Search the docs ...

numpy.random.RandomSt
numpy.random.beta
numpy.random.binomial
numpy.random.bytes
numpy.random.chisquare
numpy.random.choice
numpy.random.dirichlet
numpy.random.exponenti
numpy.random.f
numpy.random.gamma
numpy.random.geometric
numpy.random.get_state
numpy.random.gumbel
numpy.random.hypergeon
numpy.random.laplace
numpy.random.logistic
numpy.random.lognormal
numpy.random.logseries
numpy.random.multinomi
numpy.random.multivariat
numpy.random.negative_b

numpy.random.choice

random.choice(*a, size=None, replace=True, p=None*)

Generates a random sample from a given 1-D array

New in version 1.7.0.

> **Note**
>
> New code should use the `choice` method of a `default_rng()` instance instead; please see the Quick Start.

Parameters: a : *1-D array-like or int*
If an ndarray, a random sample is generated from its elements. If an int, the random sample is generated as if it were `np.arange(a)`

size : *int or tuple of ints, optional*
Output shape. If the given shape is, e.g., `(m, n, k)`, then `m * n * k` samples are drawn. Default is None, in which case a single value is returned.

replace : *boolean, optional*
Whether the sample is with or without replacement. Default is True, meaning that a value of `a` can be selected multiple times.

p : *1-D array-like, optional*
The probabilities associated with each entry in a. If not given, the sample assumes a uniform distribution over all entries in `a`.

Returns: samples : *single item or ndarray*
The generated random samples

Raises: ValueError
If a is an int and less than zero, if a or p are not 1-dimensional, if a is an array-like of size 0, if p is not a vector of probabilities, if a and p have different lengths, or if replace=False and the sample size is greater than the population size

> **See also**
>
> **randint**, **shuffle**, **permutation**
> **Generator.choice**
> which should be used in new code

Notes

Setting user-specified probabilities through `p` uses a more general but less efficient sampler than the default. The general sampler produces a different sample than the optimized sampler even if each element of `p` is 1 / len(a).

Sampling random rows from a 2-D array is not possible with this function, but is possible with **`Generator.choice`** through its `axis` keyword.

Examples

Generate a uniform random sample from np.arange(5) of size 3:

```
>>> np.random.choice(5, 3)
array([0, 3, 4]) # random
>>> #This is equivalent to np.random.randint(0,5,3)
```

Generate a non-uniform random sample from np.arange(5) of size 3:

```
>>> np.random.choice(5, 3, p=[0.1, 0, 0.3, 0.6, 0])
array([3, 3, 0]) # random
```

Generate a uniform random sample from np.arange(5) of size 3 without replacement:

```
>>> np.random.choice(5, 3, replace=False)
array([3,1,0]) # random
>>> #This is equivalent to np.random.permutation(np.arange(5))[:3]
```

Generate a non-uniform random sample from np.arange(5) of size 3 without replacement:

```
>>> np.random.choice(5, 3, replace=False, p=[0.1, 0, 0.3, 0.6, 0])
array([2, 3, 0]) # random
```

Any of the above can be repeated with an arbitrary array-like instead of just integers. For instance:

```
>>> aa_milne_arr = ['pooh', 'rabbit', 'piglet', 'Christopher']
>>> np.random.choice(aa_milne_arr, 5, p=[0.5, 0.1, 0.1, 0.3])
array(['pooh', 'pooh', 'pooh', 'Christopher', 'piglet'], # random
      dtype='<U11')
```

Previous
numpy.random.chisquare

Next
numpy.random.dirichlet