```
import numpy as np
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
x = np.array([5, 15, 25, 35, 45, 55])
y = np.array([5, 20, 14, 32, 22, 38])
print(x)
print(y)
print(x.shape)
print(y.shape)
```

```
[ 5 15 25 35 45 55]
[ 5 20 14 32 22 38]
(6,)
(6,)
```

```
model = LinearRegression()
```

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

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-31-d3dc977168f5> in <cell line: 1>()
----> 1 model.fit(x, y)

                                 3 frames
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py in check_array(array, accept_sparse, accept_large_sparse,
dtype, order, copy, force_all_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_min_features, estimator, input_name)
    900             # If input is 1D raise error
    901             if array.ndim == 1:
--> 902                 raise ValueError(
    903                     "Expected 2D array, got 1D array instead:\narray={}.\n"
    904                     "Reshape your data either using array.reshape(-1, 1) if "

ValueError: Expected 2D array, got 1D array instead:
array=[ 5 15 25 35 45 55].
Reshape your data either using array.reshape(-1, 1) if your data has a single feature or array.reshape(1, -1) if it contains a
single sample.
```

```
# x must have one column and x can have many rows. for this we use
# reshape(-1,1).
x = np.array([5, 15, 25, 35, 45, 55]).reshape(-1, 1)
y = np.array([5, 20, 14, 32, 22, 38])
print(x)
print(y)
model = LinearRegression()
print(x.shape)
print(y.shape)
```

```
[[ 5]
 [15]
 [25]
 [35]
 [45]
 [55]]
[ 5 20 14 32 22 38]
(6, 1)
(6,)
```

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

```
▾ LinearRegression
LinearRegression()
```

```
# to check whether the model works satisfactorily,
# obtain the coefficient of determination, R², with .score()
r_sq = model.score(x, y)
print(f"coefficient of determination: {r_sq}")
```

```
coefficient of determination: 0.7158756137479542
```

```
# The attributes of model are .intercept_ (b₀), and .coef_ (b₁)
print(f"intercept: {model.intercept_}")
print(f"slope: {model.coef_}")
```

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

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

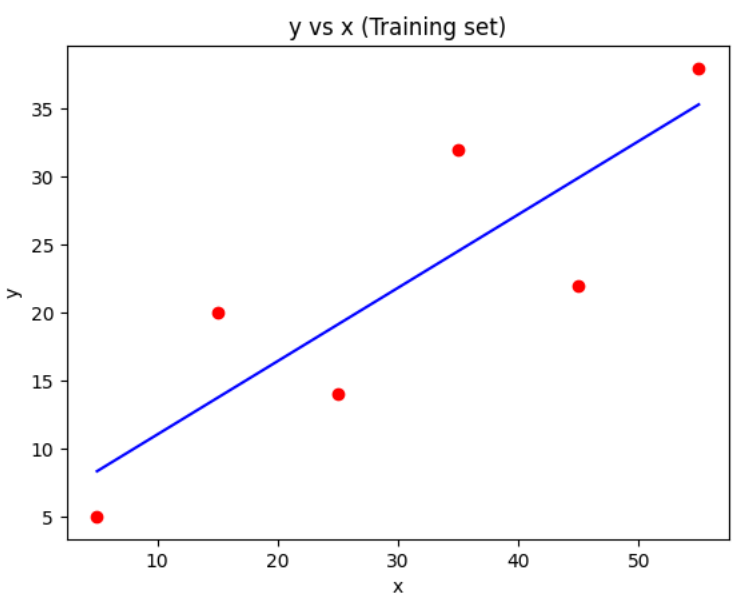
```
predicted response:
[ 8.33333333 13.73333333 19.13333333 24.53333333 29.93333333 35.33333333]
```

```
# Instead of predict(), You can also use below equation for test reponses.
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]]
```

Visualising results

```
plt.scatter(x,y, color = 'red')
plt.plot(x, y_pred, color = 'blue')
plt.title('y vs x (Training set)')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```



Start coding or generate with AI.