# Dice Game

**Using Random and external files to create a simulated Dice Game!**

Brayan Quevedo Ramos

Porterville College

Introduction To Programming: P120, 30214

Dice Game

Date: 4, April 2022

## Contents

**1 Introduction** **3**

**2 Assignment** **3**

2.1 Requirements . . . . . 3

**3 Discussion** **4**

3.1 Solution . . . . . 4

3.2 Flowchart . . . . . 5

3.3 Implementation Issues . . . . . 6

3.4 Known Bugs/Errors . . . . . 7

3.5 Edge Case . . . . . 7

**4 Lessons Learned** **8**

4.1 Lessons Learned . . . . . 8

4.2 Restarting The Project . . . . . 8

4.3 Revision of Exercise . . . . . 8

**5 Results** **9**

5.1 Questions? Answers. . . . . . 9

5.2 Program Running . . . . . 11

## 1 Introduction

Dice game is a user vs computer game where both systems roll... well dice. The user rolls dice. Then, they can re-roll any dice they don't like the numbers of, to have dice of the highest number. The computer shall do the same, and determine who the winner is!

## 2 Assignment

### 2.1 Requirements

Listed below are the indicated requirements for the program to contain:

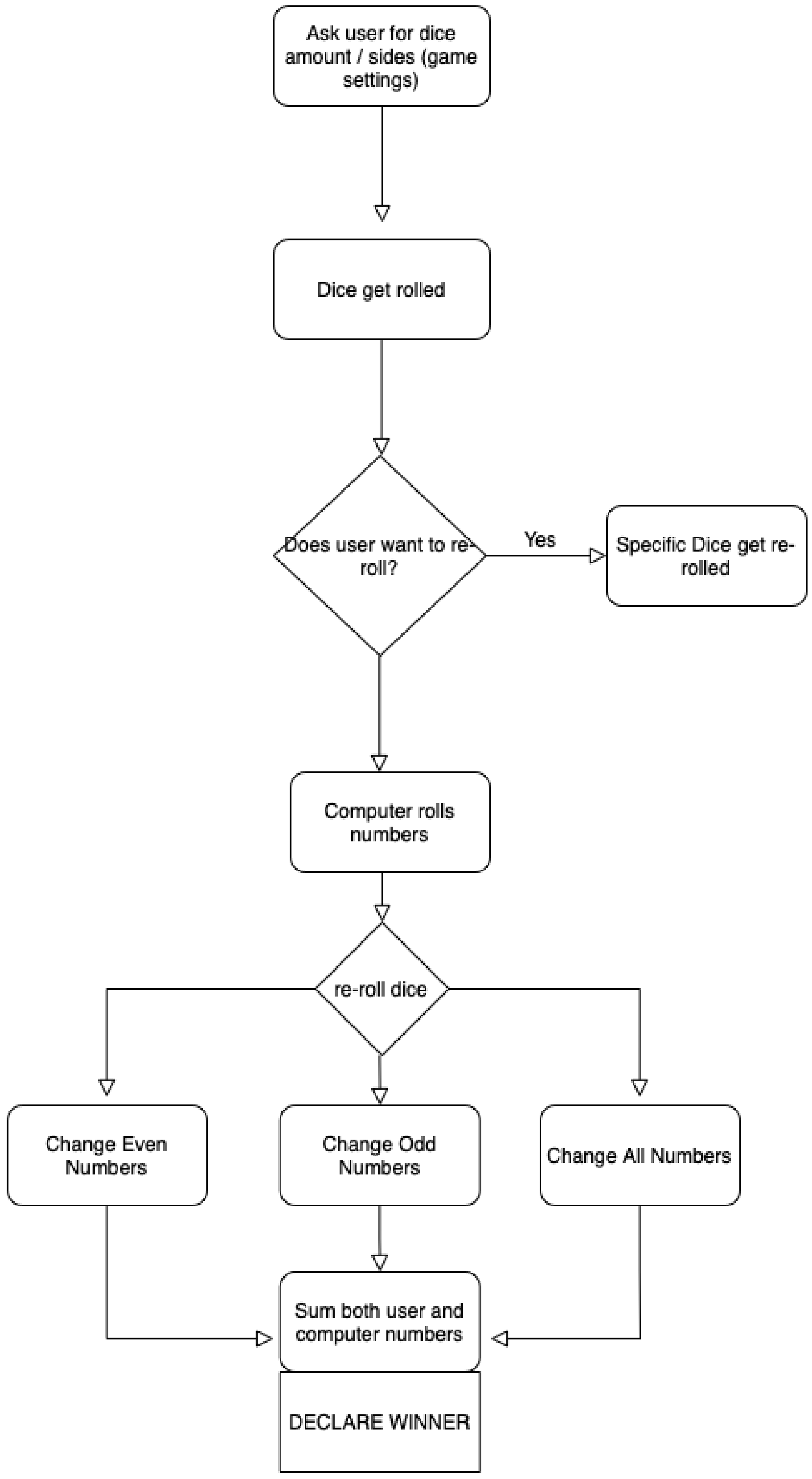
- Create two functions that determine how the computer rolls for your dice game (Write these in a separate script file).
- Discuss if you think the algorithms you have created are a disadvantage for the user or computer.

## 3 Discussion

### 3.1 Solution

The manner of which these requirements were satisfied were quite easy. The basic outline of the game goes according to the following: The user starts the program by defining the game rules. These include how many dice both sides will roll and how many dice faces each dice have. Once the initial rolling is done, the user can choose what dice to re-roll (if any). Finally, the program shall sum all their dice. Next, the computer will roll (and re-roll if applicable), and sum their total. To end, the program will decide the winner, and display the result.

### 3.2 Flowchart


Ask user for dice amount / sides (game settings)
Dice get rolled
Does user want to re-roll?
Yes
Specific Dice get re-rolled
Computer rolls numbers
re-roll dice
Change Even Numbers
Change Odd Numbers
Change All Numbers
Sum both user and computer numbers
DECLARE WINNER

## 3.3 Implementation Issues

While the project was straightforward, there was one (surprisingly difficult) implementation issue presented – that being the verification stage of what dice the user wishes to re-roll. Presented below is the following code where this issue was:

```
reRollKeys = input("[-]For Change, [+] for keep.")

# Ask user to see which numbers they want to re-roll
while True:
    # Check to see if every index in userRoll[] has a response
    if (len(reRollKeys) != len(userRoll)):
        print("Sorry! You made a bad input, let's try again (Maybe you forgot ab
        reRollKeys = input("[-]For Change, [+] for keep: ")
        print("")
        continue

    # CHeck to see if every response has a valid input
    for i in range(0, (len(reRollKeys))):
        if (reRollKeys[i] == "-") or (reRollKeys[i] == "+"):
            print("Evaluating.....")

        else:
            print("Sorry! You made a bad input (maybe you pressed a wrong button
            reRollKeys = input("[-]For Change, [+] for keep: ")
            print("")
        break

    # Everything is valid! Continue with program
    break

# For each index, if user chose to change number, change it!
for i in range(0, len(userRoll)):
    if (reRollKeys[i] == "-"):
        userRoll[i] = random.randint(0, diceSide)
```

One big programming hack is the essence of infinite loops for it's immense convenience. Users are quite unpredictable, and the goal was for them to have true input (with edge cases), so the loop had to validate different parts of the string. Without the While True statement (which was the reason why implementation was difficult), it was impossible to tell whether all the checks have passed. Technically, there could've been a function with local variables to verify each component, and then use variables as an argument under a giant if statement, but that's so inefficient to where it's considered improper programming practice.

### 3.4 Known Bugs/Errors

From current testing, there appears to be no ways to break the program.

### 3.5 Edge Case

This program was designed with edge-cases in mind. As mentioned above, there are different tests to measure what could possibly be wrong with the input. From incorrect alignment in indexing, to having invalid input, every edge case is accounted for.

## 4 Lessons Learned

To be honest, this project didn't really push me unlike the others, but I did use this opportunity to write some fairly simple and clean code!

### 4.1 Lessons Learned

There wasn't very much learned in this project. If I had to say something, gaining more experience working with different files is always good, especially when it comes to scaling code for a bigger project.

### 4.2 Restarting The Project

I don't think I would do much different if I had to restart the exercise. If I must say something, it would be to make this program more OOP friendly. As in, create an object call "Entity" with the properties of who's playing, what dice are on the table, the sides of the dice, and who's currently winning. In essence, make this game multiplayer. While not difficult to do, even with the current code, time was the culprit here.

### 4.3 Revision of Exercise

While I loved this exercise, I wish instead of having different slides with step instructions, to give the requirements initially in a provided document, and have side documents with step-by-step instructions, so students with more programming experience can get started, and people who need help get an appropriate amount of help so that they're still independent, yet don't feel stranded nor alone.

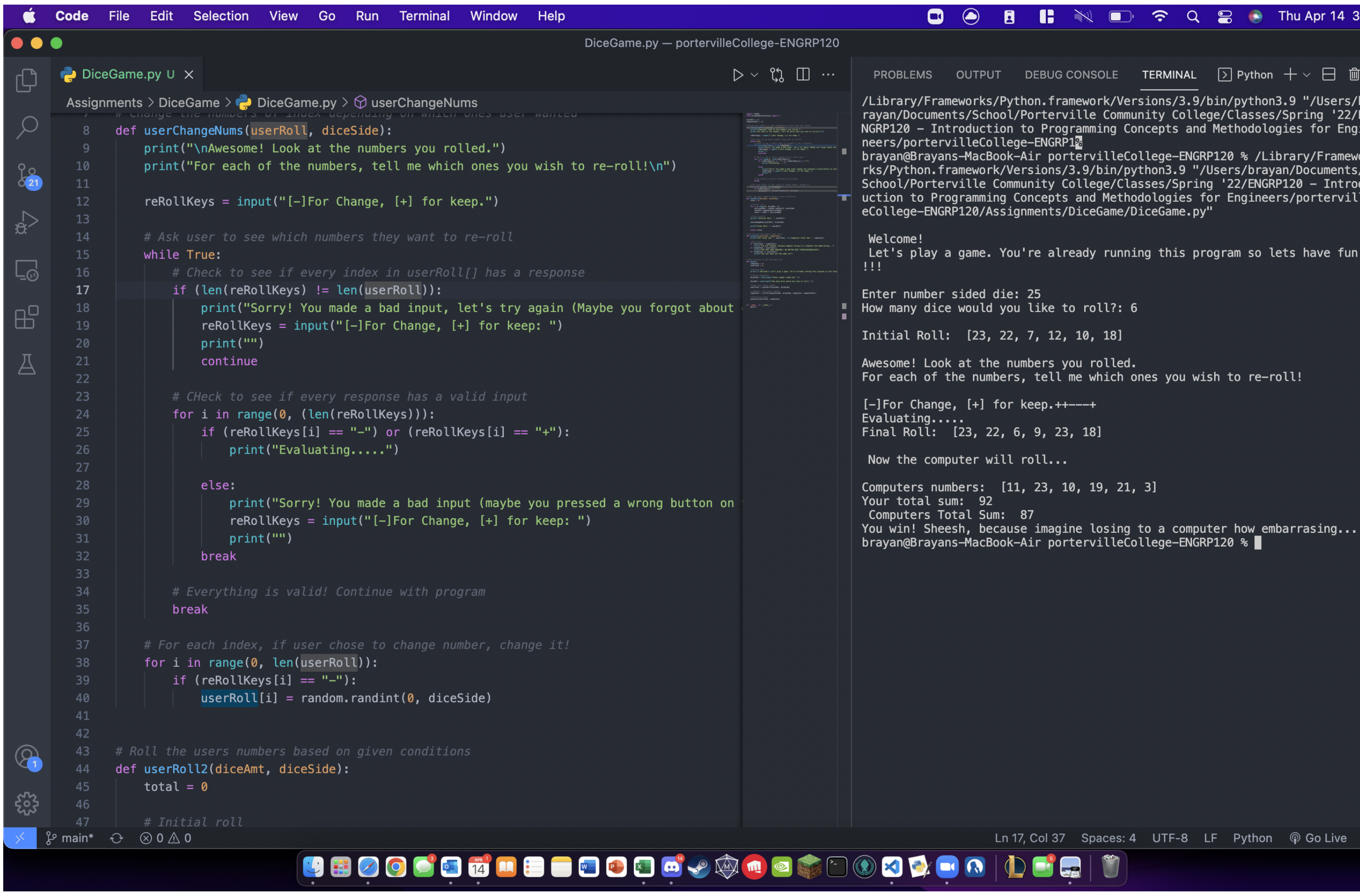
## 5 Results

### 5.1 Questions? Answers.

Since I don't know (yet tutor) statistics and probability, I'm not sure who has the edge (house or player), as the rolls don't change based on value, only category of integer. See below for reference:

```
    import random

def compChangeEvenNums(diceSide, computerRoll):
    for i in range(0, len(computerRoll)):
        if ((i % 2) == 0):
            computerRoll[i] = random.randint(0, diceSide)

    return computerRoll

def compChangeOddNums(diceSide, computerRoll):
    for i in range(0, len(computerRoll)):
        if ((i % 2) == 1):
            computerRoll[i] = random.randint(0, diceSide)



def compRandCompleteNums(diceSide, computerRoll):
    for i in range(0, len(computerRoll)):
        computerRoll[i] = random.randint(0, diceSide)

    return computerRoll

def addAllNums(computerRoll):
    compTotal = 0
    for i in range(0, len(computerRoll)):
        compTotal = compTotal + int(computerRoll[i])
    return compTotal

def changeRandomCompNumbers(diceSide,computerRoll):
    rand = random.randint(0,2)

    if rand == 0:
        compChangeEvenNums(diceSide, computerRoll)

    if rand == 1:
        compChangeOddNums(diceSide, computerRoll)
    if rand == 2:
        compRandCompleteNums(diceSide, computerRoll)

    sumTotal = addAllNums(computerRoll)

    return sumTotal

# Roll computers numbers based on given conditions
def rollForComp(diceAmt, diceSide, compTotal, computerRoll):

    print("\n Now the computer will roll...\n")

    compTotal = 0
    # Computer uses same boundaries and rolls to create sum
    for i in range(0, diceAmt, 1):
        computerRollNum = random.randint(0, diceSide)
        computerRoll.append(computerRollNum)

    compTotal = changeRandomCompNumbers(diceSide, computerRoll)

    print("Computers numbers: ", computerRoll)
    return compTotal
```

## 5.2 Program Running



```python
def userChangeNums(userRoll, diceSide):
    print("\nAwesome! Look at the numbers you rolled.")
    print("For each of the numbers, tell me which ones you wish to re-roll!\n")

    reRollKeys = input("[-]For Change, [+] for keep.")

    # Ask user to see which numbers they want to re-roll
    while True:
        # Check to see if every index in userRoll[] has a response
        if (len(reRollKeys) != len(userRoll)):
            print("Sorry! You made a bad input, let's try again (Maybe you forgot about
            reRollKeys = input("[-]For Change, [+] for keep: ")
            print("")
            continue

        # CHeck to see if every response has a valid input
        for i in range(0, (len(reRollKeys))):
            if (reRollKeys[i] == "-") or (reRollKeys[i] == "+"):
                print("Evaluating.....")

            else:
                print("Sorry! You made a bad input (maybe you pressed a wrong button on
                reRollKeys = input("[-]For Change, [+] for keep: ")
                print("")
            break

        # Everything is valid! Continue with program
        break

    # For each index, if user chose to change number, change it!
    for i in range(0, len(userRoll)):
        if (reRollKeys[i] == "-"):
            userRoll[i] = random.randint(0, diceSide)


# Roll the users numbers based on given conditions
def userRoll2(diceAmt, diceSide):
    total = 0

    # Initial roll
```

```
/Library/Frameworks/Python.framework/Versions/3.9/bin/python3.9 "/Users/
rayan/Documents/School/Porterville Community College/Classes/Spring '22/
NGRP120 - Introduction to Programming Concepts and Methodologies for Eng
neers/portervilleCollege-ENGRP1%
brayan@Brayans-MacBook-Air portervilleCollege-ENGRP120 % /Library/Framew
rks/Python.framework/Versions/3.9/bin/python3.9 "/Users/brayan/Documents
School/Porterville Community College/Classes/Spring '22/ENGRP120 - Intro
uction to Programming Concepts and Methodologies for Engineers/portervil
eCollege-ENGRP120/Assignments/DiceGame/DiceGame.py"

 Welcome!
 Let's play a game. You're already running this program so lets have fun
!!!

Enter number sided die: 25
How many dice would you like to roll?: 6

Initial Roll:  [23, 22, 7, 12, 10, 18]

Awesome! Look at the numbers you rolled.
For each of the numbers, tell me which ones you wish to re-roll!

[-]For Change, [+] for keep.++---+
Evaluating.....
Final Roll:  [23, 22, 6, 9, 23, 18]

 Now the computer will roll...

Computers numbers:  [11, 23, 10, 19, 21, 3]
Your total sum:  92
 Computers Total Sum:  87
You win! Sheesh, because imagine losing to a computer how embarrasing...
brayan@Brayans-MacBook-Air portervilleCollege-ENGRP120 %
```