```
Average of x:  0.6
x_tilde:  [ 0.4 -2.8  2.4]
Average of x_tilde:  -1.4802973661668753e-16
```

(The mean of $\tilde{x}$ is very very close to zero.)

3.3. Standard deviation

Standard deviation. We can define a function that corresponding to the VMLS definition of the standard deviation of a vector, $\mathbf{std}(x) = \|x - \mathbf{avg}(x)\mathbf{1}\|/\sqrt{n}$, where n is the length of the vector.

```
In [ ]: x = np.random.random(100)
        stdev = lambda x: np.linalg.norm(x - sum(x)/len(x))/(len(x)**0.5)
        stdev(x)

Out[ ]: 0.30440692170248823
```

You can also use the `numpy` function `np.std(x)` to obtain the standard deviation of a vector.

Return and risk. We evaluate the mean return and risk (measured by standard deviation) of the four time series Figure 3.4 of VMLS.

```
In [ ]: a = np.ones(10)
        np.mean(a), np.std(a)

Out[ ]: (1.0, 0.0)

In [ ]: b = [ 5, 1, -2, 3, 6, 3, -1, 3, 4, 1 ]
        np.mean(b), np.std(b)

Out[ ]: (2.3, 2.4103941586387903)

In [ ]: c = [ 5, 7, -2, 2, -3, 1, -1, 2, 7, 8 ]
        np.mean(c), np.std(c)

Out[ ]: (2.6, 3.7735924528226414)

In [ ]: d = [ -1, -3, -4, -3, 7, -1, 0, 3, 9, 5 ]
        np.mean(d), np.std(d)

Out[ ]: (1.2, 4.308131845707604)
```

Standardizing a vector. If a vector x isn't constant (*i.e.*, at least two of its entries are different), we can standardize it, by subtracting its mean and dividing by its standard deviation. The resulting standardized vector has mean value zero and RMS value one. Its entries are called z-scores. We'll define a `standardize` function, and then check it with a random vector.

In [ ]:
```
def standardize(x):
    x_tilde = x - np.mean(x)
    return x_tilde/np.std(x_tilde)
x = np.random.random(100)
np.mean(x), np.std(x)
```

Out[ ]:
```
(0.568533078290501, 0.282467953801772)
```

In [ ]:
```
z = standardize(x)
np.mean(z), np.std(z)
```

Out[ ]:
```
(1.3322676295501878e-17, 1.0)
```

The mean or average value of the standardized vector `z` is very nearly zero.

3.4. Angle

Angle. Let's define a function that computes the angle between two vectors. We will call it `ang`.

In [ ]:
```
#Define angle function, which returns radians
ang = lambda x,y : np.arccos(x @ y /
↪  (np.linalg.norm(x)*np.linalg.norm(y)))
a = np.array([1,2,-1])
b = np.array([2,0,-3])
ang(a,b)
```

Out[ ]:
```
0.9689825515916383
```

In [ ]:
```
#Get angle in degrees
ang(a,b) * (360/(2*np.pi))
```

Out[ ]:
```
55.51861062801842
```