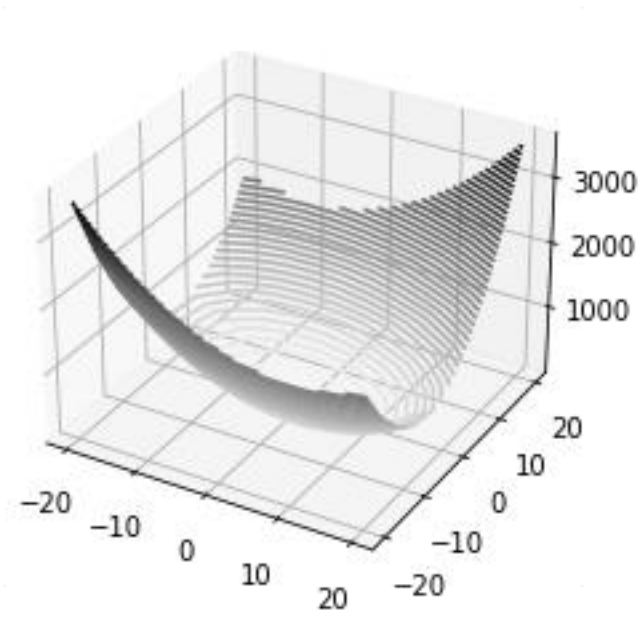
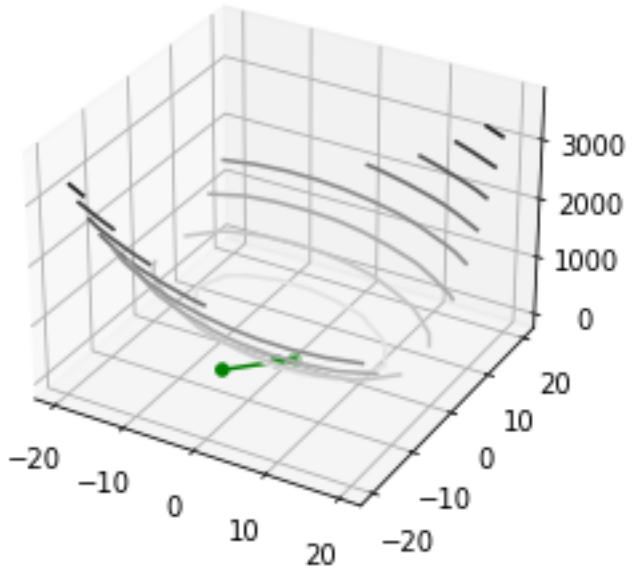
```
# programmer - Sophia Quinton
# date - 11-17-21
# class - DSC -540
# assignment - Assignment 3

#libraries
import matplotlib.pyplot as plt
import numpy as np

#(VanderPlas, nd)
def f(x,y):
    return 2.0*x**2.0 + 2.0*x*y + 5.0*y**2.0
x = np.linspace(-20, 20,10)
y = np.linspace(-20,20, 10)
X,Y = np.meshgrid(x,y)
Z = f(X,Y)
ax = plt.axes(projection='3d')
ax.contour3D(X,Y,Z, 50,cmap='binary')
```

```
<matplotlib.contour.QuadContourSet at 0x191d6007820>
```

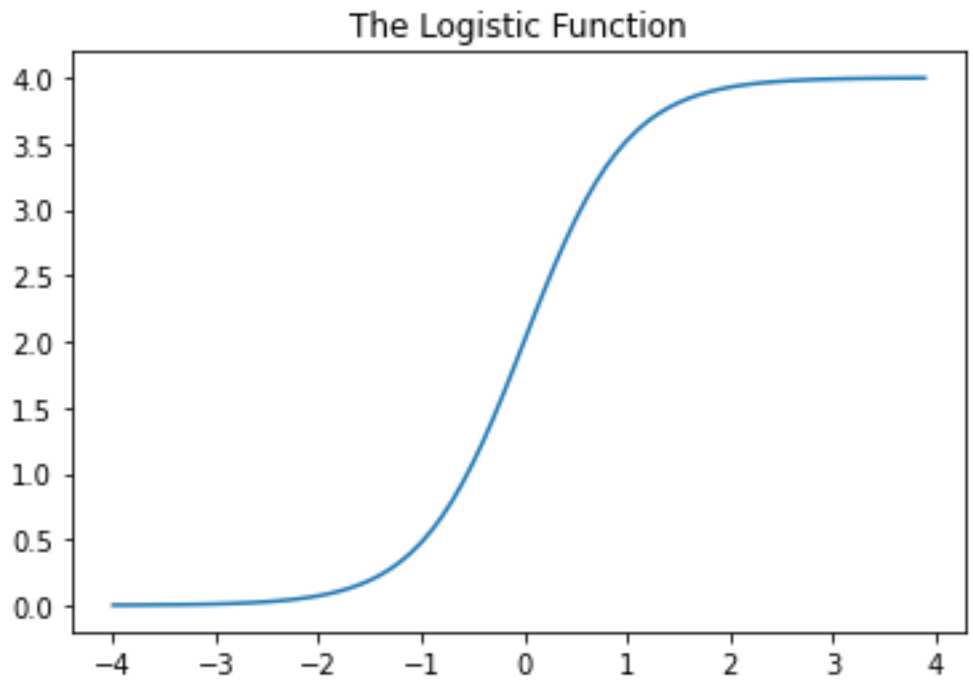



```
#(RealPython, 2021)
start_point = [2, -2, f(2,-2)]
def gradient_descent(gradient, start, learn_rate, n_itter=2):
    vector = start
    for _ in range(n_itter):
        diff = -learn_rate * gradient(vector)
        vector += diff
    return vector

first = gradient_descent(gradient=lambda v: np.array([2*v[0]**2 + 2*v[0]*v[1]
```

```
+ 5*v[1]**2]), start= start_point, learn_rate=0.2, n_itter=1)
second = gradient_descent(gradient=lambda v: np.array([2*v[0]**2 +
2*v[0]*v[1] + 5*v[1]**2]), start= start_point, learn_rate=0.2, n_itter=2)

first

array([-2., -6., 16.])

second

array([-44.4, -48.4, -26.4])

f(2,-2)

20.0

#(RealPython, 2021)
start_point = [2, -2, f(2,-2)]
def gradient_descent(gradient, start, learn_rate, n_itter=2):
    vector = start
    for _ in range(n_itter):
        diff = -learn_rate * gradient(vector)
        vector += diff
    return vector

first = gradient_descent(gradient=lambda v: np.array([2*v[0]**2 + 2*v[0]*v[1]
+ 5*v[1]**2]), start= start_point, learn_rate=0.08, n_itter=1)
second = gradient_descent(gradient=lambda v: np.array([2*v[0]**2 +
2*v[0]*v[1] + 5*v[1]**2]), start= start_point, learn_rate=0.08, n_itter=2)

first

array([ 0.4, -3.6, 18.4])

second

array([-4.5792, -8.5792, 13.4208])

#graph (Kite, nd)
x_points = [start_point[0], first[0], second[0]]
y_points = [start_point[1], first[1], second[1]]
z_points = [start_point[2], first[2], second[2]]
ax = plt.axes(projection='3d')
ax.contour3D(X,Y,Z, 10,cmap='binary')

ax.scatter(x_points, y_points, z_points, color='green')
ax.plot(x_points, y_points, z_points, color='green')

[<mpl_toolkits.mplot3d.art3d.Line3D at 0x191d62183d0>]
```

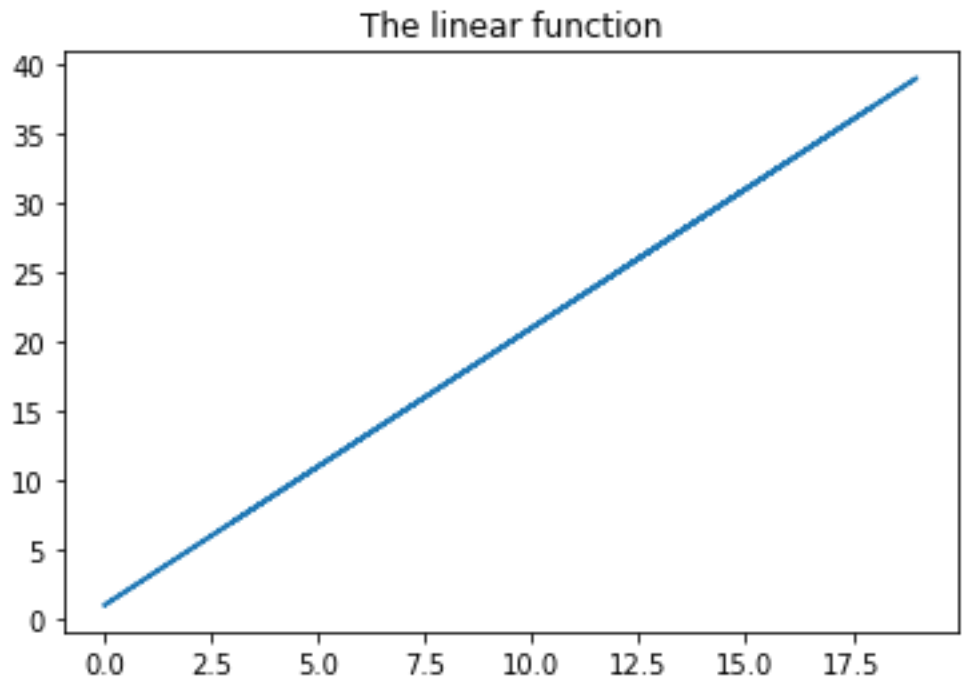



```
##part 3 (nbshare notebooks, nd)
def f2(x, x0, k, L):
    return L/(1+np.exp(-k*(x-x0)))

x = np.arange(start=-4, stop=4, step=0.1)
x0 = 0
L=4
log_funct = f2(x=x, x0=x0, k=2, L=L)
plt.plot(x, log_funct)
plt.title("The Logistic Function")
plt.show()
#x0 is starting
#x is the array of random numbers
#L = max
#k is the step
```




```
import random
def f3(x,b, m):
    final_result = []
    for i in range(len(x)):
        result = x[i]*m + b
        final_result.append(result)
    return final_result

x2 = random.sample(range(0,20), 10) #(PYNative, 2021)
print(x2)
b = 1
m = 2
lin_funct = f3(x=x2, b=b, m=m)
plt.plot(x2, lin_funct)
plt.title("The linear function")
plt.show()
```

```
[16, 3, 15, 9, 0, 14, 19, 12, 18, 11]
```


The linear function
40
35
30
25
20
15
10
5
0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5