```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/linear-regression/LinearX.npy
/kaggle/input/linear-regression/LinearY.npy
```

+ Code + Markdown

[78]:
```
import matplotlib.pyplot
```

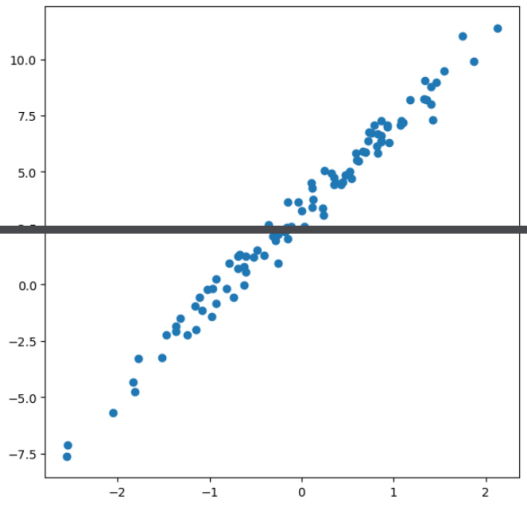
Storing data in variables

[79]:
```
X=np.load('/kaggle/input/linear-regression/LinearX.npy')
Y=np.load('/kaggle/input/linear-regression/LinearY.npy')
```

[80]:
```
X=(X-X.mean())/X.std()
```

Viewing the data in a plot

```
plt.subplots(figsize=(7,7))
plt.scatter(X,Y)
plt.show()
```



Gradient Descent Algorithm

[82]:
```
def y_hat(theta,x):
    return theta[0]+theta[1]*x

def error(x,y,theta):
    Y=y_hat(theta,x)
    m=x.shape[0]
    e=np.sum((Y-y)**2)/m
    return e

def slope(x,y,theta):
    grad=np.zeros((2,))
    Y=y_hat(theta,x)
    m=x.shape[0]

    grad[0]=2*np.sum(Y-y)/m
    grad[1]=2*np.sum((Y-y)*x)/m
    return grad
```

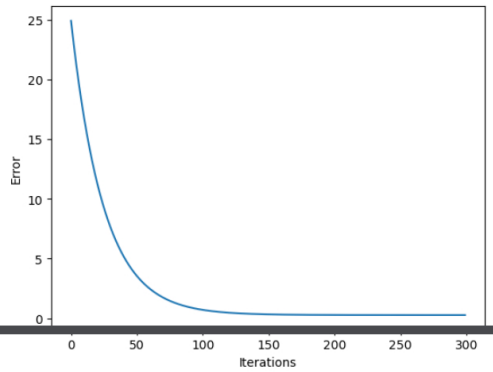
```
def desc(x,y,lr=0.01):
    theta=np.zeros((2,))
    errorList=[]
    for i in range(300):
        dy_dx =slope(x,y,theta)
        theta = theta - lr*dy_dx
        err = error(x,y,theta)
        errorList.append(err)

    return theta, errorList
```

[83]:
```
theta,errorlist=desc(X,Y)
```

Plot for error

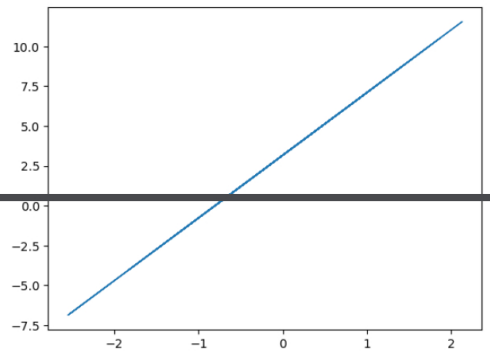
[84]:
```
plt.plot(errorlist)
plt.xlabel("Iterations")
plt.ylabel("Error")
plt.show()
```



[ ]:
```
## we see error reducing to zero
```

Line formed after linear regression

```
Y_=y_hat(theta,X)
plt.plot(X,Y_,linewidth=1)
plt.show()
```



closeness with data scatterplot

[87]:
```
plt.subplots(figsize=(7,7))
plt.scatter(X,Y)
Y_=y_hat(theta,X)
plt.plot(X,Y_,linewidth=1)
plt.show()
```