Untitled2

February 18, 2020

1 Data Analysis with Numpy - - - -

[1]:
```
lis =[1,2,3,4]
```

[2]:
```
import numpy as np
```

[3]:
```
lis
```

[3]:
```
[1, 2, 3, 4]
```

[4]:
```
arr = np.array(lis)
```

[5]:
```
arr
```

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

[6]:
```
mat = [[1,2,3], [4,5,6],[7,8,9]]
```

[7]:
```
np.array(mat)
```

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

[10]:
```
np.arange(0,11)
```

[10]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

[11]:
```
np.arange(0,11,2) # show all even number between range-
```

[11]:
```
array([ 0,  2,  4,  6,  8, 10])
```

[12]:
```
np.zeros(3) # to generate all zeros
```

[12]:
```
array([0., 0., 0.])
```

[13]:
```
 np.zeros((5,5)) # number of rows = number of column
```

[13]:
```
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.]])
```

[14]:
```
np.zeros((2,5)) # 2-rows and 5 columns
```

[14]:
```
array([[0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.]])
```

[15]:
```
np.ones(4)
```

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

[16]:
```
np.ones((3,4))
```

[16]:
```
array([[1., 1., 1., 1.],
       [1., 1., 1., 1.],
       [1., 1., 1., 1.]])
```

[17]:
```
np.linspace(0,5,10) # Gotland space
```

[17]:
```
array([0.        , 0.55555556, 1.11111111, 1.66666667, 2.22222222,
       2.77777778, 3.33333333, 3.88888889, 4.44444444, 5.        ])
```

[18]:
```
np.eye(4) # generate square matrix
```

[18]:
```
array([[1., 0., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 0., 1.]])
```

[19]:
```
np.random.rand(5)
```

[19]:
```
array([0.14719321, 0.3491152 , 0.79327175, 0.74623728, 0.39561139])
```

[20]:
```
 np.random.rand(5,5)
```

[20]:
```
array([[0.71850713, 0.68143583, 0.73883599, 0.88103162, 0.59861614],
       [0.61663485, 0.19164594, 0.94667393, 0.96647756, 0.15895158],
       [0.78358847, 0.91989626, 0.48133994, 0.72910334, 0.08784452],
       [0.92690313, 0.41784179, 0.21133948, 0.15849021, 0.29465182],
       [0.16541965, 0.03198181, 0.09617806, 0.92573836, 0.666444  ]])
```

[21]:
```
np.random.randn(2) # random numbers
```

[21]:
```
array([ 0.55136501, -0.05148369])
```

[22]:
```
np.random.rand(4,4) # 4-4 matrix
```

[22]:
```
array([[0.31257332, 0.35846895, 0.25628222, 0.00504747],
       [0.65809425, 0.17781795, 0.84403242, 0.51885589],
       [0.37636886, 0.98435047, 0.55139474, 0.16950692],
       [0.60744089, 0.21652615, 0.40482186, 0.34767082]])
```

[23]:
```
np.random.randint(1,100,10 ) # low- inclusive, high- exclusive
```

[23]:
```
array([61, 30, 89, 59, 12, 90, 52, 47, 44, 31])
```

[24]:
```
arr= np.arange(25)
```

[25]:
```
arr
```

[25]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20, 21, 22, 23, 24])
```

[26]:
```
ranarr = np.random.randint(0,50,10) # could be 10 integers there
```

[27]:
```
ranarr
```

[27]:
```
array([ 7, 34,  2,  5, 42, 48, 25,  3, 23, 18])
```

[28]:
```
arr.reshape(5,5)
```

[28]:
```
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19],
       [20, 21, 22, 23, 24]])
```

[29]:
```
ranarr.max() #find max value
```

[29]:
```
48
```

[30]:
```
ranarr.min() #find min value
```

[30]:
```
2
```

[31]:
```
ranarr
```

[31]:
```
array([ 7, 34,  2,  5, 42, 48, 25,  3, 23, 18])
```

[32]:
```
ranarr.argmax() # find the location of the max value
```

[32]: 5

[33]:
```
ranarr.argmin() # find the location of the min value
```

[33]: 2

[34]:
```
arr.shape
```

[34]: (25,)

[35]:
```
arr.dtype #data type
```

[35]: dtype('int64')

[36]:
```
from numpy.random import randint
```

[37]:
```
randint(2,10)
```

[37]: 9

2 NumPy Indexing and Selection

[38]:
```
arr = np.arange(1,11)
```

[39]:
```
arr
```

[39]: array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])

[40]:
```
arr[8]
```

[40]: 9

[41]:
```
arr[1:6]
```

[41]: array([2, 3, 4, 5, 6])

[42]:
```
arr[:6]
```

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

[43]:
```
arr[5:]
```

[43]: array([6, 7, 8, 9, 10])

[44]:
```
arr[0:5]= 100 #braodcast first 5 digit
```

[45]:
```
arr
```

[45]:
```
array([100, 100, 100, 100, 100,   6,   7,   8,   9,  10])
```

[46]:
```
arr = np.arange(1,11)
```

[47]:
```
slice_arr = arr[0:6]
```

[48]:
```
slice_arr
```

[48]:
```
array([1, 2, 3, 4, 5, 6])
```

[54]:
```
import numpy as np

ar_2d = np.array([[5,10,15],[20,30,40],[25,35,45]])
```

[55]:
```
ar_2d
```

[55]:
```
array([[ 5, 10, 15],
       [20, 30, 40],
       [25, 35, 45]])
```

[56]:
```
ar_2d[0][0] # row... then column
```

[56]:
```
5
```

[57]:
```
ar_2d[1][1]
```

[57]:
```
30
```

[58]:
```
ar_2d[:2,1:] #grab data top of the right corner
```

[58]:
```
array([[10, 15],
       [30, 40]])
```

[59]:
```
# conditonal array

arr = np.arange(1,11)
```

[60]:
```
arr
```

[60]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

[66]:
```
bool_arr = arr>5
```

[67]:
```
bool_arr
```

[67]:
```
array([False, False, False, False, False,  True,  True,  True,  True,
       True])
```

[68]:
```
arr[bool_arr]
```

[68]:
```
array([ 6,  7,  8,  9, 10])
```

[70]:
```
arr[arr<3]
```

[70]:
```
array([1, 2])
```

[71]:
```
arr_2d = np.arange(50).reshape(5,10)
```

[72]:
```
arr_2d
```

[72]:
```
array([[ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14, 15, 16, 17, 18, 19],
       [20, 21, 22, 23, 24, 25, 26, 27, 28, 29],
       [30, 31, 32, 33, 34, 35, 36, 37, 38, 39],
       [40, 41, 42, 43, 44, 45, 46, 47, 48, 49]])
```

[74]:
```
arr_2d[1:3,3:5]
```

[74]:
```
array([[13, 14],
       [23, 24]])
```

2.0.1 Array with array

2.0.2 Array with scalars

2.0.3 Universal array functions

[76]:
```
arr = np.arange(0,11)
```

[77]:
```
arr
```

[77]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

[78]:
```
arr+arr
```

[78]:
```
array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18, 20])
```

[79]:
```
arr-arr
```

[79]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

[80]:
```
arr*arr
```

[80]:
```
array([  0,   1,   4,   9,  16,  25,  36,  49,  64,  81, 100])
```

[81]:
```
arr+100 # scalar
```

[81]:
```
array([100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110])
```

[82]:
```
arr/arr
```

```
/home/ikhtiar/.local/lib/python3.6/site-packages/ipykernel_launcher.py:1:
RuntimeWarning: invalid value encountered in true_divide
  """Entry point for launching an IPython kernel.
```

[82]:
```
array([nan,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.])
```

[83]:
```
arr **2  # square
```

[83]:
```
array([  0,   1,   4,   9,  16,  25,  36,  49,  64,  81, 100])
```

[84]:
```
np.sqrt(arr)
```

[84]:
```
array([0.        , 1.        , 1.41421356, 1.73205081, 2.        ,
       2.23606798, 2.44948974, 2.64575131, 2.82842712, 3.        ,
       3.16227766])
```

[85]:
```
np.max(arr)
```

[85]:
```
10
```

[86]:
```
np.sin(arr) # numpy functions
```

[86]:
```
array([ 0.        ,  0.84147098,  0.90929743,  0.14112001, -0.7568025 ,
       -0.95892427, -0.2794155 ,  0.6569866 ,  0.98935825,  0.41211849,
       -0.54402111])
```

[87]:
```
np.log(arr) # numpy functions
```

```
/home/ikhtiar/.local/lib/python3.6/site-packages/ipykernel_launcher.py:1:
RuntimeWarning: divide by zero encountered in log
  """Entry point for launching an IPython kernel.
```

[87]:
```
array([      -inf, 0.        , 0.69314718, 1.09861229, 1.38629436,
       1.60943791, 1.79175947, 1.94591015, 2.07944154, 2.19722458,
       2.30258509])
```

[88]:
```
np.exp(arr)
```

[88]:
```
array([1.00000000e+00, 2.71828183e+00, 7.38905610e+00, 2.00855369e+01,
       5.45981500e+01, 1.48413159e+02, 4.03428793e+02, 1.09663316e+03,
       2.98095799e+03, 8.10308393e+03, 2.20264658e+04])
```

3 NumPy Exercises -

[89]:
```
import numpy as np
```

[90]:
```
np.zeros(10)
```

[90]:
```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

[91]:
```
np.ones(10)
```

[91]:
```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

[92]:
```
np.ones(10)*5
```

[92]:
```
array([5., 5., 5., 5., 5., 5., 5., 5., 5., 5.])
```

[93]:
```
np.zeros(10)+5
```

[93]:
```
array([5., 5., 5., 5., 5., 5., 5., 5., 5., 5.])
```

[94]:
```
np.arange(10,51)
```

[94]:
```
array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26,
       27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43,
       44, 45, 46, 47, 48, 49, 50])
```

[97]:
```
np.arange(10,51,2)
```

[97]:
```
array([10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
       44, 46, 48, 50])
```

[99]:
```
np.arange(9).reshape(3,3)
```

[99]:
```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

[102]:
```
a = np.arange(9)
a.reshape(3,3)
```

[102]:
```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

[103]:
```
np.eye(3) # idendtity matrix
```

[103]:
```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

[104]:
```
np.random.rand(1) #create random numbers
```

[104]:
```
array([0.29960015])
```

[105]:
```
np.random.randn(25)
```

[105]:
```
array([ 2.0566654 , -0.12133849,  0.77079527, -1.18737538, -1.227352  ,
       -0.37107253, -0.76899888, -1.01454637,  0.48949841,  0.98624969,
       -0.58206995,  0.86142424,  1.18754393, -1.86192526, -0.48344073,
        0.25706426,  0.64304915, -0.19731156,  0.30462352,  0.04442142,
       -0.45572889,  0.01835032, -0.10896754, -0.31602974,  0.48056278])
```

[108]:
```
np.arange(1,101).reshape(10,10)/100
```

[108]:
```
array([[0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1 ],
       [0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19, 0.2 ],
       [0.21, 0.22, 0.23, 0.24, 0.25, 0.26, 0.27, 0.28, 0.29, 0.3 ],
       [0.31, 0.32, 0.33, 0.34, 0.35, 0.36, 0.37, 0.38, 0.39, 0.4 ],
       [0.41, 0.42, 0.43, 0.44, 0.45, 0.46, 0.47, 0.48, 0.49, 0.5 ],
       [0.51, 0.52, 0.53, 0.54, 0.55, 0.56, 0.57, 0.58, 0.59, 0.6 ],
       [0.61, 0.62, 0.63, 0.64, 0.65, 0.66, 0.67, 0.68, 0.69, 0.7 ],
       [0.71, 0.72, 0.73, 0.74, 0.75, 0.76, 0.77, 0.78, 0.79, 0.8 ],
       [0.81, 0.82, 0.83, 0.84, 0.85, 0.86, 0.87, 0.88, 0.89, 0.9 ],
       [0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98, 0.99, 1.  ]])
```

[109]:
```
np.linspace(0.01,1,100).reshape(10,10)
```

[109]:
```
array([[0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1 ],
       [0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19, 0.2 ],
       [0.21, 0.22, 0.23, 0.24, 0.25, 0.26, 0.27, 0.28, 0.29, 0.3 ],
       [0.31, 0.32, 0.33, 0.34, 0.35, 0.36, 0.37, 0.38, 0.39, 0.4 ],
       [0.41, 0.42, 0.43, 0.44, 0.45, 0.46, 0.47, 0.48, 0.49, 0.5 ],
       [0.51, 0.52, 0.53, 0.54, 0.55, 0.56, 0.57, 0.58, 0.59, 0.6 ],
       [0.61, 0.62, 0.63, 0.64, 0.65, 0.66, 0.67, 0.68, 0.69, 0.7 ],
       [0.71, 0.72, 0.73, 0.74, 0.75, 0.76, 0.77, 0.78, 0.79, 0.8 ],
       [0.81, 0.82, 0.83, 0.84, 0.85, 0.86, 0.87, 0.88, 0.89, 0.9 ],
       [0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98, 0.99, 1.  ]])
```

[110]:
```
np.linspace(0,1,20) #create space
```

[110]:
```
array([0.        , 0.05263158, 0.10526316, 0.15789474, 0.21052632,
       0.26315789, 0.31578947, 0.36842105, 0.42105263, 0.47368421,
       0.52631579, 0.57894737, 0.63157895, 0.68421053, 0.73684211,
       0.78947368, 0.84210526, 0.89473684, 0.94736842, 1.        ])
```

[112]:
```
mat = np.arange(1,26).reshape(5,5)
```

[114]:
```
mat
```

[114]:
```
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10],
       [11, 12, 13, 14, 15],
       [16, 17, 18, 19, 20],
       [21, 22, 23, 24, 25]])
```

[115]:
```
mat[2:,1:]
```

[115]:
```
array([[12, 13, 14, 15],
       [17, 18, 19, 20],
       [22, 23, 24, 25]])
```

[116]:
```
mat[3,4]
```

[116]:
```
20
```

[117]:
```
mat[:3,1]
```

[117]:
```
array([ 2,  7, 12])
```

[119]:
```
mat[-1:]
```

[119]:
```
array([[21, 22, 23, 24, 25]])
```

[120]:
```
np.sum(mat) # mat.sum()- get same answer
```

[120]:
```
325
```

[121]:
```
mat.std() # standard deviation
```

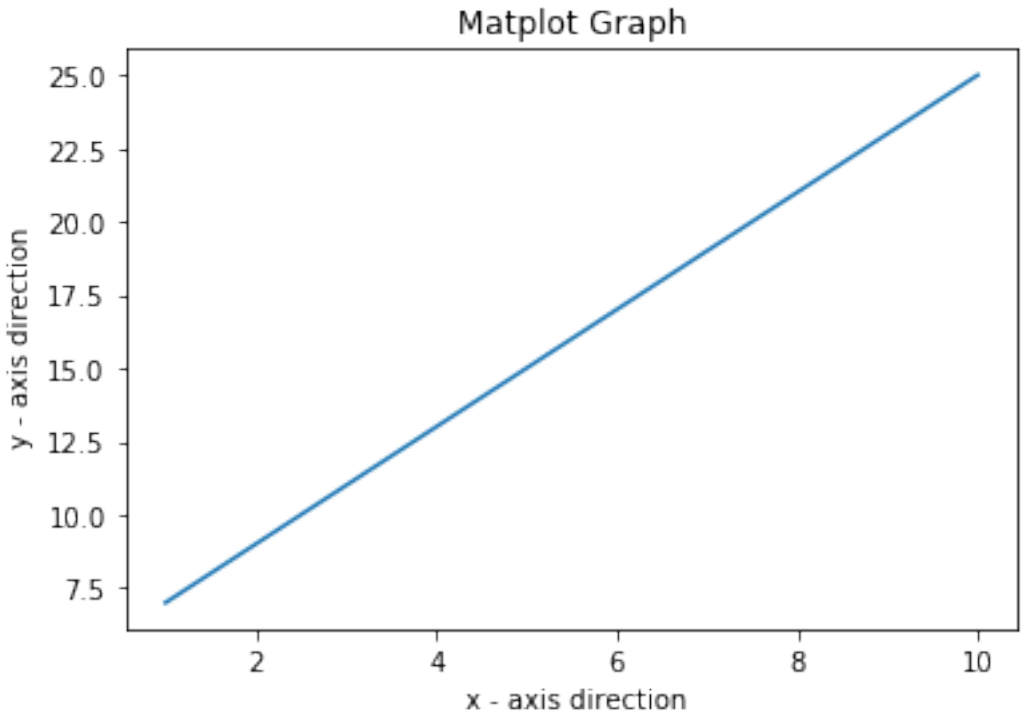
[121]:
```
7.211102550927978
```

[122]:
```
mat.sum(axis=0)
```

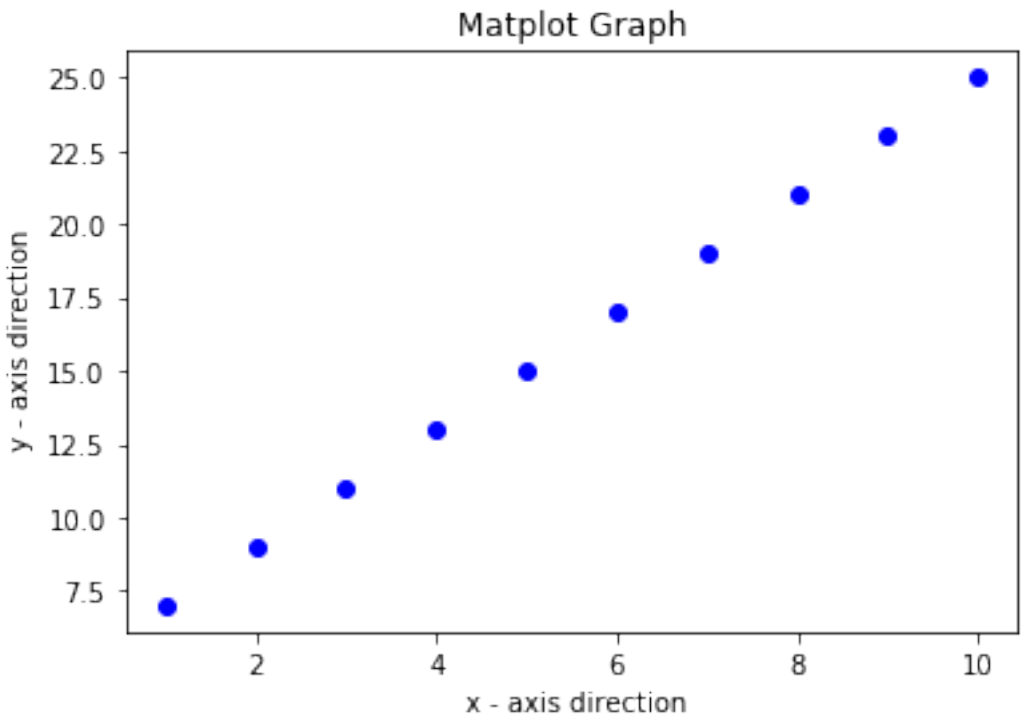
[122]:
```
array([55, 60, 65, 70, 75])
```

4 NumPy - Matplotlib

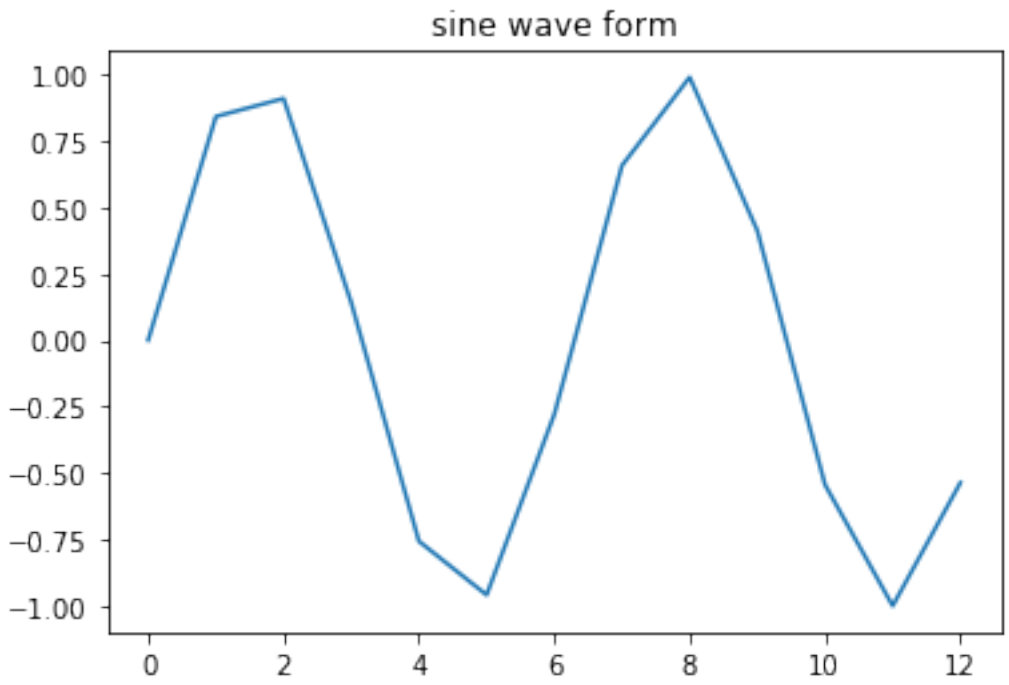
[128]:
```
import numpy as np
from matplotlib import pyplot as plt
x = np.arange(1,11)
y = 2 * x + 5
plt.title("Matplot Graph")
plt.xlabel(" x - axis direction")
plt.ylabel(" y - axis direction")
plt.plot(x,y)
plt.show()
```



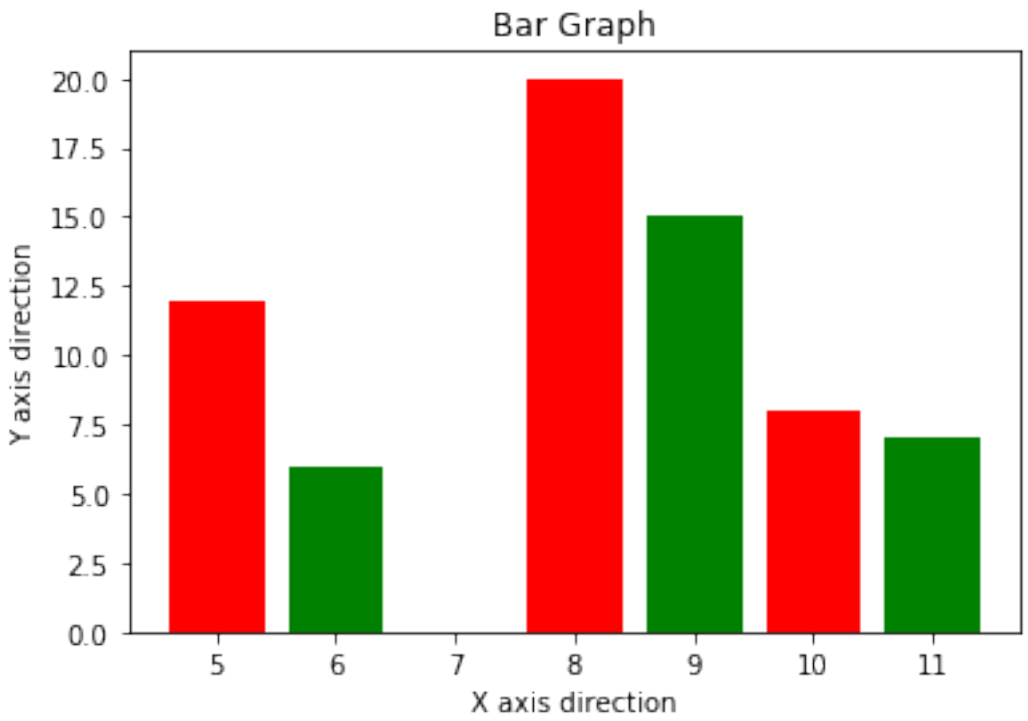
[129]:
```
import numpy as np
from matplotlib import pyplot as plt
x = np.arange(1,11)
y = 2 * x + 5
plt.title("Matplot Graph")
plt.xlabel(" x - axis direction")
plt.ylabel(" y - axis direction")
plt.plot(x,y,"ob")  # 'ob' --- format string in plot function
plt.show()
```



[132]:
```
import numpy as np
import matplotlib.pyplot as plt

# Compute the x and y coordinates for points on a sine curve
x = np.arange(0, 4 * np.pi, 1)
y = np.sin(x)
plt.title("sine wave form")

# Plot the points using matplotlib
plt.plot(x, y)
plt.show()
```



[135]:
```
from matplotlib import pyplot as plt
x = [5,8,10]
y = [12,20,8]

x2 = [6,9,11]
y2 = [6,15,7]
plt.bar(x, y, color = 'r', align = 'center')
plt.bar(x2, y2, color = 'g', align = 'center')
plt.title('Bar Graph')
plt.ylabel('Y axis direction')
plt.xlabel('X axis direction')

plt.show()
```


Bar Graph
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
Y axis direction
5
6
7
8
9
10
11
X axis direction

[ ]: