Lab 11: Simulation and software design, randrange function, and unit testing

This lab expects you to know about simulation and top-down design, randrange function, and unit testing. All tasks of this lab are the continuation of prelab 11. Hence, we strongly recommend you to complete prelab 11 first, and then start working on lab 11.

Please complete prelab 11 before starting the lab.

Prelab 11

In this lab, you are expected to complete three tasks. In task 1, you will write the function decideServingFirst, which decides who will serve first for each game. In task 2, you will write two functions checkAndHandleDeuce, and checkAndDoServeChange which handle deuce condition and change of serve respectively. Finally, you will write three unit tests for the functions mentioned above.

Prelab 11

We encourage you to work either individually or in groups on the Prelab. The tasks in the prelab 11 will not be graded, but it will help you to be better prepared for your lab session. Prelab 11 contains the following:

1. How to create a random number between 0 (inclusive) and 1 (exclusive) using random function.
2. How to create a random number between a (inclusive) and b (exclusive) using randrange(a, b) function where a and b are two integers
3. Simulation of racquetball game
4. How to write unit test

If you have any questions on the material of the prelab, first check the book and recitation slides. If you continue to have any doubt about it, attend to the assigned office hours.

Prelab 11

Task 1: Use of "randrange" function

If you carefully look at the simOneGame function of prelab 11, you will find that player A always serves first when a game starts. However, we want to make this fair between player A and B. So, we want to flip a coin and let them decide who will serve first for a game. If flipping a coin results in HEAD, player A serves, otherwise player B. In order to do this task, we change the simOneGame
function as follows:

def simOneGame(probA, probB):
    # initially, scores are zero
    scoreA = 0
    scoreB = 0

    # Task 1: Instead of always serving A first, flip a coin and decide who
    # will serve first for each game
    serving = decideServingFirst()
    # TODO: YOU NEED TO COMPLETE decideServingFirst function

    #The game will continue until any player scores 15 first
    while scoreA != 15 and scoreB != 15:
        # simulate one serve,
        # get the winner of that serve,
        # update the score or change their server
        if serving == 'A':
            if random() < probA:
                # A wins the serve, you got the winner
                scoreA = scoreA + 1
            # update the score
        else:
            serving = 'B'
            # change the server
        else:
            if random() < probB:
                # B wins the serve, you got the winner
                scoreB = scoreB + 1
            # update the score
        else:
            serving = 'A'
            # change the server

    return scoreA, scoreB

Now you need to complete the decideServingFirst function which is called from simOneGame function. Since decideServingFirst is called from simOneGame function, the top-down design of this simulation problem looks like below:
# Task 1: Complete the function decideServingFirst
# Flip a coin, if it is HEAD, A will serve, otherwise B
# @no parameters
# return a string. The two possible values of the returned string are 'A' or 'B'
def decideServingFirst():
    # To simulate flipping a coin, we can choose a random number between 0 (inclusive) and 1 (inclusive).
    # If 1 is chosen, it indicates HEAD, player 'A' will serve first
    # if 0 is chosen, it indicates TAIL, player 'B' will serve first
    # return servingFirst

You can download the task 1 skeleton from here.

**Task 2: Simulation and software design of racquetball**

For this task, we change the racquetball rules slightly. The revised rules are as follows:

- For each game, we flip a coin and decide who will serve first. (Task 1)
- For each game the gamePoint is set to 15 initially.
- Each player has two serves each in a row (this is similar to table tennis/ping pong game). Serve continues to alternate between opponents until one player scores the gamePoint, i.e., 15.
- A game shall be won by a player first scoring the gamePoint, i.e., 15 points unless both players or pairs score 14 points.
- If both players score 14 points, it is called deuce condition. In this case, the game continues until a player gains two points lead. For example, the game shall be won by a player first scoring 16 points unless both players or pairs score 15 points. Every time both players reach deuce condition, the gamePoint is reset to current gamePoint plus 1.
- Each serve results in a point either for player A or player B.
- If A is serving and if A's probability is greater than random(), A wins the serve (rally) and A's score gets increased by 1, otherwise B's score gets increased by 1.
- If B is serving and if B's probability is greater than random(), B wins the serve (rally) and B's score gets increased by 1, otherwise B's score gets increased by 1.

In order to make these changes into effect, we change the simOneGame function of prelab 11 as follows:

def simOneGame(probA, probB):
    #set the gamePoint to 11
    gamePoint = 11
    # start the game
    # initially, scores for both players are 0
    scoreA = 0
    scoreB = 0
    #initially, serveCounts for both players are 0
    serveCountA = 0 # total number of times player A serves
    serveCountB = 0 # total number of times player B serves
# Task 1: At the very beginning of the game players need to decide who will serve first.

```python
serving = decideServingFirst()
```

# Task 2:
# The game will continue until any player scores the gamePoint first

```python
while scoreA != gamePoint and scoreB != gamePoint:
    # Check deuce condition and handle it. Update gamePoint if required
    gamePoint = checkAndHandleDeuce(scoreA, scoreB, gamePoint)
    # TODO: YOU NEED TO COMPLETE checkAndHandleDeuce function

    # increment the serveCount of the corresponding player
    if serving == 'A':
        serveCountA = serveCountA + 1
    else:
        serveCountB = serveCountB + 1

    # simulate one serve and get the winner of that serve
    if serving == 'A':
        if random() < probA:  # A wins the serve
            winner = 'A'
        else:
            winner = 'B'  # otherwise, B wins the serve
    else:
        if random() < probB:  # B wins the serve
            winner = 'A'
        else:
            winner = 'B'  # Otherwise, A wins the serve

    # update the score of the winner
    if (winner == 'A'):
        scoreA = scoreA + 1
    else:
        scoreB = scoreB + 1

    # Check if serve change is required.
    That means, if player A were serving two consecutive times, player B will start serving.
    serving = checkAndDoServeChange(serveCountA, serveCountB, serving)
    # TODO: YOU NEED TO COMPLETE checkAndDoServeChange function
```

return scoreA, scoreB
checkAndHandleDeuce function: Deuce condition occurs when scores of two players are equal and just 1 point less than the current gamePoint. If deuce condition occurs, gamePoint needs to be incremented by 1.

```python
# @parameters: scoreA, scoreB, gamePoint
# scoreA: score of player A, type: integer
# scoreB: score of player B, type: integer
# gamePoint: current GamePoint, type: integer
# return updated gamePoint (integer datatype) based on deuce Condition
def checkAndHandleDeuce(scoreA, scoreB, gamePoint):
    # complete the code
    # return the gamePoint
```

checkAndDoServeChange function: Check if serve change is required. That means, if player A were serving two consecutive times, player B would start serving.

```python
# @parameters: serveCountA, serveCountB, serving
# serveCountA: total number of times player A serves, type: integer
# serveCountB: total number of times player B serves, type: integer
# serving: who just served, type: string. Possible values could be: 'A' or 'B'
# return servingNext (string datatype), Possible values are 'A' or 'B'
def checkAndDoServeChange(serveCountA, serveCountB, serving):
    # complete the code
    # HINT: You can use mod (%) operation to determine who should serve next
    # return servingNext
```

You can download the task 2 skeleton from here.

**Task 3: Unit Test**

Write unit tests for decideServingFirst, checkAndHandleDeuce and
checkAndDoServeChange functions.

- Create a python file called testCases.py
- import required functions of task2.py for which you want to write unit test
- write test cases for the imported function

**Turn in Instructions**

Please follow [this short PDF tutorial](http://courses.cs.purdue.edu/) on how to turn in your files. Talk to your lab TA if you have any questions.

**Grading Criteria**

<table>
<thead>
<tr>
<th>Task</th>
<th>%</th>
<th>Breakdown</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 1</td>
<td>30</td>
<td>For the function decideServingFirst</td>
</tr>
<tr>
<td>Task 2</td>
<td>40</td>
<td>20% for the function checkAndHandleDeuce, and 20% for function checkAndDoServiceChange</td>
</tr>
<tr>
<td>Task 3</td>
<td>30</td>
<td>10% for each of the three unit tests</td>
</tr>
</tbody>
</table>

From: [http://courses.cs.purdue.edu/](http://courses.cs.purdue.edu/) - Computer Science Courses


Last update: 2016/03/23 12:03