

## Scientific Programming in Python

### Solution

22 September 2009

#### Coin on grid

We will compare the experimental probability to the theoretical probability,  $(1 - d)^2$ . To compute the experimental probability we need to determine

1. the random event,
2. the condition for success,
3. which events to count.

**The random event** According to the problem statement, a coin is placed randomly on a  $N \times N$  grid. We will simulate this by finding a uniformly distributed point in the 2-dimensional interval  $[0, N] \times [0, N]$  representing the location of the center of the coin.

```
x = random.uniform(0, N)
y = random.uniform(0, N)
```

**The condition for success** To win the game the coin should *not* touch a grid line. The grid lines are defined to be at integer values of  $x$  and  $y$ . This means that the center of the coin has to be at least one radius,  $d/2$ , away from

- the greatest integer less than  $x$  (and  $y$ ), and
- the smallest integer greater than  $x$  (and  $y$ ).

In Python we can write this condition as

```
(x-d/2 > int(x)) and (x+d/2 < int(x)+1) and
(y-d/2 > int(y)) and (y+d/2 < int(y)+1)
```

**Which events to count** We will count the total number of successes—i.e. the number of times that the condition above is true—and the total number of experiments. The ratio between these two numbers should converge to the theoretical probability.

```

from __future__ import division
from scipy import *

N = 7 # The grid size
d = 0.25 # The diameter of the coin
total_experiments = 100000

experiment_count = 0
success_count = 0
for i in xrange(total_experiments):
    # Draw uniform random location
    x = random.uniform(0, N)
    y = random.uniform(0, N)
    # Test the condition for success
    if (x-d/2 > int(x)) and (x+d/2 < int(x)+1) and\
        (y-d/2 > int(y)) and (y+d/2 < int(y)+1):
        success_count += 1
    experiment_count += 1

print "Theoretical result:", (1-d)**2
print "Empirical result: ", success_count / experiment_count

```

## Coloured cards

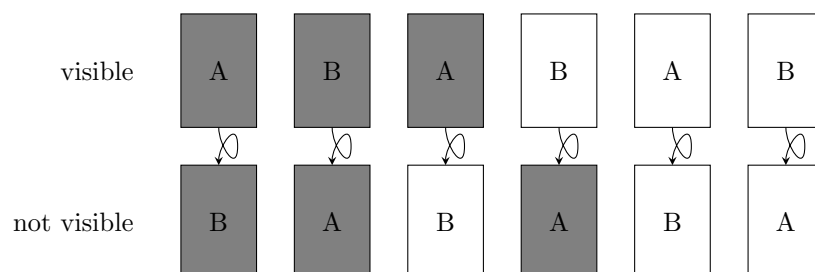
We have three different cards, which we can represent by 3 different lists. Since we are interested in the colours of the cards and will use

```
cards = [["black","white"], ["black","black"], ["white","white"]]
```

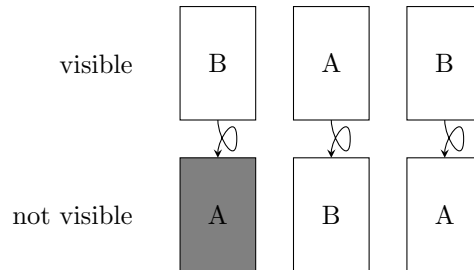
We need to draw one of the three cards uniformly at random and choose one of its sides randomly. We want to know the probability that the other

and then find the probability and then look at one of its sides randomly. If we see that this side is white, what is the probability that the other side is also white?

**Theory** There are 6 possibilities when we choose a card and one of its sides randomly. (The letters A and B are used to distinguish between the two different sides of the cards.)



We are interested only in the cases where the visible side is white.



In two of these cases the side that is not visible will also be white, making the probability  $\frac{2}{3}$ .

**The random event** We need to draw one of the three cards randomly and select one of its sides randomly.

```
card_index = random.randint(3)
side_index = random.randint(2)
```

**The condition for success** We have a positive result if the visible side is white *and* the invisible side is white.

```
(cards[card_index][side_index] == 'white') and
(cards[card_index][1-side_index] == 'white')
```

**Which events to count** We are interested only in those experiments where the first side that we observe is white. This is different from the previous problem in that we do not count all experiments, but only those where the visible side is white.

```
from __future__ import division
from scipy import *

cards = [["black","white"], ["black","black"], ["white","white"]]
total_experiments = 100000

experiment_count = 0
success_count = 0
for i in xrange(total_experiments):
    card_index = random.randint(3) # Select random card
    side_index = random.randint(2) # Select random side
    # Test if the visible side is white
    if cards[card_index][side_index] == 'white':
        # Test if the invisible side is also white
        if cards[card_index][1-side_index] == 'white':
            success_count += 1
            experiment_count += 1

print "Theoretical result: 2/3"
print "Empirical result: ", success_count / experiment_count
```