Polytechnic University of Puerto Rico

Electrical & Computer Engineering

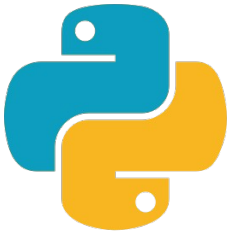
and Computer Sciences Department

CECS 3210

Advanced Programming with Python

Prof: Edwin Flórez

Project #1



Jonathan Valdesuso García

Student ID: 82052

Scope

The objective of the program volleyball.py is to simulate volleyball games.
Volleyball is a sport played between two teams that play in a contained area separated by a net (each team on different sides of the net). The game is played with a ball. A team starts the game serving the ball; if the serving team wins the rally they make a point and serve again; if the serving team loses the rally the opponent team will serve the ball the next turn. This process is repeated until a team reaches at least 15 points and is ahead by 2 points in comparison with the opponent team.

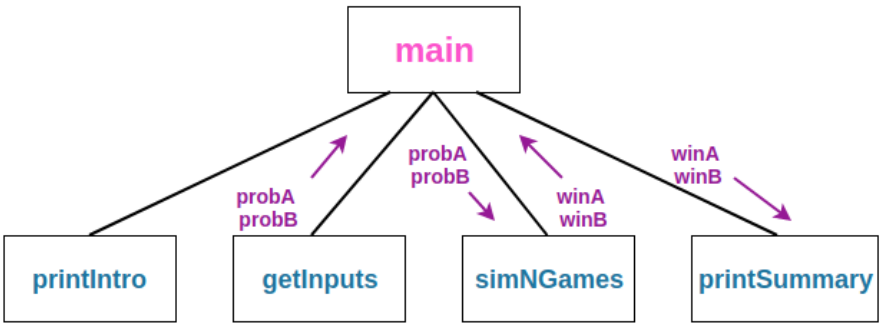
The simulation creates a model of a certain number of games between teams with different skill levels.
Th program will prompt the user for:

- The probabilities (number from 0 to 1) of winning points on serving of two different teams.
- The number of games to be simulated.

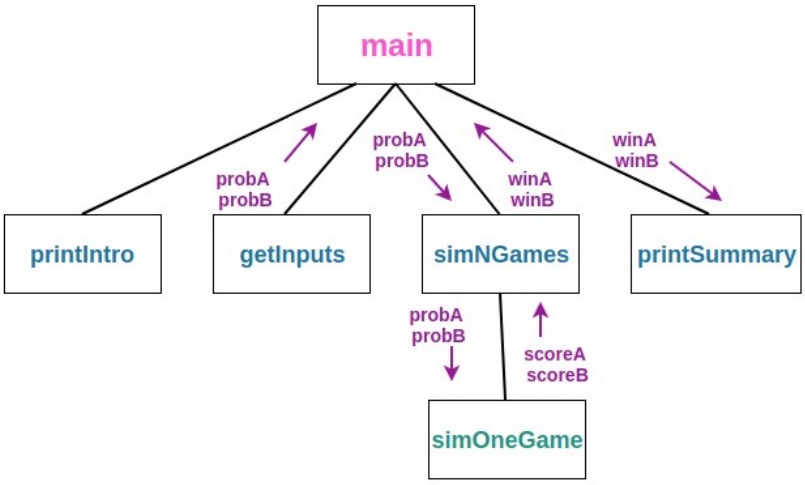
The program prints the result of the number of games simulated with the total percentage of wins of each team ('A' and 'B').

Structure Charts

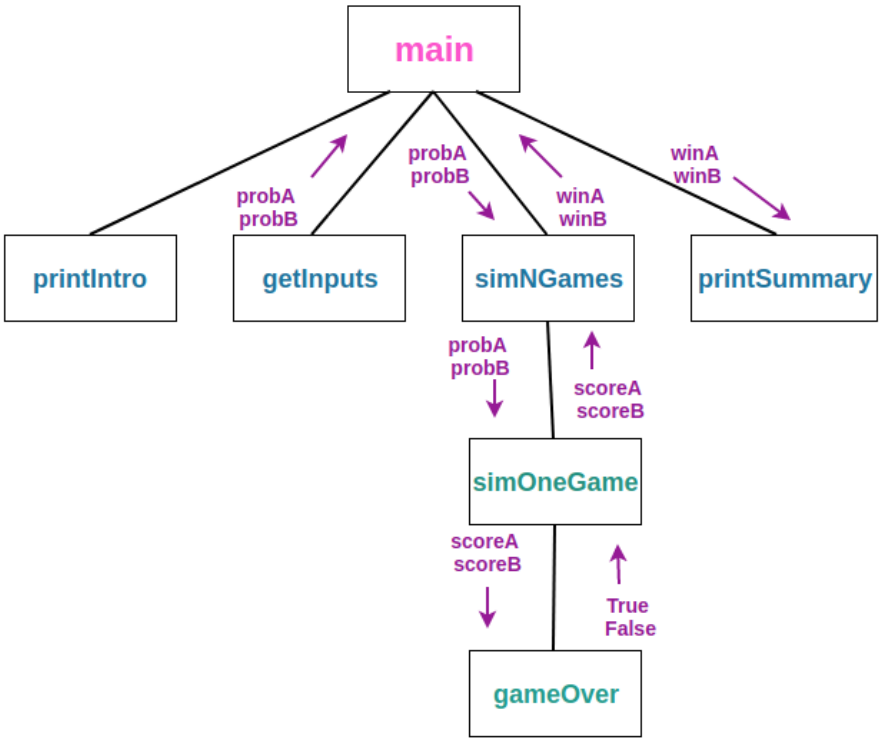
Top Level Design



Second Level Design



Third Level Design



Code Details

Top Level Design

main()

PseudoCode:

- Print an Introduction
- Get the inputs: probA, probB, n
- Simulate n games of volleyball using probA and probB
- Print a report on the wins for teamA and teamB

Code:

```
def main():
    printIntro()
    probA, probB, n = getInputs()
    winsA, winsB = simNGames(n, probA, probB)
    printSummary(winsA, winsB)
```

Second Level Design

printIntro()

This function prints the basic information of the program to the user.

Code:

```
def printIntro():
    print("This program simulates a game of volleyball between two")
    print('teams callded "A" and "B". The ability of each team is')
    print("indicated by a probability (a number between 0 and 1) that")
    print("has the first serve.")
```

getInputs()

This function prompts the user for the probabilities of team A, the probabilities of team B, and the number of times to run the simulation of the volleyball games and return these values (probA, probB, n).

Code:

```
def getInputs():
    # Returns the tree simulation parameters
    a = float(input("What is the prob. team A wins a serve? "))
    b = float(input("What is the prob. team B wins a serve? "))
    n = int(input("How many games to simulate? "))
    return a, b, n
```

simNGames(n, probA, probB)

This function simulates n games of bolleyball between teams whose abilites are represented by the probability of wining a serve. This function returns the number of wins of teamA and teamB.

Pseudo Code:

- Initialize two variables to 0 that represents the number of wins of each team (winsA, winsB)
- Run the sumulation n (number given by the user) times.
  - If TeamA wins, increase winsA by 1
  - Else If TeamB wins, increase winsB by 1
- Return winsA and winsB

Code:

```
def simNGames(n, probA, probB):
        # Simulates n games of volleyball between teams whose
        # abilities are represented by the probability of winnina a serve.
        # Returns number of wins for A and B
        winsA = winsB = 0
        for i in range(n):
                scoreA, scoreB = simOneGame(probA, probB)
                If scoreA > scoreB:
                        winsA = winsA + 1
                else:
                        winsB = winsB + 1
        return winsA, winsB
```

Third Level Design

simOneGame(probA, probB)

This function will simulate a single volleyball game between two teams using probA and probB.

Pseudo Code:

- Initialize to 'A' a variable that represents the team that is currently serving (serving). This means that the simulation always starts the game with teamA serving the ball.
- Initialize two variables to 0 that represents the score of each team (scoreA, scoreB)
- Run the game until a team reaches a wining score
  - If team A is serving
    - If the random number from 0 to 1 is less than the probability of teamA winning a rally on service
      - Increase scoreA by 1
  - Else if team B is serving
    - If the random number from 0 to 1 is less than the probability of teamB winning a rally on service
      - Increase scoreB by 1
- Return scoreA and scoreB

Code:

```
def simOneGame(probA, probB):
        # Simulates a single game of volleyball between teams whose
        # abilities are represented by the probability of winning a serve.
        # Returns final scores for A and B
        serving = 'A'
        scoreA = scoreB = 0
        while not gameOver(scoreA, scoreB):
                if serving == "A":
                        if random() < probA:
                                scoreA = scoreA + 1
                        else:
                                serving = "B"
                else:
                        if random() < probB:
                                scoreB = scoreB + 1
                        else:
                                serving = "A"
        return scoreA, scoreB
```

GameOver(scoreA, scoreB)
This function will simulate a single volleyball game between two teams using probA and probB.

Pseudo Code:

- If a team reaches a score thats above 15 and is ahead by 2 points in comparison with the other teams score return True, Else return False

Code:

```
def gameOver(a, b):
        # a and b represent scores for a volleyball game
        # Returns True if the game is over, False otherwise.
        return (a >= 15 and 2 <= a-b) or (b >= 15 and 2 <= b-a)
```

Source Code

```
# Design and implement a simulation of the game of volleyball. Normal
```

```
# volleyball is played like racquetball in that a team can only score points
# when it is serving. Games are played to 15, but must be won by at least
# two points.

# volleyball.py
from random import random

def main():
        printIntro()
        probA, probB, n = getInputs()
        winsA, winsB = simNGames(n, probA, probB)
        printSummary(winsA, winsB)

def printIntro():
        print("This program simulates a game of volleyball between two")
        print('teams callded "A" and "B". The ability of each team is')
        print("indicated by a probability (a number between 0 and 1) that")
        print("has the first serve.")

def getInputs():
        # Returns the tree simulation parameters
        a = float(input("What is the prob. team A wins a serve? "))
        b = float(input("What is the prob. team B wins a serve? "))
        n = int(input("How many games to simulate? "))
        return a, b, n

def simNGames(n, probA, probB):
        # Simulates n games of volleyball between teams whose
        # abilities are represented by the probability of winnina a serve.
        # Returns number of wins for A and B
        winsA = winsB = 0
        for i in range(n):
                scoreA, scoreB = simOneGame(probA, probB)
                if scoreA > scoreB:
                        winsA = winsA + 1
                else:
                        winsB = winsB + 1
        return winsA, winsB

def simOneGame(probA, probB):
        # Simulates a single game of volleyball between teams whose
        # abilities are represented by the probability of winning a serve.
        # Returns final scores for A and B
        serving = 'A'
        scoreA = 0
        scoreB = 0
        while not gameOver(scoreA, scoreB):
        if serving == "A":
                if random() < probA:
```

```
                scoreA = scoreA + 1
            else:
                serving = "B"
        else:
            if random() < probB:
                scoreB = scoreB + 1
            else:
                serving = "A"
        return scoreA, scoreB

def gameOver(a, b):
        # a and b represent scores for a volleyball game
        # Returns True if the game is over, False otherwise.
        return (a >= 15 and 2 <= a-b) or (b >= 15 and 2 <= b-a)

def printSummary(winsA, winsB):
        # Prints a summary of wins for each player.
        n = winsA + winsB
        print("\nGames simulated:", n)
        print("Wins for A: {0} ({1:0.1%})".format(winsA, winsA/n))
        print("Wins for B: {0} ({1:0.1%})".format(winsB, winsB/n))

if __name__ == '__main__': main()
```