[set random seed]

>> np.random.seed(<seed>=None)

* <seed>: int or 1-d array-like object. The seed of all random function

( if <seed> = None, it will not set the random seed)

[sample elements in a container]

>> np.random.choice(<a>, <size>=None, <replace>=True, <p>=None)

* <a>: if type is 1-D array (list, tuple, ndarray), <a> the choice pool

: if type is int, the choice pool is np.arange(<a>)

* <size>: the output size, or the output shape  
  (if size = None, return type is float  
  if size = <int>, return type is ndarray with shape = [<int>],  
  if size = <list>, return type is ndarray with shape = <list>)
* <replace>: boolean, whether the sample will be return each time
* <p>: the pmf for corresponding entry.   
  ( if <p> = None, then sample the entry by uniform distribution)

[sample a ramdom continuous number]

* uniform distribution

>> np.random.uniform(<low>=0, <high>=1, <size>=None)

* <low>: the lower bound
* <high>: the upper bound
* <size>: the output size, or the output shape
* normal distrinution

>> np.random.normal(loc=0, scale=1, size=None)

* loc: the mean
* scale: the standard deviation
* size: the output size, or the output shape

[random permutation]

>> np.random.permutation(<x>):

* <x>: if type is array-like (list, tuple, ndarray), <x> is the pool

: if type is int, the pool is np.arange(<x>)

[randomly shuffle]

>> np.random.shuffle(<arr>)

* <arr> : array\_like object. The object need to be shuffle
* NOTICE: it will shuffle the <array> inplace, so it will change the input array