```
#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 $\beta$ and the scalar $\nu$ 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
```

```
Out[ ]: (213.61852000000002, 234.5)
```

Our first prediction is pretty bad; our second one is better. A scatter plot of predicted and actual house prices (Figure 2.4 of VMLS) can be generated as follows. We use the `house_sales_data` data set to obtain the vector `price, areas, beds` (see Appendix B). The data sets we used in this Python language companion can be found on: `https://github.com/jessica-wyleung/VMLS-py`. You can download the jupyter notebook from the repository and work on it directly or you can copy and paste the data set onto your own jupyter notebook.

```
In [ ]: import matplotlib.pyplot as plt
        plt.ion()
        D = house_sales_data()
        price = D['price']
        area = D['area']
        beds = D['beds']
        v = 54.4017
        beta = np.array([147.7251, -18.8534])
        predicted = v + beta[0]*area + beta[1]*beds
        plt.scatter(price, predicted)
        plt.plot((0,800),(0,800) ,ls='--', c = 'r')
        plt.ylim(0,800)
        plt.xlim(0,800)
        plt.xlabel('Actual Price')
        plt.ylabel('Predicted Price')
        plt.show()
```