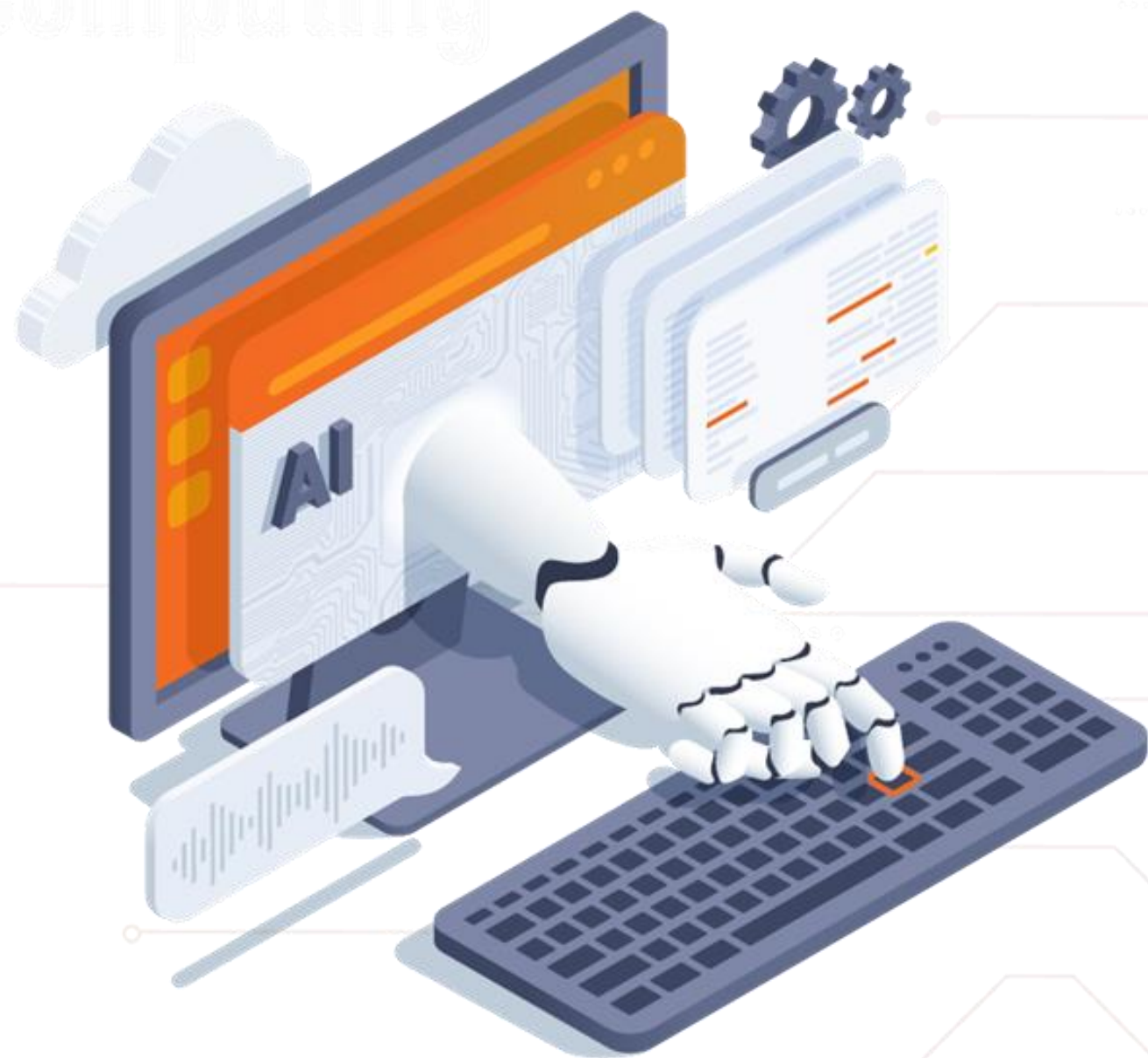


DATA AND  
ARTIFICIAL INTELLIGENCE  
Computing



**Caltech**

**Center for Technology &  
Management Education**

## Python Libraries for Data Science

# Learning Objectives

By the end of this lesson, you will be able to:

- 👁 Explain the use of Python library
- 👁 List various Python libraries
- 👁 Identify the SciPy sub-packages
- 👁 Describe web scraping with BeautifulSoup

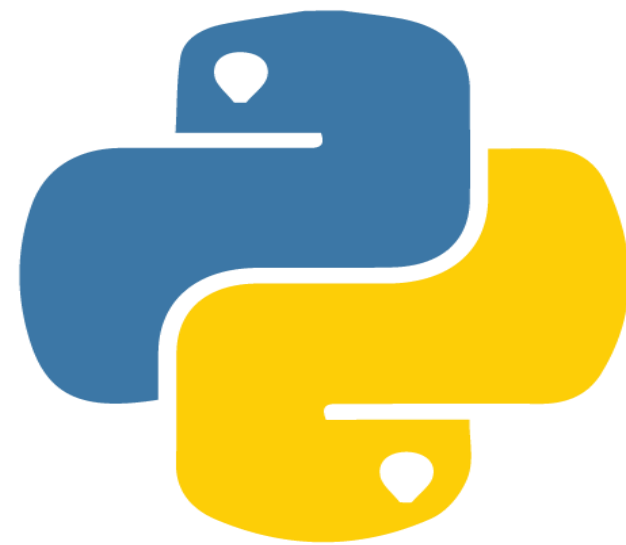


## Python Libraries for Data Science

# What Is Python Library?

---

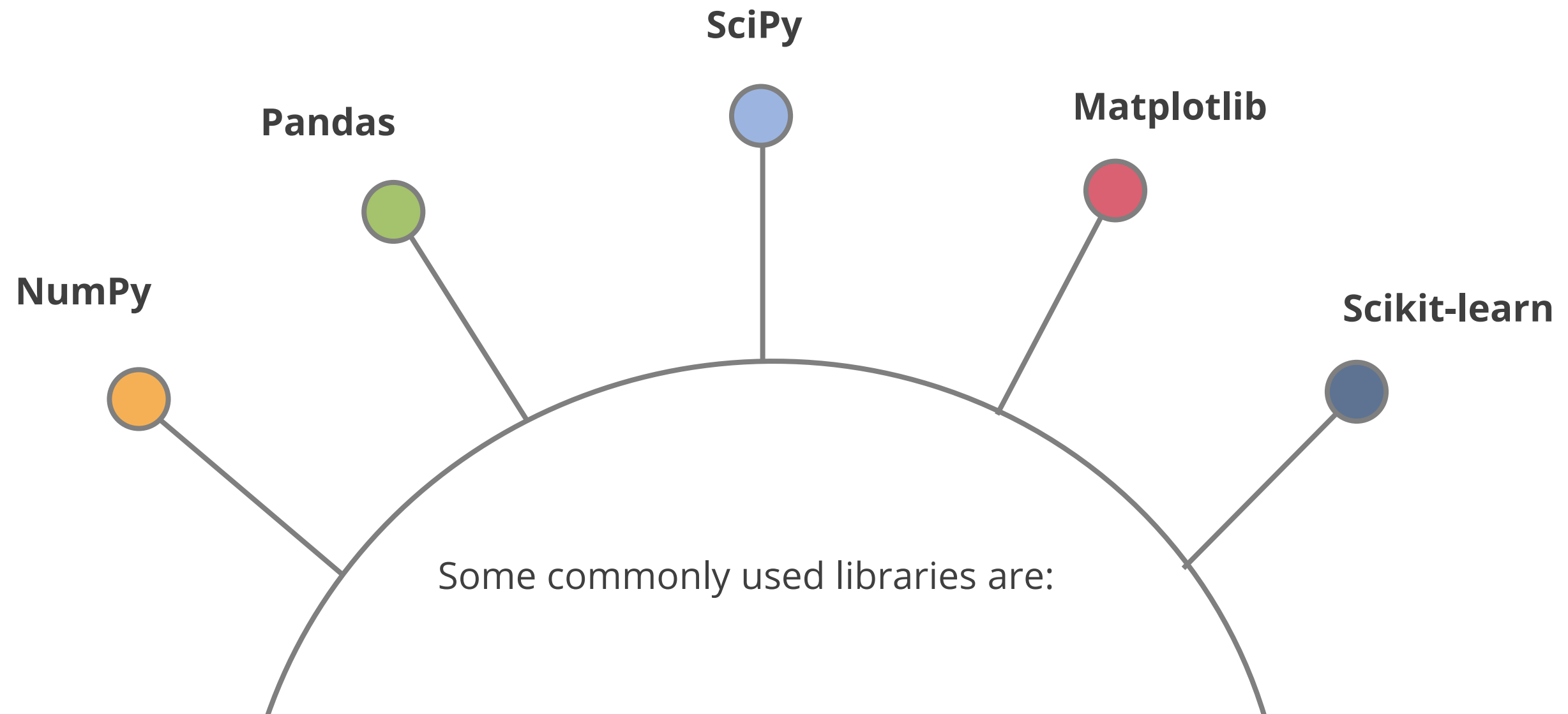
A Python library is a group of interconnected modules. It contains code bundles that can be reused in different programs and apps.



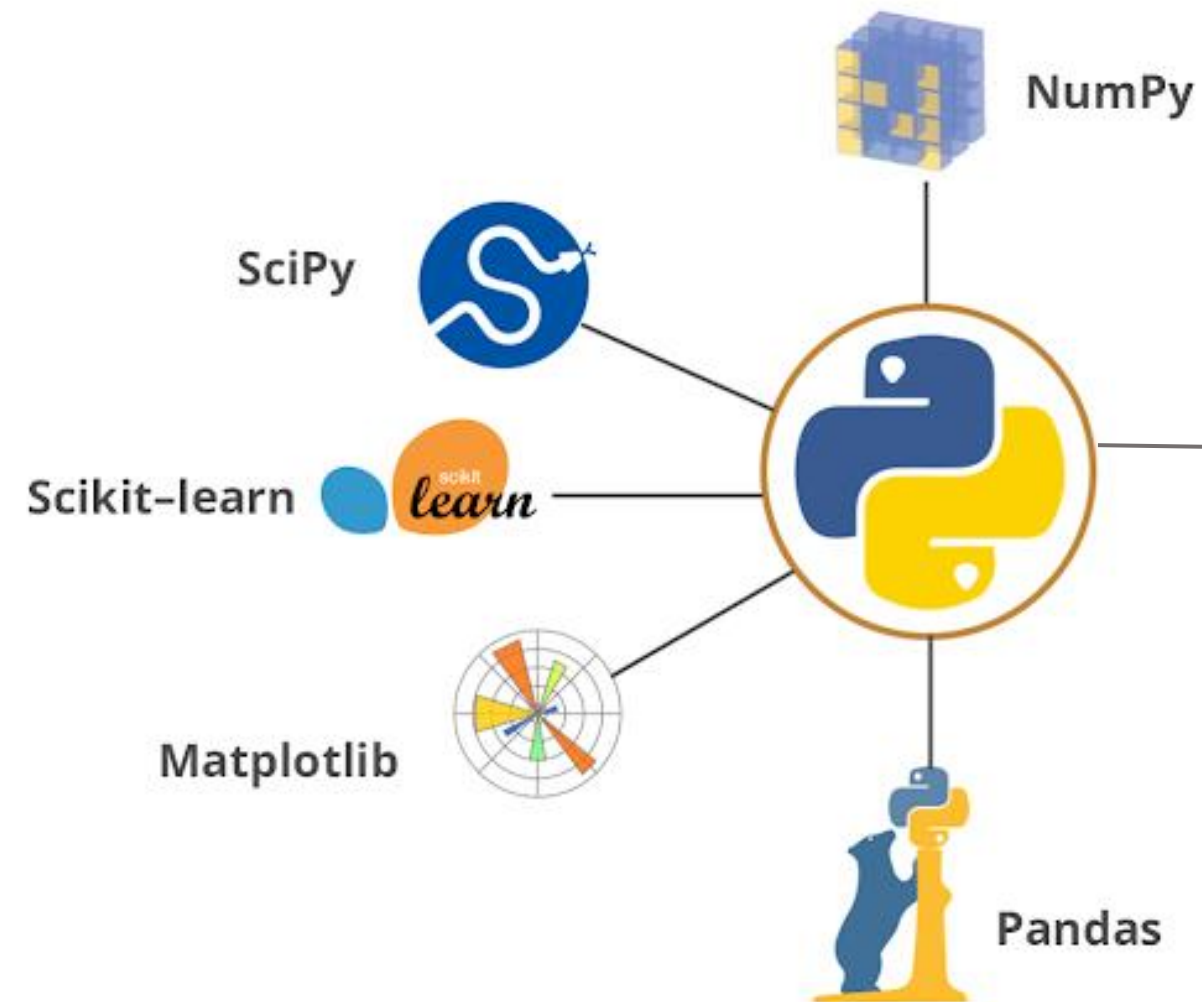
Python programming is made easier and more convenient for programmers due to its reusability.

# Python Libraries

Various other Python libraries make programming easier.



# Benefits of Python Libraries



Easy to learn

Open source

Efficient and multi-platform support

Huge collection of libraries, functions, and modules

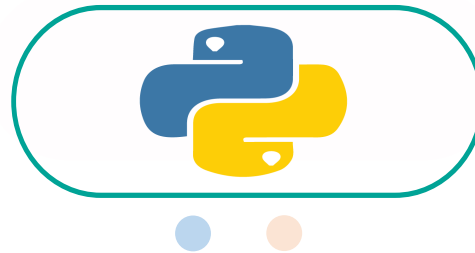
Big open-source community

Integrates well with enterprise apps and systems

Great vendor and product support

# Python Libraries: NumPy and Pandas

---



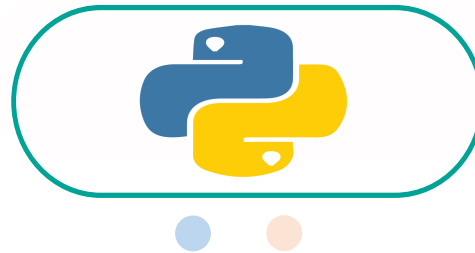
01

**NumPy:** Numerical Python is a machine learning library that can handle big matrices and multi-dimensional data.

02

**Pandas:** Pandas consist of a variety of analysis tools and configurable high-level data structures.

# Python Libraries: SciPy and Matplotlib



03

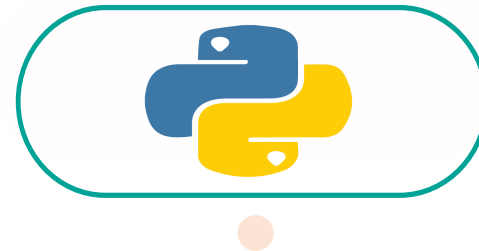
**SciPy:** Scientific Python is an open-source high-level scientific computation package. This library is based on a NumPy extension.

04

**Matplotlib:** It is also an open-source library that plots high-definition figures such as pie charts, histograms etc.



# Python Libraries: Scikit-Learn



05

**Scikit-learn:** The library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering, and dimensionality reduction.

## Import Library into Python Program

# Import Module in Python

In Python, a file is referred to as a module. The **import** keyword is used to utilize it.

Whenever we need to use a module, we import it from its library.

Example □

Importing math library

```
import math
```

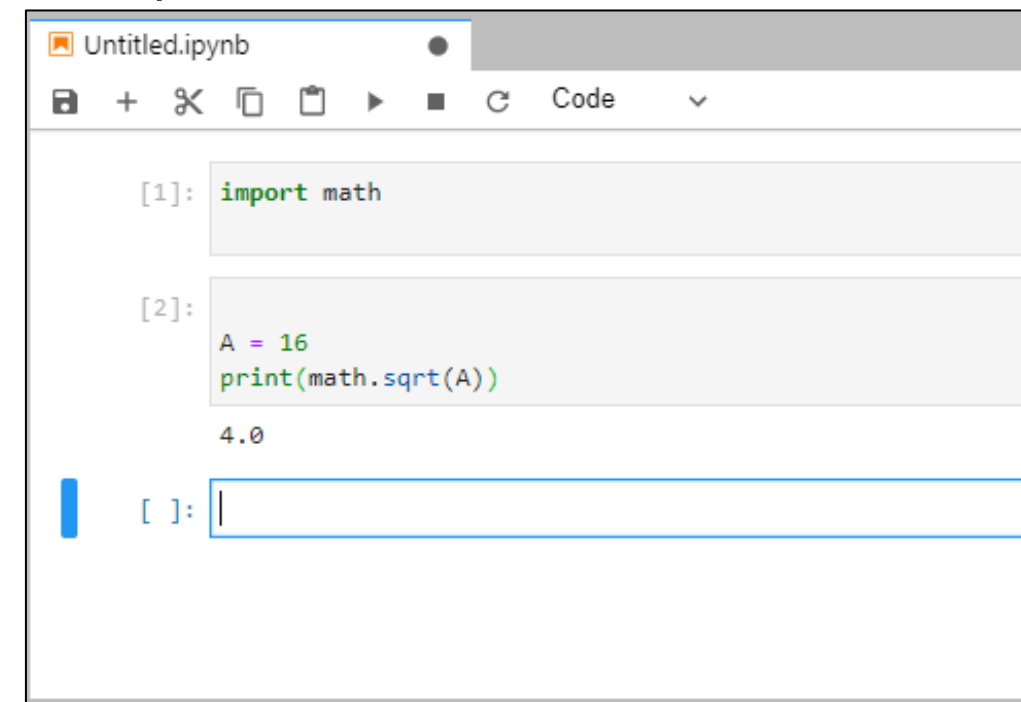
# Example: Import Module in Python

In this code, the math library is imported. One of its methods, that is sqrt(square root), is used without writing the actual code to calculate the square root of a number.

Example:

```
import math  
  
A = 16  
print(math.sqrt(A))
```

Output:



The screenshot shows a Jupyter Notebook interface with a tab titled 'Untitled.ipynb'. The toolbar includes icons for saving, adding, deleting, copying, pasting, running, and a dropdown menu currently set to 'Code'. The code area contains two cells. The first cell, labeled '[1]:', contains the code 'import math'. The second cell, labeled '[2]:', contains the code 'A = 16' followed by 'print(math.sqrt(A))'. Below the code in the second cell, the output '4.0' is displayed. A third cell, labeled '[ ]:', is currently empty with a cursor at the end of the line.

```
[1]: import math  
  
[2]: A = 16  
     print(math.sqrt(A))  
     4.0  
  
[ ]: |
```

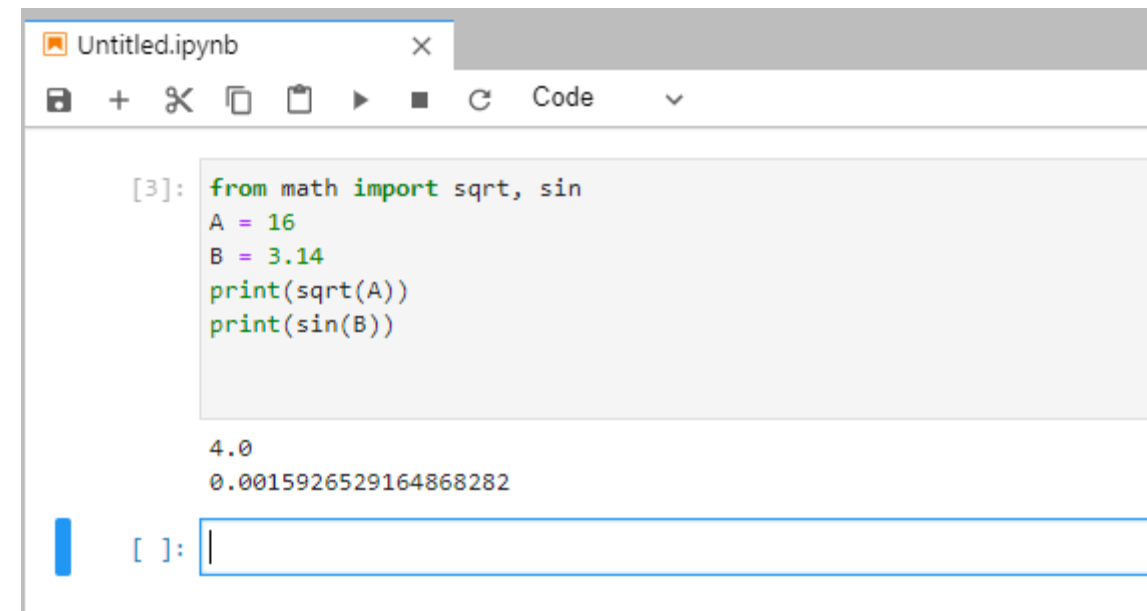
# Example: Import Module in Python

As in the previous code, a complete library is imported to use one of its methods. However, only importing “sqrt” from the math library would have worked.

Example:

```
from math import  
sqrt, sin  
A = 16  
B = 3.14  
print(sqrt(A))  
print(sin(B))
```

Output:



The screenshot shows a Jupyter Notebook window titled "Untitled.ipynb". The code cell contains the following Python code:

```
[3]: from math import sqrt, sin  
A = 16  
B = 3.14  
print(sqrt(A))  
print(sin(B))
```

The output of the code is displayed below the code cell:

```
4.0  
0.0015926529164868282
```

Below the output, there is an input prompt `[ ]:` followed by a text box for entering new code.

In the above code, only “sqrt” and “sin” methods from the math library are imported.

# NumPy

# NumPy

NumPy (Numerical Python) is a library that consists of multidimensional array objects and a collection of functions for manipulating them.

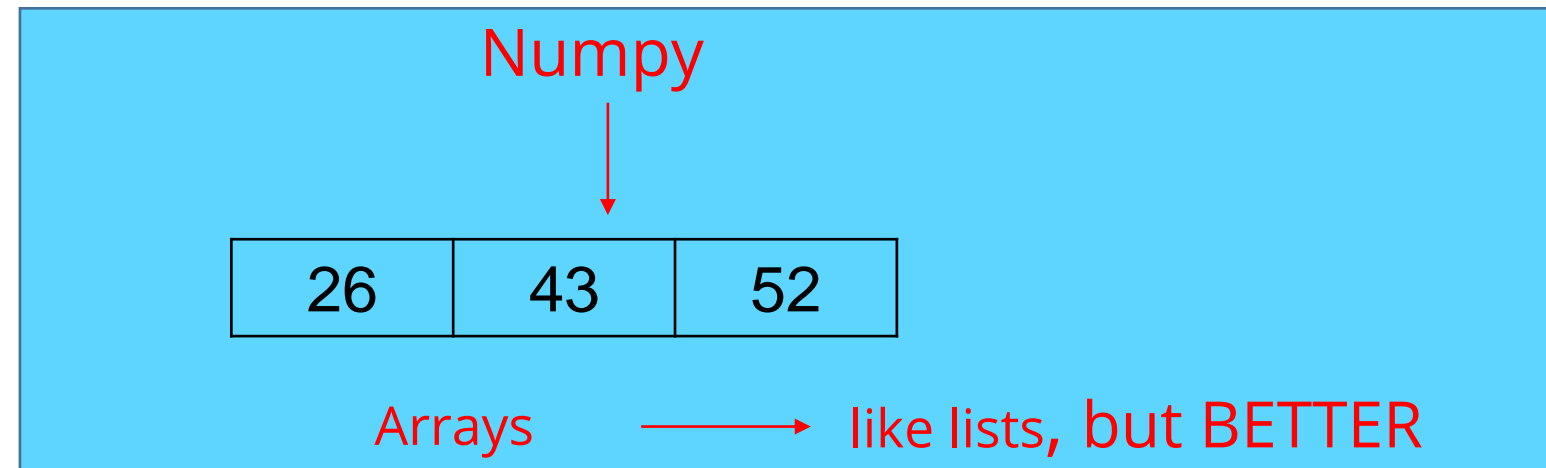


***NumPy***

NumPy conducts mathematical and logical operations on arrays.

# Why NumPy

Numerical Python (NumPy) supports multidimensional arrays over which mathematical operations can be easily applied.



```
distance=[10,15,17,26]  
time=[.30,.47,.55,1.20]
```

```
import numpy as np
```

← Import NumPy

```
np_distance = np.array(distance)  
np_time=np.array(time)  
speed=np_distance/np_time
```

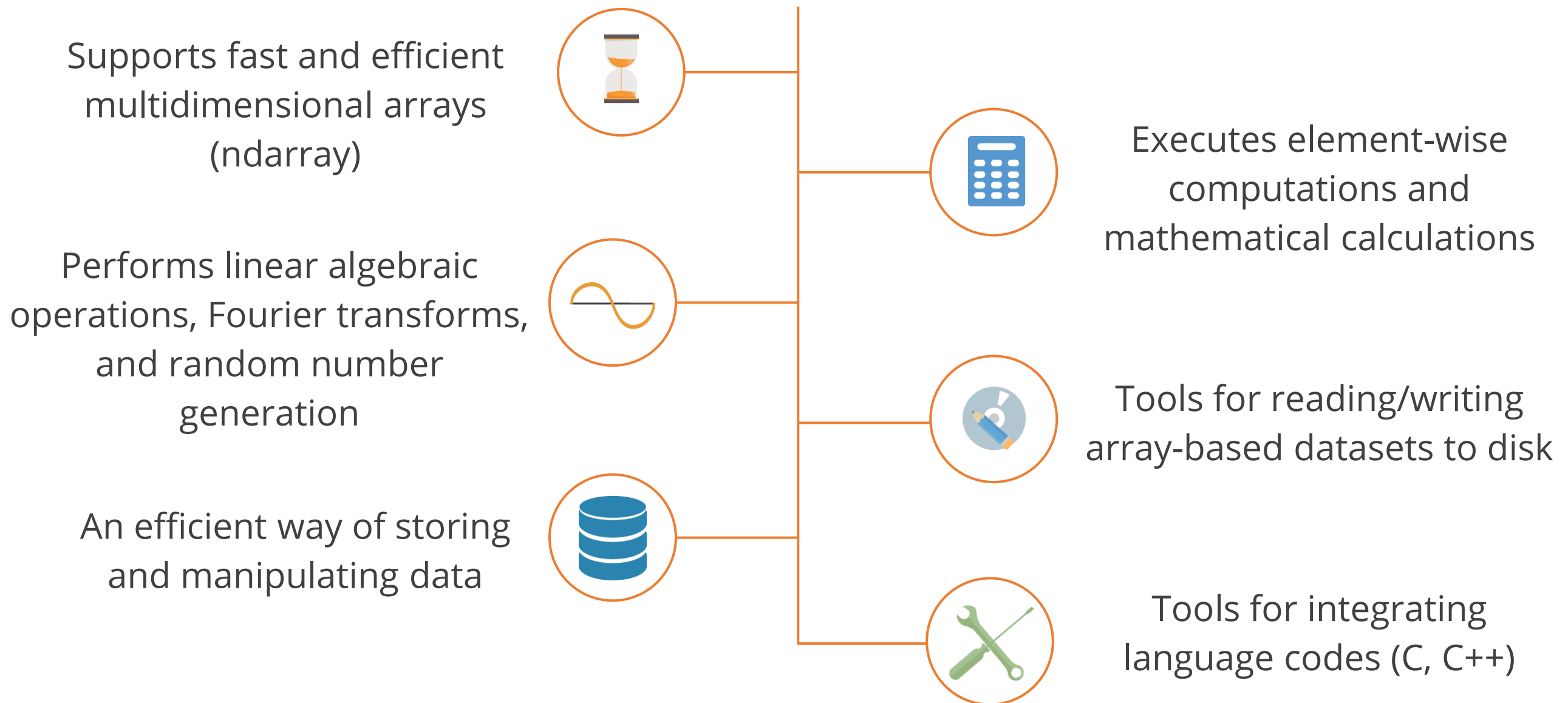
} Create “distance” and “time” NumPy arrays  
← Mathematical function applied over the entire “distance” and “time” arrays

```
speed  
array([ 33.33333333,  31.91489362,  30.90909091,  21.66666667])
```



# NumPy Overview

NumPy is the foundational package for mathematical computing in Python.  
It has the following properties:



# Functions of NumPy Module

| S.No | NumPy Module                       | Functions   |
|------|------------------------------------|---|
| 1    | NumPy array manipulation functions | numpy.reshape()<br>numpy.concatenate()<br>numpy.shape()                               |
| 2    | NumPy string functions             | numpy.char.add()<br>numpy.char.replace()<br>numpy.char.upper() and numpy.char.lower() |
| 3    | NumPy arithmetic functions         | numpy.add()<br>numpy.subtract()<br>numpy.mod() and numpy.power()                      |
| 4    | NumPy statistical functions        | numpy.median()<br>numpy.mean()<br>numpy.average()                                     |

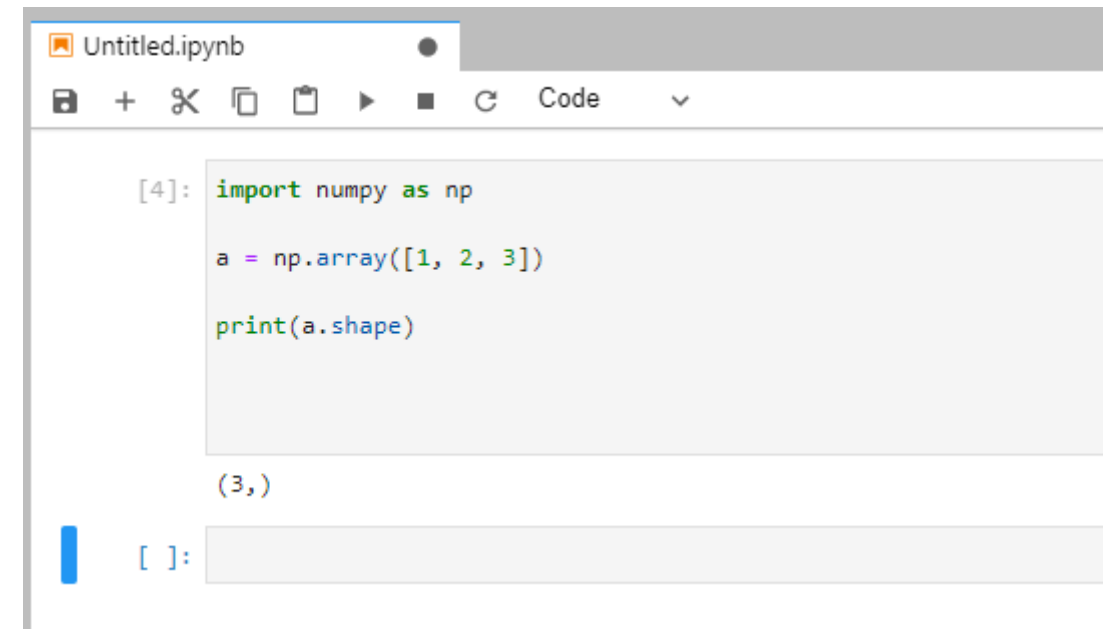
# NumPy Function: Example 1

To access NumPy and its functions, import it in the Python code as shown below:

Example:

```
import numpy as np  
  
a = np.array([1, 2, 3])  
  
print(a.shape)
```

Output:

A screenshot of a Jupyter Notebook interface. The top bar shows 'Untitled.ipynb' and a toolbar with icons for file operations and execution. The code cell contains three lines: 'import numpy as np', 'a = np.array([1, 2, 3])', and 'print(a.shape)'. The output cell below shows '(3,)' in green text. The next cell is empty, showing '[ ]:'.

```
Untitled.ipynb  
[4]: import numpy as np  
      a = np.array([1, 2, 3])  
      print(a.shape)  
  
(3,)  
[ ]:
```

In this example, the NumPy module is imported and the shape function is used.

# NumPy Function: Example 2

To access NumPy and its functions, import it in the Python code as shown below:

Example:

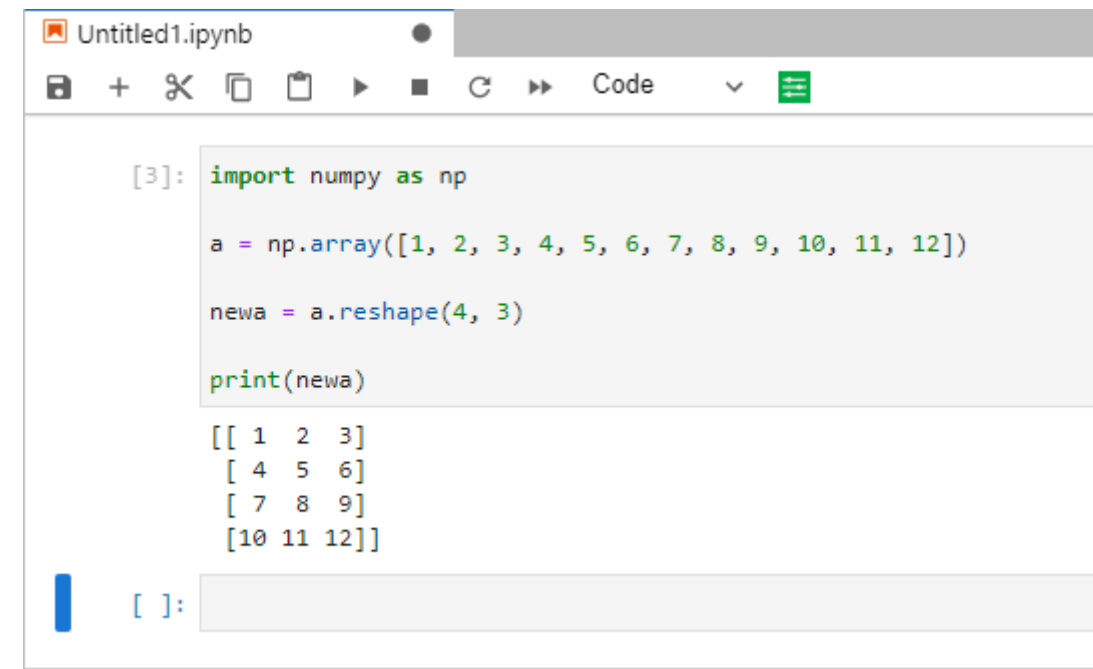
```
import numpy as np

a = np.array([1, 2, 3, 4, 5,
6, 7, 8, 9, 10, 11, 12])

newa = a.reshape(4, 3)

print(newa)
```

Output:

A screenshot of a Jupyter Notebook window titled 'Untitled1.ipynb'. The toolbar shows icons for file operations, running, and code execution. The code cell contains the following Python code:

```
[3]: import numpy as np
a = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])
newa = a.reshape(4, 3)
print(newa)
```

The output of the code is displayed below the cell:

```
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
```

In this example, the NumPy module is imported and the reshape function is used.

# Pandas

# Pandas

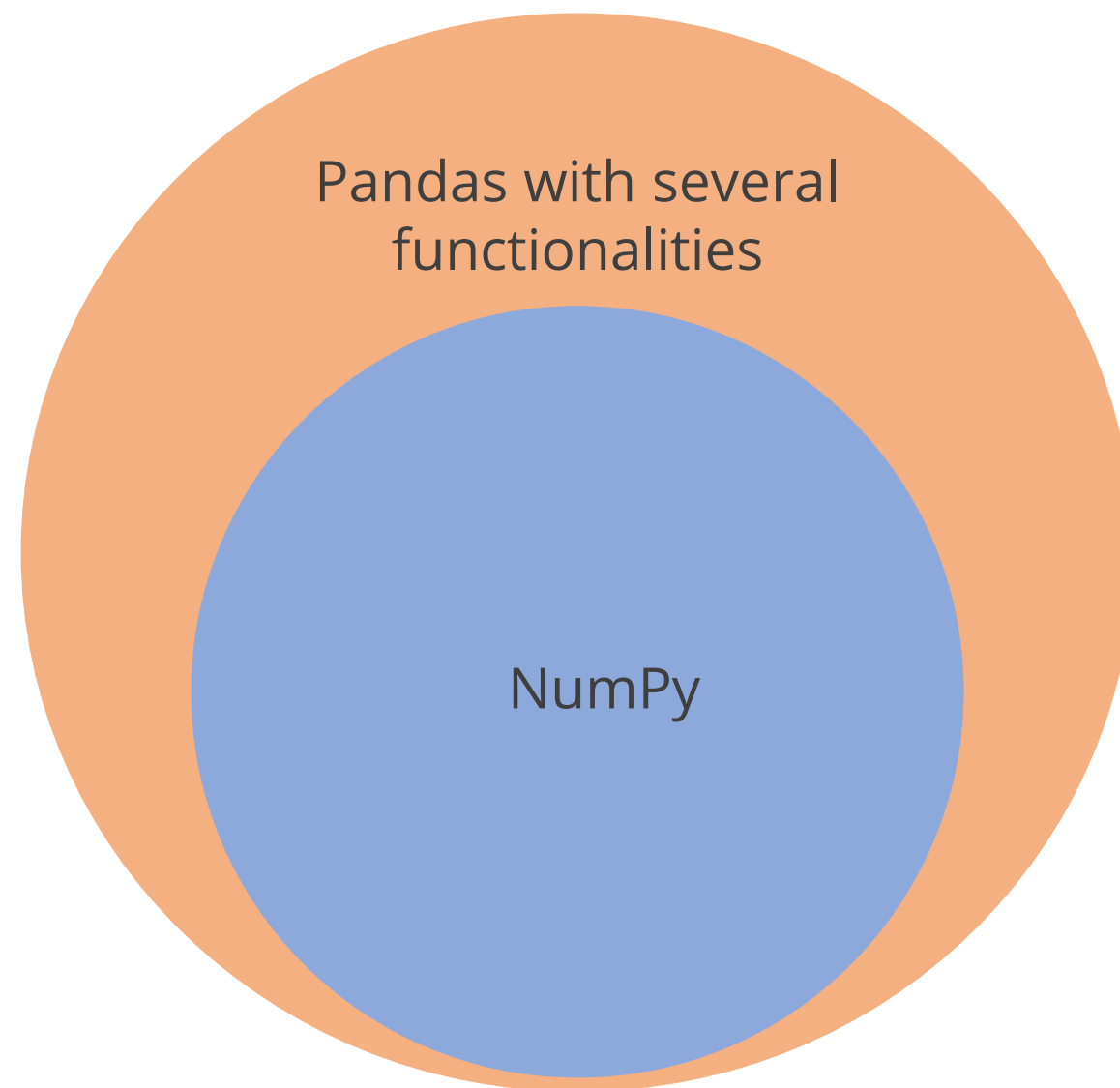
Pandas is a Python package that allows you to work with large datasets.



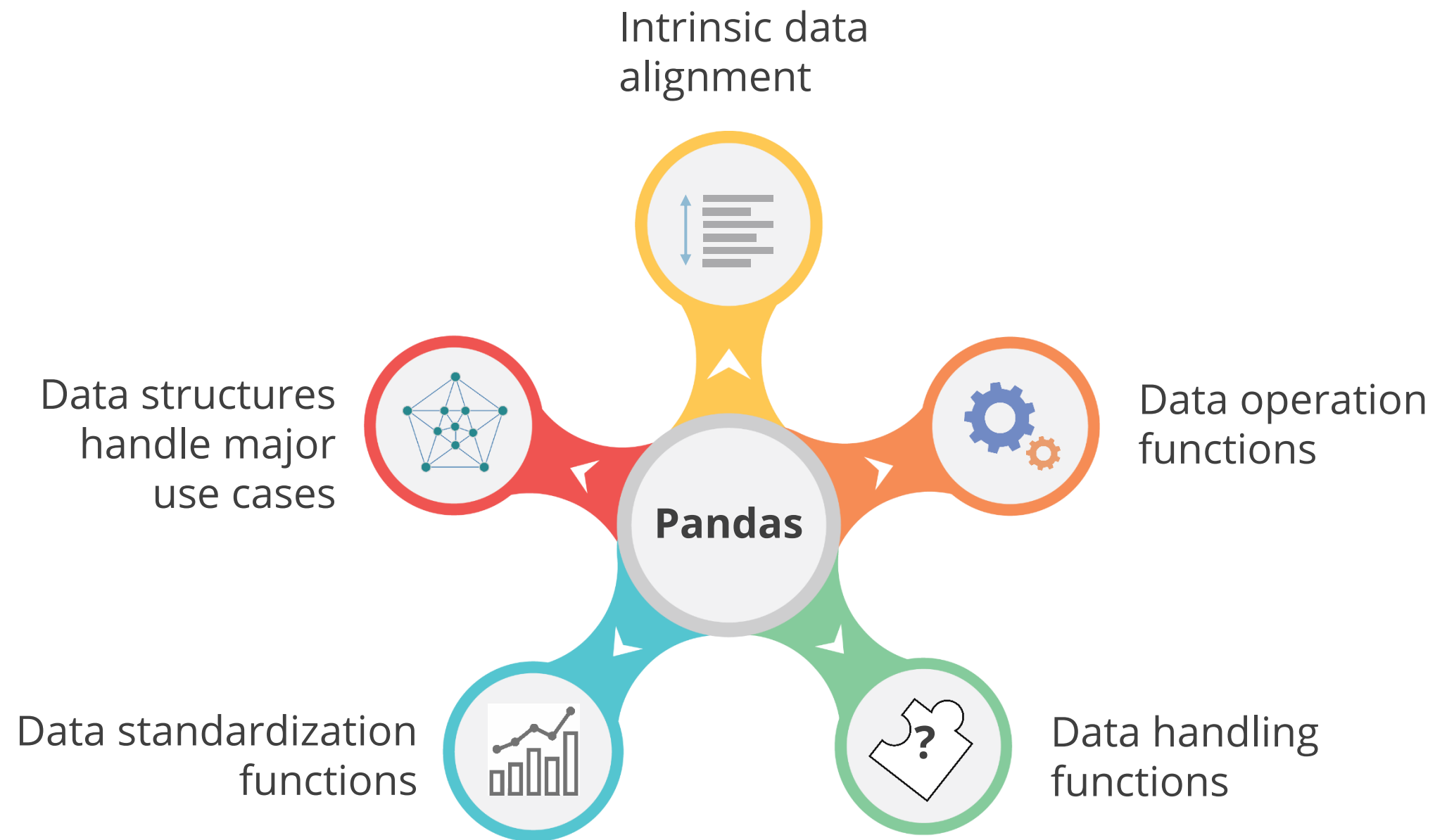
It offers tools for data analysis, cleansing, exploration, and manipulation.

# Why Pandas

NumPy is great for mathematical computing, but why do we need pandas?



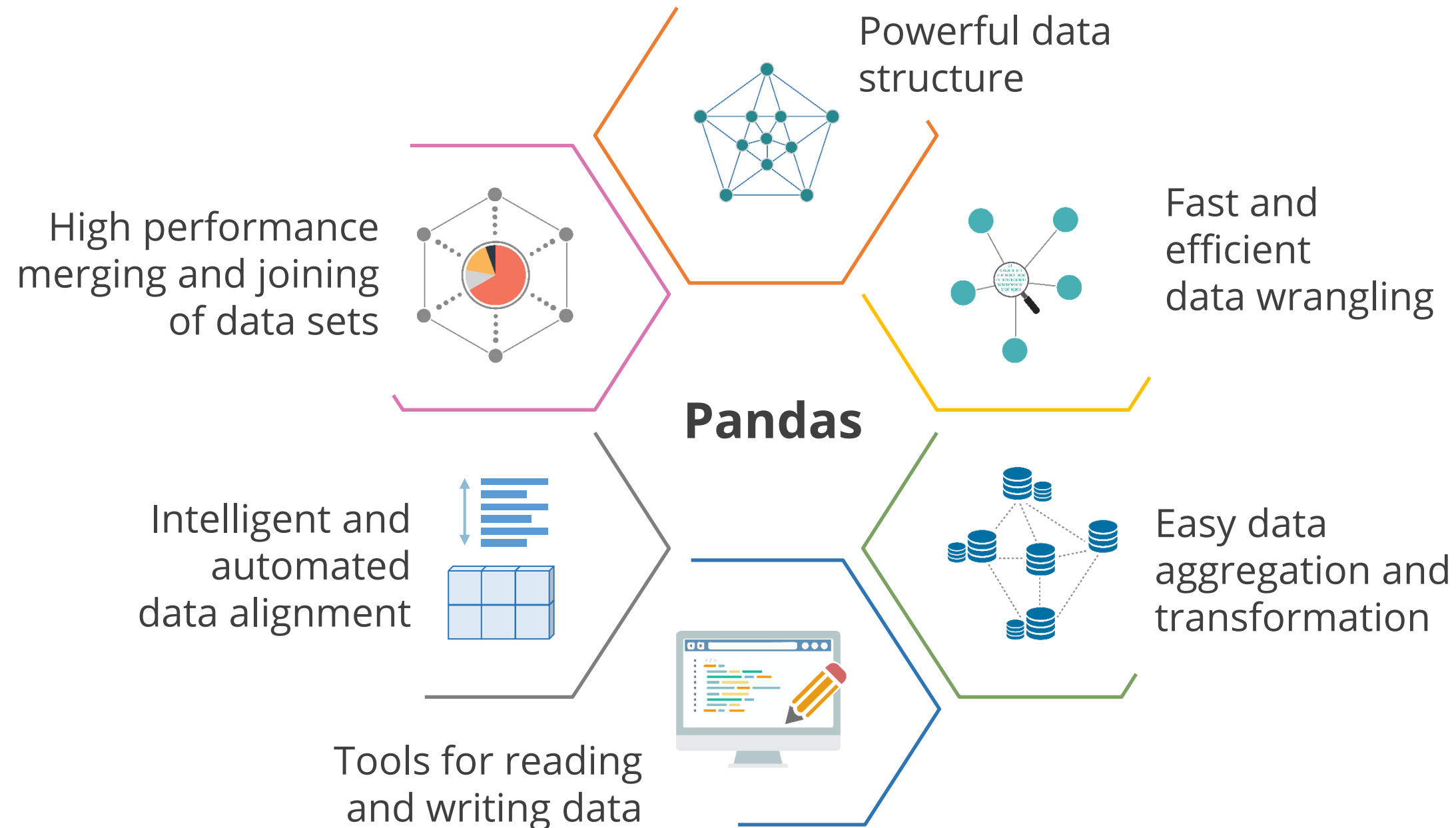
# Why Pandas





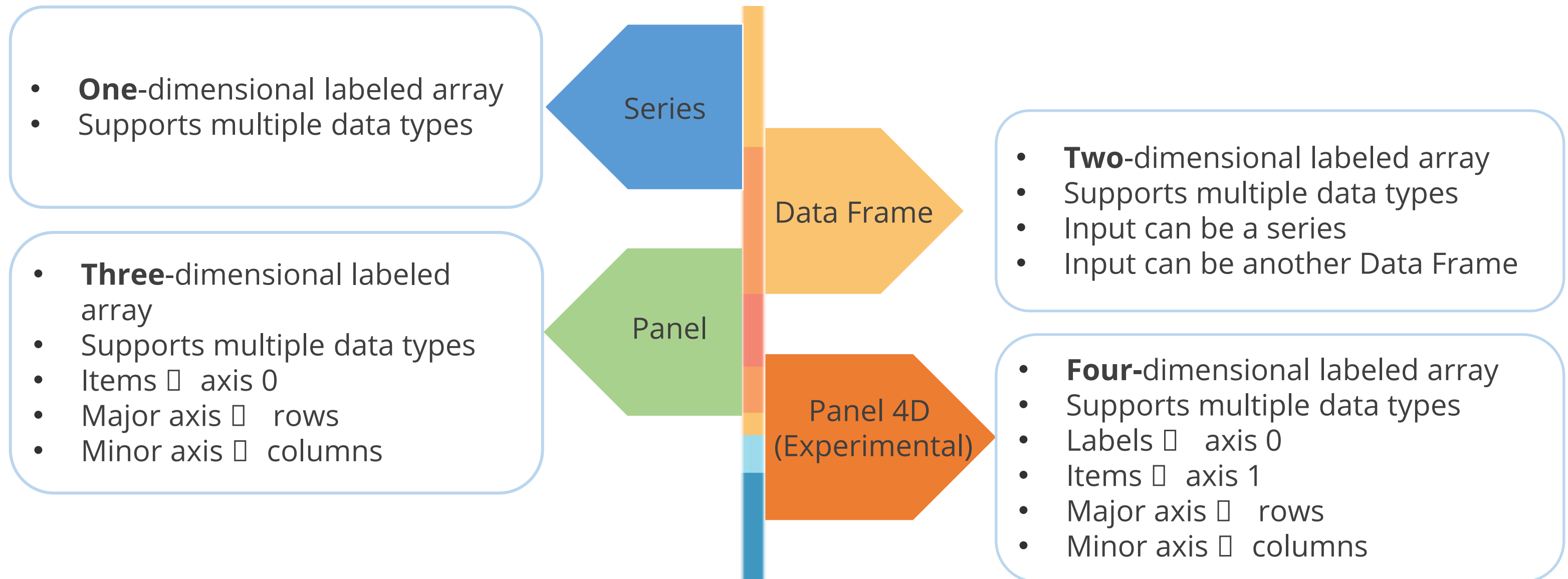
# Features of Pandas

The various features of pandas make it an efficient library for data scientists.



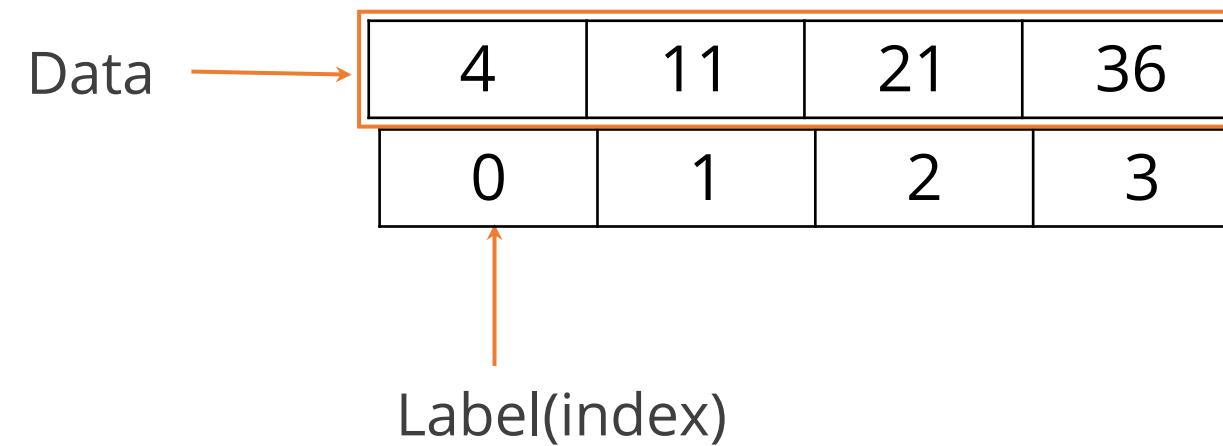
# Data Structures

The four main libraries of pandas data structure are:



# Understanding Series

Series is a one-dimensional array-like object containing data and labels (or index).



The diagram illustrates a Series structure with two rows of four cells each. The top row contains the values 4, 11, 21, and 36. The bottom row contains the indices 0, 1, 2, and 3. An orange arrow labeled 'Data' points to the top row, and another orange arrow labeled 'Label(index)' points to the bottom row.

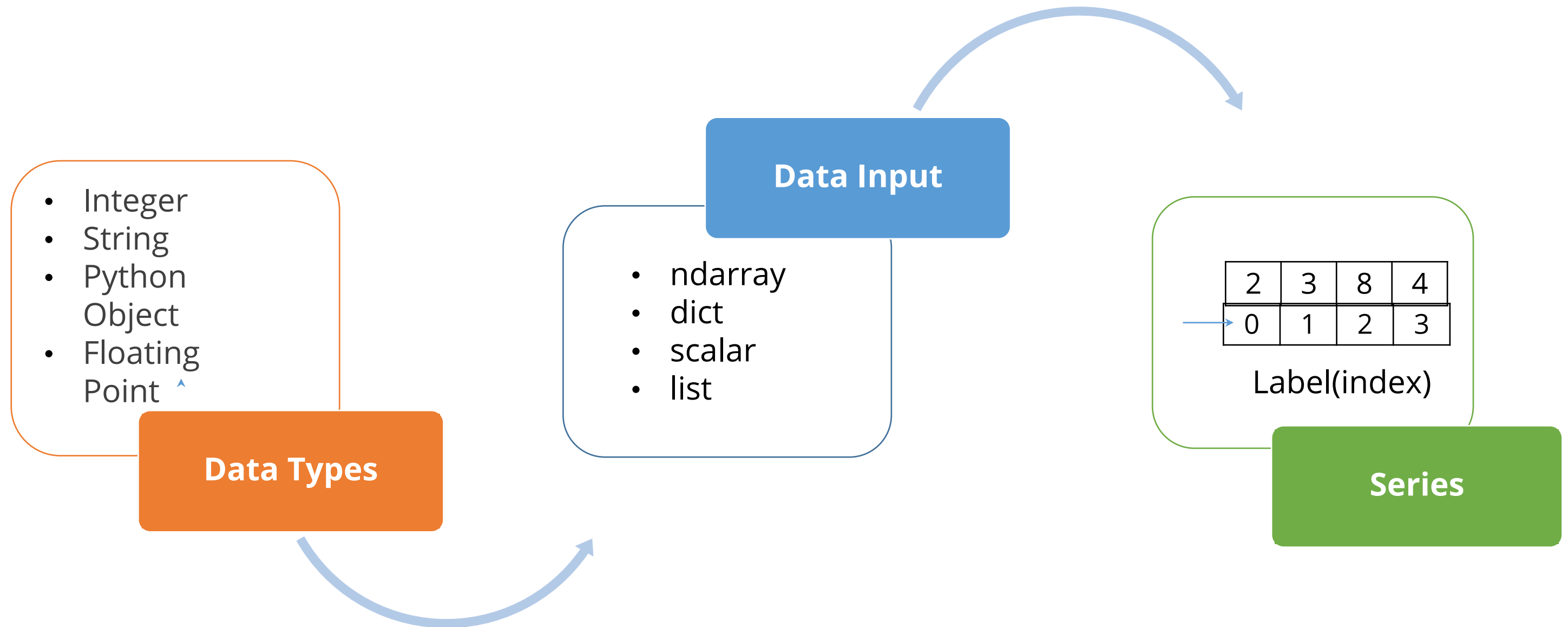
|   |    |    |    |
|---|----|----|----|
| 4 | 11 | 21 | 36 |
| 0 | 1  | 2  | 3  |



Data alignment is intrinsic and will not be broken until changed explicitly by program.

# Series

Series can be created with different data inputs:



# How to Create Series?

Key points to note while creating a series are:

- Import Pandas as it is the main library (Import pandas as pd)
- Import NumPy while working with ndarrays (Import numpy as np)
- Apply the syntax and pass the data elements as arguments

## Basic Method

```
S = pd.Series(data, index = [index])
```



|   |    |    |    |
|---|----|----|----|
| 4 | 11 | 21 | 36 |
|---|----|----|----|

Series

# Creating Series from a List

```
In [14]: import numpy as np  
import pandas as pd
```

← Import libraries

```
In [15]: first_series = pd.Series(list('abcdef'))
```

← Pass list as an argument

```
In [16]: print (first_series)
```

Index →

|   |   |
|---|---|
| 0 | a |
| 1 | b |
| 2 | c |
| 3 | d |
| 4 | e |
| 5 | f |

← Data value

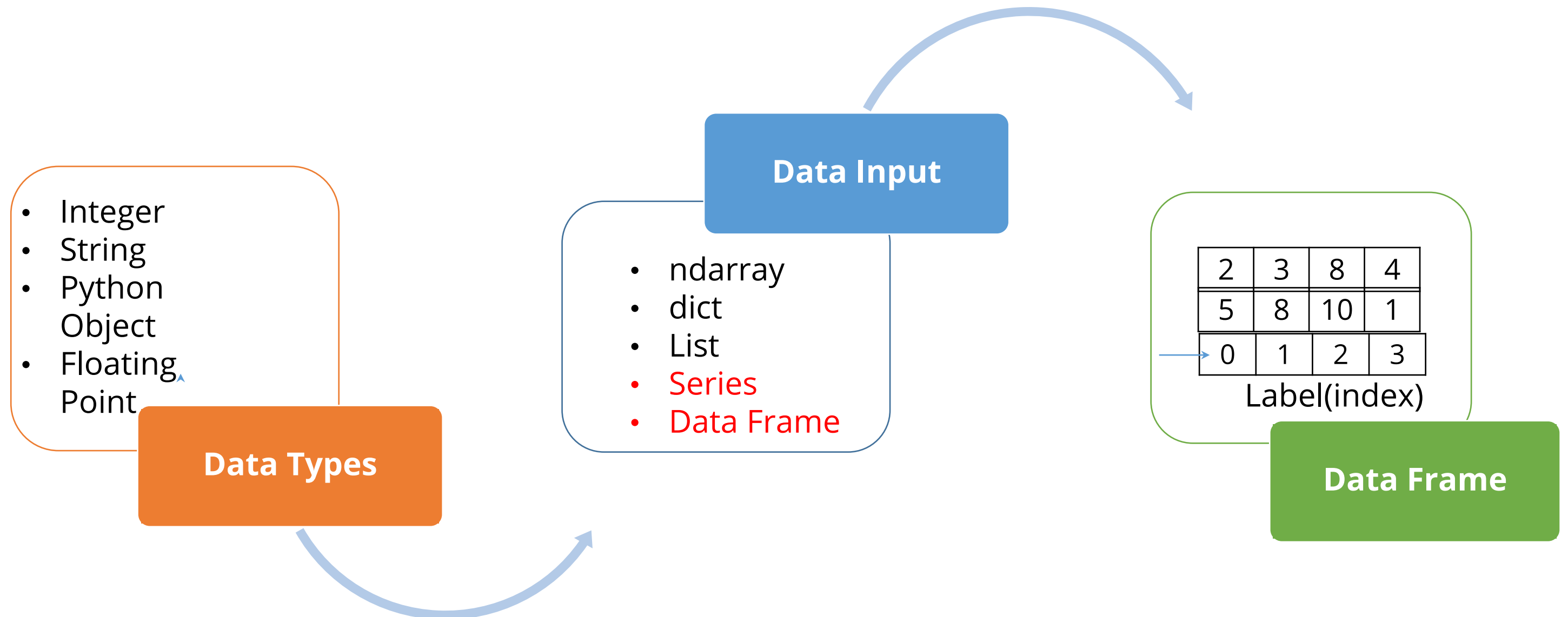
dtype: object ← Data type



The index is not created for data but notices that data alignment is done automatically.

# Data Frame

Data Frame is a two-dimensional labeled data structure with columns of potentially different types.



# Creating Data Frame from Lists

```
In [1]: import pandas as pd
```

Create DataFrame from dict of equal length lists

```
In [2]: #Last five olympics data: place, year and number of countries participated
olympic_data_list = {'HostCity': ['London', 'Beijing', 'Athens', 'Sydney', 'Atlanta'],
                    'Year': [2012, 2008, 2004, 2000, 1996],
                    'No. of Participating Countries': [205, 204, 201, 200, 197]}
}
```

```
In [3]: df_olympic_data = pd.DataFrame(olympic_data_list)
```

Pass the list to the DataFrame

```
In [4]: df_olympic_data
```

```
Out[4]:
```

|   | HostCity | No. of Participating Countries | Year |
|---|----------|--------------------------------|------|
| 0 | London   | 205                            | 2012 |
| 1 | Beijing  | 204                            | 2008 |
| 2 | Athens   | 201                            | 2004 |
| 3 | Sydney   | 200                            | 2000 |
| 4 | Atlanta  | 197                            | 1996 |



# Creating Data Frame from dict

This example shows how to create a Data Frame from a series of dicts.

Create DataFrame from dict of dicts

```
In [5]: olympic_data_dict = {'London':{2012:205}, 'Beijing':{2008:204}}
```

dict one dict two

```
In [6]: df_olympic_data_dict = pd.DataFrame(olympic_data_dict)
```

Entire dict

```
In [7]: df_olympic_data_dict
```

Out[7]:

|      | Beijing | London |
|------|---------|--------|
| 2008 | 204     | NaN    |
| 2012 | NaN     | 205    |

# Viewing Data Frame

A Data Frame can be viewed by referring to the column names or using the describe function.

```
In [8]: #select by City name  
df_olympic_data.HostCity
```

```
Out[8]: 0    London  
1    Beijing  
2    Athens  
3    Sydney  
4    Atlanta  
Name: HostCity, dtype: object
```

```
In [9]: #use describe function to display the content  
df_olympic_data.describe
```

```
Out[9]: <bound method DataFrame.describe of      HostCity  No. of Participating Countries  Year  
0    London      205      2012  
1   Beijing      204      2008  
2    Athens      201      2004  
3   Sydney      200      2000  
4   Atlanta      197      1996>
```

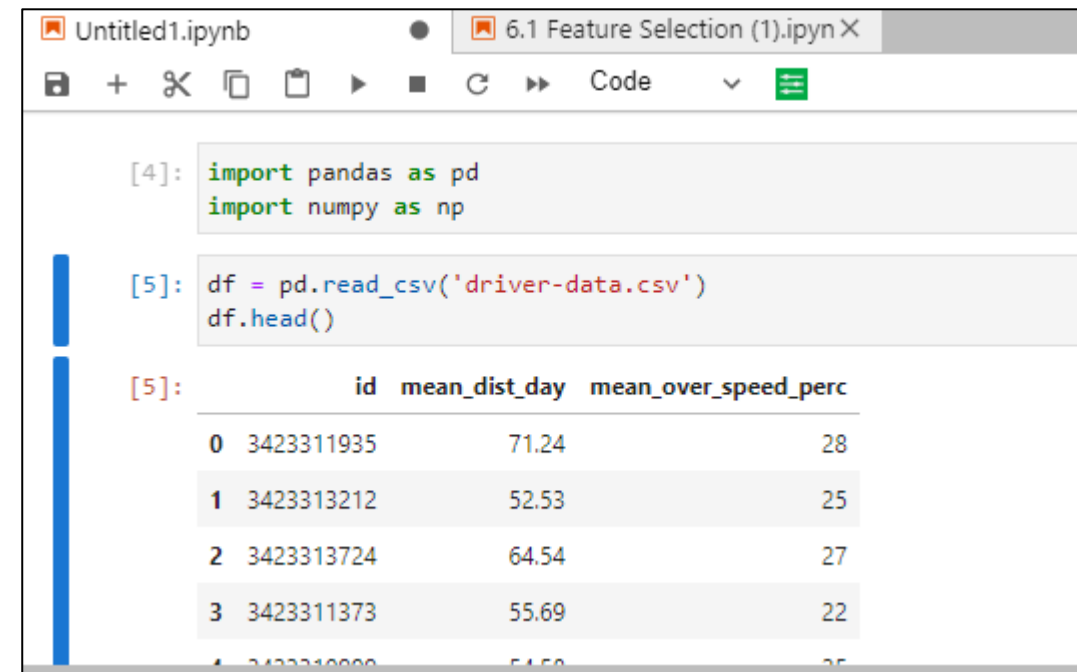
# Pandas Functions: Example 1

The example shown uses two functions: the `pd.read_csv()` function to import a dataset, and the `df.head()` function to output the first five rows of a dataset.

Output:

Example:

```
import pandas as pd
import numpy as np
df = pd.read_csv('driver-data.csv')
df.head()
```



The screenshot shows a Jupyter Notebook interface with two tabs: 'Untitled1.ipynb' and '6.1 Feature Selection (1).ipynb'. The code in the first cell is:

```
[4]: import pandas as pd
import numpy as np
```

The second cell contains:

```
[5]: df = pd.read_csv('driver-data.csv')
df.head()
```

The output of the second cell is a table with 5 rows and 3 columns:

|   | id         | mean_dist_day | mean_over_speed_perc |
|---|------------|---------------|----------------------|
| 0 | 3423311935 | 71.24         | 28                   |
| 1 | 3423313212 | 52.53         | 25                   |
| 2 | 3423313724 | 64.54         | 27                   |
| 3 | 3423311373 | 55.69         | 22                   |
| 4 | 3423310000 | 54.50         | 25                   |

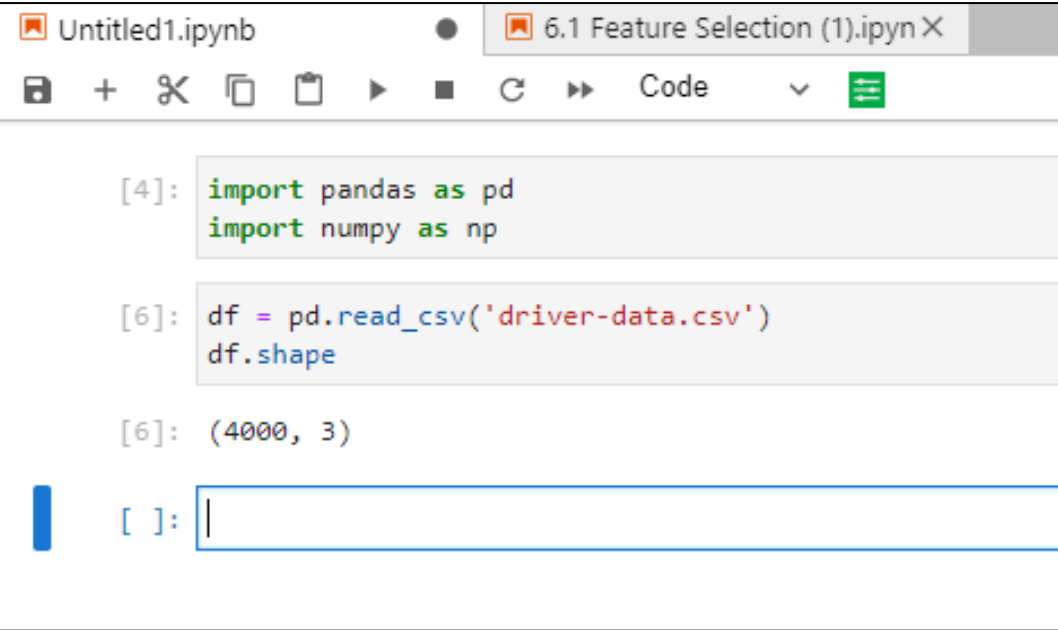
# Pandas Functions: Example 2

The example uses `df.shape()` function to output the shape of the dataset.

Example:

```
import pandas as pd
import numpy as np
df = pd.read_csv('driver-data.csv')
df.shape
```

Output:



A screenshot of a Jupyter Notebook interface. The top bar shows two tabs: 'Untitled1.ipynb' and '6.1 Feature Selection (1).ipynb'. Below the tabs is a toolbar with icons for file operations and execution. The notebook contains three code cells. The first cell, labeled '[4]:', contains the imports: `import pandas as pd` and `import numpy as np`. The second cell, labeled '[6]:', contains `df = pd.read_csv('driver-data.csv')` followed by `df.shape`. The output of the second cell is displayed below it as `[6]: (4000, 3)`. The third cell, labeled '[ ]:', is currently empty with a cursor at the end of the line.

```
[4]: import pandas as pd
import numpy as np

[6]: df = pd.read_csv('driver-data.csv')
df.shape

[6]: (4000, 3)

[ ]: |
```

# Pandas Functions: Example 3

The example uses `df.info()` function to output the information of the dataset.

Example:

```
import pandas as pd
import numpy as np
df = pd.read_csv('driver-data.csv')
df.info
```

Output:

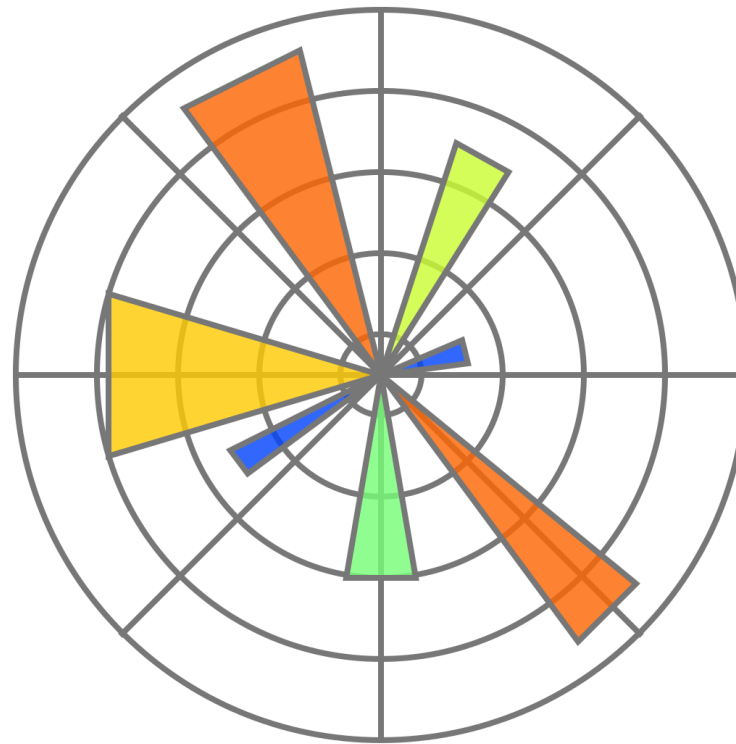
```
[7]: <bound method DataFrame.info of
0      3423311935      71.24      id  mean_dist_day  mean_over_speed_perc
1      3423313212      52.53      25
2      3423313724      64.54      27
3      3423311373      55.69      22
4      3423310999      54.58      25
...      ...      ...      ...
3995  3423310685     160.04      10
3996  3423312600     176.17       5
3997  3423312921     170.91      12
3998  3423313630     176.14       5
3999  3423311533     168.03       9

[4000 rows x 3 columns]>
```

# Matplotlib

# Matplotlib

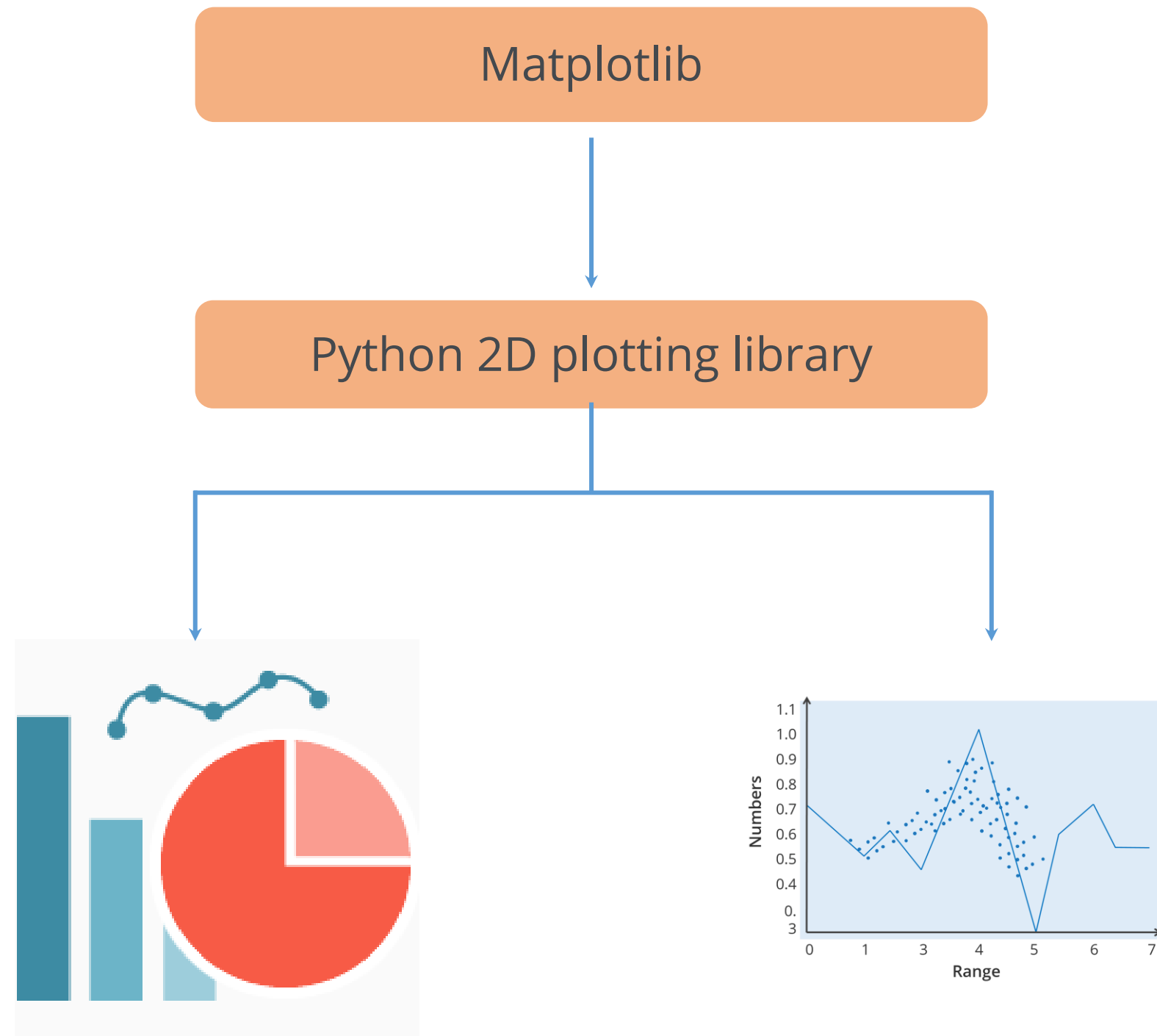
Matplotlib is a visualization tool that uses a low-level graph plotting library written in Python.



Matplotlib is an open source and can be used freely.

# Matplotlib

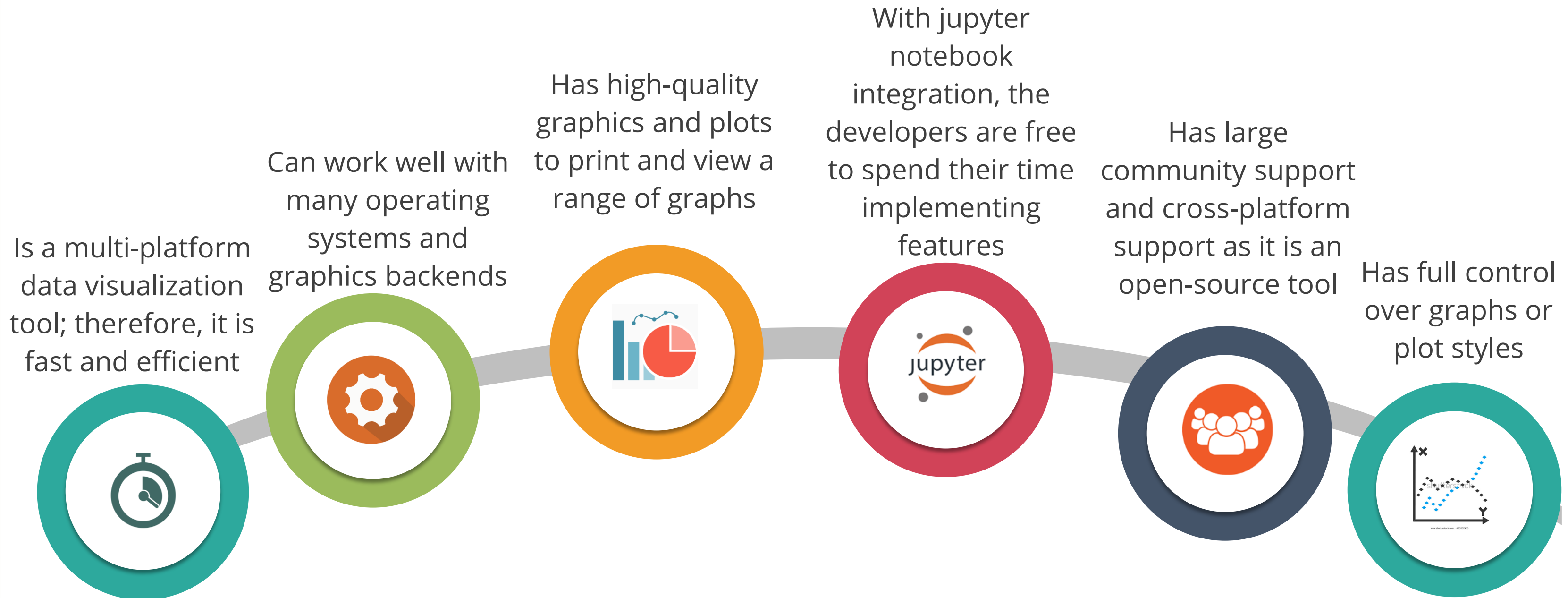
Using Python's matplotlib makes data visualization of large and complex data easy.





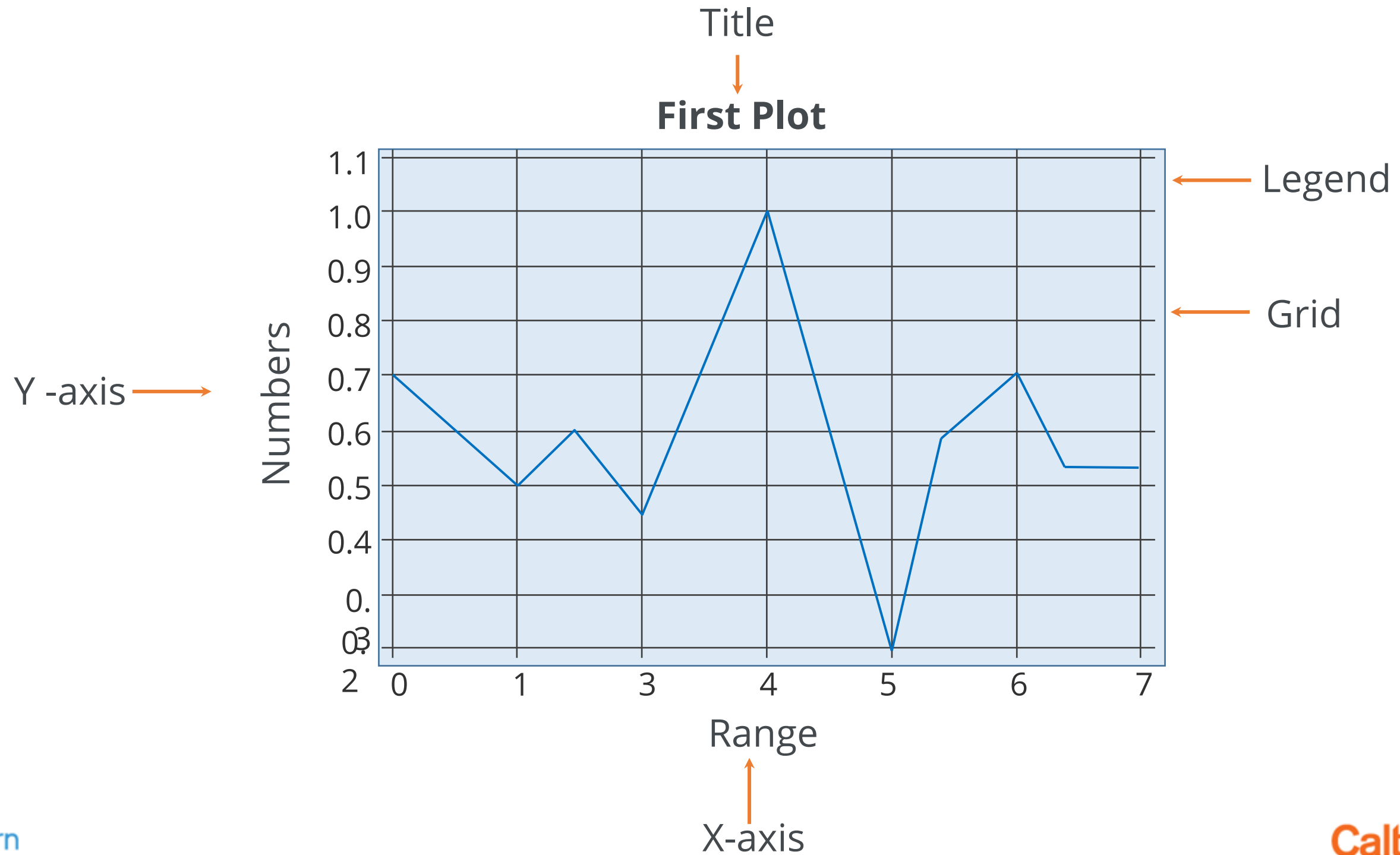
# Matplotlib: Advantages

There are several advantages of using matplotlib to visualize data. They are as follows:



# The Plot

A plot is a graphical representation of data, which shows the relationship between two variables or the distribution of data.



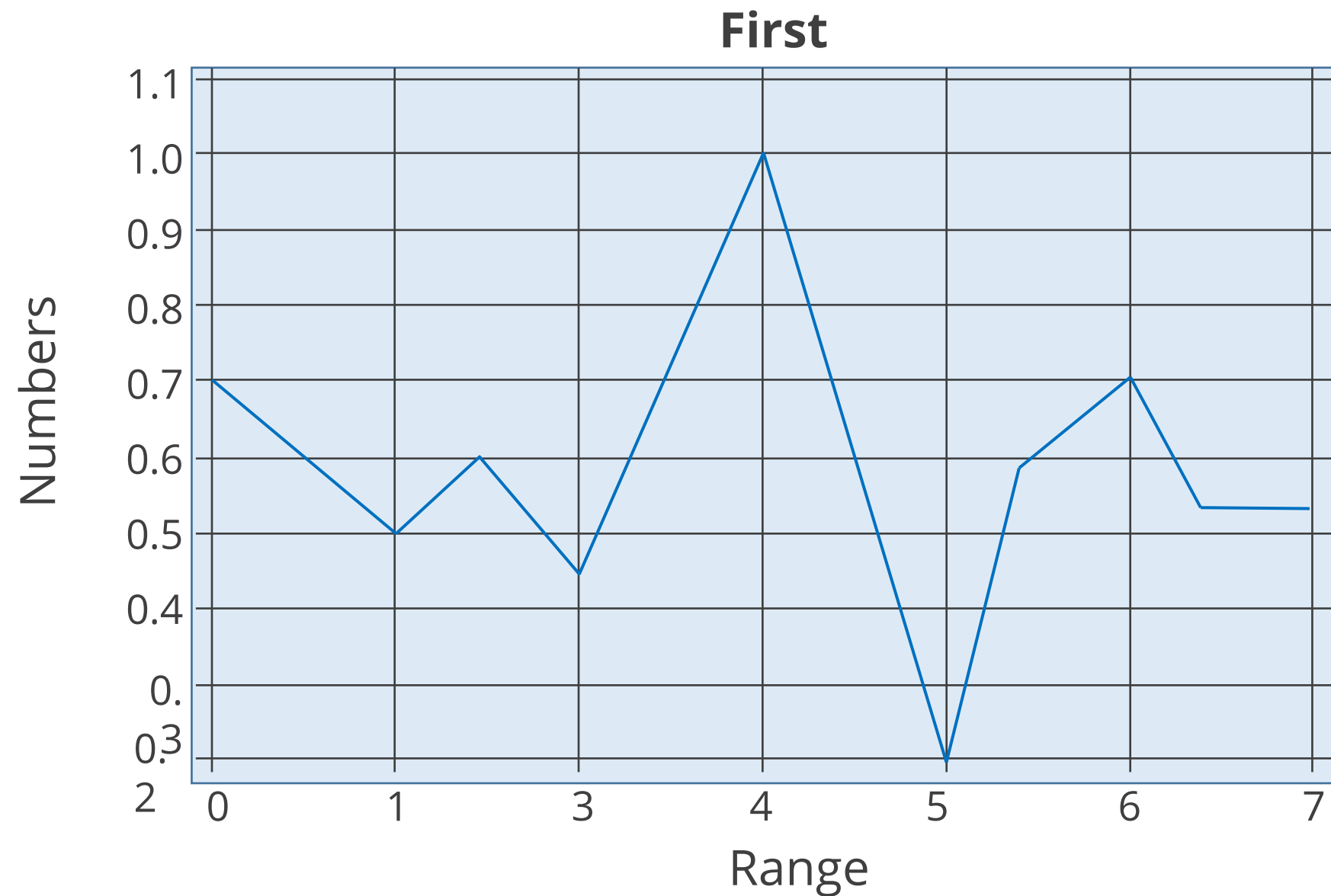
# Steps to Create a Plot

You can create a plot using four simple steps.



# Steps to Create Plot: Example

Let us consider the same example plot used earlier.



Follow the steps to obtain this plot.

# Steps to Create Plot: Example

```
In [1]: #import numpy for generating random numbers
import numpy as np
#import matplotlib library
import matplotlib.pyplot as plt
from matplotlib import style
%matplotlib inline
```

Generate random numbers → numpy  
Plot the numbers → pyplot  
set the grid style → style

```
In [21]: #generate random numbers (total 10)
randomNumber = np.random.rand(10)
```

used numpy random method → Defined the dataset

```
In [22]: #view them
print(randomNumber)
```

view the created random numbers → Print method

```
[ 0.71892609  0.49065612  0.61092193  0.43397501  0.94771363  0.31505178
 0.58568599  0.6929941   0.4288734   0.43774794]
```

```
In [23]: #select the style of the plot
style.use('ggplot')
#plot the random number
plt.plot(randomNumber, 'g', label='line one', linewidth=2)
#x axis is number of random numbers (index)
plt.xlabel('Range')
#y axis is actual random number
plt.ylabel('Numbers')
#Title of the plot
plt.title('First Plot')

plt.legend()
plt.show()
```

ggplot → Set the style  
Set the legend  
Set line width  
Set coordinates labels  
Set the title  
Plot the graph  
Display the created plot

Step 01

Import the required libraries

Step 02

Define or import the required dataset

Step 03

Set the plot parameters

Step 04

Display the created plot

SciPy

# SciPy

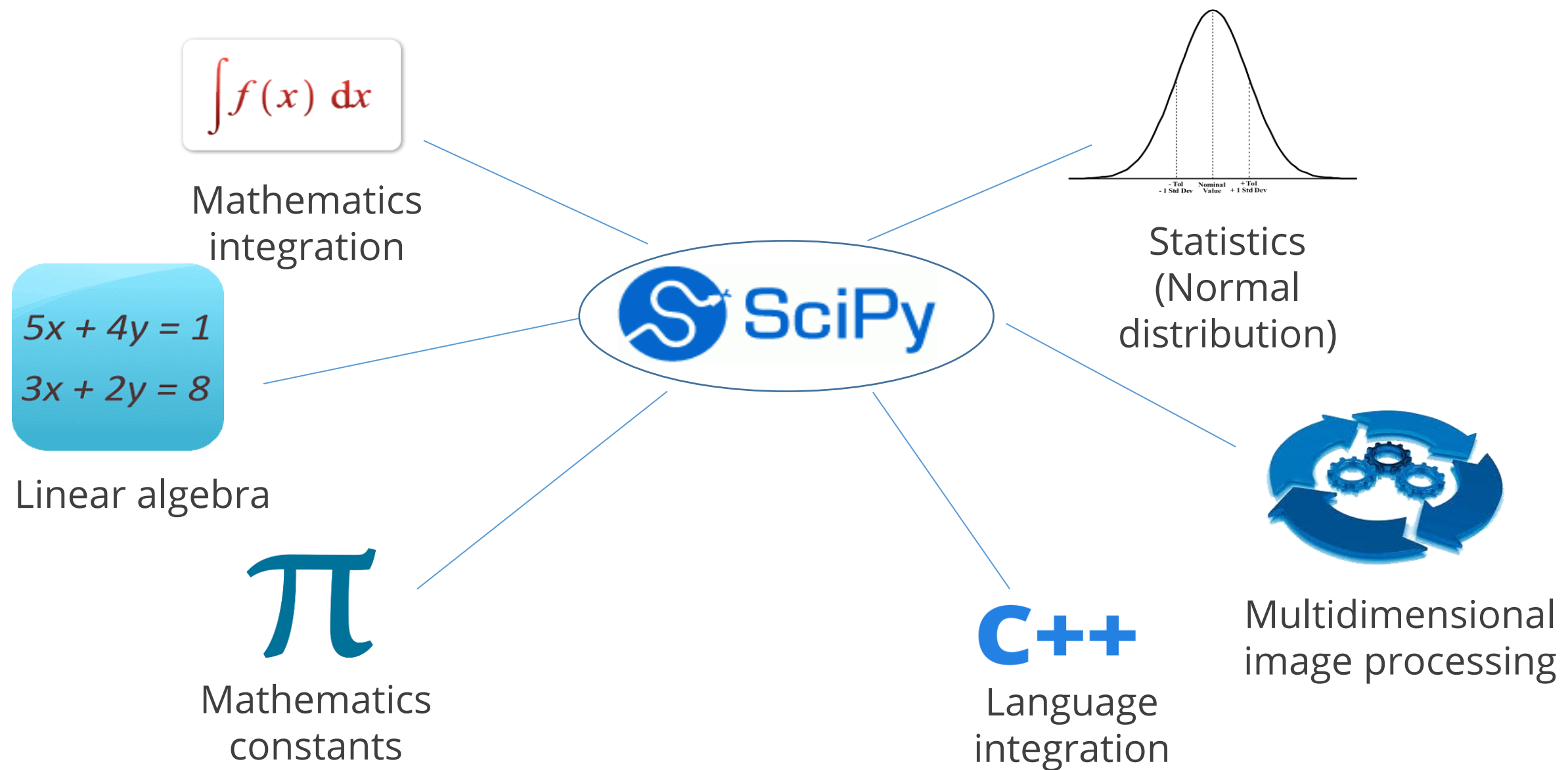
SciPy is a free and open-source Python library used for scientific and technical computing.



It has greater optimization, statistics, and signal processing functions.

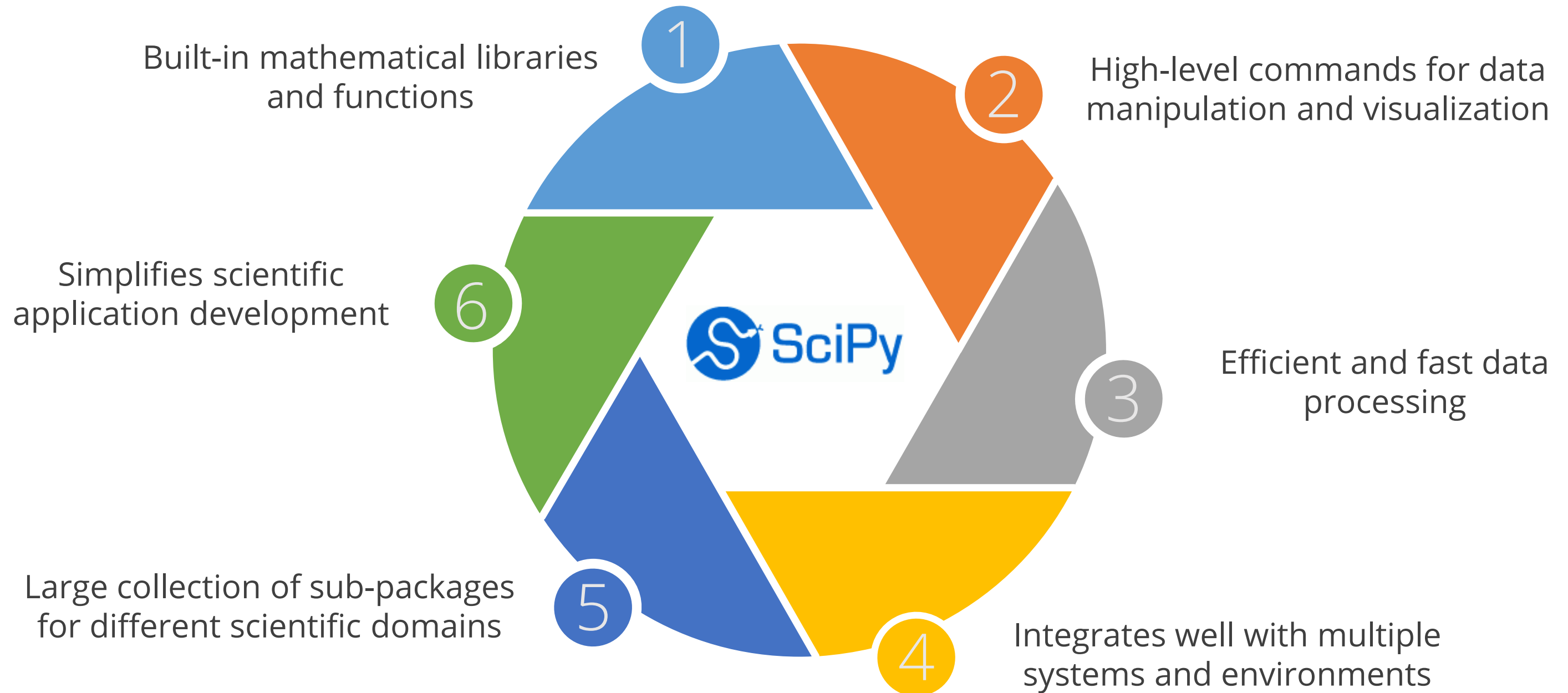
# SciPy

SciPy has built-in packages that help in handling the scientific domains.



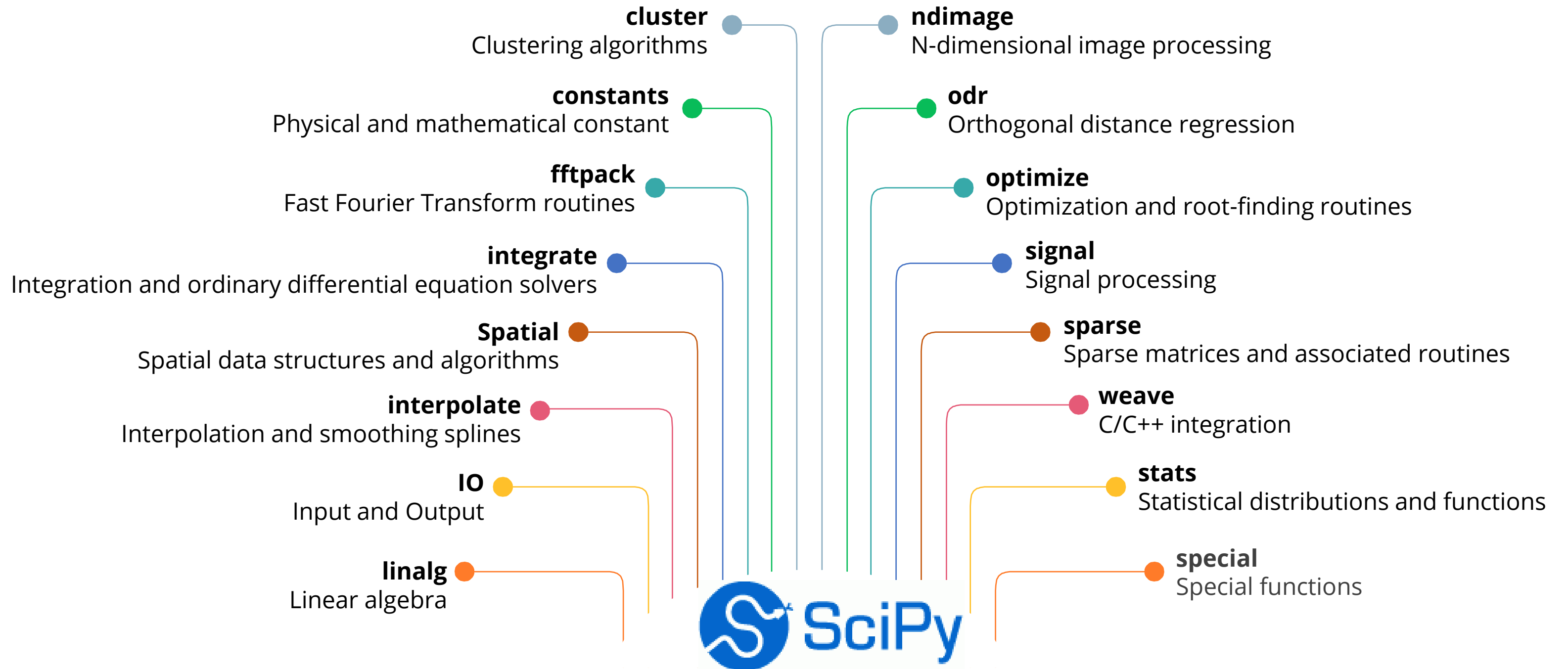


# SciPy and Its Characteristics



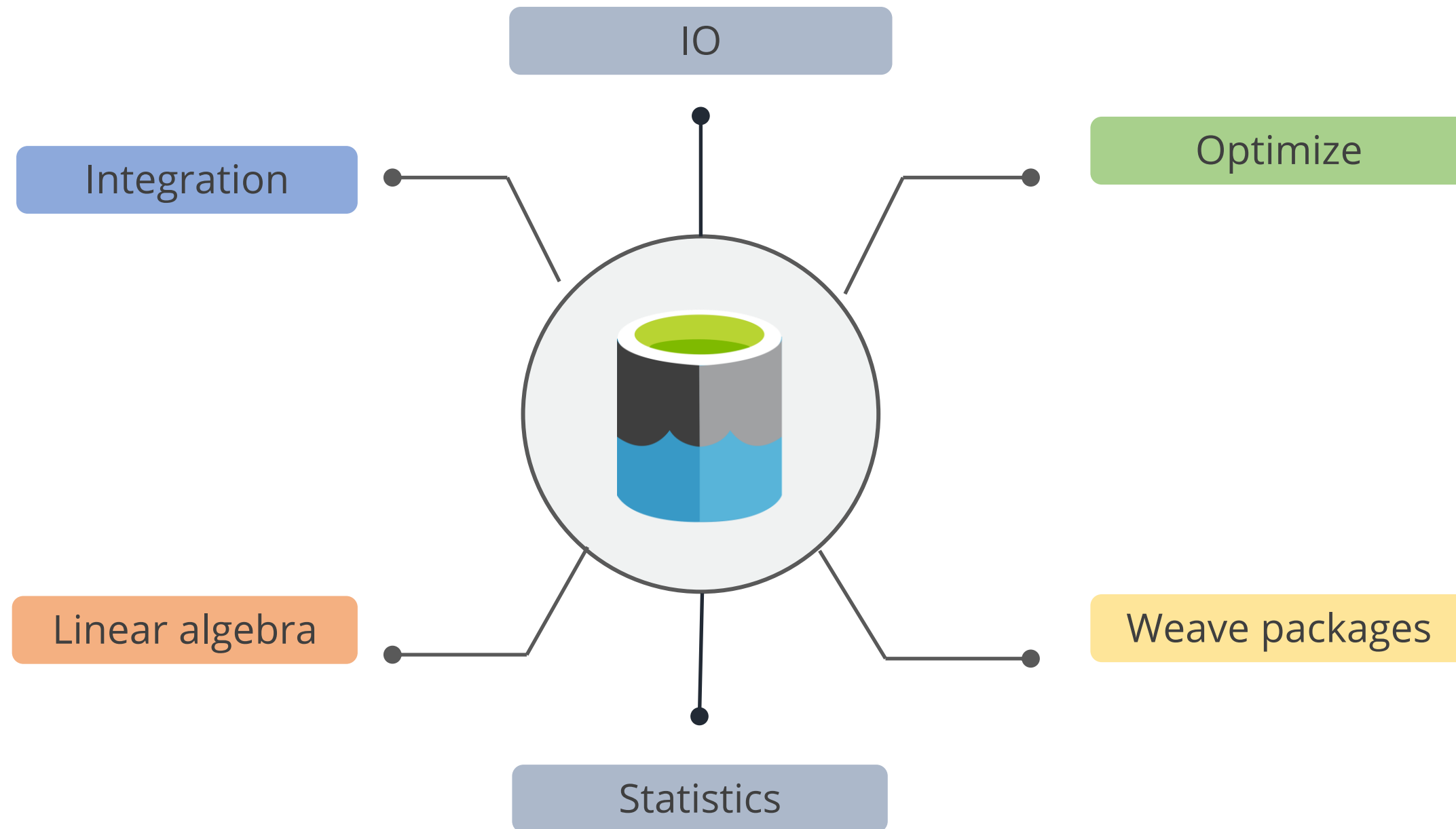
# SciPy Sub-Package

SciPy has multiple sub-packages which handle different scientific domains.



# SciPy Packages

Some widely used packages are:



# SciPy Packages: Example 1

Let's look at SciPy with `scipy.linalg` as an example.

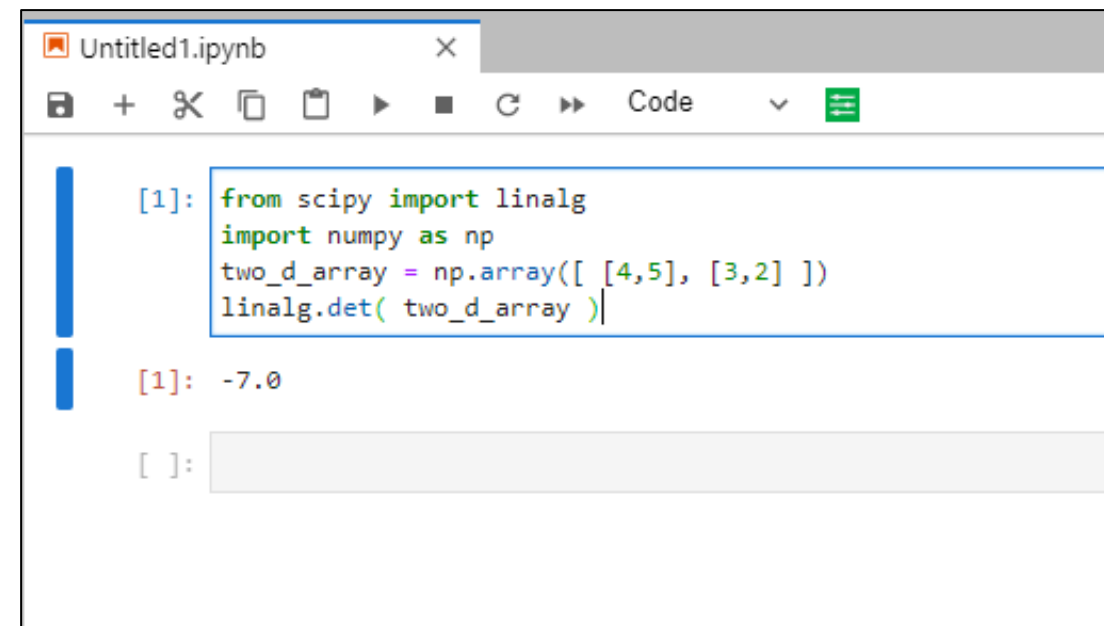
Example:

```
from scipy import linalg
import numpy as np

two_d_array = np.array([ [4,5], [3,2] ])

linalg.det( two_d_array )
```

Output:



The screenshot shows a Jupyter Notebook window titled 'Untitled1.ipynb'. The code cell contains the following Python code:

```
[1]: from scipy import linalg
import numpy as np
two_d_array = np.array([ [4,5], [3,2] ])
linalg.det( two_d_array )
```

The output cell shows the result of the calculation:

```
[1]: -7.0
```

Below the output, there is an empty cell with the prompt `[ ]:`.

The example above calculates the determinant of a two-dimensional matrix.

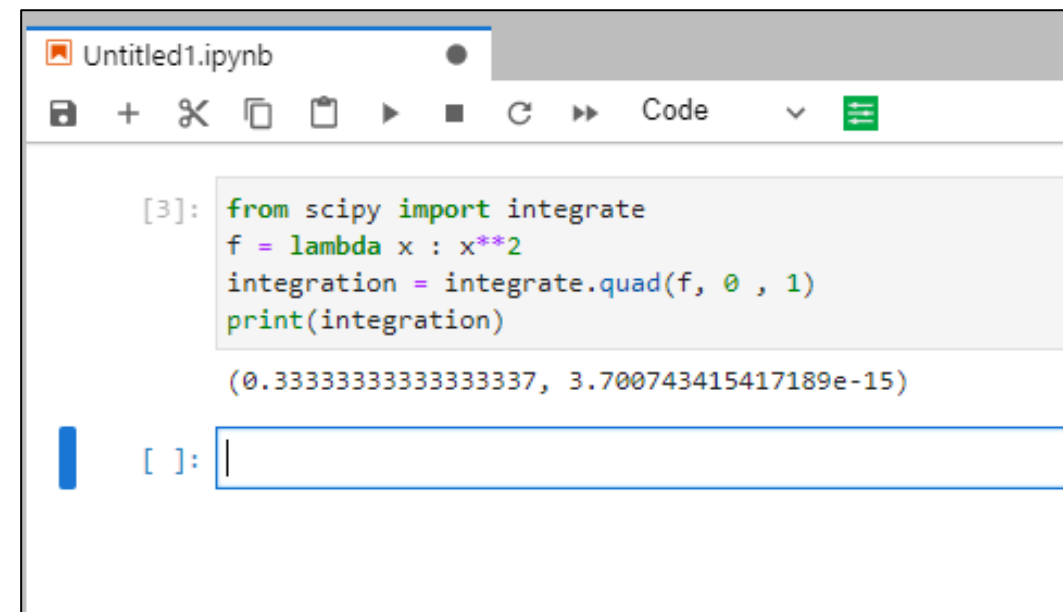
## SciPy Packages: Example 2

Let's look at SciPy with `scipy.integrate` as an example.

Example:

```
from scipy import integrate
f = lambda x : x**2
integration = integrate.quad(f, 0 , 1)
print(integration)
```

Output:

A screenshot of a Jupyter Notebook window titled 'Untitled1.ipynb'. The notebook shows a code cell with the following Python code: 

```
[3]: from scipy import integrate
f = lambda x : x**2
integration = integrate.quad(f, 0 , 1)
print(integration)
```

 Below the code cell, the output is displayed: 

```
(0.3333333333333337, 3.700743415417189e-15)
```

 A new empty code cell is visible below the output, starting with 

```
[ ]: |
```

.

```
Untitled1.ipynb
[3]: from scipy import integrate
f = lambda x : x**2
integration = integrate.quad(f, 0 , 1)
print(integration)

(0.3333333333333337, 3.700743415417189e-15)

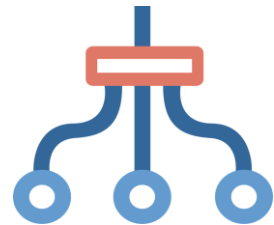
[ ]: |
```

In this example, the function returns two values in which the first value is integration, and the second value is the estimated error in integral.

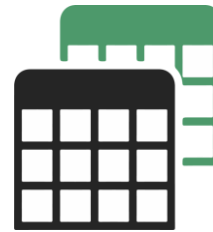
# Scikit-Learn

# Scikit-Learn

Scikit is a powerful and modern machine learning Python library. It is used for fully- and semi-automated data analysis and information extraction.



Allows many tools to identify, organize, and solve real-life problems



Provides a collection of free downloadable datasets



Consists of many libraries to learn and predict

# Scikit-Learn

Scikit is a powerful and modern machine learning Python library. It is used for fully- and semi-automated data analysis and information extraction.



Provides model support for every problem type



Maintains model persistence



Provides open-source community and vendor support



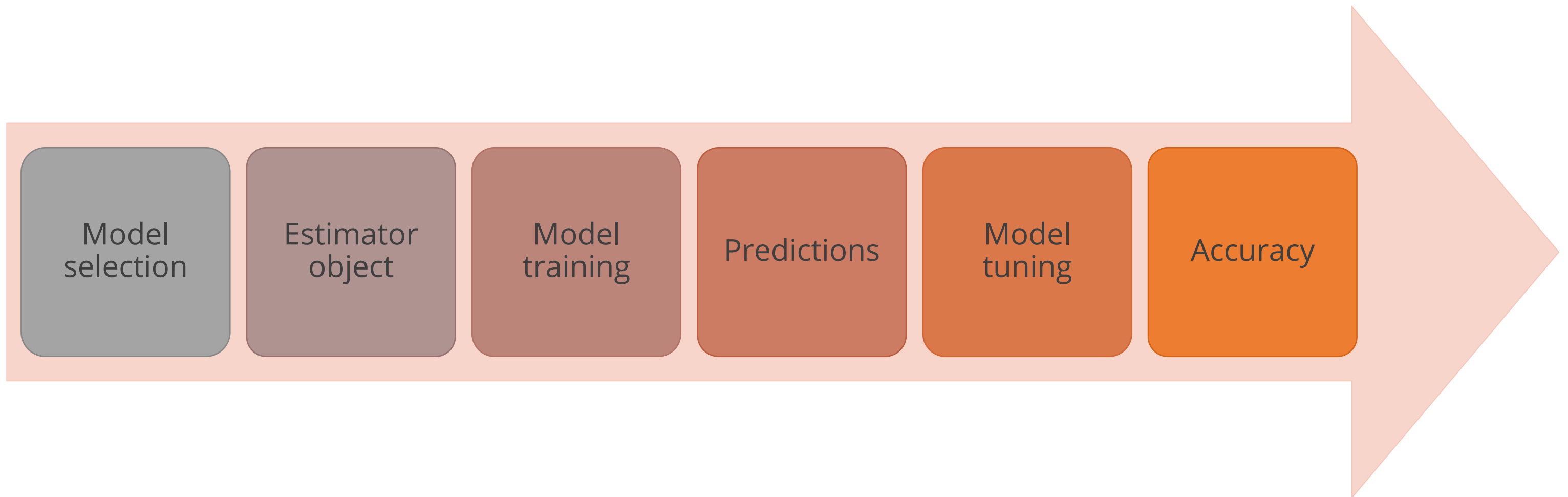
# Scikit-Learn



- It is also known as sklearn.
- It is used to build a machine learning model that has various features such as classification, regression, and clustering.
- It includes algorithms such as k-means, k-nearest neighbors, support vector machine (SVM), and decision tree.

# Scikit-Learn: Problem-Solution Approach

Scikit-learn helps data scientists and machine learning engineers to solve problems using the problem-solution approach.



# Scikit-Learn: Problem-Solution Considerations

---

Points to be considered while working with a scikit-learn dataset or loading the data to scikit-learn:



Create separate objects for features and responses



Ensure features and responses only have numeric values



Verify that the features and responses are in the form of a NumPy ndarray



Check features and responses have the same shape and size as the array

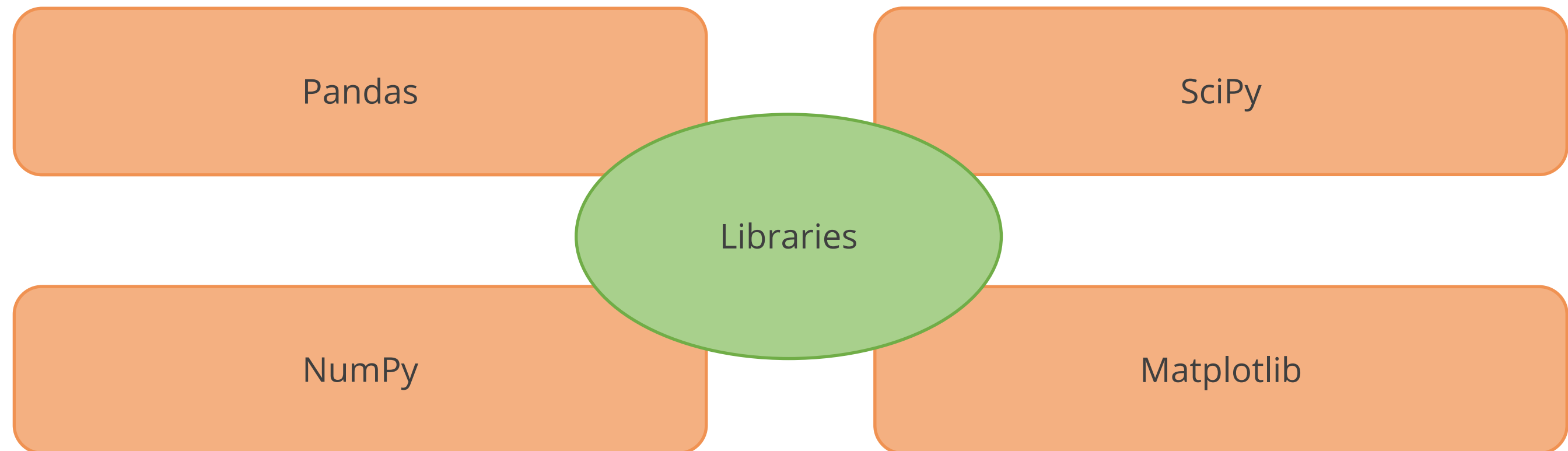


Ensure features are always mapped as  $x$ , and responses as  $y$

# Scikit-Learn: Prerequisite for Installation

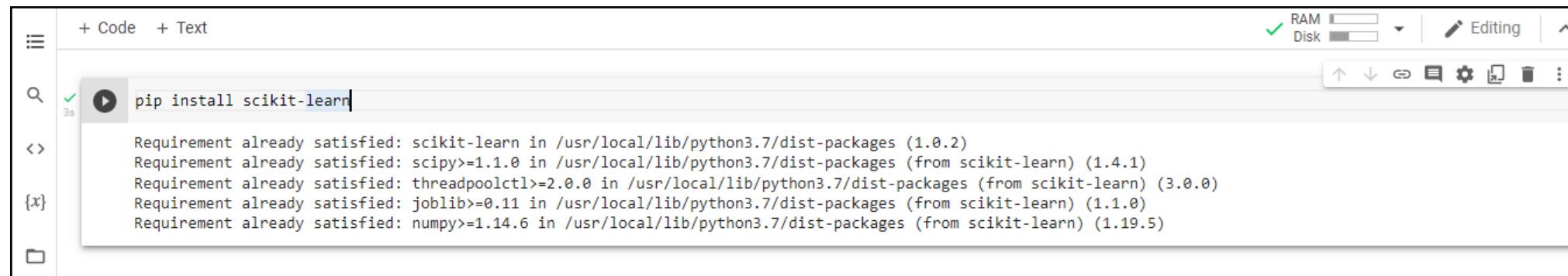
---

The libraries that must be installed before installing Scikit-learn are:



# Scikit-Learn: Installation

To install scikit-learn in Jupyter notebook via pip,  
enter the code:  
`pip install scikit-learn`



The screenshot shows a Jupyter Notebook interface. At the top, there are tabs for '+ Code' and '+ Text'. Below the tabs, a code cell contains the command `pip install scikit-learn`. The output of the command is displayed below the code, showing that the requirements are already satisfied for scikit-learn and its dependencies (scipy, threadpoolctl, joblib, and numpy) in the current environment.

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packages (1.0.2)
Requirement already satisfied: scipy>=1.1.0 in /usr/local/lib/python3.7/dist-packages (from scikit-learn) (1.4.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from scikit-learn) (3.0.0)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from scikit-learn) (1.1.0)
Requirement already satisfied: numpy>=1.14.6 in /usr/local/lib/python3.7/dist-packages (from scikit-learn) (1.19.5)
```

To install scikit-learn via command prompt,  
enter the code:  
`conda install scikit-learn`

```
conda install scikit-learn
```

# Scikit-Learn: Models

Some popular groups of models provided by scikit-learn are:

1 Clustering

2 Cross validation

3 Ensemble methods

4 Feature extraction

5 Feature selection

6 Parameter tuning

7 Supervised models

8 Dimensionality reduction

## Web Scrapping with BeautifulSoup

# Web Scraping

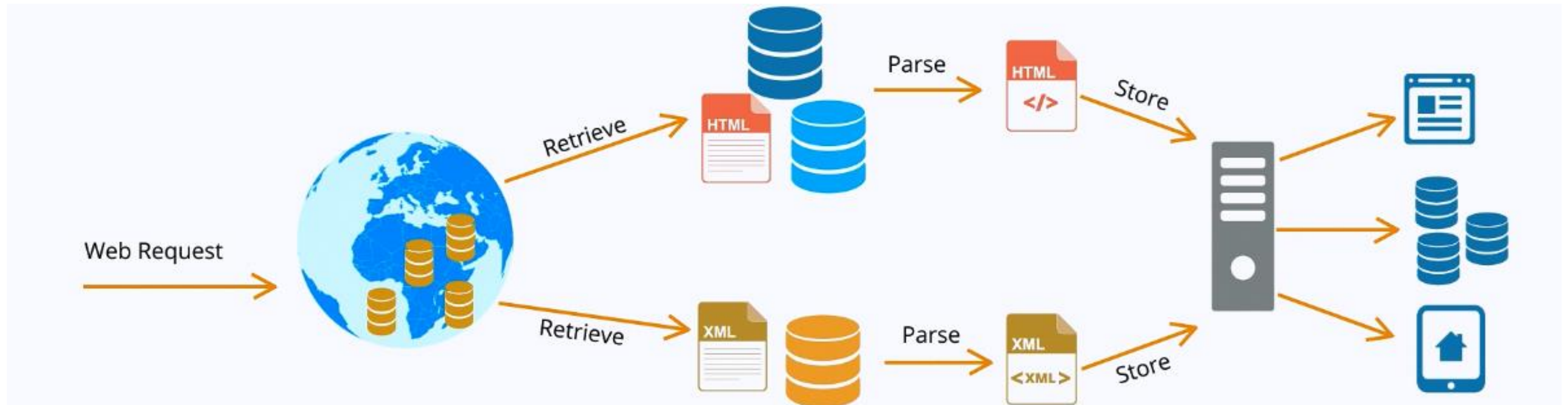
Web scraping is the process of constructing an algorithm that can extract, parse, download, and organize useful information from the web automatically.





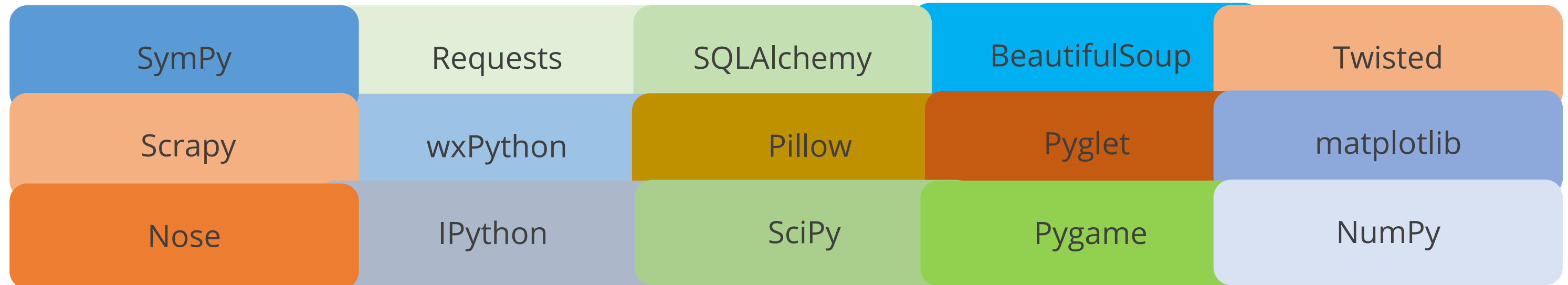
# Web Scraping: Process

The four processes followed in web scraping are web requesting, retrieving, parsing, and storing the desired data format.



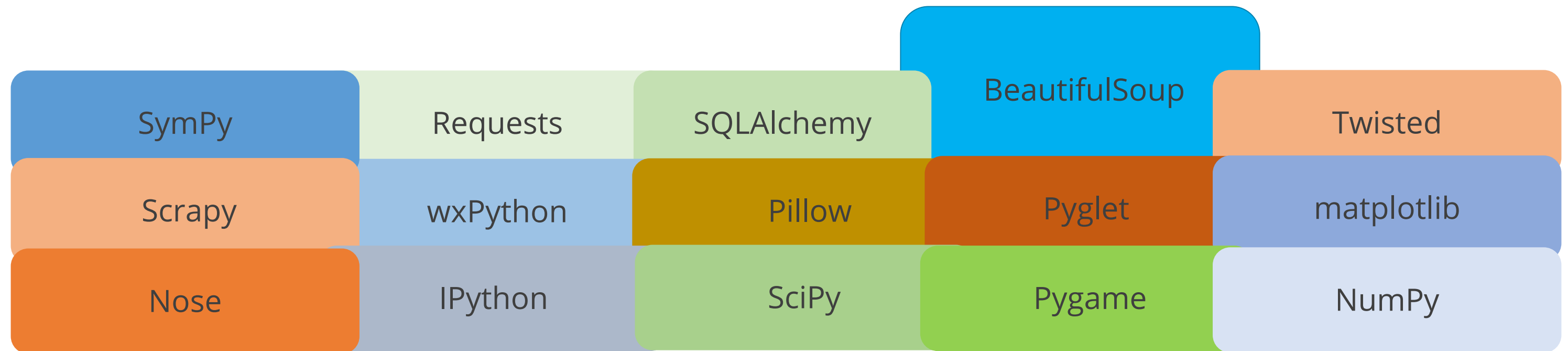
# Web Scrapping: Tools

There are many tools used for web scraping.



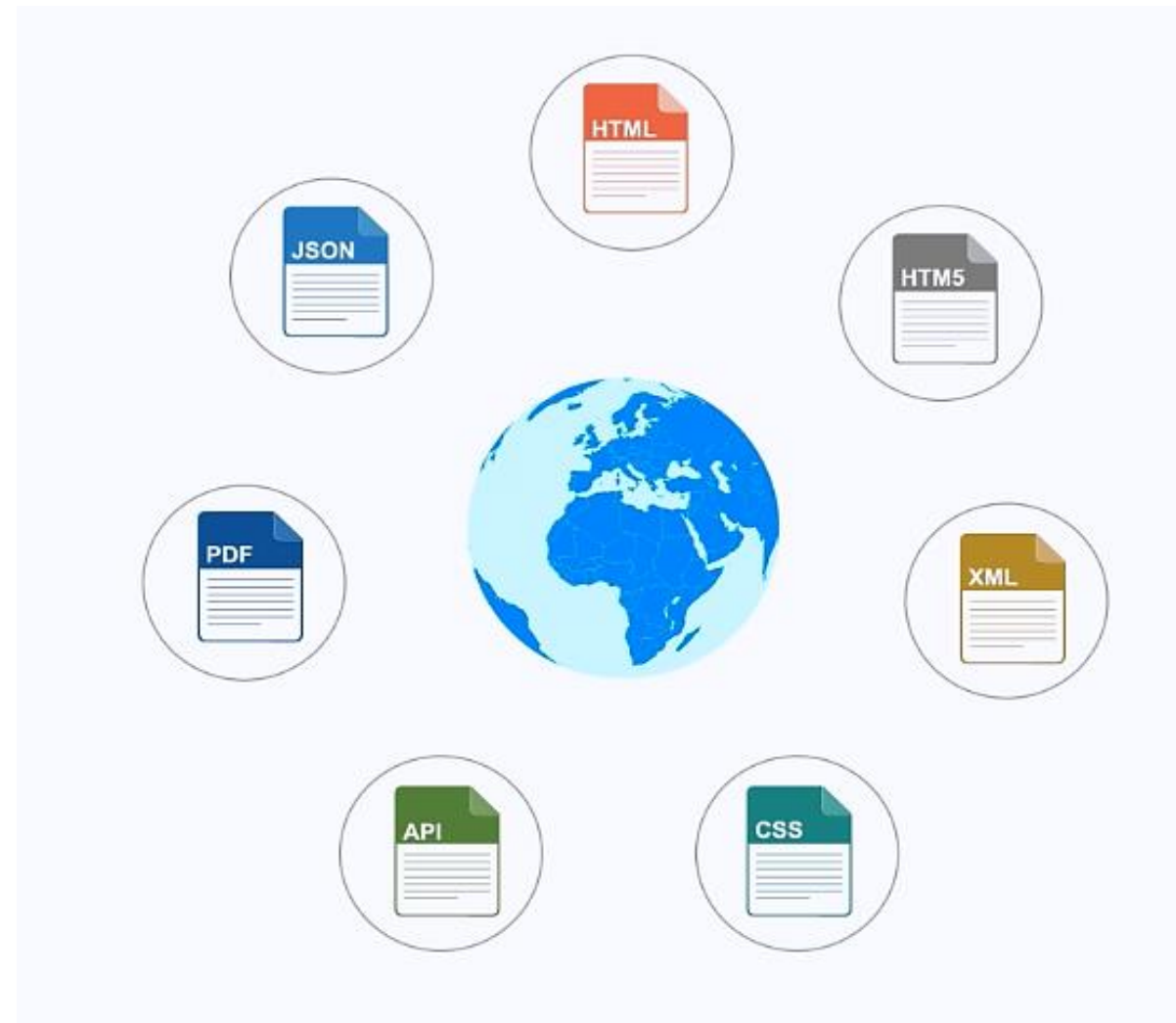
# Web Scraping Tools: BeautifulSoup

BeautifulSoup is an easy, intuitive, and robust Python library designed for web scraping.



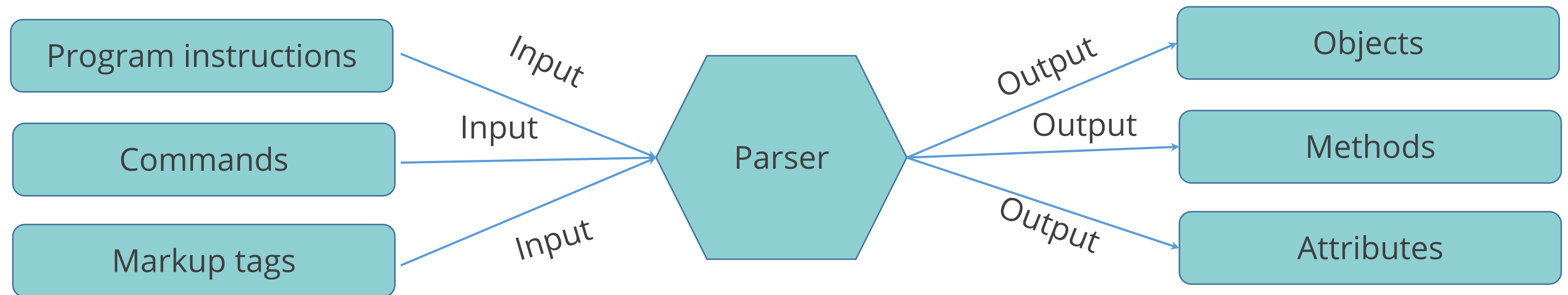
# Common Data or Page Formats on the Web

Data is stored in many file formats on the web and can be processed using web scraping.



# Parser

A parser is a basic tool to interpret or render information from a web document.  
It is also used to validate the input information before processing it.



# BeautifulSoup: Demo

## Example:

```
import requests

from bs4 import BeautifulSoup

URL =
"http://www.values.com/inspirational-quotes"

r = requests.get(URL)

soup = BeautifulSoup(r.content, 'html5lib')
# If this line causes an error, run 'pip install
html5lib' or install html5lib

print(soup.prettify())
```

## Output:

```
[4]: import requests
from bs4 import BeautifulSoup

URL = "http://www.values.com/inspirational-quotes"
r = requests.get(URL)

soup = BeautifulSoup(r.content, 'html5lib') # If this line causes an error, run 'pip install html5lib' or install html5lib
print(soup.prettify())

<!DOCTYPE html>
<html class="no-js" dir="ltr" lang="en-US">
  <head>
    <title>
      Inspirational Quotes - Motivational Quotes - Leadership Quotes | PassItOn.com
    </title>
    <meta charset="utf-8"/>
    <meta content="text/html; charset=utf-8" http-equiv="content-type"/>
    <meta content="IE=edge" http-equiv="X-UA-Compatible"/>
    <meta content="width=device-width,initial-scale=1.0" name="viewport"/>
    <meta content="The Foundation for a Better Life | Pass It On.com" name="description"/>
    <link href="/apple-touch-icon.png" rel="apple-touch-icon" sizes="180x180"/>
    <link href="/favicon-32x32.png" rel="icon" sizes="32x32" type="image/png"/>
    <link href="/favicon-16x16.png" rel="icon" sizes="16x16" type="image/png"/>
    <link href="/site.webmanifest" rel="manifest"/>
    <link color="#c8102e" href="/safari-pinned-tab.svg" rel="mask-icon"/>
    <meta content="#c8102e" name="msapplication-TileColor"/>
    <meta content="#ffffff" name="theme-color"/>
    <link crossorigin="anonymous" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="sha384-ggOyR0iXCbMQV3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcwr7x9JvoRxT2MZw1T" rel="stylesheet"/>
    <link href="/assets/application-2a7a8e6a1c3f620bac9efa66420f5579.css" media="all" rel="stylesheet"/>
    <meta content="authenticity_token" name="csrf-param"/>
    <meta content="MM+LBgRlFqy8C1IpphSa6b38ZVnlScuAt//QGzcoZZITTHGcyOigQUhFPSsQNP8eZOLYtNylhZwTuofxz0h65Q==" name="csrf-token"/>
  </head>
```



## Knowledge Check



## Knowledge Check

1

Which of the following SciPy sub-packages is incorrect?

- A. `scipy.cluster`
- B. `scipy.source`
- C. `scipy.interpolate`
- D. `scipy.signal`





## Knowledge Check

1

Which of the following SciPy sub-packages is incorrect?

- A. `scipy.cluster`
- B. `scipy.source`
- C. `scipy.interpolate`
- D. `scipy.signal`



The correct answer is **B**

`scipy.source` is not a sub-package of SciPy.

## Knowledge Check

2

\_\_\_\_\_ is an important library used for analyzing data.

- A. Math
- B. Random
- C. Pandas
- D. None of the above



## Knowledge Check

2

\_\_\_\_\_ is an important library used for analyzing data.

- A. Math
- B. Random
- C. Pandas
- D. None of the above



The correct answer is **C**

Pandas is an important library used for analyzing data.

## Knowledge Check

2

Matplotlib is a \_\_\_\_\_plotting library.

- A. 1D
- B. 2D
- C. 3D
- D. All of the above



## Knowledge Check

2

Matplotlib is a \_\_\_\_\_plotting library.

- A. 1D
- B. 2D
- C. 3D
- D. All of the above



The correct answer is **B**

Matplotlib is a 2D plotting library.

## Key Takeaways

- ❶ SciPy is a free and open-source Python library used for scientific and technical computing.
- ❷ NumPy (Numerical Python) is a library that consists of multidimensional array objects and a collection of functions for manipulating them.
- ❸ Matplotlib is a visualization tool that uses a low-level graph plotting library written in Python.
- ❹ Scikit is a powerful and modern machine learning Python library. It is used for fully- and semi-automated data analysis and information extraction.

