## Assignment #4

For this assignment you will be writing 3 programs which should be saved independently as their own “.py” file. The filename you should use for each program is outlined in the sections below. When you’re finished you should submit your programs to the Assignment #4 category inside of NYU Classes.

You may work by yourself or with a partner. Remember, collaborating just entails discussing approaches to the problems; you should not be writing actual code together. If you choose to work with someone else please be sure to give them credit in your source code and on NYU classes when you submit your work. Each member of a team must submit their own unique copy of the assignment to get credit for the assignment.

[Warm-Up |](https://emilydidthis.github.io/CSCI-UA.0002-Fall22/assignments/04.html#warm-up-star-drawer)[Problem 1 |](https://emilydidthis.github.io/CSCI-UA.0002-Fall22/assignments/04.html#problem-1-roll-the-dice)[Problem 2 |](https://emilydidthis.github.io/CSCI-UA.0002-Fall22/assignments/04.html#problem-2-pick-up-sticks)[Problem 3](https://emilydidthis.github.io/CSCI-UA.0002-Fall22/assignments/04.html#problem-3-decimal-to-binary)

### ****Warm-Up: Star Drawer****

**Part 1**

Write a program that prompts the user for two numbers. You can always assume the user will supply integers when prompted, but you cannot assume the range of these integers. Ensure that both integers are positive, and that the second integer is larger than the first. For example:

Number 1: -5

Invalid, try again

Number 1: 5

Number 2: -5

Invalid, try again

Number 2: 4

Invalid, try again

Number 2: 6

**Part 2**

Next, generate a triangle pattern that looks like the following. Note that the first line you should print should contain the first number the user supplied, followed by that many stars. The next line should increase by 1 and repeat the process until you reach the second number.

Number 1: 1

Number 2: 9

1 \*

2 \*\*

3 \*\*\*

4 \*\*\*\*

5 \*\*\*\*\*

6 \*\*\*\*\*\*

7 \*\*\*\*\*\*\*

8 \*\*\*\*\*\*\*\*

9 \*\*\*\*\*\*\*\*\*

**Part 3**

Generate the bottom half of the triangle so that it looks like an “arrow”. For this part of the shape the numbers should decrease and return back to the first number. Here are a few runnings of your program - user input is underlined:

Number 1: 1

Number 2: 9

1 \*

2 \*\*

3 \*\*\*

4 \*\*\*\*

5 \*\*\*\*\*

6 \*\*\*\*\*\*

7 \*\*\*\*\*\*\*

8 \*\*\*\*\*\*\*\*

9 \*\*\*\*\*\*\*\*\*

8 \*\*\*\*\*\*\*\*

7 \*\*\*\*\*\*\*

6 \*\*\*\*\*\*

5 \*\*\*\*\*

4 \*\*\*\*

3 \*\*\*

2 \*\*

1 \*

Number 1: -10

Invalid, try again

Number 1: -20

Invalid, try again

Number 1: 5

Number 2: -5

Invalid, try again

Number 2: 3

Invalid, try again

Number 2: 4

Invalid, try again

Number 2: 5

Invalid, try again

Number 2: 20

5 \*\*\*\*\*

6 \*\*\*\*\*\*

7 \*\*\*\*\*\*\*

8 \*\*\*\*\*\*\*\*

9 \*\*\*\*\*\*\*\*\*

10 \*\*\*\*\*\*\*\*\*\*

11 \*\*\*\*\*\*\*\*\*\*\*

12 \*\*\*\*\*\*\*\*\*\*\*\*

13 \*\*\*\*\*\*\*\*\*\*\*\*\*

14 \*\*\*\*\*\*\*\*\*\*\*\*\*\*

15 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

16 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

17 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

18 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

19 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

20 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

19 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

18 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

17 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

16 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

15 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

14 \*\*\*\*\*\*\*\*\*\*\*\*\*\*

13 \*\*\*\*\*\*\*\*\*\*\*\*\*

12 \*\*\*\*\*\*\*\*\*\*\*\*

11 \*\*\*\*\*\*\*\*\*\*\*

10 \*\*\*\*\*\*\*\*\*\*

9 \*\*\*\*\*\*\*\*\*

8 \*\*\*\*\*\*\*\*

7 \*\*\*\*\*\*\*

6 \*\*\*\*\*\*

5 \*\*\*\*\*

### Problem 1: Roll the Dice

In games with two dice (such as Monopoly), rolling two 6-sided dice and getting the same side or number on both dice is called “doubles” (i.e. the first die rolls a 5 and the second die rolls a 5 at the same time). One special type of “doubles” roll is called “snake eyes,” which is rolling two 1’s at the same time.

The most common size of dice used in games today is the 6-sided dice, but there are plenty of other sizes available. For example, popular role-playing games such as Dungeons and Dragons use dice with a variety of sizes, such as 4, 12 and 20 sided dice.

一群不同颜色的手机

中度可信度描述已自动生成

Some multi-sided dice

For this assignment you will be writing a program which first prompts the user for the number of sides to the dice they will be rolling. Your user should be limited to selecting dice with the following number of sides:

* 4 sides
* 6 sides
* 8 sides
* 10 sides
* 12 sides
* 20 sides

You can assume that that your user will enter integers when prompted. You will need to validate their input before you continue (i.e. entering -10 should cause your program to tell the user that their input is invalid). You should re-prompt the user to enter a value if they supply bad data. Hint: use a “while” loop to keep the user “trapped” until they supply you with “good” data.

Next, your program should keep “rolling the dice” until the program generates a “snake eyes” roll (1 and 1). This means that you will roll two virtual dice of the specified size at a time. The program should “announce” every pair rolled and tell the user if the roll qualifies as one of the following:

* A double roll (i.e. [2 and 2], [4 and 4], [6 and 6])
* A high roll (i.e. on a six sided die, [6 and 6])
* A high/low roll (i.e. on a six sided die, [1 and 6] or [6 and 1])
* An even roll (i.e. both die values are even: [2 and 2], [4 and 4])
* An odd roll (i.e. both die values are odd: [3 and 3], [5 and 5])
* A sum value (i.e. adding both die values together equals the size of the current die)
* Snake Eyes (1 and 1) - this should end the rolling portion of the program

At the end of the game you will want to calculate the average roll for each die and present this information to the user. See the sample program below to see what this should look like. You should format this number to two decimal places. Here’s a sample running of the program:

How many sides on your dice (4, 6, 8, 10, 12 or 20)? 13

Invalid size, try again.

How many sides on your dice (4, 6, 8, 10, 12 or 20)? 2

Invalid size, try again.

How many sides on your dice (4, 6, 8, 10, 12 or 20)? -1

Invalid size, try again.

How many sides on your dice (4, 6, 8, 10, 12 or 20)? 6

Thanks, here we go!

1. die #1 is \*3\* and die #2 is \*2\*

2. die #1 is \*1\* and die #2 is \*5\* Odds! Sum value is size value!

3. die #1 is \*3\* and die #2 is \*5\* Odds!

4. die #1 is \*3\* and die #2 is \*3\* Odds! Doubles! Sum value is size value!

5. die #1 is \*1\* and die #2 is \*2\*

6. die #1 is \*4\* and die #2 is \*2\* Evens! Sum value is size value!

7. die #1 is \*5\* and die #2 is \*1\* Odds! Sum value is size value!

8. die #1 is \*5\* and die #2 is \*5\* Odds! Doubles!

9. die #1 is \*4\* and die #2 is \*3\*

10. die #1 is \*3\* and die #2 is \*2\*

11. die #1 is \*6\* and die #2 is \*5\*

12. die #1 is \*4\* and die #2 is \*1\*

13. die #1 is \*4\* and die #2 is \*1\*

14. die #1 is \*2\* and die #2 is \*5\*

15. die #1 is \*2\* and die #2 is \*4\* Evens! Sum value is size value!

16. die #1 is \*6\* and die #2 is \*1\* High / Low!

17. die #1 is \*6\* and die #2 is \*5\*

18. die #1 is \*1\* and die #2 is \*2\*

19. die #1 is \*6\* and die #2 is \*3\*

20. die #1 is \*3\* and die #2 is \*6\*

21. die #1 is \*1\* and die #2 is \*2\*

22. die #1 is \*1\* and die #2 is \*1\* Odds! Doubles! Snake Eyes!

You finally got snake eyes on roll # 22

Along the way you rolled DOUBLES 3 times. (13.64% of all rolls were doubles)

Along the way you rolled TWO HIGH VALUES 0 times (0.00% of all rolls were two high values)

Along the way you rolled TWO EVENS 2 times. (9.09% of all rolls were two evens)

Along the way you rolled TWO ODDS 6 times. (27.27% of all rolls were two odds)

Along the way you rolled HIGH / LOW 1 times. (4.55% of all rolls were high/low)

Along the way you rolled A SUM VALUE 5 times. (22.73% of all rolls were a sum value)

Average roll for die #1: 3.36

Average roll for die #2: 3.00

Note that the “snake eyes” roll that causes your loop to end should also count as a “doubles” roll.

### Problem #2: Pick Up Sticks

For this assignment you will be creating an interface for two humans to play a simple game of “Pick Up Sticks”. Here’s how the game is played in the “real world”:

* The game begins with a number of sticks on a table (between 10 and 100)
* Each player, in turn, takes between 1-3 sticks off the table.
* The player to take the last stick loses.

Your job is to build a virtual version of the game using Python. The program is broken into a series of parts to help you understand how to put together a game with many different components.Each part builds on the previous one, so you should complete them in order and turn in your the final part as your solution.

**Part 1: Single Player Game**

Begin by asking the user for a number of sticks to be used in the game. Only accept numbers between 10 and 100 - if the user supplies a number outside of this range you should re-prompt them.

Next, continually announce to the user how many sticks are on the table and ask the user to enter in a number of sticks to take away. Only accept choices of 1,2 or 3 sticks - anything else should cause an error message to be displayed. Once a valid number of sticks has been entered you should deduct that # of sticks from the total number of sticks in the game. When you reach 0 sticks left the game is over.

Here is a sample running of the program:

How many sticks (10-100)? 999

Sorry, that's too many sticks. Try again

How many sticks (10-100)? -10

Sorry, that's too few sticks. Try again.

How many sticks (10-100)? 10

Great Let's play ...

There are 10 sticks on the table.

How many sticks do you want to take (1, 2 or 3)? 999

Sorry, that's not a valid number.

There are 10 sticks on the table.

How many sticks do you want to take (1, 2 or 3)? 0

Sorry, that's not a valid number.

There are 10 sticks on the table.

How many sticks do you want to take (1, 2 or 3)? 3

There are 7 sticks on the table.

How many sticks do you want to take (1, 2 or 3)? 2

There are 5 sticks on the table.

How many sticks do you want to take (1, 2 or 3)? 3

There are 2 sticks on the table.

How many sticks do you want to take (1, 2 or 3)? 3

Sorry, that would bring the total # of sticks below 0. Try again.

There are 2 sticks on the table.

How many sticks do you want to take (1, 2 or 3)? 2

There are no sticks left on the table! Game over.

You do not need to turn this part in - instead, once you are happy with it continue on to part 2. You will be turning that part in to Brightspace.

**Part 2: Two Player Game**

Note: Part 2 builds on Part 1, so make a copy of the code you wrote for Part 1 to get started.

As you can see, the single player version of this game isn’t very fun. To make things more interesting we are now going to add in an element of competition. For this part you will be implementing a two player game where players take turns taking sticks from the table. The same rules apply - each player can only take 1, 2 or 3 sticks per turn. The player who takes the last stick off of the table is the loser.

Here’s a sample running of the game:

How many sticks (10-100)? 999

Sorry, that's too many sticks. Try again

How many sticks (10-100)? 10

Great Let's play ...

Turn: Player 1

There are 10 sticks on the table.

How many sticks do you want to take (1, 2 or 3)? 999

Sorry, that's not a valid number.

Turn: Player 1

There are 10 sticks on the table.

How many sticks do you want to take (1, 2 or 3)? 0

Sorry, that's not a valid number.

Turn: Player 1

There are 10 sticks on the table.

How many sticks do you want to take (1, 2 or 3)? 3

Turn: Player 2

There are 7 sticks on the table.

How many sticks do you want to take (1, 2 or 3)? 2

Turn: Player 1

There are 5 sticks on the table.

How many sticks do you want to take (1, 2 or 3)? 3

Turn: Player 2

There are 2 sticks on the table.

How many sticks do you want to take (1, 2 or 3)? 1

Turn: Player 1

There is 1 stick on the table.

How many sticks do you want to take (1, 2 or 3)? 0

Sorry, that's not a valid number.

Turn: Player 1

There is 1 stick on the table.

How many sticks do you want to take (1, 2 or 3)? 1

There are no sticks left on the table! Game over.

Player 2 wins!

Here are some hints to get you started:

* You may want to use a variable to keep track of the players and who’s turn it is
* In your game you only want to switch turns when the player takes a valid # of sticks
* Do not allow players to take more than the total # of sticks on the table (i.e. if there are 2 sticks left the player should not be able to take 3 sticks)
* The player who takes the last stick is the loser.

### Problem #3: Decimal to Binary

In Computer Science we often work with different number systems to represent data. The most commonly used number system (which we refer to as “decimal”) is a “base 10” number system, which means that we use the values 0-9 to represent all numbers.

The binary number system is a “base 2” number system, which means that we use the values 0 and 1 to represent all numbers. We refer to each value in a binary number as a “bit” (i.e. the number 0101 consists of 4 bits)

You can convert a decimal number into a binary number by doing the following:

* Attempt to divide the number by 2.
* If there is a remainder you should record the bit 1.
* If there is no remainder, you should record the bit 0.
* Divide the number by 2 and throw away the remainder.
* If the number is positive, repeat the process.
* If the number is 0, you should end the process.

For example, consider the following decimal number:

decimal = 5

Here is how the algorithm works on this number:

5 ÷ 2 = 2 remainder 1 1

5 -> 2

2 ÷ 2 = 1 remainder 0 01

2 -> 1

1 ÷ 2 = 0 remainder 1 101

2 -> 0 (algorithm stops)

Your challenge for this assignment is to write a program that prompts the user for an integer greater than or equal to zero. If the user supplies a value less than 0 you should re-prompt them. Next, articulate the steps necessary to convert the number in question to binary.

Hint: Integers do not support leading 0’s (i.e. 0001 will be reduced to 1). For this program you should use a String to hold your bits (i.e. “0001” is allowed). You will need to figure out a way how to add bits to your String (hint: string concatenation)

Here are a few sample runnings of your program:

Enter a number greater than or equal to 0: -5

Invalid, try again

Enter a number greater than or equal to 0: -9

Invalid, try again

Enter a number greater than or equal to 0: 9

9 % 2 = 1 --- 1

9 / 2 = 4

4 % 2 = 0 --- 01

4 / 2 = 2

2 % 2 = 0 --- 001

2 / 2 = 1

1 % 2 = 1 --- 1001

1 / 2 = 0

9 in binary is 1001

Enter a number greater than or equal to 0: 86

86 % 2 = 0 --- 0

86 / 2 = 43

43 % 2 = 1 --- 10

43 / 2 = 21

21 % 2 = 1 --- 110

21 / 2 = 10

10 % 2 = 0 --- 0110

10 / 2 = 5

5 % 2 = 1 --- 10110

5 / 2 = 2

2 % 2 = 0 --- 010110

2 / 2 = 1

1 % 2 = 1 --- 1010110

1 / 2 = 0

86 in binary is 1010110

Enter a number greater than or equal to 0: 255

255 % 2 = 1 --- 1

255 / 2 = 127

127 % 2 = 1 --- 11

127 / 2 = 63

63 % 2 = 1 --- 111

63 / 2 = 31

31 % 2 = 1 --- 1111

31 / 2 = 15

15 % 2 = 1 --- 11111

15 / 2 = 7

7 % 2 = 1 --- 111111

7 / 2 = 3

3 % 2 = 1 --- 1111111

3 / 2 = 1

1 % 2 = 1 --- 11111111

1 / 2 = 0

255 in binary is 11111111

Enter a number greater than or equal to 0: 256

256 % 2 = 0 --- 0

256 / 2 = 128

128 % 2 = 0 --- 00

128 / 2 = 64

64 % 2 = 0 --- 000

64 / 2 = 32

32 % 2 = 0 --- 0000

32 / 2 = 16

16 % 2 = 0 --- 00000

16 / 2 = 8

8 % 2 = 0 --- 000000

8 / 2 = 4

4 % 2 = 0 --- 0000000

4 / 2 = 2

2 % 2 = 0 --- 00000000

2 / 2 = 1

1 % 2 = 1 --- 100000000

1 / 2 = 0

256 in binary is 100000000