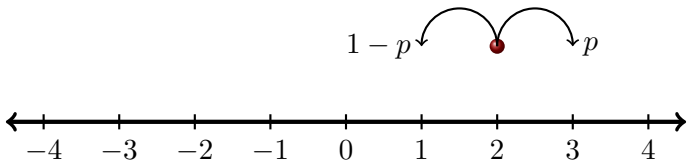
1 Background

A random walk is the process in which an objects wander away from its starting position, by taking random directions in every step. A simple example is in one dimension (1D). Imagine a point initially placed at the origin, it can move either to the left or to the right, the actual decision is random –e.g. based on a the result of tossing a coin–



In this class we will study the properties of a set of un-correlated walkers

2 Problems

1. First we will get familiar with the use of random numbers. Evaluate the statement

   ```
   print numpy.random.random()
   ```

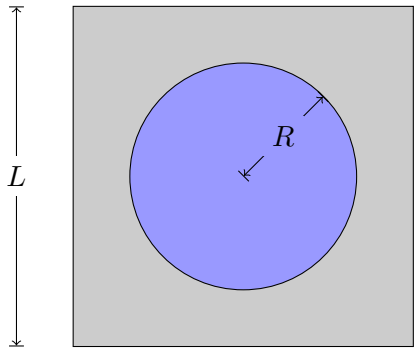
   a few times and explain the result

2. Now execute

   ```
   numpy.random.seed()
   print numpy.random.random()
   ```

   a few times and explain the result

3. As an example we will evaluate the value of π. Remember that the are of a circle of radius R if πR^2. Let us assume that $R = 1$ and choose a box of length $L = 3$.

   Now imagine you throw N darts that randomly hit the square. It is easy to realize that if n is the number of darts that fall inside the circle, then

$$\frac{n}{N} = \frac{\text{area(circle)}}{\text{area(square)}} = \frac{\pi R^2}{L^2}, \tag{1}$$

from which we can get the value of π. Create a function that finds the value of π using this method. Use $N = 100, 1000, 10000$

4. Create a function

```
def rwalk1d(N):
```

that returns the final position of a random walker after N iterations. After each position, the new step is chosen using a fair-coin.

3 Solutions

1. A random number is generated when this line is executed
2. The same random number is generated. `seed(N)` can be used then to generate the same sequence of random numbers everytime the program is executed.
3.
```
import numpy as np
import matplotlib as mplt
import matplotlib.pyplot as plt

# sets the seed
# this way I get the same result everytime I execute this code
np.random.seed(0)

# define constants
L = 3.
R = 1.
N = 10000

# generate the points
n = 0
for i in range(N):

    x = L * np.random.random() - 0.5 * L
    y = L * np.random.random() - 0.5 * L

    if x**2 + y**2 < R**2:
        n = n + 1

pi1 = (n / float(N)) * (L / R)**2

print 'Estimated value of pi = %f with %d points: %f' % (np.pi, N, pi1)


>>> Estimated value of pi = 3.141593 with 10000 points: 3.065400
```
4.
```
# returns the result of the random walk when the result of flipping the coin
# is heads with probability p and tails with probability 1-p
def rwalk1d(N, p):

    s = 0
    for i in range(N):

        r = np.random.random()
        if r < p: # heads
            s = s + 1
        else:     # tails
            s = s - 1

    return s
```