In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
from sklearn.datasets import make_regression
```

In [3]:

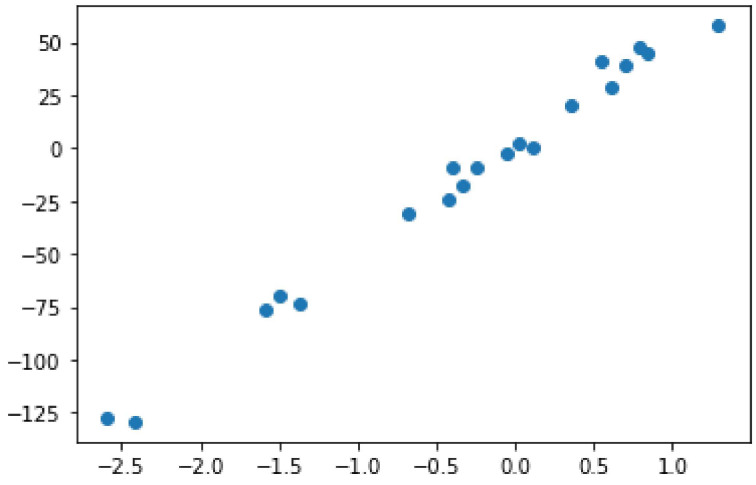
```
x, y = make_regression(n_samples = 20, n_features = 1, noise = 6)
```

In [4]:

```
plt.scatter(x,y)
```

Out[4]:

```
<matplotlib.collections.PathCollection at 0x2e33388d550>
```



In [5]:

```
from sklearn.linear_model import LinearRegression
```

In [6]:

```
lr = LinearRegression()
```

In [7]:

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

Out[7]:

```
LinearRegression()
```

In [8]:

```
m = lr.intercept_
m
```

Out[8]:

```
1.8565038513504728
```

In [9]:

```
b = lr.coef_
b
```

Out[9]:

```
array([50.97757244])
```

In [10]:

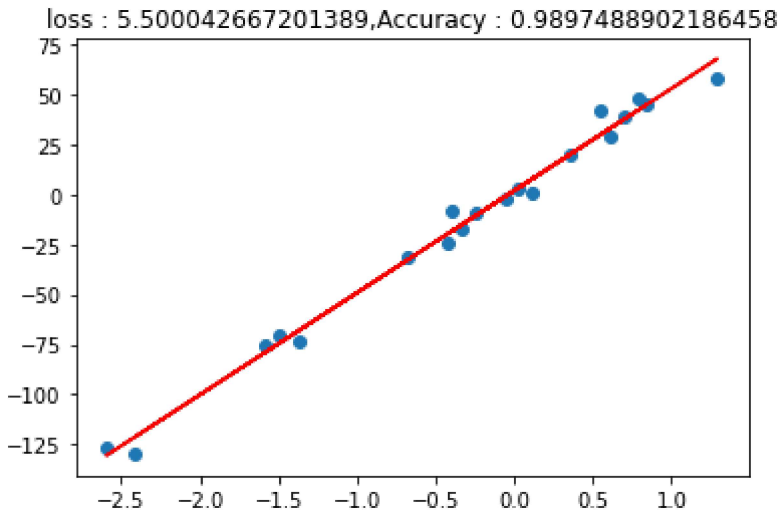
```
from sklearn.metrics import mean_squared_error, r2_score
```

In [11]:

```
plt.plot(x, lr.predict(x),'r-')
plt.scatter(x,y)
plt.title(f'loss : {np.sqrt(mean_squared_error(y, lr.predict(x)))},Accuracy : {r2_sc
```

Out[11]:

```
Text(0.5, 1.0, 'loss : 5.500042667201389,Accuracy : 0.9897488902186458')
```



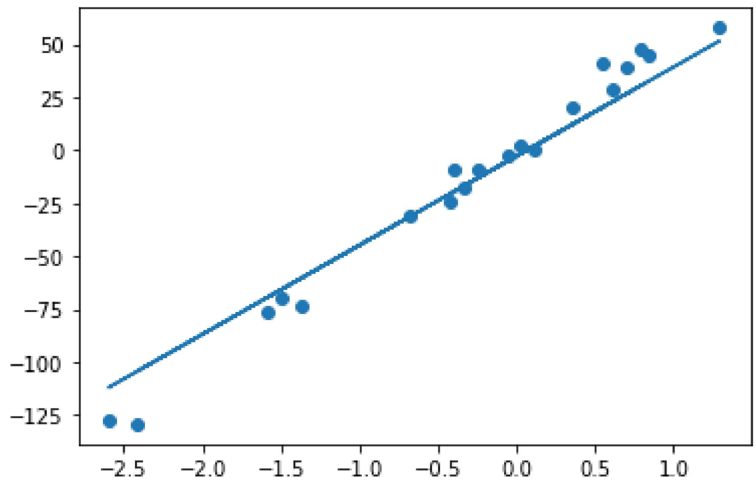
In [12]:

```
m = 0
b = 3
lr = 0.001
hh = []
slope = []
intercept = []
for i in range(50):
    loss_slope_b = -2 * np.sum(y - m*x.ravel() - b)
    loss_slope_m = -2 * np.sum((y - m*x.ravel() - b) * x.ravel())

    b = b - (lr * loss_slope_b)
    m = m - (lr * loss_slope_m)
    yhat = np.sqrt(mean_squared_error(y,(m*x)+b))
    ht = hh.append(yhat)
    ss = slope.append(m)
    ii = intercept.append(b)
    print(f"Slope {m}, Y-intercept {b}, Loss {yhat}")
#     print(hh)

    plt.plot(x, slope[i] * x + intercept[i])
    plt.scatter(x, y)
    plt.show()
```

```
Slope 42.058224960935286, Y-intercept -2.685932561489421, Loss 11.074611
955936163
```




```
Slope 42.4374100997168, Y-intercept -2.614825016664843, Loss 10.74166244
3714178
```

In [13]:

```
class GDRegressor:
    def __init__(self,learning_rate,epochs):
        self.m = 0
        self.b = 0
        self.lr = learning_rate
        self.epochs = epochs

    def fit(self,x,y):
        #Calculate b using GD
        for i in range(self.epochs):
            loss_slope_b = -2 * np.sum(y - self.m*x.ravel() - self.b)
            loss_slope_m = -2 * np.sum((y - self.m*x.ravel() - self.b) * x.ravel())

            self.b = self.b - (self.lr * loss_slope_b)
            self.m = self.m - (self.lr * loss_slope_m)
        print(self.m, self.b)

    def predict(self,x):
        return self.m * x + self.b
```

In [14]:

```
gd = GDRegressor(0.001,50)
```

In [15]:

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

```
45.61262398674062 -2.1633782708418074
```

In [ ]:

```

```