

What is Python

Python is a high-level, interpreted, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically-typed and garbage-collected

What is Python

Python is a high-level, interpreted, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically-typed and garbage-collected

Python Uses

Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis. Python is a general-purpose language, meaning it can be used to create a variety of different programs and isn't specialized for any specific problems.

How to Install Python

Step 1: Download the Python Installer binaries. Open the official Python website in your web browser. ...

- Step 2: Run the Executable Installer. Once the installer is downloaded, run the Python installer. ...
- Step 3: Add Python to environmental variables. ...
- Step 4: Verify the Python Installation.

What is Anaconda

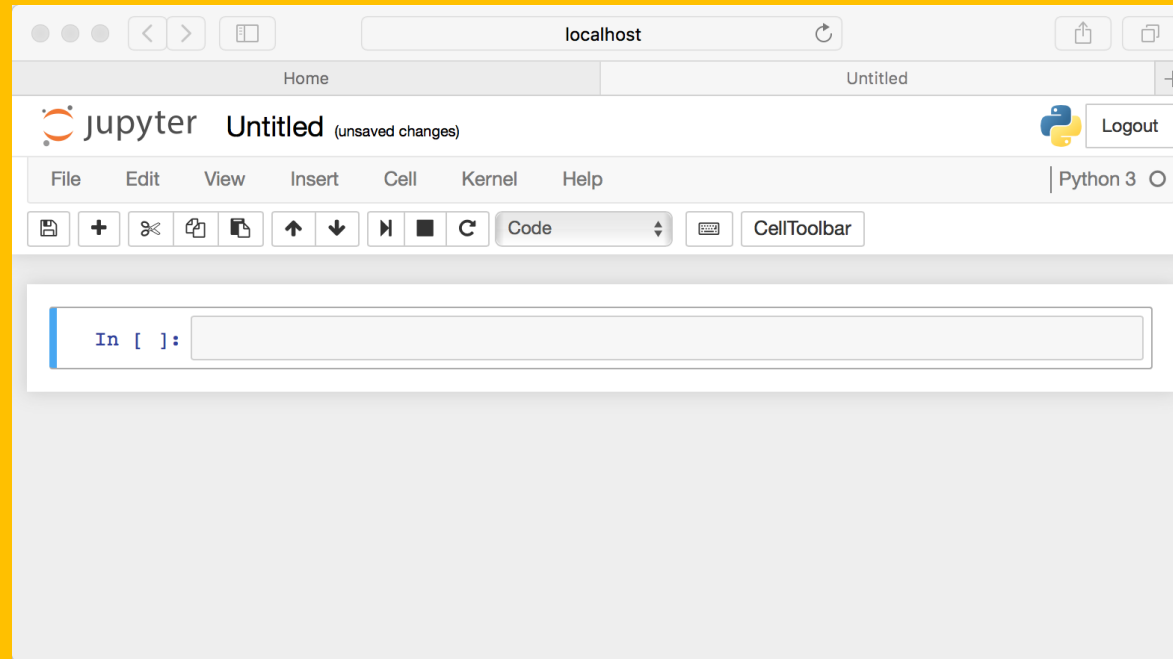
Anaconda is a distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS.

How to Install Anaconda

- Steps: Visit [Anaconda.com/downloads](https://anaconda.com/downloads). ...
- Visit the Anaconda downloads page. Go to the following link:
[Anaconda.com/downloads](https://anaconda.com/downloads). ...
- Select Windows. Select Windows where the three operating systems are listed.
- Download. ...
- Open and run the installer. ...
- Open the Anaconda Prompt from the Windows start menu.

What is a Jupyter Notebook

- JupyterLab is the latest web-based interactive development environment for notebooks, code, and data. Its flexible interface allows users to configure and arrange workflows in data science, scientific computing, computational journalism, and machine learning. A modular design invites extensions to expand and enrich functionality.



Jupyter Notebook Interface

How to Create a Folder in Jupyter Notebook

- To create a new directory using the Jupyter Notebook dashboard, you can click on the drop-down menu labeled New , and then select Folder . You can create new directories in the Jupyter dashboard by clicking on New and then selecting Folder from the drop-down menu. Note that new directories are created as Untitled.

How to Import a Folder in Jupyter Notebook

- First, navigate to the Jupyter Notebook interface home page. ...
- Click the “Upload” button to open the file chooser window.
- Choose the file you wish to upload. ...
- Click “Upload” for each file that you wish to upload.
- Wait for the progress bar to finish for each file.

What is a Library

A library is a previously combined set of codes that can be iteratively used, hence reducing time. As the term suggests it's similar to the physical library that holds reusable resources. Python has founded several open source libraries based on the fact that each library has a root source.

What are Python Libraries

Python has been widely used in the present times being a high-level programming language. The ease of use lies in its syntax which uses a lesser number of codes to express a concept. Therefore, this allows the user to apply python and write programs on both large and small scales. The language supports automatic memory management and has a large standard library.

What are Python Libraries

A Python library defines lines of code that can be reused in other programs. It is basically a collection of modules. Their usefulness lies in the fact that new codes are not required to be written every time the same process is required to run. Libraries in Python play an important role in areas of data science, machine learning, data manipulation applications, etc.

Numpy

Deviating towards the scientific computation areas, NumPy is the most used open-source packages offered by python. It supports large matrices and multidimensional data and has inbuilt mathematical functions for easy computation. The name “NumPy” defines “Numerical Python”. It can be used in linear algebra, random number capability, etc., and can act as a multi-dimensional container for generic data. Python NumPy Array is an object defining N-dimensional array in the form of rows and columns.

Numpy

Deviating towards the scientific computation areas, NumPy is the most used open-source packages offered by python. It supports large matrices and multidimensional data and has inbuilt mathematical functions for easy computation. The name “NumPy” defines “Numerical Python”. It can be used in linear algebra, random number capability, etc., and can act as a multi-dimensional container for generic data. Python NumPy Array is an object defining N-dimensional array in the form of rows and columns.

Matplotlib

This library is used for the plotting of numerical data and used in data analysis. This open-source library is used for publishing high-quality figures like graphs, pie charts, scatterplots, histograms, etc.

Matplotlib

```
import matplotlib.pyplot as plt  
  
num_list = [1.5,0.6,7.8,6]  
plt.bar(range(len(num_list)),num_list)  
plt.show()
```

Matplotlib

```
import matplotlib.pyplot as plt

name_list = ['Monday', 'Tuesday', 'Friday', 'Sunday']
num_list = [1.5, 1.6, 7.8, 6]
num_list2 = [1, 2.3, 3, 2]
x = list(range(len(num_list)))
total_width = 0.8
n = 2
width = total_width / n
plt.bar(x, num_list, width = width, label='boys')
for i in range(len(x)):
    x[i] = x[i] + width
plt.bar(x, num_list2, width = width, label='girls', tick_label
= name_list)
plt.legend(loc='best')
plt.show()
```

Pandas

The panda is an open-source library and BSD licensed. The library is widely used in the data science area. They are mostly used for the analysis, manipulation, and cleaning of data. Without the need for switching it to another language like R, panda makes it possible for the easy operations of modelling and data analysis.

Pandas

The data used by the libraries in python are:

- Tabular data
- Time series with ordered and unordered data.
- Matrix data labelling rows and columns.
- Unlabeled data

Pandas Example

NAME	CLASS	MARKS OBTAIN	TOTAL MARKS
AD	7TH	450	800
VB	8TH	780	800
BN	6TH	760	800
MN	8TH	567	800
KJ	5TH	789	800

Pandas Example Code

```
import matplotlib.pyplot as plt
import pandas as pd
myFile=pd.read_csv("C:\\Users\\CS Store\\Desktop\\fAI
Z\\ALLAH.csv")
print(myFile)
plt.bar(range(len(num_list)),num_list)
```

SciPy

Scipy is an open-source python library used for scientific computation, data computation, and high-performance computation. A large number of user-friendly routines are present in the library for easy computation. The package is built over the NumPy extension allowing the manipulation and visualization of the data with the availability of high-level commands. Along with the NumPy, Scipy is used for mathematical computation. NumPy allows the sorting, indexing of the array data, while the numerical code is stored in SciPy.

SciPy Example Code

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

def f(x):
    return x**2 + 10*np.sin(x)

x = np.arange(-10, 10, 0.1)
plt.plot(x, f(x))
```

SciPy Example Code

```
from scipy import optimize  
print(optimize.minimize(f, x0=0))
```

Scikit Learn

Scikit-learn is an open-source library under the Python programming environment used for machine learning approaches. It supports a wide range of supervised and unsupervised learning algorithms. The library contains popular algorithms along with the packages NumPy, Matplotlib, and SciPy. The famous application of Scikit-learn is in Spotify for music recommendations.

Scikit Learn

```
from sklearn.datasets import load_digits
digits = load_digits()

import numpy as np
import matplotlib.pyplot as plt
plt.figure(figsize=(20,4))
for index, (image, label) in
    enumerate(zip(digits.data[0:5], digits.target[0:5])):
    plt.subplot(1, 5, index + 1)
    plt.imshow(np.reshape(image, (8,8)),
        cmap=plt.cm.gray)
    plt.title('Training: %i\n' % label, fontsize = 20)
```

Seaborn

The package can be used for the visualization of the statistical models. The library is based on Matplotlib and allows the creation of statistical graphics through:

- Comparison of variables through an API based on datasets.
- Easy generation of complex visualization supporting multi-plot grids.
- Comparison of data subsets through Univariate and bivariate visualizations.
- Options of various color palettes to display the patterns.
- Automatic estimation of linear regression and its plotting.

TensorFlow

- TensorFlow is an open-source library used for high-performance numerical computation. It is also used in machine learning approaches and deep learning algorithms. Developed by the researchers of the Google Brain team within the Google AI organization, it is now widely used by researchers from math, physics, and machine learning for complex mathematical computations.

SymPy

The package is the closest to the Theano library and is used in all symbolic mathematics. With simple code provided by the package, the library can be effectively used for the computer algebra system. Written in python only, SymPy can be customized and applied in other applications. The source code of the package can be found in GitHub.

Stats models

Statsmodels is a python library useful in the analysis and estimation of statistical models. The library is incorporated to carry out the statistical tests, etc. providing high-performance outcomes.

PyBrains

- PyBrain is an open source-library from the available **libraries in python** used for Machine Learning algorithms for every entry-level student in research. The goal of PyBrain is to offer flexible and easy-to-use algorithms for machine learning tasks. It also provides predefined environments for comparing the algorithms. PyBrain stands for Python-Based Reinforcement Learning, Artificial Intelligence, and Neural Network Library. Compared to the other machine learning libraries provided by python, PyBrain is fast and easily understandable.

LaTeX in Python

PythonTeX is a LaTeX package that allows Python code in LaTeX documents to be executed and provides access to the output. This makes possible reproducible documents that combine results with the code required to generate them. Calculations and figures may be next to the code that created them.

LaTeX in Python

```
a = r'f(x) = \frac{\exp(-x^2/2)}{\sqrt{2\pi}}'  
ax = plt.axes([0,0,0.3,0.3]) #left,bottom,width,height  
ax.set_xticks([])  
ax.set_yticks([])  
ax.axis('off')  
plt.text(0.4,0.4,'$%s$' %a,size=50,color="green")
```

Addition and Subtraction of Matrix

Rule: Two matrix are confirmable for addition if they have same order.

For Example: $A = \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{bmatrix}$, and $B = A = \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{bmatrix}$

$$\begin{aligned} A+B &= \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{bmatrix} \\ &= \begin{bmatrix} 0+0 & 1+1 & 2+2 \\ 3+3 & 4+4 & 5+5 \end{bmatrix} \\ &= \begin{bmatrix} 0 & 2 & 4 \\ 6 & 8 & 10 \end{bmatrix} \end{aligned}$$

$$\begin{aligned} A-B &= \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{bmatrix} - \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{bmatrix} \\ &= \begin{bmatrix} 0-0 & 1-1 & 2-2 \\ 3-3 & 4-4 & 5-5 \end{bmatrix} \\ &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{aligned}$$

Multiplication of Matrix

Rule: Two matrices are confirmable for multiplication if the number of columns of A is equal to the number of rows of B .

For Example: $A = \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{bmatrix}$, and $B = \begin{bmatrix} 0 & 3 \\ 1 & 4 \\ 2 & 5 \end{bmatrix}$

$$AB = \begin{bmatrix} 0 \times 0 + 1 \times 1 + 2 \times 2 & 0 \times 3 + 1 \times 4 + 2 \times 5 \\ 3 \times 0 + 4 \times 1 + 5 \times 2 & 3 \times 3 + 4 \times 4 + 5 \times 5 \end{bmatrix}$$

$$= \begin{bmatrix} 0 + 1 + 4 & 0 + 4 + 10 \\ 0 + 4 + 10 & 9 + 16 + 10 \end{bmatrix}$$

$$= \begin{bmatrix} 5 & 14 \\ 14 & 35 \end{bmatrix}$$

Inverse of a Matrix

The inverse of a square matrix can be found by using the formula

$$A^{-1} = \frac{\text{Adj}A}{|A|}$$

For Example: $A = \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix}$, the formula to find the inverse of

a matrix is given by $A^{-1} = \frac{\text{Adj}A}{|A|}$

$$\text{Adj} = \begin{bmatrix} 4 & -2 \\ -2 & 1 \end{bmatrix} \text{ and}$$

$$|A| = \begin{vmatrix} 1 & 2 \\ 2 & 4 \end{vmatrix} = 1 \times 4 - 2 \times 2 = 4 - 4 = 0$$

So, inverse of this matrix doesn't exist. Because this is a singular matrix.

Now suppose,

$$B = \begin{bmatrix} 2 & 3 \\ 4 & 7 \end{bmatrix}, \text{ then } |B| = \begin{vmatrix} 2 & 3 \\ 4 & 7 \end{vmatrix}$$

$$= 2 \times 7 - 3 \times 4$$

$$= 14 - 12$$

$$= 2$$

$$\text{Adj}B = \begin{bmatrix} 7 & -3 \\ -4 & 2 \end{bmatrix}$$

$$A^{-1} = \frac{\text{Adj}A}{|A|}$$

$$A^{-1} = \frac{\begin{bmatrix} 7 & -3 \\ -4 & 2 \end{bmatrix}}{2}$$

$$2x - 2y = 4$$

$$3x + 2y = 6$$

Inverse Method:

We shall write the above equations in matrix form just like as,

$$\begin{bmatrix} 2 & -2 \\ 3 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 4 \\ 6 \end{bmatrix}$$

$$A \quad X = B$$

$$X = A^{-1}B$$

$$A^{-1} = \frac{\text{Adj}A}{|A|}$$

$$|A| = \begin{vmatrix} 2 & -2 \\ 3 & 2 \end{vmatrix} = 2 \times 2 - (-2) \times 3 = 4 + 6 = 10$$

$$\text{Adj}A = \begin{bmatrix} 2 & 2 \\ -3 & 2 \end{bmatrix}$$

$$X = A^{-1}B$$

$$\begin{aligned} \begin{bmatrix} x \\ y \end{bmatrix} &= \frac{1}{10} \begin{bmatrix} 2 & 2 \\ -3 & 2 \end{bmatrix} \begin{bmatrix} 4 \\ 6 \end{bmatrix} = \frac{1}{10} \begin{bmatrix} 2 & 2 \\ -3 & 2 \end{bmatrix} \begin{bmatrix} 4 \\ 6 \end{bmatrix} \\ &= \frac{1}{10} \begin{bmatrix} 2 \times 4 + 2 \times 6 \\ -3 \times 4 + 2 \times 6 \end{bmatrix} \\ &= \frac{1}{10} \begin{bmatrix} 8 + 12 \\ -12 + 12 \end{bmatrix} \end{aligned}$$

$$= \frac{1}{10} \begin{bmatrix} 20 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$$

$$x = 2 \quad \& \quad y = 0$$

Cramer's Method:

$$A = \begin{bmatrix} 2 & -2 \\ 3 & 2 \end{bmatrix}$$

$$|A| = \begin{vmatrix} 2 & -2 \\ 3 & 2 \end{vmatrix} = 2 \times 2 - (-2) \times 3 = 4 + 6 = 10$$

$$x = \frac{A_x}{|A|} = \frac{\begin{vmatrix} 4 & -2 \\ 6 & 2 \end{vmatrix}}{10} = \frac{4 \times 2 - (-2) \times 6}{10} = \frac{8 + 12}{10} = \frac{20}{10} = 2$$

$$y = \frac{A_y}{|A|} = \frac{\begin{vmatrix} 2 & 4 \\ 3 & 6 \end{vmatrix}}{10} = \frac{\begin{vmatrix} 2 & 4 \\ 3 & 6 \end{vmatrix}}{10} = \frac{6 \times 2 - 4 \times 3}{10} = \frac{12 - 12}{10} = \frac{0}{10} = 0$$

Example:

Vetor

[1 2 4 5]

Scalar

28

Matrix

$\begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix}$

$\begin{bmatrix} 1 \\ 2 \\ 4 \\ 5 \end{bmatrix}$

Tensor

$\begin{bmatrix} [1] \\ [2] \\ [1] \\ [2] \\ [6] \\ [7] \\ [9] \\ [3] \end{bmatrix}$

$\begin{bmatrix} [3] \\ [4] \\ [5] \\ [6] \\ [0] \\ [9] \\ [4] \\ [6] \end{bmatrix}$

Addition of Vectors

Vectors can be added and subtracted like

$v_1 = (1, 2, 4)$ and $v_2 = (4, 5, 6)$ then,

$$v = v_1 + v_2$$

$$= (1, 2, 4) + (4, 5, 6)$$

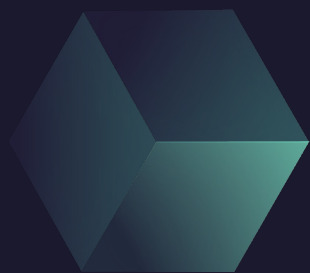
$$= (1 + 4, 2 + 5, 4 + 6)$$

$$= (5, 7, 10)$$

Subtraction of Vectors

$$\begin{aligned} v &= v_1 - v_2 = (1, 2, 4) - (4, 5, 6) \\ &= (1 - 4, 2 - 5, 4 - 6) \\ &= (-3, -3, -2) \end{aligned}$$

NumPy Array



```
In [12]: import numpy as np
```

```
In [15]: np.array([5, 8, 2, 9, 4])
```

```
Out[15]: array([5, 8, 2, 9, 4])
```

```
In [18]: np.array([4, 5, 6, 8], dtype='float32')
```

```
Out[18]: array([4., 5., 6., 8.], dtype=float32)
```

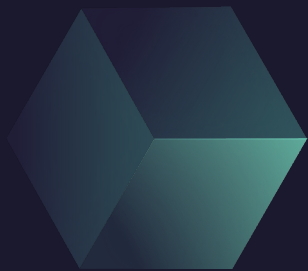
```
In [25]: np.zeros(12, dtype=int)
```

```
Out[25]: array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

```
In [26]: np.zeros(7, dtype=int)
```

```
Out[26]: array([0, 0, 0, 0, 0, 0, 0])
```

NumPy Array Creating 0 and 1 Vectors



```
Out[25]: array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

```
In [26]: np.zeros(7, dtype=int)
```

```
Out[26]: array([0, 0, 0, 0, 0, 0, 0])
```

```
In [27]: np.ones((5, 7), dtype=float)
```

```
Out[27]: array([[1., 1., 1., 1., 1., 1., 1.],  
                [1., 1., 1., 1., 1., 1., 1.],  
                [1., 1., 1., 1., 1., 1., 1.],  
                [1., 1., 1., 1., 1., 1., 1.],  
                [1., 1., 1., 1., 1., 1., 1.]])
```

```
In [28]: np.ones((2,3), dtype=float)
```

```
Out[28]: array([[1., 1., 1.],  
                [1., 1., 1.]])
```

NumPy Array Creating 0 and 1 Vectors

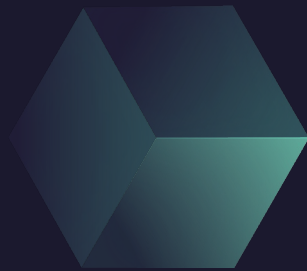
```
jupyter NumPy 1 Last Checkpoint: 18 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Not Connected Not Trusted Python 3 (ipykernel)

In [28]: np.ones((2,3), dtype=float)
Out[28]: array([[1., 1., 1.],
               [1., 1., 1.]])

In [29]: np.ones((11,13), dtype=float)
Out[29]: array([[1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
               [1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
               [1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
               [1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
               [1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
               [1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
               [1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
               [1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
               [1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
               [1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
               [1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.]])

In [30]: np.ones((50,50), dtype=float)
Out[30]: array([[1., 1., 1., ..., 1., 1., 1.],
               [1., 1., 1., ..., 1., 1., 1.],
               [1., 1., 1., ..., 1., 1., 1.],
               ...,
               [1., 1., 1., ..., 1., 1., 1.],
               [1., 1., 1., ..., 1., 1., 1.],
               [1., 1., 1., ..., 1., 1., 1.]])
```

Arrangement of Numbers Using NumPy



```
In [32]: np.full((7,9), 4.51)
```

```
Out[32]: array([[4.51, 4.51, 4.51, 4.51, 4.51, 4.51, 4.51, 4.51, 4.51],  
               [4.51, 4.51, 4.51, 4.51, 4.51, 4.51, 4.51, 4.51, 4.51],  
               [4.51, 4.51, 4.51, 4.51, 4.51, 4.51, 4.51, 4.51, 4.51],  
               [4.51, 4.51, 4.51, 4.51, 4.51, 4.51, 4.51, 4.51, 4.51],  
               [4.51, 4.51, 4.51, 4.51, 4.51, 4.51, 4.51, 4.51, 4.51],  
               [4.51, 4.51, 4.51, 4.51, 4.51, 4.51, 4.51, 4.51, 4.51],  
               [4.51, 4.51, 4.51, 4.51, 4.51, 4.51, 4.51, 4.51, 4.51]])
```

```
In [33]: np.arange(0, 10, 2)
```

```
Out[33]: array([0, 2, 4, 6, 8])
```

```
In [35]: np.arange(0, 20, 2)
```

```
Out[35]: array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18])
```

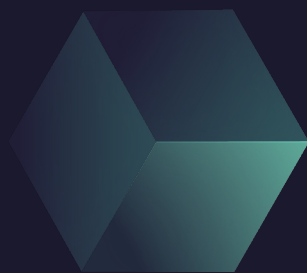
```
In [37]: np.arange(0, 30, 2)
```

```
Out[37]: array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28])
```

```
In [38]: np.arange(0, 40, 2)
```

```
Out[38]: array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38])
```

Arrangement of Numbers Using NumPy



```
In [39]: np.arange(0, 50, 2)
```

```
Out[39]: array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18, 20, 22, 24, 26,
                34, 36, 38, 40, 42, 44, 46, 48])
```

```
In [41]: np.arange(0, 10, 3)
```

```
Out[41]: array([0, 3, 6, 9])
```

```
In [42]: np.arange(0, 10, 5)
```

```
Out[42]: array([0, 5])
```

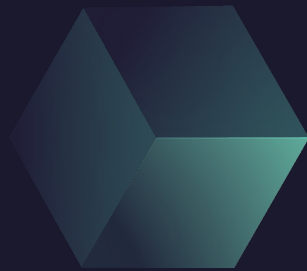
```
In [44]: np.linspace(0, 5, 5)
```

```
Out[44]: array([0. , 1.25, 2.5 , 3.75, 5.  ])
```

```
In [45]: np.linspace(0, 2, 10)
```

```
Out[45]: array([0.          , 0.22222222, 0.44444444, 0.66666667, 0.88888
                1.11111111, 1.33333333, 1.55555556, 1.77777778, 2.  ])
```

Random Arrangement of Numbers Using NumPy



```
In [44]: np.linspace(0, 5, 5)
```

```
Out[44]: array([0. , 1.25, 2.5 , 3.75, 5.  ])
```

```
In [45]: np.linspace(0, 2, 10)
```

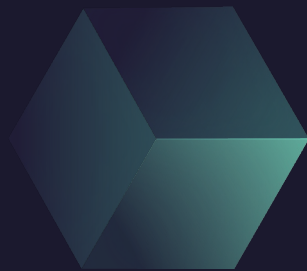
```
Out[45]: array([0.          , 0.22222222, 0.44444444, 0.66666667, 0.88888889, 1.11111111, 1.33333333, 1.55555556, 1.77777778, 2.0])
```

```
In [46]: np.linspace(0, 3, 10)
```

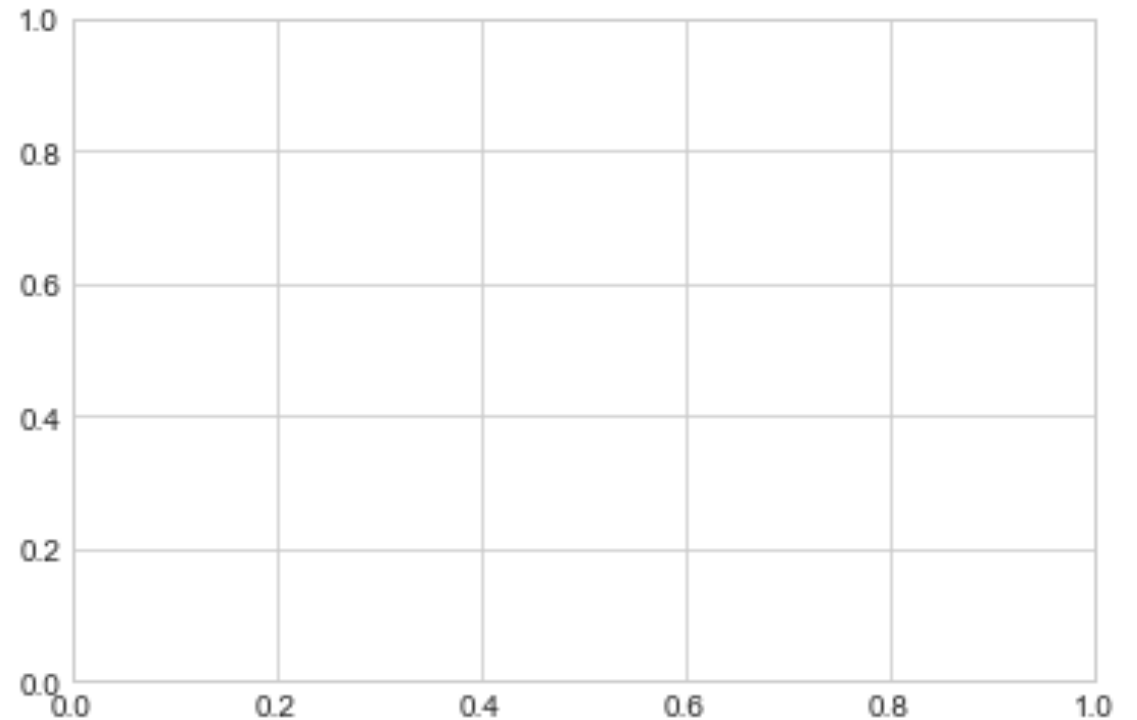
```
Out[46]: array([0.          , 0.33333333, 0.66666667, 1.0, 1.33333333, 1.66666667, 2.0, 2.33333333, 2.66666667, 3.0])
```

```
In [ ]: np.random.random((3, 3))
```

Matplotlib Grid Drawing



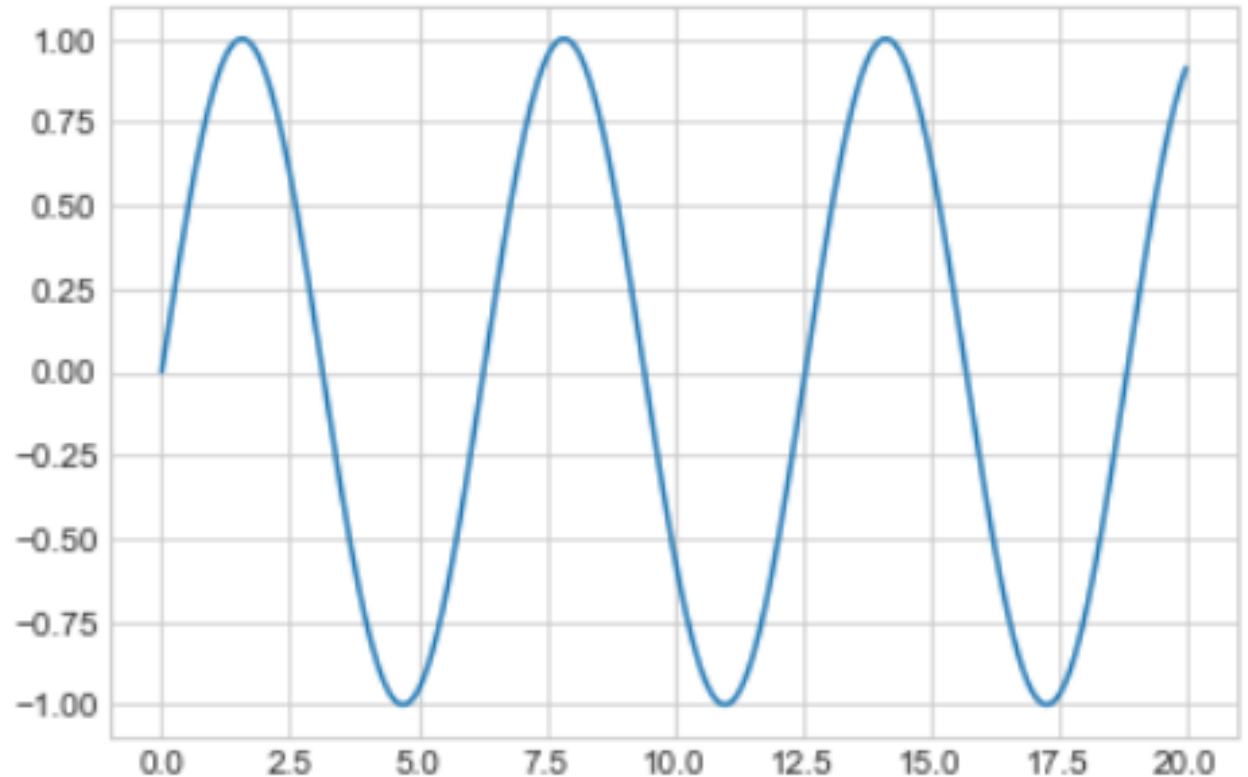
```
In [52]: import matplotlib.pyplot as plt  
plt.style.use('seaborn-whitegrid')  
import numpy as np  
fig = plt.figure()  
ax = plt.axes()
```



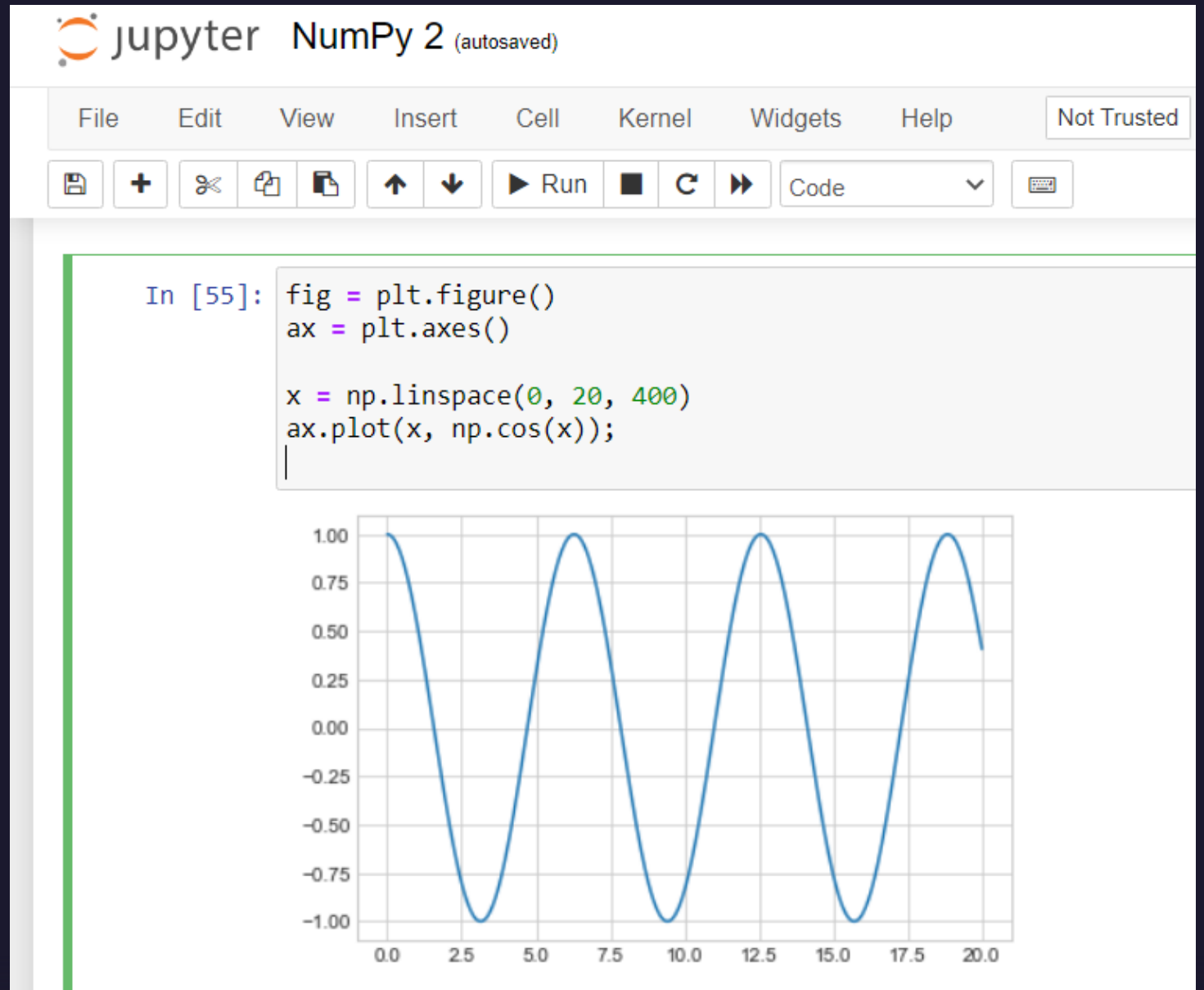
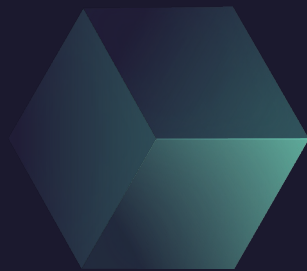
Trigonometric Drawing



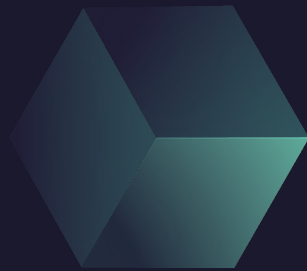
```
4]: fig = plt.figure()  
    ax = plt.axes()  
  
    x = np.linspace(0, 20, 500)  
    ax.plot(x, np.sin(x));
```



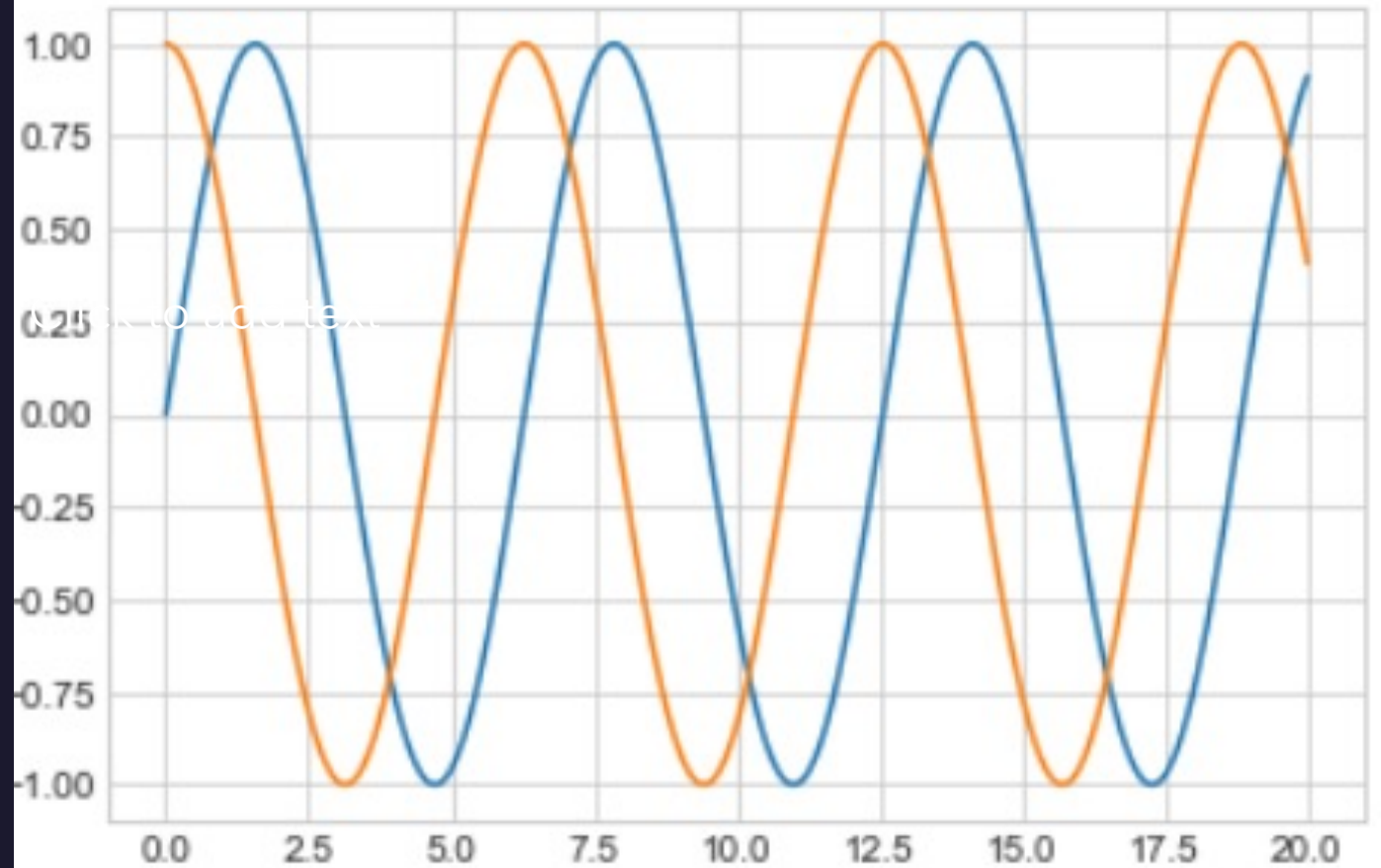
Trigonometric Drawing 2



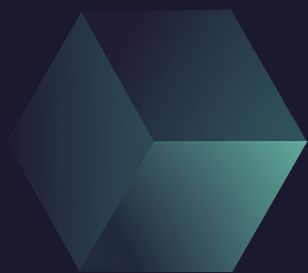
Trigonometric Drawing 3



```
plt.plot(x, np.sin(x))  
plt.plot(x, np.cos(x));
```

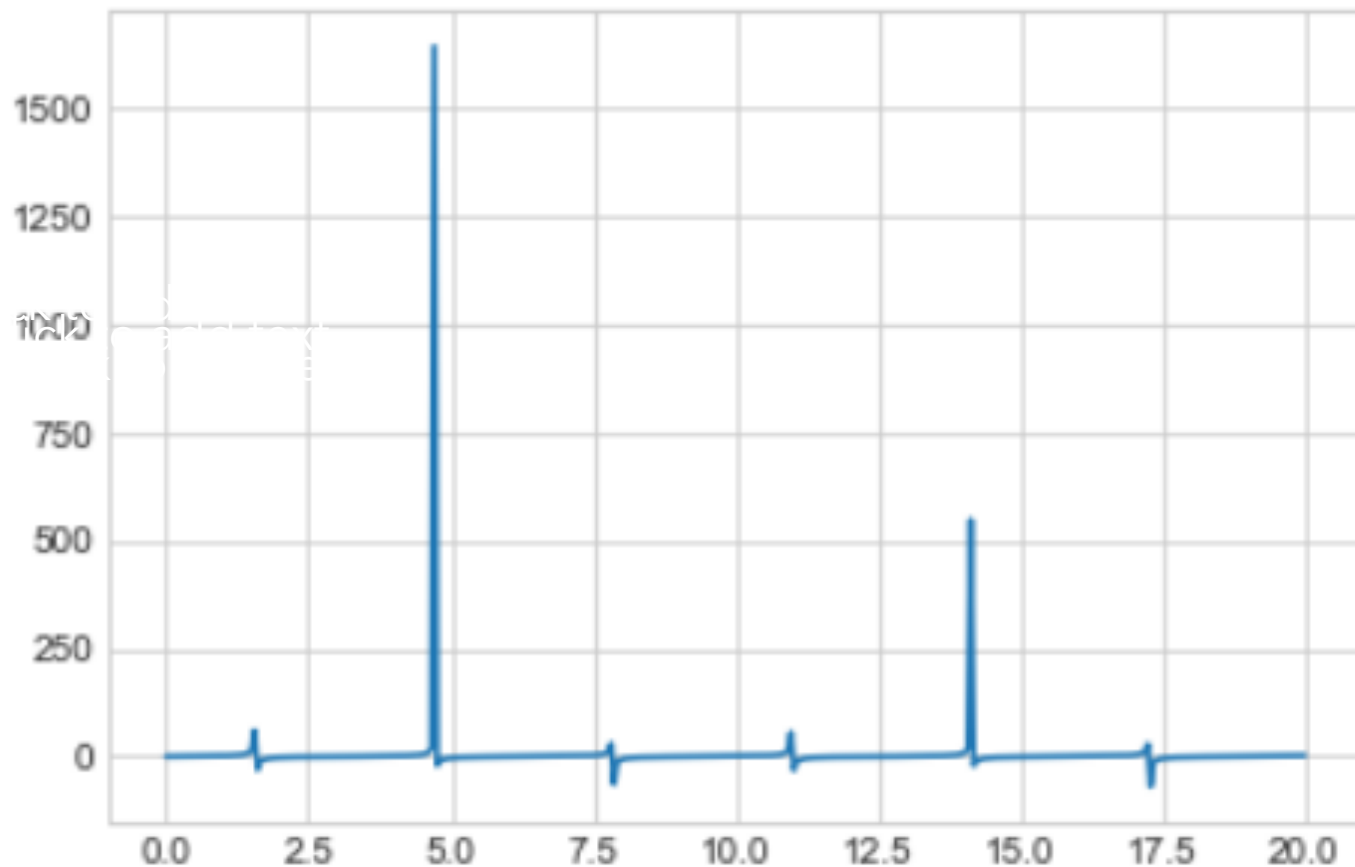


Trigonometric Drawing 4

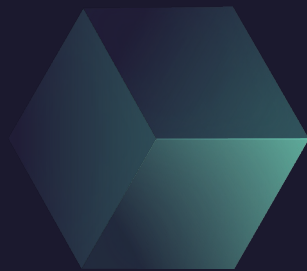


```
plt.plot(x,np.tan(x))
```

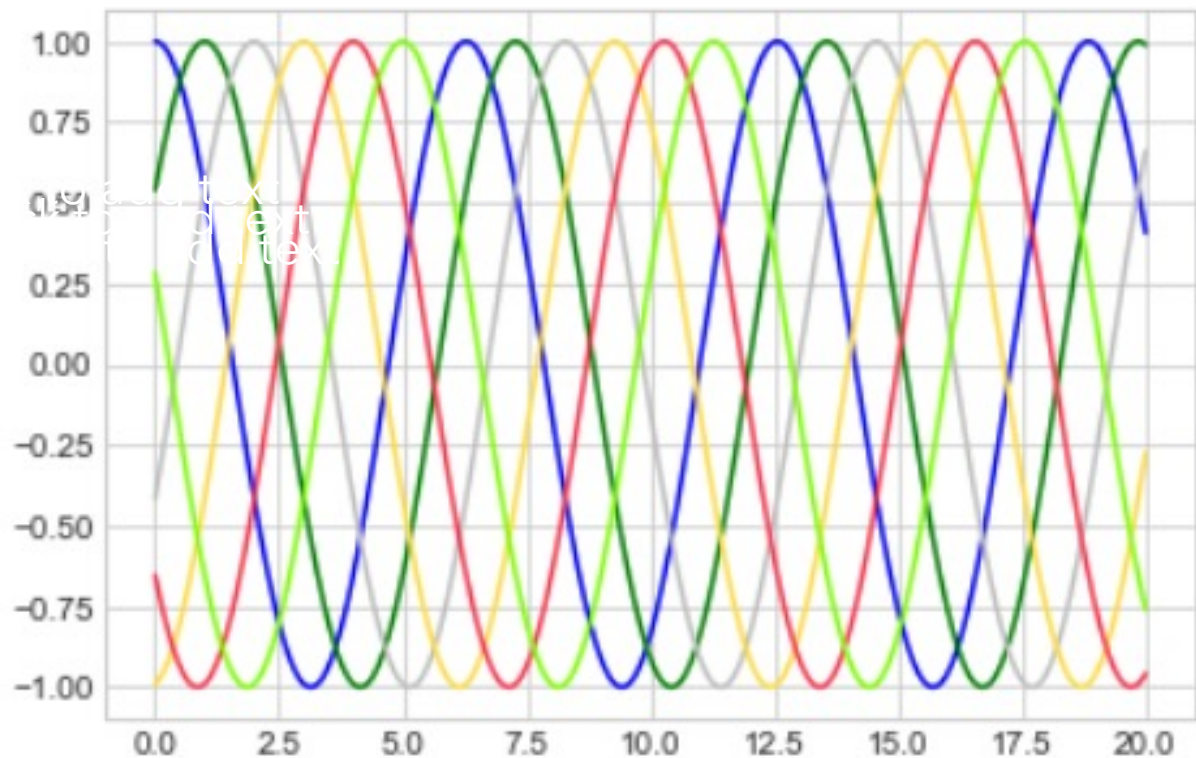
```
[<matplotlib.lines.Line2D at 0x1f033460550>]
```



Trigonometric Drawing 5

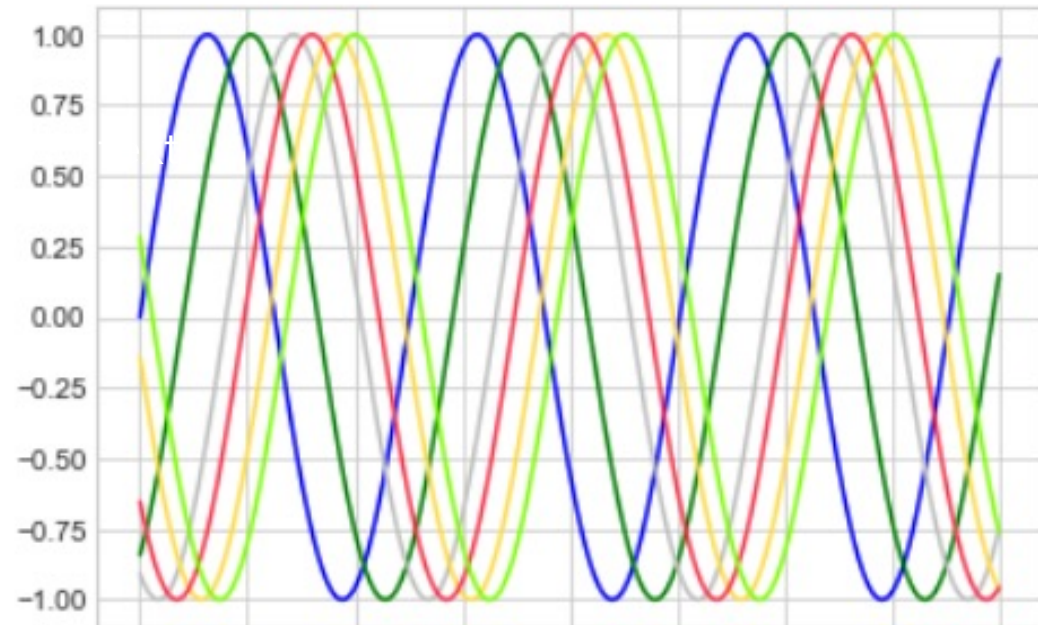


```
] : plt.plot(x, np.cos(x - 0), color='blue')  
plt.plot(x, np.cos(x - 1), color='g')  
plt.plot(x, np.cos(x - 2), color='0.75')  
plt.plot(x, np.cos(x - 3), color='#FFDD44')  
plt.plot(x, np.cos(x - 4), color=(1.0,0.2,0.3))  
plt.plot(x, np.cos(x - 5), color='chartreuse');
```



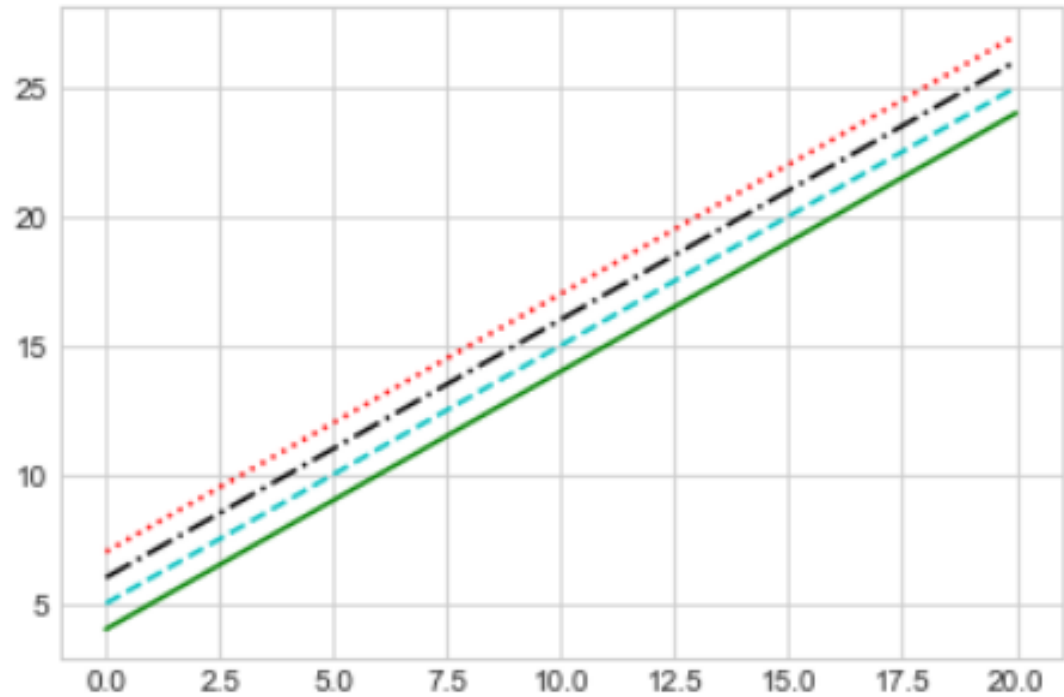
Trigonometric Drawing

```
In [62]: plt.plot(x, np.sin(x - 0), color='blue')  
plt.plot(x, np.sin(x - 1), color='g')  
plt.plot(x, np.sin(x - 2), color='0.75')  
plt.plot(x, np.sin(x - 3), color='#FFDD44')  
plt.plot(x, np.cos(x - 4), color=(1.0,0.2,0.3))  
plt.plot(x, np.cos(x - 5), color='chartreuse'); #
```



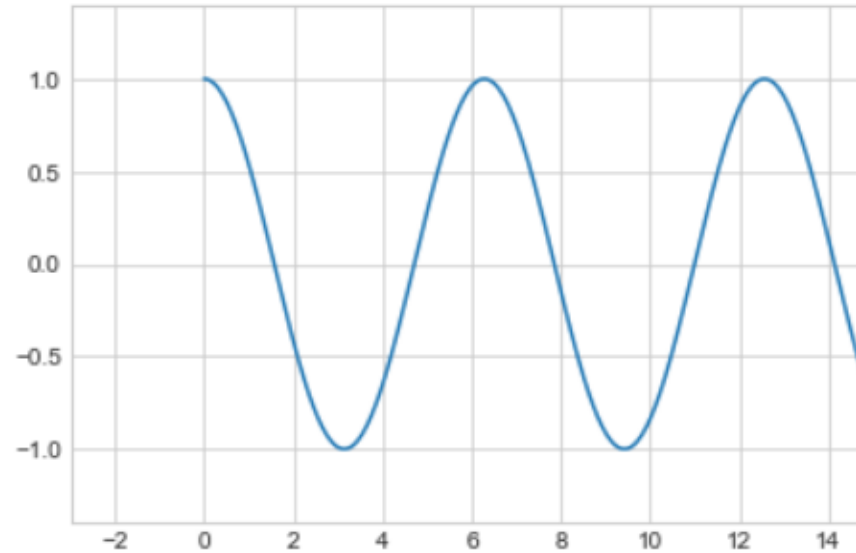
Matplotlib Plotting 1

```
In [65]: plt.plot(x, x + 4, '-g')  
plt.plot(x, x + 5, '--c')  
plt.plot(x, x + 6, '-.k')  
plt.plot(x, x + 7, ':r');
```

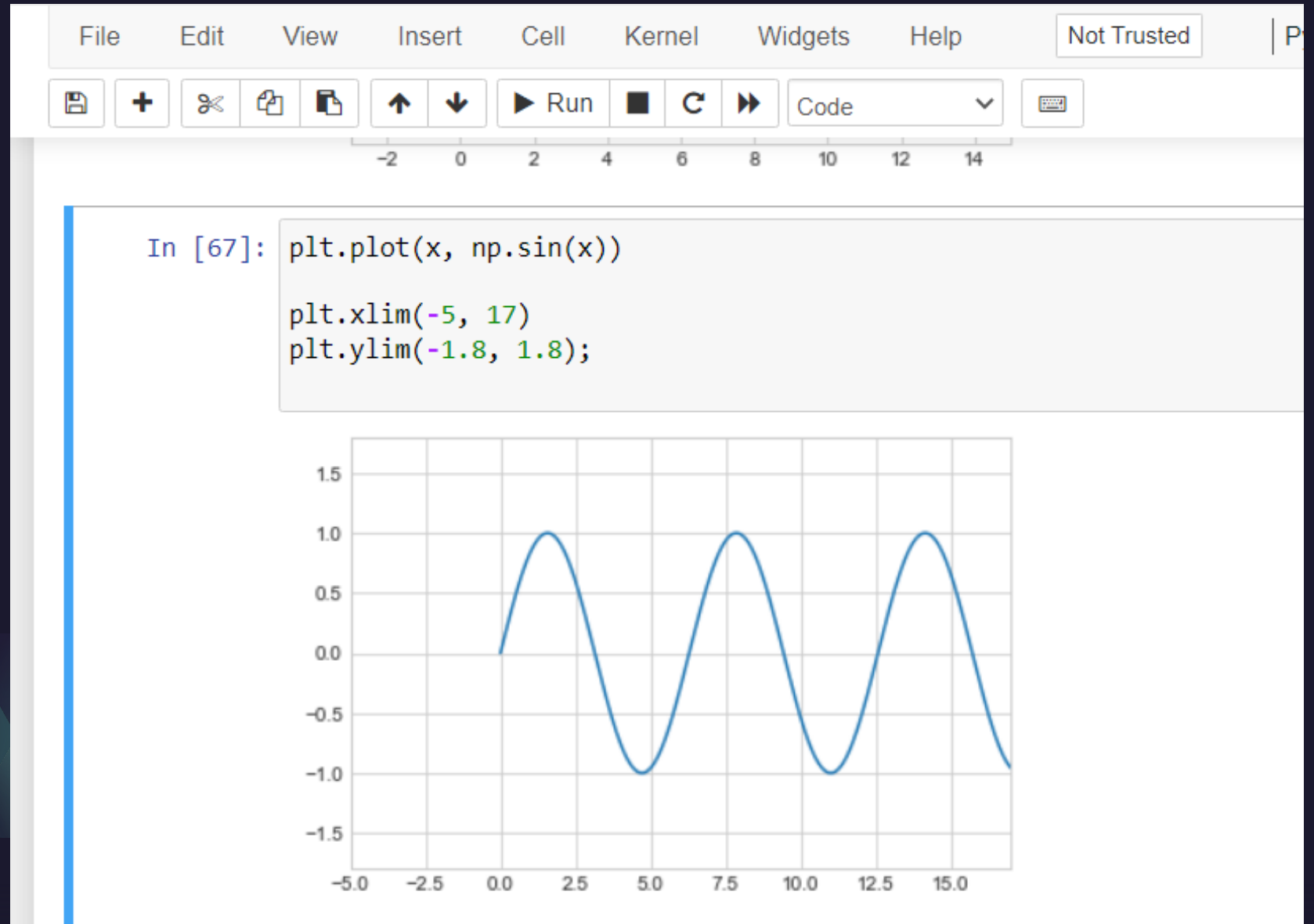


Matplotlib Plotting 2

```
In [66]: plt.plot(x, np.cos(x))  
  
plt.xlim(-3, 15)  
plt.ylim(-1.4, 1.4);
```

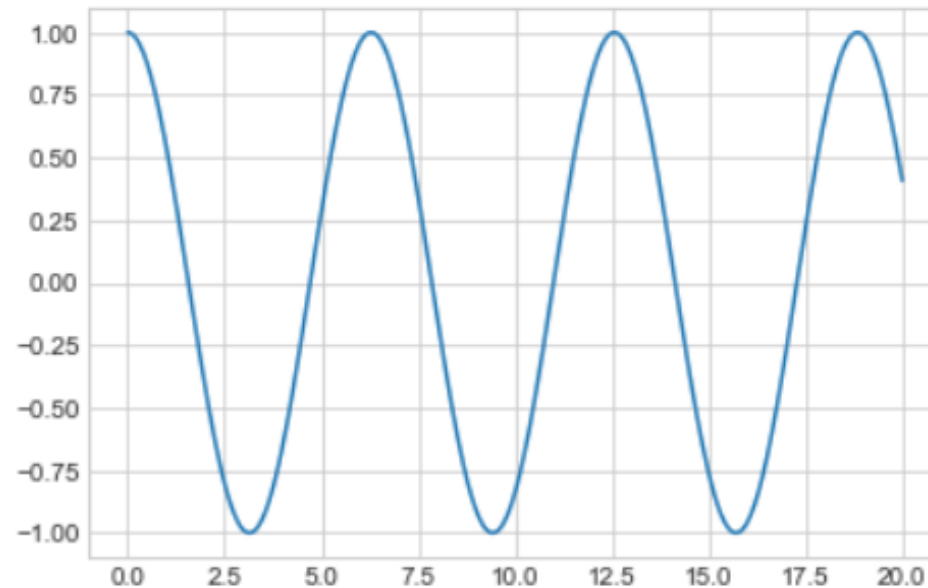


Matplotlib Plotting 3



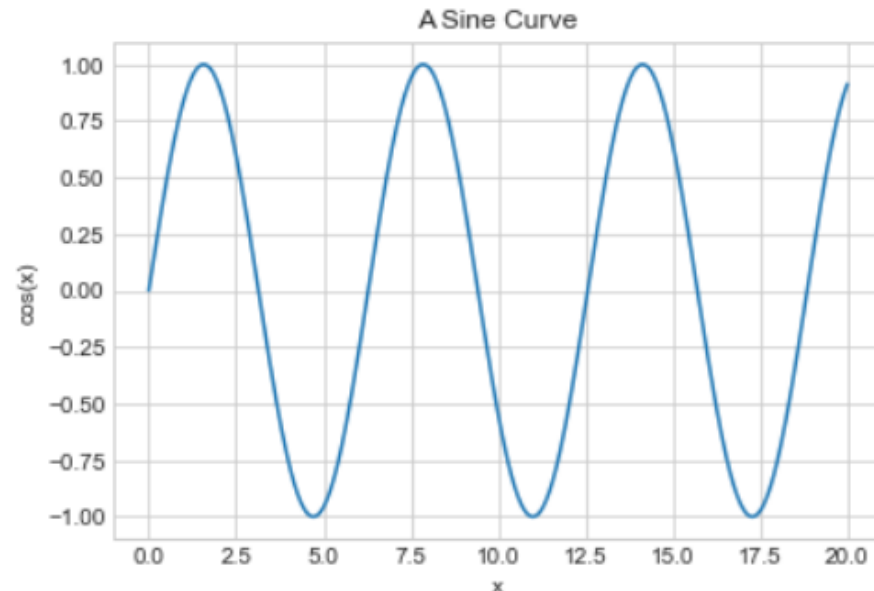
Matplotlib Plotting 4

```
In [55]: fig = plt.figure()  
         ax = plt.axes()  
  
         x = np.linspace(0, 20, 400)  
         ax.plot(x, np.cos(x));  
         |
```



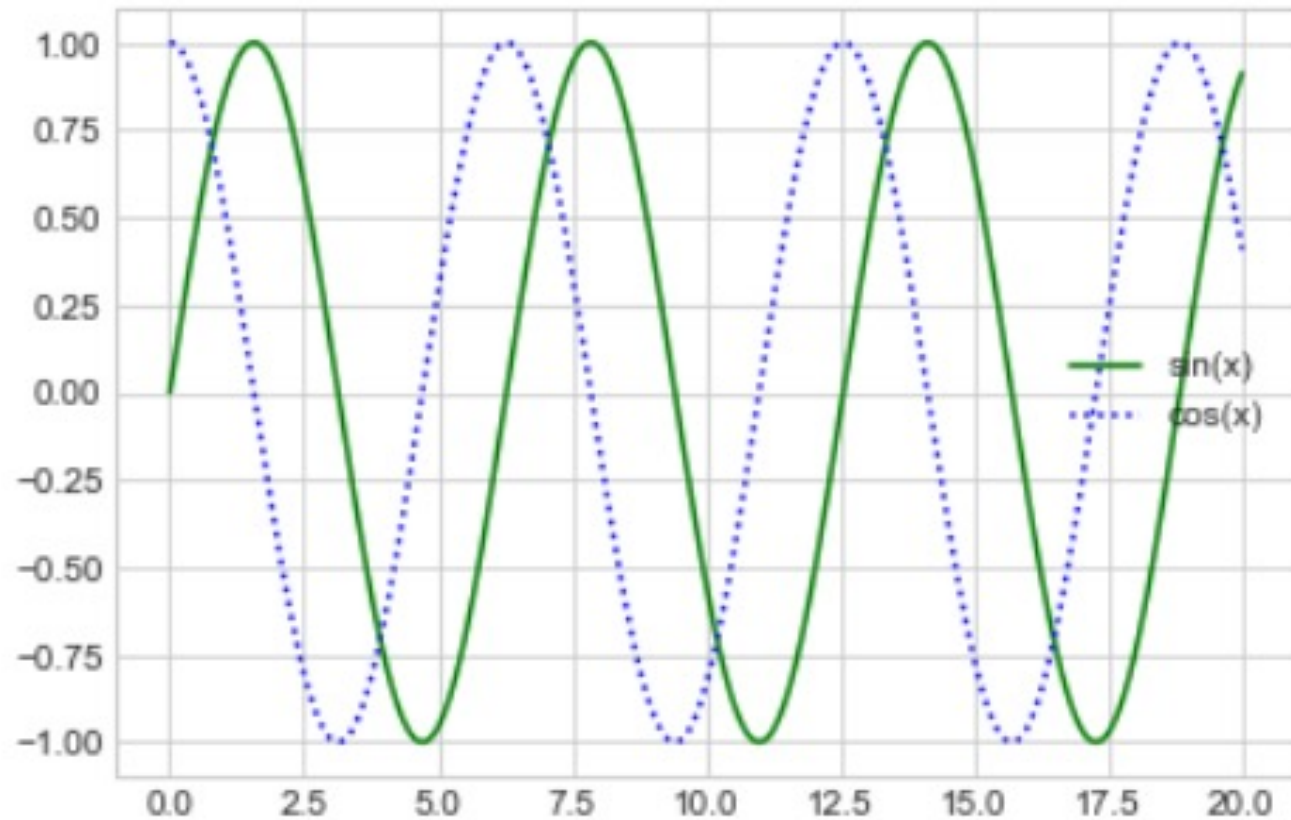
Matplotlib Plotting 5

```
In [69]: plt.plot(x, np.sin(x))  
plt.title("A Sine Curve")  
plt.xlabel("x")  
plt.ylabel("cos(x)");
```



Matplotlib Plotting 6

```
[71]: plt.plot(x, np.sin(x), '-g', label='sin(x)')  
      plt.plot(x, np.cos(x), ':b', label='cos(x)')  
      plt.axis()  
  
      plt.legend();
```



Matplotlib

Plotting 7

```
In [ ]: import matplotlib.pyplot as plt
plt.style.use('classic')
import numpy as np
import pandas as pd
```

Pandas Data Analysis 1

```
In [77]: rng = np.random.RandomState(0)
x = np.linspace(0, 20, 1000)
y = np.cumsum(rng.randn(1000, 20), 0)
rng = np.random.RandomState(0)
x = np.linspace(0, 20, 1000)
y = np.cumsum(rng.randn(1000, ), 0)
```



Pandas Data Analysis 2

```
In [84]: import seaborn as sns
sns.set()
plt.plot(x, y)
plt.legend('I', ncol=2, loc='upper left');
```



Pandas

Data

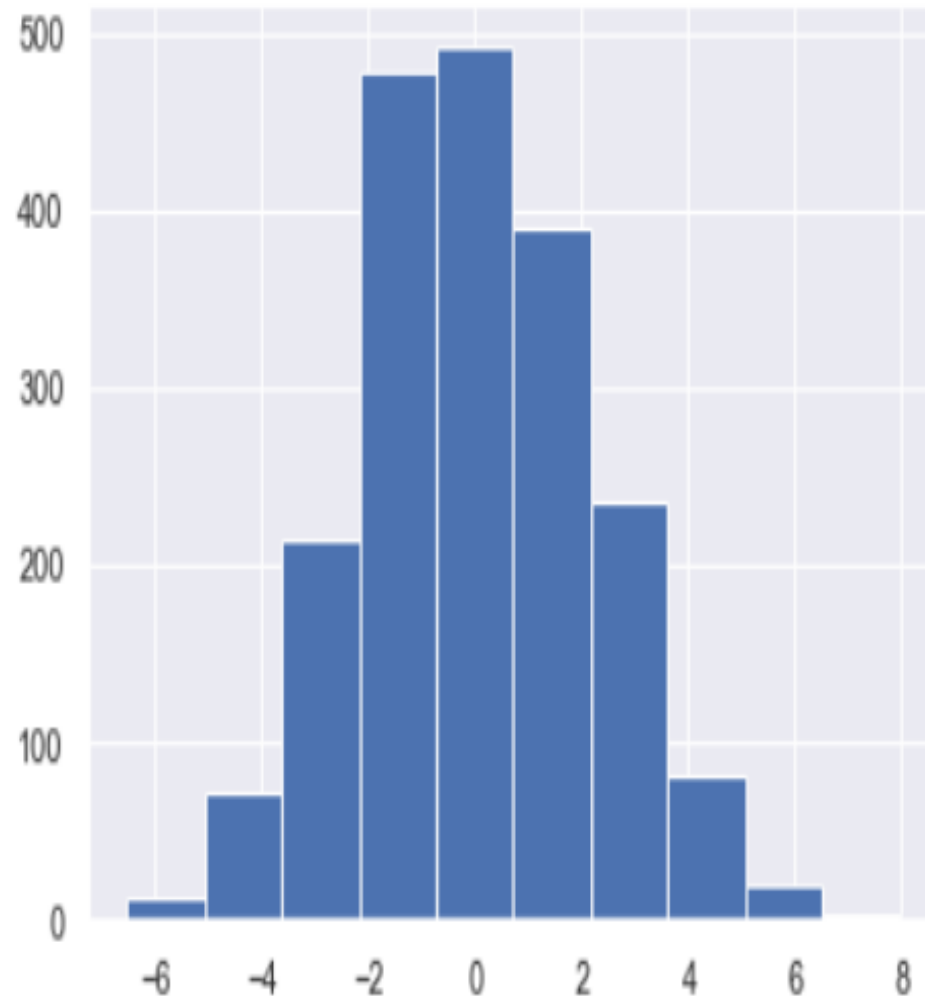
Analysis 3

```
In [98]: import pandas as pd
SAM = np.random.multivariate_normal([0, 0], [[5, 2], [2, 2]], size=2000)

SAM = pd.DataFrame(SAM, columns=['x', 'y'])

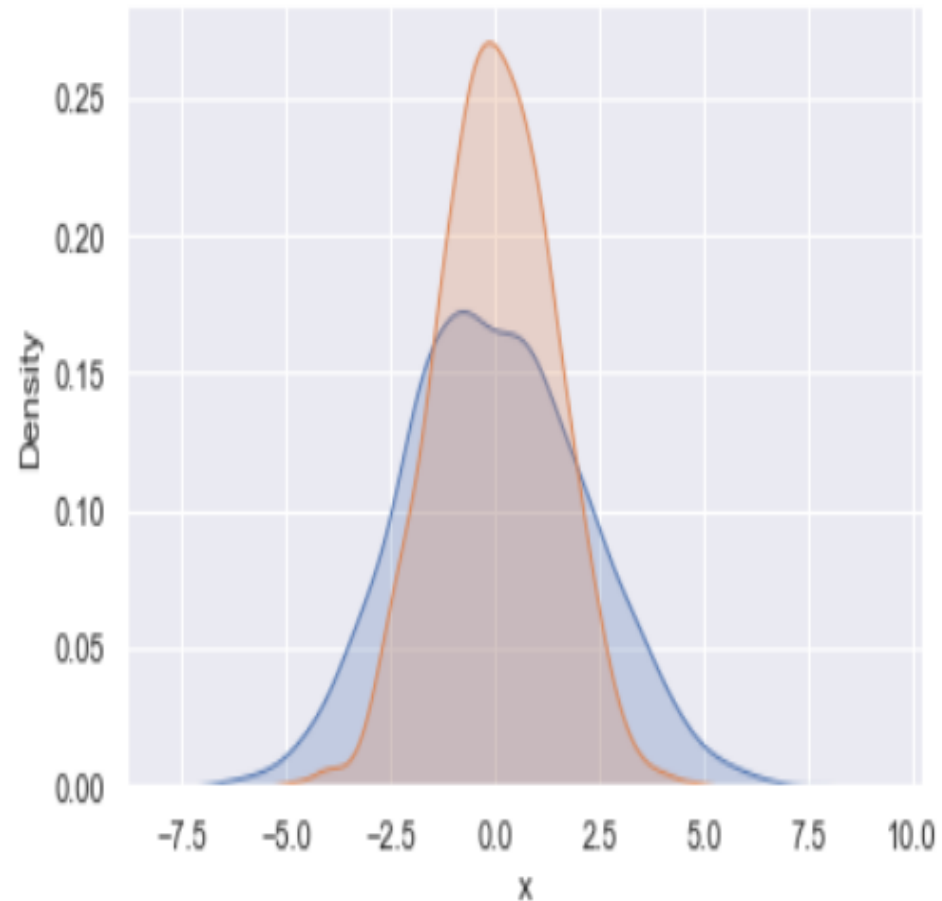
for col in 'xy': plt.hist(SAM[col], normed=True, alpha=0.5)
```

Pandas Data Analysis 3



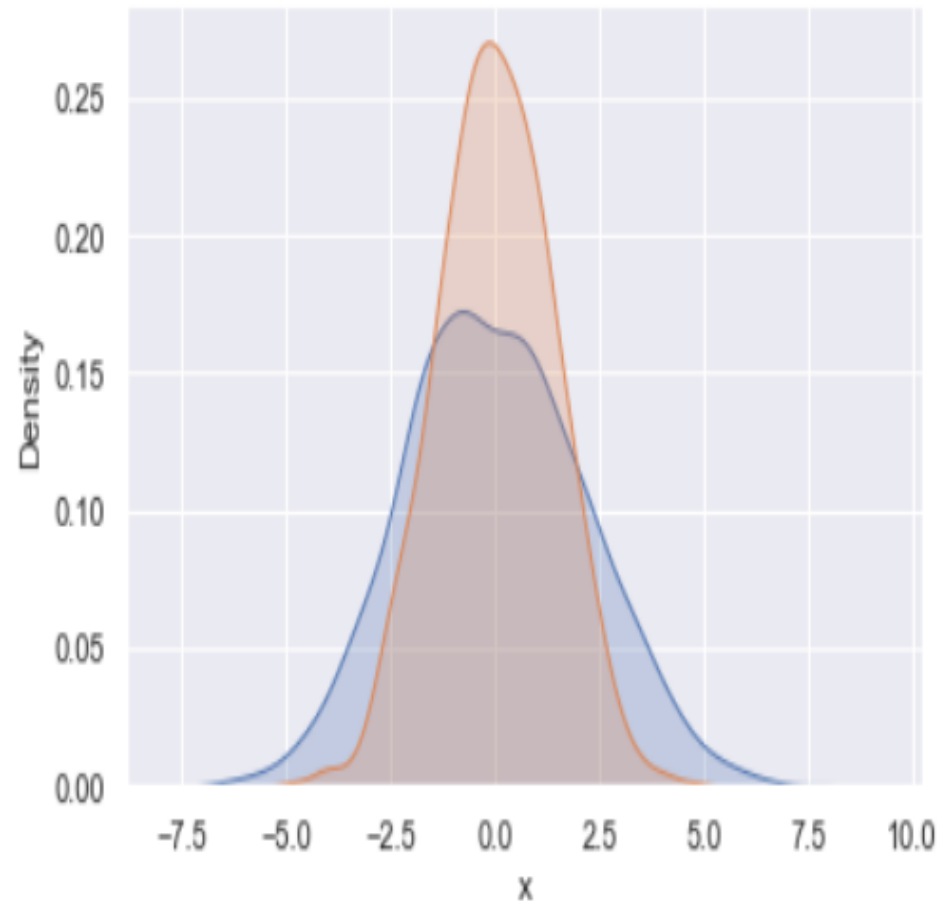
Pandas Data Analysis 4

```
In [100]: for col in 'xy':  
           sns.kdeplot(SAM[col], shade=True)
```



Pandas Data Analysis 4

```
In [100]: for col in 'xy':  
           sns.kdeplot(SAM[col], shade=True)
```





Pandas Data Analysis 4

```
In [102]: sns.distplot(SAM['x'])  
sns.distplot(SAM['y']);
```

Pandas Data Analysis 4

