Homework 1 Part 2

This is an individual assignment.

Description

Create or edit this Jupyter Notebook to answer the questions below. Use simulations to answer these questions. An analytical solution can be useful to check if your simulation is correct but analytical solutions alone will not be accepted as a solution to a problem.

Problem 9

Consider repeatedly rolling a fair 4-sided die.

1. Create a simulation to compute the probability that the top face will be 4 at least once on four rolls of the die?
2. Create a simulation to compute the probability that the top face will be 4 at least once on 20 rolls of the die?
3. Create a simulation to compute how many rolls of the die would you have to do to be 90% confident that you would see at least one 4?
4. Using the formula you have computed in problem 2 part 4, make a Python function that takes in the target value p and outputs the required number of rolls of an integer.
   A. Find the values for $p = 0.95$ and $p = 0.99$.
   B. Use your simulation to verify that the number of rolls you specified is sufficient to achieve $p \geq 0.95$.

In [6]:
```
#answer to 9 - 1
import random
import numpy as np
number_of_test = 100000
number_of_rolls = 4
success = 0
truth_array = np.zeros(number_of_test)
for test_number in list(range(0,number_of_test)): #iterates through from 1 to numbe
    for value in list(range(0,number_of_rolls)):  #iterates through from to to numb
        random_value = random.randint(1,4)        #generates random value from 1 to
        if random_value == 4:
            truth_array[test_number] = 1          #If a 4 is rolled make turth arra
print("The simulated number of times rolling a 4 at least once in " + str(number_of
```

```
The simulated number of times rolling a 4 at least once in 100000 test is 68406.0
Thus the simulated odds are: 0.68406
```

In [7]:
```
#answer to 9 - 2
import random
import numpy as np    #logic same as part 1
```

```
number_of_test = 100000
number_of_rolls = 20
success = 0
truth_array = np.zeros(number_of_test)
for test_number in list(range(0,number_of_test)):
    for value in list(range(0,number_of_rolls)):
        random_value = random.randint(1,4)
        if random_value == 4:
            truth_array[test_number] = 1
print("The simulated number of times rolling a 4 at least once in " + str(number_of
```

```
The simulated number of times rolling a 4 at least once in 100000 test is 99673.0
Thus the calculated odds are: 0.99673
```

In [8]:

```
#answer to 9 - 3
import random
import numpy as np
number_of_test = 100000
number_of_rolls = 0
success = 0
truth_array = np.zeros(number_of_test)
odds = 0
while odds < .90:          #only real new logic, repeat trials until calculated odds
    number_of_rolls = number_of_rolls + 1     #increase number of rolls every loop
    truth_array = np.zeros(number_of_test)
    for test_number in list(range(0,number_of_test)):
        for value in list(range(0,number_of_rolls)):
            random_value = random.randint(1,4)
            if random_value == 4:
                truth_array[test_number] = 1
    odds = (sum(truth_array))/number_of_test
print(number_of_rolls)
print("The simulated number of times rolling a 4 at least once in " + str(number_of
print("\nThis is the first simulation to produce odds greater than 90%")
```

```
9
The simulated number of times rolling a 4 at least once in 100000 test is 92491.0
Thus the calculated odds are: 0.92491

This is the first simulation to produce odds greater than 90%
```

In [9]:

```
#answer to 9 - 4a and running of function for aformentioned values
import random
import numpy as np
def function_calc(probability):
    number_of_test = 100000 #same logic as question above just in a function
    number_of_rolls = 0
    truth_array = np.zeros(number_of_test)
    odds = 0
    while odds < probability:
        number_of_rolls = number_of_rolls + 1
        truth_array = np.zeros(number_of_test)
        for test_number in list(range(0, number_of_test)):
            for value in list(range(0, number_of_rolls)):
                random_value = random.randint(1, 4)
                if random_value == 4:
                    truth_array[test_number] = 1
        odds = (sum(truth_array)) / number_of_test
    print(number_of_rolls)
    print("The simulated number of times rolling a 4 at least once in " + str(numbe
    sum(truth_array)) + " Thus the calculated odds are: " + str((sum(truth_array))
    print("\nThis is the first simulation to produce odds greater than " + str(prob
function_calc(.95)
function_calc(.99)
```

```
11
The simulated number of times rolling a 4 at least once in 100000 test is 95695.0
Thus the calculated odds are: 0.95695

This is the first simulation to produce odds greater than 95.0%
16
The simulated number of times rolling a 4 at least once in 100000 test is 99033.0
Thus the calculated odds are: 0.99033

This is the first simulation to produce odds greater than 99.0%
```

In [10]:

```
#Testing of running for part 4b
import random
import numpy as np
number_of_test = 100000
number_of_rolls = 17
success = 0
truth_array = np.zeros(number_of_test)
for test_number in list(range(0,number_of_test)):
    for value in list(range(0,number_of_rolls)):
        random_value = random.randint(1,4)
        if random_value == 4:
            truth_array[test_number] = 1
print("The simulated number of times rolling a 4 at least once in " + str(number_of
```

```
The simulated number of times rolling a 4 at least once in 100000 test is 99255.0
Thus the calculated odds are: 0.99255
```

Problem 10

Create a simulation function where you will roll a fair 6-sided die twice. Use simulation to find out the probability of getting a 4,5, or 6 on the first toss and a 1,2,3 on the second toss.

In [11]:

```
import numpy as np
import random
number_of_simulations = 10000
dice_rolls = np.zeros(number_of_simulations*2)  #tracks every dice roll
successes = np.zeros(number_of_simulations)     #keeps tracks of successes
for test_number in list(range(0,number_of_simulations)):
    random_value = random.randint(1,6)
    random_value2 = random.randint(1, 6)
    dice_rolls[test_number*2] = random_value     #stores roll 1
    dice_rolls[test_number*2 + 1] = random_value2#stores roll 2
for value_thing in enumerate(successes):
    if dice_rolls[value_thing[0]*2] == 1 or dice_rolls[value_thing[0]*2] == 2 or di
        if dice_rolls[value_thing[0]*2+1] == 4 or dice_rolls[value_thing[0]*2+1] ==
            successes[value_thing[0]] = 1   #checks if success, if true then alter
```

```
print("A trial of " + str(number_of_simulations) + " simulations returned a probabi
print(sum(successes))
```

```
A trial of 10000 simulations returned a probability of 2452.0
```

Problem 11

Suppose that you have a bag with 3 coins. One of them is a fair coin, but the others are biased trick coins. When flipped, the three coins come up heads with probability $\frac{1}{2}$, $\frac{1}{3}$, and $\frac{1}{4}$, respectively.

Consider the experiment where you pick one coin at random and flip it three times. Let H_i be the event that the coin comes up heads on flip i. What is the probability of the outcome $H_1 \cap H_2 \cap \overline{H_3}$?

With small modification in your code, find out the probability of the outcome $H_1 \cap \overline{H_2} \cap \overline{H_3}$.

Use simulation to find out the probability.

In [12]:
```
#Answer to first part of problem 11
import random
import numpy as np
heads_tracker = np.zeros(3)   #tracks individual events
number_of_trials = 100000
success = 0
heads_prob = [0,1]            #sample set
for test_number in list(range(0,number_of_trials)):
    for coin_flip in enumerate(heads_tracker):
        coin_value = heads_prob[random.randint(0,1)]  #implements probability
        heads_tracker[coin_flip[0]] = coin_value
    if heads_tracker[0] == 1:
        if heads_tracker[1] == 1:
            if heads_tracker[2] == 0:
                success = success + 1 #checks if success
first_coin_probability = success/number_of_trials
print("Odds for first coin:")
print(first_coin_probability)
# above code calculates probability for first coin

#Below code uses same logic, just alters probability of flip
heads_tracker = np.zeros(3)
number_of_trials = 100000
success = 0
heads_prob = [0,0,1]
for test_number in list(range(0,number_of_trials)):
    for coin_flip in enumerate(heads_tracker):
        coin_value = heads_prob[random.randint(0,2)]
        heads_tracker[coin_flip[0]] = coin_value
    if heads_tracker[0] == 1:
        if heads_tracker[1] == 1:
            if heads_tracker[2] == 0:
                success = success + 1
second_coin_probability = success/number_of_trials
print("Odds for second coin:")
print(second_coin_probability)
# above code calculates probability for second coin


heads_tracker = np.zeros(3)
number_of_trials = 100000
success = 0
heads_prob = [0,0,0,1]
for test_number in list(range(0,number_of_trials)):
    for coin_flip in enumerate(heads_tracker):
        coin_value = heads_prob[random.randint(0,3)]
        heads_tracker[coin_flip[0]] = coin_value
    if heads_tracker[0] == 1:
        if heads_tracker[1] == 1:
            if heads_tracker[2] == 0:
                success = success + 1
third_coin_probability = success/number_of_trials
print("Odds for third coin:")
print(third_coin_probability)
# above code calculates probability for third coin
value = third_coin_probability * (1/3) + second_coin_probability * (1/3) + third_co
print("Odds for first entire situation: ")
print(value)
```

```
Odds for first coin:
0.12536
Odds for second coin:
0.07351
Odds for third coin:
0.04788
Odds for first entire situation:
0.05642333333333334
```

In [13]:

```
#Answer for second part of problem 11
import random
import numpy as np
heads_tracker = np.zeros(3)
number_of_trials = 100000
success = 0
heads_prob = [0,1]
for test_number in list(range(0,number_of_trials)):
    for coin_flip in enumerate(heads_tracker):
        coin_value = heads_prob[random.randint(0,1)]
        heads_tracker[coin_flip[0]] = coin_value
    if heads_tracker[0] == 1:
        if heads_tracker[1] == 0: #one of three changed lines
            if heads_tracker[2] == 0:
                success = success + 1
first_coin_probability = success/number_of_trials
print("Odds for first coin:")
print(first_coin_probability)
# above code calculates probability for first coin
```

```
heads_tracker = np.zeros(3)
number_of_trials = 100000
success = 0
heads_prob = [0,0,1]
for test_number in list(range(0,number_of_trials)):
    for coin_flip in enumerate(heads_tracker):
        coin_value = heads_prob[random.randint(0,2)]
        heads_tracker[coin_flip[0]] = coin_value
    if heads_tracker[0] == 1:
        if heads_tracker[1] == 0: #second of three changed lines
            if heads_tracker[2] == 0:
                success = success + 1
second_coin_probability = success/number_of_trials
print("Odds for second coin:")
print(second_coin_probability)
# above code calculates probability for second coin


heads_tracker = np.zeros(3)
number_of_trials = 100000
success = 0
heads_prob = [0,0,0,1]
for test_number in list(range(0,number_of_trials)):
    for coin_flip in enumerate(heads_tracker):
        coin_value = heads_prob[random.randint(0,3)]
        heads_tracker[coin_flip[0]] = coin_value
    if heads_tracker[0] == 1:
        if heads_tracker[1] == 0: #third of third changed lines
            if heads_tracker[2] == 0:
                success = success + 1
third_coin_probability = success/number_of_trials
print("Odds for third coin:")
print(third_coin_probability)
# above code calculates probability for third coin
value = third_coin_probability * (1/3) + second_coin_probability * (1/3) + third_co
print("Odds for second entire situation: ")
print(value)
```

```
Odds for first coin:
0.12503
Odds for second coin:
0.1497
Odds for third coin:
0.14123
Odds for second entire situation:
0.1440533333333333
```

Submit Your Solutions

Confirm that you've successfully completed the assignment.

Along with the Notebook, include a PDF of the notebook with your solutions.

`add` and `commit` the final version of your work, and `push` your PDF file to your GitHub repository.

Submit the URL of your GitHub Repository as your assignment submission on Canvas.