', 'RainToday', 'RainTomorrow'],
      dtype='object')
```

```
print(df.columns)
```

```
Index(['Date', 'MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine',
       'WindGustDir', 'WindGustSpeed', 'WindDir9am', 'WindDir3pm',
       'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm',
       'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am',
       'Temp3pm', 'RainToday', 'RainTomorrow'],
      dtype='object')
```

The `df.columns` attribute in Python, when used with a DataFrame object from the `pandas` library, returns the column labels of the DataFrame as an Index object. This attribute is useful for viewing the column names, modifying them, or using them in further analysis and data manipulation tasks

df.shape

```
df.shape
```

```
(3271, 22)
```

```
print(df.shape)
```

```
(3271, 22)
```

The `df.shape` attribute in Python, when used with a DataFrame object from the `pandas` library, returns a tuple representing the dimensionality of the DataFrame. This tuple contains two elements: the number of rows and the number of columns. It does not require parentheses because it is an attribute, not a method

My First plot!!! (Introduction to Matplotlib)

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

```
x=np.linspace(0,10)
y= x**2
plt.plot(x,y, 'red')
plt.xlabel('X-axis label')
plt.ylabel('Y-axis label')
plt.title('My First plot')
plt.show()
```

`np.linspace`

The `np.linspace` function in Python, provided by the NumPy library, is used to create an array of evenly spaced numbers over a specified interval. The function returns `num` evenly spaced samples, calculated over the interval `[start, stop]`. The endpoint of the interval can optionally be excluded.

`plt.plot`

The `plt.plot` function in Python is a versatile command from the Matplotlib library used to create a variety of plots, including line plots. When you call `plt.plot`, you can specify the data points you want to plot, the type of plot you want, and various formatting options like color, marker style, and line style.

`plt.xlabel`

The `plt.xlabel` function in Matplotlib's Pyplot module is used to set the label for the x-axis of a plot.

`plt.ylabel`

The `plt.ylabel` function in Matplotlib's Pyplot module is used to set the label for the y-axis of a plot.

`plt.title`

The `plt.title` function in Matplotlib's Pyplot module is used to set the title of a plot. The title is displayed above the plot.

`plt.show()`

The `plt.show()` function in Python, which is part of the Matplotlib library, is used to display all figures and block until the figures have been closed. When you use Matplotlib to plot graphs, the figures are not displayed until you call `plt.show()`. This function will open a window with the graph and wait for any user interactions if the program is not running in an interactive mode.

Conclusion

Python libraries are a testament to the language's versatility and the collaborative spirit of the programming community. These libraries are just the tip of the iceberg in Python's vast repository of resources. Whether you're manipulating multi-dimensional arrays with `Xarray`, building machine learning models with `Scikit-learn`, creating beautiful visualizations with `Seaborn`, or processing geospatial data with `OSGeo`, Python has a library for almost every need. They simplify complex tasks, promote code reuse, and enable programmers to stand on the shoulders of giants. Remember, `pip` is a powerful tool that simplifies the process of managing Python packages.
Happy coding!