numpy arrays

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

In [9]:
```
np.zeros(10)
```

Out[9]:
```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

In [11]:
```
np.zeros(10,dtype=int)
```

Out[11]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

In [12]:
```
np.ones((3,4),dtype=float)   # 3x4 array of ones
```

Out[12]:
```
array([[1., 1., 1., 1.],
       [1., 1., 1., 1.],
       [1., 1., 1., 1.]])
```

In [13]:
```
np.full((3,4),2.1)   # 3x4 array filled with same number
```

Out[13]:
```
array([[2.1, 2.1, 2.1, 2.1],
       [2.1, 2.1, 2.1, 2.1],
       [2.1, 2.1, 2.1, 2.1]])
```

In [21]:
```
list(range(0,10))      # linear sequence in list
```

Out[21]:
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [22]:
```
np.arange(0,10)      # linear sequence in array
```

Out[22]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [23]:
```
np.arange(0,10,2)      # stepping by 2
```

Out[23]:
```
array([0, 2, 4, 6, 8])
```

In [32]:
```
np.linspace(0,10,5)      # five values evenly spaced
```

Out[32]:
```
array([ 0. ,  2.5,  5. ,  7.5, 10. ])
```

In [42]:
```
np.eye(3)               # 3x3 identity matrix
```

Out[42]:
```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

In [5]:
```
np.empty((3,3))          # empty array (values are meaningless)
```

Out[5]:
```
array([[0.00000000e+000, 0.00000000e+000, 0.00000000e+000],
       [0.00000000e+000, 0.00000000e+000, 3.26083326e-321],
       [1.11261570e-306, 8.90060779e-307, 3.49699467e-317]])
```

In [ ]:
```
## Random numbers
```

In [8]:
```
np.random.random(5)     # five obs Uniform (0,1)
```

In [9]:
```
np.random.rand(5)       # five obs standard normal
```

In [ ]:
```
np.random.normal(1,2,5)   # five obs normal(mean=1,s.deviation = 2)
```

In [ ]:
```
np.random.normal(5)       # one obs normal(mean=5)
```

In [ ]:
```
np.random.randint(0,10,size=5)     # five obs discrete uniform (0,10)
```

array attributes

In [48]:
```
np.random.seed(1)
x = np.random.randint(10,size=(3,4,5))     # 3D-integer array
x
```

Out[48]:
```
array([[[5, 8, 9, 5, 0],
        [0, 1, 7, 6, 9],
        [2, 4, 5, 2, 4],
        [2, 4, 7, 7, 9]],

       [[1, 7, 0, 6, 9],
        [9, 7, 6, 9, 1],
        [0, 1, 8, 8, 3],
        [9, 8, 7, 3, 6]],

       [[5, 1, 9, 3, 4],
        [8, 1, 4, 0, 3],
        [9, 2, 0, 4, 9],
        [2, 7, 7, 9, 8]]])
```

In [38]:
```
x.ndim
```

Out[38]:
```
3
```

In [39]:
```
x.shape
```

Out[39]:
```
(3, 4, 5)
```

In [40]:
```
x.size          # 3x4x5
```

Out[40]: 60

In [49]:
```
x[0,0,2]
```

Out[49]: 9

In [ ]:
```
# integer array does not accept other data type
```

In [50]:
```
x[0,0,2] = 1.5   # floating-point value to an integer array
```

In [51]:
```
x[0,0,2]        # truncated
```

Out[51]: 1

1D array indexing

In [15]:
```
x = np.array([2,3,6,1,7,9,8,3,1])
x
```

Out[15]: array([2, 3, 6, 1, 7, 9, 8, 3, 1])

In [18]:
```
x.ndim
```

Out[18]: 1

In [19]:
```
x.shape
```

Out[19]: (9,)

In [61]:
```
x[7]    # 8th value
```

Out[61]: 3

In [62]:
```
x[:3]    # first three values
```

Out[62]: array([2, 3, 6])

In [63]:
```
x[-1]    # last value
```

Out[63]: 1

In [64]:
```
x[-2:]   # last two values
```

Out[64]: array([3, 1])

In [65]:
```
x[2:4]    # indices 2 to 3
```

Out[65]:
```
array([6, 1])
```

In [66]:
```
x[::2]    # every other element
```

Out[66]:
```
array([2, 6, 7, 8, 1])
```

In [68]:
```
x[5::2]    # every other element, starting at index 5
```

Out[68]:
```
array([9, 3])
```

nD arrays

In [50]:
```
np.random.seed(1)
x2 = np.random.randint(10,size=(3,4))
x2
```

Out[50]:
```
array([[5, 8, 9, 5],
       [0, 0, 1, 7],
       [6, 9, 2, 4]])
```

In [51]:
```
x2[0,:]      # first row
```

Out[51]:
```
array([5, 8, 9, 5])
```

In [21]:
```
x2[:,1]            # 2nd col as 1D array
```

Out[21]:
```
array([8, 0, 9])
```

In [52]:
```
x2[:,1].shape
```

Out[52]:
```
(3,)
```

In [53]:
```
x2[:,1].ndim
```

Out[53]:
```
1
```

In [ ]:
```
# convert 1D to 2D
```

In [58]:
```
x22 = x2[:,1]
```

In [59]:
```
x22.reshape(-1,1)
```

Out[59]:
```
array([[8],
       [0],
       [9]])
```

In [61]:
```
x22.reshape(-1,1).shape
```

Out[61]:
```
(3, 1)
```

In [62]:
```
x22.reshape(-1,1).ndim
```

Out[62]:
```
2
```

In [ ]:
```

```

In [71]:
```
x2[:2,:3]     # up to row 2, up to col 3
```

Out[71]:
```
array([[5, 8, 9],
       [0, 0, 1]])
```

In [73]:
```
x2[::2]       # every other row
```

Out[73]:
```
array([[5, 8, 9, 5],
       [6, 9, 2, 4]])
```

In [74]:
```
x2[:,::2]       # every other col
```

Out[74]:
```
array([[5, 9],
       [0, 1],
       [6, 2]])
```

slices or views

In [ ]:
```
# subarrays are views
```

In [79]:
```
x3 = x2[:2,:2]
x3
```

Out[79]:
```
array([[5, 8],
       [0, 0]])
```

In [82]:
```
x3[0,1]
```

Out[82]:
```
8
```

In [83]:
```
# change subarray value
x3[0,1] = 4
```

In [84]:
```
x2
```

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

In [ ]:
```
# original array is changed
```

array copies

In [ ]:
```
x3 = x2[:2,:2].copy()
```

In [ ]:
```
# if x3 is changed, x2 will not
```

1D array

In [ ]:
```
# list is not an array
```

In [25]:
```
list1 = list(range(1,10))
list1
```

Out[25]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [ ]:
```
# 1D array
```

In [32]:
```
array1 = np.arange(1,10)
array1
```

Out[32]:
```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [33]:
```
array1.shape
```

Out[33]:
```
(9,)
```

In [34]:
```
array1.ndim
```

Out[34]:
```
1
```

In [ ]:
```
# 2D array
```

In [41]:
```
array1 = np.array([(1,7,3,4)])
array1
```

Out[41]:
```
array([[1, 7, 3, 4]])
```

In [43]:
```
array1.shape
```

Out[43]: (1, 4)

In [44]:
```
array1.ndim
```

Out[44]: 2

In [ ]:
```
# another 2D array
```

In [37]:
```
array2 = np.array([(1,7,3,4),(3,2,1,4)])
array2
```

Out[37]:
```
array([[1, 7, 3, 4],
       [3, 2, 1, 4]])
```

In [38]:
```
array2.shape
```

Out[38]: (2, 4)

In [45]:
```
array2.ndim
```

Out[45]: 2

In [ ]:
```
# convert 2D array to 1D array
```

In [47]:
```
array2.ravel()
```

Out[47]: array([1, 7, 3, 4, 3, 2, 1, 4])

In [48]:
```
array2.ravel().ndim
```

Out[48]: 1

In [49]:
```
array2.ravel().shape
```

Out[49]: (8,)

In [57]:
```
# ravel is a slice/view
```

reshape

In [57]:
```
# convert to a 3-by-3 array
```

In [58]:
```
array2 = list2.reshape((3,3))
array2
```

Out[58]:
```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

In [96]:
```
array3 = list2.reshape((1,-1))         # row vector
array3
```

Out[96]:
```
array([[1, 2, 3, 4, 5, 6, 7, 8, 9]])
```

In [97]:
```
array3.ndim
```

Out[97]:
```
2
```

In [ ]:
```
# row vector is 2D array
```

In [102]:
```
x = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9])
x
```

Out[102]:
```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [103]:
```
x.ndim
```

Out[103]:
```
1
```

In [104]:
```
# x is 1D array
```

In [105]:
```
type(x)
```

Out[105]:
```
numpy.ndarray
```

concatenate arrays

In [108]:
```
array2
```

Out[108]:
```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

In [110]:
```
np.concatenate([array2,array2],axis=0)
```

Out[110]:
```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9],
       [1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

In [111]:
```
np.concatenate([array2,array2],axis=1)
```

Out[111]:
```
array([[1, 2, 3, 1, 2, 3],
       [4, 5, 6, 4, 5, 6],
       [7, 8, 9, 7, 8, 9]])
```

In [112]:
```
array2.sum(axis=0)    # sum colapsing rows
```

Out[112]:
```
array([12, 15, 18])
```

In [113]:
```
array2.sum(axis=1)    # sum colapsing cols
```

Out[113]:
```
array([ 6, 15, 24])
```

In [ ]:
```

```

matplotlib

In [66]:
```
import matplotlib.pyplot as plt
```

In [11]:
```
x = np.linspace(-1,3.5)
```

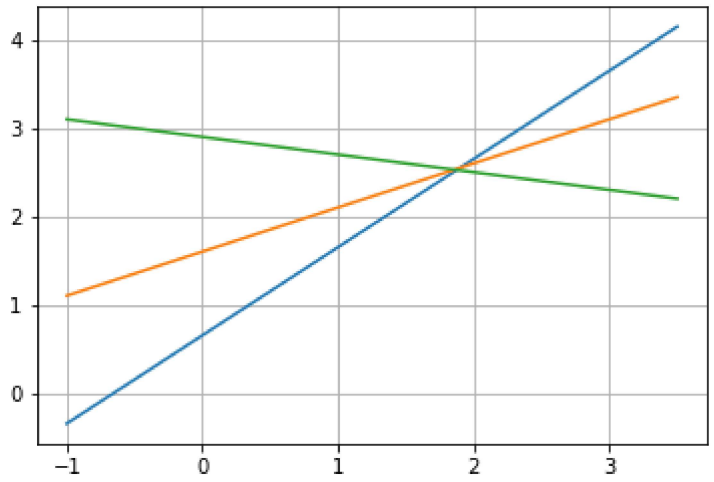
In [12]:
```
# first five
x[:5]
```

Out[12]:
```
array([-1.        , -0.90816327, -0.81632653, -0.7244898 , -0.63265306])
```

In [13]:
```
# last five
x[-5:]
```

Out[13]:
```
array([3.13265306, 3.2244898 , 3.31632653, 3.40816327, 3.5       ])
```

In [18]:
```
for m,b in [(1,0.65),(0.5,1.6),(-0.2,2.9)]:
    plt.plot(x,m*x+b)
    plt.grid()
```



In [ ]:
```

```

In [84]:
```
# Scatterplot
```

In [64]:
```
mean = [0, 0]
cov = [[1, 2],
       [2, 5]]
X = np.random.multivariate_normal(mean, cov, 100)
X.shape
```

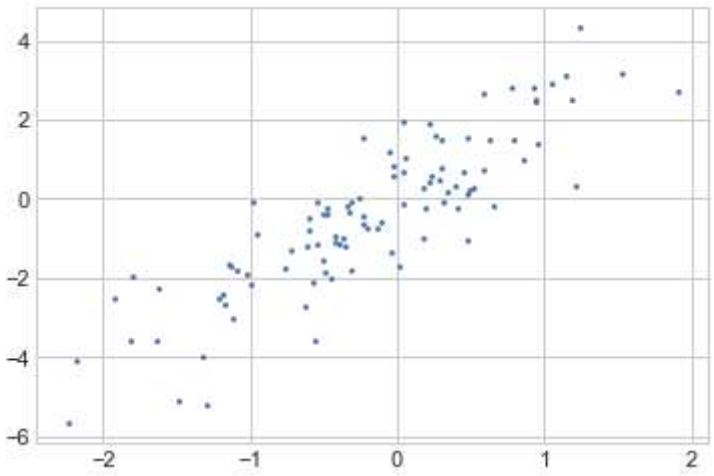
Out[64]: (100, 2)

In [69]:
```
%matplotlib inline
# import seaborn; seaborn.set()      # needed?

plt.style.use('seaborn-whitegrid')
```

In [70]:
```
plt.scatter(X[:, 0], X[:, 1],s=3)
```

Out[70]:
```
<matplotlib.collections.PathCollection at 0x2cd18a9a978>
```



In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```
# select 20 rows from X randomly
```

In [72]:
```
# seed
np.random.seed(1)
```

In [73]:
```
indices = np.random.choice(X.shape[0], 20, replace=False)
indices
```

Out[73]:
```
array([80, 84, 33, 81, 93, 17, 36, 82, 69, 65, 92, 39, 56, 52, 51, 32, 31,
       44, 78, 10])
```

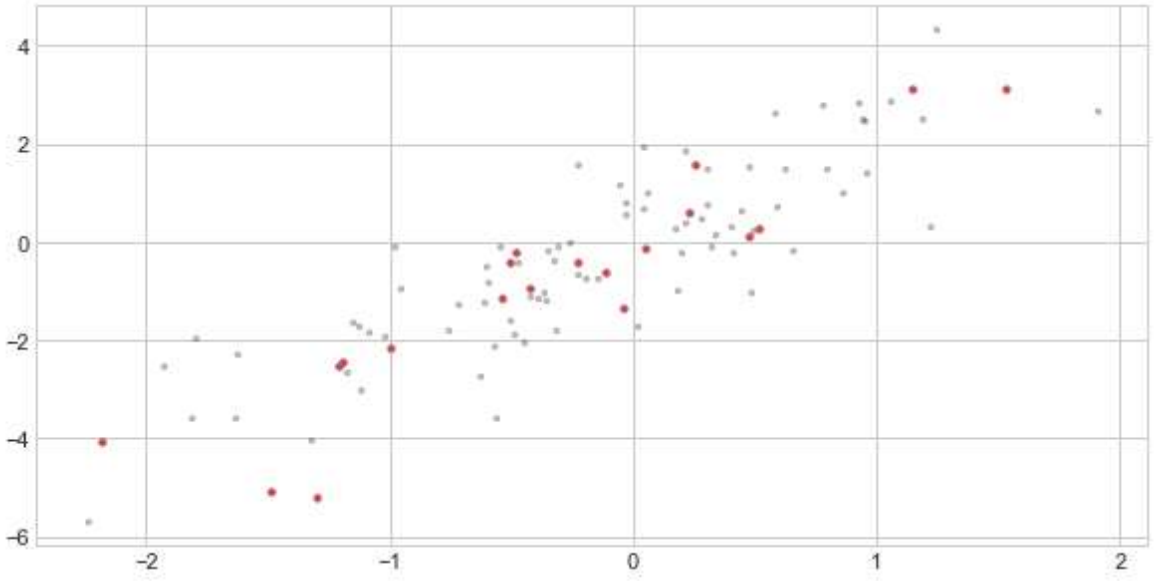
In [74]:
```
selection = X[indices]
selection.shape
```

Out[74]:
```
(20, 2)
```

In [77]:
```
from matplotlib.pyplot import figure
```

In [82]:
```
figure(figsize=(10,5))
plt.scatter(X[:, 0], X[:, 1], alpha=0.25,s=6,color = 'k')
plt.scatter(selection[:, 0], selection[:, 1],color = 'r',s=10)
```

Out[82]: <matplotlib.collections.PathCollection at 0x2cd189f6be0>



In [ ]: