Code for part 1

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

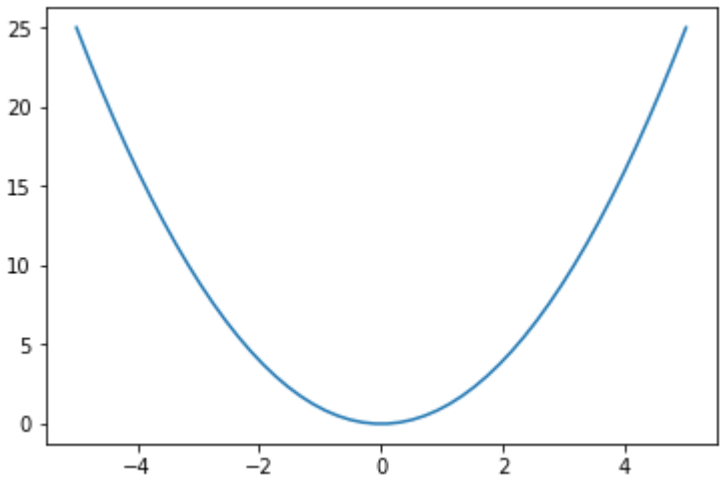
Plotting

In [3]:

```
#function f(x)=x^2
fig, ax = plt.subplots()
a = np.linspace(-5,5,50)
ax.plot(a,a*a, c='C0')
```

Out[3]:

```
[<matplotlib.lines.Line2D at 0x16ee4233940>]
```



Gradient Descent

In [4]:

```
x=3
# Parameters required for Gradient Descent
alpha = 0.1   #learning rate
np.random.seed(10)
def gradient_descent(x,alpha):
    cost_list=[]
    theta_list=[]
    prediction_list=[]
    cost_list.append(1e2)#large inital cost=10^2
    run=True
    i=0
    #iterating gradient descent
    while run:
        cost=x #cost=x
        cost_list.append(cost)
        x=x-(alpha*2*x) #x=x- alpha * 2*x
        if cost_list[i]-cost_list[i+1]< 1e-9:#checking if the change in cost function
            run=False
        i=i+1
    cost_list.pop(0)#remove intital cost
    return prediction_list, cost_list, theta_list
```

In [17]:

```
prediction_list, cost_list, theta_list = gradient_descent(x,alpha)
```

In [18]:

```
#Final Cost
cost_list[-1]
```

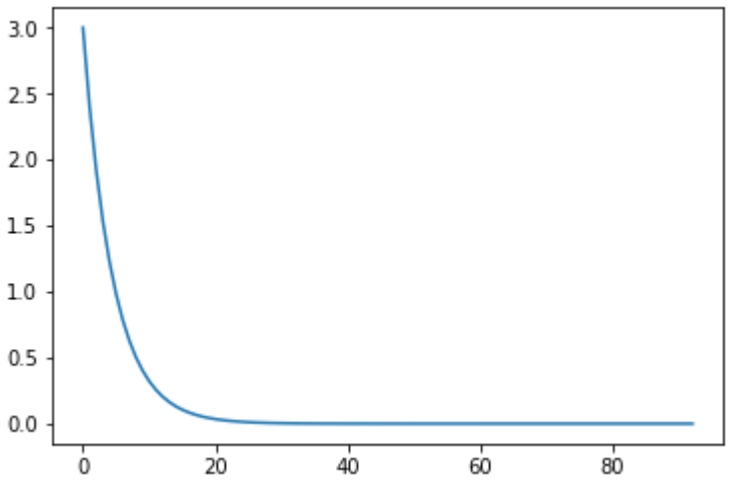
Out[18]:

```
3.6425041729232455e-09
```

Cost Function

In [19]:

```
#plot of decreasing cost
plt.plot(cost_list)
plt.show()
```



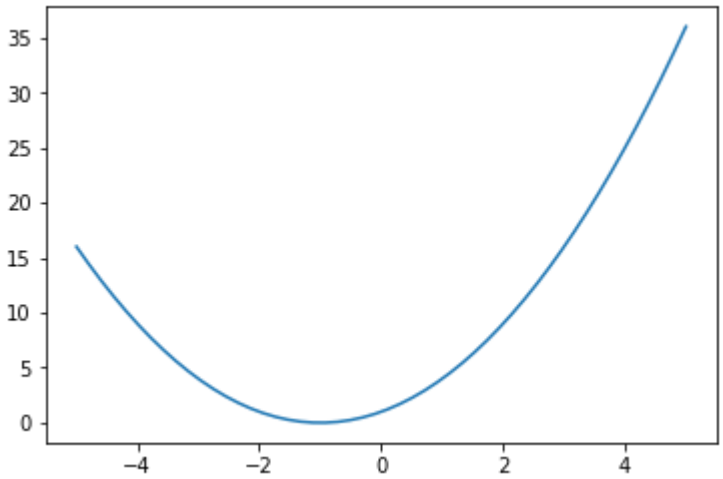
Code for part 2

In [6]:

```
#Function f(x)=(x+1)^2
fig, ax = plt.subplots()
a = np.linspace(-5,5,50)
ax.plot(a,(a+1)**2, c='C0')
```

Out[6]:

```
[<matplotlib.lines.Line2D at 0x16ee49a0470>]
```



In [7]:

```
f_x1=(x+1)**2
alpha = 0.1   #learning rate
#m = y.size  #no. of samples
np.random.seed(10)
def gradient_descent(x,alpha):
    cost_list=[]
    theta_list=[]
    prediction_list=[]
    cost_list.append(1e2)#large inital cost=10^2
    run=True
    i=0
    #iterating gradient descent
    while run:
        cost=(x+1)**2
        cost_list.append(cost)
        x=x-(alpha*2*(x+1)) #x=x- alpha * 2(x+1)
        if cost_list[i]-cost_list[i+1]< 1e-9:#checking if the change in cost function
            run=False
        i=i+1
    cost_list.pop(0)#remove intital cost
    return prediction_list, cost_list, theta_list
prediction_list, cost_list, theta_list = gradient_descent(x,alpha)
```

In [8]:

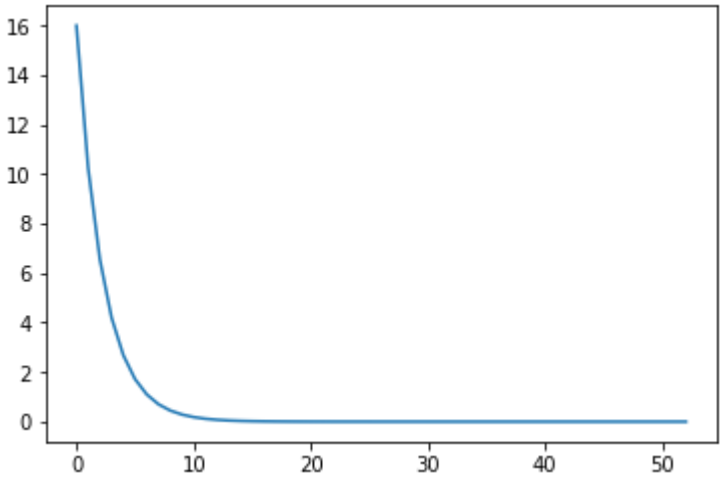
```
#Final Cost
cost_list[-1]
```

Out[8]:

```
1.3349918974486118e-09
```

In [9]:

```
#plot of Decreasing cost
plt.plot(cost_list)
plt.show()
```



Code for Part 3

In [10]:

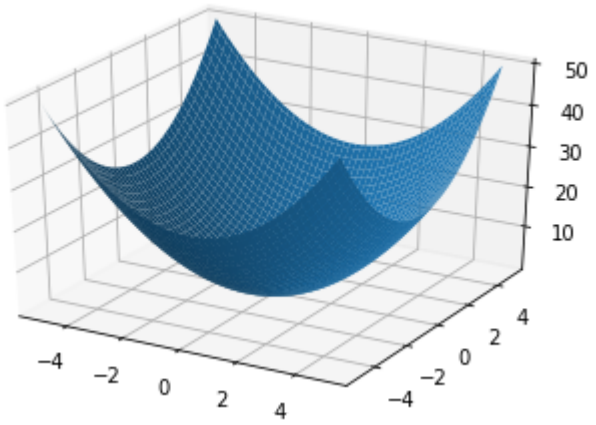
```
alpha = 0.1   #learning rate
np.random.seed(10)
x1=3
x2=3
def gradient_descent(x1,x2,alpha):
    cost_list=[]
    theta_list=[]
    prediction_list=[]
    cost_list.append(1e2)#large inital cost=10^2
    run=True
    i=0
    #iterating gradient descent
    while run:
        cost=x1**2+x2**2 #cost=x1^2+x2^2
        cost_list.append(cost)
        x1=x1-(alpha*2*x1) #x1=x1- alpha * 2 * x1
        x2=x2-(alpha*2*x2) #x2=x2- alpha * 2 * x2
        if cost_list[i]-cost_list[i+1]< 1e-9:#checking if the change in cost function
            run=False
        i=i+1
    cost_list.pop(0)#remove intital cost
    return prediction_list, cost_list, theta_list
prediction_list, cost_list, theta_list = gradient_descent(x1,x2,alpha)
```

In [11]:

```
#Plotting of f(x1,x2)=x1^2+x2^2
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = fig.add_subplot(1,1,1,projection='3d')
a0 = np.linspace(-5,5,100)
a1 = np.linspace(-5,5,100)
aa0, aa1 = np.meshgrid(a0, a1)
ax.plot_surface(aa0, aa1, aa0*aa0+aa1*aa1)
```

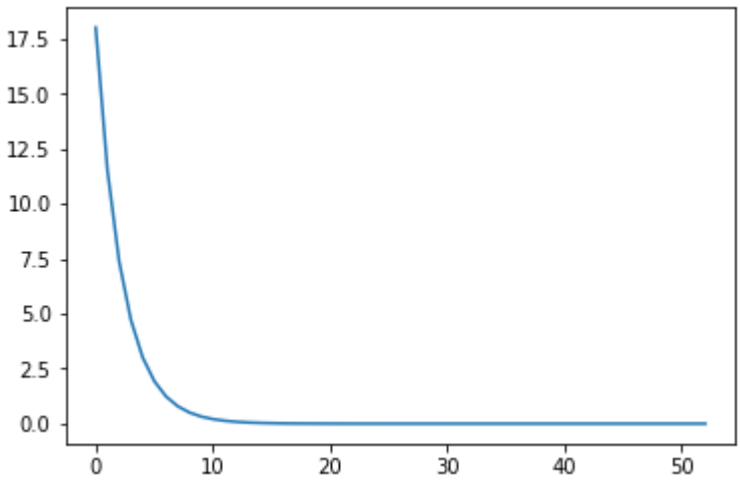
Out[11]:

```
<mpl_toolkits.mplot3d.art3d.Poly3DCollection at 0x16ee4b38780>
```



In [12]:

```
#plot of decreasing cost
plt.plot(cost_list)
plt.show()
```



In [ ]:

```

```

In [ ]:

```

```