HW1

Authors:

- Luis Gascon
- Ethan Webb

Importing Dependencies

[21]:

```
import numpy as np
import os
from numpy.linalg import norm

from matplotlib import pyplot
from mpl_toolkits.mplot3d import Axes3D
#matplotlib inline
```

1. Simple Numpy Function

This function simply returns a 5x5 identity matrix

[78]:

```
def warmUpExercise() -> np.ndarray:
    return np.identity(5)

print(warmUpExercise())
```

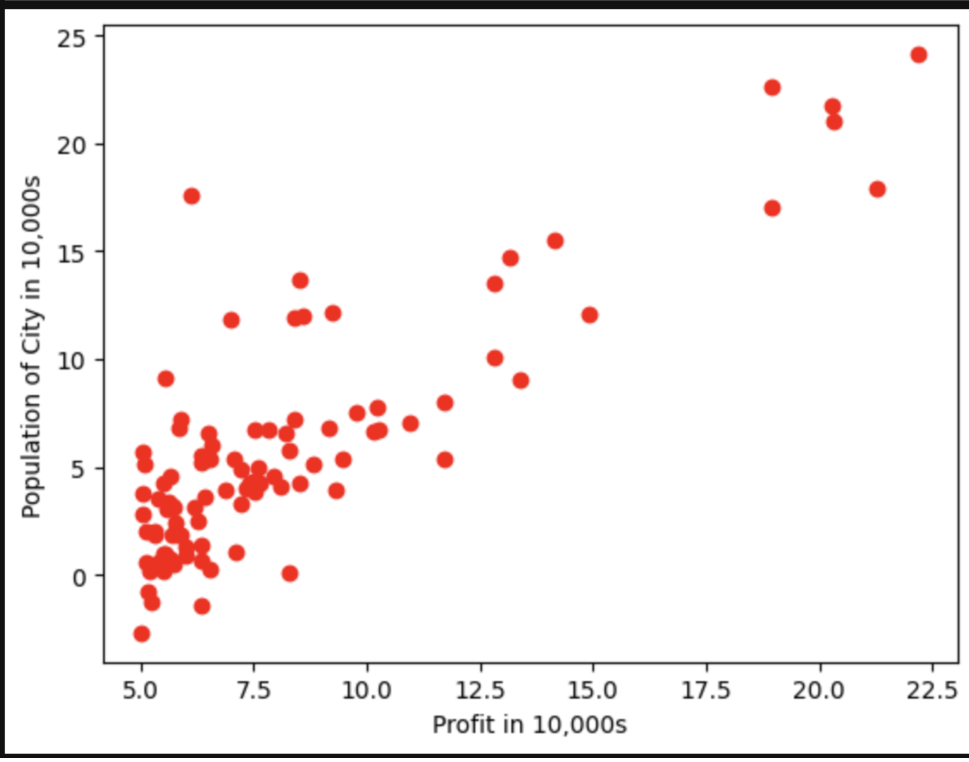
```
[[1. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 0. 1. 0. 0.]
 [0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 1.]]
```

2.1 Plotting the Data

Plot the data so that y-axis display the “profit in Profit in 10,000” and x-axis displays the “Population of City in 10,000s”

[65]:
```python
def plotData(x, y) -> None:
    fig = pyplot.figure()
    ax = fig.add_subplot(1, 1, 1)
    ax.plot(X, y, 'ro')
    ax.set_xlabel("Profit in 10,000s")
    ax.set_ylabel("Population of City in 10,000s")

plotData(X, y)
```



2.2 Computing the optimal weights for the linear regression model

We implement the following mathematical function as a Python function:

$$
\begin{aligned}
E(h) &= \frac{1}{N}\sum_{n=1}^{N}(h(x_n) - y_n)^2 \\
&= \frac{1}{N}\|Xw - y\|^2
\end{aligned}
$$

$E(w)$ computes the errors for our linear function

```python
def computeError(X, y, w) -> float:
    N:int = y.size
    E:float = 0.0

    # We sum the errors for each weights
    for weight in w:
        E += (norm(X.dot(weight) - y) ** 2) / N

    return E

print(f"With w = [0, 0] \nError computed = {computeError(X, y, w=np.array([0.0, 0.0]))}")
print("Expected error value (approximately) 32.07\n")

print(f"With w = [-1,2] \nError computed = {computeError(X, y, w=np.array([-1, 2]))}")
print("Expected error value (approximately) 54.24")
```

```
With w = [0, 0]
Error computed = 128.2909355098227
Expected error value (approximately) 32.07

With w = [-1,2]
Error computed = 404.6529501290185
Expected error value (approximately) 54.24
```