ISOM 2600 Business Analytics

TOPIC 1: LIST, ARRAY AND GRAPHING

XUHU WAN

HKUST

JANUARY 15, 2022

Contents

1. Why Python ?
2. Review of basic data types
3. List and List comprehension
4. Numpy and scipy
5. Matplotlib and Statistical Graphing

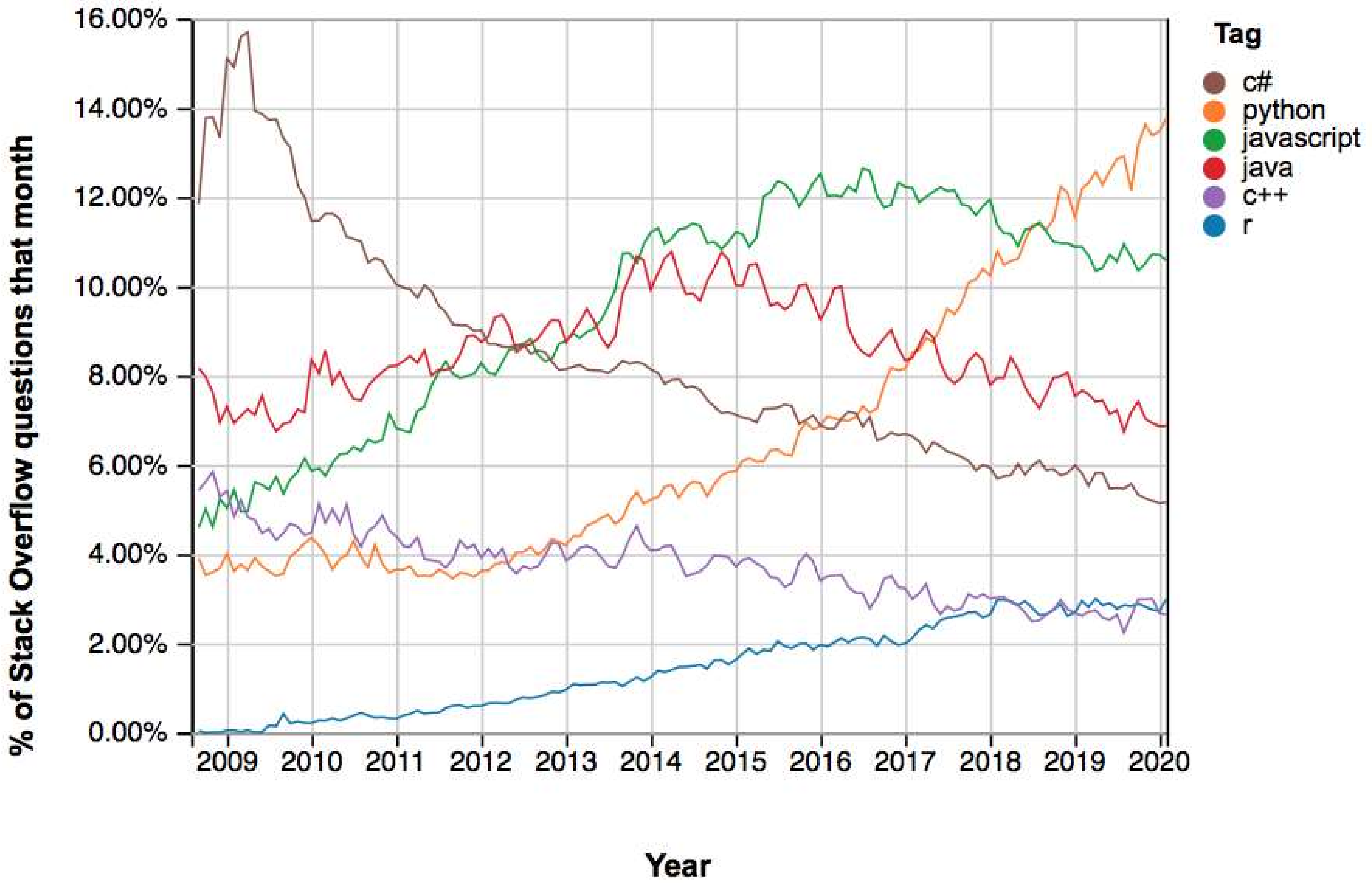
Why Python

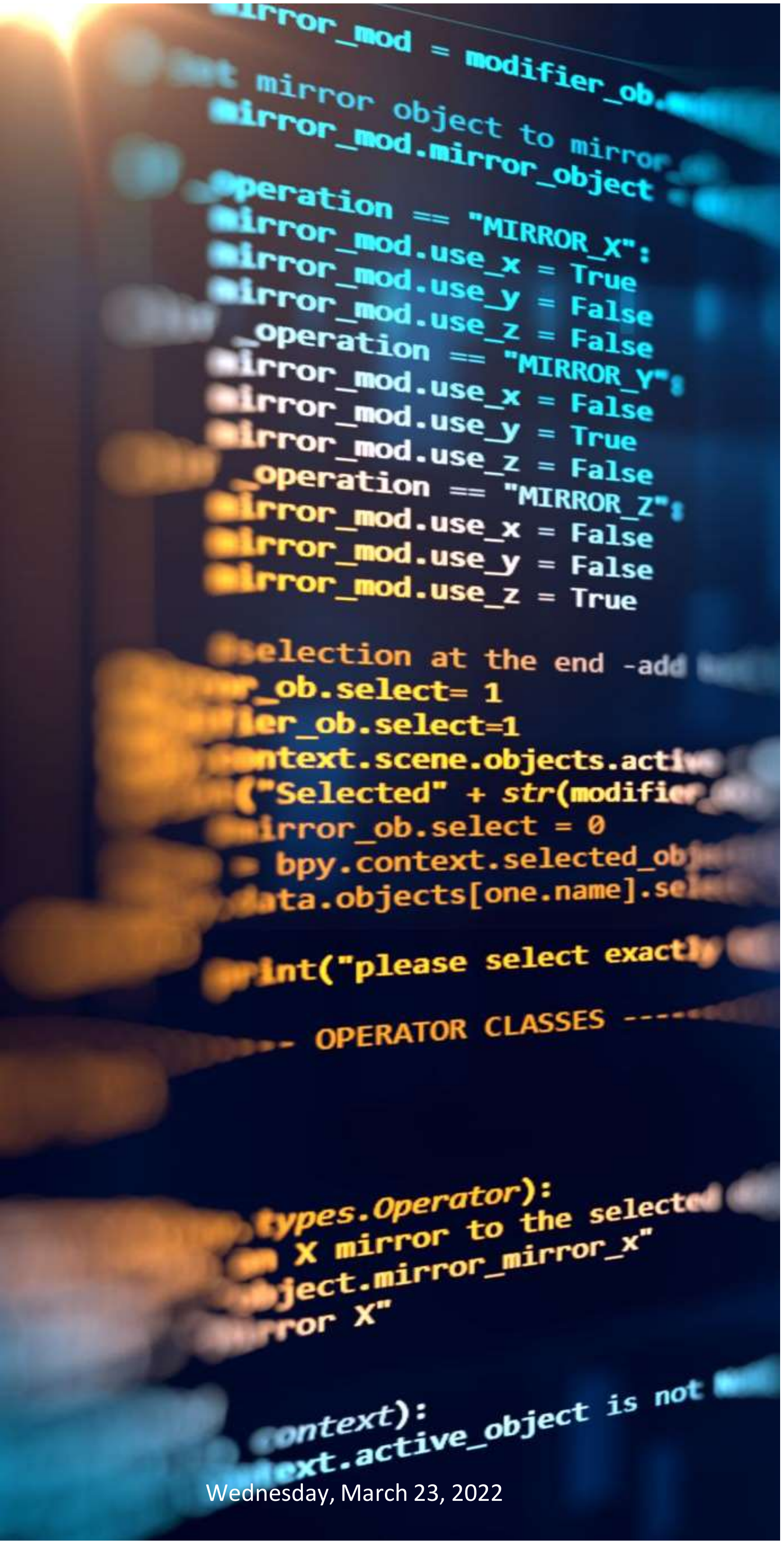
Learning Business Analytics with Real Data

1. Business analytics itself is a combination of statistics, and computer science and domain knowledge.
2. To gain insights of statistical models or theory, it is necessary to implement and compare different theoretical models in the real data.
3. In getting a broader perspective, we should not only know how to implement the models but understand how they connect and are related to the deeper logic behind them.

.

Which Programming Language?





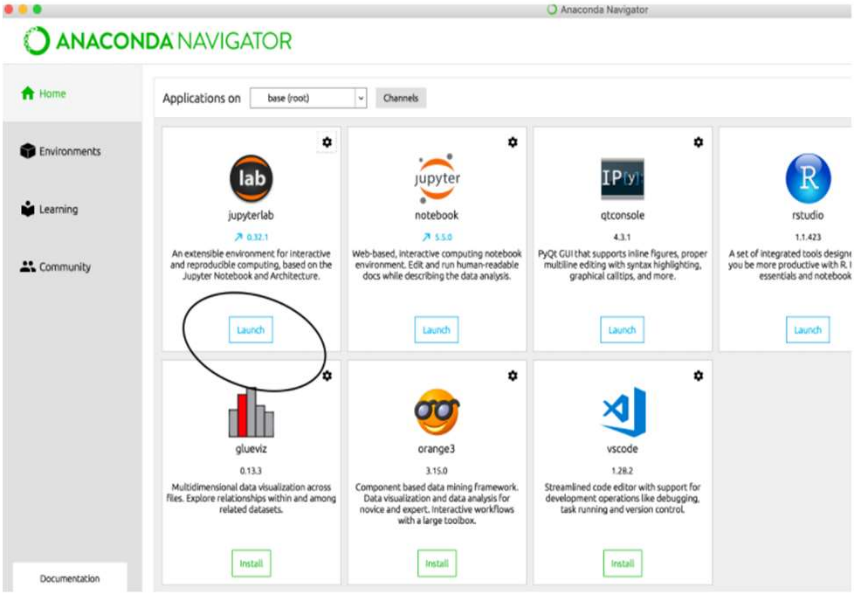
```
irror_mod = modifier_ob.
 mirror object to mirror
irror_mod.mirror_object
peration == "MIRROR_X":
irror_mod.use_x = True
irror_mod.use_y = False
irror_mod.use_z = False
 _operation == "MIRROR_Y":
irror_mod.use_x = False
irror_mod.use_y = True
irror_mod.use_z = False
 _operation == "MIRROR_Z":
irror_mod.use_x = False
irror_mod.use_y = False
irror_mod.use_z = True

election at the end -add
 _ob.select= 1
er_ob.select=1
ntext.scene.objects.acti
("Selected" + str(modifie
irror_ob.select = 0
 bpy.context.selected_ob
ata.objects[one.name].se

int("please select exactl

-- OPERATOR CLASSES ----

types.Operator):
 X mirror to the selected
ject.mirror_mirror_x"
rror X"

ontext):
xt.active_object is not
```

Use of Python

Jupyter Notebook/Lab

- In this course, instead of running python or ipython in the command or editor (pycharm, spyder), we will only use jupyter notebook/jupyter lab.
- This will open another Python interface in a web browser. it does not actually need any Internet connection to run.

Fire lab/notebook in navigator



jupyterlab
↗ 0.32.1
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

notebook
↗ 5.5.0
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

qtconsole
4.3.1
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

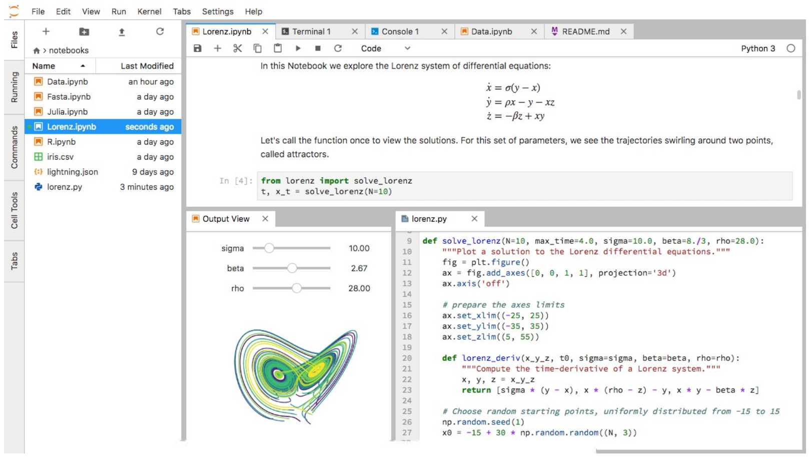
rstudio
1.1.423
A set of integrated tools designe you be more productive with R. I essentials and notebook

glueviz
0.13.3
Multidimensional data visualization across files. Explore relationships within and among related datasets.

orange3
3.15.0
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

vscode
1.28.2
Streamlined code editor with support for development operations like debugging, task running and version control.

lab



Name	Last Modified
Data.ipynb	an hour ago
Fasta.ipynb	a day ago
Julia.ipynb	a day ago
Lorenz.ipynb	seconds ago
R.ipynb	a day ago
iris.csv	a day ago
lightning.json	9 days ago
lorenz.py	3 minutes ago

In this Notebook we explore the Lorenz system of differential equations:

$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = \rho x - y - xz$$
$$\dot{z} = -\beta z + xy$$

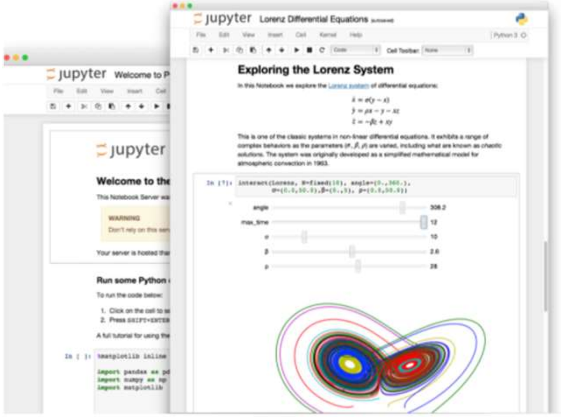
Let's call the function once to view the solutions. For this set of parameters, we see the trajectories swirling around two points, called attractors.

```
In [4]: from lorenz import solve_lorenz
        t, x_t = solve_lorenz(N=10)
```

sigma 10.00
beta 2.67
rho 28.00

```
def solve_lorenz(N=10, max_time=4.0, sigma=10.0, beta=8./3, rho=28.0):
    """Plot a solution to the Lorenz differential equations."""
    fig = plt.figure()
    ax = fig.add_axes([0, 0, 1, 1], projection='3d')
    ax.axis('off')

    # prepare the axes limits
    ax.set_xlim((-25, 25))
    ax.set_ylim((-35, 35))
    ax.set_zlim((5, 55))

    def lorenz_deriv(x_y_z, t0, sigma=sigma, beta=beta, rho=rho):
        """Compute the time-derivative of a Lorenz system."""
        x, y, z = x_y_z
        return [sigma * (y - x), x * (rho - z) - y, x * y - beta * z]

    # Choose random starting points, uniformly distributed from -15 to 15
    np.random.seed(1)
    x0 = -15 + 30 * np.random.random((N, 3))
```

notebook



Exploring the Lorenz System

In this Notebook we explore the Lorenz system of differential equations:

$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = \rho x - y - xz$$
$$\dot{z} = -\beta z + xy$$

This is one of the classic systems in non-linear differential equations. It exhibits a range of complex behaviors as the parameters (σ, β, ρ) are varied, including what are known as chaotic solutions. The system was originally developed as a simplified mathematical model for atmospheric convection in 1963.

```
In [7]: interact(Lorenz, N=fixed(10), angle=(0.,360.),
                 σ=(0.0,50.0),β=(0.,5), ρ=(0.0,50.0))
```

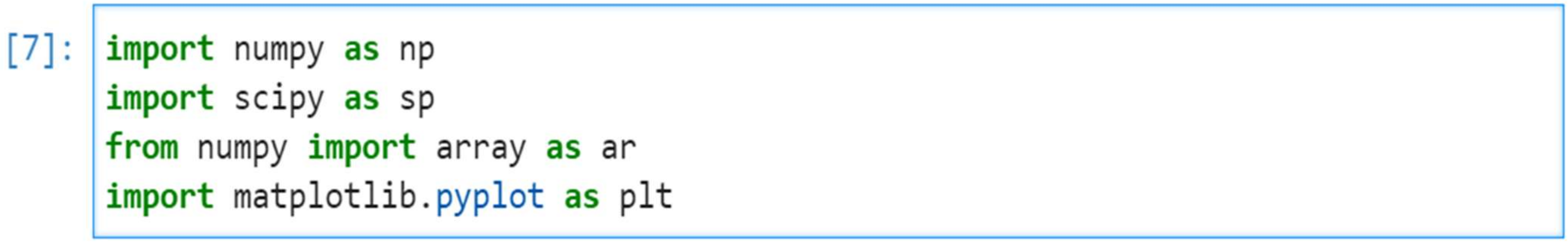
angle 308.2
max_time 12
σ 10
β 2.6
ρ 28

Install Packages and Import Libraries

Importing Libraries

Libraries provide additional functionality in an organized and packaged way. Basic python includes many functionality. But there are many methods and attributes we need to import from external library. There libraries are still python based but provide tools for many application.

- Numpy : array and matrix, random number generators
- Scipy: Numerical routines for optimization, linear algebra and statistics
- Matplotlib: a comprehensive library mainly for creating static visualization.

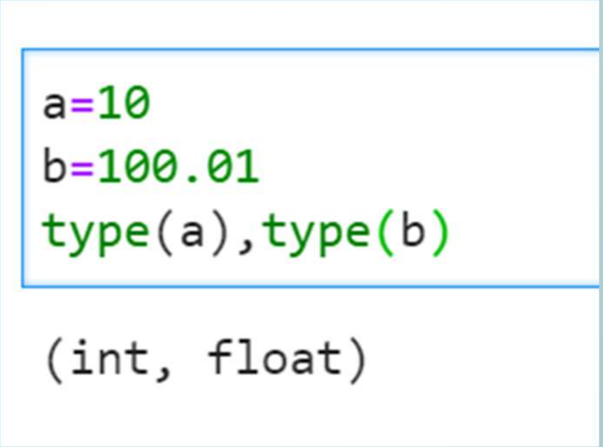


```
import numpy as np
import scipy as sp
from numpy import array as ar
import matplotlib.pyplot as plt
```

Quick Review of Int, float, bool and string

Basic Data Types in Python

CODE

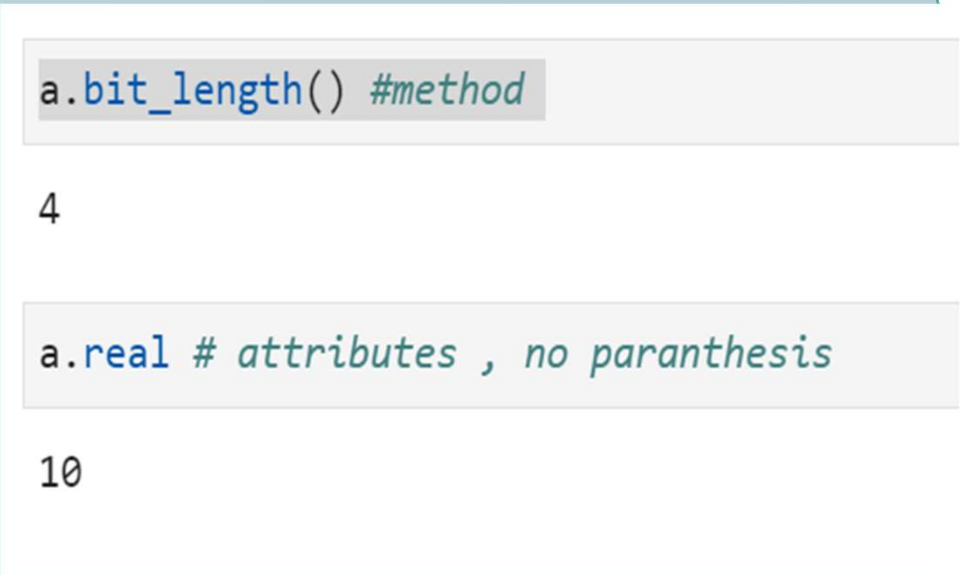


```
a=10
b=100.01
type(a),type(b)
```

```
(int, float)
```

DATA

- Data attributes
- Data Methods



```
a.bit_length() #method
```

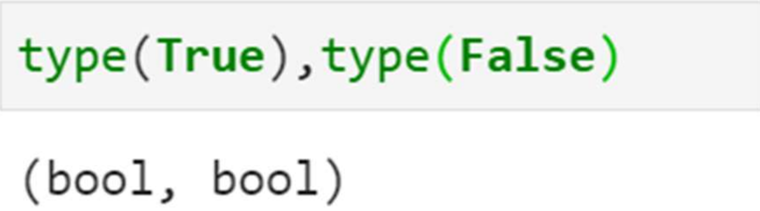
```
4
```

```
a.real # attributes , no paranthesis
```

```
10
```

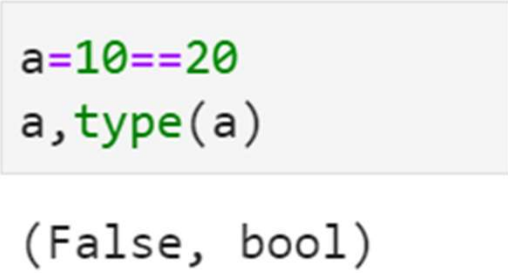
Bool

A Boolean value is either true or false. It is named after the British mathematician, George Boole— some rules for reasoning about and combining these values. This is the basis of all modern computer logic.



```
type(True),type(False)
```

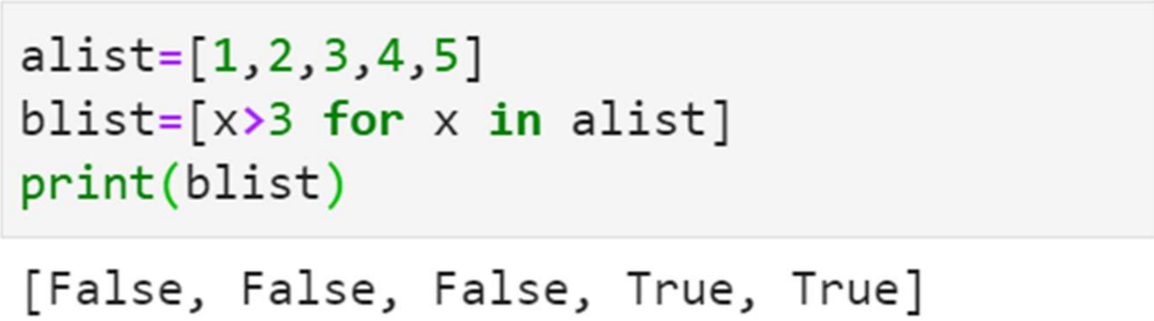
```
(bool, bool)
```



```
a=10==20
a,type(a)
```

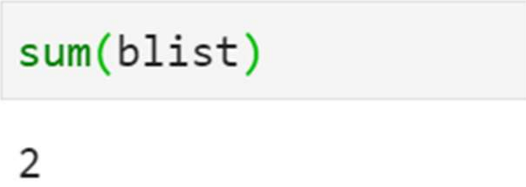
```
(False, bool)
```

In Python, the two Boolean values are True and False (the capitalization must be exactly as shown), and the Python type is bool. True is counted as 1 and Falses is counted as 0 .



```
alist=[1,2,3,4,5]
blist=[x>3 for x in alist]
print(blist)
```

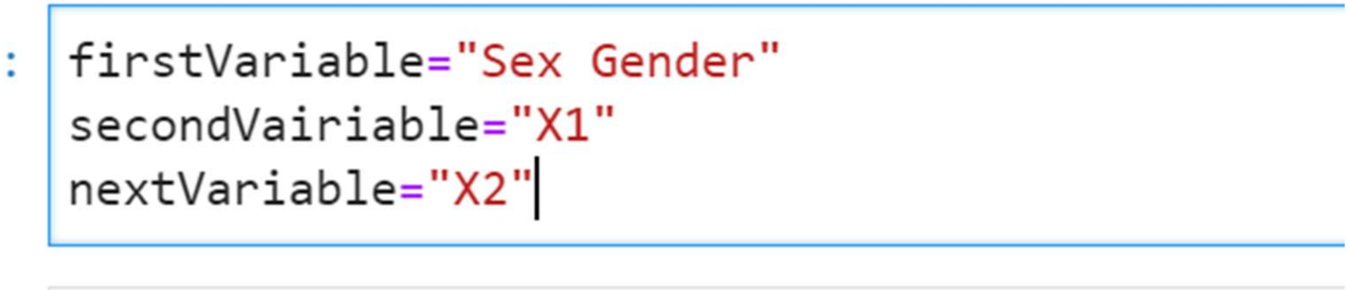
```
[False, False, False, True, True]
```



```
sum(blist)
```

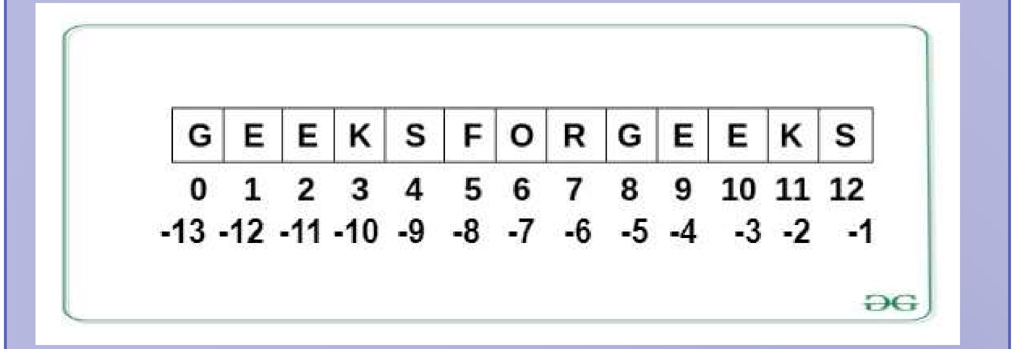
```
2
```

String

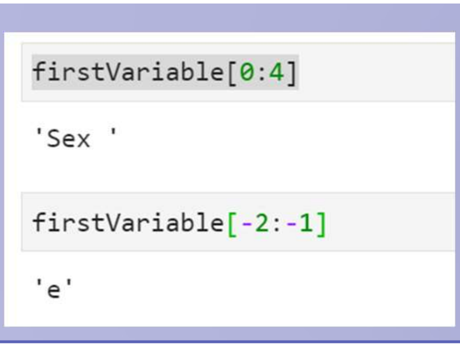


```
firstVariable="Sex Gender"
secondVairiable="X1"
nextVariable="X2"
```

- Index

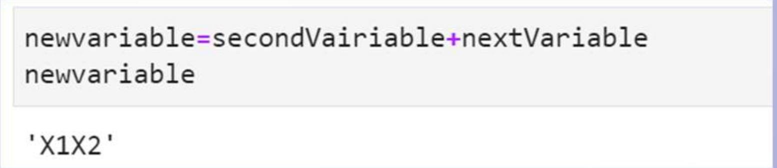


G	E	E	K	S	F	O	R	G	E	E	K	S
0	1	2	3	4	5	6	7	8	9	10	11	12
-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1



```
firstVariable[0:4]

'Sex '

firstVariable[-2:-1]

'e'
```

- Concatenate



```
newvariable=secondVairiable+nextVariable
newvariable

'X1X2'
```

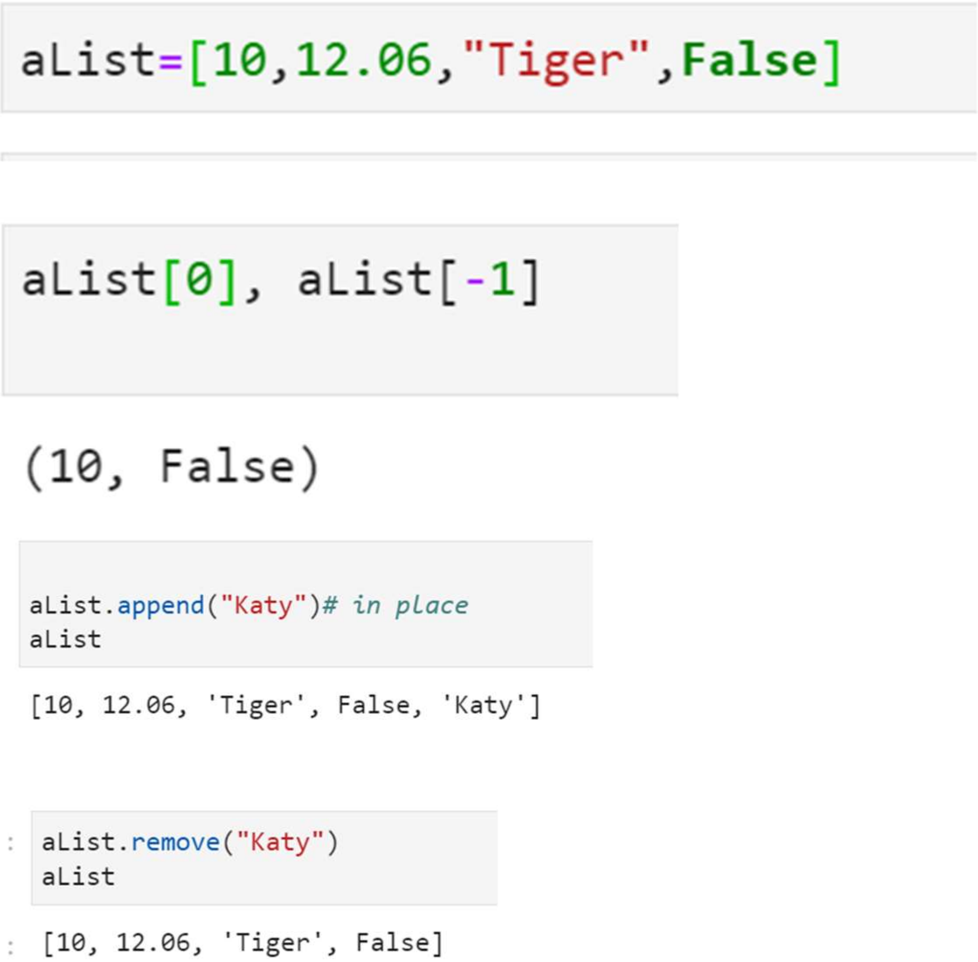
Practice:

1) Input Your last name AAA
2) Print Mr/Ms AAA

List

List: A universal data container

- Lists are used to store heterogeneous data, and are created with a pair of square brackets, [].
- Lists are one of 4 built-in data types in Python used to store collections of data, (Tuple, Set, and Dictionary)



```
aList=[10,12.06,"Tiger",False]
```

```
aList[0], aList[-1]
```

```
(10, False)
```

```
aList.append("Katy")# in place
aList
```

```
[10, 12.06, 'Tiger', False, 'Katy']
```

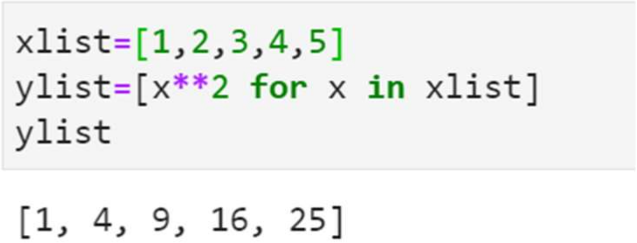
```
aList.remove("Katy")
aList
```

```
[10, 12.06, 'Tiger', False]
```

List Comprehension

List comprehension is often used when doing data preprocessing.
There are two different kinds of comprehension

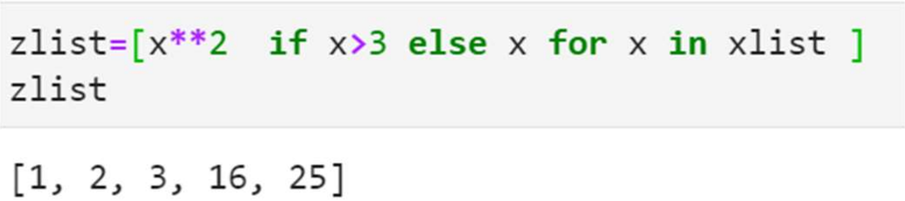
- ❖ Plain comprehension



```
xlist=[1,2,3,4,5]
ylist=[x**2 for x in xlist]
ylist
```

```
[1, 4, 9, 16, 25]
```

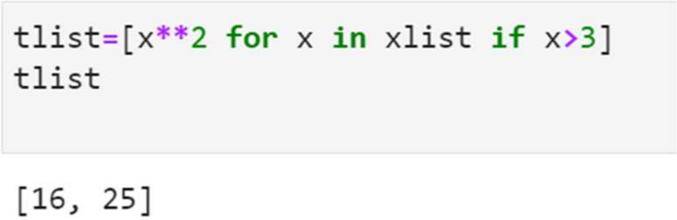
- ❖ Conditional comprehension



```
zlist=[x**2  if x>3 else x for x in xlist ]
zlist
```

```
[1, 2, 3, 16, 25]
```

- ❖ Filtered comprehension



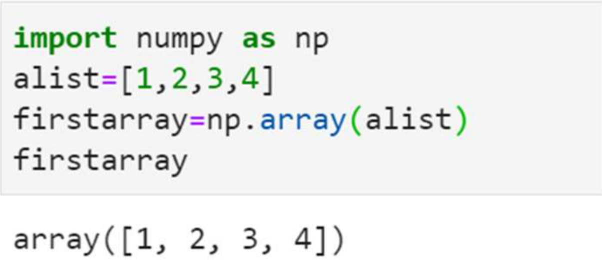
```
tlist=[x**2 for x in xlist if x>3]
tlist
```

```
[16, 25]
```

Numpy and Scipy

Numpy ndarray

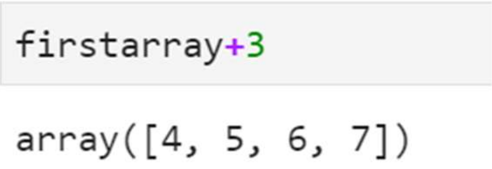
The numpy library gives Python the ability to work with matrices and arrays. It provides a high-performance scientific computation tools working with data 1D or 2D,3D arrays. For statistics, it also provides a lot of fast generators of random variables. It can be defined from the list



```
import numpy as np
alist=[1,2,3,4]
firstarray=np.array(alist)
firstarray
```

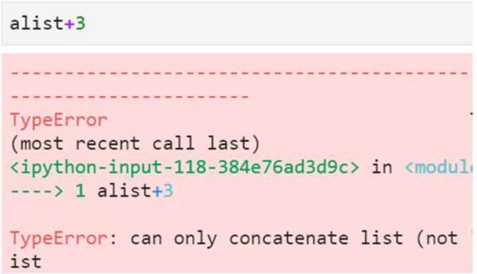
```
array([1, 2, 3, 4])
```

However it has its own convenience that the list does not have



```
firstarray+3
```

```
array([4, 5, 6, 7])
```

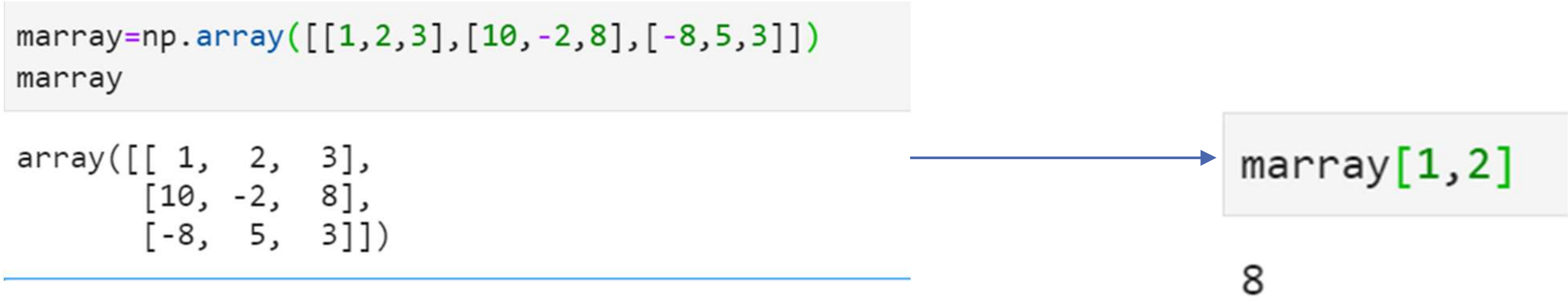


```
alist+3
```

```
-------------------------------------------
----------------------
TypeError
(most recent call last)
<ipython-input-118-384e76ad3d9c> in <modul
----> 1 alist+3

TypeError: can only concatenate list (not
ist
```

The numpy array can do element-wise computation

Multidimensional array



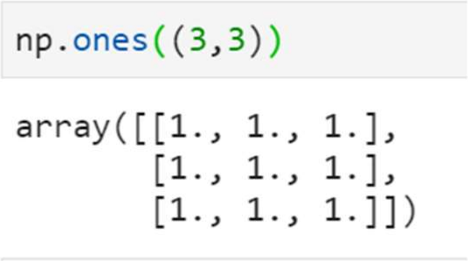
```
marray=np.array([[1,2,3],[10,-2,8],[-8,5,3]])
marray
```

```
array([[ 1,  2,  3],
       [10, -2,  8],
       [-8,  5,  3]])
```

```
marray[1,2]
```

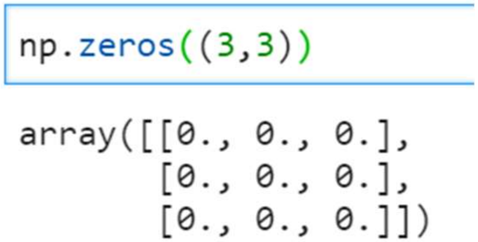
```
8
```

Built- in Function to Create Arrays



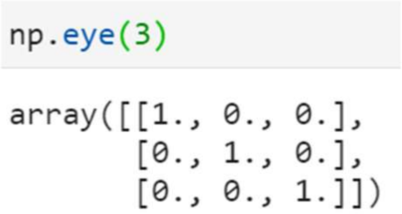
```
np.ones((3,3))
```

```
array([[1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.]])
```



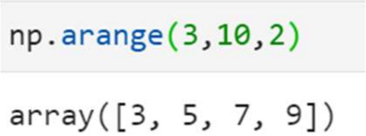
```
np.zeros((3,3))
```

```
array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])
```



```
np.eye(3)
```

```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

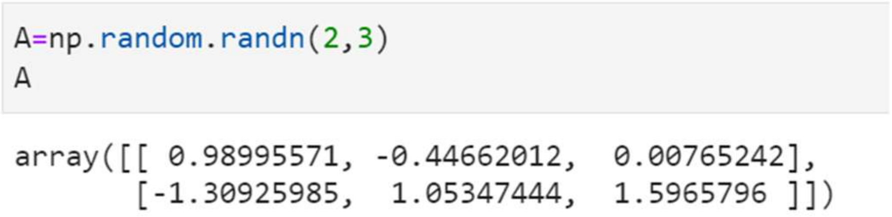


```
np.arange(3,10,2)
```

```
array([3, 5, 7, 9])
```

Random Number Generators

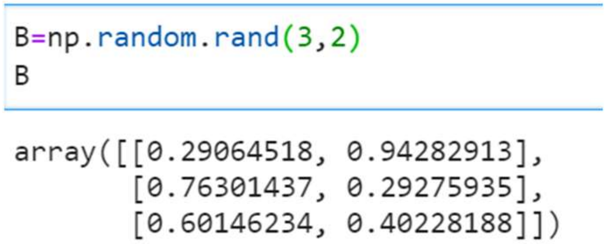
Standard normal random variables



```
A=np.random.randn(2,3)
A
```

```
array([[ 0.98995571, -0.44662012,  0.00765242],
       [-1.30925985,  1.05347444,  1.5965796 ]])
```

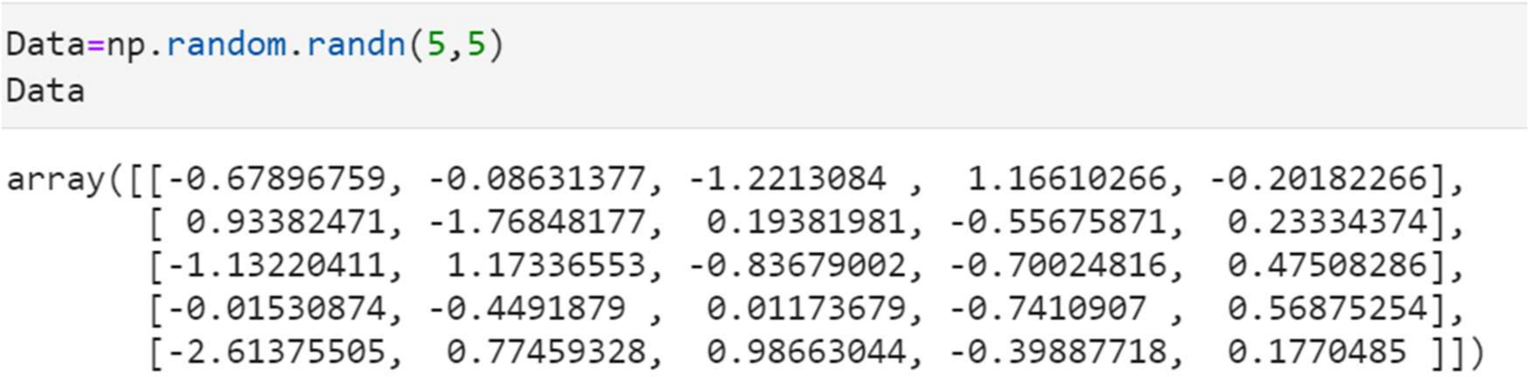
Uniform [0,1] random variables



```
B=np.random.rand(3,2)
B
```

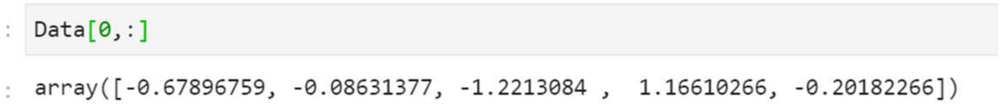
```
array([[0.29064518, 0.94282913],
       [0.76301437, 0.29275935],
       [0.60146234, 0.40228188]])
```

Array index



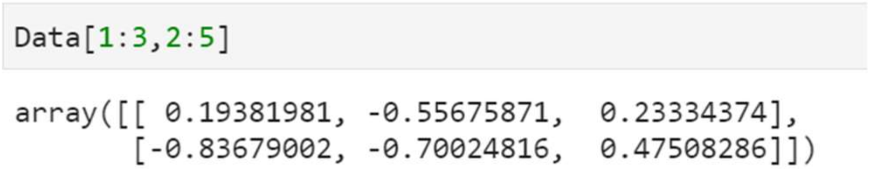
```
Data=np.random.randn(5,5)
Data
```

```
array([[-0.67896759, -0.08631377, -1.2213084 ,  1.16610266, -0.20182266],
       [ 0.93382471, -1.76848177,  0.19381981, -0.55675871,  0.23334374],
       [-1.13220411,  1.17336553, -0.83679002, -0.70024816,  0.47508286],
       [-0.01530874, -0.4491879 ,  0.01173679, -0.7410907 ,  0.56875254],
       [-2.61375505,  0.77459328,  0.98663044, -0.39887718,  0.1770485 ]])
```



```
Data[0,:]
```

```
array([-0.67896759, -0.08631377, -1.2213084 ,  1.16610266, -0.20182266])
```

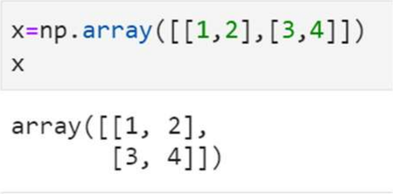


```
Data[1:3,2:5]
```

```
array([[ 0.19381981, -0.55675871,  0.23334374],
       [-0.83679002, -0.70024816,  0.47508286]])
```

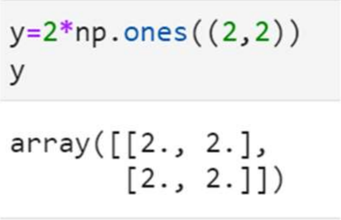
Mathematics with Array

- Element-wise computation



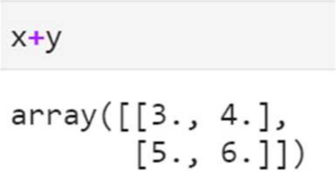
```
x=np.array([[1,2],[3,4]])
x
```

```
array([[1, 2],
       [3, 4]])
```



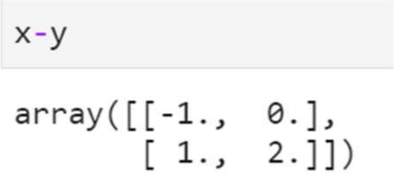
```
y=2*np.ones((2,2))
y
```

```
array([[2., 2.],
       [2., 2.]])
```



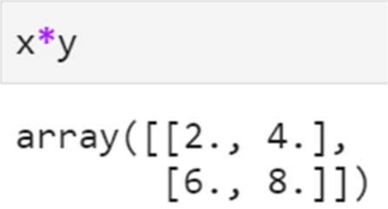
```
x+y
```

```
array([[3., 4.],
       [5., 6.]])
```



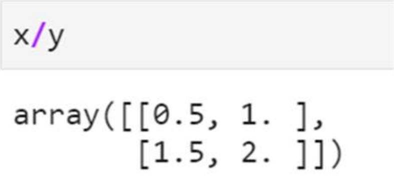
```
x-y
```

```
array([[-1.,  0.],
       [ 1.,  2.]])
```



```
x*y
```

```
array([[2., 4.],
       [6., 8.]])
```

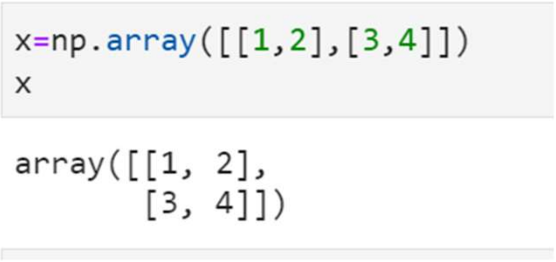


```
x/y
```

```
array([[0.5, 1. ],
       [1.5, 2. ]])
```

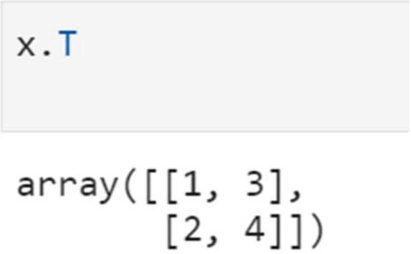
Other method: sum, mean, T

Transpose



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

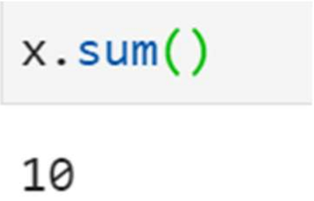
```
array([[1, 2],
       [3, 4]])
```



```
x.T
```

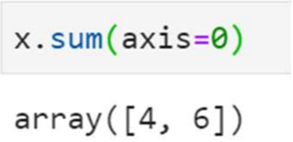
```
array([[1, 3],
       [2, 4]])
```

sum



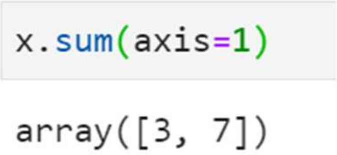
```
x.sum()
```

```
10
```



```
x.sum(axis=0)
```

```
array([4, 6])
```

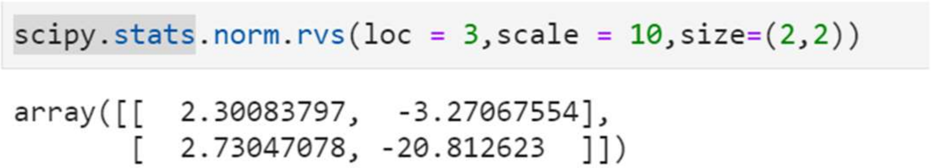


```
x.sum(axis=1)
```

```
array([3, 7])
```

Statistics in Scipy

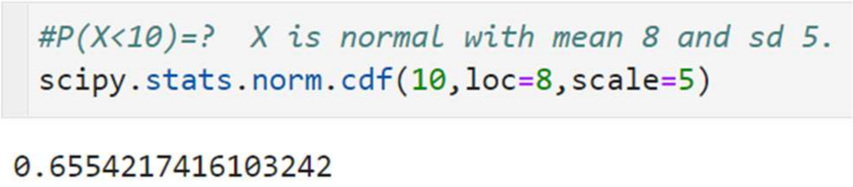
Generate normal random variable



```
scipy.stats.norm.rvs(loc = 3,scale = 10,size=(2,2))
```

```
array([[  2.30083797,  -3.27067554],
       [  2.73047078, -20.812623  ]])
```

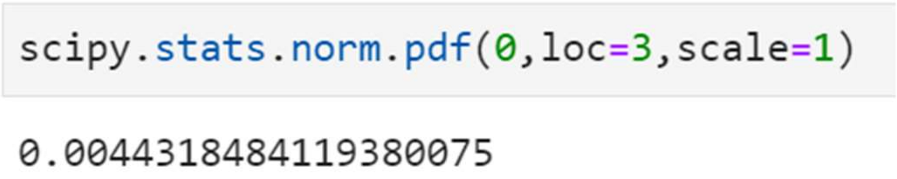
Compute cumulative probability:
$i.e. P(X < 10)$



```
#P(X<10)=?  X is normal with mean 8 and sd 5.
scipy.stats.norm.cdf(10,loc=8,scale=5)
```

```
0.6554217416103242
```

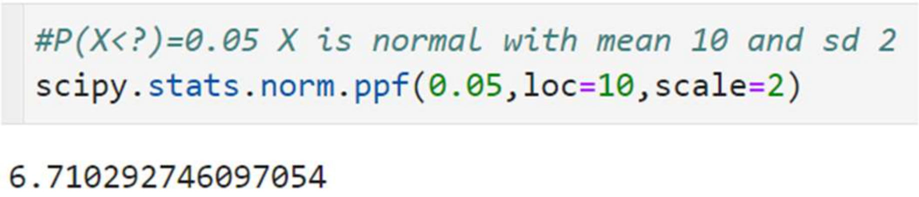
Compute probability density



```
scipy.stats.norm.pdf(0,loc=3,scale=1)
```

```
0.0044318484119380075
```

Compute critical value
i.e. P(X<?)=0.05



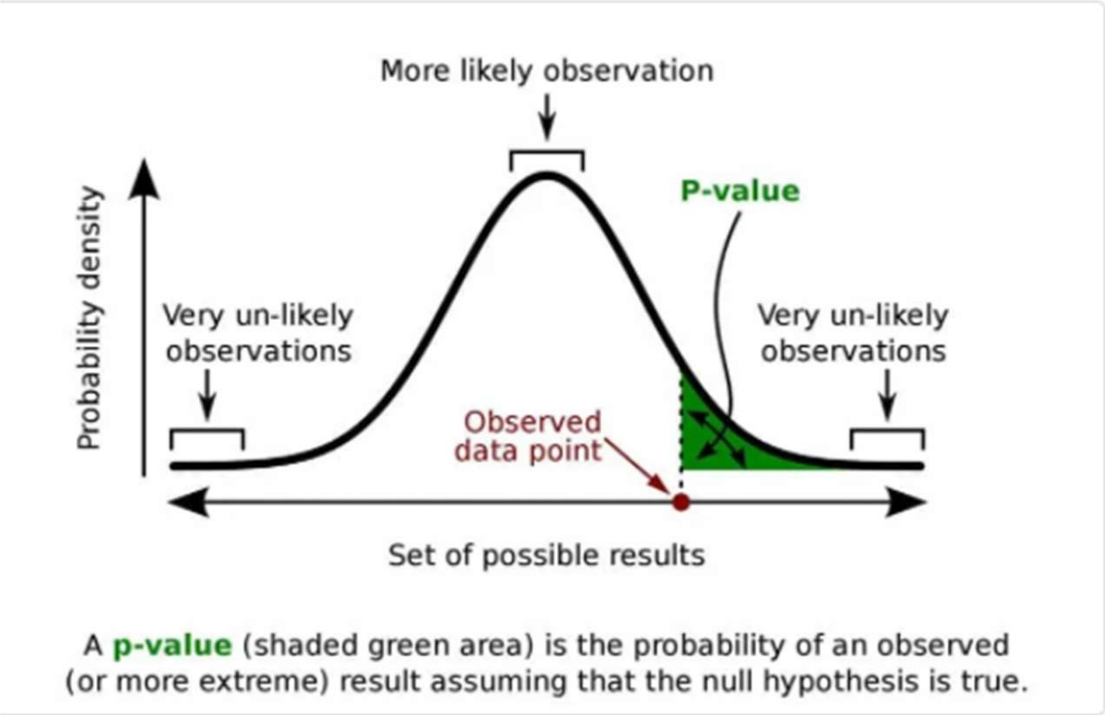
```
#P(X<?)=0.05 X is normal with mean 10 and sd 2
scipy.stats.norm.ppf(0.05,loc=10,scale=2)
```

```
6.710292746097054
```

Practice

1) **Find 95% VaR** $: P(X > VaR) = 0.95$ where Wealth X is normal with mean=1M and std=0.5M

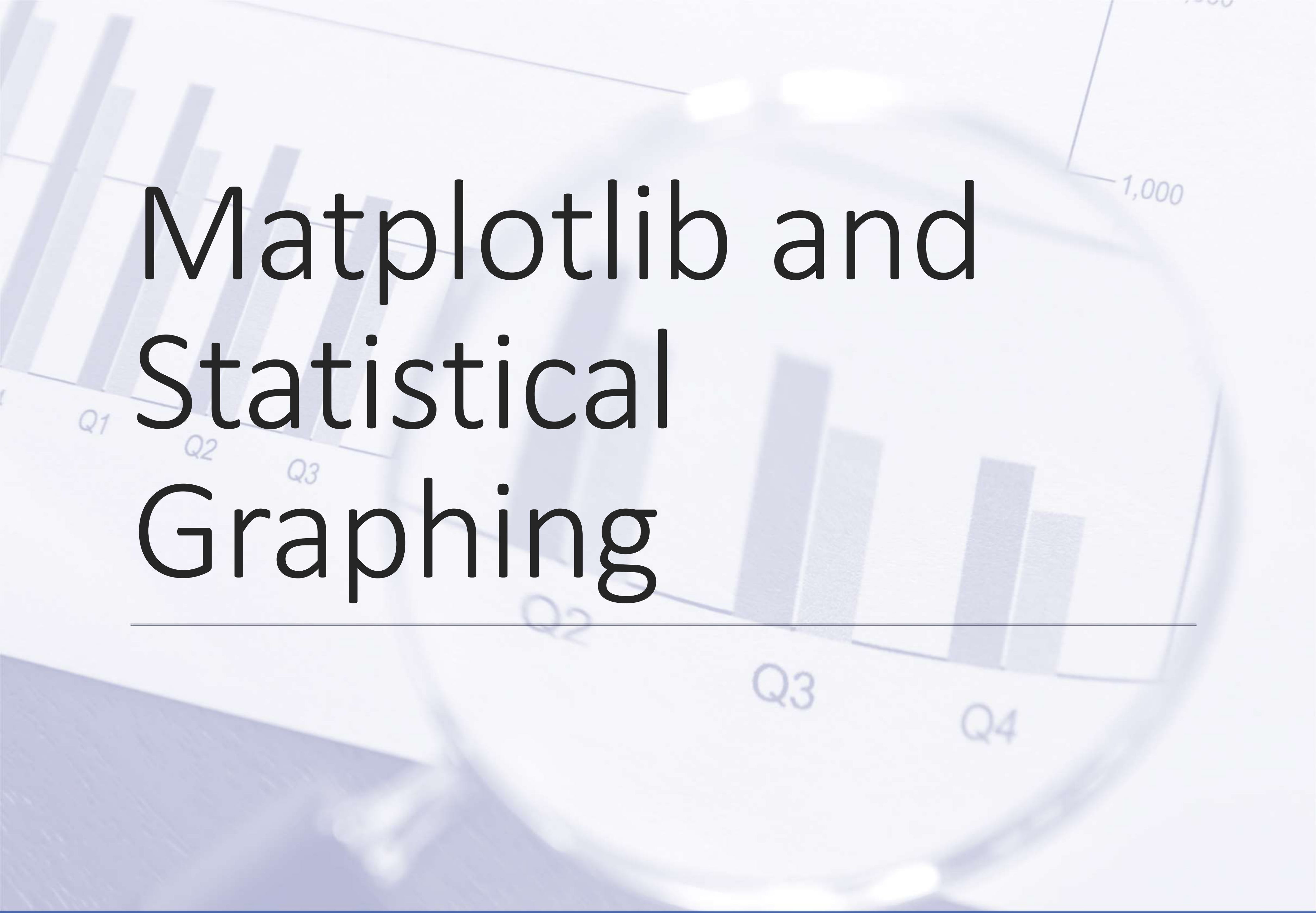
2) In upper tail hypothesis testing, find p-value= $P(\hat{z} > 2.3)$



A **p-value** (shaded green area) is the probability of an observed (or more extreme) result assuming that the null hypothesis is true.

$$H_0: \mu \leq 100$$
$$H_a: \mu > 100$$

$$\hat{z} = \frac{\bar{x} - \mu}{\frac{\sigma}{\sqrt{n}}} = 2.3$$



Matplotlib and Statistical Graphing

Visualizing Data

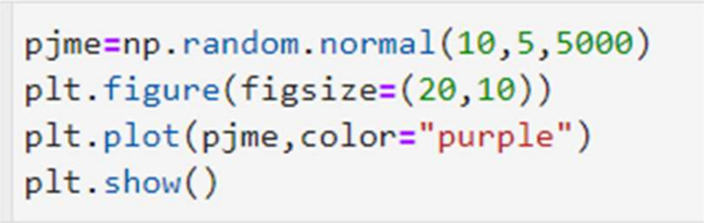
Data visualization is as much a part of the data processing step as the data presentation step.

- It is much easier to compare values when they are plotted than numeric values.

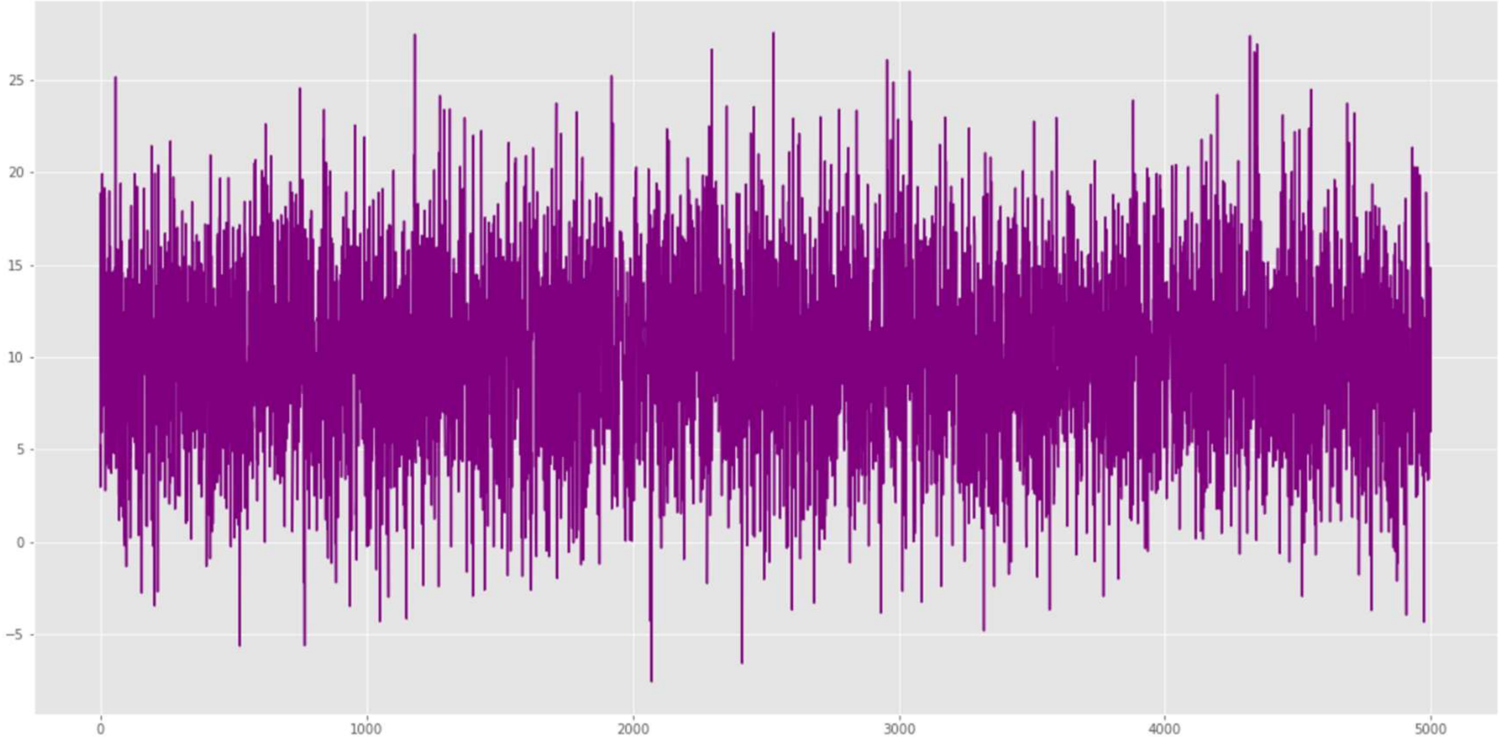
- By visualizing data we are able to get a better intuitive sense of the data than would be possible by looking at tables of values alone.

- Visualizations can bring to light hidden patterns in data, that you, the analyst, can exploit for model selection

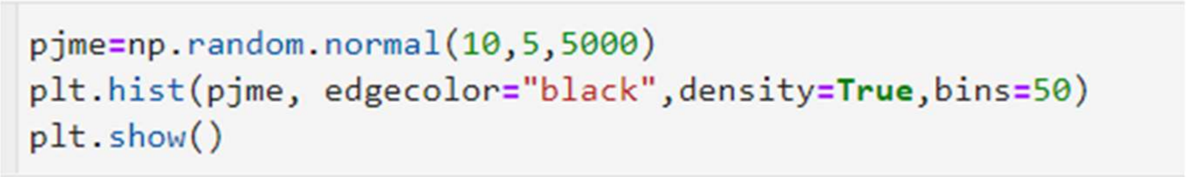
Line Plot



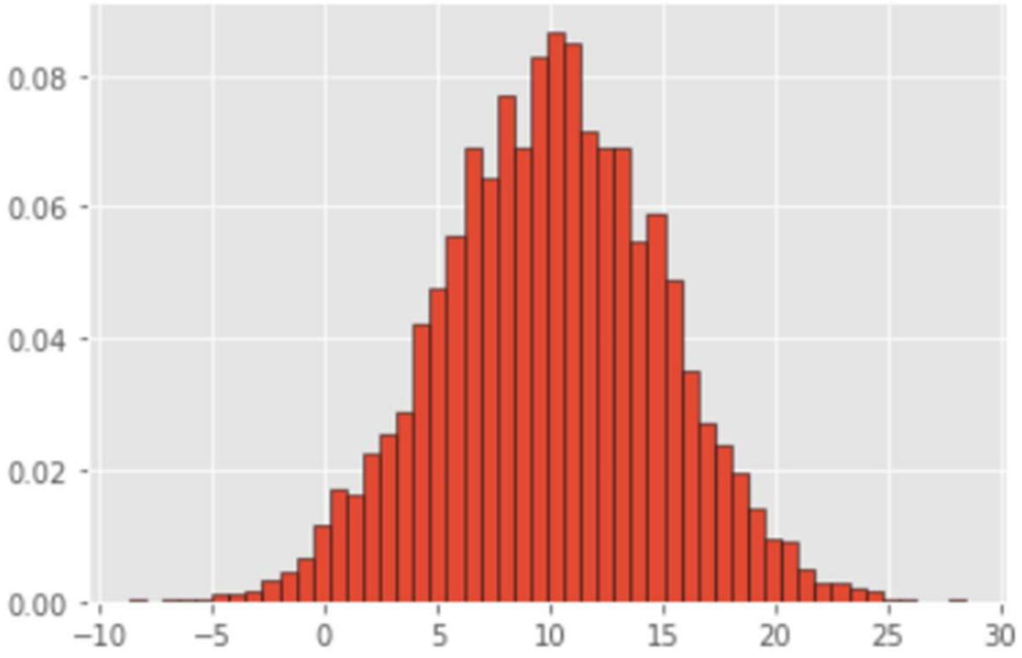
```
pjme=np.random.normal(10,5,5000)
plt.figure(figsize=(20,10))
plt.plot(pjme,color="purple")
plt.show()
```



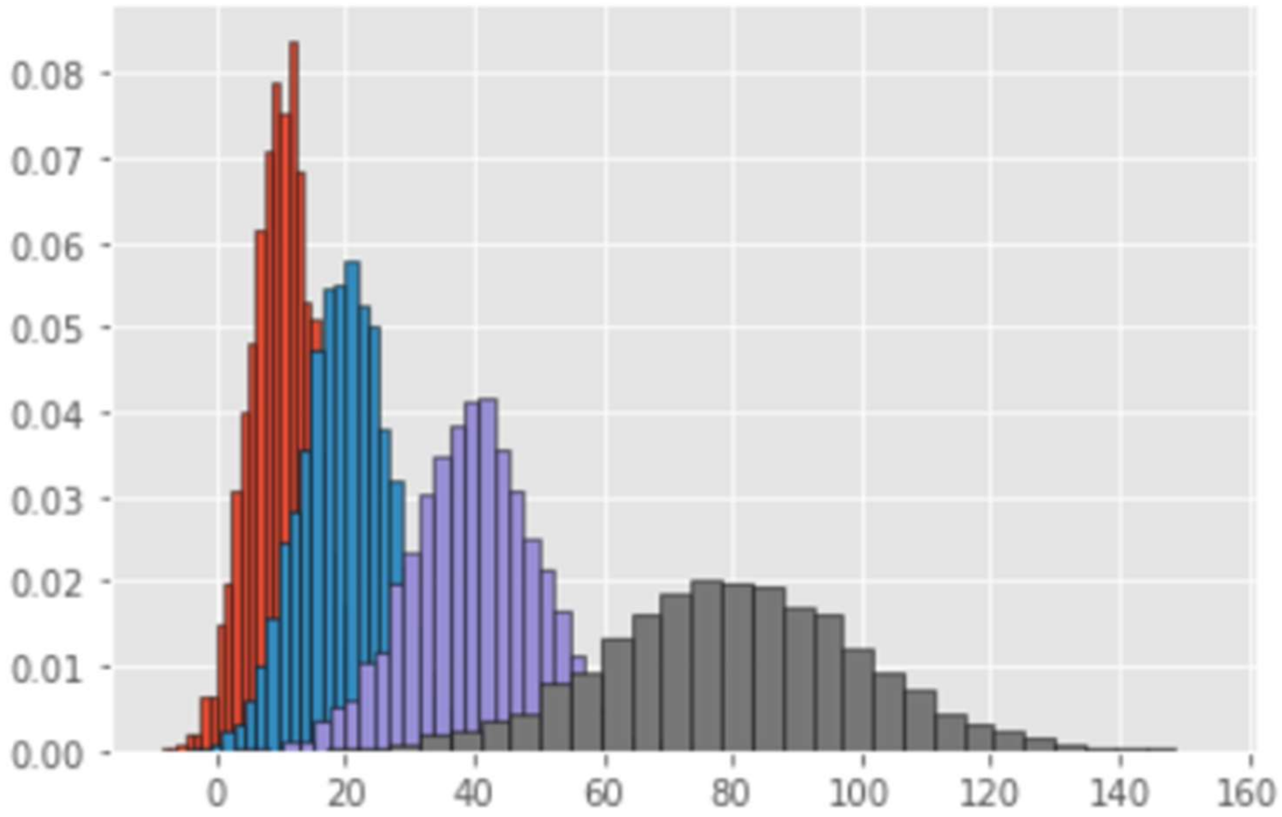
Histogram (Profile)



```
pjme=np.random.normal(10,5,5000)
plt.hist(pjme, edgecolor="black",density=True,bins=50)
plt.show()
```

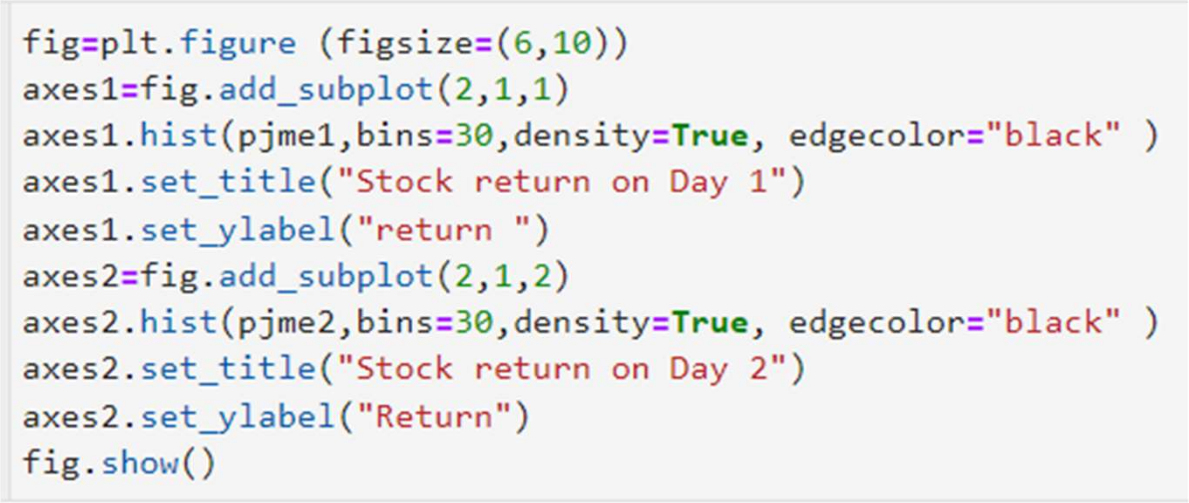


Dynamic changes of profiles across time are important in identifying the structural change of pattern.

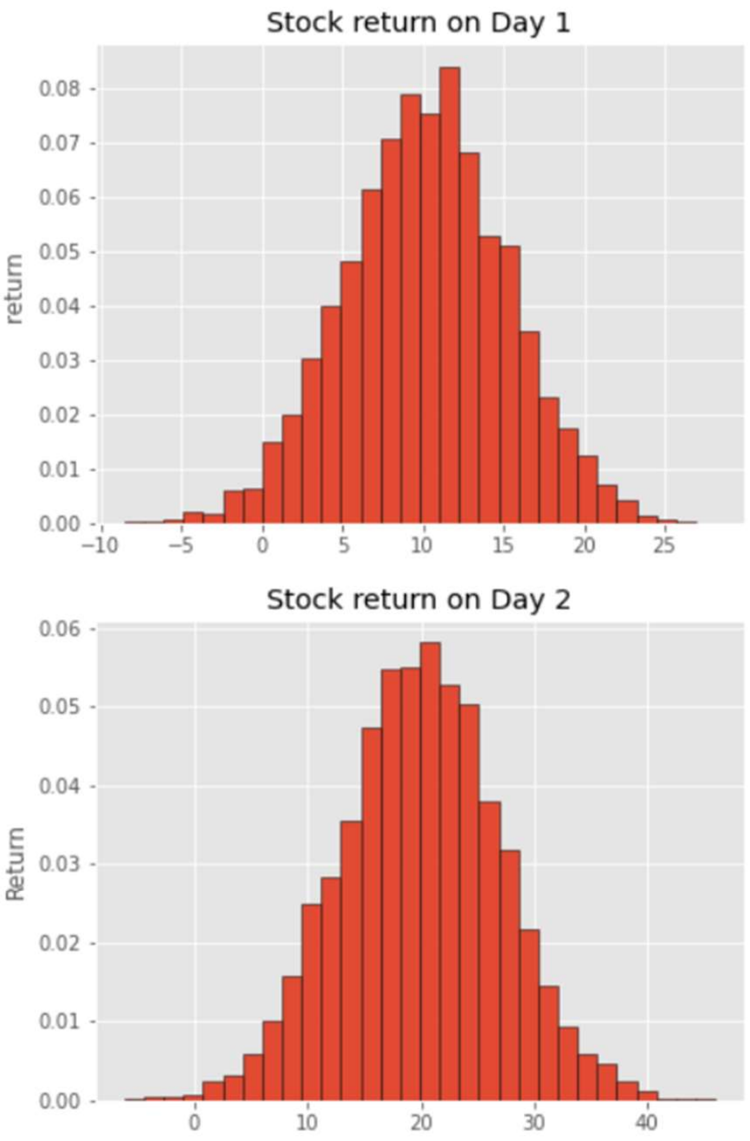


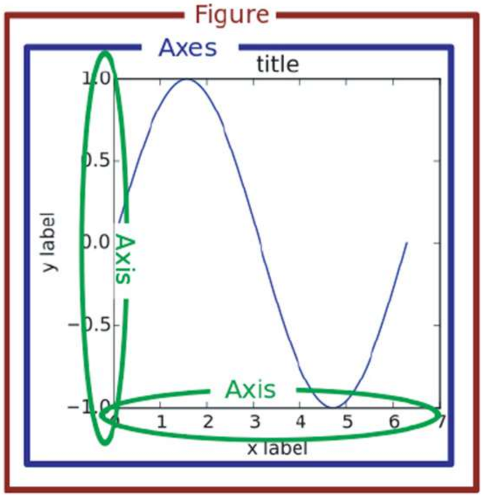
Dynamic changes of profiles across time are important in identifying the structural change of pattern.

Figure object (Optional)

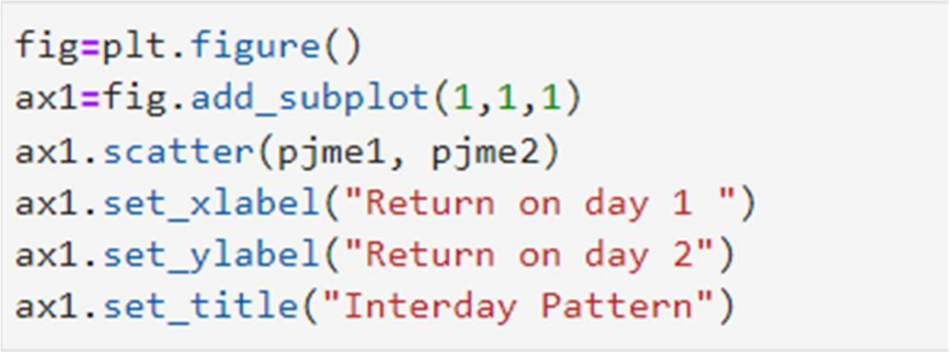


```
fig=plt.figure (figsize=(6,10))
axes1=fig.add_subplot(2,1,1)
axes1.hist(pjme1,bins=30,density=True, edgecolor="black" )
axes1.set_title("Stock return on Day 1")
axes1.set_ylabel("return ")
axes2=fig.add_subplot(2,1,2)
axes2.hist(pjme2,bins=30,density=True, edgecolor="black" )
axes2.set_title("Stock return on Day 2")
axes2.set_ylabel("Return")
fig.show()
```

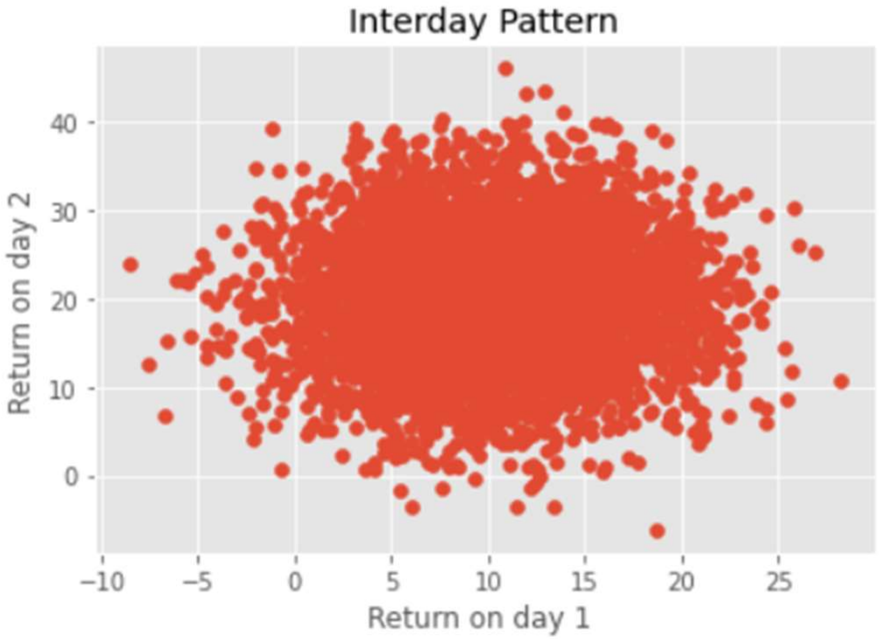




Graph of Bivariate Variables



```
fig=plt.figure()
ax1=fig.add_subplot(1,1,1)
ax1.scatter(pjme1, pjme2)
ax1.set_xlabel("Return on day 1 ")
ax1.set_ylabel("Return on day 2")
ax1.set_title("Interday Pattern")
```



Practice :

a) We will use random number generator to generate daily changes of stock price(252 days). For simplicity, we assume that the daily change follows a standard normal distribution.

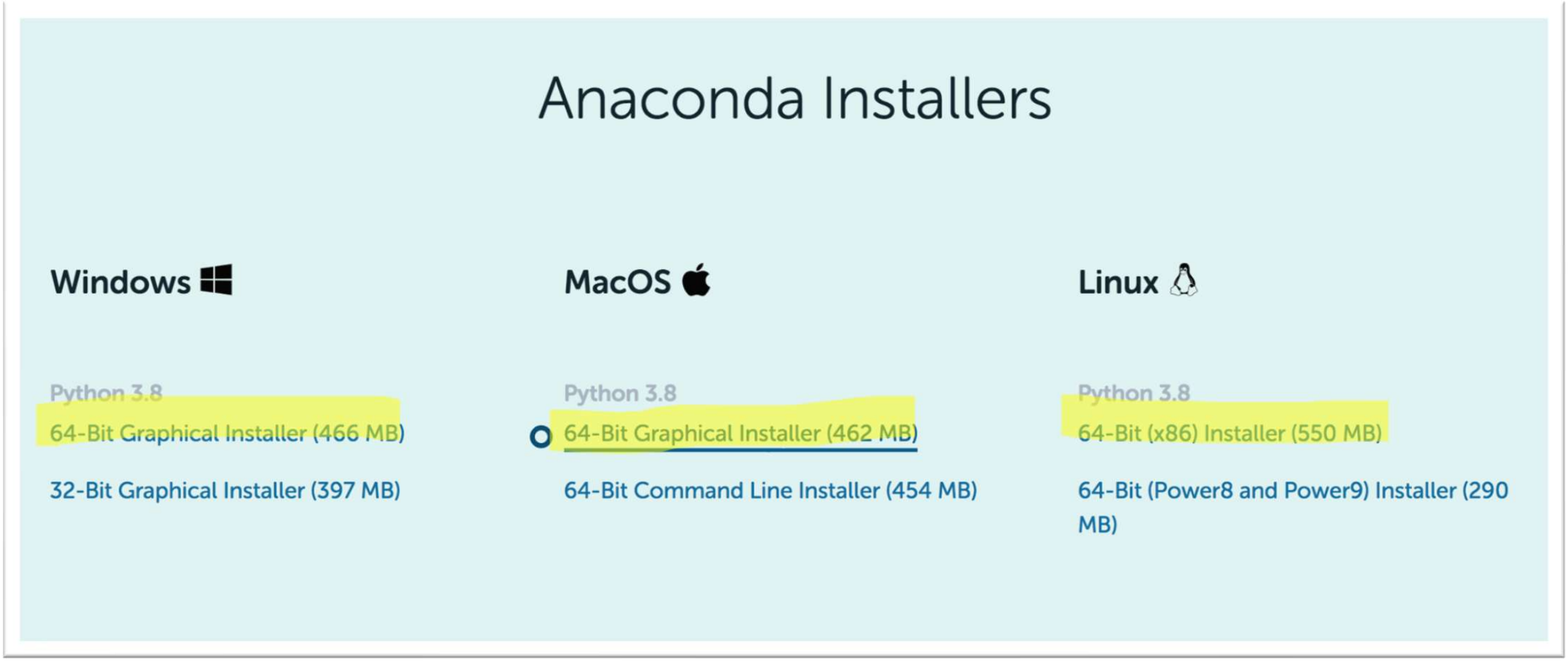
b) Apply cumulative sum method of numpy array to compute accumulative sum of daily change, which is used to mimic stock price.

c) Plot stock price.

Appendix: Installation of Anaconda

Official Website:

https://www.anaconda.com/products/individual



Anaconda Installers

Windows

Python 3.8

64-Bit Graphical Installer (466 MB)

32-Bit Graphical Installer (397 MB)

MacOS

Python 3.8

64-Bit Graphical Installer (462 MB)

64-Bit Command Line Installer (454 MB)

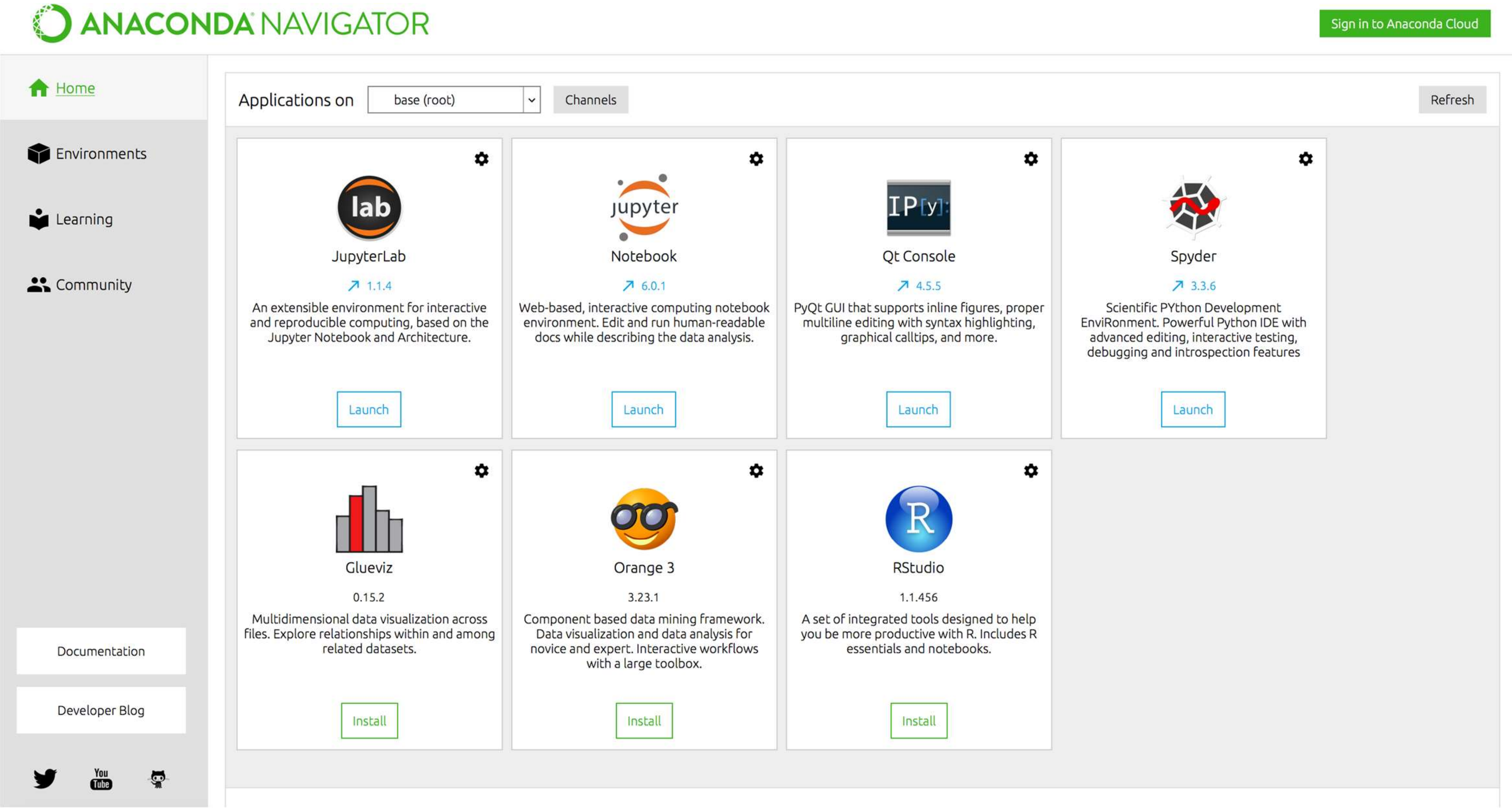
Linux

Python 3.8

64-Bit (x86) Installer (550 MB)

64-Bit (Power8 and Power9) Installer (290 MB)

Anaconda Navigator:



JupyterLab 1.1.4 — An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture. Launch

Notebook 6.0.1 — Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis. Launch

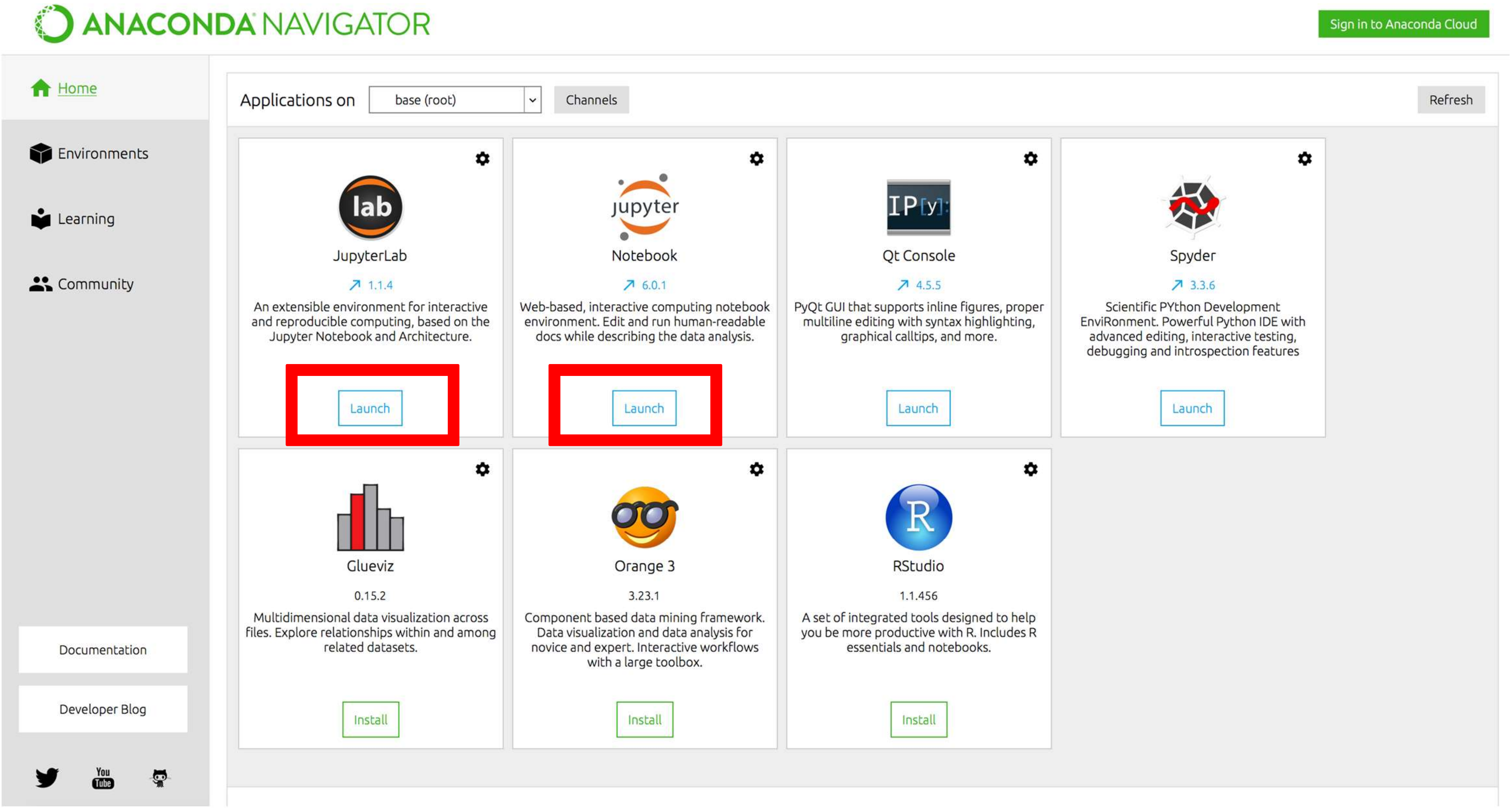
Qt Console 4.5.5 — PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more. Launch

Spyder 3.3.6 — Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features Launch

Glueviz 0.15.2 — Multidimensional data visualization across files. Explore relationships within and among related datasets. Install

Orange 3 3.23.1 — Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox. Install

RStudio 1.1.456 — A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks. Install



JupyterLab
1.1.4
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Notebook
6.0.1
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Qt Console
4.5.5
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Spyder
3.3.6
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

Glueviz
0.15.2
Multidimensional data visualization across files. Explore relationships within and among related datasets.

Orange 3
3.23.1
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

RStudio
1.1.456
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.