Kernel: SageMath 10.7

Random Walks

MATH 242 Modern Computational Math

Imagine you are standing at the origin of a number line. You flip a fair coin. If the coin lands heads, you move to +1. If the coin lands tails, you move to -1. You flip the coin again and again. Whenever it lands heads, you move 1 unit in the positive direction. Whenever it lands tails, you move 1 unit in the negative direction. Your path on the number line is called a **one-dimensional random walk**.

What does a one-dimensional random walk look like? Let's create some with Python!

1. Building a Random Walk

Use `choice([-1,1])` to simulate one step of the random walk. Try it out:

In [8]:
```
choice( [-1,1] )
```

Out[8]: 1

Now simulate 100 steps of the random walk. Create a list called `locations` consisting of one hundred 0s. The first 0 is the starting location; the other 0s are placeholders for later locations. Compute each location after the first by choosing a +1 or -1 step at random and adding it to the previous location.

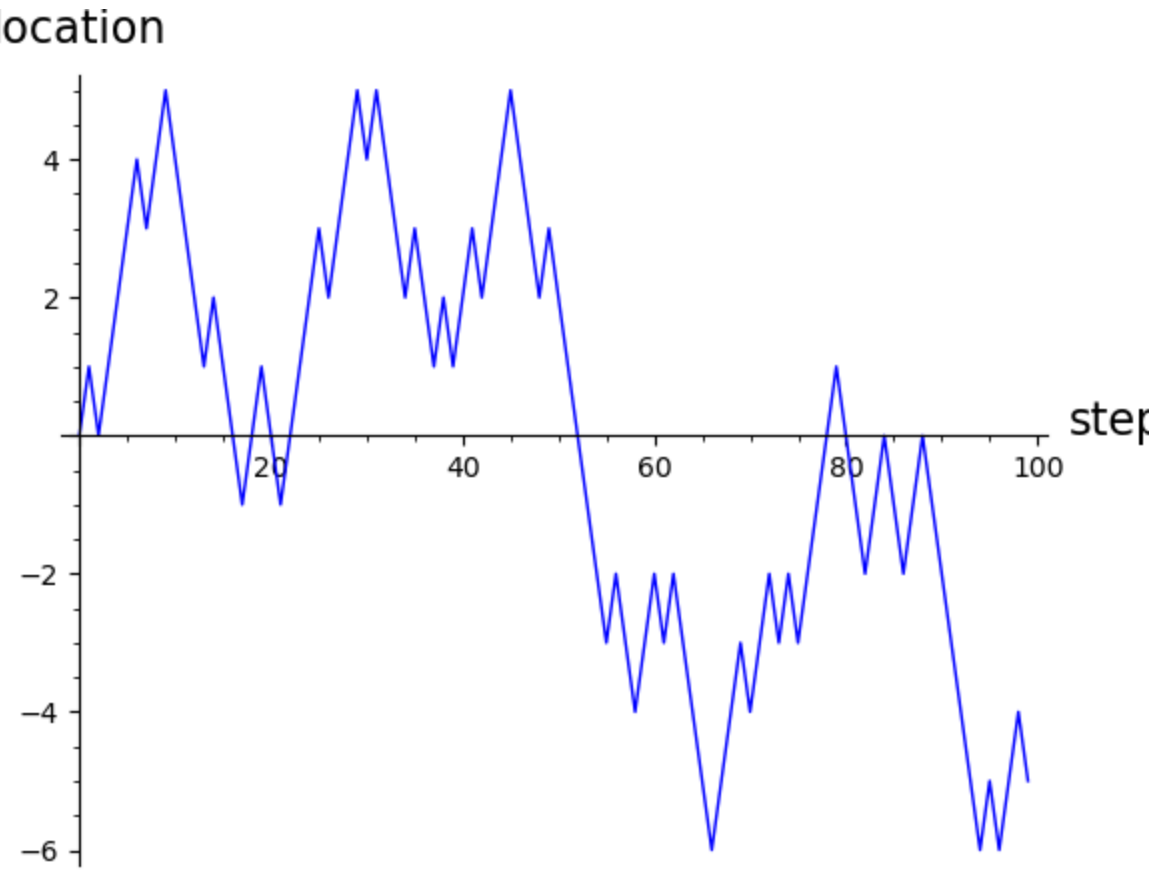
In [11]:
```
locations = [0]*100
for i in range(1,100):
    move = choice( [-1,1] )
    #print(move)
    locations[i] = locations[i-1] + move


print( locations )
```

Out[11]:
```
[0, 1, 0, 1, 2, 3, 4, 3, 4, 5, 4, 3, 2, 1, 2, 1, 0, -1, 0, 1, 0, -1, 0, 1, 2, 3, 2,
3, 4, 5, 4, 5, 4, 3, 2, 3, 2, 1, 2, 1, 2, 3, 2, 3, 4, 5, 4, 3, 2, 3, 2, 1, 0, -1,
-2, -3, -2, -3, -4, -3, -2, -3, -2, -3, -4, -5, -6, -5, -4, -3, -4, -3, -2, -3, -2,
-3, -2, -1, 0, 1, 0, -1, -2, -1, 0, -1, -2, -1, 0, -1, -2, -3, -4, -5, -6, -5, -6,
-5, -4, -5]
```

Next, make a plot of the random walk. The horizontal axis will give the number of steps, and the vertical axis will give the location at each step.

In [13]:
```
list_plot( locations , plotjoined=True, axes_labels=["steps","location"],
figsize=6)
```

Out[13]:



We will need to generate a lot of random walks. To make this easy, let's write a function that returns a random walk.

In [1]:

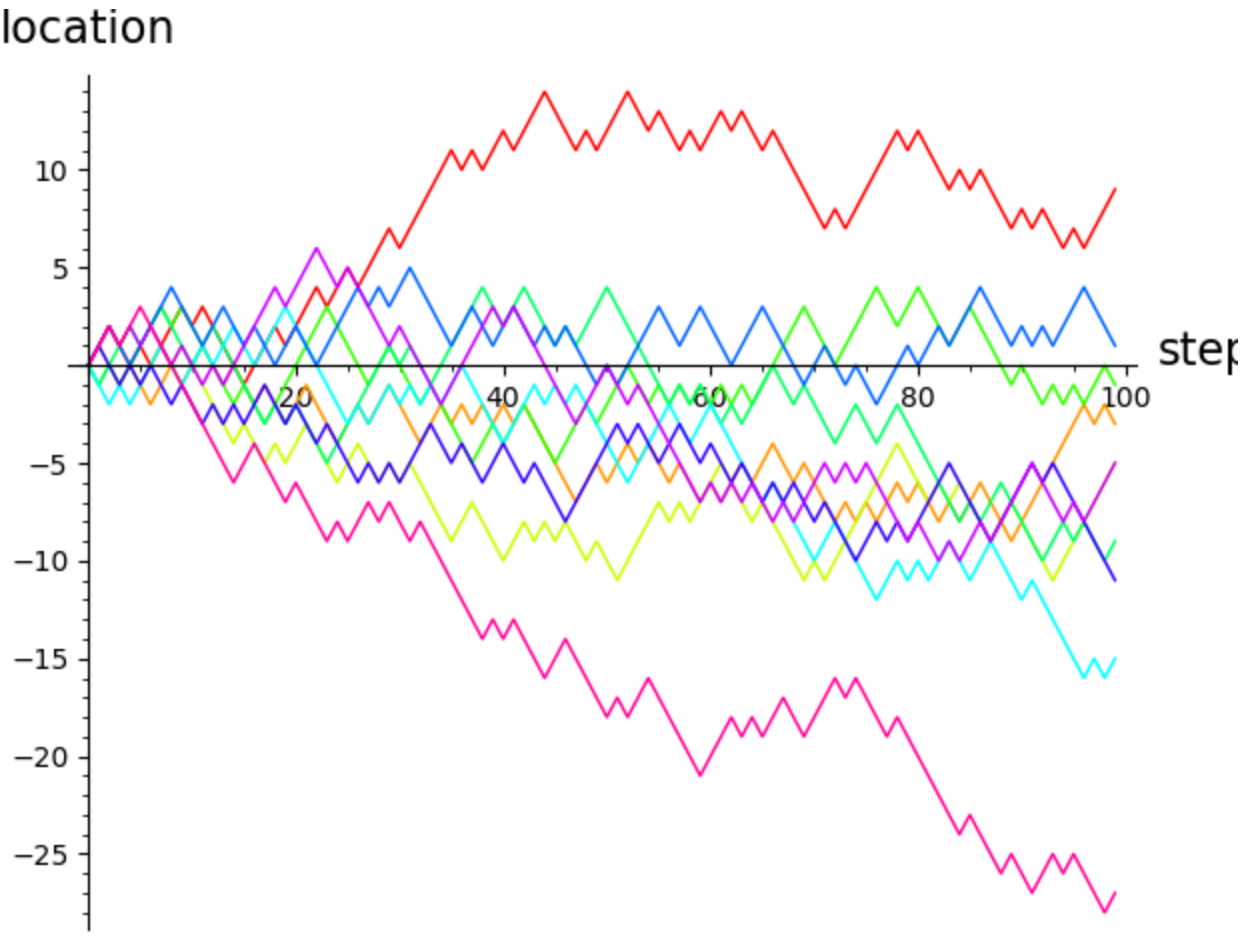
```
# function: randomWalk
# input: number of steps
# output: a random walks, returned as a list of numbers
def randomWalk(numSteps):
    locations = [0]*numSteps
    for i in range(1, numSteps):
        move = choice( [-1,1] )
        #print(move)
        locations[i] = locations[i-1] + move

    return locations
```

We can plot a bunch of random walks like this:

In [15]:

```
rwplots = [list_plot(randomWalk(100), hue=i/10, plotjoined=True) for i in
range(10) ]
rwplot = sum(rwplots)

show(rwplot, axes_labels=["steps","location"])
```

Out[15]:



2. Diameter of a Random Walk

The **diameter** of a random walk is the difference between the maximum and minimum locations in the walk.

Write a function that computes the diameter of a random walk. The input to your function should be a random walk (i.e., a list), and your function should return the diameter of the walk.

Note that Python has built-in functions **`min`** and **`max`** that return the minimum and maximum values in a list.

In [2]:

```
# function: diameter
# input: a random walk, specified as a list of numbers
# output: the diameter of the random walk, computed as (max value) - (min value)
def diameter(aWalk):
    return (max(aWalk) - min(aWalk))
```

Test your **`diameter`** function and confirm that it works as expected.

In [0]:

```

```

Now find the average diameter for random walks of a specified number of steps. Write a function **`avgDiam`** that takes two parameters: the number of steps in a random walk, and the number of

random walks to simulate. The function then returns the average diameter of those random walks.

In [5]:
```
# function: avgDiam
# input: numSteps, the number of steps per random walk
#        numWalks, the number of random walks to simulate
# output: the average diameter of the simulated random walks
def avgDiam(numSteps, numWalks):
    diams = [diameter(randomWalk(numSteps)) for _ in range(numWalks)]
    return sum(diams)/numWalks
```

In [6]:
```
#testing: average diameter of 100 random walks, each with 1000 steps
avgDiam(1000,100).n()
```

Out[6]: 48.4000000000000

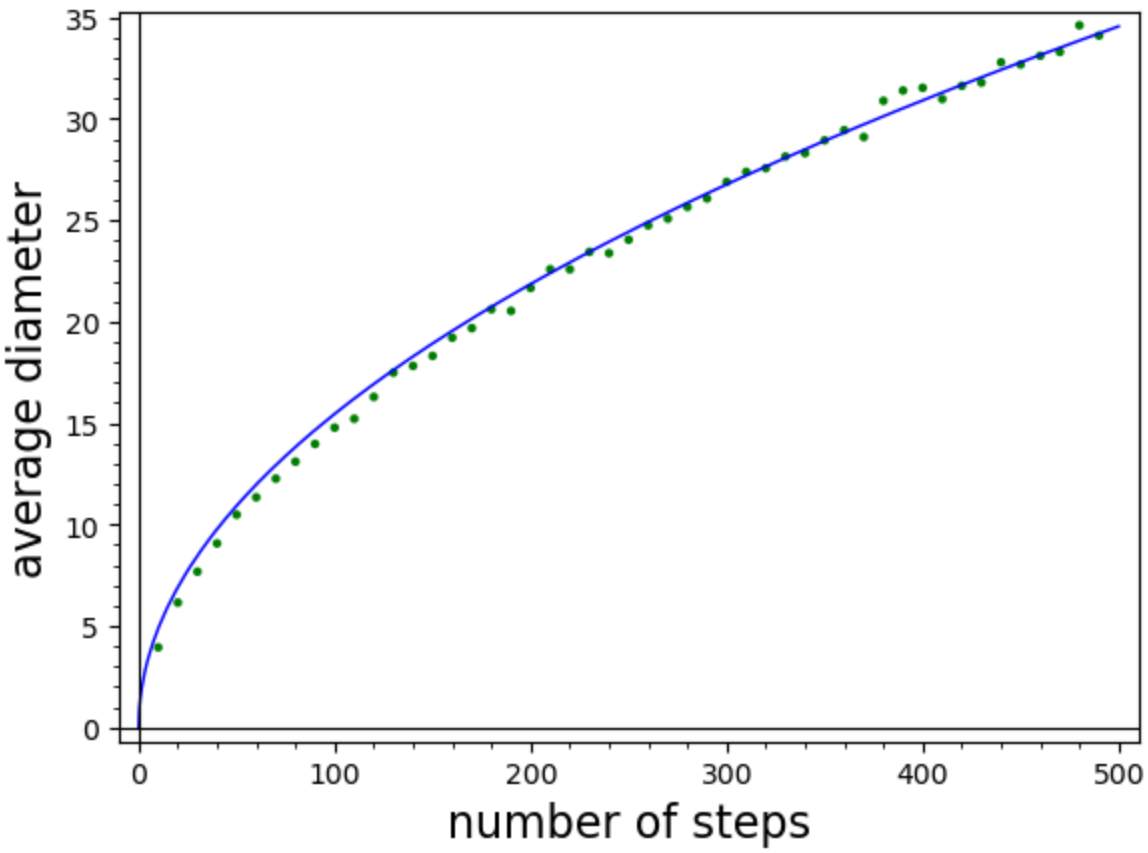
How does the diameter depend on the number of steps?

Create a plot that shows the average diameter of a random walk as a function of the number of steps. Make a conjecture for the growth rate of the average diameter.

In [8]:
```
#define some values of n
nvals = range(10, 500, 10)
# compute the average diameter for each value of n
avgDiams = [avgDiam(n,500) for n in nvals]
```

In [10]:
```
# make a plot
list_plot(list(zip(nvals,avgDiams)), color="green", axes_labels=["number of
steps","average diameter"], frame=True, figsize=6) + plot(sqrt(2.388*x), (x, 0,
500))
```

Out[10]:



3. Investigate your own questions!

What questions do you have about one-dimensional random walks? Investigate!

In [0]:

In [0]:

In [0]: