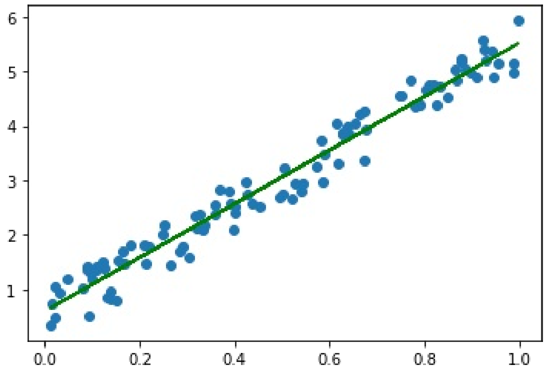
In [1]:
```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model
import time

X = np.random.random(100).reshape(-1, 1)
c = np.random.random(100).reshape(-1, 1)
m = 10
Y = m * X + c
W = 10000
```

In [9]:
```python
#Using SKLearn

def LinearReg_sklearn():
    reg = linear_model.LinearRegression()
    reg.fit(X,Y)
    print( 'Slope:', reg.coef_,'\n' 'Intercept: ', reg.intercept_, '\n')
    p=reg.coef_[0][0]
    q=reg.intercept_[0]
    plt.plot(X,p*X+q,color="green")
    plt.scatter(X,Y,label="GD using SKlearn")
    plt.show()
```

In [10]:
```python
start=time.process_time()
LinearReg_sklearn()
print("Runtime of the program is", time.process_time() - start)
```

```
Slope: [[4.92083055]]
Intercept:  [0.59908767]
```



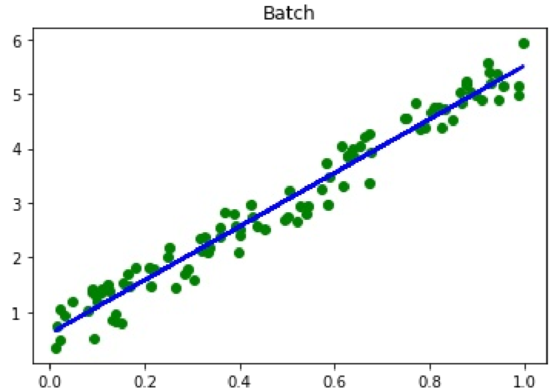

```
Runtime of the program is 0.546875
```

In [16]:
```python
#Batch GD
def Batch_GD(W):
    mr = np.random.random()
    cr = np.random.random()
    alpha = 0.01
    N = len(X)
    J = 0
    for j in range(W):
        dm = 0
        dc = 0
        for i in range(1, N):
            dm += (2/N)*X[i]*(mr*X[i]+cr - Y[i])
            dc += (2/N)*(mr*X[i]+cr - Y[i])
            J += (1/N)*(mr*X[i]+cr - Y[i])**2
        mr = mr - alpha * dm
        cr = cr - alpha * dc

    plt.scatter(X, Y,  color='green')
    plt.plot(X, mr*X+cr, color='blue', linewidth=2)
    plt.title("Batch")

    return mr,cr,J
```

In [17]:
```python
start=time.process_time()
slope,intercept,loss= Batch_GD(W)
print("slope=",slope)
print("intercept=",intercept)
```

```
print("loss=",loss)
print("Runtime of the program is", time.process_time() - start)
```

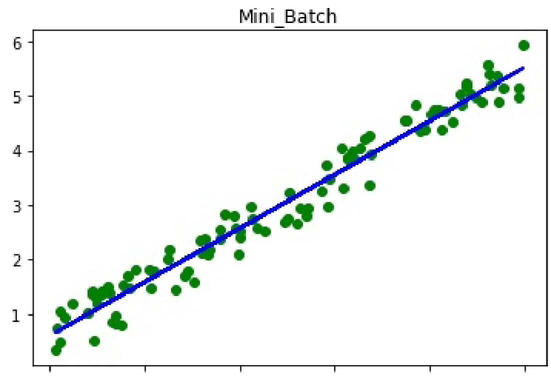
```
slope= [4.91657527]
intercept= [0.59863951]
loss= [1334.70280743]
Runtime of the program is 10.5
```



In [18]:
```
#Mini Batch GD
def minibatch_GD(W):
    mr = np.random.random()
    cr = np.random.random()

    g1 = np.array_split(X, 10)
    P = np.array(g1)
    g2 = np.array_split(Y, 10)
    Q = np.array(g2)
    N = len(P[0])
    alpha = 0.01
    J = 0
    for k in range(W):
        for j in range(0, len(P)):

            x = P[j]
            y = Q[j]
            dm = 0
            dc = 0
            for i in range(N):
                dm += (2/N)*x[i]*(mr*x[i]+cr - y[i])
                dc += (2/N)*(mr*x[i]+cr - y[i])
                J += (1/N)*(mr*x[i]+cr - y[i])**2
            mr = mr - alpha * dm
            cr = cr - alpha * dc

    Y_minibatch = mr*X+cr
    plt.scatter(X, Y,  color='green')
    plt.plot(X, Y_minibatch, color='blue', linewidth=2)
    plt.title("Mini_Batch")


    return mr, cr, J
```

In [19]:
```
start=time.process_time()
slope,intercept,loss= minibatch_GD(W)
print("slope=",slope)
print("intercept=",intercept)
print("loss=",loss)
print("Runtime of the program is", time.process_time() - start)
```

```
slope= [4.92133103]
intercept= [0.59953479]
loss= [8321.61401566]
Runtime of the program is 10.328125
```



0.0 0.2 0.4 0.6 0.8 1.0

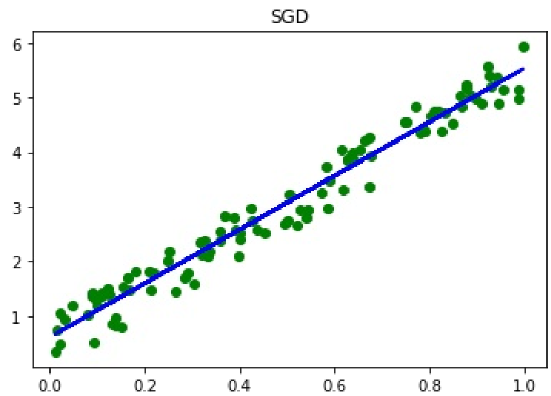
In [22]:

```
#SGD
def SGD(W):
    mr = np.random.random()
    cr = np.random.random()
    alpha = 0.01
    N = 1
    J = 0
    for j in range(W):
        for i in range(len(X)):
            dm = 0
            dc = 0

            dm += (2/N)*X[i]*(mr*X[i]+cr - Y[i])
            dc += (2/N)*(mr*X[i]+cr - Y[i])
            J += (1/N)*(mr*X[i]+cr - Y[i])**2
            mr = mr - alpha * dm
            cr = cr - alpha * dc
    Y_batch = mr*X+cr
    plt.scatter(X, Y,  color='green')
    plt.plot(X, Y_batch, color='blue', linewidth=2)
    plt.title("SGD")


    return mr, cr, J
```

In [23]:

```
start=time.process_time()
slope,intercept,loss= SGD(W)
print("slope=",slope)
print("intercept=",intercept)
print("loss=",loss)
print("Runtime of the program is", time.process_time() - start)
```

```
slope= [4.92773241]
intercept= [0.60773116]
loss= [81171.79686153]
Runtime of the program is 11.84375
```



In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js