matplotlib

December 22, 2022

1 Matplotlib

- matplotlib + numpy is more compatiable / best combination for scientic compution/ analysis

- Line Plot
- Histogram
- Barplot
- Scatter Plot
- 3D Plot
- Save the plot/visualization

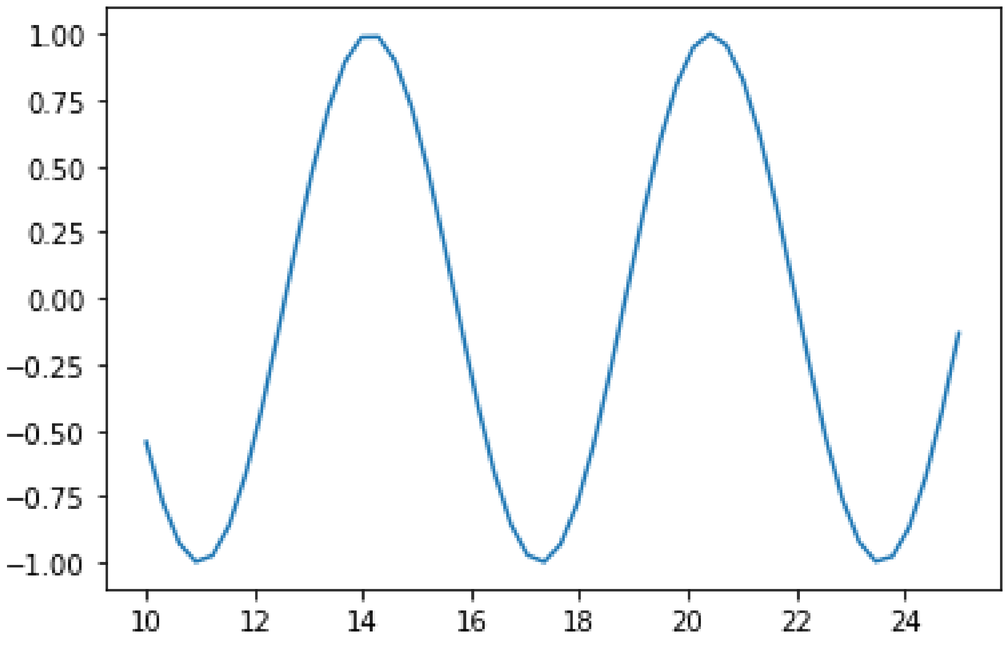
1.0.1 Import Library

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

[2]:
```
x = np.linspace(10, 25)
y = np.sin(x)    # sin of x, it will convert the result of x to sin, after␣
 ↪conversion we will get actual y
```

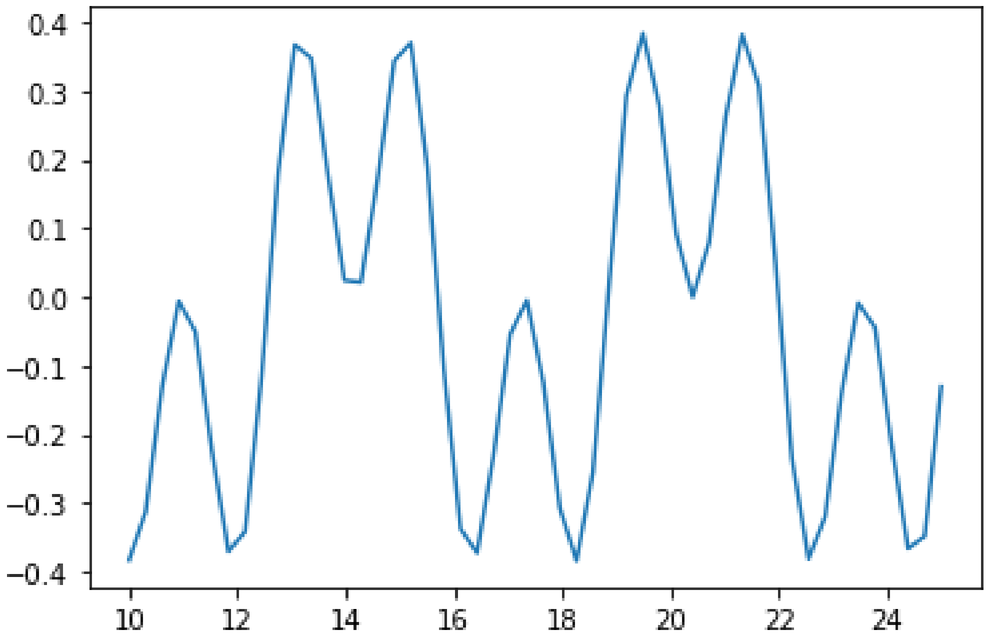
2 Line plot

[3]:
```
plt.plot(x, y)  # here plot is function, by default is line plot, we can␣
 ↪specify other also
plt.show()    # its recommended to show the visualization, show is kind of␣
 ↪print function for visualiation
```



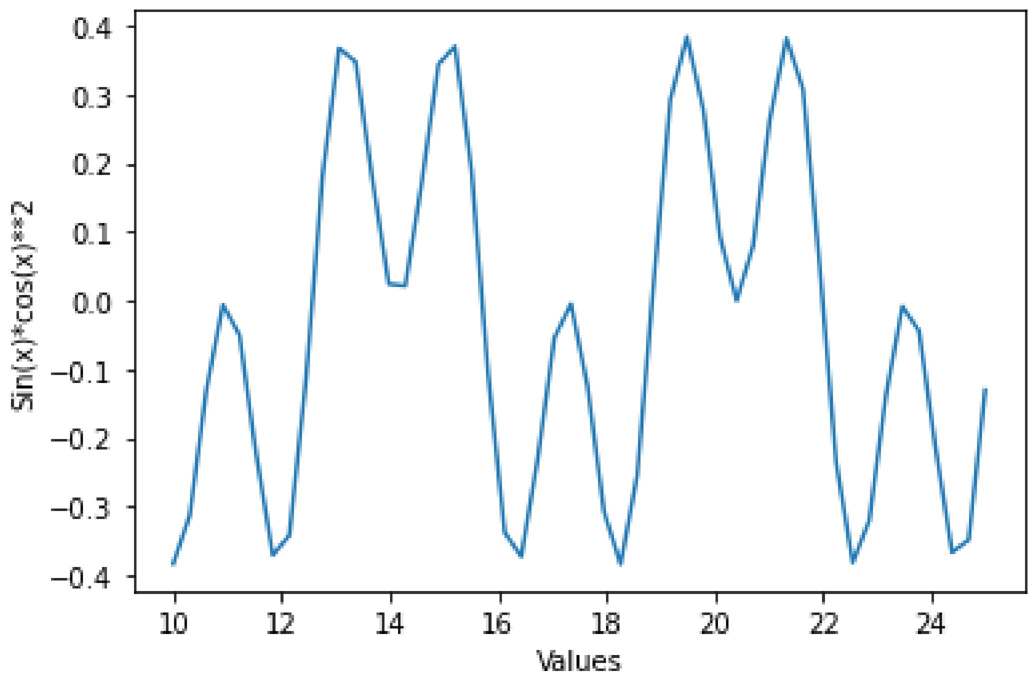
```
[4]: # Visualization 2
     x = np.linspace(10, 25)
     y = np.sin(x) * np.cos(x) ** 2
```

```
[5]: plt.plot(x, y)  # plot x over y; there are two axis (x & y)
     plt.show()
```

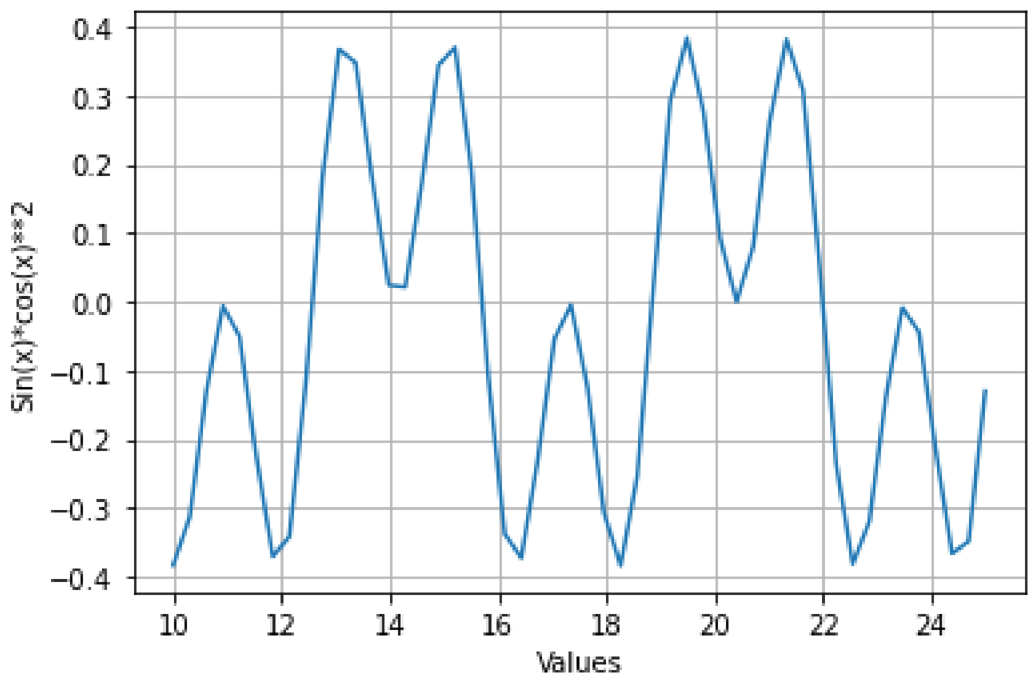


2.0.1 Adding Label

[6]:
```
# Label takes str inside function
plt.plot(x, y)
plt.xlabel("Values")
plt.ylabel ('Sin(x)*cos(x)**2')
plt.show()
```

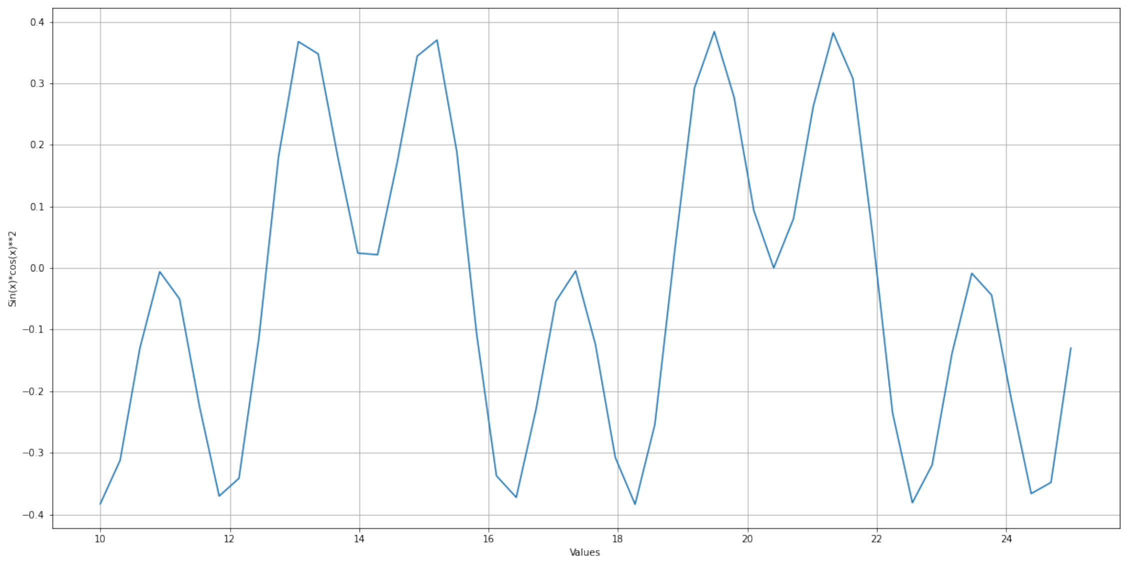


[7]:
```
# Grid - default is false
plt.plot(x, y)
plt.xlabel("Values")
plt.ylabel ('Sin(x)*cos(x)**2')
plt.grid(True)
plt.show()
```



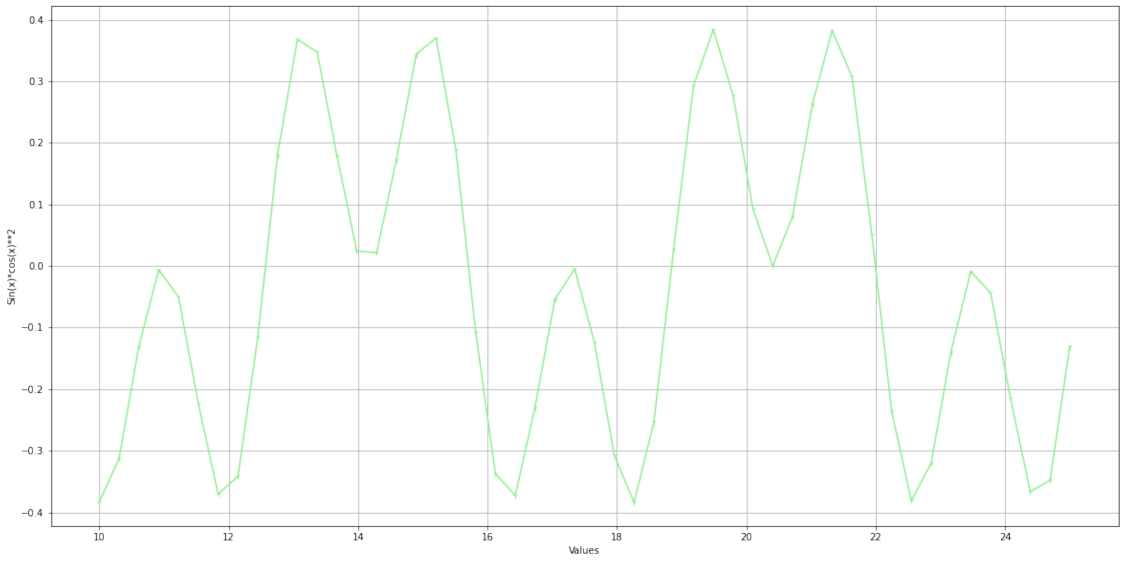
2.0.2 Adding Figure Size

[8]:
```
plt.figure(1, figsize=(20, 10))  # 1: Number of viz, 20: width, 10: height
plt.plot(x, y)
plt.xlabel ("Values")
plt.ylabel ('Sin(x)*cos(x)**2')
plt.grid(True)
plt.show()
```

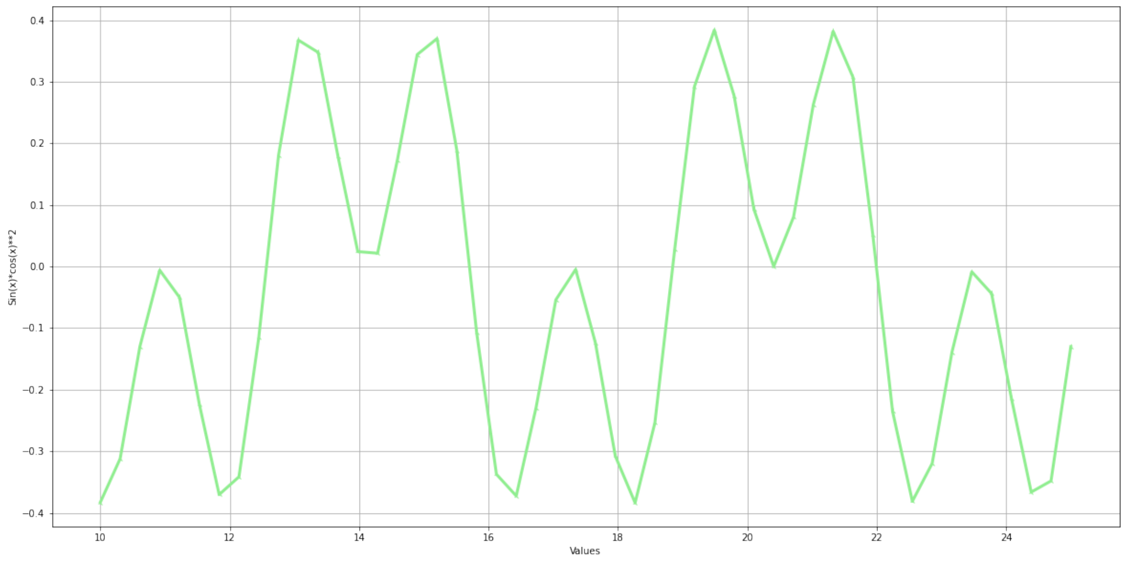


2.0.3 Colour & Marker

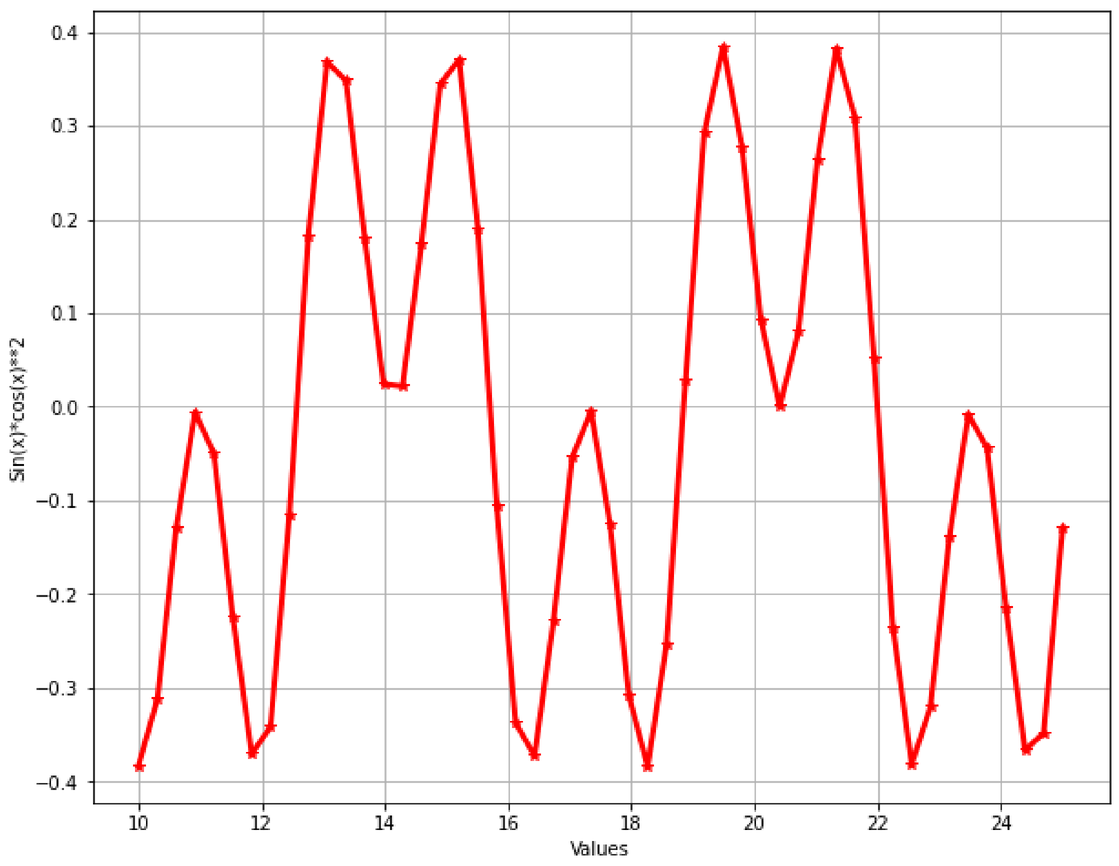
[9]:
```
plt.figure(1, figsize=(20, 10))
plt.plot(x, y, color='lightgreen', marker='1')
plt.xlabel ("Values")
plt.ylabel ('Sin(x)*cos(x)**2')
plt.grid(True)
plt.show()
```



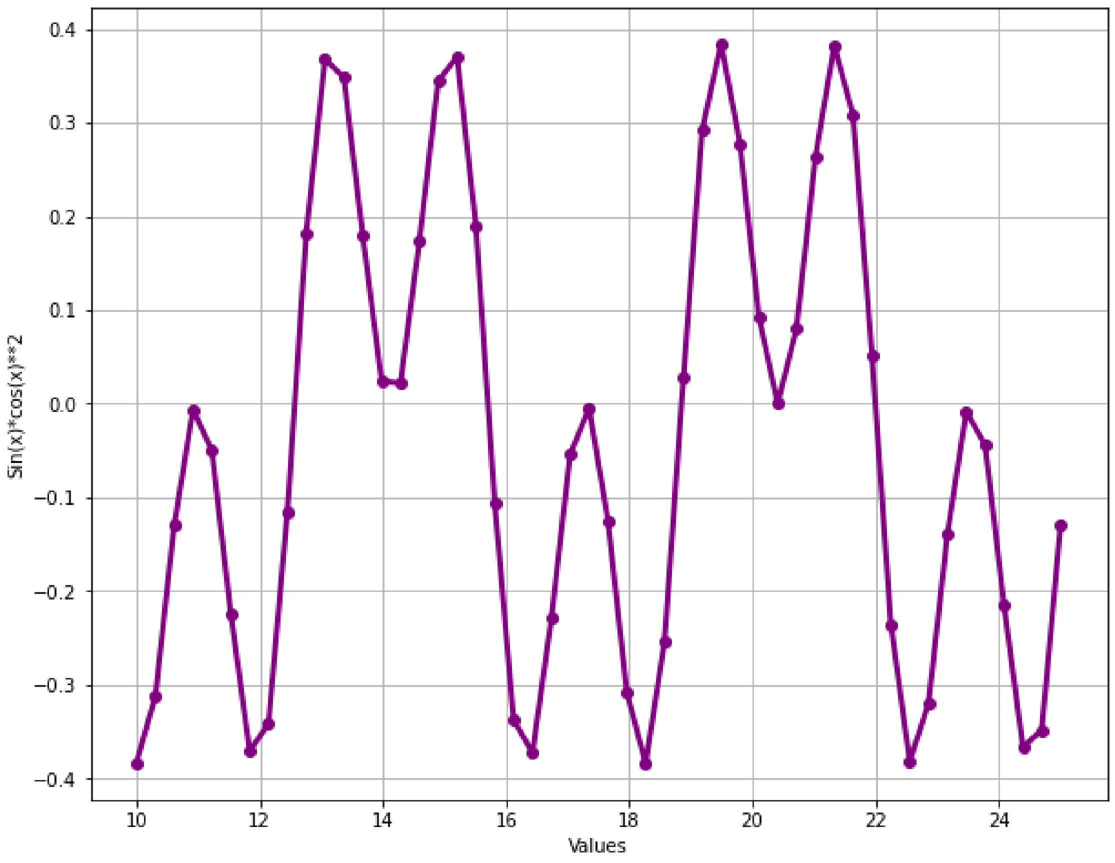
[10]:
```
plt.figure(1, figsize=(20, 10))
plt.plot(x, y, color='lightgreen', marker='2', linewidth=3)
plt.xlabel ("Values")
plt.ylabel ('Sin(x)*cos(x)**2')
plt.grid(True)
plt.show()
```



```
[11]: plt.figure(1, figsize=(10, 8))
      plt.plot(x, y, color='red', marker='*', linewidth=3)
      plt.xlabel ("Values")
      plt.ylabel ('Sin(x)*cos(x)**2')
      plt.grid(True)
      plt.show()
```

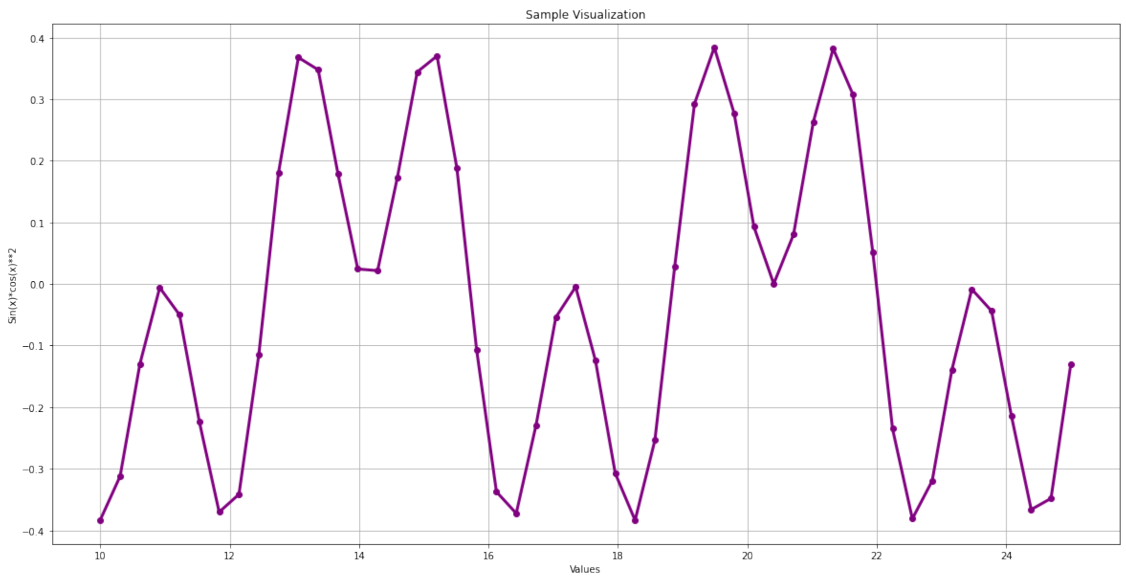


```
[12]: plt.figure(1, figsize=(10, 8))
      plt.plot(x, y, color='purple', marker='o', linewidth=3)  # marker is circle/ o
      plt.xlabel ("Values")
      plt.ylabel ('Sin(x)*cos(x)**2')
      plt.grid(True)
      plt.show()
```



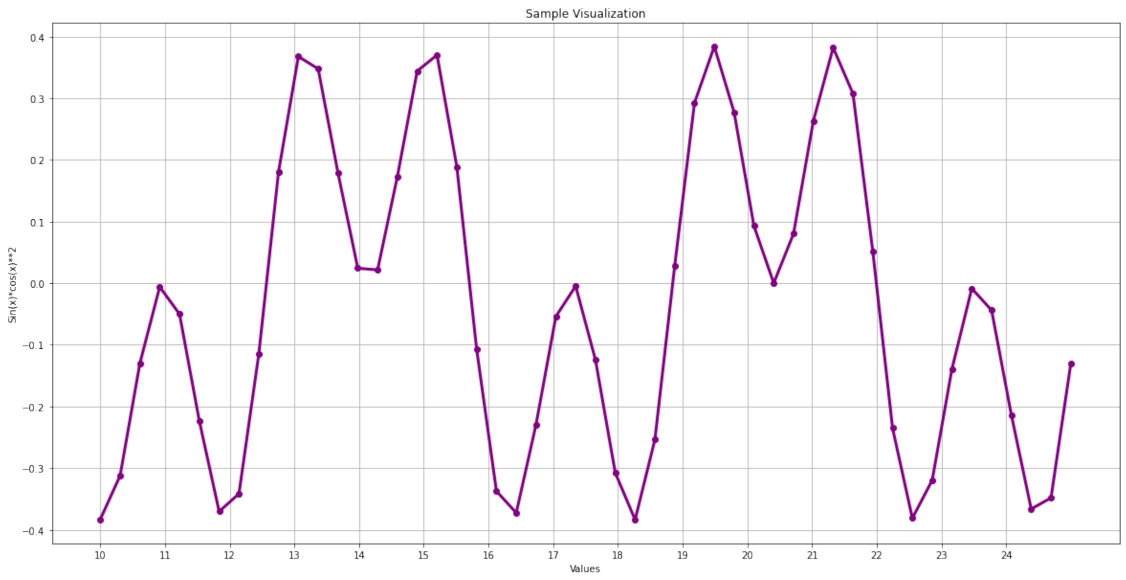
2.0.4 Adding Title

[13]:
```python
plt.figure(1, figsize=(20, 10))
plt.plot(x, y, color='purple', marker='o', linewidth=3)  # marker is circle/ o
plt.xlabel ("Values")
plt.ylabel ('Sin(x)*cos(x)**2')
plt.grid(True)
plt.title("Sample Visualization")
plt.show()
```

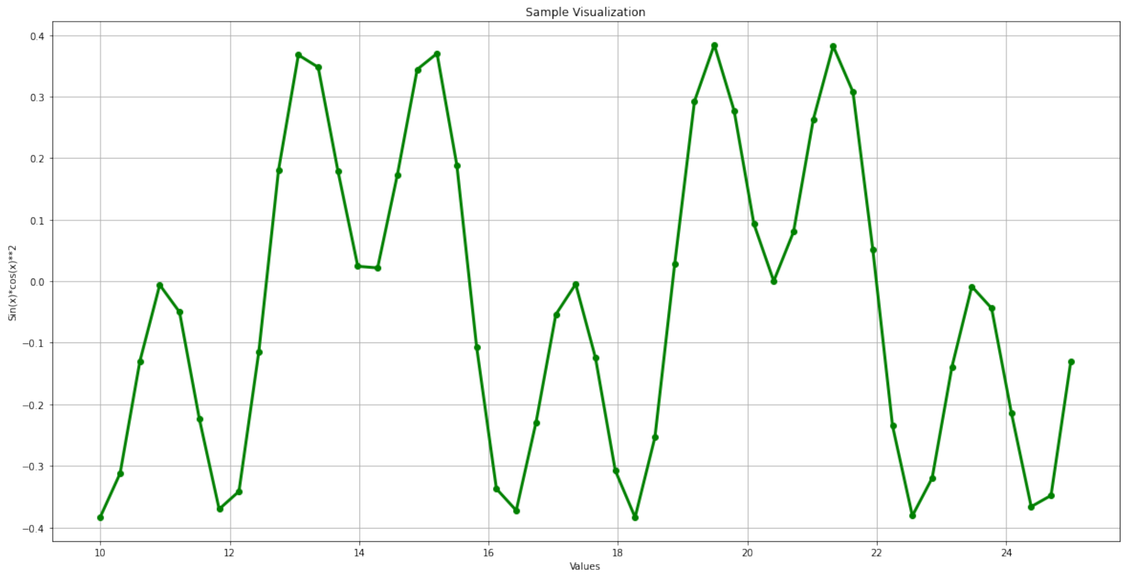


2.0.5 ticks

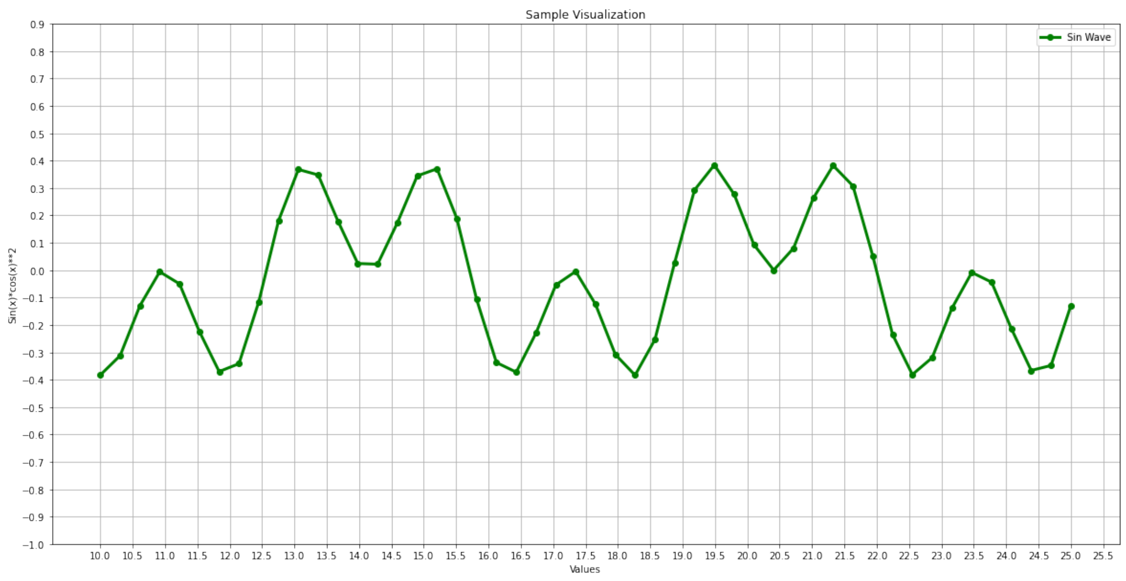
```
[14]: plt.figure(1, figsize=(20, 10))
plt.plot(x, y, color='purple', marker='o', linewidth=3)  # marker is circle/ o
plt.xlabel ("Values")
plt.ylabel ('Sin(x)*cos(x)**2')
plt.grid(True)
plt.title("Sample Visualization")
plt.xticks(np.arange(10,25))          # ticks will control the x value range
plt.show()
```



[15]:
```
plt.figure(1, figsize=(20, 10))
plt.plot(x, y, color='green', marker='o', linewidth=3)  # marker is circle/ o
plt.xlabel ("Values")
plt.ylabel ('Sin(x)*cos(x)**2')
plt.grid(True)
plt.title("Sample Visualization")
plt.xticks(np.arange(10,26,2))          # ticks with step size
plt.show()
```



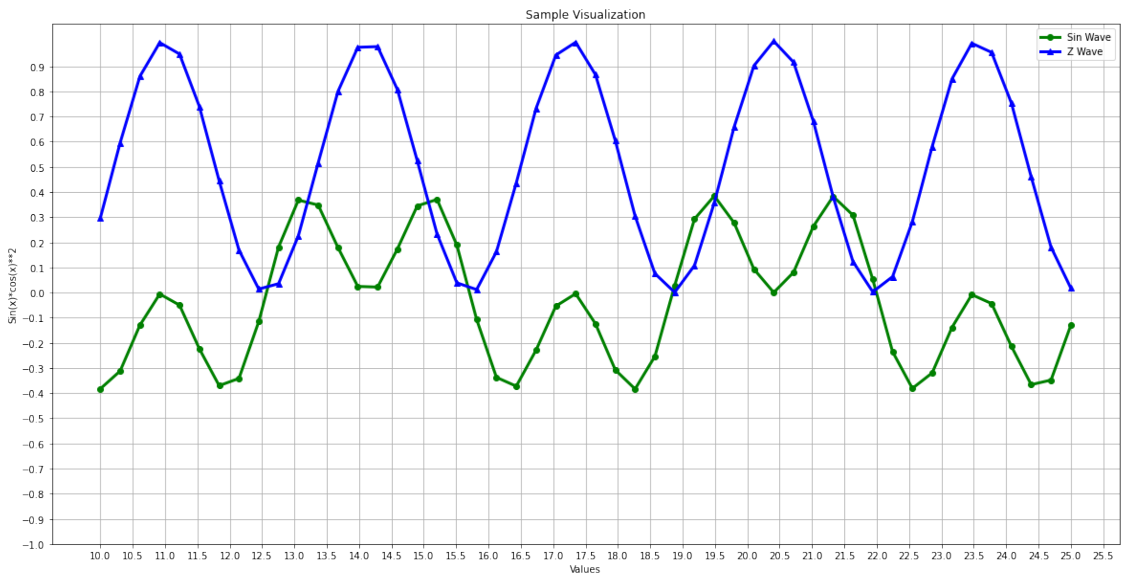
[16]:
```
plt.figure(1, figsize=(20, 10))
plt.plot(x, y, color='green', marker='o', linewidth=3, label='Sin Wave')  #␣
 ↪marker is circle/ o
plt.xlabel ("Values")
plt.ylabel ('Sin(x)*cos(x)**2')
plt.grid(True)
plt.title("Sample Visualization")
plt.xticks(np.arange(10,26, 0.5))          # ticks with step size
plt.yticks(np.arange(-1, 1, 0.1))
plt.legend()
plt.show()
```



2.0.6 Labels & Legend

[17]:
```python
# Visualization 3- Legend with z
x = np.linspace(10, 25)
y = np.sin(x) * np.cos(x) ** 2
z = np.sin(x) ** 2                    # square of sin wave
```

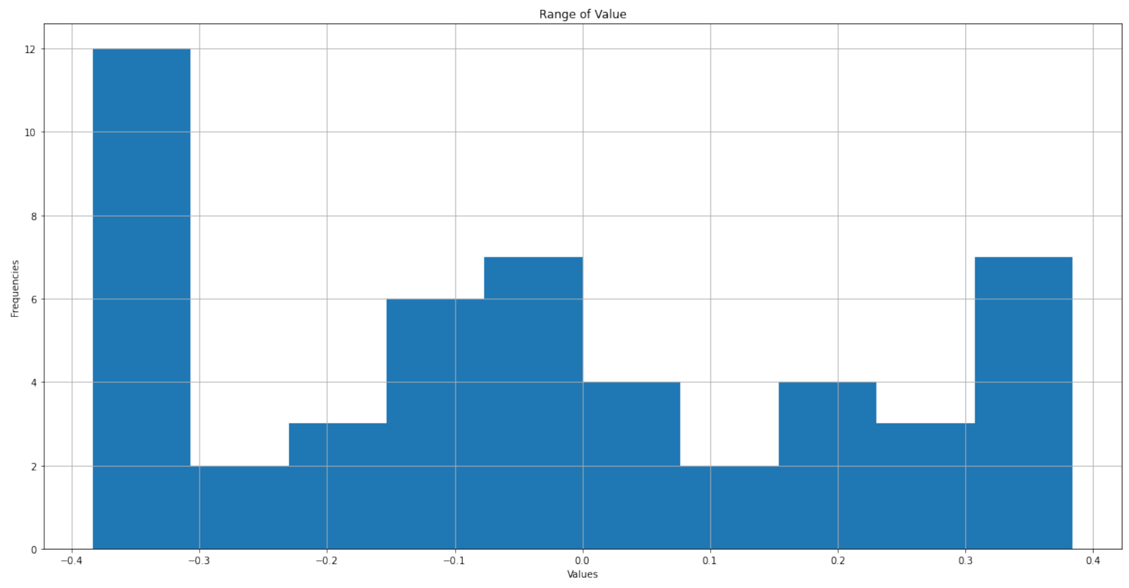
[18]:
```python
plt.figure(1, figsize=(20, 10))
plt.plot(x, y, color='green', marker='o', linewidth=3, label='Sin Wave')
plt.plot(x, z, color='blue', marker='^', linewidth=3, label='Z Wave')
plt.xlabel ("Values")
plt.ylabel ('Sin(x)*cos(x)**2')
plt.grid(True)
plt.title("Sample Visualization")
plt.xticks(np.arange(10,26, 0.5))         # ticks with step size
plt.yticks(np.arange(-1, 1, 0.1))
plt.legend()
plt.show()
```



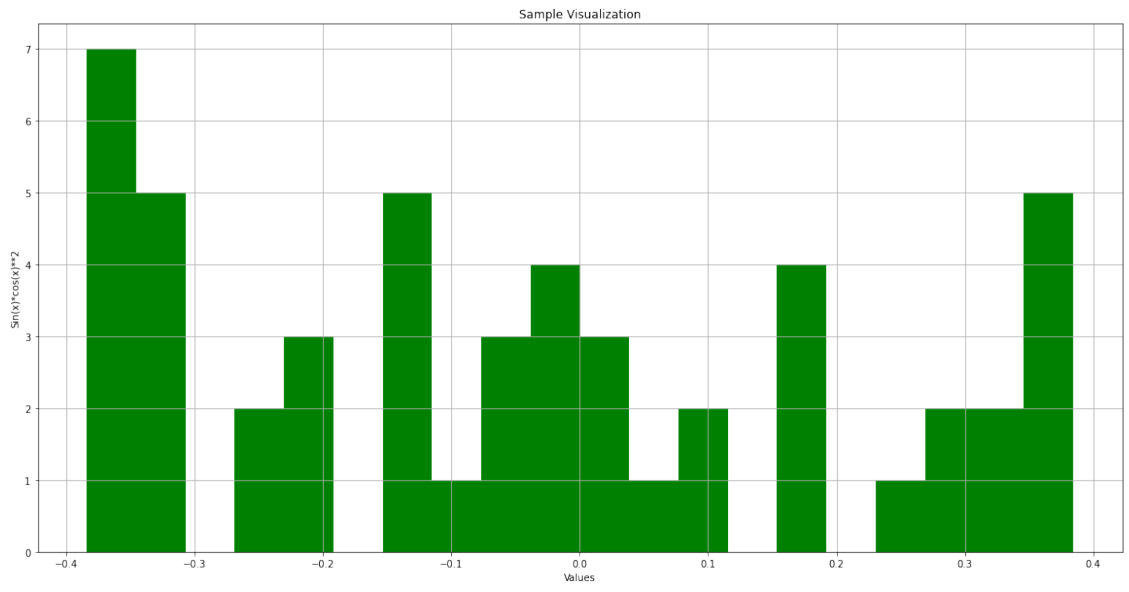
3 Histogram

- Bins: Do value counts by range/ bins
- x axis: value range / distribution
- y axis: Number of counts withing range
- it will take a single param & divide into number of bins
- Example: Number of employees within salary range / distribution

[19]:
```
plt.figure(1, figsize=(20, 10))
plt.hist(y)  # it will take a single param & divide into number of bins
plt.xlabel ("Values")
plt.ylabel ('Frequencies')
plt.grid(True)
plt.title("Range of Value")
plt.show()
```



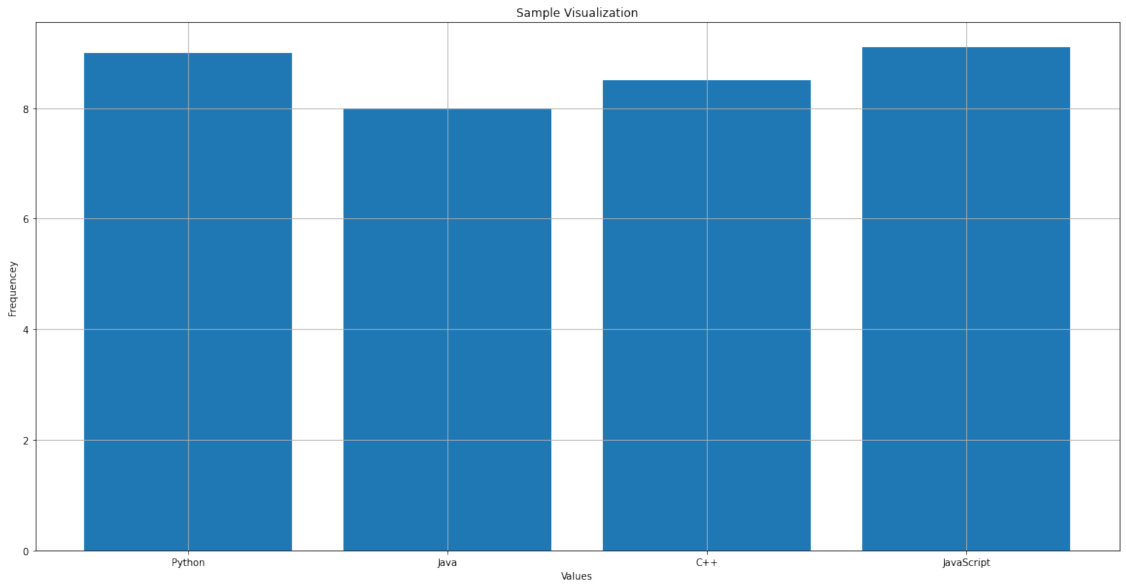
```
[20]: plt.figure(1, figsize=(20, 10))
      plt.hist(y, bins=20, color='green')  # Bin size & color customization
      plt.xlabel ("Values")
      plt.ylabel ('Sin(x)*cos(x)**2')
      plt.grid(True)
      plt.title("Sample Visualization")
      plt.show()
```



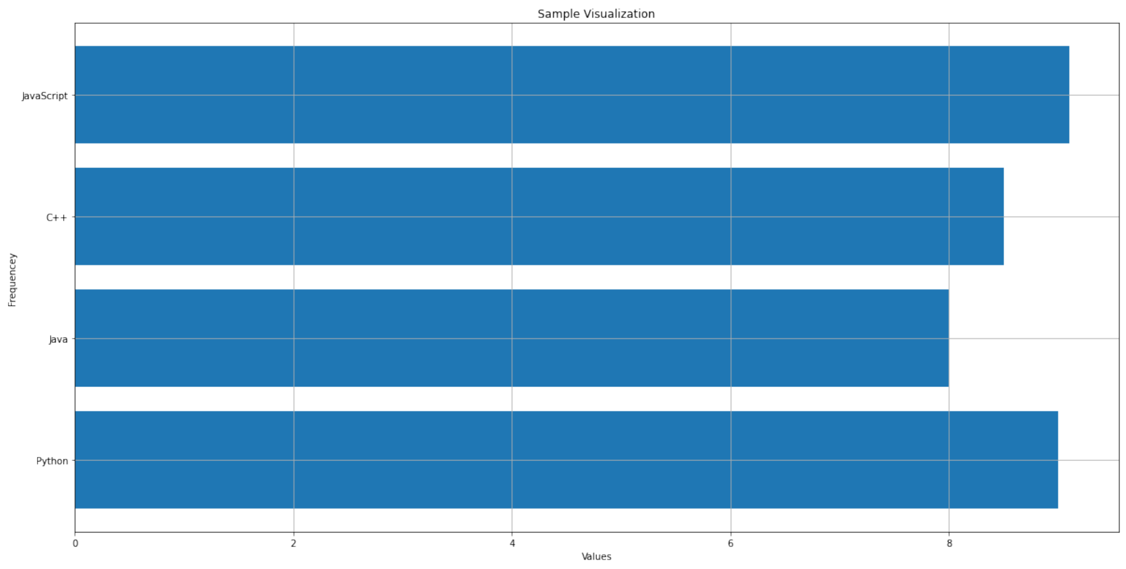
4 Bar Plot

- Deal with Categorical Distribution
- Histogram takes value range & count the frequency within the range or distribution
- Barplot divide into categorical distribution.
- Barplot represent the count/ frequence withing a population or given dataset, whereas histogram show the frequency within given range/distribution.

[21]:
```
plt.figure(1, figsize=(20, 10))
plt.bar(['Python','Java','C++','JavaScript'], [9.0, 8.0, 8.5, 9.10])
plt.xlabel ("Values")
plt.ylabel ('Frequencey')
plt.grid(True)
plt.title("Sample Visualization")
plt.show()
```



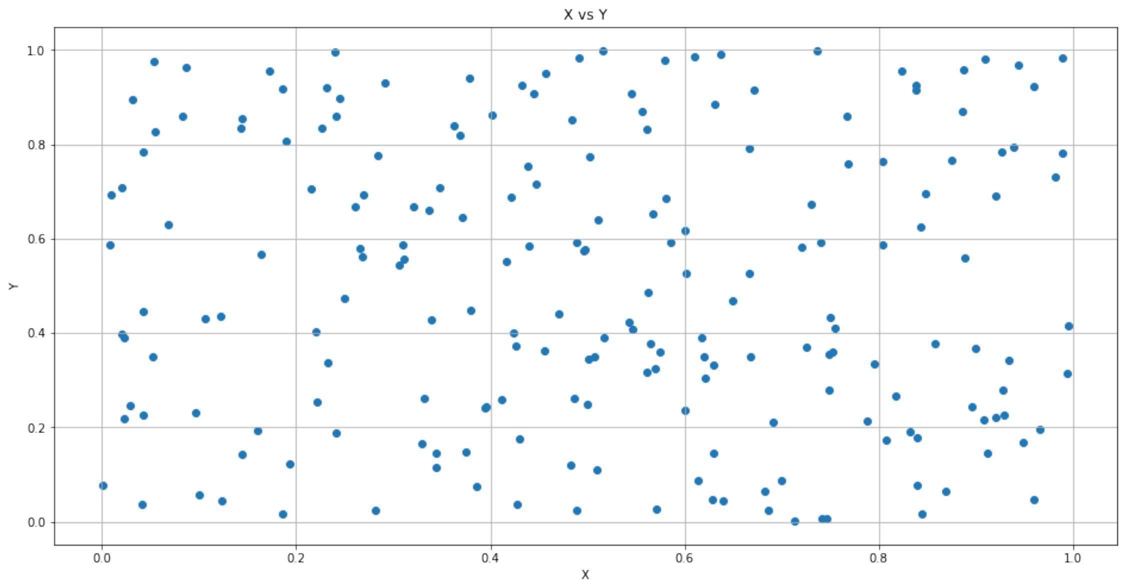
[22]:
```
plt.figure(1, figsize=(20, 10))
plt.barh(['Python','Java','C++','JavaScript'], [9.0, 8.0, 8.5, 9.10])  # barh:␣
 ↪Horizontal Bar plot
plt.xlabel ("Values")
plt.ylabel ('Frequencey')
plt.grid(True)
plt.title("Sample Visualization")
plt.show()
```



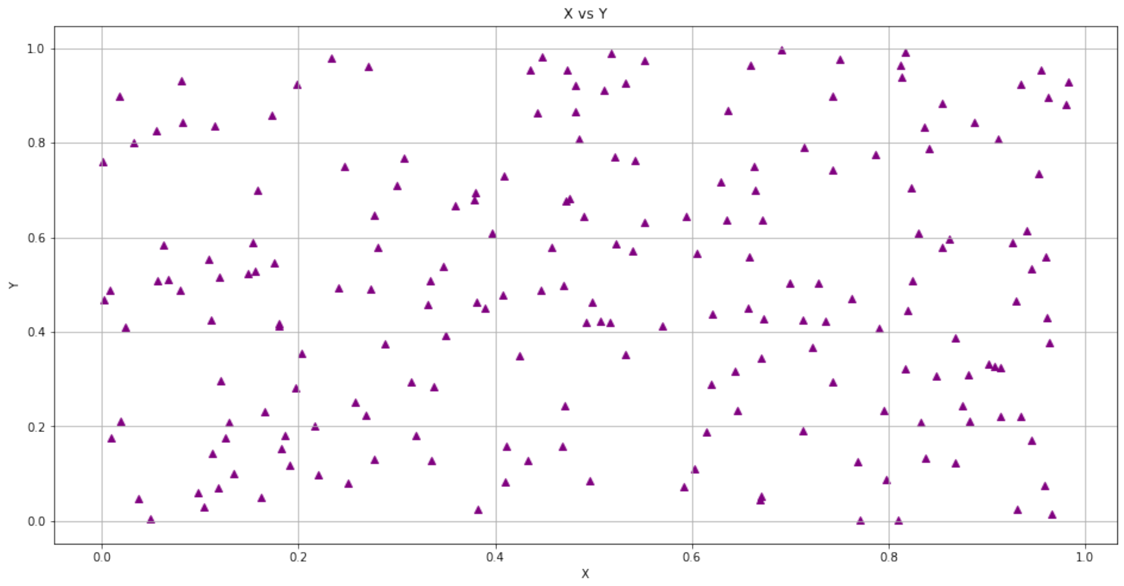
5 ScatterPlot

- x & y values will be plotted as discrete data points
- Multiple plots can be generated at a time

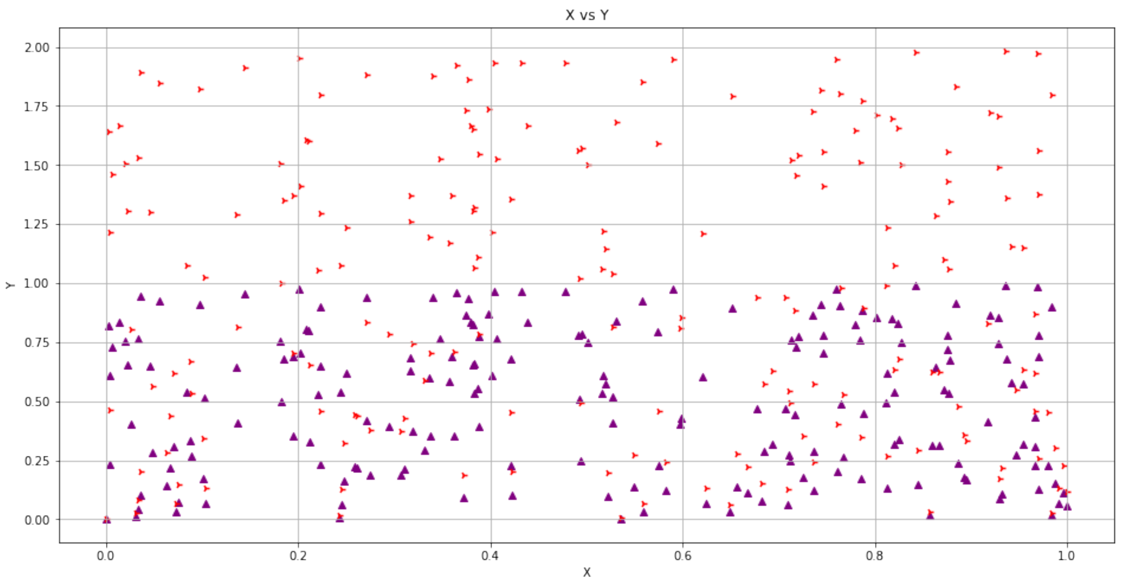
[23]:
```
plt.figure(1, figsize=(16, 8))
x = np.random.random(200) # random value generation, x & y must be same in␣
  ↪shape or size e.g 200
y = np.random.random(200)
plt.scatter(x,y)
plt.xlabel ('X')
plt.ylabel ('Y')
plt.grid(True)
plt.title("X vs Y")
plt.show()
```



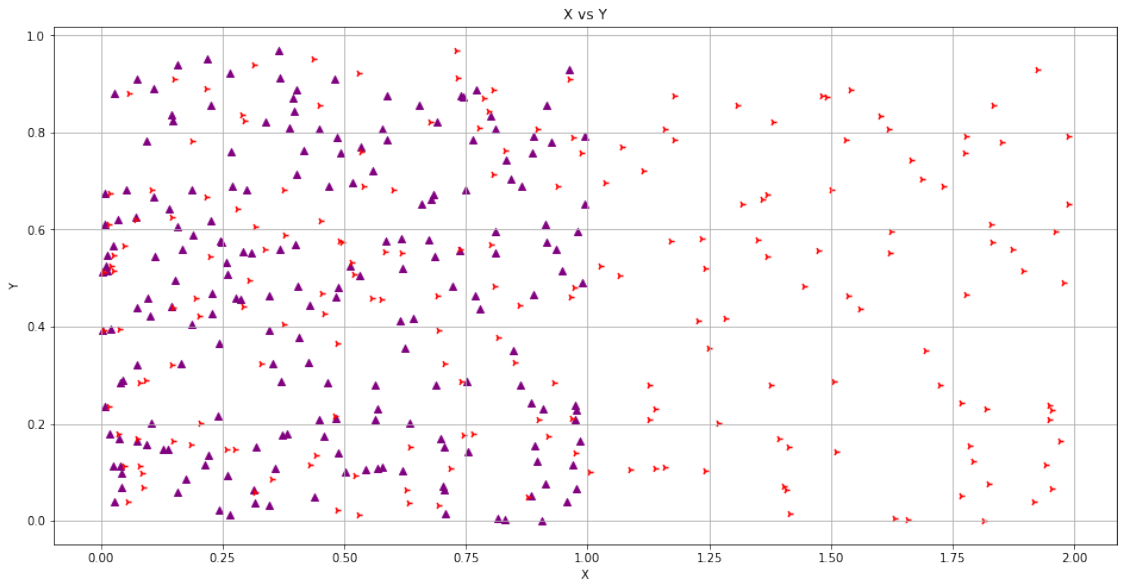
[24]:
```
plt.figure(1, figsize=(16, 8))
x = np.random.random(200) # random value generation, x & y must be same in␣
↪shape or size e.g 200
y = np.random.random(200)
plt.scatter(x,y, color='purple', marker='^') # Customization
plt.xlabel ('X')
plt.ylabel ('Y')
plt.grid(True)
plt.title("X vs Y")
plt.show()
```



[25]:
```
plt.figure(1, figsize=(16, 8))
x = np.random.random(200) # random value generation, x & y must be same in␣
 ↪shape or size e.g 200
y = np.random.random(200)
plt.scatter(x,y, color='purple', marker='^') # Customization
plt.scatter(x,y*2, color='red', marker='4') # Customization on y 2
plt.xlabel ('X')
plt.ylabel ('Y')
plt.grid(True)
plt.title("X vs Y")
plt.show()
```



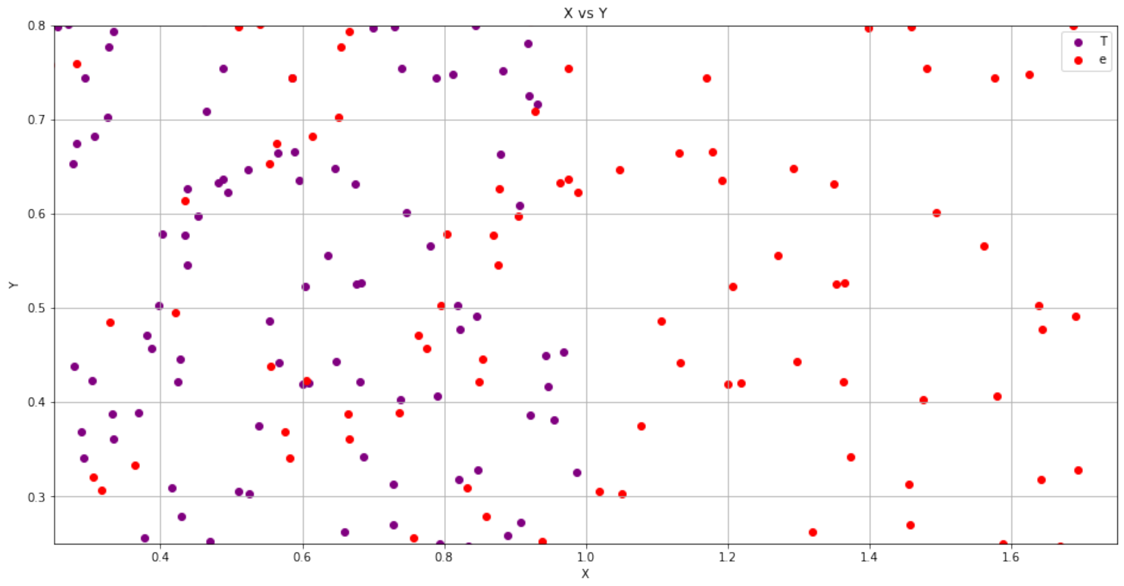
[26]:
```
plt.figure(1, figsize=(16, 8))
x = np.random.random(200) # random value generation, x & y must be same in␣
 ↪shape or size e.g 200
y = np.random.random(200)
plt.scatter(x,y, color='purple', marker='^') # Customization
plt.scatter(x*2, y, color='red', marker='4') # More customization with x
plt.xlabel ('X')
plt.ylabel ('Y')
plt.grid(True)
plt.title("X vs Y")
plt.show()
```



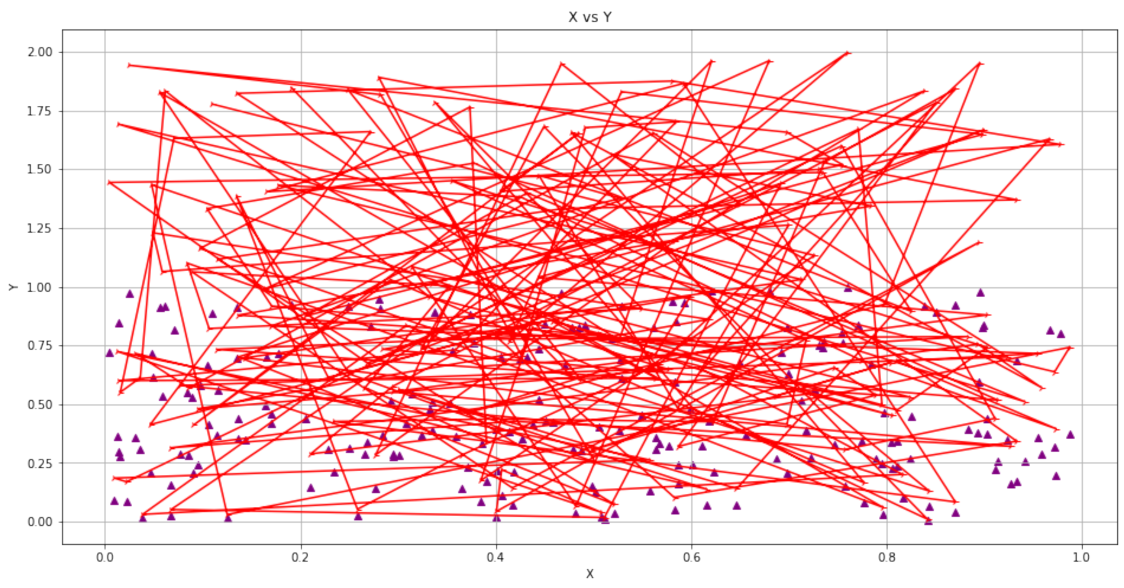
xlim & ylim

- plt.xlim() : allow setting the limit for x axis, customization allow
- Customization size / can set limit from left & right
- it used then there are not data out of the range

[27]:
```
plt.figure(1, figsize=(16, 8))
x = np.random.random(200)
y = np.random.random(200)
plt.scatter(x,y, color='purple', marker='o')
plt.scatter(x*2, y, color='red', marker='o')
plt.xlabel ('X')
plt.ylabel ('Y')
plt.grid(True)
plt.title("X vs Y")
plt.legend('Test')
plt.ylim(.25, 0.80)
plt.xlim(.25, 1.75)    # set limit: left / x will be started from .25  or range .
  ↪25 -1.75
plt.show()
```



[28]:
```
plt.figure(1, figsize=(16, 8))
x = np.random.random(200) # random value generation, x & y must be same in␣
 ↪shape or size e.g 200
y = np.random.random(200)
plt.scatter(x,y, color='purple', marker='^') # Customization
plt.plot(x,y*2, color='red', marker='4') # Line plot , multiple plot at a time
plt.xlabel ('X')
plt.ylabel ('Y')
plt.grid(True)
plt.title("X vs Y")
plt.show()
```



6 3D Visualization

```
[29]: x
```

```
[29]: array([0.10994779, 0.72595436, 0.38469238, 0.43729941, 0.57705293,
           0.0952752 , 0.61773503, 0.04937672, 0.05862509, 0.69976699,
           0.58621261, 0.44415978, 0.52728345, 0.1637131 , 0.67978335,
           0.4819785 , 0.31399708, 0.8236837 , 0.11460124, 0.50563057,
           0.80614379, 0.33215985, 0.06182443, 0.03622691, 0.01577493,
           0.0706898 , 0.2718478 , 0.04632158, 0.45091007, 0.22847579,
           0.13458127, 0.25843655, 0.36517009, 0.56722568, 0.89509261,
           0.7978537 , 0.37131024, 0.56334797, 0.59904185, 0.28442169,
           0.91334106, 0.37778883, 0.10451804, 0.1707612 , 0.65530333,
           0.13647842, 0.78422863, 0.51872924, 0.933073  , 0.58252362,
           0.40036852, 0.8994454 , 0.32387216, 0.44934735, 0.8416583 ,
           0.60183178, 0.33775454, 0.54965245, 0.10666193, 0.08425052,
           0.38599621, 0.8830259 , 0.33443347, 0.62091918, 0.52621693,
           0.13503512, 0.58120952, 0.97756633, 0.16579171, 0.59230167,
           0.87024597, 0.23392522, 0.4010324 , 0.62249774, 0.73615315,
           0.87041387, 0.61153166, 0.81621668, 0.03864509, 0.30847146,
           0.79653615, 0.05608989, 0.25044287, 0.50159089, 0.28217157,
           0.02502917, 0.69782758, 0.75503041, 0.73109796, 0.40465229,
           0.48456599, 0.09080776, 0.29242696, 0.77690449, 0.47742932,
           0.95799568, 0.89347923, 0.06734407, 0.41798304, 0.92739374,
           0.2954708 , 0.90230887, 0.44378579, 0.96654836, 0.64656247,
           0.30063592, 0.6156656 , 0.03154661, 0.08608503, 0.56116148,
           0.92340447, 0.07745989, 0.75750739, 0.91078981, 0.75412453,
           0.26888892, 0.00456344, 0.73495354, 0.82520853, 0.24884192,
           0.48701919, 0.09734613, 0.37357912, 0.39035889, 0.26620115,
           0.35264559, 0.08918621, 0.416935  , 0.89366228, 0.40067869,
           0.40600984, 0.5081438 , 0.01297145, 0.55765338, 0.0097175 ,
           0.02218107, 0.77107906, 0.79522922, 0.33487925, 0.89831544,
           0.52857628, 0.41419771, 0.75980444, 0.06776966, 0.51114289,
           0.69199611, 0.52897971, 0.13657446, 0.5210855 , 0.1156539 ,
           0.46991829, 0.93297877, 0.28034481, 0.42638022, 0.4916494 ,
           0.58462604, 0.35884537, 0.0143257 , 0.56410219, 0.45828702,
           0.81073022, 0.90290172, 0.35588173, 0.71750235, 0.69798351,
           0.14366864, 0.71312416, 0.01371463, 0.94093111, 0.43223717,
           0.20521818, 0.65015684, 0.46719867, 0.27705239, 0.58280705,
           0.83862063, 0.16996455, 0.9551732 , 0.81023437, 0.97337579,
           0.77433939, 0.42507928, 0.17744494, 0.97183616, 0.98721736,
           0.12519989, 0.04783727, 0.68719594, 0.80408744, 0.49821685,
           0.74629355, 0.78907286, 0.58688283, 0.85039677, 0.29460317,
           0.21064819, 0.72275297, 0.84326282, 0.47863774, 0.19105863])
```

```
[30]: y
```

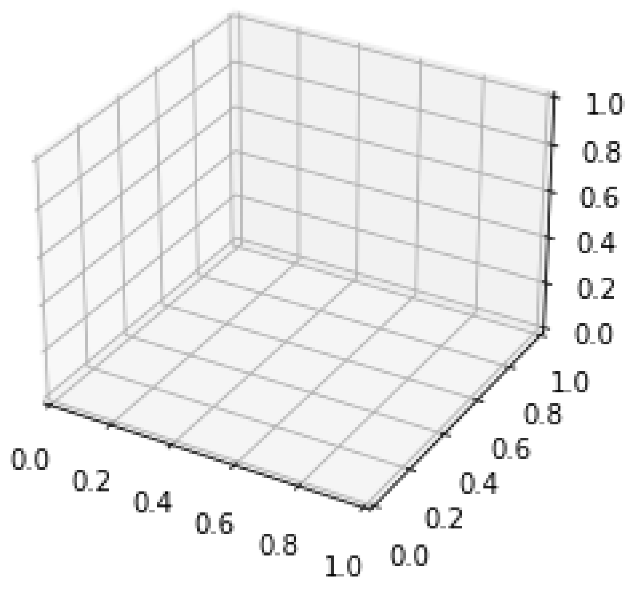
```
[30]: array([0.88834433, 0.56629367, 0.08749913, 0.41271138, 0.32384385,
        0.24002543, 0.43032082, 0.61450775, 0.53207261, 0.63105841,
        0.24121999, 0.73666486, 0.68792691, 0.49514553, 0.98076933,
        0.03331436, 0.54269352, 0.26492756, 0.36630852, 0.4013912 ,
        0.22572448, 0.47818466, 0.91634676, 0.30607433, 0.27531472,
        0.81561284, 0.82976179, 0.20628945, 0.42706024, 0.30693253,
        0.69318926, 0.02646921, 0.14156294, 0.33520513, 0.97578403,
        0.46263345, 0.23192175, 0.35593919, 0.47917386, 0.36667217,
        0.25873395, 0.76286397, 0.66596772, 0.45869602, 0.36700895,
        0.43939842, 0.67151498, 0.78174011, 0.17116567, 0.05156411,
        0.39389086, 0.83406819, 0.36715808, 0.84061734, 0.00522855,
        0.24055191, 0.89252714, 0.45313855, 0.41088053, 0.55006037,
        0.33076756, 0.3937792 , 0.38767652, 0.98043987, 0.38564415,
        0.91129947, 0.93737359, 0.80258098, 0.7020154 , 0.93245725,
        0.04192866, 0.21310929, 0.21837837, 0.21252956, 0.76464335,
        0.92077559, 0.32371173, 0.10052227, 0.01546254, 0.41740272,
        0.02989112, 0.91331829, 0.31399076, 0.12659146, 0.90851657,
        0.97098282, 0.82852084, 0.75908527, 0.74958617, 0.70277316,
        0.82623691, 0.20543168, 0.51469571, 0.08259269, 0.82783802,
        0.28574265, 0.37745505, 0.15696088, 0.2134186 , 0.16014577,
        0.27651399, 0.43966916, 0.51995197, 0.81424475, 0.07026839,
        0.28124047, 0.06909518, 0.35767432, 0.28451504, 0.82682842,
        0.35033172, 0.28969664, 0.15353254, 0.21979259, 0.79956192,
        0.33706734, 0.72169072, 0.74276983, 0.44731994, 0.9186832 ,
        0.30162422, 0.58120575, 0.88225726, 0.17168062, 0.28507896,
        0.50168306, 0.53132891, 0.07170369, 0.59338255, 0.02170569,
        0.1105775 , 0.01961221, 0.36184603, 0.13130038, 0.09265455,
        0.08471327, 0.83505399, 0.24827399, 0.50665528, 0.82391582,
        0.91466128, 0.38376351, 0.99810192, 0.02585762, 0.00887817,
        0.71987008, 0.60919905, 0.35046439, 0.03685349, 0.55818527,
        0.66635063, 0.68476765, 0.94532455, 0.3521065 , 0.8383318 ,
        0.85052555, 0.36545462, 0.29999764, 0.30538026, 0.42576951,
        0.23746705, 0.37067346, 0.72450167, 0.38646493, 0.20742163,
        0.34868883, 0.51522844, 0.84582944, 0.2555175 , 0.69944002,
        0.43736104, 0.67536708, 0.97401874, 0.14204398, 0.5965722 ,
        0.9165994 , 0.41594418, 0.35771081, 0.34156505, 0.19759147,
        0.34293119, 0.69110606, 0.71573708, 0.31864524, 0.37093176,
        0.01495551, 0.71658751, 0.26816771, 0.33848595, 0.15215167,
        0.327569  , 0.26919993, 0.1599721 , 0.89129309, 0.2878828 ,
        0.14409546, 0.27241054, 0.06535849, 0.30727578, 0.92292183])
```

[31]:
```
z
```

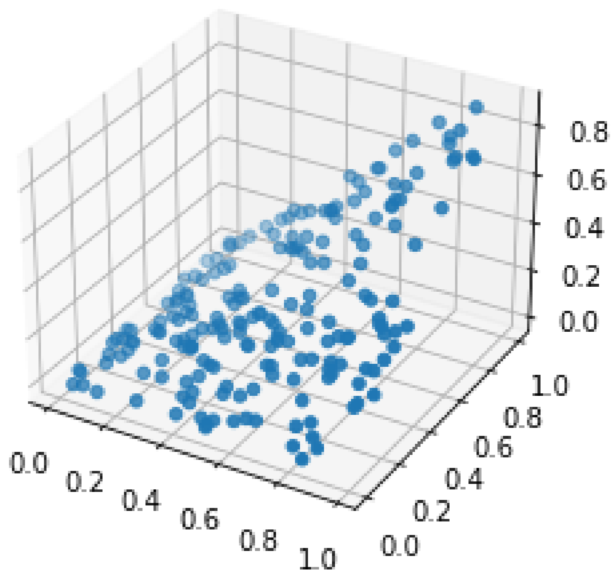
[31]:
```
array([2.95958969e-01, 5.95358573e-01, 8.60116300e-01, 9.94050923e-01,
       9.48506653e-01, 7.40028818e-01, 4.44353278e-01, 1.68893087e-01,
       1.37174662e-02, 3.51986373e-02, 2.25532917e-01, 5.15575641e-01,
       7.99960043e-01, 9.75374916e-01, 9.78095517e-01, 8.07133507e-01,
       5.24595990e-01, 2.33123238e-01, 3.86015168e-02, 1.16967057e-02,
       1.62182789e-01, 4.35391160e-01, 7.32070640e-01, 9.44443463e-01,
       9.95358797e-01, 8.66320114e-01, 6.04204604e-01, 3.04233630e-01,
       7.53806922e-02, 7.83520840e-04, 1.07541787e-01, 3.56872332e-01,
       6.58198318e-01, 9.02053995e-01, 9.99851436e-01, 9.16062772e-01,
       6.81126761e-01, 3.80390993e-01, 1.23106804e-01, 2.74043094e-03,
       6.30185896e-02, 2.82043421e-01, 5.80247551e-01, 8.49299339e-01,
       9.91457610e-01, 9.55079087e-01, 7.53379364e-01, 4.59631948e-01,
       1.80549444e-01, 1.75169858e-02])
```

- Neeed 3 values (e.g x, y, z)

[32]:
```
axes = plt.axes(projection='3d')    # This is pre-requisition for 3d visualizaion
```



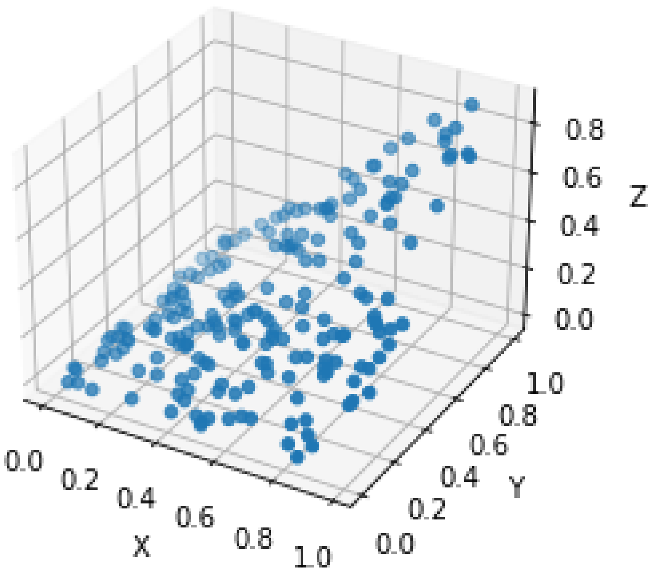
[33]:
```
axes = plt.axes(projection='3d')
z = x * y
axes.scatter3D(x, y, z)
plt.show()
```



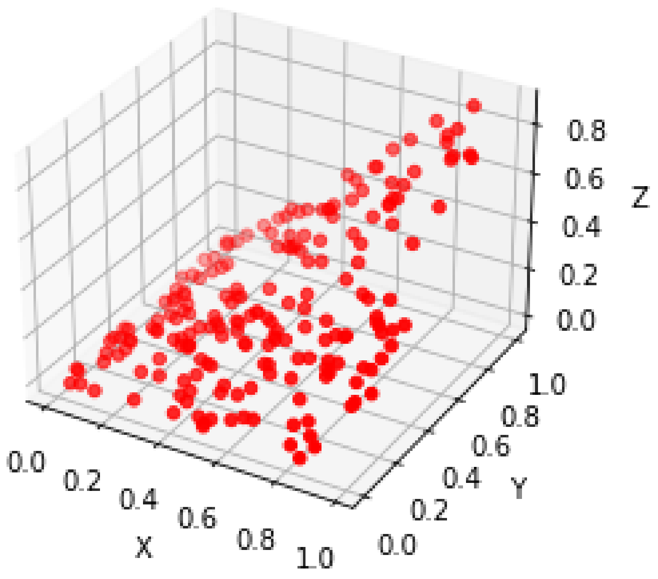
6.0.1 3D plot Customization

- In 3D plot interaction with other plot not possible
- Jupyter notebook in brower & pyfile will allow the interactivity
- For interaction Java Script used in backend

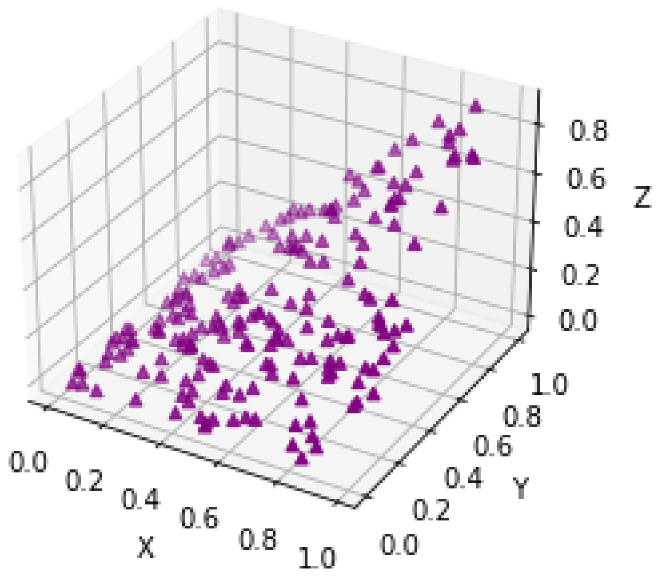
[34]:
```
# Adding Labels
axes = plt.axes(projection='3d')
z = x * y
axes.scatter3D(x, y, z)
axes.set_xlabel('X')
axes.set_ylabel('Y')
axes.set_zlabel('Z')
plt.show()
```



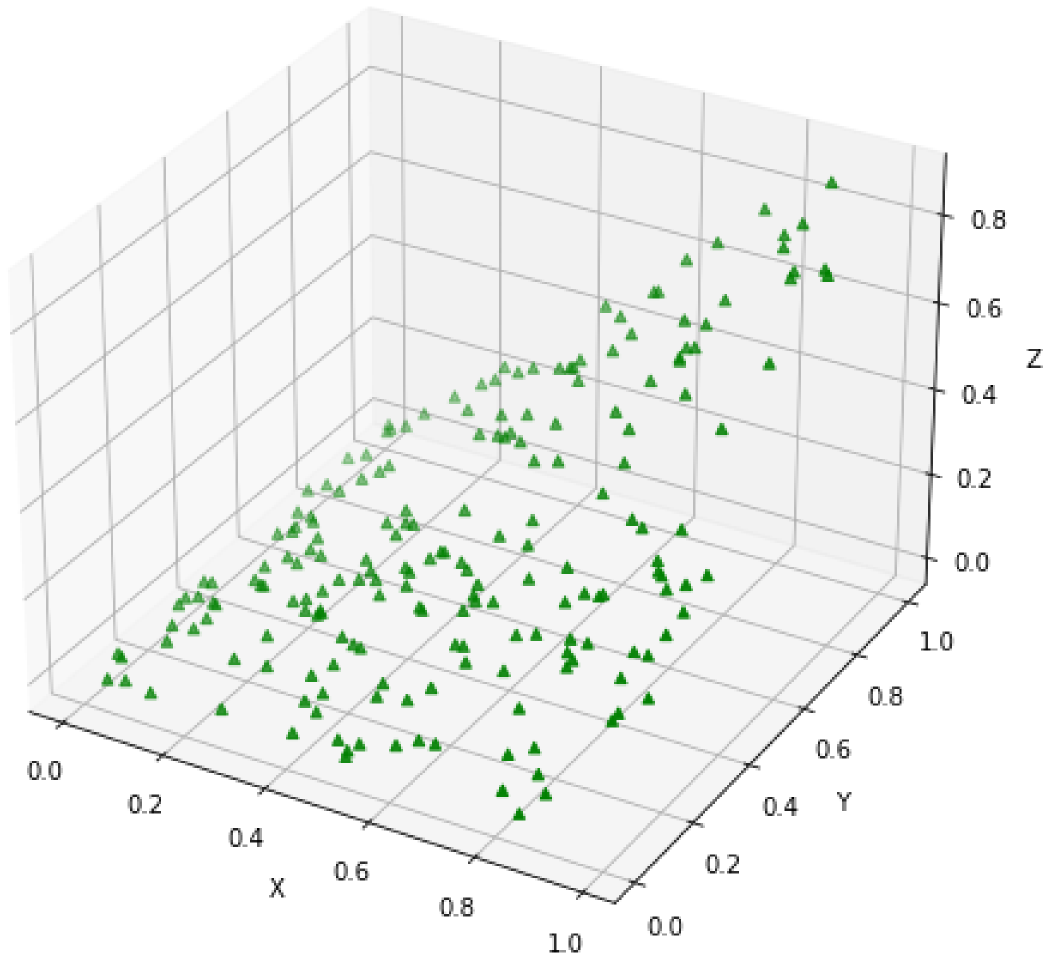
[35]:
```
# Color Customization
axes = plt.axes(projection='3d')
z = x * y
axes.scatter3D(x, y, z, color='red')
axes.set_xlabel('X')
axes.set_ylabel('Y')
axes.set_zlabel('Z')
plt.show()
```



[36]:
```
# Marker Customization
axes = plt.axes(projection='3d')
z = x * y
axes.scatter3D(x, y, z, color='purple', marker='^')
axes.set_xlabel('X')
axes.set_ylabel('Y')
axes.set_zlabel('Z')
plt.show()
```

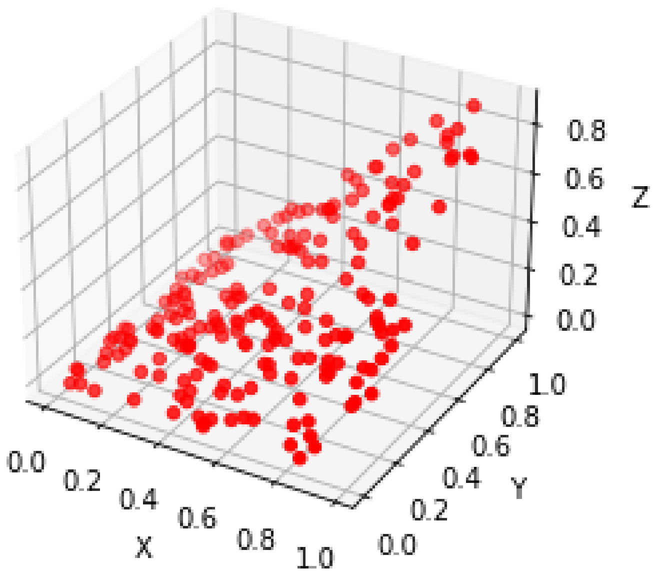


[37]:
```
# Size Customization
plt.figure(1, figsize=(16, 8))
axes = plt.axes(projection='3d')
z = x * y
axes.scatter3D(x, y, z, color='green', marker='^')
axes.set_xlabel('X')
axes.set_ylabel('Y')
axes.set_zlabel('Z')
plt.show()
```



7 Save the Plot / Visualization

[38]:
```
axes = plt.axes(projection='3d')
z = x * y
axes.scatter3D(x, y, z, color='red')
axes.set_xlabel('X')
axes.set_ylabel('Y')
axes.set_zlabel('Z')
plt.savefig('test.png')  # file name test, formate, png, jpg any can define
```


0.0
0.2
0.4
0.6
0.8
1.0
X
0.0
0.2
0.4
0.6
0.8
1.0
Y
0.0
0.2
0.4
0.6
0.8
Z