Task8

June 20, 2020

1 Task 8

[118]:
```
import random
import numpy as np
import matplotlib.pyplot as plt
import math
import statistics as st
```

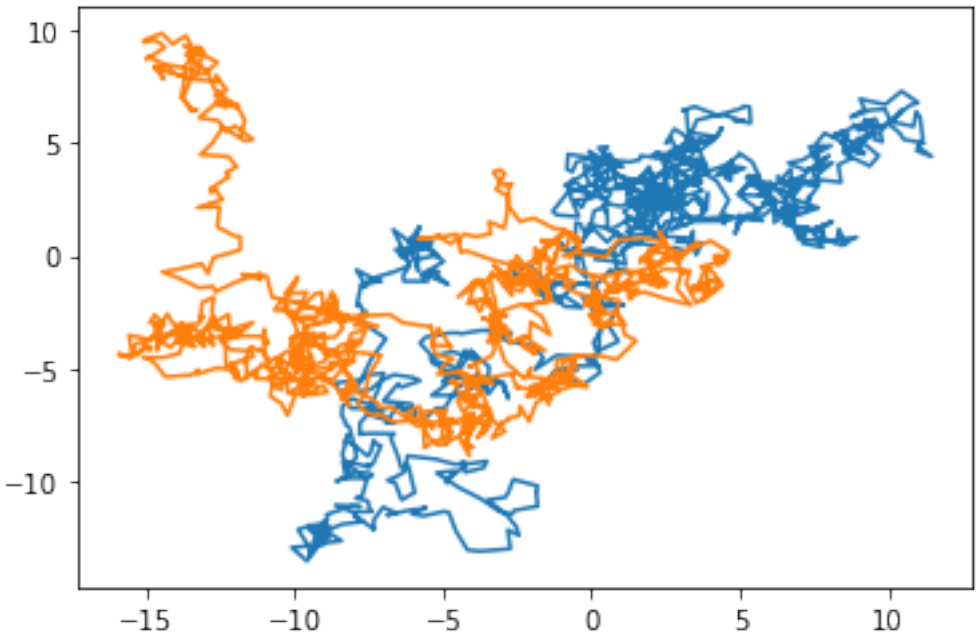
[119]:
```
def task5(steps,x,y):
    theta_vals = [0]
    r_vals = [0, 0.5, 1]
    theta_val = 0
    x_vals, y_vals = [], []
    x,y = 0,0

    for i in range(steps):
        step = random.uniform(0,1)
        theta_step = random.uniform(0, 2*math.pi)
        x += step*math.cos(theta_step)
        y += step*math.sin(theta_step)
        x_vals.append(x)
        y_vals.append(y)
        if math.sqrt(x**2+y**2) > 100:
            x = -x
            y = -y

    return x_vals, y_vals
```

[120]:
```
def location_choice(num_loc):
    locs = []
    for i in range(num_loc):
        r = random.uniform(0,100)
        theta = random.uniform(0, 2*math.pi)
        x = round(r*math.cos(theta),2)
        y = round(r*math.sin(theta), 2)
        locs.append((x,y))
```

```
    return locs
```

[121]:
```
locs = location_choice(2)
x_vals_a, y_vals_a = task5(1000, locs[0][0], locs[0][1])
x_vals_b, y_vals_b = task5(1000, locs[1][0], locs[1][1])
```

[122]:
```
plt.plot(x_vals_a, y_vals_a)
plt.plot(x_vals_b, y_vals_b)
plt.show()
```



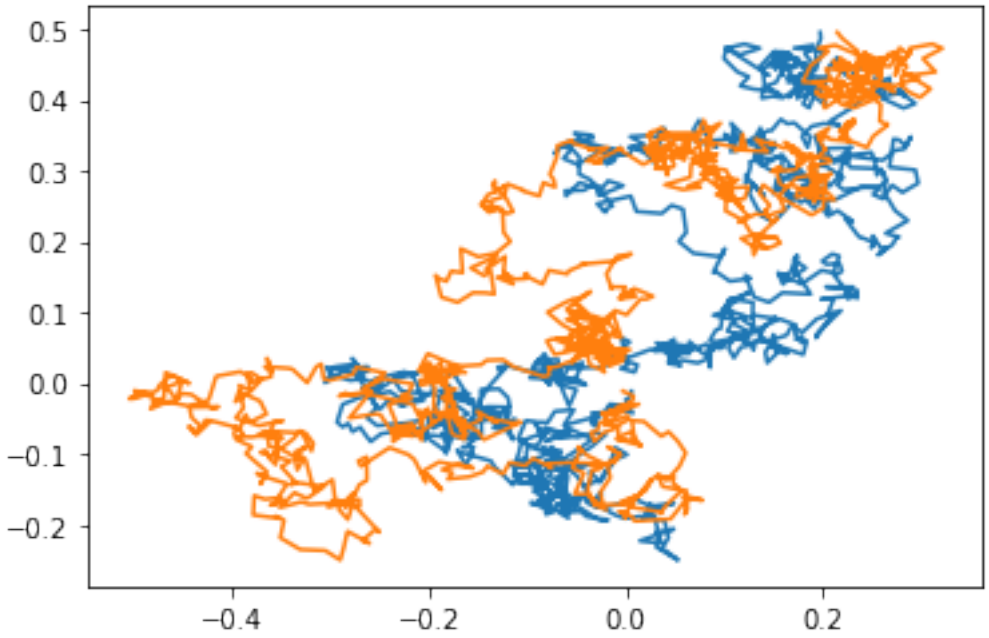
1.1 Simulation to find the Average Step

[123]:
```
def RunSimulation(sim_num):
    locs = location_choice(2)
    x_valsa_mult = []
    y_valsa_mult = []
    x_valsb_mult = []
    y_valsb_mult = []
    for i in range(sim_num):

        x_vals_a, y_vals_a = task5(1000, locs[0][0], locs[0][1])
        x_vals_b, y_vals_b = task5(1000, locs[1][0], locs[1][1])

        x_valsa_mult.append(x_vals_a)
```

```
        y_valsa_mult.append(y_vals_a)
        x_valsb_mult.append(x_vals_b)
        y_valsb_mult.append(y_vals_b)

    return x_valsa_mult, y_valsa_mult, x_valsb_mult, y_valsb_mult
```

[124]:
```
def AvgRandWalk(x_vals_mult, y_vals_mult):
    x_avg_dist = []
    y_avg_dist = []
    for i in range(len(x_vals_mult[0])):
        x = 0
        y = 0
        for j in range(len(x_vals_mult)):
            x += x_vals_mult[j][i]
            y += y_vals_mult[j][i]

        x_avg_dist.append(x/len(x_vals_mult))
        y_avg_dist.append(y/len(y_vals_mult))

    return x_avg_dist, y_avg_dist
```

[125]:
```
x_valsa_mult, y_valsa_mult, x_valsb_mult, y_valsb_mult = RunSimulation(1000)
avg_axvals, avg_ayvals = AvgRandWalk(x_valsa_mult, y_valsa_mult)
avg_bxvals, avg_ayvals = AvgRandWalk(x_valsb_mult, y_valsb_mult)

plt.plot(avg_axvals, avg_ayvals)
plt.plot(avg_bxvals, avg_ayvals)
plt.show()
```



[133]:
```
def AvgCloseStep(x_valsa_mult, y_valsa_mult, x_valsb_mult, y_valsb_mult):
    close_steps = [0 for i in range(len(x_vals_mult))]
    for i in range(len(x_valsa_mult)):
        for j in range(len(x_valsa_mult[i])):
            if math.sqrt((x_valsa_mult[i][j] - x_valsb_mult[i][j])**2 +␣
↪(y_valsb_mult[i][j] - y_valsa_mult[i][j])**2) <=1:
                close_steps[i] = j
                break
    return (st.mean(close_steps), round(st.stdev(close_steps),3))
```

[136]:
```
print("Expecated Step and Standard Deviation")
print((AvgCloseStep(x_valsa_mult, y_valsa_mult, x_valsb_mult, y_valsb_mult)))
```

```
Expecated Step and Standard Deviation
(6.571, 45.991)
```

[139]:
```
def Averages(Sim1, Sim2):
    Expected_Steps = []
    for i in range(Sim1):
        x_valsa_mult, y_valsa_mult, x_valsb_mult, y_valsb_mult =␣
↪RunSimulation(Sim2)
        Expected_Steps.append(AvgCloseStep(x_valsa_mult, y_valsa_mult,␣
↪x_valsb_mult, y_valsb_mult)[0])
    return Expected_Steps
```

[ ]:
```
Expected_Steps = Averages(100, 100)
plt.plot([i for i in range(100)], Expected_Steps)
plt.show()
```

[ ]:
```

```