VINUNIVERSITY

Lecture 25
Python for Machine Learning I

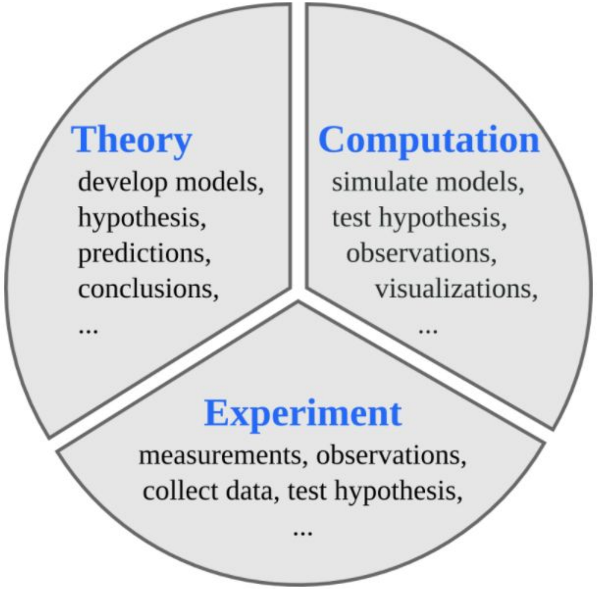
(Numpy, Pandas and Matplotlib)

Hieu Pham, CECS

Hanoi, Dec. 2021

The role of computing in science

- Science has traditionally been divided into experimental and theoretical disciplines.
- During the last several decades, computing has emerged as a very important part of science.
- Computational work is an important complement to both experiments and theory, serving for numerical calculations, simulations or computer modeling.



The figure is in the public domain and may be used without permission.

What is Machine Learning?

What is Machine Learning?

- Machine Learning (ML) is that field of computer science with the help of which computer systems can provide sense to data in much the same way as human beings do.

- ML is a type of artificial intelligence that extract patterns out of raw data by using an algorithm or method, allowing computer systems learn from experience without human intervention.

What is Machine Learning?

Machine learning = Data + Representation Learning

- Work with high-dimensional, large-scale datasets
- Parallel processing with processes and threads
- High-performance computing clusters (e.g. GPUs)

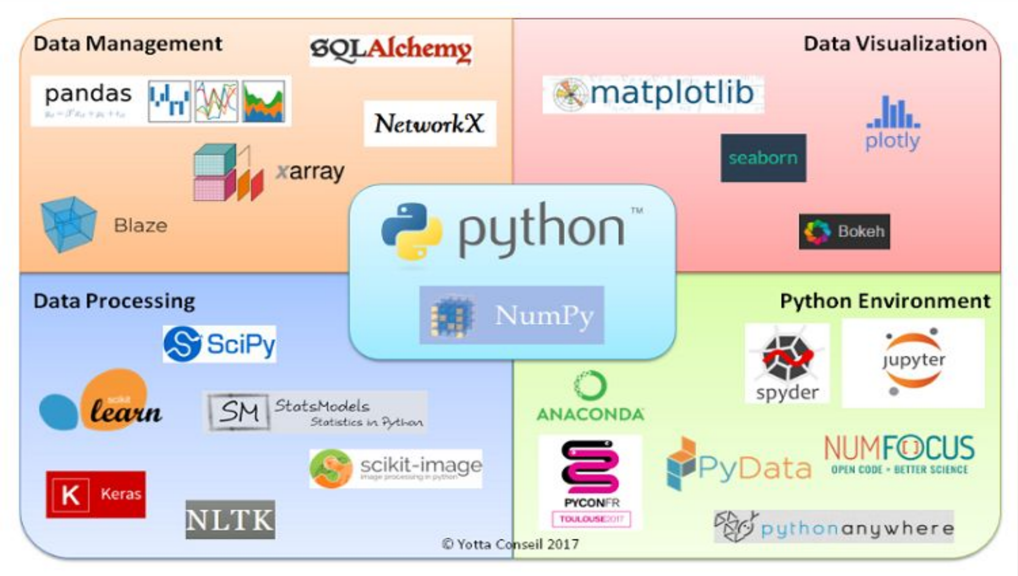
→ need of high-performance computing libraries/frameworks for machine learning

What makes python suitable for scientific computing?

- Python has a strong position in scientific computing with large community of users, easy to find help and documentation.

- Extensive ecosystem of scientific libraries and environments (**Numpy** for Numerical Python, **Scipy** for Scientific Python, and **Matplotlib** for graphics library, etc.

- Support for GPU computing.



The figure is in the public domain and may be used without permission.

Learning outcomes

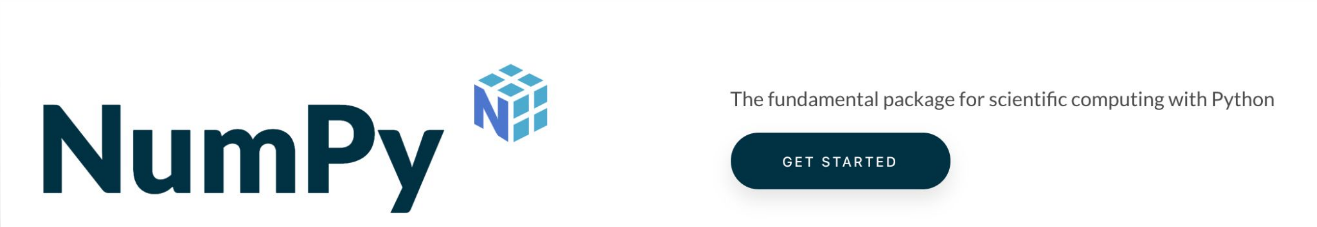
Upon the completion of this lecture, students will be able to:

- Understand and perform the basic operations in NumPy.
- Understand key features of Pandas and use it for processing tabular data.
- Able to use Matplotlib and explore it for visualizing 2D data.

NumPy

- NumPy is the core library for scientific computing in Python and is used in almost all numerical computation using Python.

- It provides high-performance vector, matrix and higher-dimensional data computations. It also has functions for working in domain of linear algebra, fourier transform, etc.

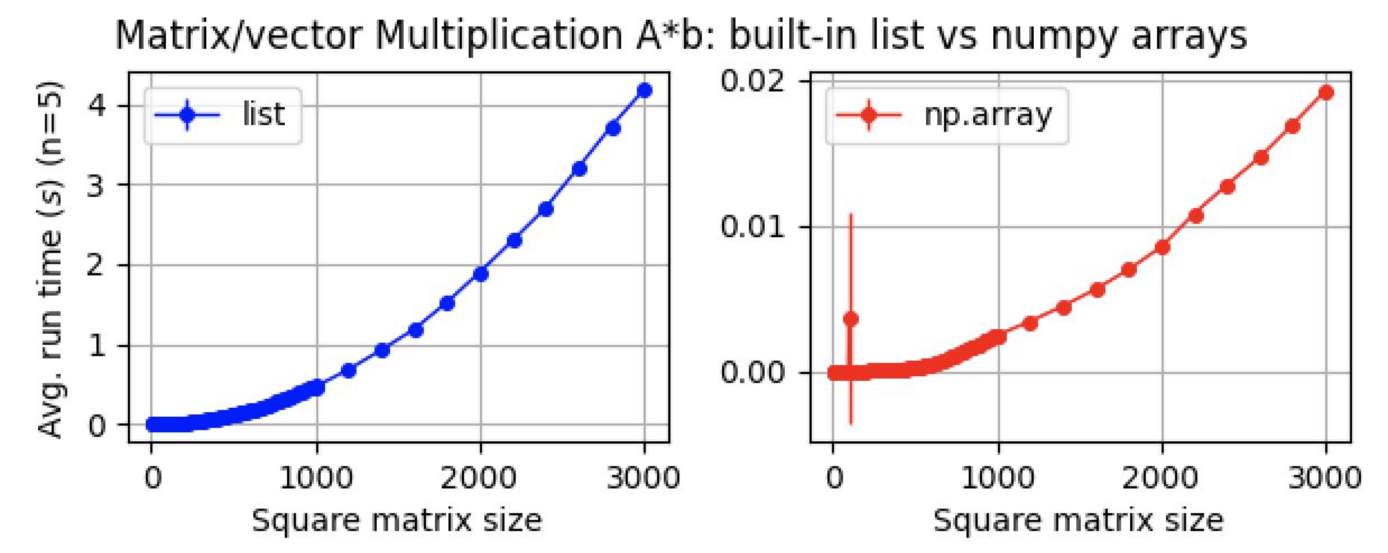


The fundamental package for scientific computing with Python

Official website:
https://numpy.org/

NumPy

In Python we have lists that serve the purpose of arrays, but they are slow to process.
NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.



The figure is in the public domain and may be used without permission.

NumPy

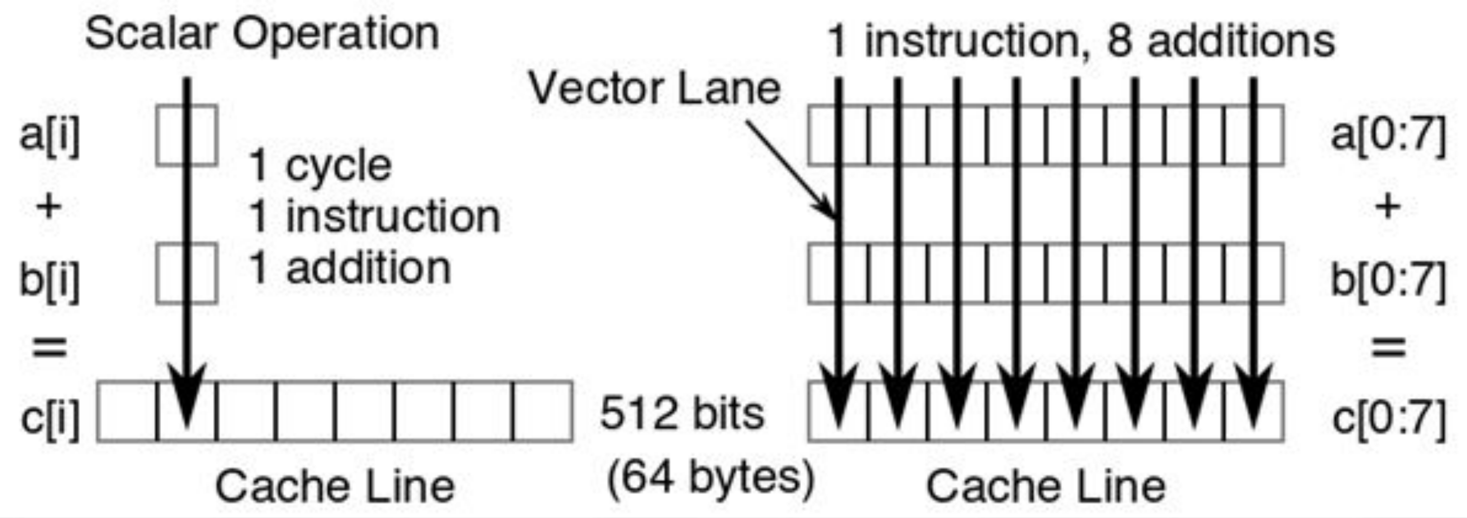
Why is NumPy faster than lists?

- Written in C and Fortran
- Vectorized computations

What is “vectorization”?

NumPy

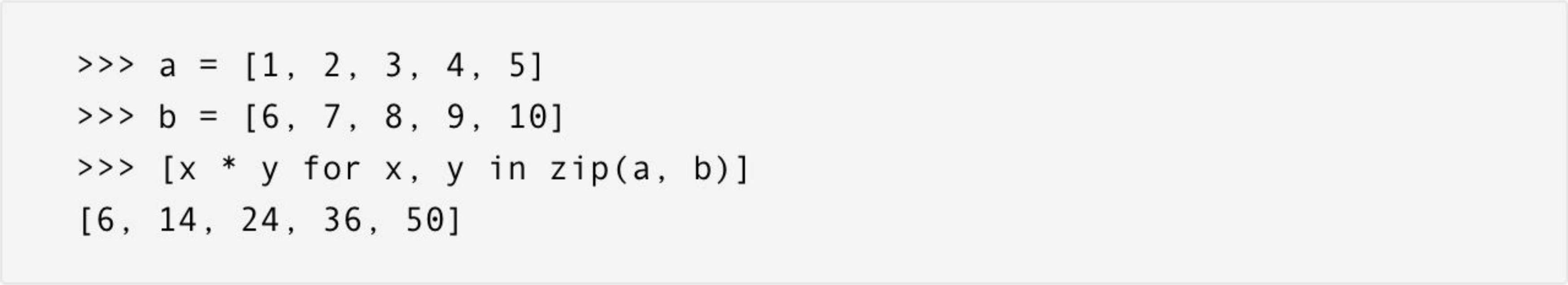
“Vectorization is the process of converting an algorithm from operating on a single value at a time to operating on a set of values (vector) at one time”.



https://livebook.manning.com/book/parallel-and-high-performance-computing/chapter-6/v-5/

NumPy

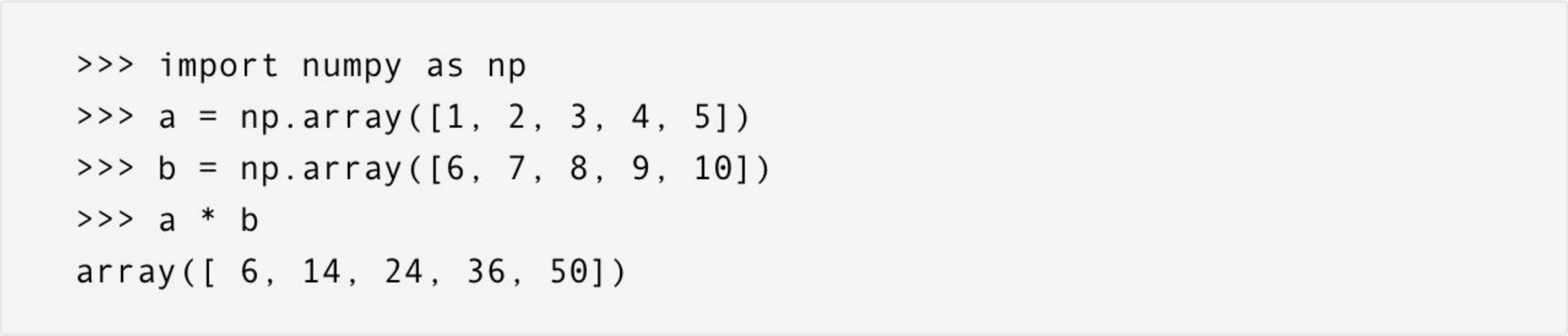
In Python, we can multiply two sequences with a list comprehension:



```
>>> a = [1, 2, 3, 4, 5]
>>> b = [6, 7, 8, 9, 10]
>>> [x * y for x, y in zip(a, b)]
[6, 14, 24, 36, 50]
```

NumPy

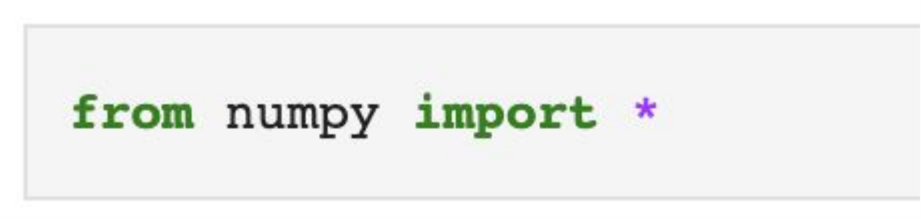
When we put the data into NumPy arrays, we can write the multiplication as follows:



```
>>> import numpy as np
>>> a = np.array([1, 2, 3, 4, 5])
>>> b = np.array([6, 7, 8, 9, 10])
>>> a * b
array([ 6, 14, 24, 36, 50])
```

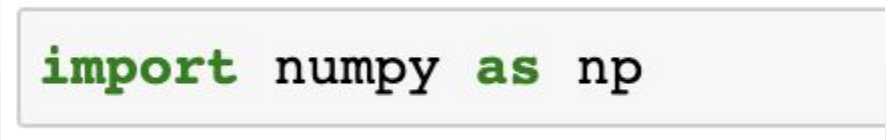
NumPy

To use NumPy you need to import the module, using for example:



```
from numpy import *
```

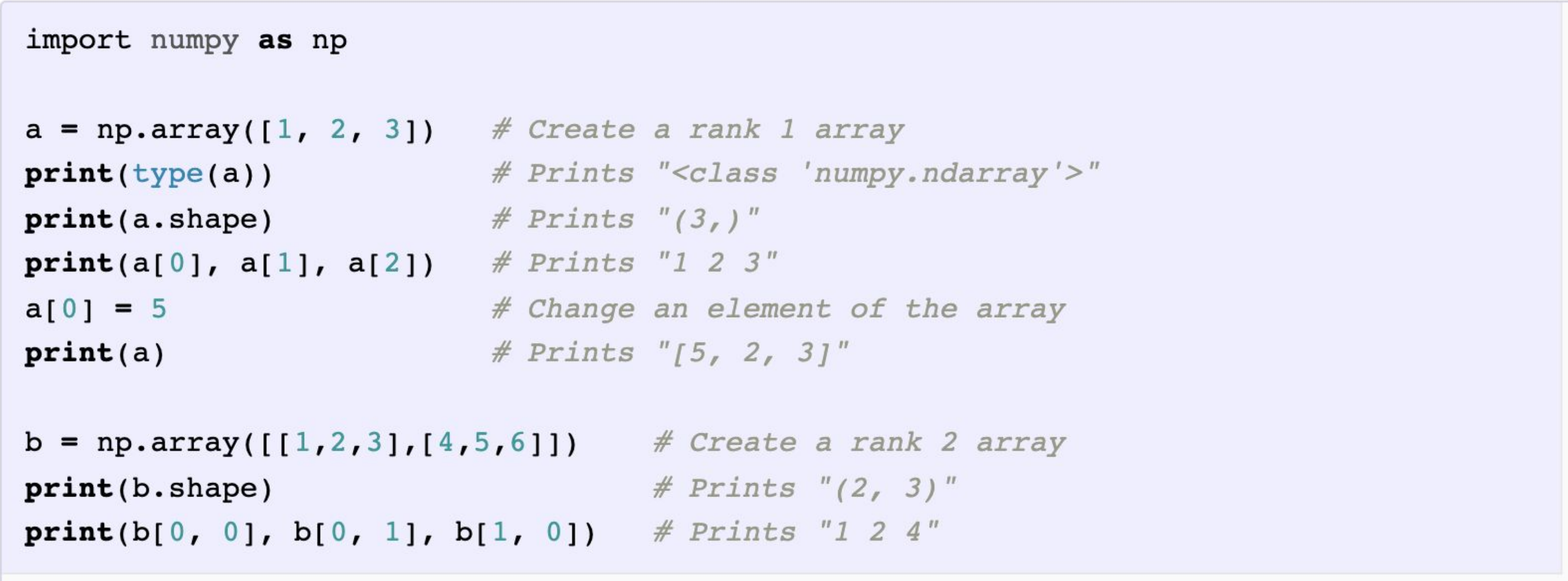
NumPy is usually imported under the np alias. In Python alias are an alternate name for referring to the same thing.



```
import numpy as np
```

NumPy

Create NumPy array: We can initialize NumPy arrays (vector and matrix) from nested Python lists.

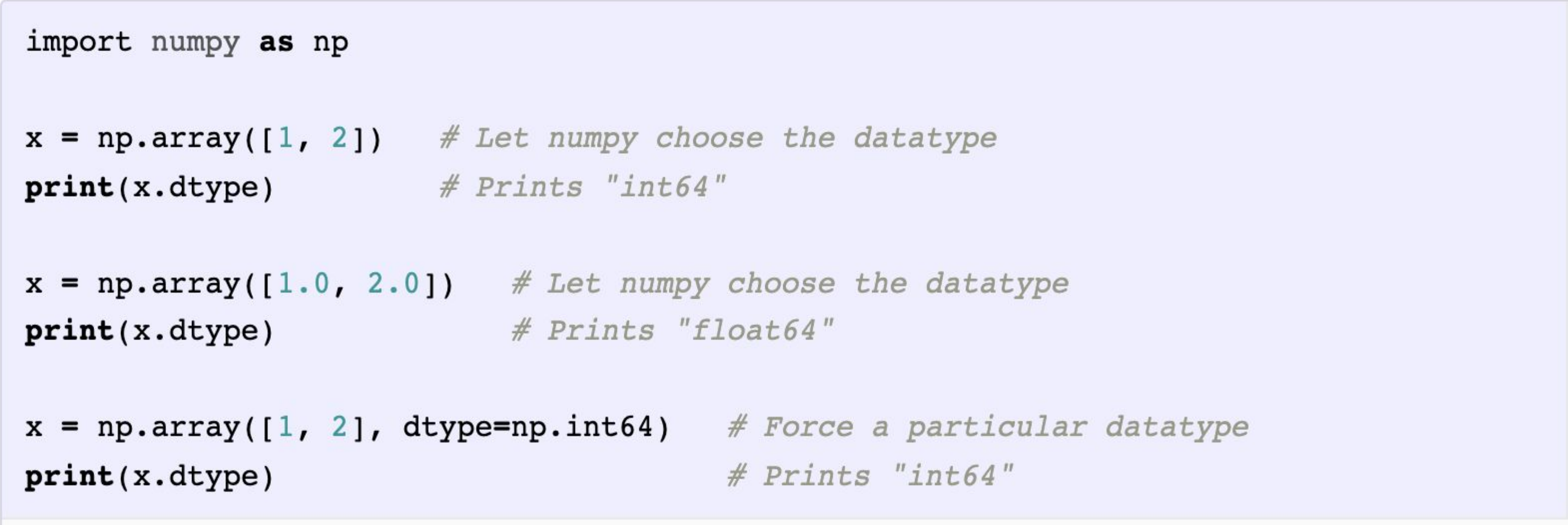


```
import numpy as np

a = np.array([1, 2, 3])   # Create a rank 1 array
print(type(a))            # Prints "<class 'numpy.ndarray'>"
print(a.shape)            # Prints "(3,)"
print(a[0], a[1], a[2])   # Prints "1 2 3"
a[0] = 5                  # Change an element of the array
print(a)                  # Prints "[5, 2, 3]"

b = np.array([[1,2,3],[4,5,6]])    # Create a rank 2 array
print(b.shape)                     # Prints "(2, 3)"
print(b[0, 0], b[0, 1], b[1, 0])   # Prints "1 2 4"
```

NumPy

Datatypes

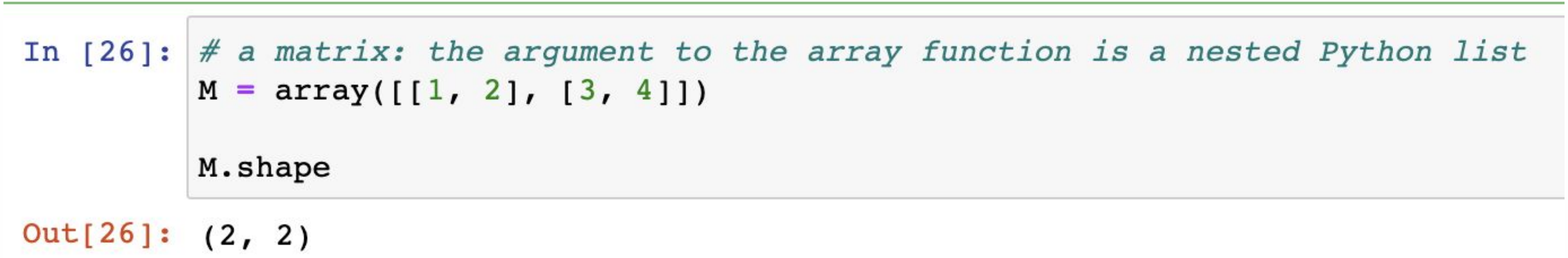
Every numpy array is a grid of elements of the same type. NumPy provides a large set of numeric datatypes that you can use to construct arrays.



```
import numpy as np

x = np.array([1, 2])   # Let numpy choose the datatype
print(x.dtype)         # Prints "int64"

x = np.array([1.0, 2.0])   # Let numpy choose the datatype
print(x.dtype)             # Prints "float64"

x = np.array([1, 2], dtype=np.int64)   # Force a particular datatype
print(x.dtype)                         # Prints "int64"
```

NumPy

We can get information about the shape of an array by using the ***numpy.shape*** property.



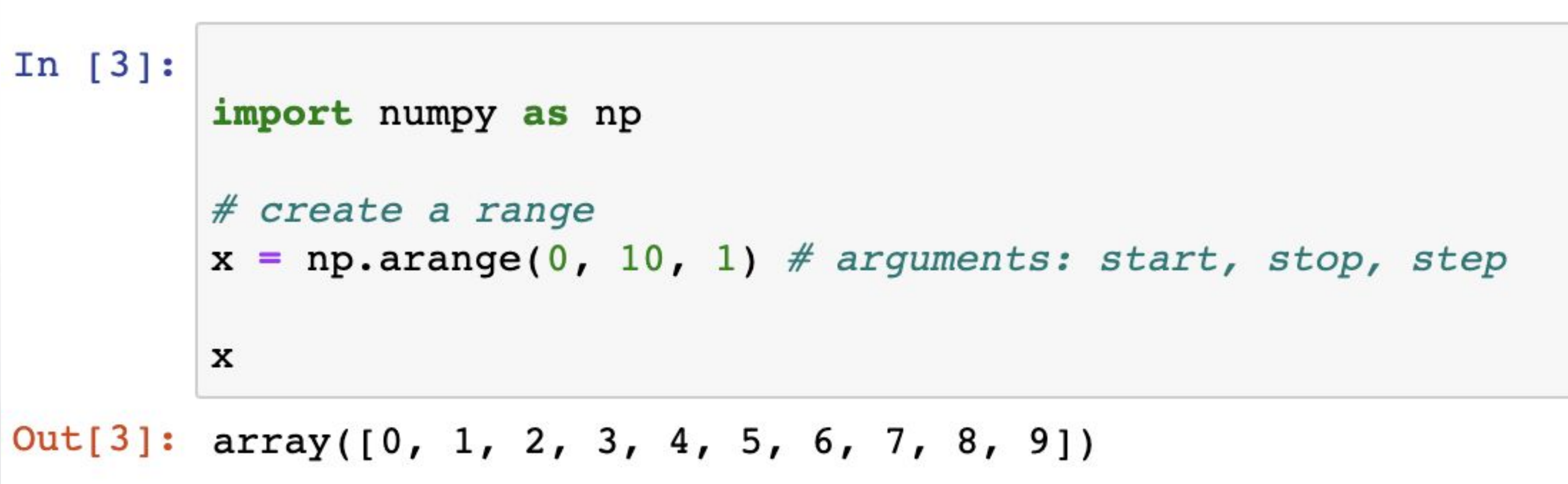
```
In [26]: # a matrix: the argument to the array function is a nested Python list
         M = array([[1, 2], [3, 4]])

         M.shape

Out[26]: (2, 2)
```

Equivalently, we could use the function ***numpy.size***

NumPy

Using array-generating functions

For larger arrays it is inpractical to initialize the data manually, using explicit python lists. Instead we can use one of the many functions in numpy that generate arrays of different forms. Some of the more common are:

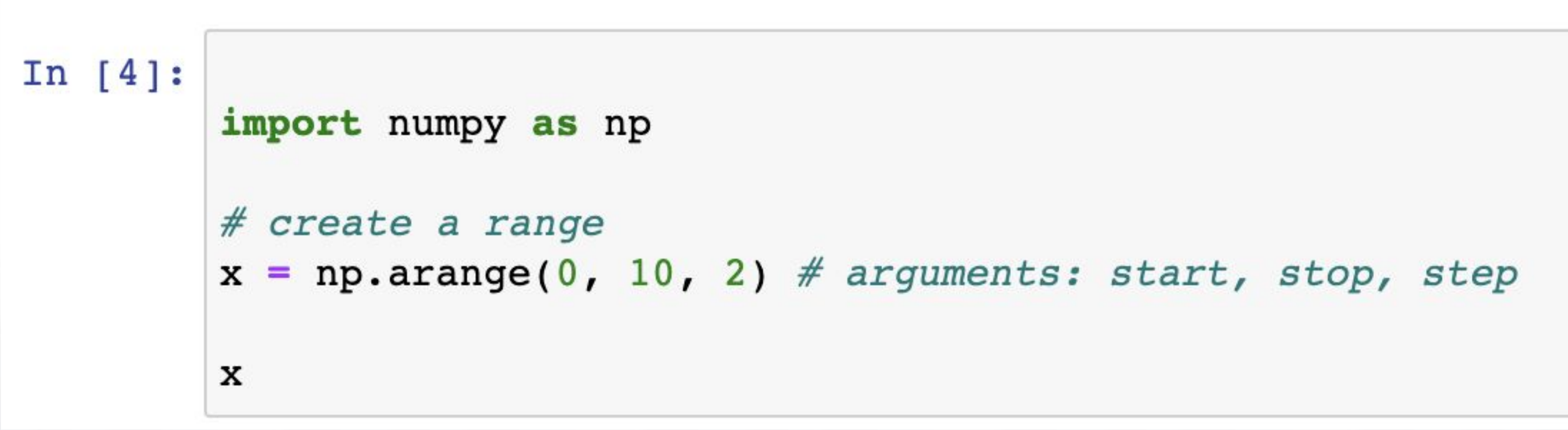


```
In [3]:
import numpy as np

# create a range
x = np.arange(0, 10, 1) # arguments: start, stop, step

x

Out[3]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

NumPy

Using array-generating functions

On Jupyter, what is the output of script given below?

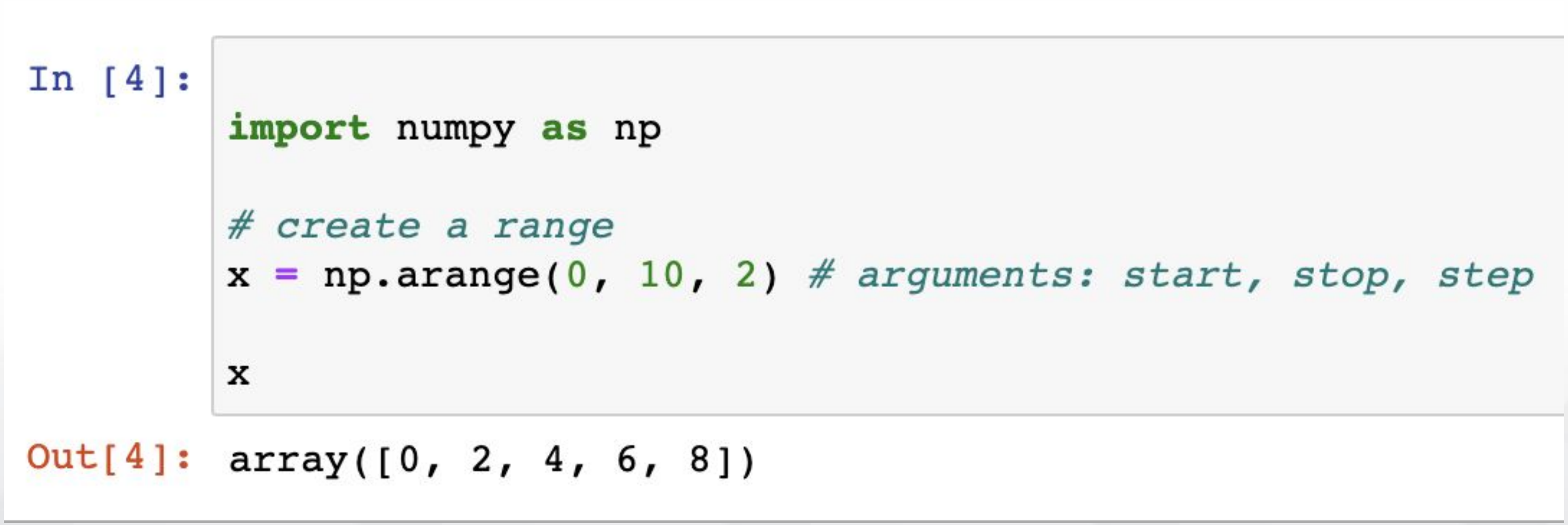


```
In [4]:

import numpy as np

# create a range
x = np.arange(0, 10, 2) # arguments: start, stop, step

x
```

NumPy

Using array-generating functions

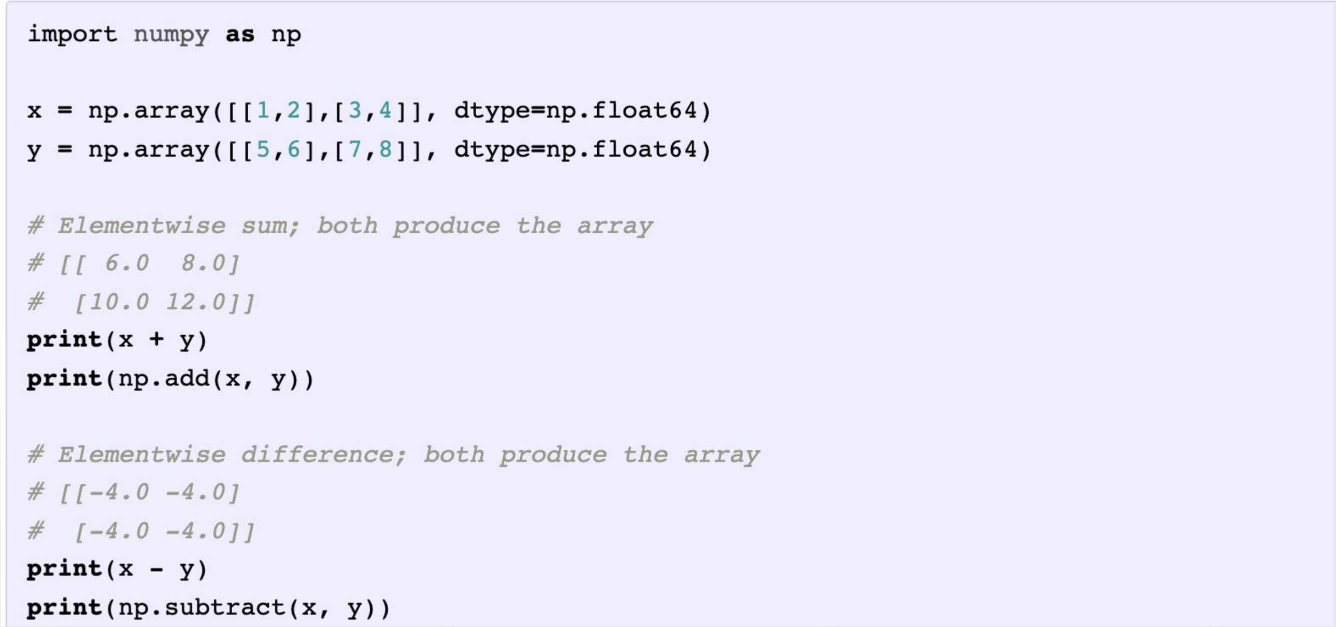
On Jupyter, what is the output of script given below?



```
In [4]:
         import numpy as np

         # create a range
         x = np.arange(0, 10, 2) # arguments: start, stop, step

         x

Out[4]:  array([0, 2, 4, 6, 8])
```

NumPy

Mathematical functions: Basic mathematical functions operate elementwise on arrays, and are available both as operator overloads and as functions in the numpy module:

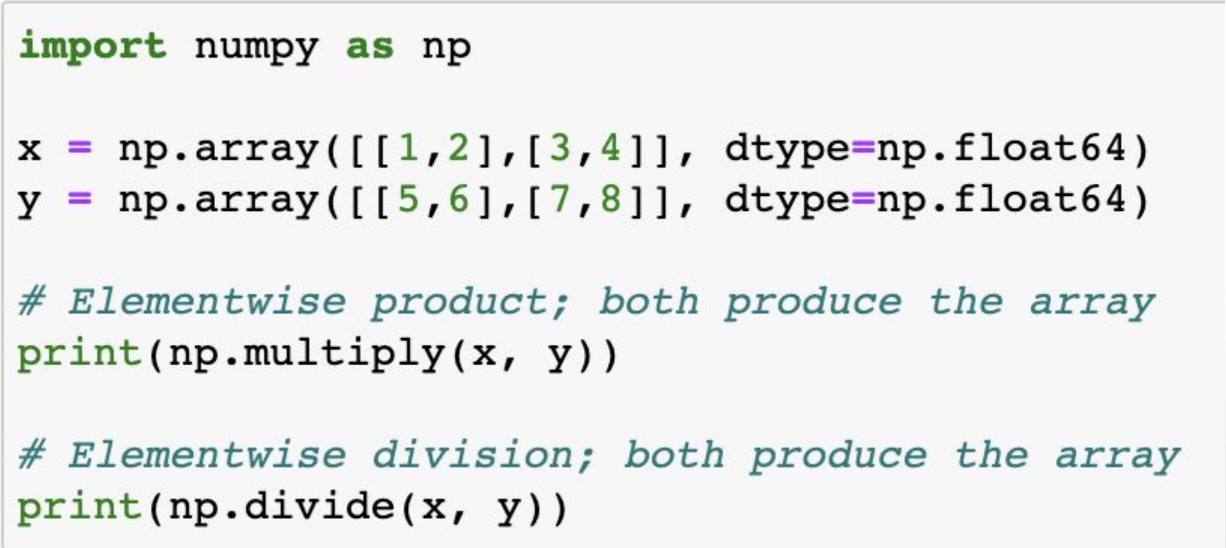


```
import numpy as np

x = np.array([[1,2],[3,4]], dtype=np.float64)
y = np.array([[5,6],[7,8]], dtype=np.float64)

# Elementwise sum; both produce the array
# [[ 6.0  8.0]
#  [10.0 12.0]]
print(x + y)
print(np.add(x, y))

# Elementwise difference; both produce the array
# [[-4.0 -4.0]
#  [-4.0 -4.0]]
print(x - y)
print(np.subtract(x, y))
```

NumPy

Multiply, divide arguments element-wise

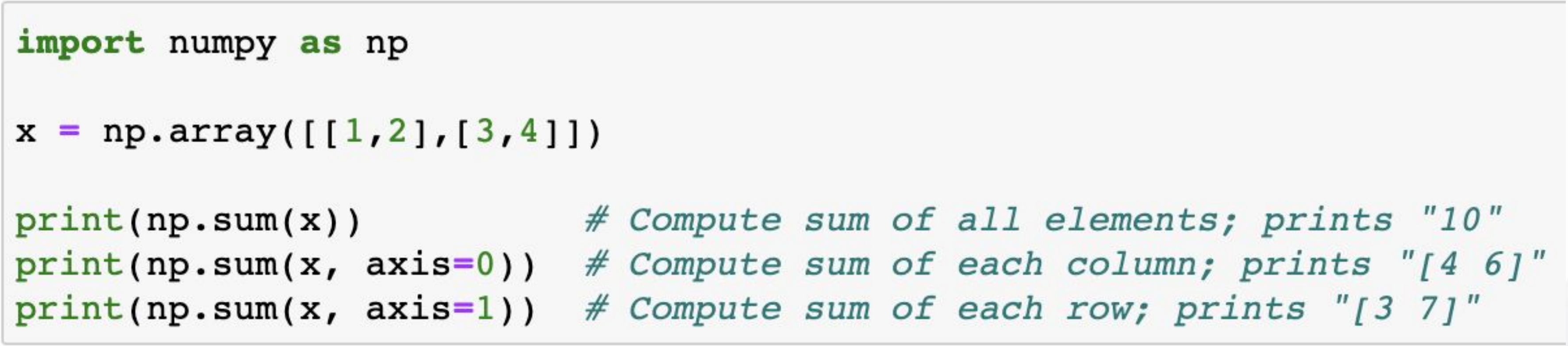
Class practice: What is the output of script given below?



```
import numpy as np

x = np.array([[1,2],[3,4]], dtype=np.float64)
y = np.array([[5,6],[7,8]], dtype=np.float64)

# Elementwise product; both produce the array
print(np.multiply(x, y))

# Elementwise division; both produce the array
print(np.divide(x, y))
```

NumPy

NumPy provides many useful functions for performing computations on arrays; one of the most useful is ***sum***:

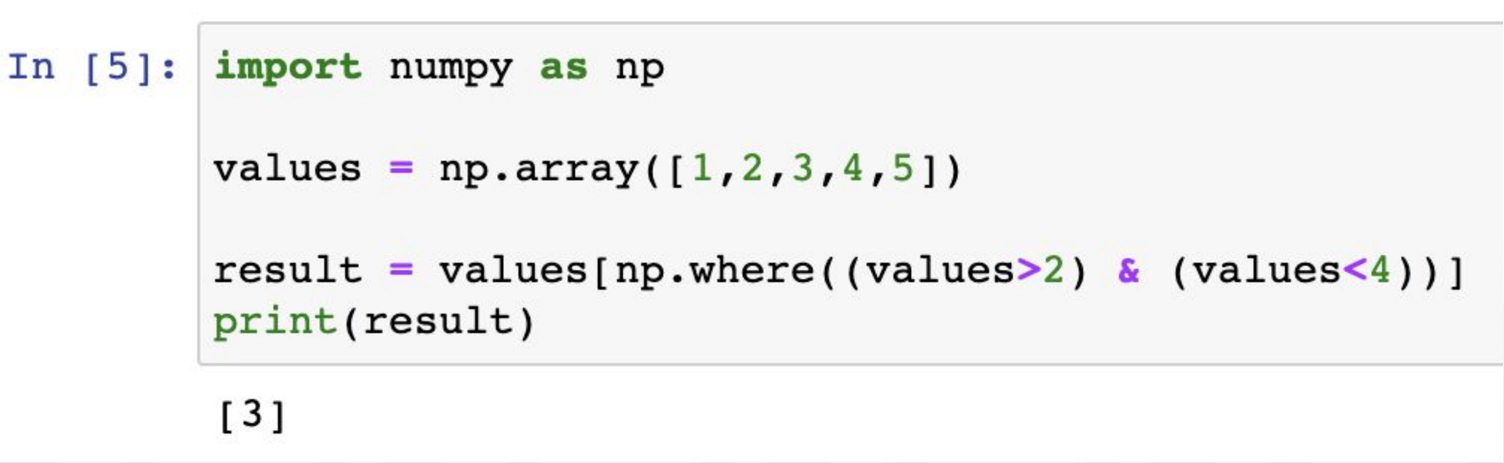


```
import numpy as np

x = np.array([[1,2],[3,4]])

print(np.sum(x))          # Compute sum of all elements; prints "10"
print(np.sum(x, axis=0))  # Compute sum of each column; prints "[4 6]"
print(np.sum(x, axis=1))  # Compute sum of each row; prints "[3 7]"
```

NumPy

Arrays in conditions

The ***numpy.where()*** function is used to select some elements from an array after applying a specified condition.



In [5]:

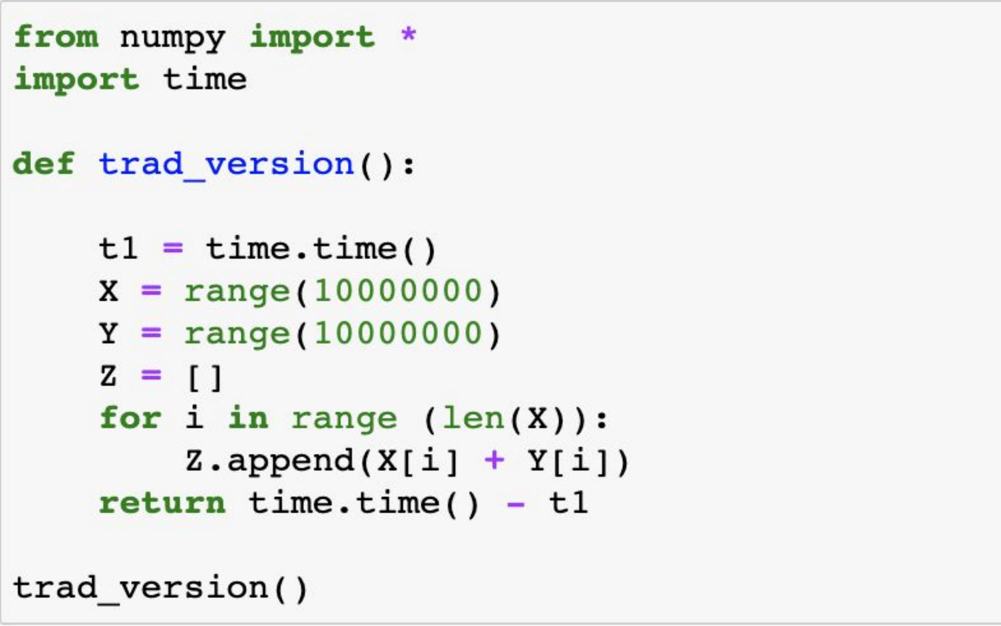
```
import numpy as np

values = np.array([1,2,3,4,5])

result = values[np.where((values>2) & (values<4))]
print(result)
```

```
[3]
```

NumPy

Class practice: python list vs numpy array performance

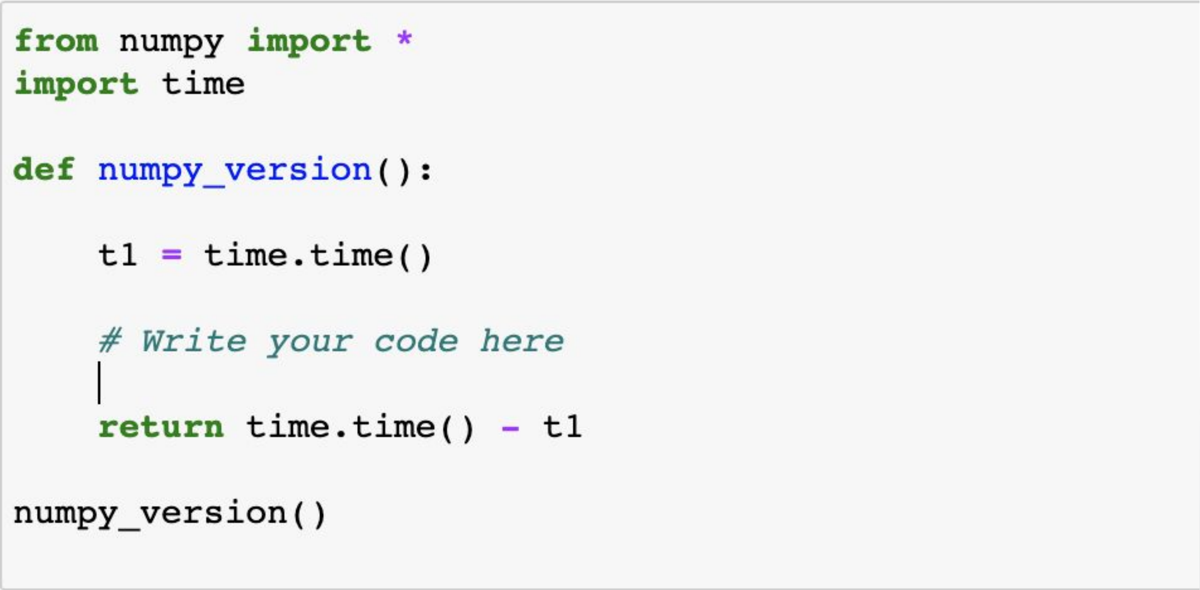
<u>Step 1</u>: Run the following traditional Python script code



```
from numpy import *
import time

def trad_version():

    t1 = time.time()
    X = range(10000000)
    Y = range(10000000)
    Z = []
    for i in range (len(X)):
        Z.append(X[i] + Y[i])
    return time.time() - t1

trad_version()
```

NumPy

Step 2: Complete and perform the task above using NumPy and compare the runtimes of the two programs.



```
from numpy import *
import time

def numpy_version():

    t1 = time.time()

    # Write your code here

    return time.time() - t1

numpy_version()
```

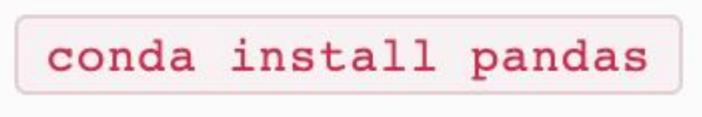
Introduction to Python Pandas for Data Analytics

- Pandas is an open source, BSD-licensed library.
- High-performance, easy-to-use data structures and data analysis tools.
- Built for the Python programming language.

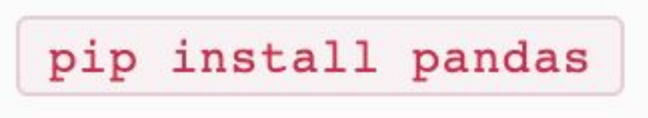
Introduction to Python Pandas for Data Analytics

Install Pandas

Open up your terminal program or command line and install it using either of the following commands:

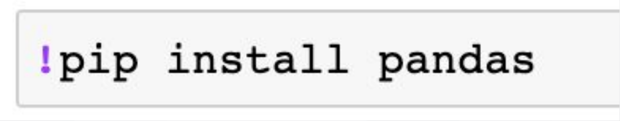


```
conda install pandas
```



```
pip install pandas
```

Alternatively, if you're currently viewing this article in a Jupyter notebook you can run this cell:

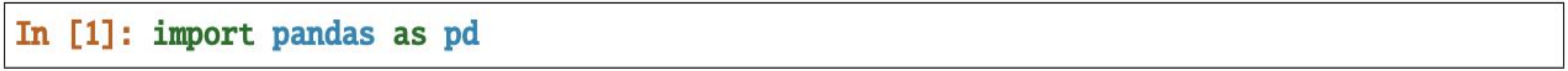


```
!pip install pandas
```

Introduction to Python Pandas for Data Analytics

Create a dataframe

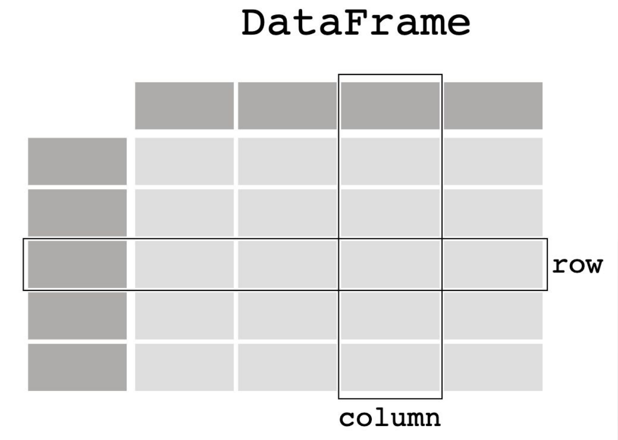
To load the pandas package and start working with it, import the package.



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

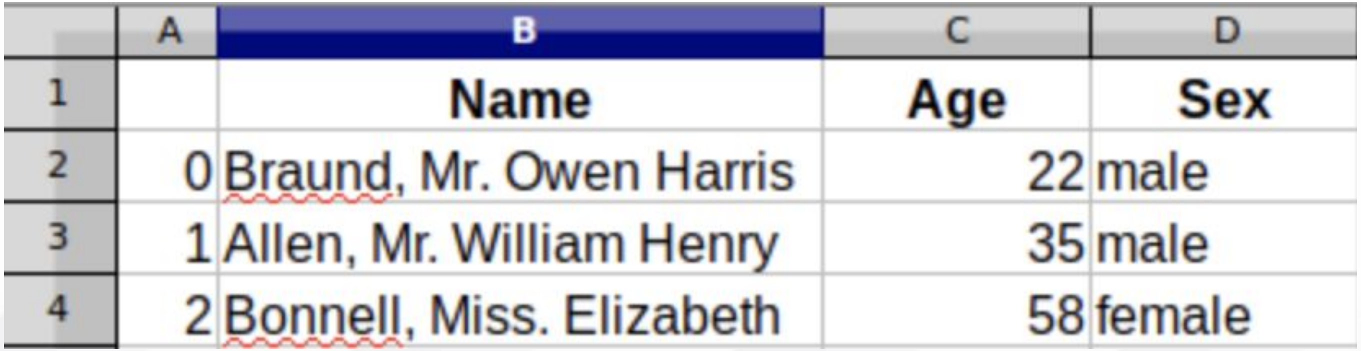
Pandas data table representation

A DataFrame is a 2-dimensional data structure that can store data of different types (including characters, integers, floating point values, categorical data and more) in columns.



Introduction to Python Pandas for Data Analytics

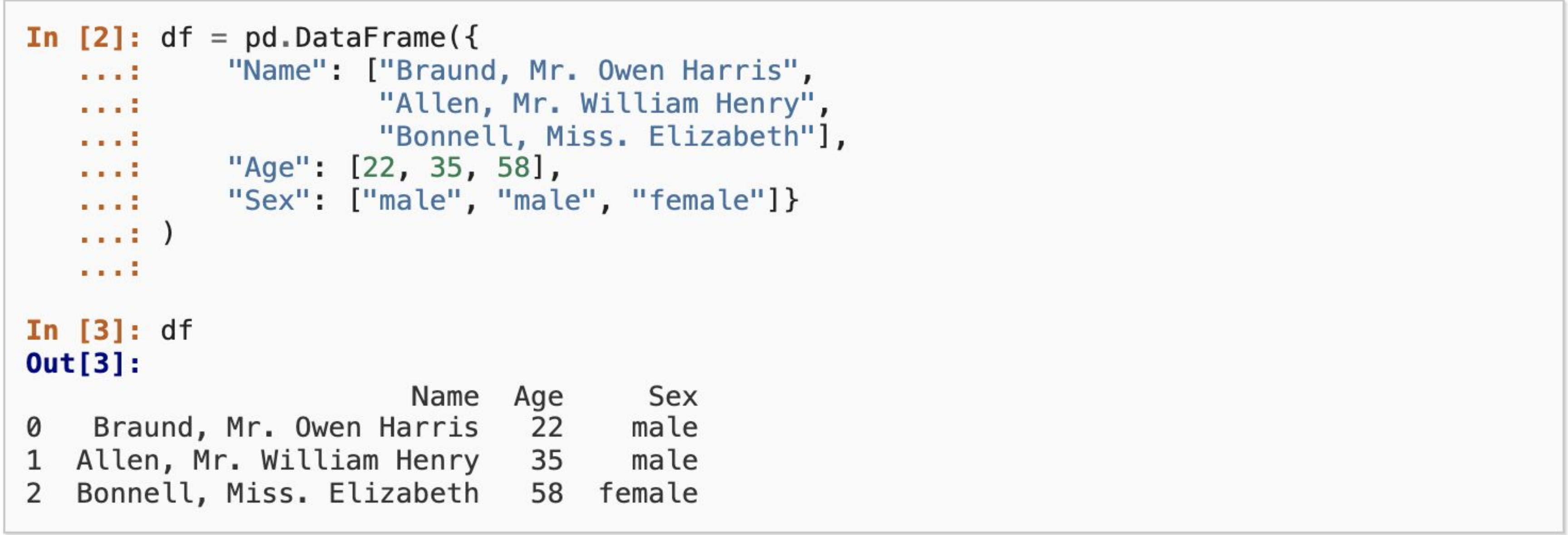
How to create a dataframe to store the spreadsheet below:



	A	B	C	D
1		Name	Age	Sex
2	0	Braund, Mr. Owen Harris	22	male
3	1	Allen, Mr. William Henry	35	male
4	2	Bonnell, Miss. Elizabeth	58	female

Introduction to Python Pandas for Data Analytics

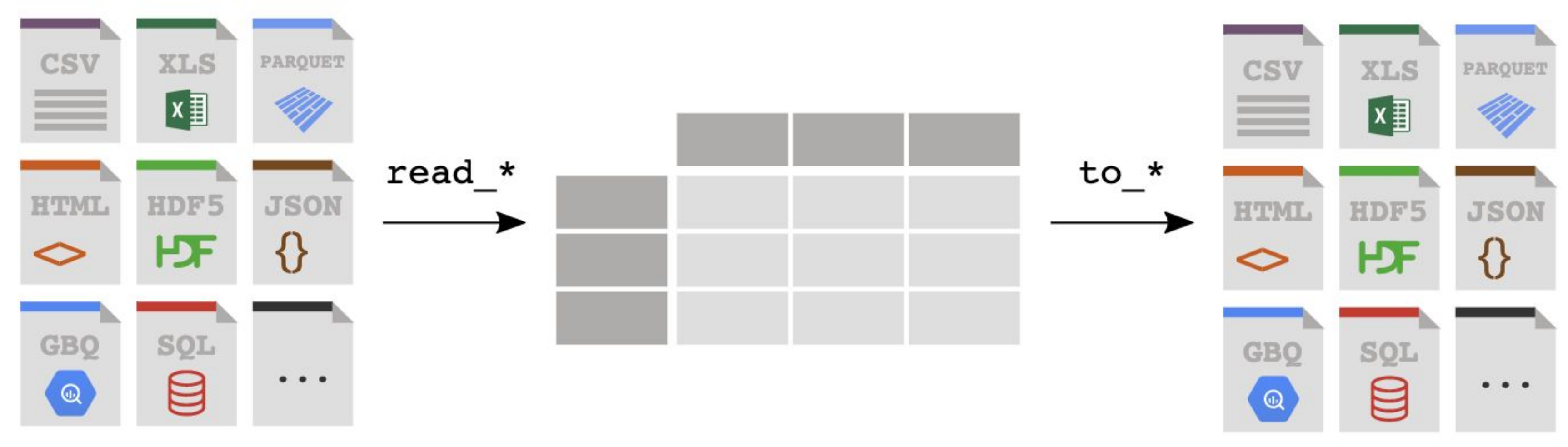
To manually store data in a table, create a DataFrame. When using a Python dictionary of lists, the dictionary keys will be used as column headers and the values in each list as columns of the DataFrame.



```
In [2]: df = pd.DataFrame({
   ...:     "Name": ["Braund, Mr. Owen Harris",
   ...:              "Allen, Mr. William Henry",
   ...:              "Bonnell, Miss. Elizabeth"],
   ...:     "Age": [22, 35, 58],
   ...:     "Sex": ["male", "male", "female"]}
   ...: )
   ...:

In [3]: df
Out[3]:
                       Name  Age     Sex
0   Braund, Mr. Owen Harris   22    male
1  Allen, Mr. William Henry   35    male
2  Bonnell, Miss. Elizabeth   58  female
```

Introduction to Python Pandas for Data Analytics

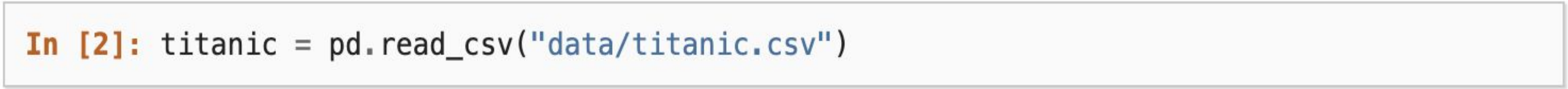
Read and write tabular data using Pandas



Introduction to Python Pandas for Data Analytics

Read and write tabular data using Pandas

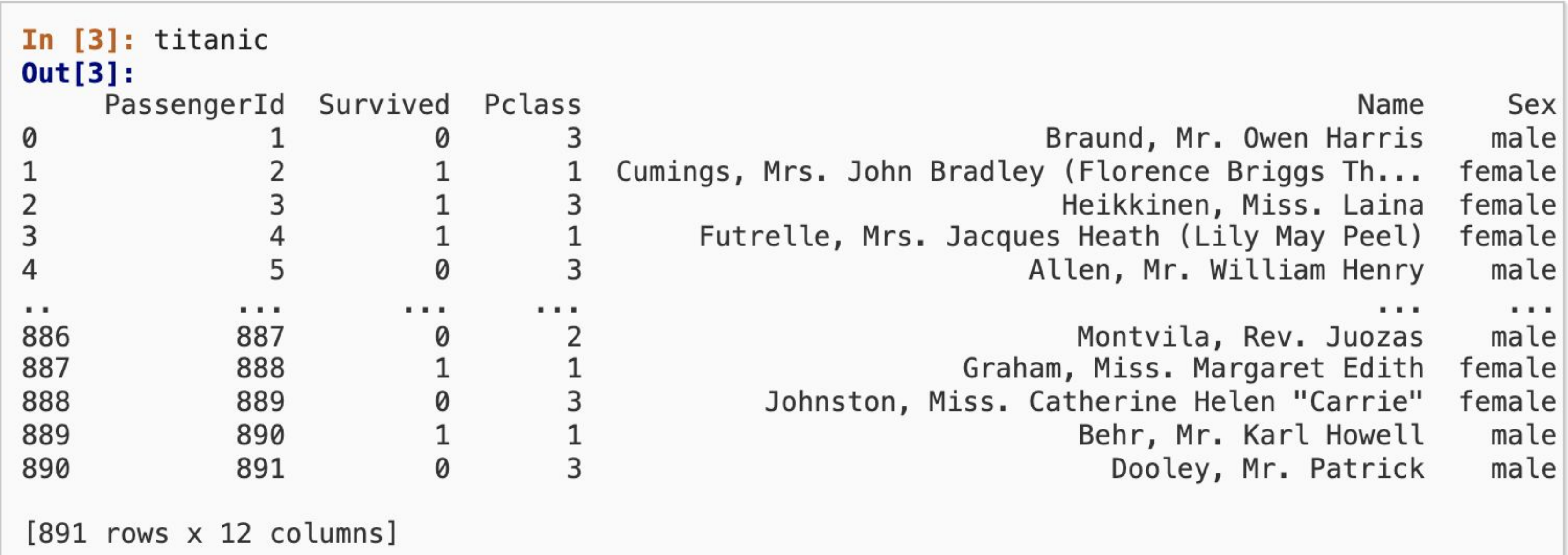
E.g., I want to analyze the Titanic passenger data, available as a CSV file.



```
In [2]: titanic = pd.read_csv("data/titanic.csv")
```

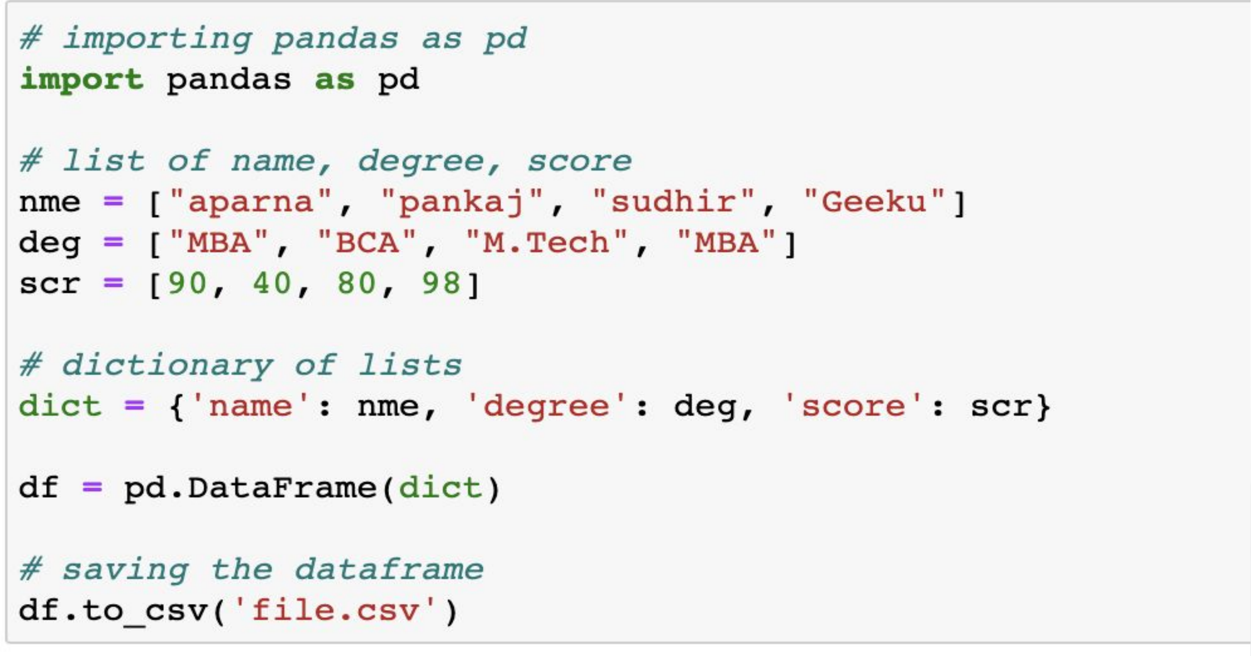
Introduction to Python Pandas for Data Analytics

Read and write tabular data using Pandas



```
In [3]: titanic
Out[3]:
     PassengerId  Survived  Pclass                                               Name     Sex
0              1         0       3                            Braund, Mr. Owen Harris    male
1              2         1       1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female
2              3         1       3                             Heikkinen, Miss. Laina  female
3              4         1       1       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female
4              5         0       3                           Allen, Mr. William Henry    male
..           ...       ...     ...                                                ...     ...
886          887         0       2                              Montvila, Rev. Juozas    male
887          888         1       1                       Graham, Miss. Margaret Edith  female
888          889         0       3           Johnston, Miss. Catherine Helen "Carrie"  female
889          890         1       1                              Behr, Mr. Karl Howell    male
890          891         0       3                                Dooley, Mr. Patrick    male

[891 rows x 12 columns]
```

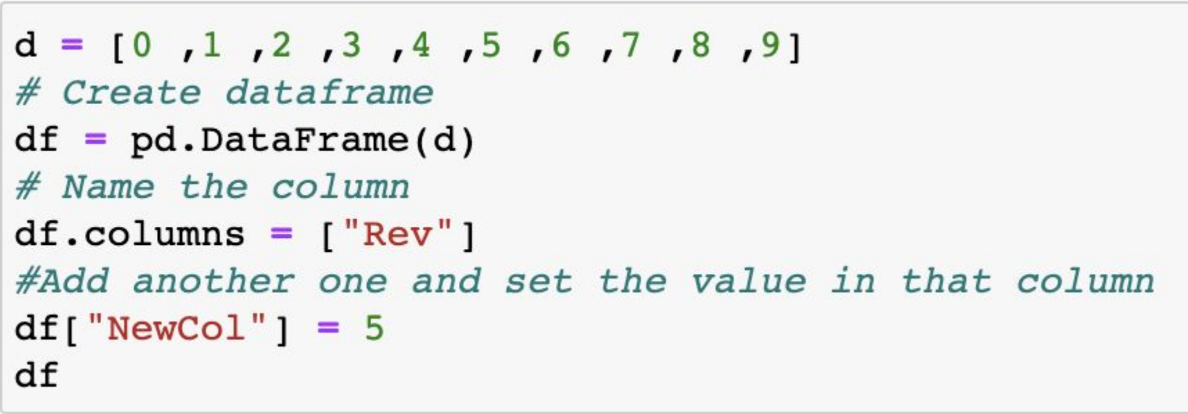
Saving a Pandas Dataframe as a CSV



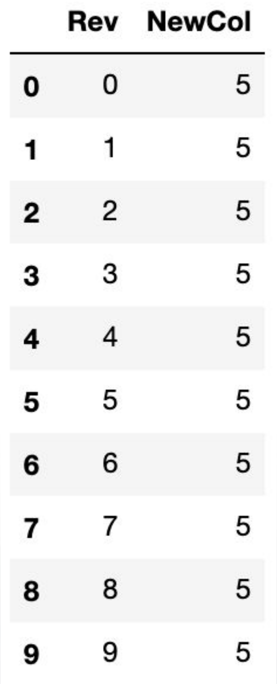
```
# importing pandas as pd
import pandas as pd

# list of name, degree, score
nme = ["aparna", "pankaj", "sudhir", "Geeku"]
deg = ["MBA", "BCA", "M.Tech", "MBA"]
scr = [90, 40, 80, 98]

# dictionary of lists
dict = {'name': nme, 'degree': deg, 'score': scr}

df = pd.DataFrame(dict)

# saving the dataframe
df.to_csv('file.csv')
```

Introduction to Python Pandas for Data Analytics

Working on the data with Pandas



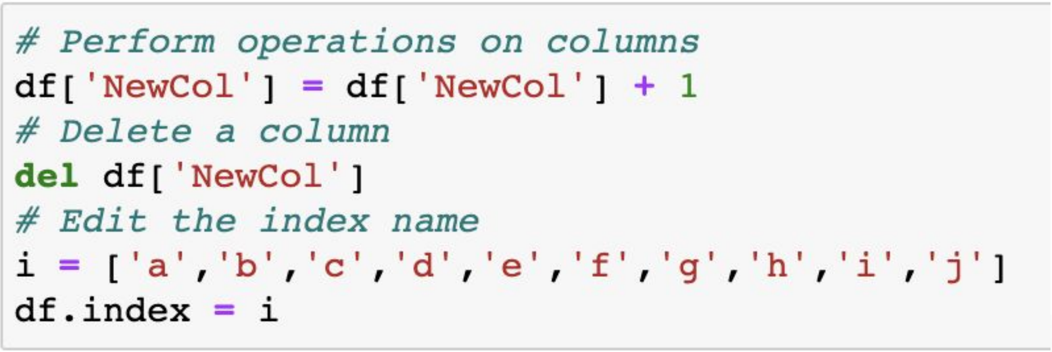
```
d = [0 ,1 ,2 ,3 ,4 ,5 ,6 ,7 ,8 ,9]
# Create dataframe
df = pd.DataFrame(d)
# Name the column
df.columns = ["Rev"]
#Add another one and set the value in that column
df["NewCol"] = 5
df
```



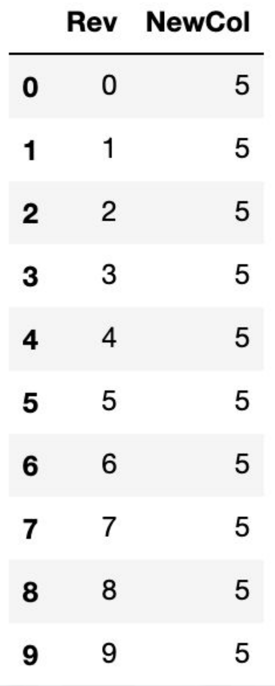
	Rev	NewCol
0	0	5
1	1	5
2	2	5
3	3	5
4	4	5
5	5	5
6	6	5
7	7	5
8	8	5
9	9	5

Introduction to Python Pandas for Data Analytics

Working on the data with Pandas

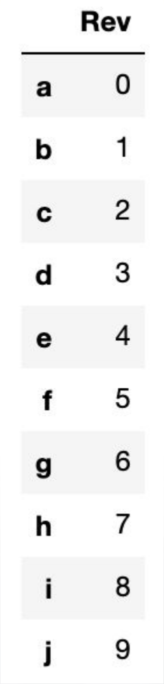


```
# Perform operations on columns
df['NewCol'] = df['NewCol'] + 1
# Delete a column
del df['NewCol']
# Edit the index name
i = ['a','b','c','d','e','f','g','h','i','j']
df.index = i
```



	Rev	NewCol
0	0	5
1	1	5
2	2	5
3	3	5
4	4	5
5	5	5
6	6	5
7	7	5
8	8	5
9	9	5

Before



	Rev
a	0
b	1
c	2
d	3
e	4
f	5
g	6
h	7
i	8
j	9

After

Introduction to Python Pandas for Data Analytics

Class practice: In this exercise, we are using Automobile Dataset (from here) for data analysis. This Dataset has different characteristics of an auto such as body-style, wheel-base, engine-type, price, mileage, horsepower, etc.

(a) From the given dataset print the first and last five rows.
(b) Print All Toyota Cars details.
(c) Print most expensive car’s company name and price.

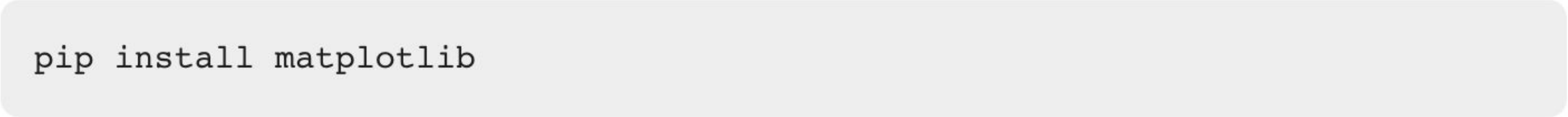
Plotting with Matplotlib

- Matplotlib is plotting library, used for generating 2D and 3D scientific plots.
- Support for LaTeX.
- Many output file formats including PNG, PDF, SVG, EPS.

Plotting with Matplotlib

Install Matplotlib

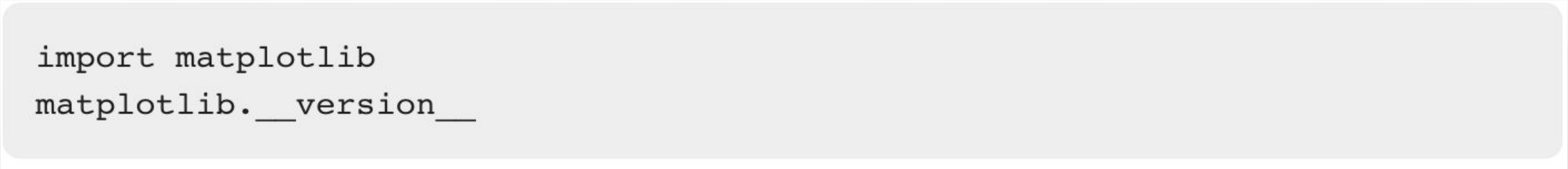
Matplotlib can be installed using pip. The following command is run in the command prompt to install Matplotlib.



```
pip install matplotlib
```

Import Matplotlib

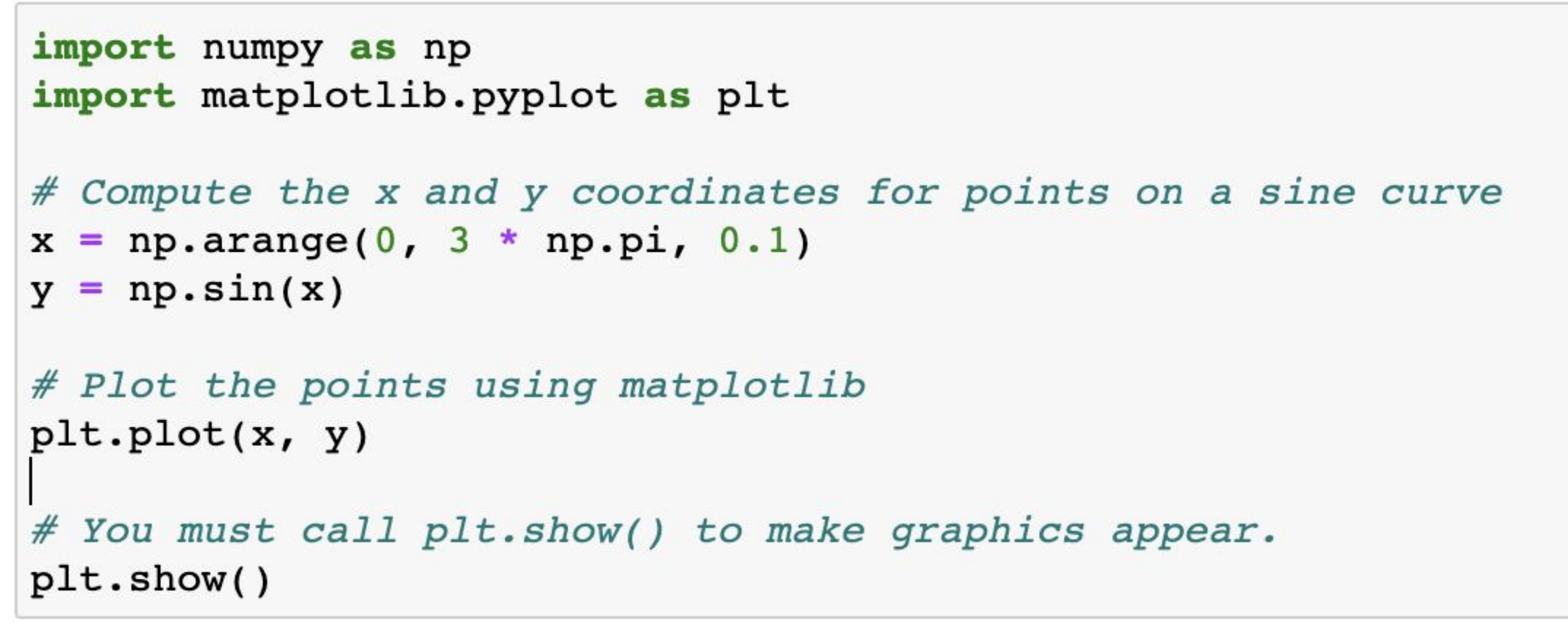
To verify that matplotlib is successfully installed on your system, execute the following command in the command prompt.



```
import matplotlib
matplotlib.__version__
```

Plotting with Matplotlib

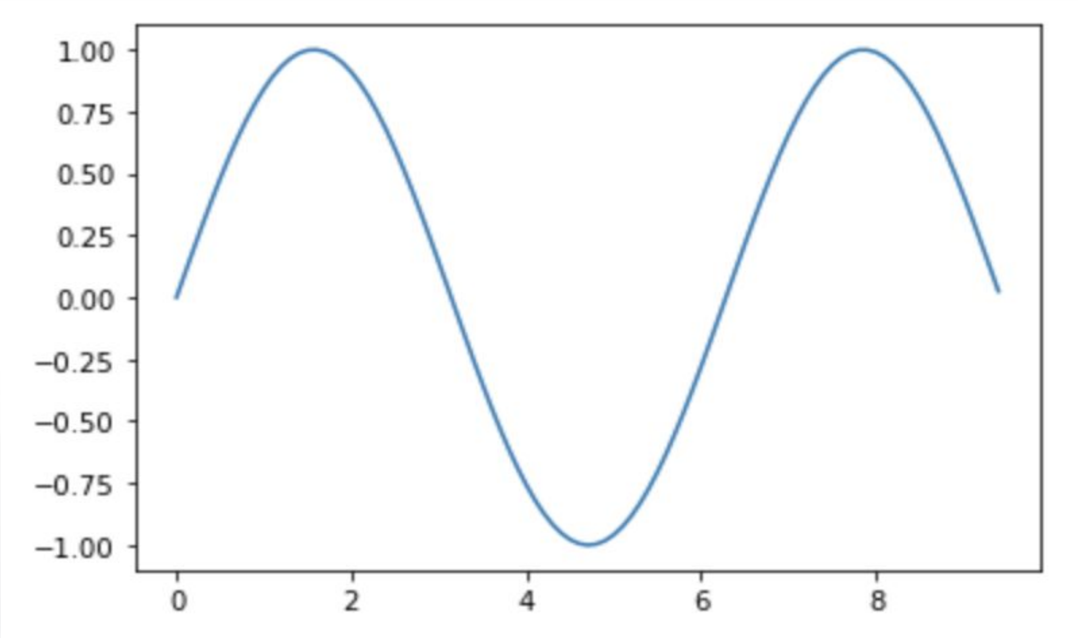
Generate and plot data points using .pyplot module



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

# Compute the x and y coordinates for points on a sine curve
x = np.arange(0, 3 * np.pi, 0.1)
y = np.sin(x)

# Plot the points using matplotlib
plt.plot(x, y)

# You must call plt.show() to make graphics appear.
plt.show()
```

Plotting with Matplotlib

Running this code produces the following plot:



Matplotlib: Class practice

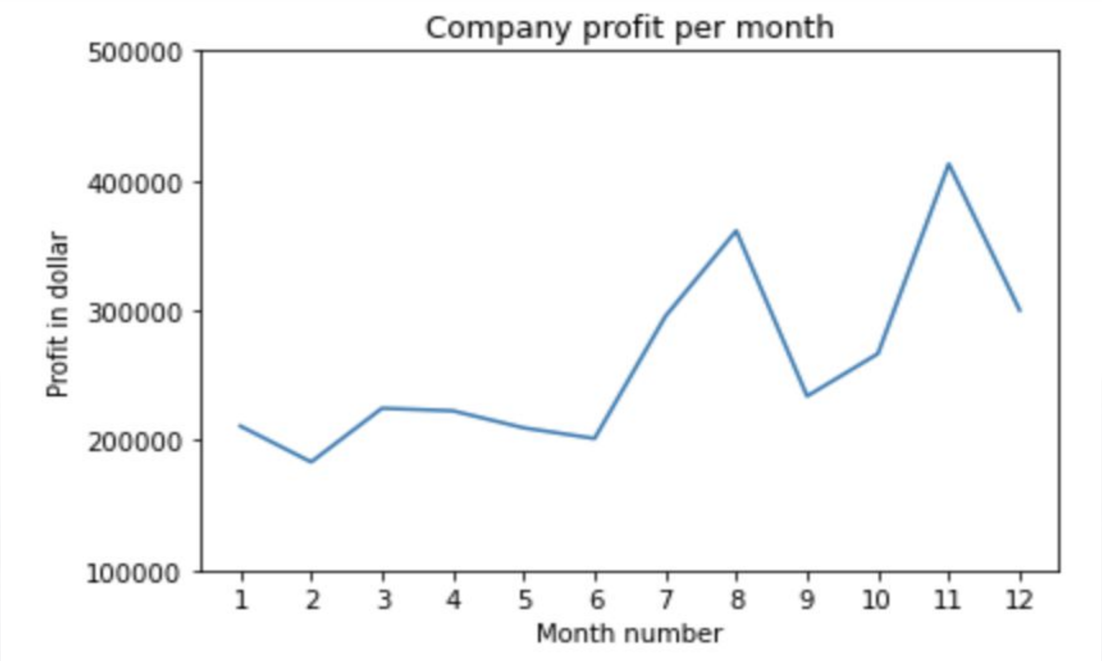
Read total profit of all months and show it using a line plot.

Use the CSV file “company_sales_data.csv” from here for this exercise. Read this file using Pandas or NumPy or using in-built matplotlib function. Total profit data provided for each month. Generated line plot must include the following properties:

- X label name = Month Number
- Y label name = Profit in dollar

Matplotlib: Class practice

The line plot graph should look like this.



References

https://numpy.org/

https://pandas.pydata.org/

https://matplotlib.org/

Thank you for attending !