Webinar_HandsOn_Part1

July 21, 2019

```
In [1]: import numpy as np

In [2]: a=np.array([1,2,3])   #create a rank 1 array

In [3]: type(a)

Out[3]: numpy.ndarray

In [4]: a.shape

Out[4]: (3,)

In [5]: print(a[0],a[1],a[2])

1 2 3


In [6]: a[0]=7

In [7]: a

Out[7]: array([7, 2, 3])

In [8]: b=np.array([[1,2,3],[4,5,6]]) #create a rank 2 array

In [9]: b.shape

Out[9]: (2, 3)

In [10]: print(b[0,0],b[0,1],b[1,0])

1 2 4


In [11]: #Various functions

In [12]: a=np.zeros((3,3))

In [13]: a
```

```
Out[13]: array([[ 0.,  0.,  0.],
                [ 0.,  0.,  0.],
                [ 0.,  0.,  0.]])

In [14]: b=np.ones((1,2))

In [15]: b

Out[15]: array([[ 1.,  1.]])

In [16]: c=np.full((4,4),8)
         c

Out[16]: array([[8, 8, 8, 8],
                [8, 8, 8, 8],
                [8, 8, 8, 8],
                [8, 8, 8, 8]])

In [17]: np.eye(4)

Out[17]: array([[ 1.,  0.,  0.,  0.],
                [ 0.,  1.,  0.,  0.],
                [ 0.,  0.,  1.,  0.],
                [ 0.,  0.,  0.,  1.]])

In [18]: np.random.random((2,2))

Out[18]: array([[ 0.52855236,  0.43899333],
                [ 0.76735095,  0.86328466]])

In [19]: x=np.ones((4,4))

In [20]: x

Out[20]: array([[ 1.,  1.,  1.,  1.],
                [ 1.,  1.,  1.,  1.],
                [ 1.,  1.,  1.,  1.],
                [ 1.,  1.,  1.,  1.]])

In [22]: y=np.empty_like(x)

In [23]: y

Out[23]: array([[ 1.,  1.,  1.,  1.],
                [ 1.,  1.,  1.,  1.],
                [ 1.,  1.,  1.,  1.],
                [ 1.,  1.,  1.,  1.]])

In [24]: a=np.array([[0,1,2,3],[4,5,6,7],[8,9,10,11]]) # rank 3 matrix
```

```
In [25]: b=a[:2,1:3]
         b

Out[25]: array([[1, 2],
                [5, 6]])

In [26]: print(a[1,1])

5


In [27]: b[0,1]=100

In [28]: print(b)

[[  1 100]
 [  5   6]]


In [29]: A=np.array([[1,2,3],[4,5,6],[7,8,9]])
         A

Out[29]: array([[1, 2, 3],
                [4, 5, 6],
                [7, 8, 9]])

In [30]: b=np.array([0,2,0])

In [31]: np.arange(4)

Out[31]: array([0, 1, 2, 3])

In [33]: a[np.arange(3),b]

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

In [34]: a[np.arange(3),b]+=5

In [35]: a

Out[35]: array([[  5,   1, 100,   3],
                [  4,   5,  11,   7],
                [ 13,   9,  10,  11]])

In [36]: a=np.array([[1,2],[3,4],[5,6]])
         a

Out[36]: array([[1, 2],
                [3, 4],
                [5, 6]])
```

```
In [37]: boolean_index= a>2

In [38]: print(boolean_index)

[[False False]
 [ True  True]
 [ True  True]]


In [39]: a[boolean_index]

Out[39]: array([3, 4, 5, 6])

In [40]: a[a>2]

Out[40]: array([3, 4, 5, 6])

In [55]: A=np.array([[1,2,3],[4,5,6],[7,8,9],[10,11,12]])
         A

Out[55]: array([[ 1,  2,  3],
                [ 4,  5,  6],
                [ 7,  8,  9],
                [10, 11, 12]])

In [56]: b=np.array([1,1,1,1])

In [59]: t=np.arange(4)
         t

Out[59]: array([0, 1, 2, 3])

In [60]: A[t,b]   # t= 0,1,2,3  b= 1,1 ,1 ,1

Out[60]: array([ 2,  5,  8, 11])

In [61]: A[np.arange(4),b]+=100

In [62]: A

Out[62]: array([[  1, 102,   3],
                [  4, 105,   6],
                [  7, 108,   9],
                [ 10, 111,  12]])

In [63]: x=np.array([1,2])
         print(x.dtype)

int64
```

```
In [64]: x=np.array([1.0,2.0])
         print(x.dtype)

float64


In [65]: x=np.array([1,2],dtype=np.int64)
         print(x.dtype)

int64


In [66]: x=np.array([1,2,3,4,5,6])
         y=np.array([1,2,3,4,5,6])
         np.add(x,y)

Out[66]: array([ 2,  4,  6,  8, 10, 12])

In [67]: np.subtract(x,y)

Out[67]: array([0, 0, 0, 0, 0, 0])

In [68]: np.multiply(x,y)

Out[68]: array([ 1,  4,  9, 16, 25, 36])

In [69]: np.divide(x,y)

Out[69]: array([ 1.,  1.,  1.,  1.,  1.,  1.])

In [70]: np.sqrt(x)

Out[70]: array([ 1.        ,  1.41421356,  1.73205081,  2.        ,  2.23606798,
                2.44948974])

In [71]: np.dot(x,y)

Out[71]: 91

In [72]: x=np.arange(0,2*np.pi,0.1)
         x

Out[72]: array([ 0. ,  0.1,  0.2,  0.3,  0.4,  0.5,  0.6,  0.7,  0.8,  0.9,  1. ,
                1.1,  1.2,  1.3,  1.4,  1.5,  1.6,  1.7,  1.8,  1.9,  2. ,  2.1,
                2.2,  2.3,  2.4,  2.5,  2.6,  2.7,  2.8,  2.9,  3. ,  3.1,  3.2,
                3.3,  3.4,  3.5,  3.6,  3.7,  3.8,  3.9,  4. ,  4.1,  4.2,  4.3,
                4.4,  4.5,  4.6,  4.7,  4.8,  4.9,  5. ,  5.1,  5.2,  5.3,  5.4,
                5.5,  5.6,  5.7,  5.8,  5.9,  6. ,  6.1,  6.2])

In [73]: np.sin(x)
```

```
Out[73]: array([ 0.        ,  0.09983342,  0.19866933,  0.29552021,  0.38941834,
                 0.47942554,  0.56464247,  0.64421769,  0.71735609,  0.78332691,
                 0.84147098,  0.89120736,  0.93203909,  0.96355819,  0.98544973,
                 0.99749499,  0.9995736 ,  0.99166481,  0.97384763,  0.94630009,
                 0.90929743,  0.86320937,  0.8084964 ,  0.74570521,  0.67546318,
                 0.59847214,  0.51550137,  0.42737988,  0.33498815,  0.23924933,
                 0.14112001,  0.04158066, -0.05837414, -0.15774569, -0.2555411 ,
                -0.35078323, -0.44252044, -0.52983614, -0.61185789, -0.68776616,
                -0.7568025 , -0.81827711, -0.87157577, -0.91616594, -0.95160207,
                -0.97753012, -0.993691  , -0.99992326, -0.99616461, -0.98245261,
                -0.95892427, -0.92581468, -0.88345466, -0.83226744, -0.77276449,
                -0.70554033, -0.63126664, -0.55068554, -0.46460218, -0.37387666,
                -0.2794155 , -0.1821625 , -0.0830894 ])

In [74]: np.cos(x)

Out[74]: array([ 1.        ,  0.99500417,  0.98006658,  0.95533649,  0.92106099,
                 0.87758256,  0.82533561,  0.76484219,  0.69670671,  0.62160997,
                 0.54030231,  0.45359612,  0.36235775,  0.26749883,  0.16996714,
                 0.0707372 , -0.02919952, -0.12884449, -0.22720209, -0.32328957,
                -0.41614684, -0.5048461 , -0.58850112, -0.66627602, -0.73739372,
                -0.80114362, -0.85688875, -0.90407214, -0.94222234, -0.97095817,
                -0.9899925 , -0.99913515, -0.99829478, -0.98747977, -0.96679819,
                -0.93645669, -0.89675842, -0.84810003, -0.79096771, -0.7259323 ,
                -0.65364362, -0.57482395, -0.49026082, -0.40079917, -0.30733287,
                -0.2107958 , -0.11215253, -0.01238866,  0.08749898,  0.18651237,
                 0.28366219,  0.37797774,  0.46851667,  0.55437434,  0.63469288,
                 0.70866977,  0.77556588,  0.83471278,  0.88551952,  0.92747843,
                 0.96017029,  0.98326844,  0.9965421 ])

In [75]: np.tan(x)

Out[75]: array([  0.00000000e+00,   1.00334672e-01,   2.02710036e-01,
                  3.09336250e-01,   4.22793219e-01,   5.46302490e-01,
                  6.84136808e-01,   8.42288380e-01,   1.02963856e+00,
                  1.26015822e+00,   1.55740772e+00,   1.96475966e+00,
                  2.57215162e+00,   3.60210245e+00,   5.79788372e+00,
                  1.41014199e+01,  -3.42325327e+01,  -7.69660214e+00,
                 -4.28626167e+00,  -2.92709751e+00,  -2.18503986e+00,
                 -1.70984654e+00,  -1.37382306e+00,  -1.11921364e+00,
                 -9.16014290e-01,  -7.47022297e-01,  -6.01596613e-01,
                 -4.72727629e-01,  -3.55529832e-01,  -2.46405394e-01,
                 -1.42546543e-01,  -4.16166546e-02,   5.84738545e-02,
                  1.59745748e-01,   2.64316901e-01,   3.74585640e-01,
                  4.93466730e-01,   6.24733075e-01,   7.73556091e-01,
                  9.47424650e-01,   1.15782128e+00,   1.42352648e+00,
                  1.77777977e+00,   2.28584788e+00,   3.09632378e+00,
                  4.63733205e+00,   8.86017490e+00,   8.07127630e+01,
                 -1.13848707e+01,  -5.26749307e+00,  -3.38051501e+00,
```

```
                -2.44938942e+00,  -1.88564188e+00,  -1.50127340e+00,
                -1.21754082e+00,  -9.95584052e-01,  -8.13943284e-01,
                -6.59730572e-01,  -5.24666222e-01,  -4.03110900e-01,
                -2.91006191e-01,  -1.85262231e-01,  -8.33777149e-02])

In [76]: x=np.array([[1,2],[3,4],[5,6]])

In [77]: x

Out[77]: array([[1, 2],
                [3, 4],
                [5, 6]])

In [78]: x.T

Out[78]: array([[1, 3, 5],
                [2, 4, 6]])

In [79]: x=np.arange(10)

In [80]: x

Out[80]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In [81]: x.reshape((5,2))

Out[81]: array([[0, 1],
                [2, 3],
                [4, 5],
                [6, 7],
                [8, 9]])

In [82]: t=x.reshape((5,2))

In [83]: t

Out[83]: array([[0, 1],
                [2, 3],
                [4, 5],
                [6, 7],
                [8, 9]])

In [84]: np.ravel(t)

Out[84]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In [85]: start=np.zeros((4,3))

In [87]: add_rows=np.array([[1,0,2]])

In [88]: y=start+add_rows
```

```
In [89]: y
```

```
Out[89]: array([[ 1.,  0.,  2.],
                [ 1.,  0.,  2.],
                [ 1.,  0.,  2.],
                [ 1.,  0.,  2.]])
```

```
In [90]: add_cols=np.array([[0,1,2,3]])
         add_cols=add_cols.T
         print(add_cols)
```

```
[[0]
 [1]
 [2]
 [3]]
```

```
In [91]: #broadcasting in both dimensions
```

```
In [92]: add_scala=np.array([1])
         print(start+add_scala)
```

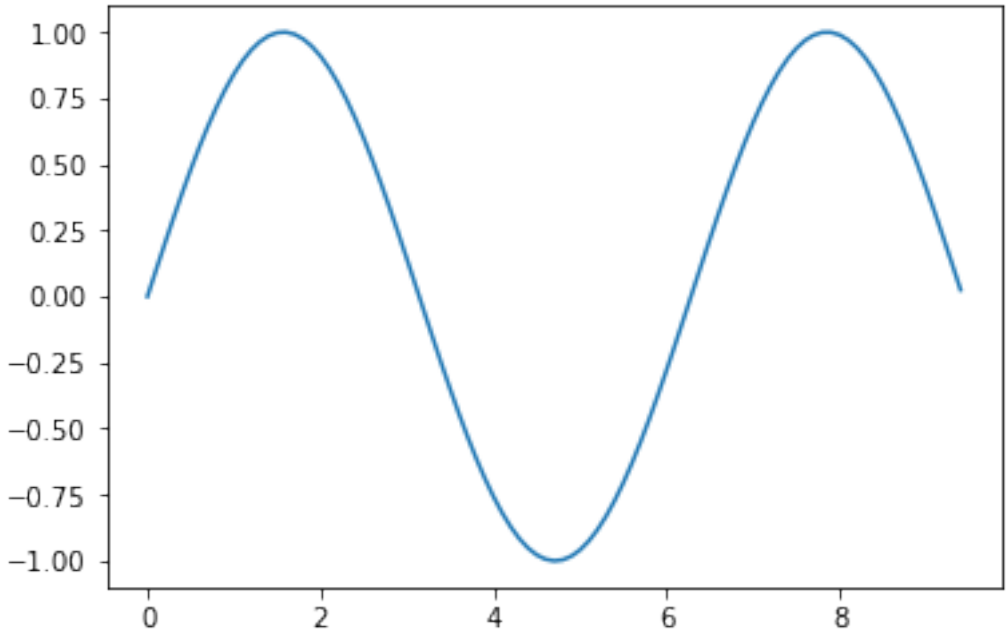
```
[[ 1.  1.  1.]
 [ 1.  1.  1.]
 [ 1.  1.  1.]
 [ 1.  1.  1.]]
```

```
In [93]: import matplotlib.pyplot as plt
```

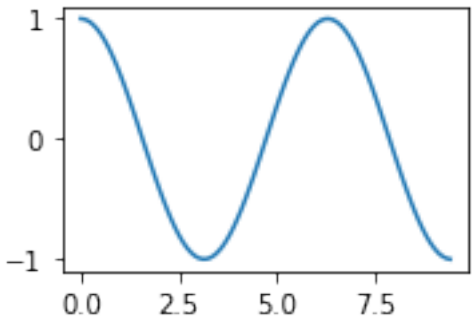
```
In [94]: x=np.arange(0,3*np.pi,0.1)
         x
```

```
Out[94]: array([ 0. ,  0.1,  0.2,  0.3,  0.4,  0.5,  0.6,  0.7,  0.8,  0.9,  1. ,
                 1.1,  1.2,  1.3,  1.4,  1.5,  1.6,  1.7,  1.8,  1.9,  2. ,  2.1,
                 2.2,  2.3,  2.4,  2.5,  2.6,  2.7,  2.8,  2.9,  3. ,  3.1,  3.2,
                 3.3,  3.4,  3.5,  3.6,  3.7,  3.8,  3.9,  4. ,  4.1,  4.2,  4.3,
                 4.4,  4.5,  4.6,  4.7,  4.8,  4.9,  5. ,  5.1,  5.2,  5.3,  5.4,
                 5.5,  5.6,  5.7,  5.8,  5.9,  6. ,  6.1,  6.2,  6.3,  6.4,  6.5,
                 6.6,  6.7,  6.8,  6.9,  7. ,  7.1,  7.2,  7.3,  7.4,  7.5,  7.6,
                 7.7,  7.8,  7.9,  8. ,  8.1,  8.2,  8.3,  8.4,  8.5,  8.6,  8.7,
                 8.8,  8.9,  9. ,  9.1,  9.2,  9.3,  9.4])
```

```
In [96]: y=np.sin(x)
         plt.plot(x,y)
         plt.show()
```

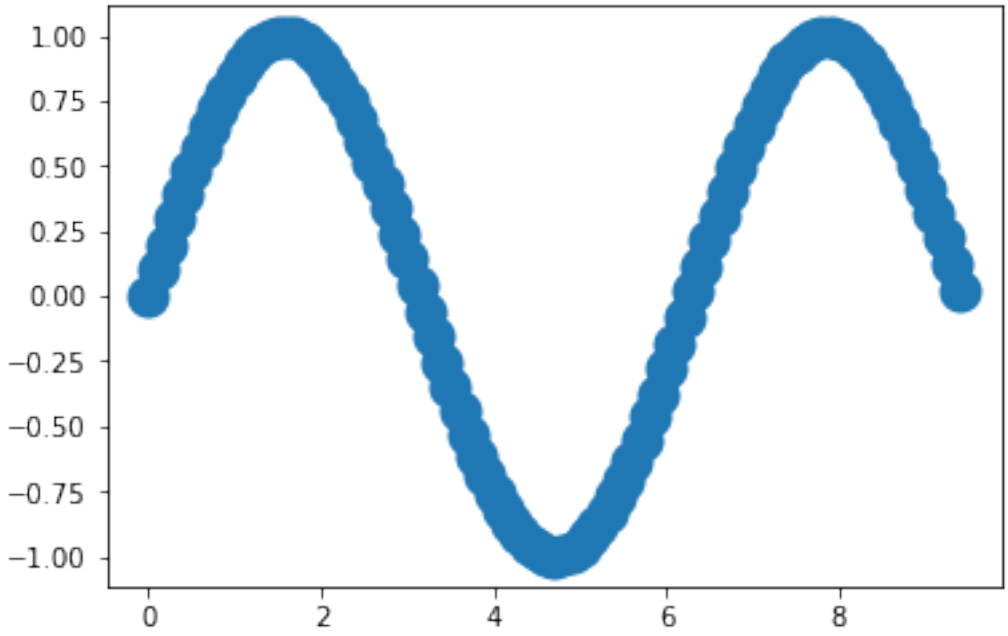


In [109]:
```
y_sin=np.sin(x)
y_cos=np.cos(x)
plt.plot(x,y_sin,x,y_cos)
plt.xlabel("x axis label")
plt.ylabel("y axis label")
plt.title("Sine and cosine graph")
plt.legend(['sine','cosine'])
plt.subplot(2,1,1)
plt.plot(x,y_sin)
plt.subplot(2,2,2)
plt.plot(x,y_cos)
plt.show()
```



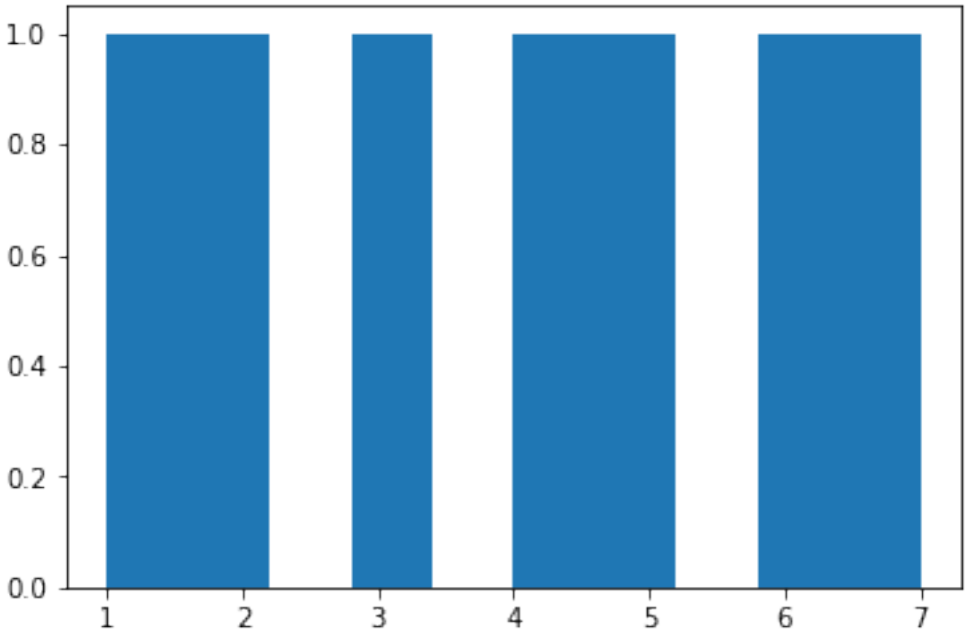
```
In [112]: plt.scatter(x,y_sin,linewidth=10)

Out[112]: <matplotlib.collections.PathCollection at 0x7fd6fdacf320>
```



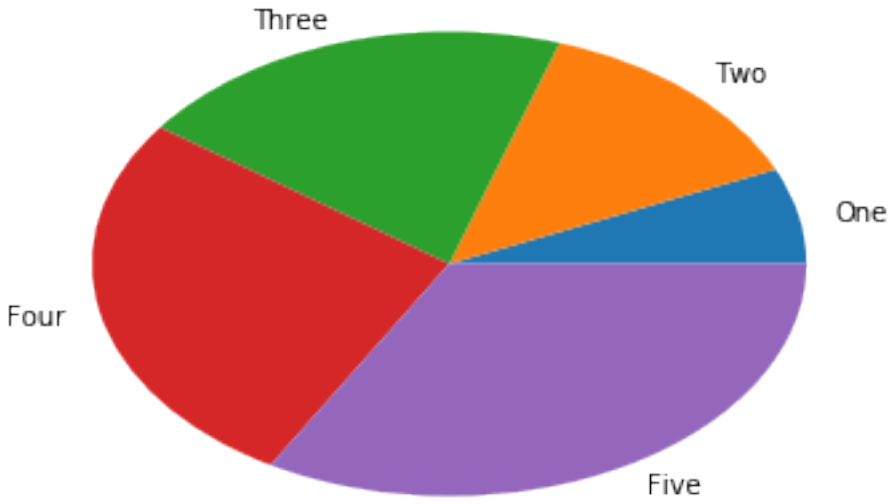

```
In [113]: x=np.array([1,2,3,4,5,6,7])
          plt.hist(x)

Out[113]: (array([ 1.,  1.,  0.,  1.,  0.,  1.,  1.,  0.,  1.,  1.]),
           array([ 1. ,  1.6,  2.2,  2.8,  3.4,  4. ,  4.6,  5.2,  5.8,  6.4,  7. ]),
           <a list of 10 Patch objects>)
```




```
In [114]: x=np.array([1,2,3,4,5])
          labels=["One","Two","Three","Four","Five"]
          plt.pie(x,labels=labels)

Out[114]: ([<matplotlib.patches.Wedge at 0x7fd6fdb70748>,
            <matplotlib.patches.Wedge at 0x7fd6fdb70c18>,
            <matplotlib.patches.Wedge at 0x7fd6fdb64198>,
            <matplotlib.patches.Wedge at 0x7fd6fdb646d8>,
            <matplotlib.patches.Wedge at 0x7fd6fdb64c18>],
           [Text(1.07596,0.228703,'One'),
            Text(0.736044,0.817459,'Two'),
            Text(-0.339919,1.04616,'Three'),
            Text(-1.07596,-0.228703,'Four'),
            Text(0.55,-0.952628,'Five')])
```




```
In [115]: import pandas as pd

In [116]: series=pd.Series([1,2,3,4,5])
          series

Out[116]: 0    1
          1    2
          2    3
          3    4
          4    5
          dtype: int64

In [119]: df=pd.DataFrame([1,2,3,4,5])
          df

Out[119]:    0
          0  1
          1  2
          2  3
          3  4
          4  5

In [120]: s=pd.Series([1,3,5,np.nan,6,8])

In [121]: s

Out[121]: 0    1.0
          1    3.0
```

```
2    5.0
3    NaN
4    6.0
5    8.0
dtype: float64
```

```
In [122]: dates=pd.date_range('20190720',periods=5)
          dates
```

```
Out[122]: DatetimeIndex(['2019-07-20', '2019-07-21', '2019-07-22', '2019-07-23',
                         '2019-07-24'],
                        dtype='datetime64[ns]', freq='D')
```

```
In [124]: pd.Timestamp('20190720')
```

```
Out[124]: Timestamp('2019-07-20 00:00:00')
```

```
In [125]: #importing a dataframe
```

```
In [126]: # csv( comma seperate values, ), txt , xlsx ,
```

```
In [127]: cd
```

```
/home/subarna1
```

```
In [128]: cd Downloads
```

```
/home/subarna1/Downloads
```

```
In [130]: df=pd.read_csv('gpa.csv')
```

```
In [132]: #viewing data
```

```
In [138]: df.shape
```

```
Out[138]: (1000, 2)
```

```
In [139]: # Thousand rows and 2 columns
```

```
In [140]: df.describe()
```

```
Out[140]:             HS GPA    SAT Score
          count  1000.000000  1000.000000
          mean      3.203700  1033.290000
          std       0.542541   142.873681
          min       1.800000   530.000000
          25%       2.800000   930.000000
          50%       3.200000  1030.000000
          75%       3.700000  1130.000000
          max       4.500000  1440.000000
```

In [143]:

Out[143]:

	SAT Score	HS GPA
0	1270	3.4
1	1220	4.0
2	1160	3.8
3	950	3.8
4	1070	4.0
5	1110	4.0
6	1220	2.8
7	1150	3.8
8	1440	4.0
9	850	2.6
10	1280	3.8
11	1020	3.8
12	880	2.8
13	1080	3.8
14	1200	3.5
15	1120	3.8
16	970	3.2
17	1210	4.0
18	1030	3.8
19	1020	4.0
20	1400	4.0
21	720	3.3
22	1170	3.9
23	1070	3.5
24	860	3.5
25	920	2.0
26	1350	4.0
27	910	2.3
28	980	3.5
29	800	3.5
..	...	...
970	860	3.5
971	950	3.0
972	1050	3.3
973	1160	3.0
974	740	2.9
975	1030	3.7
976	1020	2.5
977	910	3.0
978	980	3.0
979	890	2.5
980	700	2.3
981	970	2.5
982	1130	2.2
983	1100	3.0

```
984        880     3.1
985       1190     3.5
986       1180     3.2
987        990     2.8
988       1020     3.5
989       1060     4.0
990       1010     4.0
991       1410     4.0
992        880     2.5
993       1140     3.0
994       1040     3.0
995       1000     3.7
996       1080     3.3
997       1140     3.5
998       1200     2.3
999        930     2.7

[1000 rows x 2 columns]
```

In [ ]: