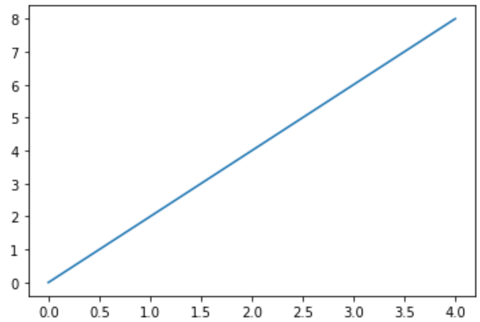
```
import numpy as np
import pandas as pd
import math
```

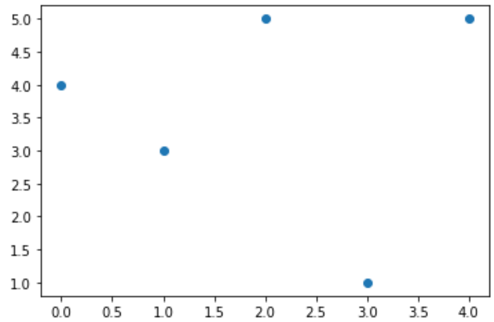
```
import matplotlib.pyplot as plt
%matplotlib inline
```

```
x = [0,1,2,3,4]
y = [0,2,4,6,8]

plt.plot(x,y)
plt.show()
```

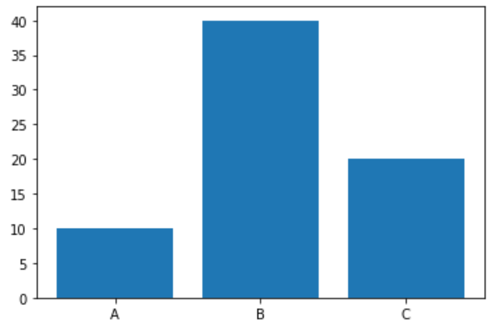
```
[<matplotlib.lines.Line2D at 0x7fae739f5700>]
```



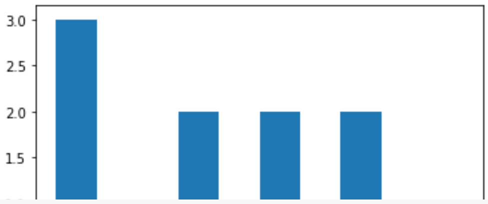

```
x = [0,1,2,3,4]
y = [4,3,5,1,5]

plt.scatter(x,y)
plt.show()
```




```
labels = ['A', 'B', 'C']
values = [10,40,20]

plt.bar(labels, values)
plt.show()
```

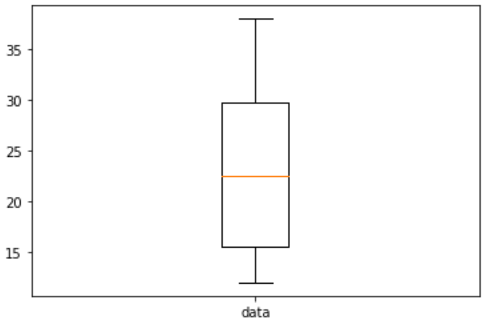



```
l1 = [20,20,14,26,14,25,31,12,32,38]
plt.hist(l1)
plt.show()
```



```
plt.pie([85, 75, 30])
plt.show()
```

```
l1 = [20,20,14,26,14,25,31,12,32,38]
bp = plt.boxplot(l1, labels=['data'])
plt.show()
```




```
import io
from google.colab import files

uploaded = files.upload()
```

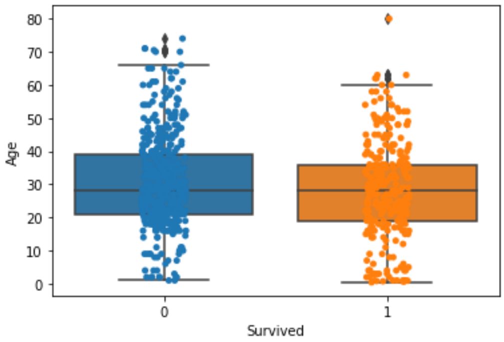
Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving titanic.csv to titanic.csv

```
df = pd.read_csv(io.BytesIO(uploaded['titanic.csv']))
df
```

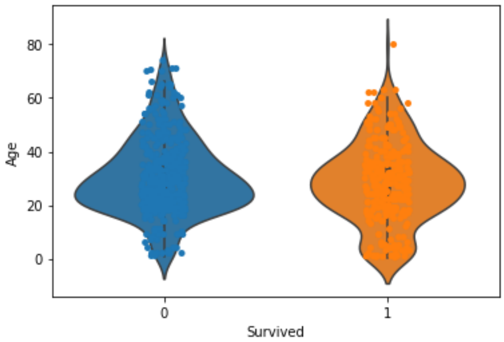
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S

```
import seaborn as sns
```

```
ax = sns.boxplot(x='Survived',y='Age', data=df)
ax = sns.stripplot(x='Survived',y='Age', data=df)
```



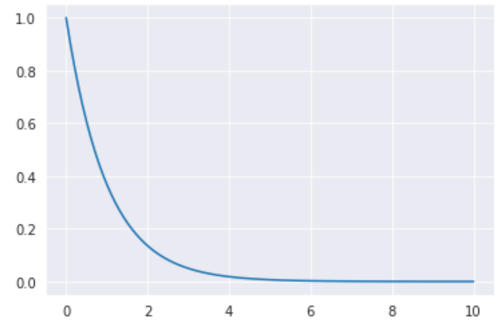

```
ax = sns.violinplot(x='Survived',y='Age', data=df)
ax = sns.stripplot(x='Survived',y='Age', data=df)
```




```
a = np.linspace(0,10,100)
b = np.exp(-a)
print(a)
print(b)
```

```
[ 0.          0.1010101   0.2020202   0.3030303   0.4040404   0.50505051
  0.60606061  0.70707071  0.80808081  0.90909091  1.01010101  1.11111111
  1.21212121  1.31313131  1.41414141  1.51515152  1.61616162  1.71717172
  1.81818182  1.91919192  2.02020202  2.12121212  2.22222222  2.32323232
  2.42424242  2.52525253  2.62626263  2.72727273  2.82828283  2.92929293
  3.03030303  3.13131313  3.23232323  3.33333333  3.43434343  3.53535354
  3.63636364  3.73737374  3.83838384  3.93939394  4.04040404  4.14141414
  4.24242424  4.34343434  4.44444444  4.54545455  4.64646465  4.74747475
  4.84848485  4.94949495  5.05050505  5.15151515  5.25252525  5.35353535
  5.45454545  5.55555556  5.65656566  5.75757576  5.85858586  5.95959596
  6.06060606  6.16161616  6.26262626  6.36363636  6.46464646  6.56565657
  6.66666667  6.76767677  6.86868687  6.96969697  7.07070707  7.17171717
  7.27272727  7.37373737  7.47474747  7.57575758  7.67676768  7.77777778
  7.87878788  7.97979798  8.08080808  8.18181818  8.28282828  8.38383838
  8.48484848  8.58585859  8.68686869  8.78787879  8.88888889  8.98989899
  9.09090909  9.19191919  9.29292929  9.39393939  9.49494949  9.5959596
  9.6969697   9.7979798   9.8989899  10.        ]
[1.00000000e+00 9.03923902e-01 8.17078421e-01 7.38576715e-01
 6.67617146e-01 6.03475096e-01 5.45495564e-01 4.93086479e-01
 4.45712654e-01 4.02890322e-01 3.64182192e-01 3.29192988e-01
 2.97565410e-01 2.68976487e-01 2.43134276e-01 2.19774883e-01
 1.98659770e-01 1.79573314e-01 1.62320611e-01 1.46725480e-01
 1.32628669e-01 1.19886224e-01 1.08368023e-01 9.79564464e-02
 8.85451733e-02 8.00380986e-02 7.23483504e-02 6.53974032e-02
 5.91142759e-02 5.34348070e-02 4.83009992e-02 4.36604277e-02
 3.94657042e-02 3.56739933e-02 3.22465753e-02 2.91484502e-02
 2.63479808e-02 2.38165696e-02 2.15283666e-02 1.94600051e-02
 1.75903638e-02 1.59003503e-02 1.43727066e-02 1.29918331e-02
 1.17436285e-02 1.06153465e-02 9.59546540e-03 8.67357053e-03
 7.84024772e-03 7.08698731e-03 6.40609723e-03 5.79062440e-03
 5.23428381e-03 4.73139424e-03 4.27682035e-03 3.86592014e-03
 3.49449762e-03 3.15875992e-03 2.85527860e-03 2.58095457e-03
 2.33298653e-03 2.10884229e-03 1.90623295e-03 1.72308953e-03
 1.55754181e-03 1.40789927e-03 1.27263380e-03 1.15036411e-03
 1.03984162e-03 9.39937692e-04 8.49632147e-04 7.68002806e-04
 6.94216093e-04 6.27518520e-04 5.67228989e-04 5.12731841e-04
 4.63470567e-04 4.18942123e-04 3.78691799e-04 3.42308569e-04
 3.09420897e-04 2.79692945e-04 2.52821138e-04 2.28531070e-04
 2.06574696e-04 1.86727806e-04 1.68787727e-04 1.52571261e-04
 1.37912809e-04 1.24662685e-04 1.12685581e-04 1.01859190e-04
 9.20729562e-05 8.32269459e-05 7.52308257e-05 6.80029415e-05
 6.14694843e-05 5.55637361e-05 5.02253892e-05 4.53999298e-05]
```

```
plt.plot(a,b)
plt.show()
```



```
import sklearn
from sklearn import datasets
```

```
df = datasets.load_iris()
df
```

```
           [7.4, 2.8, 6.1, 1.9],
           [7.9, 3.8, 6.4, 2. ],
           [6.4, 2.8, 5.6, 2.2],
           [6.3, 2.8, 5.1, 1.5],
           [6.1, 2.6, 5.6, 1.4],
           [7.7, 3. , 6.1, 2.3],
           [6.3, 3.4, 5.6, 2.4],
           [6.4, 3.1, 5.5, 1.8],
           [6. , 3. , 4.8, 1.8],
           [6.9, 3.1, 5.4, 2.1],
           [6.7, 3.1, 5.6, 2.4],
           [6.9, 3.1, 5.1, 2.3],
           [5.8, 2.7, 5.1, 1.9],
           [6.8, 3.2, 5.9, 2.3],
           [6.7, 3.3, 5.7, 2.5],
           [6.7, 3. , 5.2, 2.3],
           [6.3, 2.5, 5. , 1.9],
           [6.5, 3. , 5.2, 2. ],
           [6.2, 3.4, 5.4, 2.3],
           [5.9, 3. , 5.1, 1.8]]),
 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 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, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
        2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
        2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]),
 'frame': None,
 'target_names': array(['setosa', 'versicolor', 'virginica'], dtype='<U10'),
 'DESCR': '.. _iris_dataset:\n\nIris plants dataset\n--------------------\n\n**Data Set Characteristics:**\n\n    :Number of Instances: 150 (50 in each of three classes)\n    :Number of Attributes: 4 numeric, predictive attributes and the class\n    :Attribute Information:\n        - sepal length in cm\n        - sepal width in cm\n        - petal length in cm\n        - petal width in cm\n        - class:\n                - Iris-Setosa\n                - Iris-Versicolour\n                - Iris-Virginica\n                \n    :Summary Statistics:\n\n    ============== ==== ==== ======= ===== ====================\n                    Min  Max   Mean    SD   Class Correlation\n    ============== ==== ==== ======= ===== ====================\n    sepal length:   4.3  7.9   5.84   0.83    0.7826\n    sepal width:    2.0  4.4   3.05   0.43   -0.4194\n    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)\n    petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)\n    ============== ==== ==== ======= ===== ====================\n\n    :Missing Attribute Values: None\n    :Class Distribution: 33.3% for each of 3 classes.\n    :Creator: R.A. Fisher\n    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)\n    :Date: July, 1988\n\nThe famous Iris database, first used by Sir R.A. Fisher. The dataset is taken\nfrom Fisher\'s paper. Note that it\'s the same as in R, but not as in the UCI\nMachine Learning Repository, which has two wrong data points.\n\nThis is perhaps the best known database to be found in the\npattern recognition literature.  Fisher\'s paper is a classic in the field and\nis referenced frequently to this day.  (See Duda & Hart, for example.)  The\ndata set contains 3 classes of 50 instances each, where each class refers to a\ntype of iris plant.  One class is linearly separable from the other 2; the\nlatter are NOT linearly separable from each other.\n\n.. topic:: References\n\n   - Fisher, R.A. "The use of multiple measurements in taxonomic problems"\n     Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to\n     Mathematical Statistics" (John Wiley, NY, 1950).\n   - Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis.\n     (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.\n   - Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System\n     Structure and Classification Rule for Recognition in Partially Exposed\n     Environments".  IEEE Transactions on Pattern Analysis and Machine\n     Intelligence, Vol. PAMI-2, No. 1, 67-71.\n   - Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE Transactions\n     on Information Theory, May 1972, 431-433.\n   - See also: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s AUTOCLASS II\n     conceptual clustering system finds 3 classes in the data.\n   - Many, many more ...',
 'feature_names': ['sepal length (cm)',
  'sepal width (cm)',
  'petal length (cm)',
  'petal width (cm)'],
 'filename': 'iris.csv',
 'data_module': 'sklearn.datasets.data'}
```

```
df.data
```

```
       [5.6, 2.7, 4.2, 1.3],
       [5.7, 3. , 4.2, 1.2],
       [5.7, 2.9, 4.2, 1.3],
       [6.2, 2.9, 4.3, 1.3],
       [5.1, 2.5, 3. , 1.1],
       [5.7, 2.8, 4.1, 1.3],
       [6.3, 3.3, 6. , 2.5],
       [5.8, 2.7, 5.1, 1.9],
       [7.1, 3. , 5.9, 2.1],
       [6.3, 2.9, 5.6, 1.8],
       [6.5, 3. , 5.8, 2.2],
       [7.6, 3. , 6.6, 2.1],
       [4.9, 2.5, 4.5, 1.7],
       [7.3, 2.9, 6.3, 1.8],
       [6.7, 2.5, 5.8, 1.8],
       [7.2, 3.6, 6.1, 2.5],
       [6.5, 3.2, 5.1, 2. ],
       [6.4, 2.7, 5.3, 1.9],
       [6.8, 3. , 5.5, 2.1],
       [5.7, 2.5, 5. , 2. ],
       [5.8, 2.8, 5.1, 2.4],
       [6.4, 3.2, 5.3, 2.3],
       [6.5, 3. , 5.5, 1.8],
       [7.7, 3.8, 6.7, 2.2],
       [7.7, 2.6, 6.9, 2.3],
       [6. , 2.2, 5. , 1.5],
       [6.9, 3.2, 5.7, 2.3],
       [5.6, 2.8, 4.9, 2. ],
       [7.7, 2.8, 6.7, 2. ],
       [6.3, 2.7, 4.9, 1.8],
       [6.7, 3.3, 5.7, 2.1],
       [7.2, 3.2, 6. , 1.8],
       [6.2, 2.8, 4.8, 1.8],
       [6.1, 3. , 4.9, 1.8],
       [6.4, 2.8, 5.6, 2.1],
       [7.2, 3. , 5.8, 1.6],
       [7.4, 2.8, 6.1, 1.9],
       [7.9, 3.8, 6.4, 2. ],
       [6.4, 2.8, 5.6, 2.2],
       [6.3, 2.8, 5.1, 1.5],
       [6.1, 2.6, 5.6, 1.4],
       [7.7, 3. , 6.1, 2.3],
       [6.3, 3.4, 5.6, 2.4],
       [6.4, 3.1, 5.5, 1.8],
       [6. , 3. , 4.8, 1.8],
       [6.9, 3.1, 5.4, 2.1],
       [6.7, 3.1, 5.6, 2.4],
       [6.9, 3.1, 5.1, 2.3],
       [5.8, 2.7, 5.1, 1.9],
       [6.8, 3.2, 5.9, 2.3],
       [6.7, 3.3, 5.7, 2.5],
       [6.7, 3. , 5.2, 2.3],
       [6.3, 2.5, 5. , 1.9],
       [6.5, 3. , 5.2, 2. ],
       [6.2, 3.4, 5.4, 2.3],
       [5.9, 3. , 5.1, 1.8]])
```

```
df.target
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 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, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

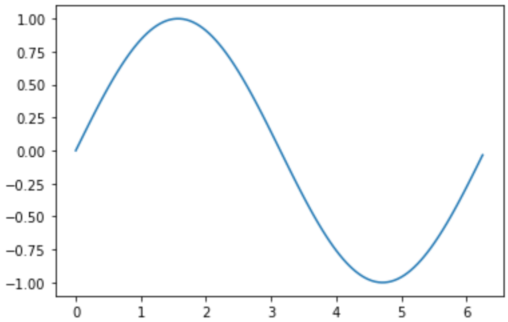
```
x = np.arange(0, math.pi*2, 0.05)
x
```

```
array([0.  , 0.05, 0.1 , 0.15, 0.2 , 0.25, 0.3 , 0.35, 0.4 , 0.45, 0.5 ,
       0.55, 0.6 , 0.65, 0.7 , 0.75, 0.8 , 0.85, 0.9 , 0.95, 1.  , 1.05,
       1.1 , 1.15, 1.2 , 1.25, 1.3 , 1.35, 1.4 , 1.45, 1.5 , 1.55, 1.6 ,
       1.65, 1.7 , 1.75, 1.8 , 1.85, 1.9 , 1.95, 2.  , 2.05, 2.1 , 2.15,
       2.2 , 2.25, 2.3 , 2.35, 2.4 , 2.45, 2.5 , 2.55, 2.6 , 2.65, 2.7 ,
       2.75, 2.8 , 2.85, 2.9 , 2.95, 3.  , 3.05, 3.1 , 3.15, 3.2 , 3.25,
       3.3 , 3.35, 3.4 , 3.45, 3.5 , 3.55, 3.6 , 3.65, 3.7 , 3.75, 3.8 ,
       3.85, 3.9 , 3.95, 4.  , 4.05, 4.1 , 4.15, 4.2 , 4.25, 4.3 , 4.35,
       4.4 , 4.45, 4.5 , 4.55, 4.6 , 4.65, 4.7 , 4.75, 4.8 , 4.85, 4.9 ,
       4.95, 5.  , 5.05, 5.1 , 5.15, 5.2 , 5.25, 5.3 , 5.35, 5.4 , 5.45,
       5.5 , 5.55, 5.6 , 5.65, 5.7 , 5.75, 5.8 , 5.85, 5.9 , 5.95, 6.  ,
       6.05, 6.1 , 6.15, 6.2 , 6.25])
```

```
y = np.sin(x)
y
```

```
array([ 0.        ,  0.04997917,  0.09983342,  0.14943813,  0.19866933,
        0.24740396,  0.29552021,  0.34289781,  0.38941834,  0.43496553,
        0.47942554,  0.52268723,  0.56464247,  0.60518641,  0.64421769,
        0.68163876,  0.71735609,  0.75128041,  0.78332691,  0.8134155 ,
        0.84147098,  0.86742323,  0.89120736,  0.91276394,  0.93203909,
        0.94898462,  0.96355819,  0.97572336,  0.98544973,  0.99271299,
        0.99749499,  0.99978376,  0.9995736 ,  0.99686503,  0.99166481,
        0.98398595,  0.97384763,  0.9612752 ,  0.94630009,  0.92895972,
        0.90929743,  0.88736237,  0.86320937,  0.83689879,  0.8084964 ,
        0.7780732 ,  0.74570521,  0.71147335,  0.67546318,  0.6377647 ,
        0.59847214,  0.55768372,  0.51550137,  0.47203054,  0.42737988,
        0.38166099,  0.33498815,  0.28747801,  0.23924933,  0.19042265,
        0.14112001,  0.09146464,  0.04158066, -0.00840725, -0.05837414,
       -0.10819513, -0.15774569, -0.20690197, -0.2555411 , -0.30354151,
       -0.35078323, -0.39714817, -0.44252044, -0.48678665, -0.52983614,
       -0.57156132, -0.61185789, -0.65062514, -0.68776616, -0.72318812,
       -0.7568025 , -0.78852525, -0.81827711, -0.8459837 , -0.87157577,
       -0.89498936, -0.91616594, -0.93505258, -0.95160207, -0.96577306,
       -0.97753012, -0.98684386, -0.993691  , -0.99805444, -0.99992326,
       -0.99929279, -0.99616461, -0.99054654, -0.98245261, -0.97190307,
       -0.95892427, -0.94354867, -0.92581468, -0.90576664, -0.88345466,
       -0.85893449, -0.83226744, -0.80352016, -0.77276449, -0.74007731,
       -0.70554033, -0.66923986, -0.63126664, -0.59171558, -0.55068554,
       -0.50827908, -0.46460218, -0.41976402, -0.37387666, -0.32705481,
       -0.2794155 , -0.23107779, -0.1821625 , -0.13279191, -0.0830894 ,
       -0.03317922])
```

```
plt.plot(x,y)
plt.show()
```




```
from mpl_toolkits import mplot3d

# Creating dataset
z = np.random.randint(100, size =(50))
x = np.random.randint(80, size =(50))
y = np.random.randint(60, size =(50))

print(x)
print(y)
print(z)
```

```
[27 65  4 22 60 31 52 11 76 51 72 25 49 71 56 40 30 76 19 47 17 65 53 74
 36 40 58 76 74 77 20 49 47  1 10 76 39 32 68 29 70 29 63 30  7 57 22 67
 10 26]
[50 16 52 48 45 25 19 29 54 47 35  7 47 46  9  1 26 20 18 14 46  2  1 13
 54  5 20  0 45 57 51 30 38 29  9  7  0  6 51 32 33 50 16  4  1 17 41 22
 57  5]
[60 51 84 32 33 14 32 95 48 45 23 51 79 87 81 36 78 90 32 43 39 64 88 97
 23 67  2 83 40 48 50 15 44 13 15  0 41  1 68 32 42 80 63 39 19 11 11 32
  3 20]
```

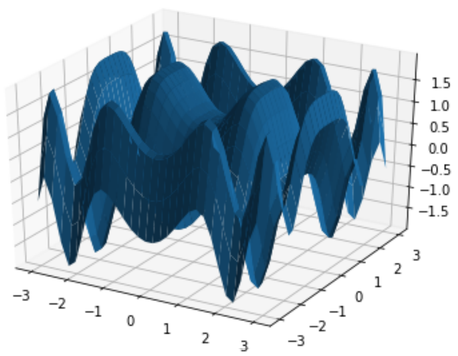
```
fig = plt.figure(figsize = (7, 4))
ax = plt.axes(projection ="3d")
ax.scatter3D(x, y, z, color = "green")

plt.title("simple 3D scatter plot")
plt.show()
```

simple 3D scatter plot

```
x = np.outer(np.linspace(-3, 3, 32), np.ones(32))
y = x.copy().T # transpose
z = (np.sin(x **2) + np.cos(y **2) )

print(x)
print(y)
print(z)
```

```
[[-3.         -3.         -3.         ... -3.         -3.
  -3.        ]
 [-2.80645161 -2.80645161 -2.80645161 ... -2.80645161 -2.80645161
  -2.80645161]
 [-2.61290323 -2.61290323 -2.61290323 ... -2.61290323 -2.61290323
  -2.61290323]
 ...
 [ 2.61290323  2.61290323  2.61290323 ...  2.61290323  2.61290323
   2.61290323]
 [ 2.80645161  2.80645161  2.80645161 ...  2.80645161  2.80645161
   2.80645161]
 [ 3.          3.          3.         ...  3.          3.
   3.        ]]
[[-3.         -2.80645161 -2.61290323 ...  2.61290323  2.80645161
   3.        ]
 [-3.         -2.80645161 -2.61290323 ...  2.61290323  2.80645161
   3.        ]
 [-3.         -2.80645161 -2.61290323 ...  2.61290323  2.80645161
   3.        ]
 ...
 [-3.         -2.80645161 -2.61290323 ...  2.61290323  2.80645161
   3.        ]
 [-3.         -2.80645161 -2.61290323 ...  2.61290323  2.80645161
   3.        ]
 [-3.         -2.80645161 -2.61290323 ...  2.61290323  2.80645161
   3.        ]]
[[-0.49901178  0.38993128  1.26772341 ...  1.26772341  0.38993128
  -0.49901178]
 [ 0.08862357  0.97756663  1.85535876 ...  1.85535876  0.97756663
   0.08862357]
 [-0.39350085  0.49544221  1.37323434 ...  1.37323434  0.49544221
  -0.39350085]
 ...
 [-0.39350085  0.49544221  1.37323434 ...  1.37323434  0.49544221
  -0.39350085]
 [ 0.08862357  0.97756663  1.85535876 ...  1.85535876  0.97756663
   0.08862357]
 [-0.49901178  0.38993128  1.26772341 ...  1.26772341  0.38993128
  -0.49901178]]
```

```
fig = plt.figure(figsize =(7, 5))
ax = plt.axes(projection ='3d')
ax.plot_surface(x, y, z)
plt.show()
```

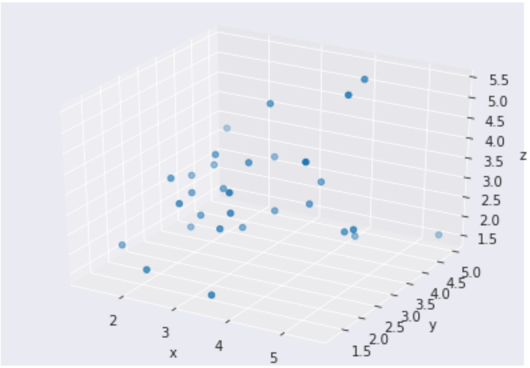



```
import seaborn as sns
sns.set_style ("darkgrid")

plot_mean = 3
min_num = 30
plot1 = np.random.normal (plot_mean, 1, size = min_num)
plot2 = np.random.normal (plot_mean, 1, size = min_num)
plot3 = np.random.normal (plot_mean, 1, size = min_num)

print(plot1)
print(plot2)
print(plot3)
```

```
[2.9259915  4.32866762 4.67625567 2.29690892 1.302331   5.45662561
 3.27418982 3.89056709 1.73547246 2.23482351 5.01669753 2.38295745
 1.83981805 2.84230473 3.45277658 3.30223874 1.62587773 2.08893361
 2.94654463 3.21663214 2.76932616 3.07814944 3.97987457 1.96989246
 1.94111184 4.46121116 4.39048688 4.19543621 2.50656238 2.35613056]
[3.01465101 3.89323102 3.277174   1.22372971 1.9682073  4.98734078
 2.1183886  3.8384081  3.34798375 3.29779094 2.97607297 2.04155946
 2.60923112 3.97270528 1.31147281 3.38415818 4.55914202 2.8262447
 3.17509235 2.22993355 4.21638949 2.13489301 3.34271441 3.64991246
 2.98898113 3.97583722 2.54991708 4.60536359 3.10440585 2.67142145]
[2.12580775 5.34798897 2.36687957 1.85240192 1.77781182 1.57922605
 3.57535154 3.12030713 2.96223362 3.64719369 2.68442291 3.09890409
 3.29413255 4.73947971 1.50290663 2.46109982 3.53984671 2.88395886
 3.6922279  2.99052694 3.26791366 2.60839804 2.84382089 3.14038618
 1.87461878 1.81114957 4.40003539 5.36678771 2.95675503 2.46603709]
```

```
plt.figure (figsize = (7,5))
seaborn_plot = plt.axes (projection='3d')
print (type (seaborn_plot))
seaborn_plot.scatter3D (plot1, plot2, plot3)
seaborn_plot.set_xlabel ('x')
seaborn_plot.set_ylabel ('y')
seaborn_plot.set_zlabel ('z')
plt.show ()
```

```
<class 'matplotlib.axes._subplots.Axes3DSubplot'>
```




```
# Two matrices
mx1 = np.array([[5, 10], [15, 20]])
mx2 = np.array([[25, 30], [35, 40]])

print("Matrix1 =\n",mx1)
print("\nMatrix2 =\n",mx2)
```

```
Matrix1 =
 [[ 5 10]
 [15 20]]

Matrix2 =
 [[25 30]
 [35 40]]
```

```
# The addition() is used to add matrices
print ("\nAddition of two matrices: ")
print (np.add(mx1,mx2))

# The subtract() is used to subtract matrices
print ("\nSubtraction of two matrices: ")
print (np.subtract(mx1,mx2))

# The divide() is used to divide matrices
print ("\nMatrix Division: ")
print (np.divide(mx1,mx2))

# The multiply()is used to multiply matrices
print ("\nMultiplication of two matrices: ")
print (np.multiply(mx1,mx2))
```

```
Addition of two matrices:
[[30 40]
 [50 60]]

Subtraction of two matrices:
[[-20 -20]
 [-20 -20]]

Matrix Division:
```

```
[[0.2        0.33333333]
 [0.42857143 0.5       ]]

Multiplication of two matrices:
[[125 300]
 [525 800]]
```

```
mx = np.array([[5, 10], [15, 20]])
print("Matrix =\n",mx)

print ("\n last element from each row.")
for i in range(len(mx)):
    print(mx[i][-1])

print ("\nThe summation of elements=")
print (np.sum(mx))

print ("\nThe column wise summation=")
print (np.sum(mx,axis=0))

print ("\nThe row wise summation=")
print (np.sum(mx,axis=1))

print ("\nThe Transpose =")
print (mx.T)
```

```
Matrix =
 [[ 5 10]
 [15 20]]

 last element from each row.
10
20

The summation of elements=
50

The column wise summation=
[20 30]

The row wise summation=
[15 35]

The Transpose =
[[ 5 15]
 [10 20]]
```