

Introduction to Programming and Data Structures

Python – Randomization

Malay Bhattacharyya

Associate Professor

MIU, CAIML, TIH
Indian Statistical Institute, Kolkata

September, 2024



1 Randomization

Shuffling of data

```
import random
ls = [10, 20, 30, 40, 50]
random.shuffle(ls)
print(ls)
```

Output: [40, 20, 10, 50, 30]

Randomization of data

```
import random
random.randint(1, 100) # The interval is [1, 100]
```

Output: 15

```
import random
random.random() # The interval is [0.0, 1.0]
```

Output: 0.4289859005273219

Random sampling with numpy

```
import numpy as np
np.random.randint(1, 100) # From uniform distribution
```

Output: 97

The interval is [1, 100)

Random sampling with numpy

```
import numpy as np
np.random.randint(1, 100) # From uniform distribution
```

Output: 97

The interval is [1, 100)

```
import numpy as np
np.random.random(5) # From uniform distribution
```

Output: array([0.37076725, 0.10547203, 0.21298417, 0.96296838,
0.15390583])

The interval is [0.0, 1.0)

Random sampling with numpy

```
import numpy as np
np.random.randint(1, 100) # From uniform distribution
```

Output: 97 The interval is [1, 100)

```
import numpy as np
np.random.random(5) # From uniform distribution
```

Output: array([0.37076725, 0.10547203, 0.21298417, 0.96296838, 0.15390583]) The interval is [0.0, 1.0)

Note: Unlike the `random()` function in `random` module, the one in the `random` submodule of `numpy` module optionally takes size of the array.

Random sampling with numpy

```
import numpy as np
np.random.randn(5) # From normal distribution
```

Output: array([1.62029576, 0.29112406, 1.21198839,
0.26851418, -0.46712281])

```
import numpy as np
np.random.randn(3, 3) # From normal distribution
```

Output: array([[1.13217437, 0.56093212, -2.36792091],
[1.42652606, 0.68953983, 0.521175],
[1.36766496, 0.9488446 , -1.10042531]])

Effect of the seed value on random sampling

Code 1:

```
import numpy as np
print(np.random.randint(1, 100))
print(np.random.randint(1, 100))
print(np.random.randint(1, 100))
```

Effect of the seed value on random sampling

Code 1:

```
import numpy as np
print(np.random.randint(1, 100))
print(np.random.randint(1, 100))
print(np.random.randint(1, 100))
```

Code 2:

```
import numpy as np
np.random.seed(20)
print(np.random.randint(1, 100))
print(np.random.randint(1, 100))
print(np.random.randint(1, 100))
```

Effect of the seed value on random sampling

Code 1:

```
import numpy as np
print(np.random.randint(1, 100))
print(np.random.randint(1, 100))
print(np.random.randint(1, 100))
```

Code 2:

```
import numpy as np
np.random.seed(20)
print(np.random.randint(1, 100))
print(np.random.randint(1, 100))
print(np.random.randint(1, 100))
```

Note: Unlike Code 1, Code 2 will generate the same set of random values whenever executed.

Drawing samples from different distributions using numpy

<code>beta(a, b[, size])</code>	– Samples from Beta distribution
<code>binomial(n, p[, size])</code>	– Samples from binomial distribution
<code>chisquare(df[, size])</code>	– Samples from chi-square distribution
<code>dirichlet(alpha[, size])</code>	– Samples from Dirichlet distribution
<code>exponential([scale, size])</code>	– Samples from exponential distribution
<code>f(dfnum, dfden[, size])</code>	– Samples from F distribution
<code>gamma(shape[, scale, size])</code>	– Samples from Gamma distribution
<code>geometric(p[, size])</code>	– Samples from geometric distribution
<code>gumbel([loc, scale, size])</code>	– Samples from Gumbel distribution

Table: Functions in `random` submodule of `numpy`

Drawing samples from different distributions using numpy

<code>hypergeometric(ngood, nbad, nsample[, size])</code>	– Samples from hypergeometric distribution
<code>laplace([loc, scale, size])</code>	– Samples from Laplace distribution
<code>logistic([loc, scale, size])</code>	– Samples from logistic distribution
<code>lognormal([mean, sigma, size])</code>	– Samples from log-normal distribution
<code>logseries(p[, size])</code>	– Samples from logarithmic series distribution

Table: Functions in `random` submodule of `numpy`

Drawing samples from different distributions using numpy

<code>multinomial(n, pvals[, size])</code>	– Samples from multinomial distribution
<code>multivariate_normal(mean, cov[, size, ...])</code>	– Samples from multivariate normal distribution
<code>negative_binomial(n, p[, size])</code>	– Samples from negative binomial distribution
<code>noncentral_chisquare(df, nonc[, size])</code>	– Samples from noncentral chi-square distribution
<code>noncentral_f(dfnum, dfden, nonc[, size])</code>	– Samples from noncentral F distribution
<code>normal([loc, scale, size])</code>	– Samples from normal distribution

Table: Functions in `random` submodule of `numpy`

Drawing samples from different distributions using numpy

<code>pareto(a[, size])</code>	– Samples from pareto distribution
<code>poisson([lam, size])</code>	– Samples from Poisson distribution
<code>power(a[, size])</code>	– Samples from power distribution
<code>rayleigh([scale, size])</code>	– Samples from Rayleigh distribution
<code>standard_cauchy([size])</code>	– Samples from standard Cauchy distribution
<code>standard_exponential([size])</code>	– Samples from standard exponential distribution
<code>standard_gamma(shape[, size])</code>	– Samples from standard Gamma distribution
<code>standard_normal([size])</code>	– Samples from standard normal distribution

Table: Functions in `random` submodule of `numpy`

Drawing samples from different distributions using numpy

<code>standard_t(df[, size])</code>	– Samples from standard Student's t distribution
<code>triangular(left, mode, right[, size])</code>	– Samples from triangular distribution
<code>uniform([low, high, size])</code>	– Samples from uniform distribution
<code>vonmises(mu, kappa[, size])</code>	– Samples from von Mises distribution
<code>wald(mean, scale[, size])</code>	– Samples from Wald distribution
<code>weibull(a[, size])</code>	– Samples from Weibull distribution
<code>zipf(a[, size])</code>	– Samples from Zipf distribution

Table: Functions in `random` submodule of `numpy`

Visualizing a distribution

```
import numpy as np
sample = np.random.poisson(10, 50000)
import matplotlib.pyplot as plt
plt.hist(sample, 10)
plt.show()
```

