In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.path import Path
import random
```

1a. [5pts] Suppose that an ant wandered randomly by taking steps (x,y), one per second, where at each ant step, x and y come from a normal distribution with a mean of 0 and a standard deviation of 1.0mm (assume this for all questions below). Plot a trace of the ant's path over the course of an hour.

In [ ]:

```
#Pretty much says that ant chooses x and y from a NORMAL DISTRBITUION BECAUSE A
0 MEAN AND 1 STDEV IS a
#normal distribitoin
```

In [23]:

```
#60*60 is how many steps are made
path = [(0,0)]
current_x = 0
current_y = 0
for _ in range(360):
    r_x = np.random.normal()
    r_y = np.random.normal()
    current_x += r_x
    current_y += r_y
    path.append((current_x, current_y))
```

In [26]:

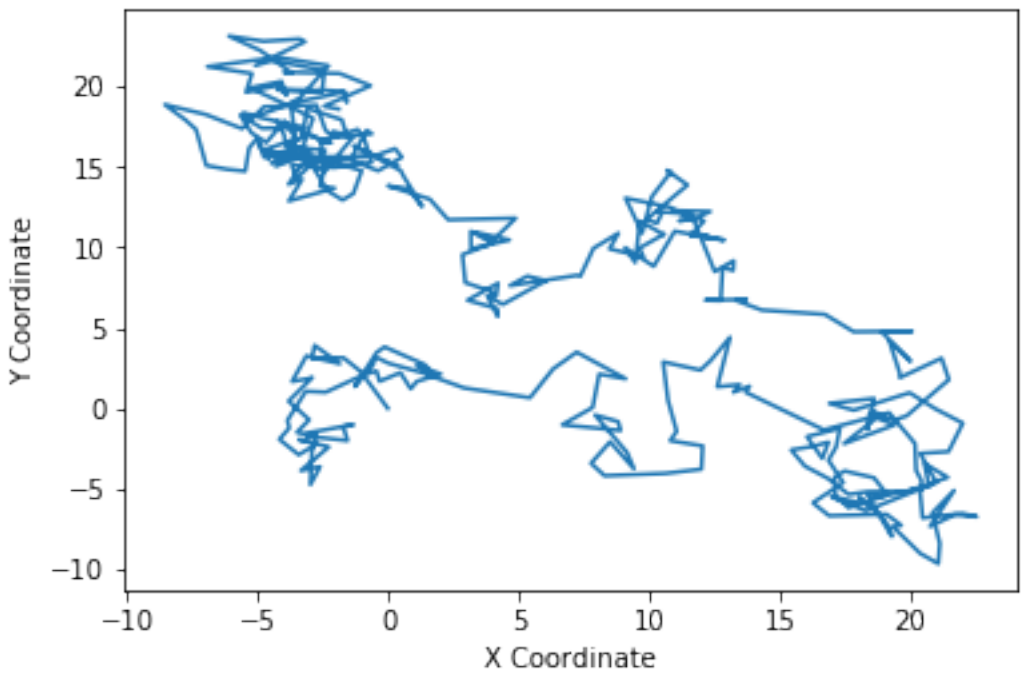
```
x, y = zip(*path)
```

In [30]:

```
plt.plot(x, y)
plt.xlabel('X Coordinate')
plt.ylabel('Y Coordinate')
```

Out[30]:

```
Text(0, 0.5, 'Y Coordinate')
```



1b. [10pts] Let's think about why ants need to perform path integration. Suppose that instead of path integration, when an ant found food, it just continued to wander with random steps until it got back to the nest. Using a simulation, find the probability that an ant who finds food after 1 hour will make its way back to within 10mm of the nest over the course of the next hour (note that if it comes within 10mm of a nest, it stops). Is this a good strategy? Why or why not?

In [81]:

```
food_x = current_x
food_y = current_y
success = 0

for _ in range(10000):
    for i in range(360):
        r_x = np.random.normal()
        r_y = np.random.normal()
        food_x += r_x
        food_y += r_y
        distance_from_nest = np.sqrt(food_x**2 + food_y**2 )
        if distance_from_nest < 10:
            success +=1
            break

success/10000
```

Out[81]:

```
0.0239
```

Random walking is not a good strategy, it leaves it up to chance, and there is the chance that the ant will head in the wrong direction. There is one right direction and multiple wrong directions. the further the ant is away from the nest the less likely it is to make it back home.

1c. [10pts] If the ant searches for an hour, finds food, and then searches for the nest by continuing to walk at random, what is the average closest distance it will come to the nest over the course of the next hour? (Do not assume it stops if it comes within 10mm) Find this with a simulation.

In [95]:

```
distances_from_nest = []
for _ in range(1000):
    distance_from_nest =  np.inf
    for i in range(360):
        r_x = np.random.normal()
        r_y = np.random.normal()
        food_x += r_x
        food_y += r_y
        distance_from_nest = min(np.sqrt(food_x**2 + food_y**2 ), distance_from_
nest)
    distances_from_nest.append(distance_from_nest)

np.mean(distances_from_nest)
```

Out[95]:

```
2480.4014138763177
```

1. [25pts] Now let's think about path integration. Assume that each step (x,y) is "remembered" (integrated) internally in the ant's brain with a standard deviation on each component of S. Thus, if we store the total X component, it gets updated with a new x step via X ← X+x+ex where ex ~ Gaussian(0,S) and similarly for Y (Y ← Y+y+ ey with ey ~ Gaussian(0,S)). Suppose that, upon finding food after an hour (as above, one step per second), the ant then heads straight back to where it thinks the nest is (e.g. it travels back via the vector (-X,-Y)). Thus, the outbound trip is noisy, but the return trip is noiseless. Run a simulation to see how far the ant will end from the nest for various S from 1.0mm down to 0.0001mm. Plot the mean distance the ant ends from the nest as a function of S. Be sure to show a range of S values that make it clear what's going on.

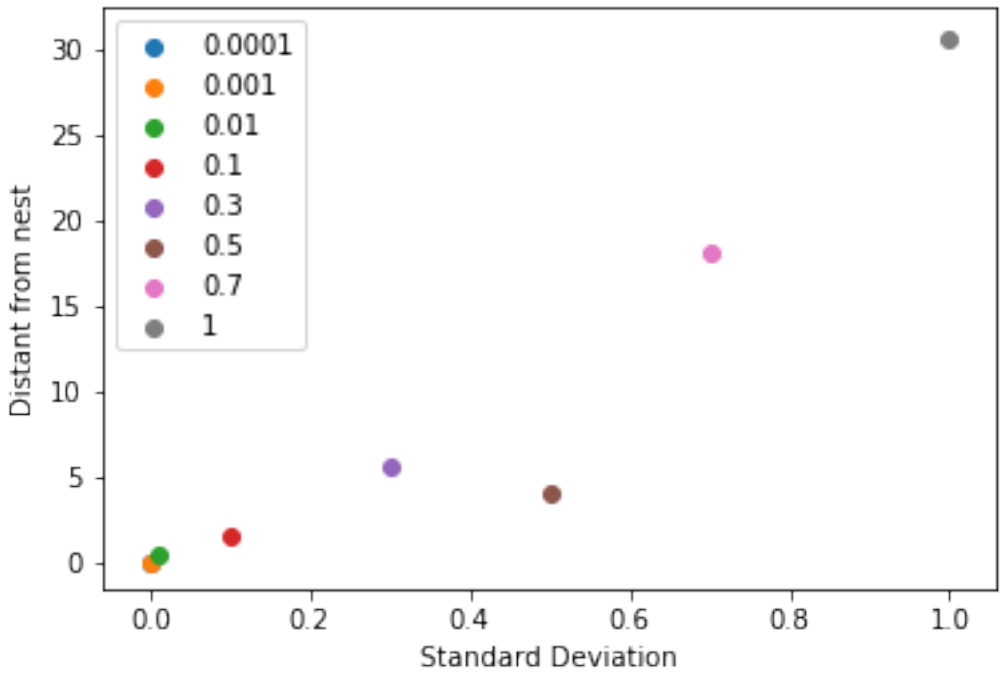
In [126]:

```
for S in [0.0001, 0.001, 0.01, 0.1, 0.3, 0.5, 0.7, 1]:
    for _ in range(1000):
        distances =  []
        current_x2 = 0
        current_y2 = 0
        total_x = 0
        total_y = 0
        for _ in range(360):
            r_x2 = np.random.normal()
            r_y2 = np.random.normal()
            x_noise = r_x2 + np.random.normal(scale = S)
            y_noise = r_y2 + np.random.normal(scale = S)
            total_x += x_noise
            total_y += y_noise
            current_x2 += r_x2
            current_y2 += r_y2
        distances.append(np.sqrt((current_x2 - total_x)**2 + (current_y2 - total
_y)**2))
    plt.scatter(S, np.mean(distances), label = S)

plt.legend()
plt.xlabel("Standard Deviation")
plt.ylabel("Distant from nest")
```

Out[126]:

```
Text(0, 0.5, 'Distant from nest')
```



3a. [20pts] Next, let's just assume that it requires exp(0.1/S) energy units to run an integrator with a standard deviation of S for an hour. Suppose further that if you end up at a distance d from the nest after your return trip, it will take you d2 energy units to find the nest. By using a simulation, plot the average energy expended while on a foraging trip (out for an hour and back) as a function of S. Be sure you have found a range of S to plot that shows the shape of the curve near its minimum.

In [171]:

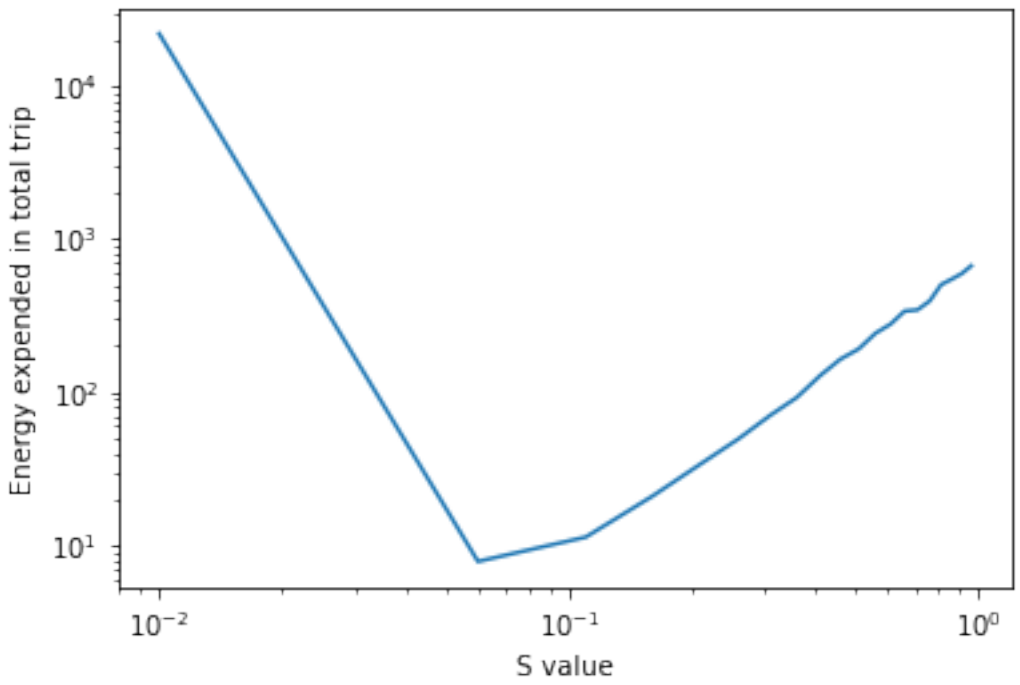
```
averages = []
for S in np.arange(0.01, 1.0, 0.05):
    ins= 0.1/S
    energies =  []
    for _ in range(500):
        #The walk out for an hour
        energy = np.exp(ins)
        current_x2 = 0
        current_y2 = 0
        total_x = 0
        total_y = 0
        for _ in range(360):
            r_x2 = np.random.normal()
            r_y2 = np.random.normal()
            current_x2 += r_x2
            current_y2 += r_y2
            x_noise = r_x2 + np.random.normal(scale = S)
            y_noise = r_y2 + np.random.normal(scale = S)
            total_x += x_noise
            total_y += y_noise
        d = np.sqrt((current_x2 - total_x)**2 + (current_y2 - total_y)**2)
        energy+= d**2
        energies.append(energy)
    averages.append(np.mean(energies))
```

In [173]:

```
S_values = np.arange(0.01, 1.0, 0.05)
plt.plot(S_values, averages)
plt.xscale('log')
plt.yscale('log')
plt.xlabel("S value")
plt.ylabel("Energy expended in total trip")
```

Out[173]:

```
Text(0, 0.5, 'Energy expended in total trip')
```



3b. [10pts] What is the evolutionary significance of the minimum of the plot in 3a?

There is a lot of fine tuning when it comes to the noise. Years of evolution have perfected the method of path integration.