Kernel: SageMath 10.4

Two-Dimensional Random Walks

Math 242 Modern Computational Math

Today we will start to understand properties of two-dimensional (2D) random walks.

Each position of a 2D random walk is an ordered pair of integers. Thus, a list of positions of a 2D random walk can be stored as a $N \times 2$ matrix, where N is the number of steps in the random walk.

Before we generate 2D random walks, we should practice using matrices in Sage.

In [4]:
```
mat = matrix( [[1,2,3], [4,5,6], [7,8,9]] )
mat
```

Out[4]:
```
[1 2 3]
[4 5 6]
[7 8 9]
```

In [10]:
```
# select a row
mat[0]
```

Out[10]:
```
(1, 2, 3)
```

In [12]:
```
# select a column
mat[:,1]
```

Out[12]:
```
[2]
[5]
[8]
```

In [8]:
```
# select an individual entry
mat[1,2]
```

Out[8]:
```
6
```

In [15]:
```
for r in mat.rows():
    print(r)
```

Out[15]:
```
(1, 2, 3)
(4, 5, 6)
(7, 8, 9)
```

In [16]:
```
for c in mat.columns():
    print(c)
```

Out[16]:
```
(1, 4, 7)
(2, 5, 8)
(3, 6, 9)
```

In [19]:
```
mat2 = mat.insert_row(1, [10, 11, 12])
```

In [18]:
```
mat
```

Out[18]:
```
[1 2 3]
[4 5 6]
[7 8 9]
```

In [20]:
```
mat2
```

Out[20]:
```
[ 1  2  3]
[10 11 12]
[ 4  5  6]
[ 7  8  9]
```

Now we can generate a 2D random walk.

In [24]:
```
# define the possible moves at each step of the random walk
dirs = matrix( [[0,1],[0,-1],[1,0],[-1,0]] )

# define the number of steps to take
numSteps = 50

# set up a numpy 2-D array to store the locations visited.
locations = zero_matrix(numSteps, 2)

# take steps
for i in range(1, numSteps):
    r = randrange(4)  # random value 0, 1, 2, or 3
    move = dirs[r]    # choose the direction at index r of dirs
    print(move)
    locations[i] = locations[i-1] + move

# output the random walk
print(locations)
```

Out[24]:
```
(-1, 0)
(1, 0)
(-1, 0)
(0, -1)
(0, 1)
(1, 0)
(1, 0)
(0, -1)
(0, -1)
(1, 0)
(-1, 0)
(0, 1)
(0, 1)
(1, 0)
(1, 0)
(0, 1)
(0, -1)
(0, 1)
(0, 1)
(-1, 0)
(-1, 0)
(0, -1)
(-1, 0)
(0, 1)
(1, 0)
(0, -1)
(1, 0)
(0, -1)
(-1, 0)
(0, -1)
(0, -1)
(0, 1)
(0, 1)
(0, -1)
(1, 0)
(0, -1)
(0, 1)
(0, 1)
(1, 0)
(0, -1)
(0, 1)
(0, 1)
(1, 0)
(0, 1)
(-1, 0)
(1, 0)
(0, -1)
(1, 0)
(-1, 0)
[ 0  0]
[-1  0]
[ 0  0]
[-1  0]
[-1 -1]
[-1  0]
[ 0  0]
[ 1  0]
[ 1 -1]
[ 1 -2]
[ 2 -2]
[ 1 -2]
```

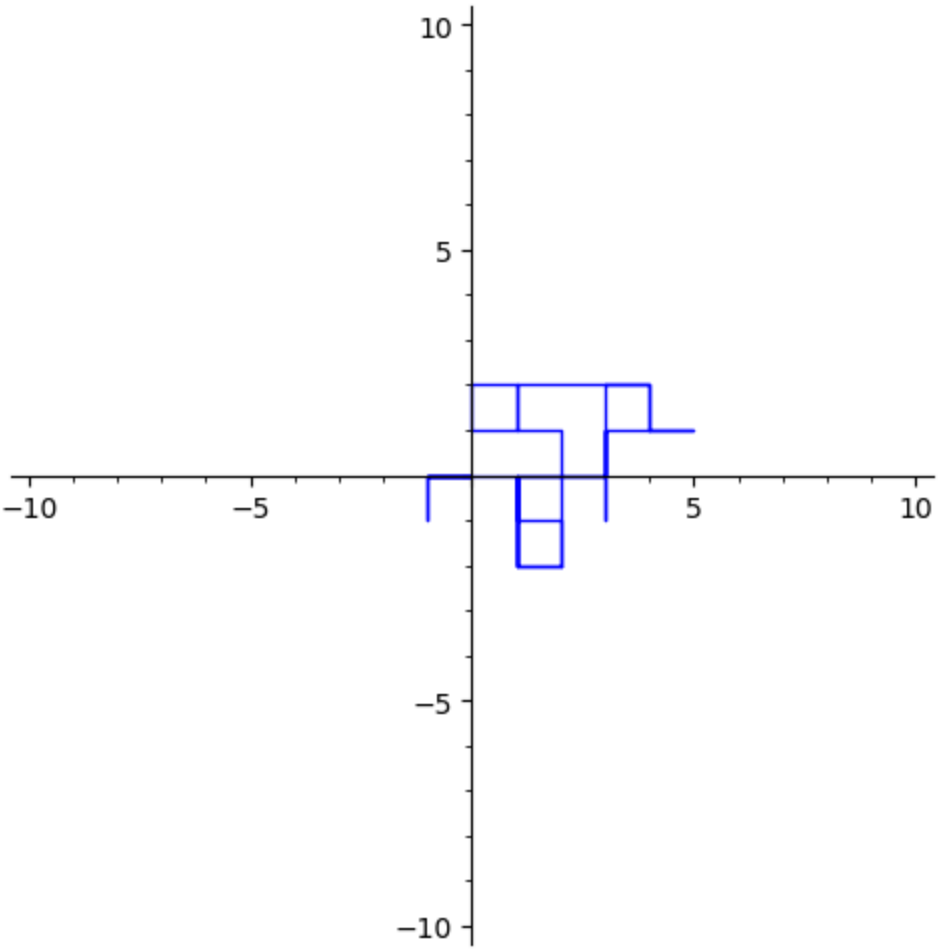
```
[ 1 -1]
[ 1  0]
[ 2  0]
[ 3  0]
[ 3  1]
[ 3  0]
[ 3  1]
[ 3  2]
[ 2  2]
[ 1  2]
[ 1  1]
[ 0  1]
[ 0  2]
[ 1  2]
[ 1  1]
[ 2  1]
[ 2  0]
[ 1  0]
[ 1 -1]
[ 1 -2]
[ 1 -1]
[ 1  0]
[ 1 -1]
```

```
[ 2 -1]
[ 2 -2]
[ 2 -1]
[ 2  0]
[ 3  0]
[ 3 -1]
[ 3  0]
[ 3  1]
[ 4  1]
[ 4  2]
[ 3  2]
[ 4  2]
[ 4  1]
[ 5  1]
[ 4  1]
```

We can also plot the random walk. Here, we will make a 2D plot.

In [25]:
```
list_plot(locations, plotjoined=True, xmin=-10, xmax=10, ymin=-10,
ymax=10, figsize=[5,5])
```

Out[25]:



Now let's put our 2D random walk code into a function, so we can easily generate lots of 2D random walks.

In [26]:
```
def randomWalk2D(numSteps):
    # define the possible moves at each step of the random walk
    dirs = matrix( [[0,1],[0,-1],[1,0],[-1,0]] )
```

```
    # set up a numpy 2-D array to store the locations visited.
    locations = zero_matrix(numSteps, 2)

    # take steps
    for i in range(1, numSteps):
        r = randrange(4)  # random value 0, 1, 2, or 3
        move = dirs[r]    # choose the direction at index r of dirs
        #print(move)
        locations[i] = locations[i-1] + move

    # output the random walk
    return locations
```

In [28]:
```
rw= randomWalk2D(40)
print(rw)
```

Out[28]:
```
[ 0  0]
[ 0  1]
[ 0  0]
[-1  0]
[-1  1]
[-1  0]
[-2  0]
[-3  0]
[-3 -1]
[-3 -2]
[-3 -1]
[-2 -1]
[-2 -2]
[-2 -3]
[-1 -3]
[-2 -3]
[-3 -3]
[-4 -3]
[-4 -4]
[-3 -4]
[-2 -4]
[-1 -4]
[-2 -4]
[-2 -3]
[-3 -3]
[-2 -3]
[-2 -4]
[-2 -3]
[-3 -3]
[-4 -3]
[-5 -3]
[-5 -4]
[-5 -5]
[-5 -6]
[-6 -6]
[-6 -5]
[-6 -6]
[-5 -6]
[-6 -6]
[-7 -6]
```

```
randomWalk2D(40)
```

Diameter of a 2D Random Walk

Define the *width* of a 2D random walk to be the maximum difference between any two x-coordinates attained by the walk. Similarly, define the *height* of the walk to be the maximum difference between any two y-coordinates. Define the *diameter* of the random walk to be the maximum of its width and height.

What is the average diameter of a 2D random walk after N steps? How does this depend on N?

In [32]:
```
# horizontal distance between min and max x-values
xdist = max(rw[:,0]) - min(rw[:,0])
xdist[0]
```

Out[32]: 7

In [33]:
```
# vertical distance between min and max y-values
ydist = max(rw[:,1]) - min(rw[:,1])
ydist[0]
```

Out[33]: 7

In [35]:
```
max(xdist[0], ydist[0])
```

Out[35]: 7

Four Quadrants

How likely is it that a 2D random walk visits all four quadrants of the plane within N steps? How does this depend on N?

In [0]:

In [0]:

In [0]: