Exploring Python libraries

Python is an ocean of libraries that serve various purposes

Top 10 Python Libraries for machine learning

1. TensorFlow
2. Scikit-Learn
3. Numpy
4. Keras
5. PyTorch
6. LightGBM
7. Eli5
8. SciPy

TensorFlow

- TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.
- This library was developed by Google in collaboration with Brain Team. TensorFlow is a part of almost every Google application for machine learning.
- TensorFlow works like a computational library for writing new algorithms that involve a large number of tensor operations, since neural networks can be easily expressed as computational graphs they can be implemented using TensorFlow as a series of operations on Tensors.
- Plus, tensors are N-dimensional matrices which represent your data.

Features of TensorFlow

TensorFlow is optimized for speed, it makes use of techniques like XLA for quick linear algebra operations.

1. Responsive Construct

With TensorFlow, we can easily visualize each and every part of the graph which is not an option while using Numpy or SciKit.

2. Flexible

One of the very important Tensorflow Features is that it is flexible in its operability, meaning it has modularity and the parts of it which you want to make standalone, it offers you that option.

3. Easily Trainable

It is easily trainable on CPU as well as *GPU* for distributed computing.

4. Parallel Neural Network Training

TensorFlow offers pipelining in the sense that you can train multiple *neural networks*and multiple GPUs which makes the models very efficient on large-scale systems.

5. Large Community

Needless to say, if it has been developed by Google, there already is a large team of software engineers who work on stability improvements continuously.

6. Open Source

The best thing about this machine learning library is that it is open source so anyone can use it as

long as they have internet connectivity.

Uses of TensorFlow?

You are using TensorFlow daily but indirectly with applications like Google Voice Search or Google Photos. These are the applications of TensorFlow.

All the libraries created in TensorFlow are written in C and C++. However, it has a complicated front-end for Python. Your Python code will get compiled and then executed on TensorFlow distributed execution engine built using C and C++.

Scikit-Learn

What Is 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.

There are a lot of changes being made in this library. One modification is the cross-validation feature, providing the ability to use more than one metric. Lots of training methods like logistics regression and nearest neighbors have received some little improvements.

Features Of Scikit-Learn

1. Cross-validation: There are various methods to check the accuracy of supervised models on unseen data.

2. Unsupervised learning algorithms: Again there is a large spread of algorithms in the offering – starting from clustering, factor analysis, principal component analysis to unsupervised neural networks.

3. Feature extraction: Useful for extracting features from images and text (e.g. Bag of words)

Where are we using Scikit-Learn?

It contains a numerous number of algorithms for implementing standard machine learning and data mining tasks like reducing dimensionality, classification, regression, clustering, and model selection.

Numpy

What Is Numpy?

Numpy is considered as one of the most popular machine learning library in Python.

TensorFlow and other libraries uses Numpy internally for performing multiple operations on Tensors. Array interface is the best and the most important feature of Numpy.

Features Of Numpy

1. **Interactive:** Numpy is very interactive and easy to use.
2. **Mathematics:** Makes complex mathematical implementations very simple.
3. **Intuitive:** Makes coding real easy and grasping the concepts is easy.
4. **Lot of Interaction:** Widely used, hence a lot of open source contribution.

Uses of Numpy?

This interface can be utilized for expressing images, sound waves, and other binary raw streams as an array of real numbers in N-dimensional.

Keras

What Is Keras?

Keras is considered as one of the coolest machine learning libraries in Python. It provides an easier mechanism to express neural networks. Keras also provides some of the best utilities for compiling models, processing data-sets, visualization of graphs, and much more.

In the backend, Keras uses either Theano or TensorFlow internally. Some of the most popular neural networks like CNTK can also be used. Keras is comparatively slow when we compare it with other machine learning libraries. Because it creates a computational graph by using back-end infrastructure and then makes use of it to perform operations. All the models in Keras are portable.

Features Of Keras

1. It runs smoothly on both CPU and GPU.
2. Keras supports almost all the models of a neural network – fully connected, convolutional, pooling, recurrent, embedding, etc. Furthermore, these models can be combined to build more complex models.
3. Keras, being modular in nature, is incredibly expressive, flexible, and apt for innovative research.
4. Keras is a completely Python-based framework, which makes it easy to debug and explore.

Where are we using Keras?

1. You are already constantly interacting with features built with Keras — it is in use at Netflix, Uber, Yelp, Instacart, Zocdoc, Square, and many others. It is especially popular among startups that place deep learning at the core of their products.
2. Keras contains numerous implementations of commonly used neural network building blocks such as layers, objectives, activation functions, optimizers and a host of tools to make working with image and text data easier.
3. Plus, it provides many pre-processed data-sets and pre-trained models like MNIST, VGG, Inception, SqueezeNet, ResNet etc.
4. Keras is also a favorite among deep learning researchers, coming in at #2. Keras has also been adopted by researchers at large scientific organizations, in partic,ular CERN and NASA.

PyTorch

What Is PyTorch?

PyTorch is the largest machine learning library that allow developers to perform tensor computations wan ith acceleration of GPU, creates dynamic computational graphs, and calculate gradients automatically. Other than this, PyTorch offers rich APIs for solving application issues related to neural networks.

This machine learning library is based on Torch, which is an open source machine library implemented in C with a wrapper in Lua.

Features Of PyTorch

Hybrid Front-End

A new hybrid front-end provides ease-of-use and flexibility in eager mode, while seamlessly transitioning to graph mode for speed, optimization, and functionality in C++ runtime environments.

Distributed Training

Optimize performance in both research and production by taking advantage of native support for asynchronous execution of collective operations and peer-to-peer communication that is accessible from Python and C++.

Python First

PyTorch is not a Python binding into a monolithic C++ framework. It’s built to be deeply integrated into Python so it can be used with popular libraries and packages such as Cython and Numba.

Libraries And Tools

An active community of researchers and developers have built a rich ecosystem of tools and libraries for extending PyTorch and supporting development in areas from computer vision to reinforcement learning.

Applications of PyTorch?

1. PyTorch is primarily used for applications such as natural language processing.
2. It is primarily developed by Facebook’s artificial-intelligence research group and Uber’s “Pyro” software for probabilistic programming is built on it.
3. PyTorch is outperforming TensorFlow in multiple ways and it is gaining a lot of attention in the recent days.

LightGBM

What Is LightGBM?

Gradient Boosting is one of the best and most popular machine learning library, which helps developers in building new algorithms by using redefined elementary models and namely decision trees. Therefore, there are special libraries which are available for fast and efficient implementation of this method.

These libraries are LightGBM, XGBoost, and CatBoost. All these libraries are competitors that helps in solving a common problem and can be utilized in almost the similar manner.

Features of LightGBM

1. Very fast computation ensures high production efficiency.
2. Intuitive, hence makes it user friendly.
3. Faster training than many other deep learning libraries.
4. Will not produce errors when you consider NaN values and other canonical values.

What are the applications of LightGBM?

These library provides provide highly scalable, optimized, and fast implementations of gradient boosting, which makes it popular among machine learning developers. Because most of the machine learning full stack developers won machine learning competitions by using these algorithms.

What Is Eli5?

- Most often the results of machine learning model predictions are not accurate, and Eli5 machine learning library built in Python helps in overcoming this challenge. It is a combination of visualization and debug all the machine learning models and track all working steps of an algorithm.

Features of Eli5

- Moreover, Eli5 supports wother libraries XGBoost, lightning, scikit-learn, and sklearn-crfsuite libraries.

What are the applications of Eli5?

1. Mathematical applications which requires a lot of computation in a short time.
2. Eli5 plays a vital role where there are dependencies with other Python packages.
3. Legacy applications and implementing newer methodologies in various fields.

SciPy

What Is SciPy?

SciPy is a machine learning library for application developers and engineers. However, you still need to know the difference between SciPy library and SciPy stack. SciPy library contains modules for optimization, linear algebra, integration, and statistics.

Features Of SciPy

The main feature of SciPy library is that it is developed using NumPy, and its array makes the most use of NumPy.

In addition, SciPy provides all the efficient numerical routines like optimization, numerical integration, and many others using its specific submodules.

All the functions in all submodules of SciPy are well documented.

Applications of SciPy?

- SciPy is a library that uses NumPy for the purpose of solving mathematical functions. SciPy uses NumPy arrays as the basic data structure, and comes with modules for various commonly used tasks in scientific programming.
- Tasks including linear algebra, integration (calculus), ordinary differential equation solving and signal processing execute easily by SciPy.

Theano

What Is Theano?

Theano is a computational framework machine learning library in Python for computing multidimensional arrays. Theano works similar to TensorFlow, but it not as efficient as TensorFlow. Because of its inability to fit into production environments.

Moreover, Theano can also be used on a distributed or parallel environments just similar to TensorFlow.

Features Of Theano

1. Tight integration with NumPy – Ability to use completely NumPy arrays in Theano-compiled functions.
2. Transparent use of a GPU – Perform data-intensive computations much faster than on a CPU.
3. Efficient symbolic differentiation – Theano does your derivatives for functions with one or many inputs.
4. Speed and stability optimizations – Get the right answer for log(1+x) even when x is very tiny. This is just one of the examples to show the stability of Theano.
5. Dynamic C code generation – Evaluate expressions faster than ever before, thereby, increasing efficiency by a lot.
6. Extensive unit-testing and self-verification – Detect and diagnose multiple types of errors and ambiguities in the model.

Where are we using Theano?

- The actual syntax of Theano expressions is symbolic, which can be off putting to beginners used to normal software development. Specifically, expression are defined in the abstract sense, compiled and later actually used to make calculations.
- It specifically handles the types of computation for large neural network algorithms in Deep Learning. It was one of the first libraries of its kind (development started in 2007) and is an industry standard for Deep Learning research and development.
- Theano is the strength of multiple neural network projects today and the popularity of Theano is only growing with time.

Pandas

What Is Pandas?

Pandas is a machine learning library in Python that provides data structures of high-level and a wide variety of tools for analysis. One of the great feature of this library is the 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.

Features Of Pandas

Pandas make sure that the entire process of manipulating data will be easier. Support for operations such as Re-indexing, Iteration, Sorting, Aggregations, Concatenations and Visualizations are among the feature highlights of Pandas.

Applications of Pandas?

Currently, there are fewer releases of pandas library which includes hundred of new features, bug fixes, enhancements, and changes in API. The improvements in pandas regards its ability to group and sort data, select best suited output for the apply method, and provides support for performing custom types operations.

Data Analysis among everything else takes the highlight when it comes to usage of Pandas. But, Pandas when used with other libraries and tools ensure high functionality and good amount of flexibility.

Python Read Text File

with open('readme.txt') as f:

lines = f.readlines()

Steps for reading a text file in Python

1. First, open a text file for reading by using the open() function.
2. Second, read text from the text file using the file read(), readline(), or readlines() method of the file object.
3. Third, close the file using the file close() method.

1) open() function

- The open() function has many parameters but you'll be focusing on the first two.
- The path_to_file parameter specifies the path to the text file.
- If the file is in the same folder as the program, you just need to specify the name of the file. Otherwise, you need to specify the path to the file.
- To specify the path to the file, you use the forward-slash ('/') even if you're working in Windows.
- For example, if the file is readme.txt stored in the sample folder as the program, you need to specify the path to the file as c:/sample/readme.txt

The mode is an optional parameter. It’s a string that specifies the mode in which you want to open the file.

2) Reading text methods

- The file object provides you with three methods for reading text from a text file:
- read() – read all text from a file into a string. This method is useful if you have a small file and you want to manipulate the whole text of that file.
- readline() – read the text file line by line and return all the lines as strings.
- readlines() – read all the lines of the text file and return them as a list of strings.

Mode	Description
'r'	Open for text file for reading text
'w'	Open a text file for writing text
'a'	Open a text file for appending text

```
with open('the-zen-of-python.txt') as f:
    contents = f.read()
    print(contents)
```

```
Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
...
```

```
lines = []
with open('the-zen-of-python.txt') as f:
    lines = f.readlines()
count = 0
for line in lines:
    count += 1
    print(f'line {count}: {line}’)
```

```
line 1: Beautiful is better than ugly.

line 2: Explicit is better than implicit.

line 3: Simple is better than complex.
...
```

```
with open('the-zen-of-python.txt') as f:
    while True:
        line = f.readline()
        if not line:
            break
        print(line)
```

```
Explicit is better than implicit.

Simple is better than complex.

Complex is better than complicated.
...
```

What Is a CSV File?

- A CSV file (Comma Separated Values file) is a type of plain text file that uses specific structuring to arrange tabular data. Because it's a plain text file, it can contain only actual text data.
- The structure of a CSV file is given away by its name. Normally, CSV files use a comma to separate each specific data value
  CSV File

column 1 name,column 2 name, column 3 name

first row data 1,first row data 2,first row data 3

second row data 1,second row data 2,second row data 3

The separator character is called a delimiter, and the comma is not the only one used. Other popular delimiters include the tab (\t), colon (:) and semi-colon (;) characters. Properly parsing a CSV file requires us to know which delimiter is being used.

Where Do CSV Files Come From?

- CSV files are normally created by programs that handle large amounts of data. They are a convenient way to export data from spreadsheets and databases as well as import or use it in other programs. For example, you might export the results of a data mining program to a CSV file and then import that into a spreadsheet to analyze the data, generate graphs for a presentation, or prepare a report for publication.

- CSV files are very easy to work with programmatically. Any language that supports text file input and string manipulation (like Python) can work with CSV files directly.

Parsing CSV Files With Python's Built-in CSV Library

- The csv library provides functionality to both read from and write to CSV files. Designed to work out of the box with Excel-generated CSV files, it is easily adapted to work with a variety of CSV formats.

Reading CSV Files With csv

- Reading from a CSV file is done using the reader object.

CSV

name,department,birthday month

John Smith,Accounting,November

Erica Meyers,IT,March

Output:

Column names are name, department, birthday month

John Smith works in the Accounting department, and was born in November.

Erica Meyers works in the IT department, and was born in March.

Processed 3 lines.

```
import csv

with open('employee_birthday.txt') as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')
    line_count = 0
    for row in csv_reader:
        if line_count == 0:
            print(f'Column names are {", ".join(row)}')
            line_count += 1
        else:
            print(f'\t{row[0]} works in the {row[1]} department, and was born in {row[2]}.')
            line_count += 1
    print(f'Processed {line_count} lines.')
```

Reading an excel file using Python

- Using xlrd module, one can retrieve information from a spreadsheet. For example, reading, writing or modifying the data can be done in Python.
- xlrd module is used to extract data from a spreadsheet.

Command to install xlrd module:

pip install xlrd

Code #1: Extract a specific cell

```
# Reading an excel file using Python
import xlrd
# Give the location of the file
loc = ("path of file")
# To open Workbook
wb = xlrd.open_workbook(loc)
sheet = wb.sheet_by_index(0)
# For row 0 and column 0
print(sheet.cell_value(0, 0))
```

Output :

'NAME'

Code #2: Extract the number of rows
Code #3: Extract the number of columns

```
# Program to extract number
# of rows using Python
import xlrd

# Give the location of the file
loc = ("path of file")

wb = xlrd.open_workbook(loc)
sheet = wb.sheet_by_index(0)
sheet.cell_value(0, 0)

# Extracting number of rows
print(sheet.nrows)
```

```
# Program to extract number of
# columns in Python
import xlrd

loc = ("path of file")

wb = xlrd.open_workbook(loc)
sheet = wb.sheet_by_index(0)

# For row 0 and column 0
sheet.cell_value(0, 0)

# Extracting number of columns
print(sheet.ncols)
```

Code #4 : Extracting all columns name

```
# Program extracting all columns
# name in Python
import xlrd
loc = ("path of file")
wb = xlrd.open_workbook(loc)
sheet = wb.sheet_by_index(0)
# For row 0 and column 0
sheet.cell_value(0, 0)
for i in range(sheet.ncols):
        print(sheet.cell_value(0, i))
```

Output :

NAME
SEMESTER
ROLL NO

```
# Program extracting first column
import xlrd
loc = ("path of file")
wb = xlrd.open_workbook(loc)
sheet = wb.sheet_by_index(0)
sheet.cell_value(0, 0)
for i in range(sheet.nrows):
        print(sheet.cell_value(i, 0))
```

Sampling in Python

- Many a times the dataset we are dealing with can be too large to be handled in python. A workaround is to take random samples out of the dataset and work on it.
- There are situations where sampling is appropriate, as it gives a near representations of the underlying population.

```
import pandas as pd

Online_Retail=pd.read_csv("datasets\\Online Retail Sales Data\\Online Retail.csv",
encoding = "ISO-8859-1")
Online_Retail.shape
```

Out[1]:

(541909, 8)

Using function .sample() on our data set we have taken a random sample of 1000 rows out of total 541909 rows of full data.

```
sample_data=Online_Retail.sample(n=1000,replace="False")
sample_data.shape
Out[2]:
(1000, 8)
```

Using function .sample() on our data set we have taken a random sample of 1000 rows out of total 541909 rows of full data

Practice : Sampling in Python

Import “Census Income Data/Income_data.csv”

Create a new dataset by taking a random sample of 5000 records

In [3]:

```
Income_Data=pd.read_csv("datasets\\Census Income Data\\Income_data.csv", encoding = "ISO-8859-1")
```

```
Income_Data.shape
```

Out[3]:

(32561, 15)

In [4]:

```
sample_Income_Data=Income_Data.sample(n=5000,replace="False")
sample_Income_Data.shape
```

Out[4]:

(5000, 15)

Relational Databases In Python

- Relational database management systems (RDBMS) are ubiquitous.
- They provide an easy way to store vast quantities of information. Furthermore, with the aid of SQL, accessing, maintaining, and modifying the data stored in them is all too easy.
- such an arrangement comes with certain caveats. If you wish to draw insights from the information stored in databases, you are limited to the commands provided by SQL.
- While SQL allows you to conduct joins or utilize aggregation functions, it does not provide the means to perform more advanced techniques such as conducting statistical tests or building predictive models. If you wish to carry out such operations, you'll need the aid of Python.

How do you implement Python code on databases that respond to SQL queries?

Database Adapter

To access databases in Python, you'll need to use a *database adapter*.

Python offers database adapters through its modules that allow access to major databases such as MySQL, PostgreSQL, SQL Server, and SQLite.

Furthermore, all of these modules rely on Python's database API (DB-API) for managing databases. As a result, the code used for managing databases is consistent across all database adapters.

- Once you are able to successfully communicate with one relational database, you should be able to communicate with them all.
- Leveraging database adapters requires an understanding of the key objects and methods that will be used to facilitate interaction with the database in question.

SELECT * FROM Person

id integer	first_name character varying (50)	last_name character varying (50)	gender character varying (50)	country character varying (50)	job character varying (50)	income integer
1	Bank	Merkel	Male	China	Nuclear Power Engineer	74116
2	Reinald	Coskerry	Male	Russia	Senior Financial Analyst	146391
3	Lemmie	Lambillion	Female	Brazil	VP Product Management	124466
4	Glennis	Mullin	Female	China	Chief Design Engineer	233965
5	Jeramie	Baggallay	Female	Russia	Senior Editor	81950

Connecting to the database

- First, we need to establish a connection to a database so that we can communicate with it using Python. We can achieve this by creating a *connection* object.
- When creating a connection object, you provide it with the information needed to locate and access the database. This includes the host, username, password, and database name.

```
import psycopg2


# connect to the database with the credientials
try:
    connection = psycopg2.connect(
        host=host,  # host on which the database is running
        database=database,  # name of the database to connect to
        user=username,  # username to connect with
        password=password)  # password associated with your username
except:
    print('Can not access database')
```

Writing queries to the database from Python

- Now that we have connected to the database, we should be able to write queries to the database directly from Python.
- Although we are on Python now, we'll need to continue using SQL queries to retrieve any information.
- Before writing any query, we need to create a *cursor* object.
- The cursor object uses the *execute* method to carry out the given query. The resulting output of the query can be retrieved with the *fetchall* method.

Let’s use this procedure to select the first 5 rows in the Person table.

```
# create a cursor object
cursor = connection.cursor()

# execute query with cursor
query = """SELECT *
        FROM Person
        LIMIT 5"""
cursor.execute(query)

# retrieve results of query
customers = cursor.fetchall()

# show retrieved data
for customer in customers:
    print(customer)
```

```
(1, 'Bank', 'Merkel', 'Male', 'China', 'Nuclear Power Engineer', 74116)
(2, 'Reinald', 'Coskerry', 'Male', 'Russia', 'Senior Financial Analyst', 146391)
(3, 'Lemmie', 'Lambillion', 'Female', 'Brazil', 'VP Product Management', 124466)
(4, 'Glennis', 'Mullin', 'Female', 'China', 'Chief Design Engineer', 233965)
(5, 'Jeramie', 'Baggallay', 'Female', 'Russia', 'Senior Editor', 81950)
```

To obtain the column names of the table, you can use the *description* attribute on the cursor object.

```
columns = [col[0] for col in
cursor.description]

print(f'Columns: {columns}')
```

```
Columns: ['id', 'first_name', 'last_name', 'gender', 'country', 'job', 'income']
```

3. Performing advanced analysis with the database

- Although we can write SQL queries to databases from Python, there's little point in using an adapter if that's all we can do.
- Let's perform an operation with the data in the "Person" table that we wouldn't be able to do with SQL queries alone.

To do this, we should first store the table in Python as a pandas data frame.

```
import pandas as pd

# obtain all data in Person table
cursor.execute('SELECT * FROM Person')
values = cursor.fetchall()
columns = [col[0] for col in cursor.description]

# store data in data frame
df = pd.DataFrame(values, columns=columns).sort_values('id')
df.head()
```

id	first_name	last_name	gender	country	job	income
1	Bank	Merkel	Male	China	Nuclear Power Engineer	74116
2	Reinald	Coskerry	Male	Russia	Senior Financial Analyst	146391
3	Lemmie	Lambillion	Female	Brazil	VP Product Management	124466
4	Glennis	Mullin	Female	China	Chief Design Engineer	233965
5	Jeramie	Baggallay	Female	Russia	Senior Editor	81950

What is SciPy?

- SciPy is a scientific computation library that uses NumPy underneath.
- 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 was created by NumPy's creator Travis Olliphant.

Why Use SciPy?

- If SciPy uses NumPy underneath, why can we not just use NumPy?
- SciPy has optimized and added functions that are frequently used in NumPy and Data Science.

Which Language is SciPy Written in?

- SciPy is predominantly written in Python, but a few segments are written in C.

Import SciPy

Once SciPy is installed, import the SciPy module(s) you want to use in your applications by adding the from scipy import module statement:

from scipy import constants

- Now we have imported the *constants* module from SciPy, and the application is ready to use it:
- Example
- How many cubic meters are in one liter:
-
```
from scipy import constants

print(constants.liter)
```

- Checking SciPy Version
- The version string is stored under the __version__ attribute.
- Example
-
```
import scipy

print(scipy.__version__)
```

SciPy Constants

Constants in SciPy

- As SciPy is more focused on scientific implementations, it provides many built-in scientific constants.
- These constants can be helpful when you are working with Data Science.
- PI is an example of a scientific constant.

Example

Print the constant value of PI:

```
from scipy import constants

print(constants.pi)
```

Constant Units

A list of all units under the constants module can be seen using the `dir()` function.

List all constants:

```
from scipy import constants

print(dir(constants))
```

Unit Categories

- The units are placed under these categories:
- Metric
- Binary
- Mass
- Angle
- Time
- Length
- Pressure
- Volume
- Speed
- Temperature
- Energy
- Power
- Force

SciPy Sparse Data

What is Sparse Data

Sparse data is data that has mostly unused elements (elements that don't carry any information).

It can be an array like this one:

[1, 0, 2, 0, 0, 3, 0, 0, 0, 0, 0, 0]

- **Sparse Data:** is a data set where most of the item values are zero.
- **Dense Array:** is the opposite of a sparse array: most of the values are *not* zero.

How to Work With Sparse Data

- SciPy has a module, scipy.sparse that provides functions to deal with sparse data.
- There are primarily two types of sparse matrices that we use:
- CSC - Compressed Sparse Column. For efficient arithmetic, fast column slicing.
- CSR - Compressed Sparse Row. For fast row slicing, faster matrix vector products
- We will use the CSR matrix in this tutorial.

CSR Matrix

We can create CSR matrix by passing an arrray into function scipy.sparse.csr_matrix().

Create a CSR matrix from an array:

```
import numpy as np
from scipy.sparse import csr_matrix

arr =
np.array([0, 0, 0, 0, 0, 1, 1, 0, 2])

print(csr_matrix(arr))
```

```
  (0, 5)    1
  (0, 6)    1
  (0, 8)    2
```

Sparse Matrix Methods

```
import numpy as np
from scipy.sparse import csr_
matrix

arr = np.array([[0, 0, 0],
[0, 0, 1], [1, 0, 2]])

print(csr_matrix(arr).data)
```

```
[1 1 2]
```

Counting nonzeros with the count_nonzero() method:

```
import numpy as np
from scipy.sparse import csr_matrix

arr = np.array([[0, 0, 0],
[0, 0, 1], [1, 0, 2]])

print(csr_matrix(arr).count_nonzero
())
```

```
3
```

Removing zero-entries from the matrix with the eliminate_zeros() method:

```
import numpy as np

from scipy.sparse import csr_matrix


arr = np.array([[0, 0, 0], [0, 0, 1], [1, 0, 2]])


mat = csr_matrix(arr)

mat.eliminate_zeros()


print(mat)
 (1, 2)     1
  (2, 0)    1
  (2, 2)    2
```

Eliminating duplicate entries with the sum_duplicates() method:

```
import numpy as np
from scipy.sparse import csr_matrix

arr = np.array([[0, 0, 0],
[0, 0, 1], [1, 0, 2]])

mat = csr_matrix(arr)
mat.sum_duplicates()

print(mat)
```

```
  (1, 2)    1
  (2, 0)    1
  (2, 2)    2
```

SciPy Graphs

Working with Graphs. Graphs are an essential data structure.

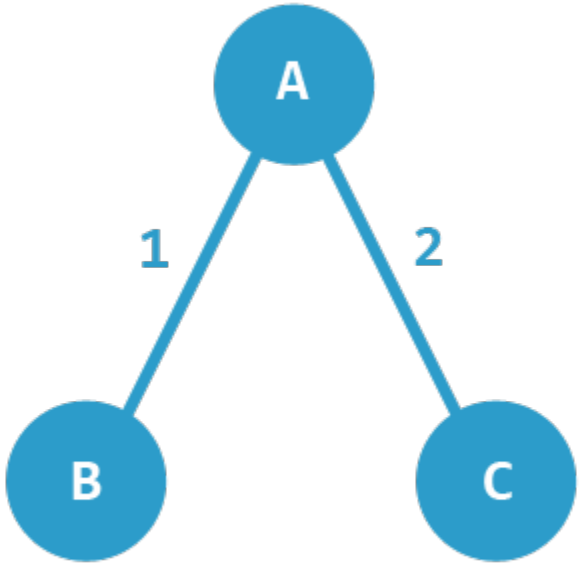
SciPy provides us with the module scipy.sparse.csgraph for working with such data structures.

Adjacency Matrix

Adjacency matrix is a nxn matrix where n is the number of elements in a graph.

And the values represents the connection between the elements.

For a graph like this, with elements A, B and C, the connections are:
A & B are connected with weight 1.
A & C are connected with weight 2.
C & B is not connected.



A B C

A:[0 1 2]

B:[1 0 0]

C:[2 0 0]

Connected Components

Find all of the connected components with the `connected_components()` method.

```
import numpy as np
from scipy.sparse.csgraph import connec
ted_components
from scipy.sparse import csr_matrix

arr = np.array([
  [0, 1, 2],
  [1, 0, 0],
  [2, 0, 0]
])

newarr = csr_matrix(arr)

print(connected_components(newarr))
```

(1, array([0, 0, 0], dtype=int32))