# 1. Simple Linear regression

Step 1: Import packages and classes

```
In [1]: import numpy as np
        from sklearn.linear_model import LinearRegression
```

Step 2: Provide data

```
In [2]: x = np.array([5, 15, 25, 35, 45, 55]).reshape((-1, 1))
        y = np.array([5, 20, 14, 32, 22, 38])
        x, y
```

```
Out[2]: (array([[ 5],
                [15],
                [25],
                [35],
                [45],
                [55]]),
         array([ 5, 20, 14, 32, 22, 38]))
```

Step 3: Create a model and fit it

```
In [3]: model = LinearRegression()
        model.fit(x, y)
        model = LinearRegression().fit(x, y)
```

Step 4: Get results

```
In [4]: # obtain the coefficient of determination, R², with .score() of model:
        r_sq = model.score(x, y)
        print(f"coefficient of determination: {r_sq}")
```

```
coefficient of determination: 0.715875613747954
```

```
In [5]: # The attributes of model are .intercept_, which represents the
        # coefficient b₀, and .coef_, which represents b₁:
        print(f"intercept: {model.intercept_}")
        print(f"slope: {model.coef_}")
```

```
intercept: 5.633333333333329
slope: [0.54]
```

Step 5: Predict response

```
In [6]: y_pred = model.predict(x)
        print(f"predicted response:\n{y_pred}")
```

```
predicted response:
[ 8.33333333 13.73333333 19.13333333 24.53333333 29.93333333 35.33333333]
```

In [7]:

```
# also you get the predicted response
y_pred = model.intercept_ + model.coef_ * x
print(f"predicted response:\n{y_pred}")
```

```
predicted response:
[[ 8.33333333]
 [13.73333333]
 [19.13333333]
 [24.53333333]
 [29.93333333]
 [35.33333333]]
```

In [8]:

```
# predictions on inputs
x_new = np.arange(5).reshape((-1, 1))
x_new
```

Out[8]:

```
array([[0],
       [1],
       [2],
       [3],
       [4]])
```

In [9]:

```
y_new = model.predict(x_new)
y_new
```

Out[9]:

```
array([5.63333333, 6.17333333, 6.71333333, 7.25333333, 7.79333333])
```

## 2. Multiple Linear Regression

Steps 1 and 2: Import packages and classes, and provide data

In [10]:

```
import numpy as np
from sklearn.linear_model import LinearRegression
x = [
 [0, 1], [5, 1], [15, 2], [25, 5], [35, 11], [45, 15], [55]
 ]
y = [4, 5, 20, 14, 32, 22, 38, 43]
x, y = np.array(x), np.array(y)
x,y
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_7648\3724308309.py:7: VisibleDepre
cationWarning: Creating an ndarray from ragged nested sequences (which is a li
st-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) i
s deprecated. If you meant to do this, you must specify 'dtype=object' when cr
eating the ndarray.
  x, y = np.array(x), np.array(y)
```

Out[10]:

```
(array([list([0, 1]), list([5, 1]), list([15, 2]), list([25, 5]),
        list([35, 11]), list([45, 15]), list([55])], dtype=object),
 array([ 4,  5, 20, 14, 32, 22, 38, 43]))
```

Step 3: Create a model and fit it

In [ ]:

```
model = LinearRegression().fit(x, y)
```

Step 4: Get results

In [ ]:

```
r_sq = model.score(x, y)
print(f"coefficient of determination: {r_sq}")
#     output: coefficient of determination: 0.8615939258756776

print(f"intercept: {model.intercept_}")
# output:intercept: 5.52257927519819

print(f"coefficients: {model.coef_}")
# output: coefficients: [0.44706965 0.25502548]
```

Step 5: Predict response

In [ ]:

```
y_pred = model.predict(x)
print(f"predicted response:\n{y_pred}")
# or
y_pred = model.intercept_ + np.sum(model.coef_ * x, axis=1)
print(f"predicted response:\n{y_pred}")

# output:
# predicted response:
# [ 5.77760476 8.012953 12.73867497 17.9744479 23.97529728 29.4660957
# 38.78227633 41.27265006]
```

In [ ]:

```
# Predictions
x_new = np.arange(10).reshape((-1, 2))

y_new = model.predict(x_new)
x_new, y_new

# Output
# array([[0, 1],
# [2, 3],
# [4, 5],
# [6, 7],
# [8, 9]])

# array([ 5.77760476, 7.18179502, 8.58598528, 9.99017554, 11.3943658 ])
```

# 3. Polynomial Regression With scikitlearn

Step 1: Import packages and classes

In [15]:

```
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
```

Step 2a: Provide data

In [17]:

```
x = np.array([5, 15, 25, 35, 45, 55]).reshape((-1, 1))
y = np.array([15, 11, 2, 8, 25, 32])
```

Step 2b: Transform input data

In [23]:

```
transformer = PolynomialFeatures(degree=2, include_bias=False)
transformer.fit(x)
```

Out[23]:

```
PolynomialFeatures(include_bias=False)
```

In [24]:

```
x_ = transformer.transform(x)
```

In [32]:

```
x_ = PolynomialFeatures(degree=2, include_bias=False).fit_transform
x_
```

Out[32]:

```
<bound method TransformerMixin.fit_transform of PolynomialFeatures(include_bia
s=False)>
```

Step 3: Create a model and fit it

In [ ]:

```
model = LinearRegression().fit(x_, y)
```

In [ ]: