

## Assignment 4

**Language used:** Python version 3.6.7

**Name of the functions:**

bernoulli <p>

binomial <n> <p>

geometric <p>

neg\_binomial <k> <p>

poisson < $\lambda$ >

arb\_discrete <p 0 > <p 1 > <p 2 > ... <p n >

uniform <a> <b>

exponential < $\lambda$ >

gamma < $\alpha$ > < $\lambda$ >

normal < $\mu$ > < $\sigma$ >

**NOTE:** arb\_discrete and neg\_binomial have underscores and not '-' in-between them.

Seed value –1343

The seed value is just below the imported libraries at the top of the program.

1. Bernoulli input:

python3 simulateDist.py 10 bernoulli 0.3

Bernoulli output:

```
The bernoulli distribution is  
[0, 1, 1, 0, 0, 1, 0, 0, 0, 1]
```

2. Binomial input: python3 simulateDist.py 10 binomial 8 0.3

Binomial output:

```
[0, 1, 1, 0, 0, 1, 0, 0] 3 is the number of successes
[0, 1, 0, 0, 0, 0, 1, 1] 3 is the number of successes
[0, 1, 0, 1, 0, 0, 0, 1] 3 is the number of successes
[0, 0, 1, 0, 0, 1, 0, 1] 3 is the number of successes
[0, 1, 0, 0, 1, 0, 0, 0] 2 is the number of successes
[0, 0, 0, 1, 0, 0, 0, 0] 1 is the number of successes
[1, 0, 1, 0, 0, 0, 0, 1] 3 is the number of successes
[0, 0, 0, 1, 1, 1, 1, 1] 5 is the number of successes
[0, 1, 0, 1, 0, 0, 0, 0] 2 is the number of successes
[1, 0, 0, 1, 0, 0, 0, 0] 2 is the number of successes
```

3. Geometric input: python3 simulateDist.py 5 geometric 0.32

Geometric output:

```
2 sample(s) were generated for getting a success
[0, 1]
1 sample(s) were generated for getting a success
[1]
3 sample(s) were generated for getting a success
[0, 0, 1]
4 sample(s) were generated for getting a success
[0, 0, 0, 1]
5 sample(s) were generated for getting a success
[0, 0, 0, 0, 1]
```

4. Negative binomial input: - python3 simulateDist.py 10 neg\_binomial 4 0.031

Negative binomial output: -

```
210 sample(s) were generated before 4 getting successes
38 sample(s) were generated before 4 getting successes
179 sample(s) were generated before 4 getting successes
245 sample(s) were generated before 4 getting successes
124 sample(s) were generated before 4 getting successes
94 sample(s) were generated before 4 getting successes
136 sample(s) were generated before 4 getting successes
123 sample(s) were generated before 4 getting successes
238 sample(s) were generated before 4 getting successes
160 sample(s) were generated before 4 getting successes
```

5. Exponential input: python3 simulateDist.py 10 exponential 0.4

Exponential output:-

```
-1.0169688691344214
-0.5899100961213822
-0.5040991132861214
-3.5166422738444933
-1.8724275627034082
-0.22064418501856356
-5.383653266334867
-1.5904337229573842
-2.40199765215259
-0.360100234973273
```

6. Uniform input: python3 simulateDist.py 10 uniform 10 30

Uniform output:-

```
23.31571245989656
25.796181523636633
26.34778849973138
14.899216841200252
19.457057124606543
28.31049878694294
12.321633347405331
20.586279589852566
17.65174106923258
27.31706063722103
```

7. Arb-discrete input: python3 simulateDist.py 100 arb\_discrete 0.33 0.33 0.33 0.01

**Note!! : python has this weird way of not rounding  $0.3+0.3+0.3+0.1$  to a  $1.0$ (it keeps it at a  $0.9999999$ ). Kindly try a different probability combination if that happens.**

Arb-discrete output:

```
[1, 0, 0, 2, 1, 0, 2, 1, 2, 0]
```

8. Gamma input: python3 simulateDist.py 3 gamma 3 3

Gamma output:

```
0.28146374380559
0.7479618695421953
1.2501446188593122
```

9. Poisson input: python3 simulateDist.py 10 poisson 5

Poisson output:

```
[4, 3, 3, 6, 5, 2, 8, 5, 5, 3]
```

10. Normal input: python3 simulateDist.py 3 normal 5 10

Normal output:

```
[8.664703252126255, 19.344492984764102, 5.583788779725449, -13.43201211075938]
```