Exploring the Math Behind Crash | Roobet Cryptocurrency Casino Game

Exploring the math, statistics, and code of the Roobet Crash cryptocurrency casino game in order to calculate the expected value and average loss per game. Using Python, we are able to simulate different betting strategies and determine the effectiveness of these approaches.

Necessary Libraries and Constants

In [2]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import hashlib
import random
import string
import hmac

e = 2**52
salt = "0000000000000000000fa3b65e43e4240d71762a5bf397d5304b2596d116859c"
```

Get Result and Get Previous Game Function

$$int \left(\dfrac{100\cdot e - h}{e-h} \right) \cdot \dfrac{1}{100}$$

In [3]:
```
def get_result(game_hash):
    hm = hmac.new(str.encode(game_hash), b'', hashlib.sha256)
    hm.update(salt.encode("utf-8"))
    h = hm.hexdigest()
    if (int(h, 16) % 33 == 0):
        return 1
    h = int(h[:13], 16)
    e = 2**52
    return (((100 * e - h) / (e-h)) // 1) / 100.0

def get_prev_game(hash_code):
    m = hashlib.sha256()
    m.update(hash_code.encode("utf-8"))
    return m.hexdigest()
```

Collecting All Game Results

In [4]:
```
game_hash = '100af1b49f5e9f87efc81f838bf9b1f5e38293e5b4cf6d0b366c004e0a8d9987'
first_game = "77b271fe12fca03c618f63dfb79d4105726ba9d4a25bb3f1964e435ccf9cb209

results = []
count = 0
while game_hash != first_game:
    count += 1
```

```
    results.append(get_result(game_hash))
    game_hash = get_prev_game(game_hash)

results = np.array(results)
```

In [5]:
```
len(results)
```

Out[5]:
```
618990
```

We could use more result. but i don't think that necessory

In [68]:
```
game_hash = 'hash of the last game'

results = []
count = 0
for x in range(10**7):
    count += 1
    results.append(get_result(game_hash))
    game_hash = get_prev_game(game_hash)

results = np.array(results)
```

In [69]:
```
len(results)
```

Out[69]:
```
10000000
```

Testing Probability Formula

Probability of lossing of using the data and using the formula

Let U ~ $Uniform(0,e)$ Where $e = 2^{52}$

$Multiplier = \dfrac{100e-U}{e-U} \cdot \dfrac{1}{100}$

$Multiplier = \dfrac{99e + (e-U)}{e-U} \cdot \dfrac{1}{100}$

$\approx \dfrac{99e}{100U} + 0.01$

$\approx 0.01 + \dfrac{0.99}{Uniform(0,1)}$

$\dfrac{0.99}{Uniform(0,1)}$

So, $P(X\leq x) \approx 1 - \dfrac{1}{x}$

According to the code:

$$P(X\leq x) = \dfrac{1}{33} + \dfrac{32}{33}(0.01+0.99(1-\dfrac{1}{x}))$$

Probability of lossing

In [26]:
```
multiplier = 2
1/33 + (32/33)*(.01 + .99*(1 - 1/multiplier))
```

Out[26]: 0.52

In [27]:
```
(results <= multiplier).mean()
```

Out[27]: 0.5194978917268454

Testing Expected Value Formula

In [13]:
```
multiplier = 2
((1/33) + (32/33)*(.01 + .99*(1 - 1/(multiplier-.01))))*-1 + (multiplier-1)*(1
```

Out[13]: -0.03517587939698519

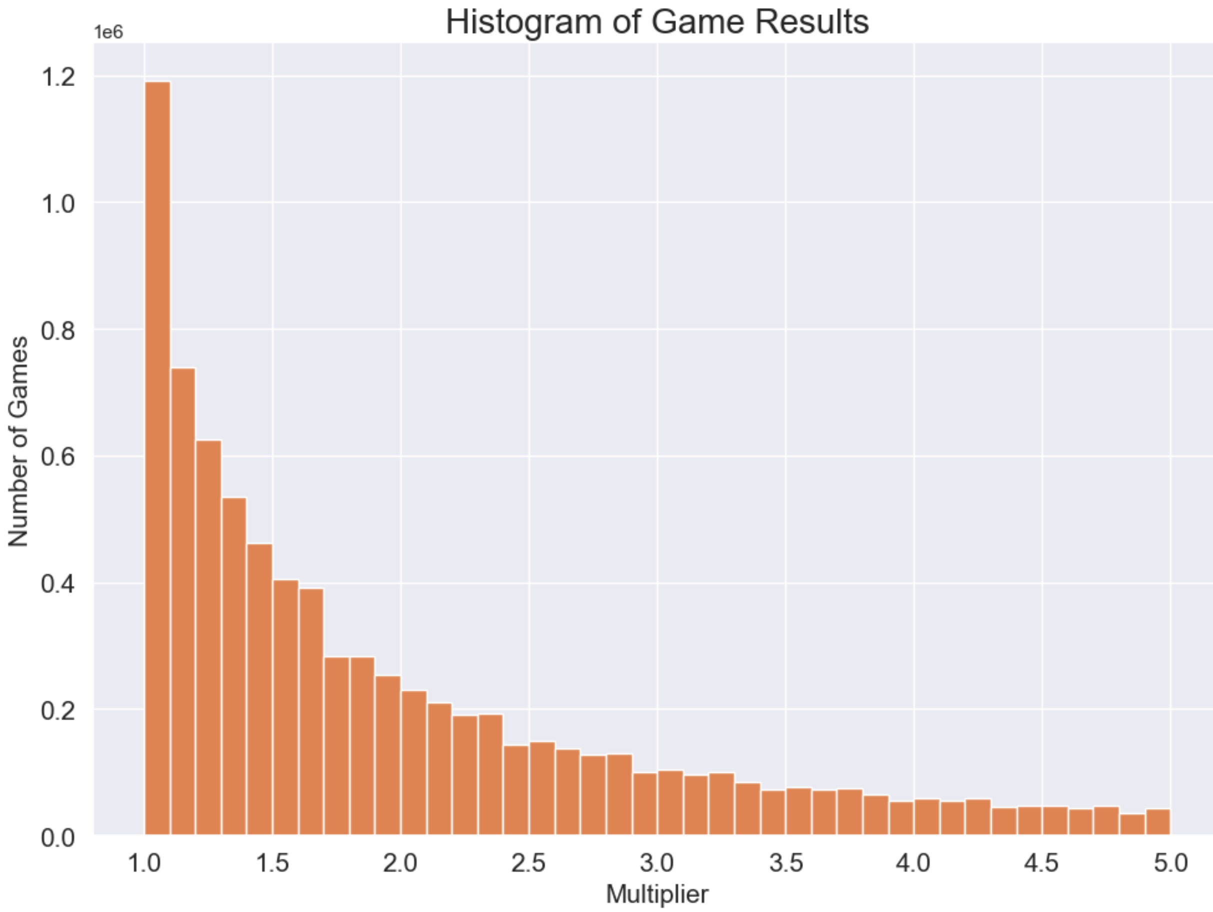
In [28]:
```
(results < multiplier).mean() * -1 + (multiplier - 1)*(results >= multiplier).
```

Out[28]: -0.03412009887074108

Visualizations

In [70]:
```
sns.set(rc={'figure.figsize':(11.7,8.27)})
plt.hist(results_guess, bins=40, range=(1,5))
plt.hist(results, bins=40, range=(1,5))
plt.title("Histogram of Game Results", fontsize=20)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.ylabel("Number of Games", fontsize=15)
plt.xlabel("Multiplier", fontsize=15)
```

Out[70]: Text(0.5, 0, 'Multiplier')

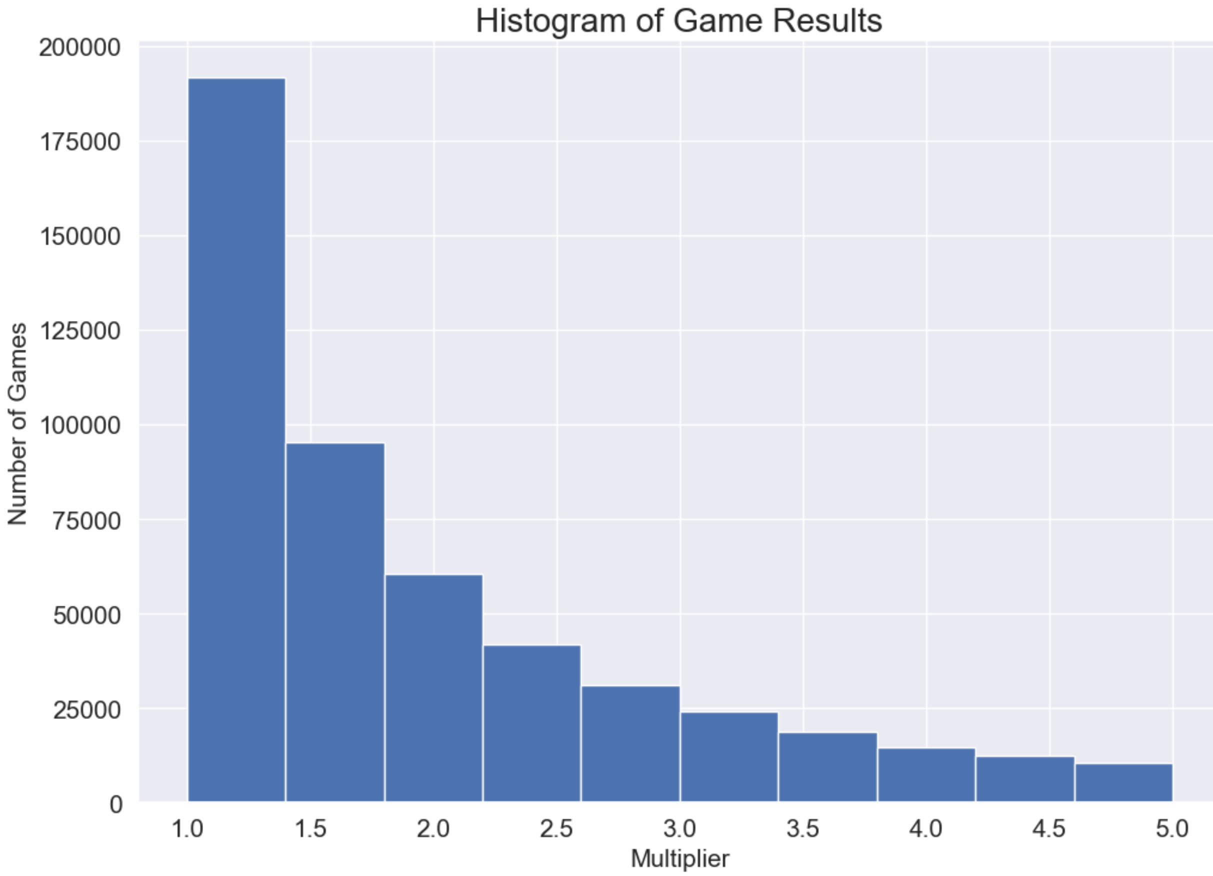


In [30]:

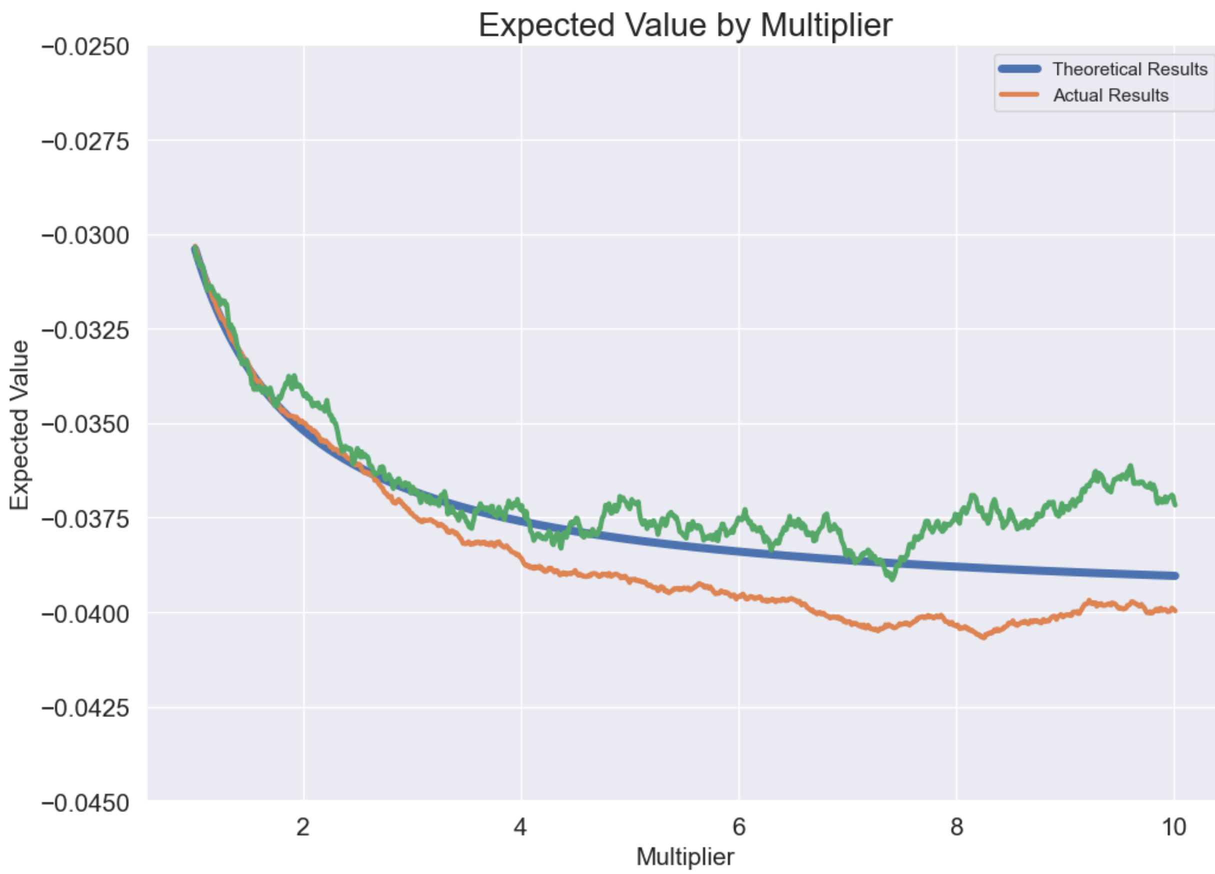
```
import seaborn as sns
sns.set(rc={'figure.figsize':(11.7,8.27)})
plt.hist(results, range=(1, 5))
plt.title("Histogram of Game Results", fontsize=20)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.ylabel("Number of Games", fontsize=15)
plt.xlabel("Multiplier", fontsize=15)
```

Out[30]:

```
Text(0.5, 0, 'Multiplier')
```



In [71]:
```
def calculate_ev(multiplier):
    return ((1/33) + (32/33)*(.01 + .99*(1 - 1/(multiplier-.01))))*-1 + (multi

xs = np.linspace(101, 1001, 901) / 100
ys = [calculate_ev(x) for x in xs]
y2s = [(results < x).mean() * -1 + (x - 1)*(results >= x).mean() for x in xs]

plt.plot(xs, ys, linewidth=5)

plt.xlabel("Multiplier", fontsize=15)
plt.ylabel("Expected Value", fontsize=15)
plt.ylabel("Expected Value", fontsize=15)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.ylim(-.045, -.025)

plt.plot(xs, y2s, linewidth=3)

plt.title("Expected Value by Multiplier", fontsize=20)
plt.legend(["Theoretical Results", "Actual Results"])
plt.show()
```



Martingale Strategy

In [74]:
```
negatives = []
in_a_row = 0
for multiplier in results:
    if multiplier < 2:
        in_a_row += 1
    else:
        in_a_row = 0
    negatives.append(in_a_row)
negatives = np.array(negatives)
```

In [87]:
```
for x in range(13):
    print(0.5174943**x)
```

```
1.0
0.5174943
0.26780035053248996
0.13858515493856552
0.0717170277453245
0.03711315307114728
0.01920584516934621
0.009938915401819197
0.005143332068623644
0.002661645028519944
0.0013773861308824084
0.0007127894716307003
0.00036886448866889907
```

In [75]:
```
for i in range(1, 14):
    print("Probability of Losing %d Game(s) in a Row:"%i, (negatives >= i).mea
```

```
Probability of Losing 1 Game(s) in a Row: 0.5174943
Probability of Losing 2 Game(s) in a Row: 0.2678253
Probability of Losing 3 Game(s) in a Row: 0.1386671
Probability of Losing 4 Game(s) in a Row: 0.0718481
Probability of Losing 5 Game(s) in a Row: 0.0372029
Probability of Losing 6 Game(s) in a Row: 0.019249
Probability of Losing 7 Game(s) in a Row: 0.009946
Probability of Losing 8 Game(s) in a Row: 0.0051366
Probability of Losing 9 Game(s) in a Row: 0.0026559
Probability of Losing 10 Game(s) in a Row: 0.0013808
Probability of Losing 11 Game(s) in a Row: 0.0007182
Probability of Losing 12 Game(s) in a Row: 0.0003701
Probability of Losing 13 Game(s) in a Row: 0.0001901
```

Martingale Strategy - Our Version

In [391…
```
def bid(begin_value=100, multiplier=1.2, end_value=100000):
    spend_list = [begin_value]
    bid_list = [begin_value]

    while spend_list[-1]<=end_value:
        bid_list.append(round((spend_list[-1]+begin_value)*(1/(multiplier-1)))
        spend_list.append(sum(bid_list))

    if spend_list[-1]>=end_value:
        spend_list.pop()
        bid_list.pop()


    print("Total Spend : \t\t\t\t\t", spend_list)
    print("Bid of Each Turn : \t\t\t\t", bid_list)
    print()
    print("No of turns you can play before runout of money: ", len(bid_list))

    negatives = []
    in_a_row = 0
    for result in results:
        if result < multiplier:
            in_a_row += 1
        else:
            in_a_row = 0
        negatives.append(in_a_row)
    negatives = np.array(negatives)
    print("\t\t\t\t\t\t\t From Data \t From Formula")
    print("\t\t\t\t\t\t\t-------------------------")
    for i in range(1, len(bid_list)+1):
        print("Probability of Losing %d Game(s) in a Row:\t"%i, round((negativ
```

In [393…
```
x = 1.145454545454 54
print("Our Guess :\t\t\t\t\t ", x)
bid(begin_value=100, multiplier=x, end_value=100000)
print("\n" ,"-"*100,"\n")
```

```
Our Guess :                                               1.1454545454545454
Total Spend :                                            [100, 1475, 12303, 97574]
Bid of Each Turn :                                       [100, 1375, 10828, 85271]

No of turns you can play before runout of money:  4
                                                         From Data       From Formula
                                                        -------------------------
Probability of Losing 1 Game(s) in a Row:        0.157804        0.161905
Probability of Losing 2 Game(s) in a Row:        0.02489         0.026213
Probability of Losing 3 Game(s) in a Row:        0.003934        0.004244
Probability of Losing 4 Game(s) in a Row:        0.00062         0.000687

 ------------------------------------------------------------------------------
----------------------
```

In [132…

```
for x in np.arange(1.1,2.1,0.1):
    print("Our Guess :\t\t\t\t\t ", round(x,5))
    bid(begin_value=100, multiplier=x, end_value=100000)
    print("\n" ,"-"*100,"\n")
```

```
Our Guess :                                       1.1
Total Spend :                                    [100, 2100, 24100]
Bid of Each Turn :                               [100, 2000, 22000]

No of turns you can play before runout of money:  3
                                                 From Data       From Formula
                                                ------------------------
Probability of Losing 1 Game(s) in a Row:        0.119189        0.127273
Probability of Losing 2 Game(s) in a Row:        0.014226        0.016198
Probability of Losing 3 Game(s) in a Row:        0.001697        0.002062

 -----------------------------------------------------------------------------
----------------------

Our Guess :                                       1.2
Total Spend :                                    [100, 1100, 7100, 43100]
Bid of Each Turn :                               [100, 1000, 6000, 36000]

No of turns you can play before runout of money:  4
                                                 From Data       From Formula
                                                ------------------------
Probability of Losing 1 Game(s) in a Row:        0.199963        0.2
Probability of Losing 2 Game(s) in a Row:        0.040024        0.04
Probability of Losing 3 Game(s) in a Row:        0.008002        0.008
Probability of Losing 4 Game(s) in a Row:        0.001602        0.0016

 -----------------------------------------------------------------------------
----------------------

Our Guess :                                       1.3
Total Spend :                                    [100, 767, 3657, 16180, 7044
7]
Bid of Each Turn :                               [100, 667, 2890, 12523, 5426
7]

No of turns you can play before runout of money:  5
                                                 From Data       From Formula
                                                ------------------------
Probability of Losing 1 Game(s) in a Row:        0.261434        0.261538
Probability of Losing 2 Game(s) in a Row:        0.068308        0.068402
Probability of Losing 3 Game(s) in a Row:        0.017876        0.01789
Probability of Losing 4 Game(s) in a Row:        0.004715        0.004679
Probability of Losing 5 Game(s) in a Row:        0.00125         0.001224

 -----------------------------------------------------------------------------
----------------------

Our Guess :                                       1.4
Total Spend :                                    [100, 600, 2350, 8475, 29912]
Bid of Each Turn :                               [100, 500, 1750, 6125, 21437]

No of turns you can play before runout of money:  5
                                                 From Data       From Formula
                                                ------------------------
Probability of Losing 1 Game(s) in a Row:        0.314213        0.314286
Probability of Losing 2 Game(s) in a Row:        0.098762        0.098776
Probability of Losing 3 Game(s) in a Row:        0.031093        0.031044
Probability of Losing 4 Game(s) in a Row:        0.009826        0.009757
Probability of Losing 5 Game(s) in a Row:        0.003113        0.003066
```

```
 ------------------------------------------------------------------------------
----------------------

Our Guess :                                        1.5
Total Spend :                                     [100, 500, 1700, 5300, 16100,
48500]
Bid of Each Turn :                                [100, 400, 1200, 3600, 10800,
32400]

No of turns you can play before runout of money:  6
                                                  From Data       From Formula
                                                 -------------------------
Probability of Losing 1 Game(s) in a Row:        0.359947        0.36
Probability of Losing 2 Game(s) in a Row:        0.129602        0.1296
Probability of Losing 3 Game(s) in a Row:        0.046692        0.046656
Probability of Losing 4 Game(s) in a Row:        0.016867        0.016796
Probability of Losing 5 Game(s) in a Row:        0.006116        0.006047
Probability of Losing 6 Game(s) in a Row:        0.002228        0.002177

 ------------------------------------------------------------------------------
----------------------

Our Guess :                                        1.6
Total Spend :                                     [100, 433, 1321, 3689, 10004,
26844, 71751]
Bid of Each Turn :                                [100, 333, 888, 2368, 6315, 1
6840, 44907]

No of turns you can play before runout of money:  7
                                                  From Data       From Formula
                                                 -------------------------
Probability of Losing 1 Game(s) in a Row:        0.399975        0.4
Probability of Losing 2 Game(s) in a Row:        0.160022        0.16
Probability of Losing 3 Game(s) in a Row:        0.064047        0.064
Probability of Losing 4 Game(s) in a Row:        0.025709        0.0256
Probability of Losing 5 Game(s) in a Row:        0.010333        0.01024
Probability of Losing 6 Game(s) in a Row:        0.004166        0.004096
Probability of Losing 7 Game(s) in a Row:        0.001678        0.001638

 ------------------------------------------------------------------------------
----------------------

Our Guess :                                        1.7
Total Spend :                                     [100, 386, 1080, 2766, 6860,
16803, 40950, 99593]
Bid of Each Turn :                                [100, 286, 694, 1686, 4094, 9
943, 24147, 58643]

No of turns you can play before runout of money:  8
                                                  From Data       From Formula
                                                 -------------------------
Probability of Losing 1 Game(s) in a Row:        0.435306        0.435294
Probability of Losing 2 Game(s) in a Row:        0.189512        0.189481
Probability of Losing 3 Game(s) in a Row:        0.082533        0.08248
Probability of Losing 4 Game(s) in a Row:        0.036014        0.035903
Probability of Losing 5 Game(s) in a Row:        0.015734        0.015628
Probability of Losing 6 Game(s) in a Row:        0.006863        0.006803
Probability of Losing 7 Game(s) in a Row:        0.002985        0.002961
Probability of Losing 8 Game(s) in a Row:        0.001294        0.001289
```

```
 ------------------------------------------------------------------------------
----------------------

Our Guess :                                        1.8
Total Spend :                                     [100, 350, 912, 2177, 5023, 1
1427, 25836, 58256]
Bid of Each Turn :                                [100, 250, 562, 1265, 2846, 6
404, 14409, 32420]

No of turns you can play before runout of money:  8
                                                  From Data       From Formula
                                                 -------------------------
Probability of Losing 1 Game(s) in a Row:         0.466678        0.466667
Probability of Losing 2 Game(s) in a Row:         0.217864        0.217778
Probability of Losing 3 Game(s) in a Row:         0.101735        0.10163
Probability of Losing 4 Game(s) in a Row:         0.047589        0.047427
Probability of Losing 5 Game(s) in a Row:         0.022285        0.022133
Probability of Losing 6 Game(s) in a Row:         0.0104          0.010329
Probability of Losing 7 Game(s) in a Row:         0.00485         0.00482
Probability of Losing 8 Game(s) in a Row:         0.002261        0.002249

 ------------------------------------------------------------------------------
----------------------

Our Guess :                                        1.9
Total Spend :                                     [100, 322, 791, 1781, 3871, 8
283, 17597, 37260, 78771]
Bid of Each Turn :                                [100, 222, 469, 990, 2090, 44
12, 9314, 19663, 41511]

No of turns you can play before runout of money:  9
                                                  From Data       From Formula
                                                 -------------------------
Probability of Losing 1 Game(s) in a Row:         0.494674        0.494737
Probability of Losing 2 Game(s) in a Row:         0.244779        0.244765
Probability of Losing 3 Game(s) in a Row:         0.121167        0.121094
Probability of Losing 4 Game(s) in a Row:         0.060045        0.05991
Probability of Losing 5 Game(s) in a Row:         0.029746        0.02964
Probability of Losing 6 Game(s) in a Row:         0.014717        0.014664
Probability of Losing 7 Game(s) in a Row:         0.007271        0.007255
Probability of Losing 8 Game(s) in a Row:         0.003586        0.003589
Probability of Losing 9 Game(s) in a Row:         0.001779        0.001776

 ------------------------------------------------------------------------------
----------------------

Our Guess :                                        2.0
Total Spend :                                     [100, 300, 700, 1500, 3100, 6
300, 12700, 25500, 51100]
Bid of Each Turn :                                [100, 200, 400, 800, 1600, 32
00, 6400, 12800, 25600]

No of turns you can play before runout of money:  9
                                                  From Data       From Formula
                                                 -------------------------
Probability of Losing 1 Game(s) in a Row:         0.519902        0.52
Probability of Losing 2 Game(s) in a Row:         0.27033         0.2704
Probability of Losing 3 Game(s) in a Row:         0.14063         0.140608
Probability of Losing 4 Game(s) in a Row:         0.073208        0.073116
Probability of Losing 5 Game(s) in a Row:         0.038091        0.03802
```

```
Probability of Losing 6 Game(s) in a Row:        0.019801        0.019771
Probability of Losing 7 Game(s) in a Row:        0.010278        0.010281
Probability of Losing 8 Game(s) in a Row:        0.005332        0.005346
Probability of Losing 9 Game(s) in a Row:        0.00277         0.00278

 ----------------------------------------------------------------------------
-----------------------
```

In [356…

```
# Probability of Losing All the money
p_theorical = []
p_practical = []


def bid(begin_value=100, multiplier=1.2, end_value=100000):
    # At Begining
    spend_list = [begin_value]
    bid_list = [begin_value]

    # While Playing
    while spend_list[-1]<=end_value:
        bid_list.append(round((spend_list[-1]+begin_value)*(1/(multiplier-1)))
        spend_list.append(sum(bid_list))

    # We can't exeed the end value
    if spend_list[-1]>=end_value:
        spend_list.pop()
        bid_list.pop()

#     Calculate the Practical value
    negatives = []
    in_a_row = 0
    for result in results:
        if result < multiplier:
            in_a_row += 1
        else:
            in_a_row = 0
        negatives.append(in_a_row)
    negatives = np.array(negatives)

    p_practical.append((negatives >= len(bid_list)).mean())

    p_theorical.append((1/33 + (32/33)*(.01 + .99*(1 - 1/multiplier)))**len(bi
```

In [357…

```
len(np.linspace(1.1,2.0,100))
```

Out[357]:

```
100
```

In [358…

```
from tqdm import tqdm

xs = np.linspace(1.1,2.0,100)
for x in tqdm(xs):
    bid(begin_value=100, multiplier=x, end_value=100000)
```

```
100%|████████████████████████████████████████████████████████████████████| 100/1
00 [02:03<00:00,  1.24s/it]
```

In [359…

```
len(p_practical) , len(p_theorical)
```

Out[359]:

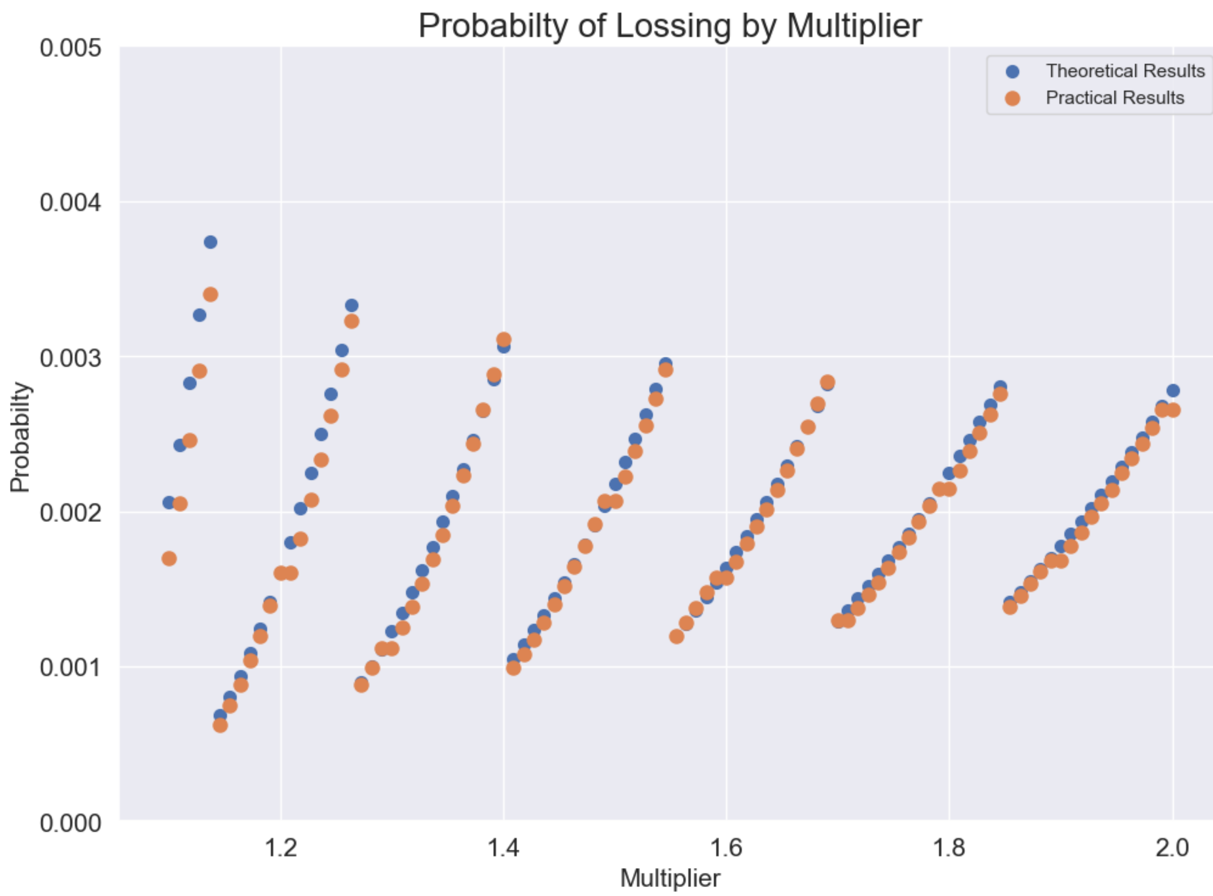
```
(100, 100)
```

In [360…

```
plt.plot(xs, p_theorical, linewidth=3)
plt.plot(xs, p_practical, linewidth=3)

plt.xlabel("Multiplier", fontsize=15)
plt.ylabel("Probabilty", fontsize=15)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.ylim(0, 0.005)

plt.title("Probabilty of Lossing by Multiplier", fontsize=20)
plt.legend(["Theoretical Results", "Practical Results"])
plt.show()
```

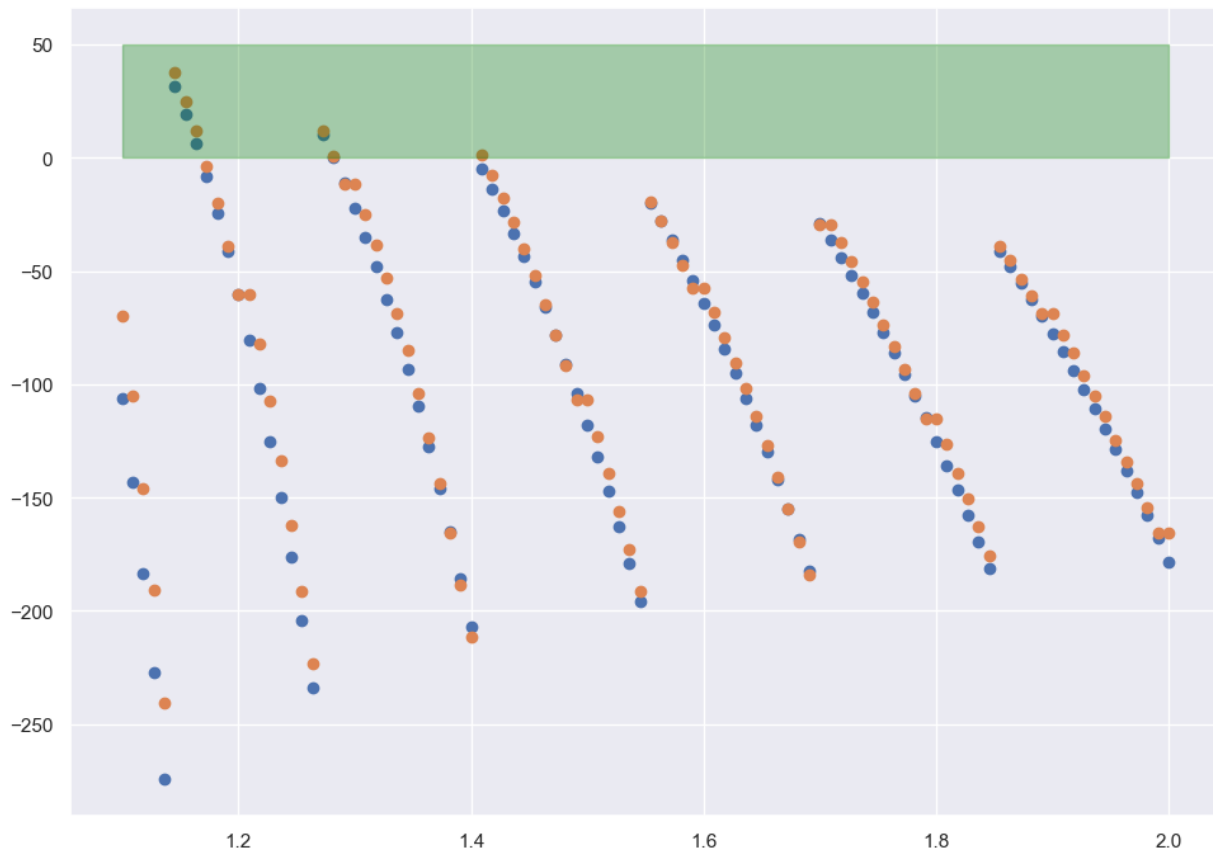
In [361…

```
plt.scatter(xs, p_theorical, linewidth=2)
plt.scatter(xs, p_practical, linewidth=3)

plt.xlabel("Multiplier", fontsize=15)
plt.ylabel("Probabilty", fontsize=15)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.ylim(0, 0.005)

plt.title("Probabilty of Lossing by Multiplier", fontsize=20)
plt.legend(["Theoretical Results", "Practical Results"])
plt.show()
```



In [389…

```
new1 = []
for i in range(len(p_theorical)):
    new1.append((1-p_theorical[i])*100-100000*p_theorical[i])
plt.scatter(xs, new1)

new2 = []
for i in range(len(p_practical)):
    new2.append((1-p_practical[i])*100-100000*p_practical[i])
plt.scatter(xs, new2)

plt.fill_between(xs, np.ones(len(new))*50, alpha=0.3, color='green')
plt.show()
```



In [377…

```
p_theorical[new2.index(max(new2))] , min(p_theorical)
```

Out[377]:

```
(0.0006871293339709279, 0.0006871293339709279)
```

In [383…

```
best = min(p_theorical)
best
```

Out[383]:

```
0.0006871293339709279
```

In [384…

```
xs[p_theorical.index(min(p_theorical))]
```

Out[384]:

```
1.1454545454545455
```

In [385…

```
(1-best)/best
```

Out[385]:

```
1454.330096622406
```

In [386…

```
(1-0.00278)*100-100000*0.00278
```

Out[386]:

```
-178.27800000000002
```

In [387…

```
(1-best)*100-100000*best
```

Out[387]:

```
31.21835366951011
```

If you Play until end of the World You will have profit of Rs.31

In [402…

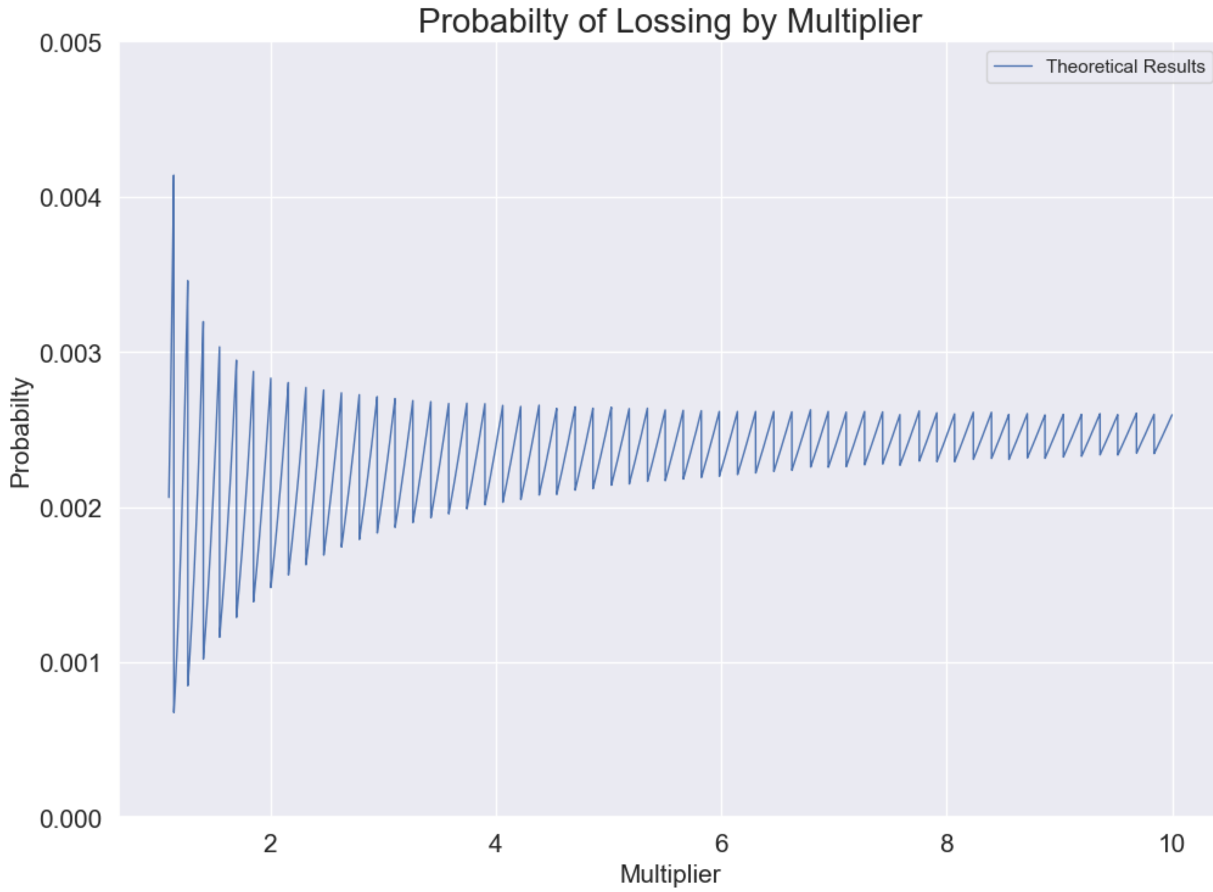
```
# Probability of Losing All the money
p_theorical2 = []
```

```python
p_practical2 = []


def bid(begin_value=100, multiplier=1.2, end_value=100000):
    # At Begining
    spend_list = [begin_value]
    bid_list = [begin_value]

    # While Playing
    while spend_list[-1]<=end_value:
        bid_list.append(round((spend_list[-1]+begin_value)*(1/(multiplier-1)))
        spend_list.append(sum(bid_list))

    # We can't exeed the end value
    if spend_list[-1]>=end_value:
        spend_list.pop()
        bid_list.pop()

#     Calculate the Practical value
#     negatives = []
#     in_a_row = 0
#     for result in results:
#         if result < multiplier:
#             in_a_row += 1
#         else:
#             in_a_row = 0
#         negatives.append(in_a_row)
#     negatives = np.array(negatives)

#     p_practical2.append((negatives >= len(bid_list)).mean())

    p_theorical2.append((1/33 + (32/33)*(.01 + .99*(1 - 1/multiplier)))**len(b

from tqdm import tqdm

xs = np.linspace(1.1,10.0,10000)
for x in tqdm(xs):
    bid(begin_value=100, multiplier=x, end_value=100000)
```

```
100%|██████████████████████████████████████████████████████████| 10000/10000
[00:00<00:00, 23714.04it/s]
```

In [403…

```python
plt.plot(xs, p_theorical2, linewidth=1)
# plt.scatter(xs, p_theorical2)
# plt.plot(xs, p_practical2, linewidth=3)

plt.xlabel("Multiplier", fontsize=15)
plt.ylabel("Probabilty", fontsize=15)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.ylim(0, 0.005)

plt.title("Probabilty of Lossing by Multiplier", fontsize=20)
plt.legend(["Theoretical Results", "Practical Results"])
plt.show()
```



In [405…

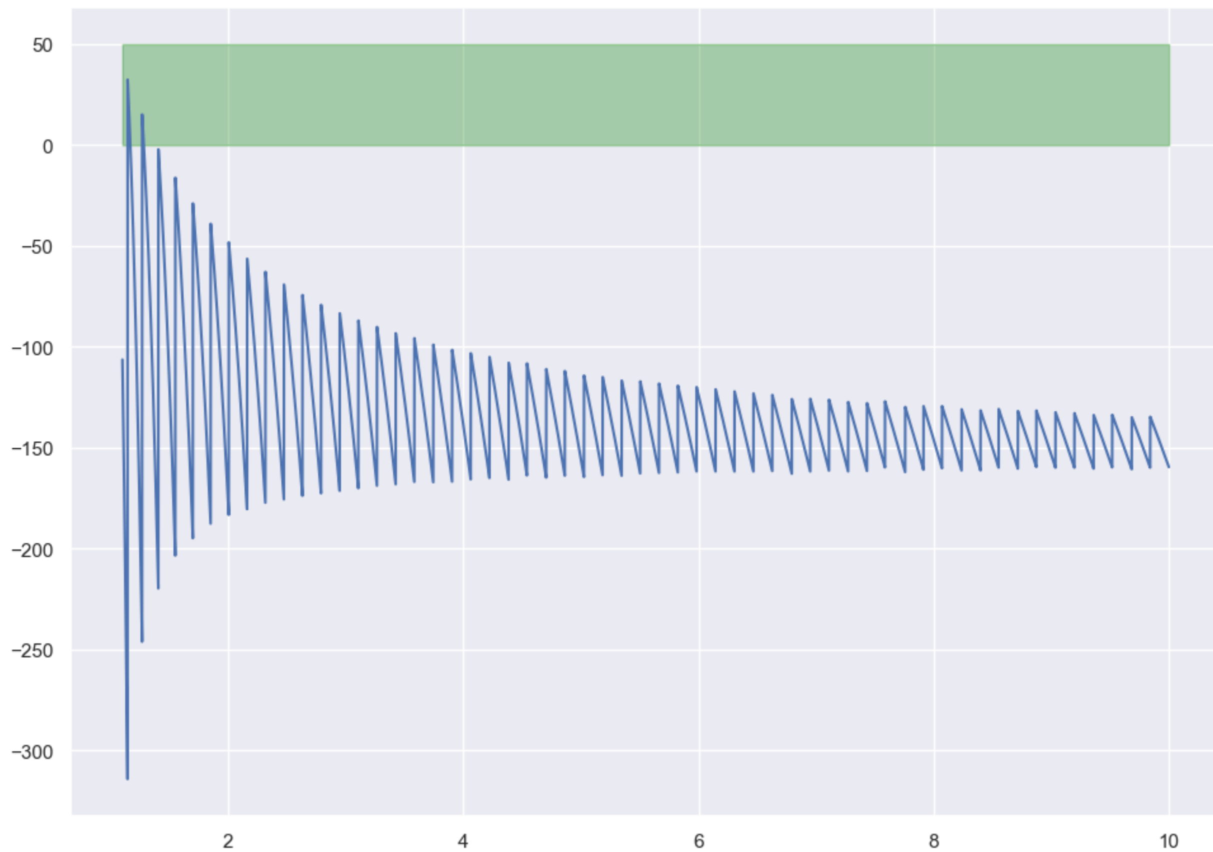
```
len(p_theorical2)
```

Out[405]:

```
10000
```

In [409…

```
new = []
for i in range(len(p_theorical2)):
    new.append((1-p_theorical2[i])*100-100000*p_theorical2[i])

plt.plot(xs, new)
plt.fill_between(xs, np.ones(len(new))*50, alpha=0.3, color='green')
plt.show()
```




```
In [ ]:
```

```
In [ ]:
```