Create random numbers

The `random` subpackage of `numpy` can be used to generate random numbers:

- float values in $[0, 1[$
- integers

In [1]:
```
import numpy.random as npr
```

In [2]:
```
npr.random((2, 5))
```

Out[2]:
```
array([[0.38477226, 0.81225982, 0.36261269, 0.93270438, 0.99471903],
       [0.96847033, 0.31830317, 0.84150116, 0.3289386 , 0.92886831]])
```

In [3]:
```
npr.randint(1, 10, (8, 3))   # (8, 3) integers in [1, 10[ (10 excluded)
```

Out[3]:
```
array([[9, 6, 4],
       [7, 4, 6],
       [9, 4, 7],
       [2, 1, 9],
       [4, 6, 4],
       [8, 6, 3],
       [9, 7, 7],
       [2, 4, 7]])
```

Random numbers can also be generated following specific statistical distribution:

In [4]:
```
print(npr.uniform(0, 5, (3, 4)))                    # uniform probability of a number in
                                                    # with shape (3, 4)
print(npr.normal(loc=0, scale=5, size=(3, 4)))      # normal probability with mu=0 and s
                                                    # with shape=(3, 4)
```

```
[[2.41357887 0.82780798 1.42760635 1.72300491]
 [0.73441211 3.23757418 2.83133007 2.74198725]
 [2.39981143 0.50201135 4.66832312 2.1018886 ]]
[[ 1.5065057   2.56094143 -8.18036362 -2.56765554]
 [ 3.77544338 10.03128866 -1.69567199 -5.75076198]
 [-2.84877235 -5.60960816  1.45432011 -2.16071159]]
```

Create deterministic random

The scientific approach needs reproducible computation steps. Whenever these steps imply random number generation, this can leads to problems: values differ from one execution to another.

In [5]:
```
my_physical_variable = npr.random(4)
print(my_physical_variable)
```

```
[0.75796379 0.23139233 0.92266523 0.54818521]
```

In [6]:
```
my_physical_variable = npr.random(4)
print(my_physical_variable)
```

```
[0.50611365 0.72792523 0.1837617  0.64778748]
```

Fortunately, **one can create reproducible random**, i.e. a way to get the same random values whenever the code is ran.

To this purpose, one must define a random number generator and **initialize it** with the same initial *state* for all executions:

In [7]:
```
rng = npr.default_rng(42)
my_physical_variable = rng.random(4)
print(my_physical_variable)
```

```
[0.77395605 0.43887844 0.85859792 0.69736803]
```

In [8]:
```
rng_new = npr.default_rng(42)
my_physical_variable = rng_new.random(4)
print(my_physical_variable)
```

```
[0.77395605 0.43887844 0.85859792 0.69736803]
```

In [9]:
```
rng_new = npr.default_rng(65)      # different seed
my_physical_variable = rng_new.random(4)
print(my_physical_variable)
```

```
[0.04739149 0.51822218 0.37485856 0.22867852]
```

Note that:

- defining the initial state returns a new instance that must be used to generate random numbers
- using this instance provides reproducible random numbers generation
- a different initial state gives different random numbers