```
beta = -3.7
LHS = np.inner(alpha*x + beta*y, a)
RHS = alpha*np.inner(x,a) + beta*np.inner(y,a)
print('LHS:', LHS)
print('RHS:', RHS)
```

```
LHS: -8.3
RHS: -8.3
```

For the function $f(x) = a^T x$, we have $f(e_3) = a_3$. Let's check that this holds in our example.

```
In [ ]: a = np.array([-2,0,1,-3])
        e3 = np.array([0,0,1,0])
        print(e3 @ a)
```

```
1
```

Examples. Let's define the average function in Python and check its value of a specific vector. (Numpy also contains an average function, which can be called with `np.mean`.

```
In [ ]: avg = lambda x: sum(x)/len(x)
        x = [1,-3,2,-1]
        avg(x)
```

```
Out[ ]: -0.25
```

2.2. Taylor approximation

Taylor approximation. The (first-order) Taylor approximation of function $f : \mathbf{R}^n \to \mathbf{R}$, at the point z, is the affine function $\hat{f}(x)$ given by

$$\hat{f}(x) = f(z) + \nabla f(z)^T (x - z)$$

For x near z, $\hat{f}(x)$ is very close to $f(x)$. Let's try a numerical example (see page 36 of textbook) using Python.

```
In [ ]: f = lambda x: x[0] + np.exp(x[1] - x[0])
        grad_f = lambda z: np.array([1 - np.exp(z[1] - z[0]), np.exp(z[1]
        ↪  - z[0])])
        z = np.array([1,2])
```

```
#Taylor approximation
f_hat = lambda x: f(z) + grad_f(z) @ (x - z)
f([1,2]), f_hat([1,2])
```

Out[ ]:
```
(3.718281828459045, 3.718281828459045)
```

In [ ]:
```
f([0.96, 1.98]), f_hat([0.96,1.98])
```

Out[ ]:
```
(3.7331947639642977, 3.732647465028226)
```

In [ ]:
```
f([1.10, 2.11]), f_hat([1.10, 2.11])
```

Out[ ]:
```
(3.845601015016916, 3.845464646743635)
```

2.3. Regression model

Regression model. The regression model is the affine function of x given by $f(x) = x^T\beta + \nu$, where the n-vector β and the scalar ν are the parameters in the model. The regression model is used to guess or approximate a real or observed value of the number y that is associated with x (We'll see later how to find the parameters in a regression model using data).

Let's define the regression model for house sale price described on page 39 of VMLS, and compare its prediction to the true house sale price y for a few values of x.

In [ ]:
```
# parameters in regression model
beta = np.array([148.73, -18.85])
v = 54.40
y_hat = lambda x: x @ beta + v
#Evaluate regression model prediction
x = np.array([0.846, 1])
y = 115
y_hat(x), y
```

Out[ ]:
```
(161.37557999999999, 115)
```

In [ ]:
```
x = np.array([1.324, 2])
y = 234.50
y_hat(x), y
```