Programming: Practical 5

We wish to determine the properties on which a player is most likely to land during a game of monopoly. To simplify things, we assume there is only a single player, ignore everything to do with money and also ignore the 'Get out of Jail Free Cards'.

Monopoly: overview of the problem

The algorithm we will use is:

1) Begin the game on GO;
2) `current := current` + dice roll
3) Make a note of the new position.

- If we land on "Chance" or "Community Chest", draw a card;
- If we land on "Go To Jail", move to Jail;
- If we move, make a note of the new position;

4) Go back to step 2

After rolling the dice 100,000 times or so, stop.

Dice rolling

When we roll a single die, each side has an equal probability of occurring. This means we can use the `randint()` function from the **random** library to simulate a die roll:

```
from random import randint
randint(1, 6)
```

```
## 4
```

To roll two dice, we simply call this function

```
def RollTwoDice():
    total = randint(1, 6) + randint(1, 6)
    return total
```

The Monopoly board

In monopoly there are 40 properties or squares, see table 1 at the end of this practical for a complete list. The first square in Monopoly is 'Go', as Python indexes start at 0, we will set `Go` to be number 0. We can represent all forty squares as a list in Python. For example

```
# This creates a list of 40 values;
# All values are initially zero
landings = [0]*40
```

Then, when we land on a square we simply increase the associated landings entry by one. Suppose we landed on 'Old Kent Rd', we would represent this as:

```
landings[1] += 1
```

Since 'Old Kent Road' is square 2 on the board, it is index 1 in Python (see table 1).

Going round the monopoly board

Our first go at simulating Monopoly will ignore community chest, chance cards, and the 'Go To Jail' square. This means that we are simply going round the board. The code in the `SimulateMonopoly()` function, rolls the dice `no_of_rolls` time, and stores the squares that are landed on in the list landings. Notice any time we go over the max index, 39, we are going to take 40 away from the total, this simulating the squares of a monopoly game.

```
def SimulateMonopoly(no_of_rolls):
    landings = [0]*40
    #  Start at GO
    current = 0
    for i in range(0, no_of_rolls):
        current = current + RollTwoDice()
        if current > 39:
            current = current - 40
        landings[current] += 1
    return landings

no_of_rolls = 100000
```

We can then call the function using:

```
sim = SimulateMonopoly(no_of_rolls)
```

We can then plot the probabilities of landing in each square:

```
import matplotlib.pyplot as plt
import numpy as np

sim = sim/np.sum(sim)
plt.plot(sim)
plt.ylim(0, 0.1)
plt.show()
```

```
## (0, 0.1)
```

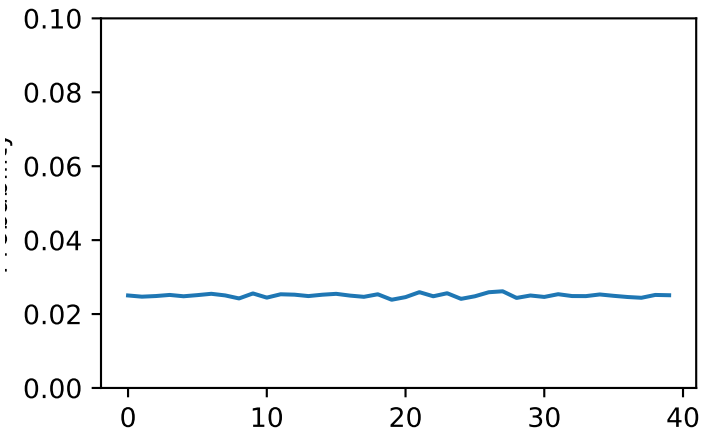

Figure 1: Probability of landing on a monopoly square.

Incorporating Community Chest Cards

There are three community chest squares on the board - squares 3, 18 and 34. In the code below we will just consider square 3. There are sixteen cards in total, hence the probability of drawing any particular card is 1/16. In the code below we will **only implement the first two community chest cards**:

```
from random import uniform

def CommunityChest(current):
    goto = current
    u = uniform(0, 1)
    if u < 1/16:
        goto = 0  # Move to Go
    elif u < 2/16:
        goto = 10  # Go to Jail :(
    return goto
```

This function takes in the current position, with probability 1/16 we 'Move to Go', with probability 1/16 we 'Go to Jail' and with probability 14/16 we stay in our current position. We now alter the simulate function to incorporate the `CommunityChest()` function:

```
def SimulateMonopoly(no_of_rolls):
    landings = [0]*40
    #  Start at GO
    current = 0
    for i in range(0, no_of_rolls):
        current = current + RollTwoDice()
        if current > 39:
            current = current - 40
        landings[current] += 1
        if current == 3:
            cc_move = CommunityChest(current)
            if cc_move != current:
                current = cc_move
                landings[current] += 1
    return landings
no_of_rolls = 1000000
```

We can then call this function

```
sim2 = SimulateMonopoly(no_of_rolls)
```

We then plot the results

```
sim2 = sim2/np.sum(sim2)
plt.plot(sim2)
plt.ylim(0, 0.06)
plt.show()
```

```
## (0, 0.1)
```

Additional questions

Each question adds an additional layer of complexity to your code.

1) Add in the two other community squares, i.e. squares 18 and 34 into the `SimulateMonopoly()` code.
2) Add in 'Go to Old Kent Road' into your `CommunityChest()` function.

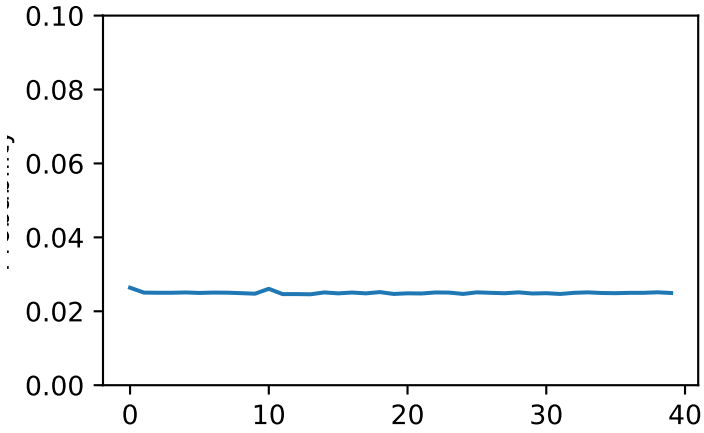

Figure 2: Probability of landing on monopoly square with the first community chest card implemented. Incorporating a single cummunity chest card has very little effect. For this graphic, I used 2 million simulations!

3) Square 31 is 'Go To Jail.' Implement this in your main simulation function.
4) Create a `Chance()` function, that implements the first six Chance cards. When you land on a Chance square, call this function.
5) Add in Community Chest card four.
6) Add in Chance card 8.
7) Add in Chance card 7, 'Go back 3 spaces'.
8) Rolling a double (a pair of 1's, 2's, ..., 6's) is special:
    1) Roll two dice (`total1`): `total_score = total1`
    2) If you get a double, roll again (`total2`) and `total_score = total1 + total2`
    3) If you get a double, roll again (`total3`) and `total_score = total1 + total2 + total3`
    4) If roll three is a double, Go To Jail, otherwise move `total_score`

Additional Information

Community Chest Cards

There are three community chest areas on the board (see Table 1). In total, there are 16 community chest cards.

1) Advance to Go;
2) Go to jail;
3) Go to Old Kent Road;
4) Take a Chance card instead;

Chance Cards

A Chance card is most likely to move players. There are three chance areas on the board (see Table 1). There are 16 chance cards in total, of which eight cards move the player:

1) Advance to Go;
2) Advance to Trafalgar Square;
3) Advance to Pall Mall;
4) Go directly to Jail;
5) Take a trip to Marylebone Station;
6) Advance to Mayfair;
7) Go back 3 spaces;
8) Advance token to nearest Utility. The utility squares are the water works and the electric company.

Square Number	Name	Square Number	Name
1	Go	11	Jail
2	Old Kent Road	12	Pall Mall
3	Community Chest	13	Electric Company
4	WhiteChapel Road	14	Whitehall
5	Income Tax	15	Northumberland Avenue
6	King's Cross Station	16	Marylebone Station
7	The Angel Islington	17	Bow Street
8	Chance	18	Community Chest
9	Euston Road	19	Marlborough Street
10	Pentonville Road	20	Vine Street
21	Free Parking	31	Go To Jail
22	Strand	32	Regent Street
23	Chance	33	Oxford Street
24	Fleet Street	34	Community Chest
25	Trafalgar Square	35	Bond Street
26	Fenchurch Street Station	36	Liverpool St Station
27	Leicester Square	37	Chance
28	Coventry St	38	Park Lane
29	Water Works	39	Super Tax
30	Piccadilly	40	Mayfair

Table 1: Monopoly squares with associated square numbers