

# Introduction to Python

A walk through in learning Python effectively

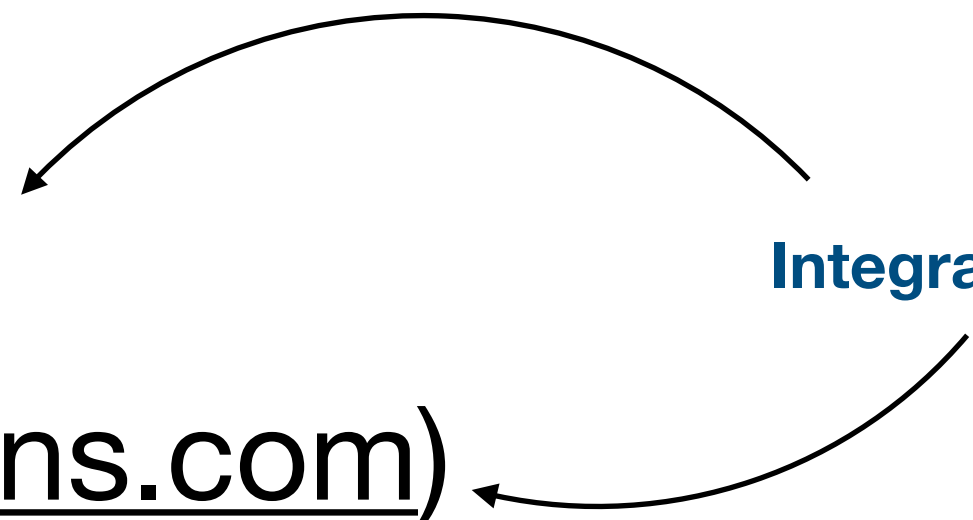
**Norhasliza Yusof**

email : [norhaslizay@um.edu.my](mailto:norhaslizay@um.edu.my)

# Basic info

- Python 3.X installed in your computer
- Libraries need to learn scientific computing or machine learning are :
  - \* Matplotlib (library for visualisation - plotting graph)
  - \* Numpy (numerical python library to call special mathematical function, handling arrays)
  - \* Scipy (scientific computing library and utilised dumpy as its backbone)
  - \* Seaborn (statistical data visualisation - based on Matplotlib)
  - \* Pytorch (deep learning library)
  - \* Tensorflow/Keras (machine learning library)

# How to install Python in your computer

- [python.org](https://python.org)
  - [www.anaconda.com](https://www.anaconda.com)
  - pycharm ([www.jetbrains.com](https://www.jetbrains.com))
- 
- Integrated Graphic User Interface GUI + libraries automatically install

Or

If you are using Linux distro or MacOSX, popular option is via `sudo apt-get install python3` or `brew install python3` (in MacOSX)

Additional remarks:

anaconda uses `conda -install` command to install additional library  
[python.org](https://python.org) or distribution via linux/Mac uses `pip/pip3 -install` command

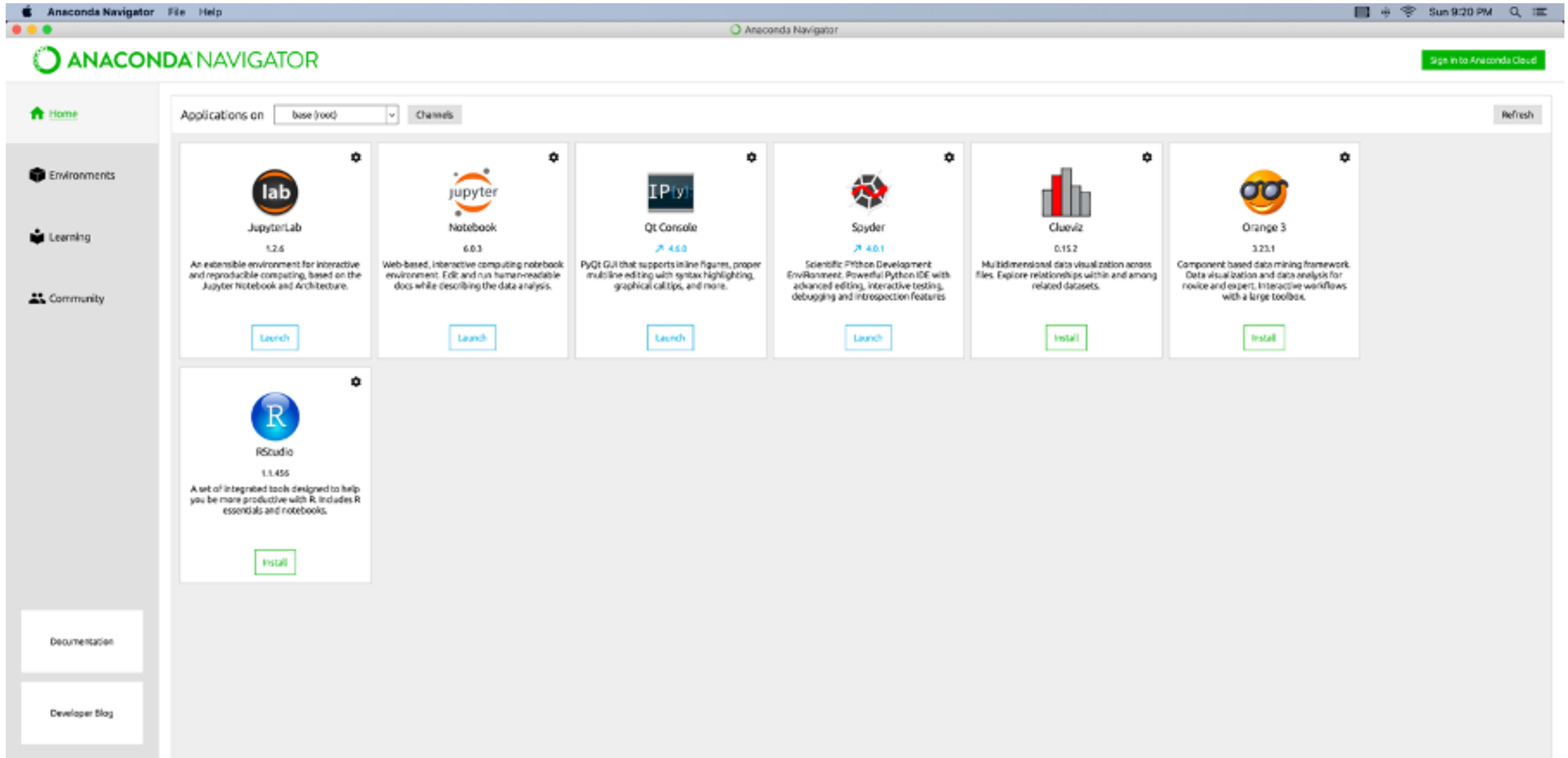
# Installation

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

## Anaconda Installers

Windows 	MacOS 	Linux 
Python 3.8	Python 3.8	Python 3.8
64-Bit Graphical Installer (466 MB)	64-Bit Graphical Installer (462 MB)	64-Bit (x86) Installer (550 MB)
32-Bit Graphical Installer (397 MB)	64-Bit Command Line Installer (454 MB)	64-Bit (Power8 and Power9) Installer (290 MB)

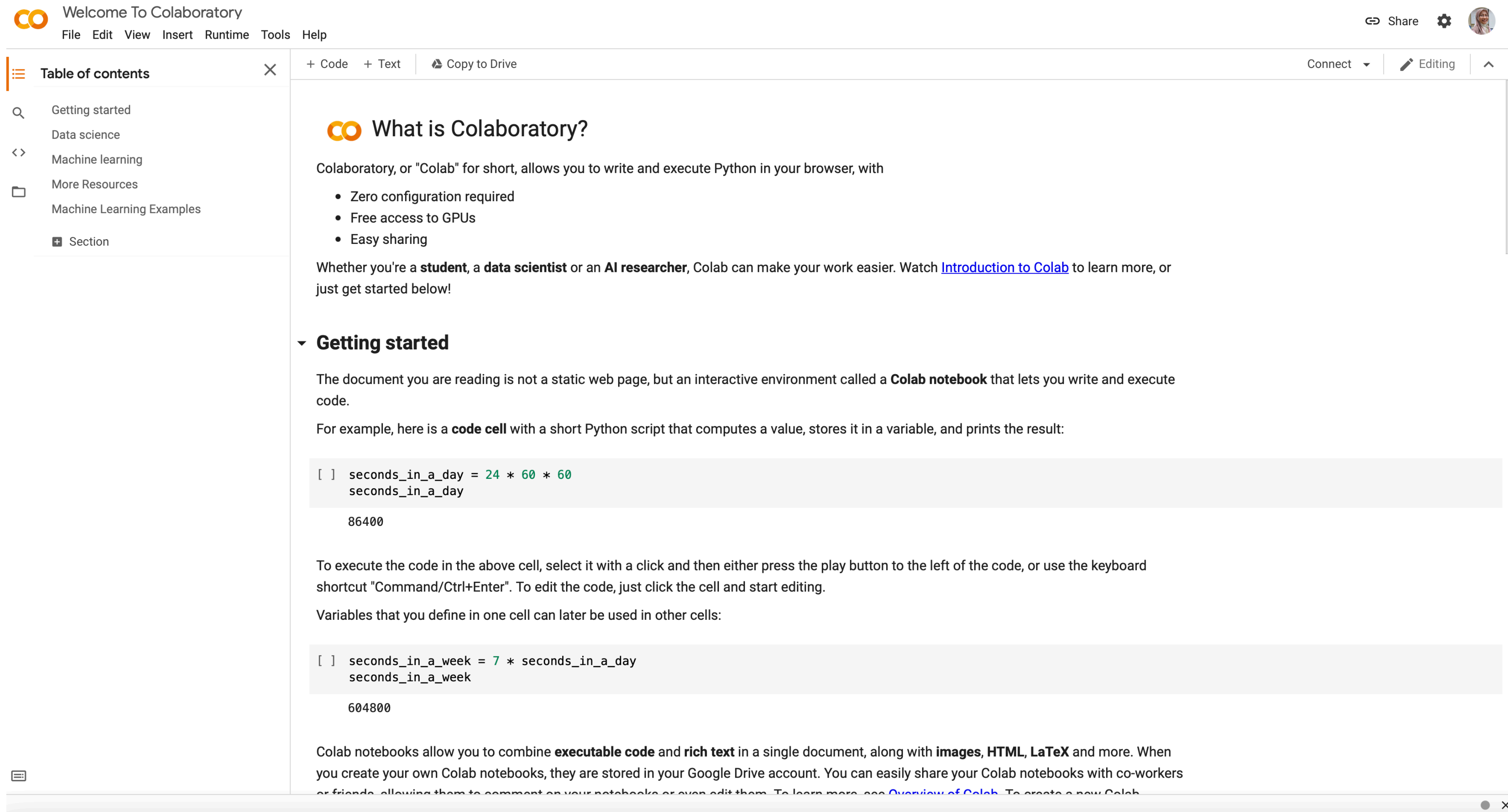
# Anaconda Navigation



**Another solution to use Python  
without installation?**

# Google Colab (powered by Google)

<https://colab.research.google.com/notebooks>



The screenshot displays the Google Colaboratory web interface. At the top, a navigation bar includes the Colab logo, the text 'Welcome To Colaboratory', and a menu with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. On the right side of the bar are links for 'Share', a settings gear, and a user profile icon. Below the navigation bar, a sidebar on the left contains a 'Table of contents' section with links to 'Getting started', 'Data science', 'Machine learning', 'More Resources', 'Machine Learning Examples', and a 'Section' button. The main content area is titled 'What is Colaboratory?' and features the Colab logo. It explains that Colaboratory, or 'Colab', allows users to write and execute Python in their browser. A bulleted list highlights three key features: 'Zero configuration required', 'Free access to GPUs', and 'Easy sharing'. Below this, a paragraph states that Colab can make work easier for students, data scientists, and AI researchers, with a link to 'Introduction to Colab'. A section titled 'Getting started' follows, explaining that the document is an interactive 'Colab notebook' and providing an example of a code cell. The example code calculates the number of seconds in a day (24 \* 60 \* 60) and prints the result, 86400. It then shows how to execute the code by clicking a play button or using a keyboard shortcut, and how to edit the code. Another code cell is shown, calculating the number of seconds in a week (7 \* seconds\_in\_a\_day) and printing the result, 604800. The interface concludes by stating that Colab notebooks allow combining executable code, rich text, images, HTML, LaTeX, and more, and that they are stored in the user's Google Drive account for easy sharing.

CO Welcome To Colaboratory  
File Edit View Insert Runtime Tools Help

CO What is Colaboratory?

Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser, with

- Zero configuration required
- Free access to GPUs
- Easy sharing

Whether you're a **student**, a **data scientist** or an **AI researcher**, Colab can make your work easier. Watch [Introduction to Colab](#) to learn more, or just get started below!

### Getting started

The document you are reading is not a static web page, but an interactive environment called a **Colab notebook** that lets you write and execute code.

For example, here is a **code cell** with a short Python script that computes a value, stores it in a variable, and prints the result:

```
[ ] seconds_in_a_day = 24 * 60 * 60
seconds_in_a_day

86400
```

To execute the code in the above cell, select it with a click and then either press the play button to the left of the code, or use the keyboard shortcut "Command/Ctrl+Enter". To edit the code, just click the cell and start editing.

Variables that you define in one cell can later be used in other cells:

```
[ ] seconds_in_a_week = 7 * seconds_in_a_day
seconds_in_a_week

604800
```

Colab notebooks allow you to combine **executable code** and **rich text** in a single document, along with **images**, **HTML**, **LaTeX** and more. When you create your own Colab notebooks, they are stored in your Google Drive account. You can easily share your Colab notebooks with co-workers or friends, allowing them to comment on your notebooks or even edit them. To learn more, see [Overview of Colab](#). To create a new Colab



# Introduction to Python

## Basic information

- In programming numbers/values must be assigned by variable. We use “=” to connect values to a variable.
- Numbers/values can be either integer or real (floating number)
- In Python 3.0, all numbers are considered real unless we declared or set its identity.
- Python 3.0 can handle complex number and assign the complex number value as  $j$  instead of  $i$ .
- Python command for different type of numbers : integer (**int**), real (**float**), complex (**complex**)
- To produce output on your screen, use command **print()**



# Introduction to Python

Basic example for learning python syntax

```
In [1]: print('Hello! Welcome to Astrostatics and Machine Learning School!') #print characters/statement
```

Hello! Welcome to Astrostatics and Machine Learning School!

```
In [2]: a = 10  
b = 4  
print(a) #print value of a  
print(b) #print value of b
```

10  
4

```
In [3]: print (a,b) #print a and b side by side
```

10 4

```
In [4]: print('a,b') #can you see the differences?
```

a,b

To check the type of object in Python

```
In [5]: x = 10    #integer  
y = 10.5    #floating number/real  
z = 10j    #complex number
```

```
In [6]: print(type(x))  
print (type(y))  
print(type(z))
```

<class 'int'>  
<class 'float'>  
<class 'complex'>

# Introduction to Python

## Basic Mathematical Operations

Operator	Description	Example
+	Addition	a+b
-	Substraction	a-b
*	Multiplication	a*b
/	Division	a/b
**	Power	a**b
>	Greater than	a>b
<	Lower than	a<b
>=	Equal and greater than	a>=b
<=	Less and less than	A<=b

# Example

## Mathematical operations

We have to add numpy to call array. We have to import numpy and put the shortform np

```
In [1]: import numpy as np
a = np.array([1,2,3])
b = np.array([1,0,0])
```

For example we want to test the summation of a and b array

```
In [3]: a+b
```

```
Out[3]: array([2, 2, 3])
```

and test for 0.5 multiply by a

```
In [2]: 0.5*a
```

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

To create 0 vector or 0 matrix use np.zeros command

```
In [6]: np.zeros(5)
```

```
Out[6]: array([0., 0., 0., 0., 0.])
```

```
In [7]: np.zeros((2,2))
```

```
Out[7]: array([[0., 0.],
               [0., 0.]])
```

All standard matrix operations can be done using matrix product

```
In [11]: a = [[1,0],[0,1]]
b = [[4,1],[2,2]]
np.dot(a,b)
```

```
Out[11]: array([[4, 1],
               [2, 2]])
```

# Introduction to Python

## Mathematical function

- To call mathematical functions, we need numpy
- Numpy provides a high-performance multi-D array and basic tools to compute and manipulate array
- In python, all trigonometry values are in radian
- Example to write  $\sin(2\pi)$  using numpy

```
import numpy as np
A = np.sin(2*np.pi)
```

log()	sin()	cos()	tan()
pi	round	sqrt	truncate
absolute	exp	array	degree

# Introduction to Python

## Loops

We will use for loops intensively

```
In [1]: for i in range(10):  
        print(i)
```

```
0  
1  
2  
3  
4  
5  
6  
7  
8  
9
```

python counted iteration from zero not one

```
In [3]: for i in range(2,5):  
        print(i)
```

```
2  
3  
4
```

while loop is used when we have condition to fulfill

```
In [6]: i = 0  
        while i<10:  
            i = i+1  
            print(i)
```

```
1  
2  
3  
4  
5  
6  
7  
8  
9  
10
```

# Introduction to Python

## Arrays, Vectors and Matrices

We have to add numpy to call array. We have to import numpy and put the shortform np

```
In [1]: import numpy as np
a = np.array([1,2,3])
b = np.array([1,0,0])
```

For example we want to test the summation of a and b array

```
In [3]: a+b
```

```
Out[3]: array([2, 2, 3])
```

and test for 0.5 multiply by a

```
In [2]: 0.5*a
```

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

To create 0 vector or 0 matrix use np.zeros command

```
In [6]: np.zeros(5)
```

```
Out[6]: array([0., 0., 0., 0., 0.])
```

```
In [7]: np.zeros((2,2))
```

```
Out[7]: array([[0., 0.],
               [0., 0.]])
```

All standard matrix operations can be done using matrix product

```
In [11]: a = [[1,0],[0,1]]
b = [[4,1],[2,2]]
np.dot(a,b)
```

```
Out[11]: array([[4, 1],
                [2, 2]])
```

# Introduction to Python

## User defined function

We have to define our own functions. For example let us define a function :

$$f(x) = 3x^2 + x + 4$$

```
In [1]: def f(x):  
        return 3*x**2+x+4
```

```
In [ ]: x = 1 #define the value of x
```

```
In [3]: print(f(x))  
8
```

or we could do define x directly in f(x)

```
In [4]: f(1)
```

```
Out[4]: 8
```



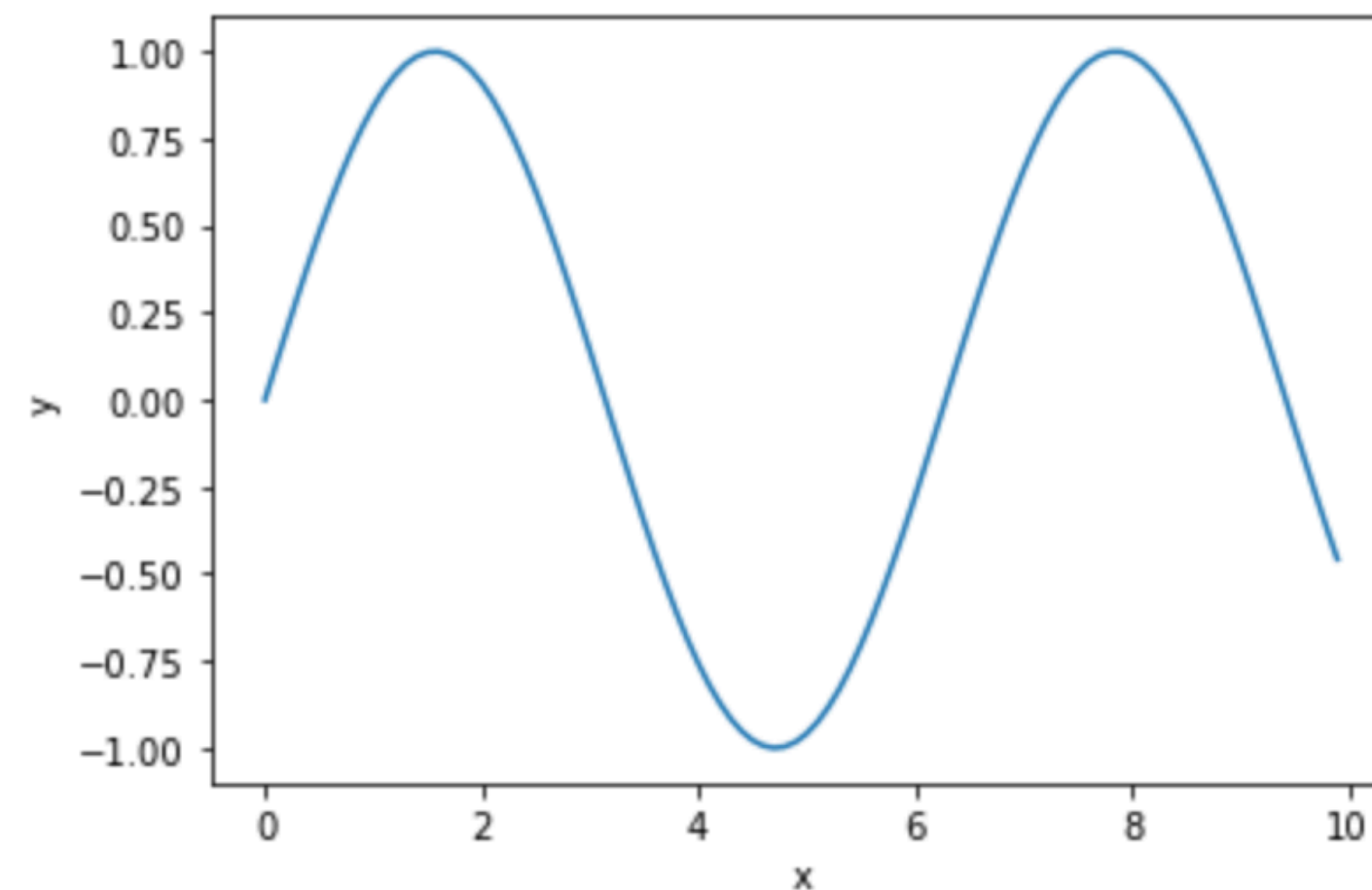
# Visualisation/Making plots

We will use Matplotlib package to create plots

In this example, we are going to plot a sine plot using matplotlib library

```
In [8]: import matplotlib.pyplot as plt #import matplotlib library
import numpy as np #import numpy library
x = np.arange(0,10,0.1) #set range for x
y = np.sin(x) #assign y-axis
plt.plot(x,y) #plot x and y value
plt.xlabel('x') #label x-axis
plt.ylabel('y') #label y-axis
```

Out[8]: Text(0, 0.5, 'y')



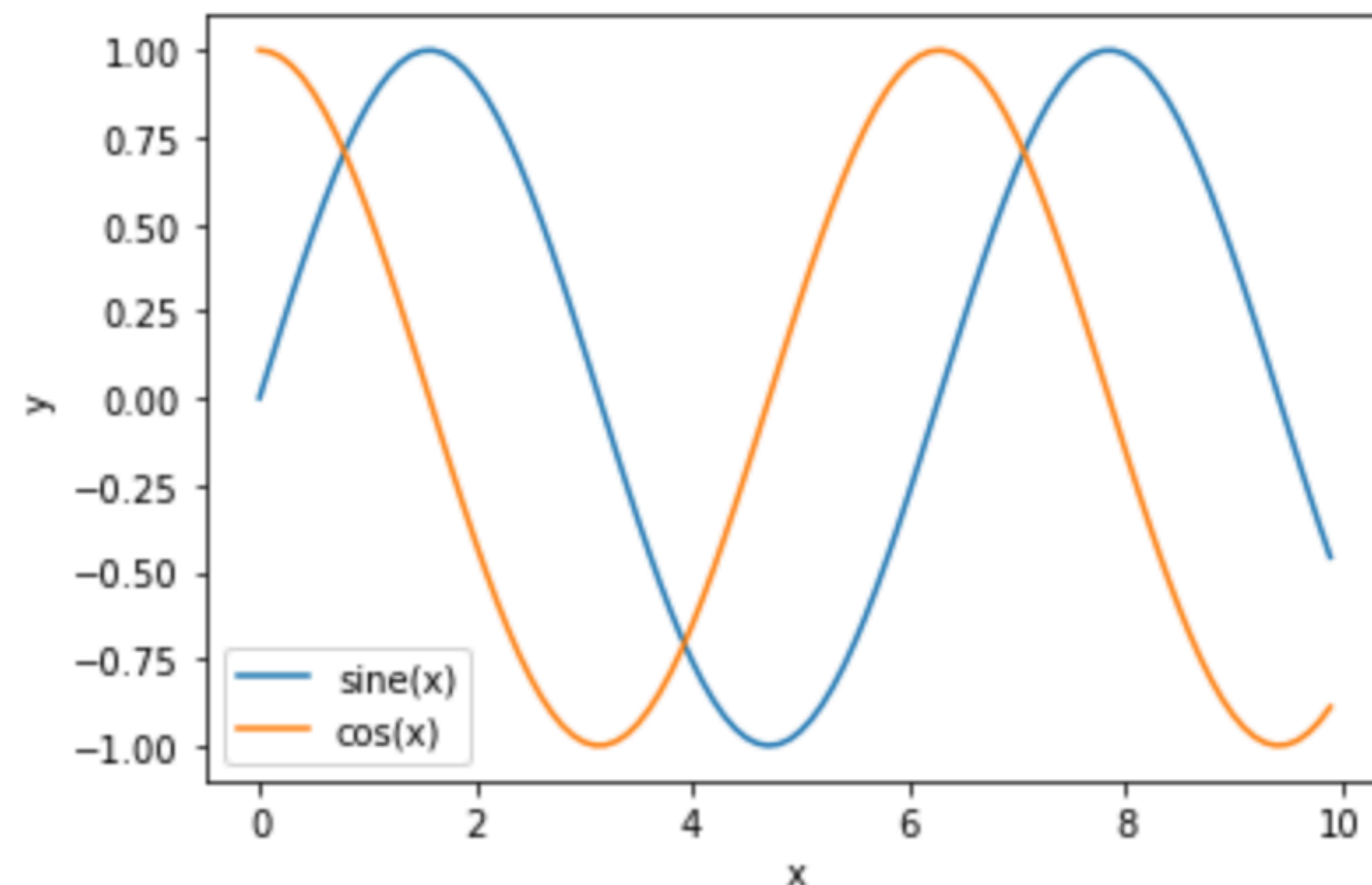
To label our plot, we add additional label in the code

# Introduction to Python

## Plotting two function in one graph

```
In [12]: import matplotlib.pyplot as plt #import matplotlib library
import numpy as np #import numpy library
x = np.arange(0,10,0.1) #set range for x
y = np.sin(x) #assign y-axis
z = np.cos(x) #assign second y-axis
plt.plot(x,y, label='sine(x)') #plot for sine
plt.plot (x,z, label = 'cos(x)') #plot for cos
plt.xlabel('x') #label x-axis
plt.ylabel ('y') #label y-axis
plt.legend() #to show legend
```

Out[12]: <matplotlib.legend.Legend at 0x7ff816bf02d0>



All codes available in [https://github.com/lizayusof/IVC\\_Astrostat\\_ML](https://github.com/lizayusof/IVC_Astrostat_ML)