PSYC 027: Scientific Computing for Psychology

24 October 2019

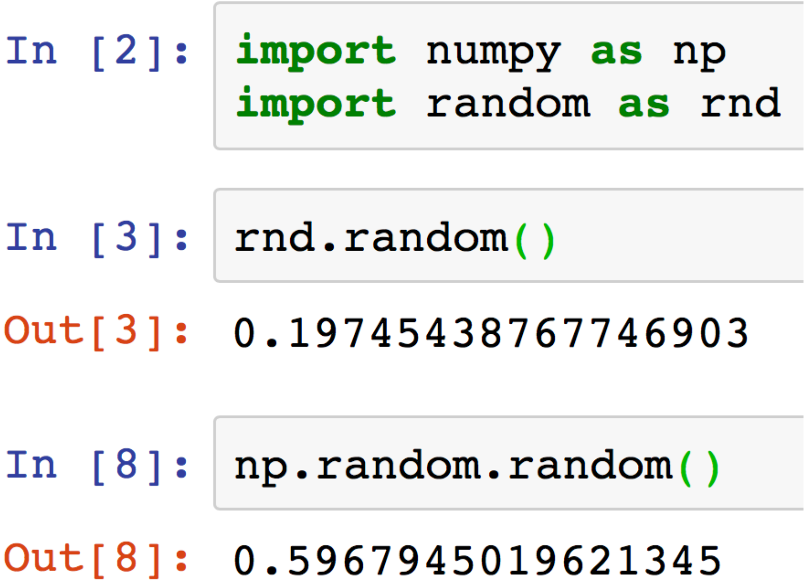
Professor Youssef Ezzyat
McCabe Library 306
T/Th 9:55-11:10

Analyzing free and categorized recall

Random Numbers

- Two libraries for generating random numbers:
  - Numpy
  - Random

- Both do basically the same thing, Numpy's is a little better for generating arrays of multiple random numbers

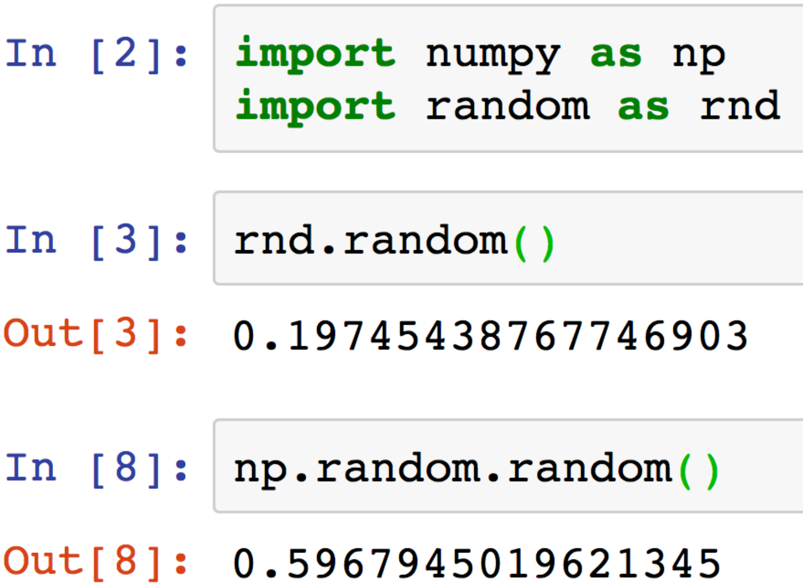


```
In [2]: import numpy as np
        import random as rnd

In [3]: rnd.random()

Out[3]: 0.1974543876774690З
```

```
In [8]: np.random.random()

Out[8]: 0.5967945019621345
```

Random Numbers

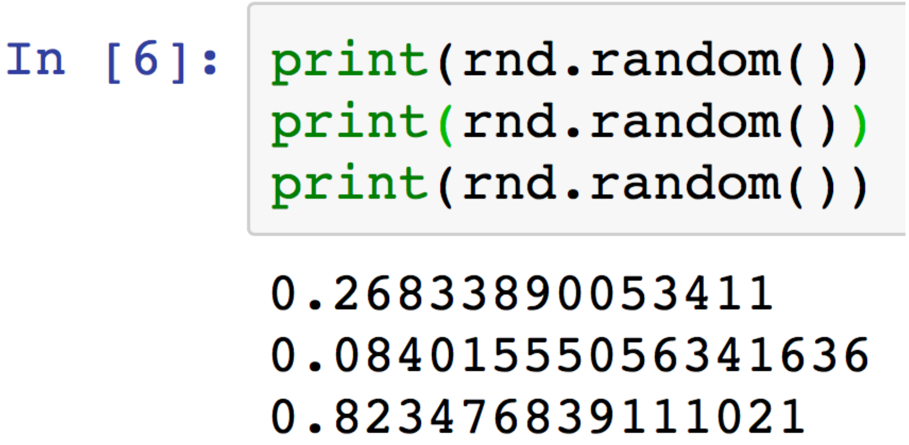
- `rnd.random()` and `np.random.random()` each return a random number between 0 and 1 drawn from a uniform distribution



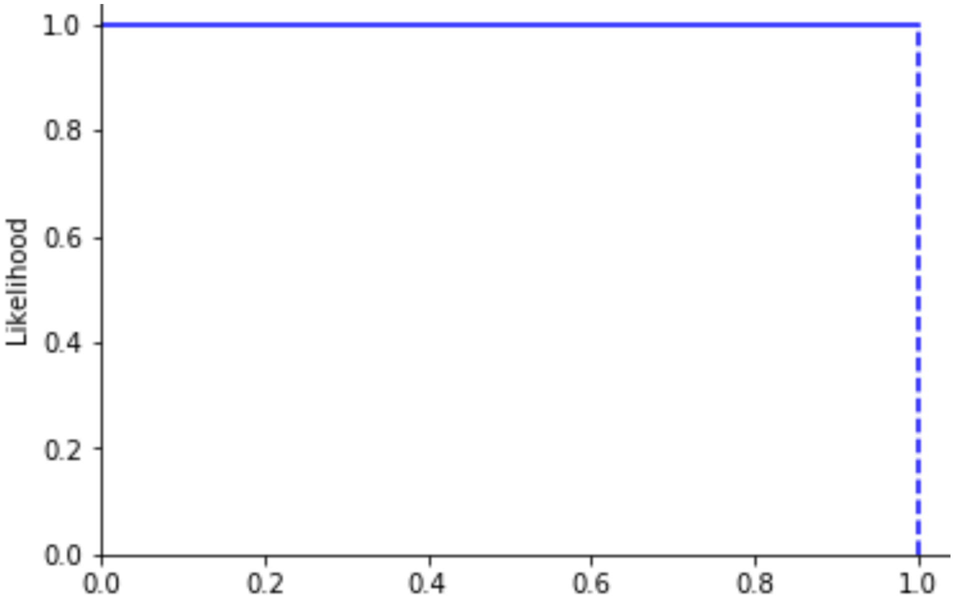
```
In [2]:  import numpy as np
         import random as rnd

In [3]:  rnd.random()

Out[3]:  0.19745438767746903

In [8]:  np.random.random()

Out[8]:  0.5967945019621345
```

Random Numbers

- Repeated calls to these functions produce more of these random numbers



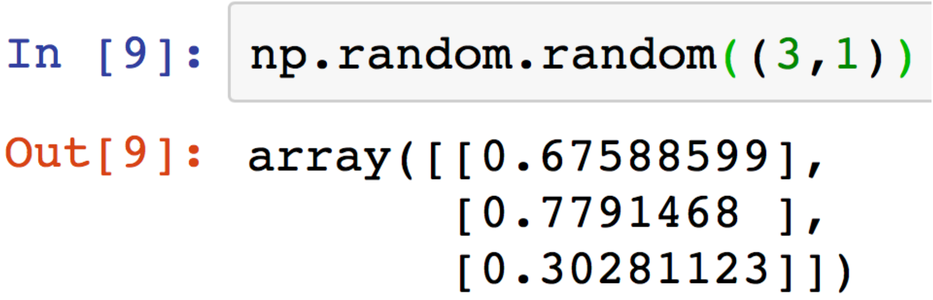
```
In [6]: print(rnd.random())
        print(rnd.random())
        print(rnd.random())

        0.26833890053411
        0.0840155505634163 6
        0.82347683911102 1
```



Uniform Distribution

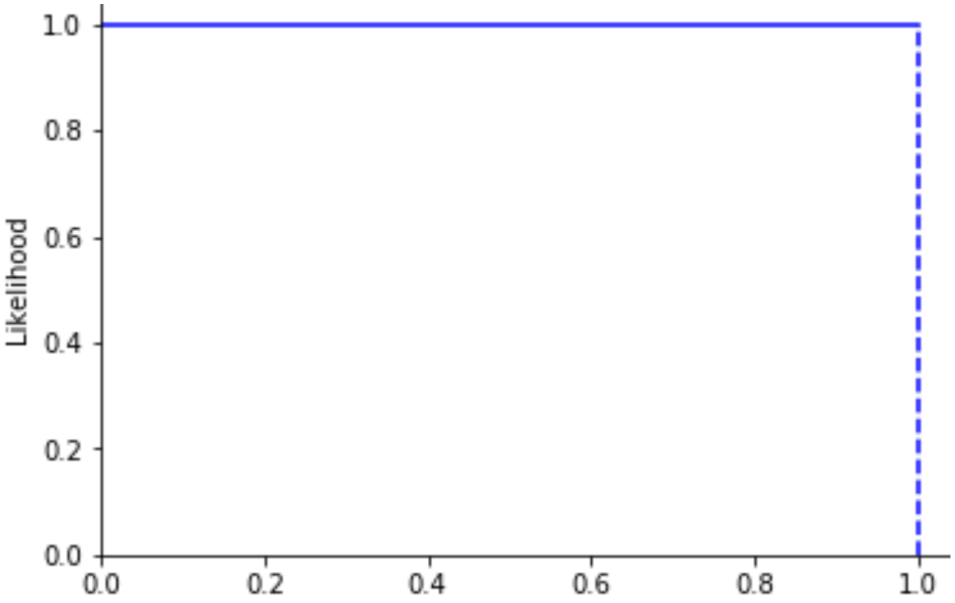
Random Numbers

- You can get an array of random numbers within a single call to the numpy version



```
In [9]: np.random.random((3,1))

Out[9]: array([[0.67588599],
               [0.7791468 ],
               [0.30281123]])
```

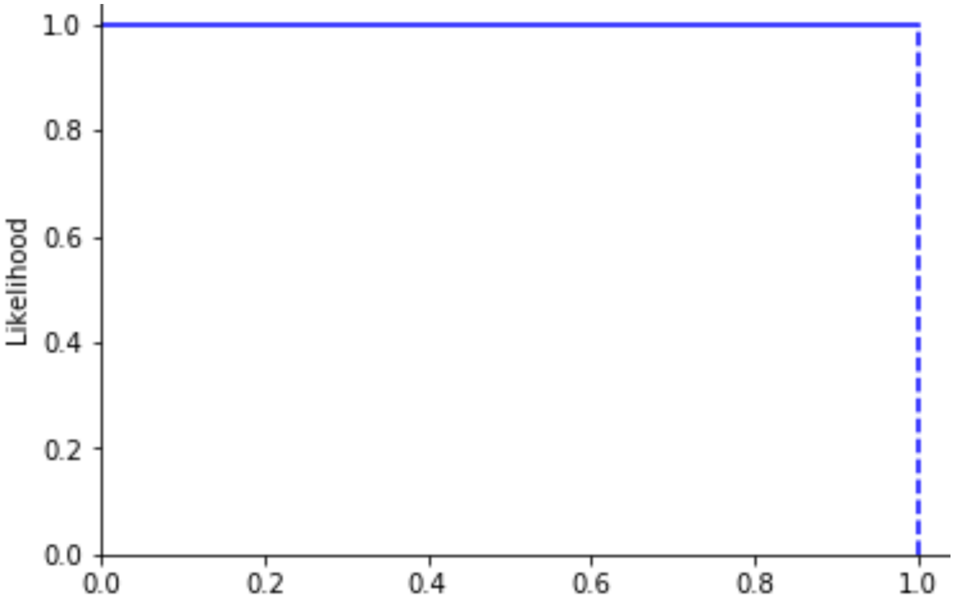
Uniform Distribution



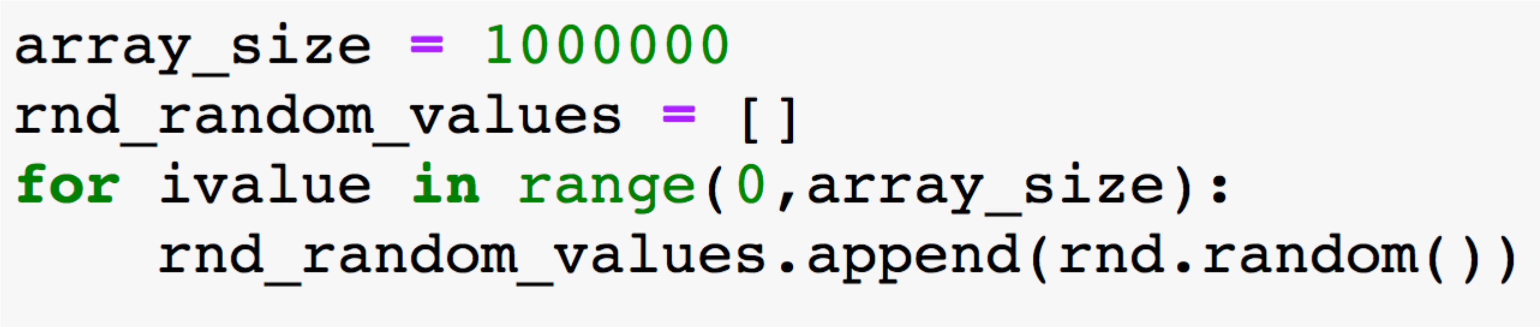
Random Numbers

- How could we confirm for ourselves (empirically) that these random number generators are drawing values from a uniform distribution?
  - rnd.random()?
  - np.random.random?

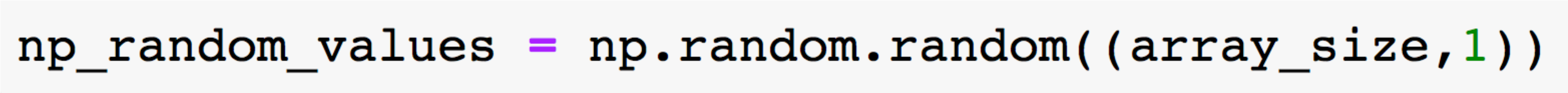
Uniform Distribution



Random Numbers

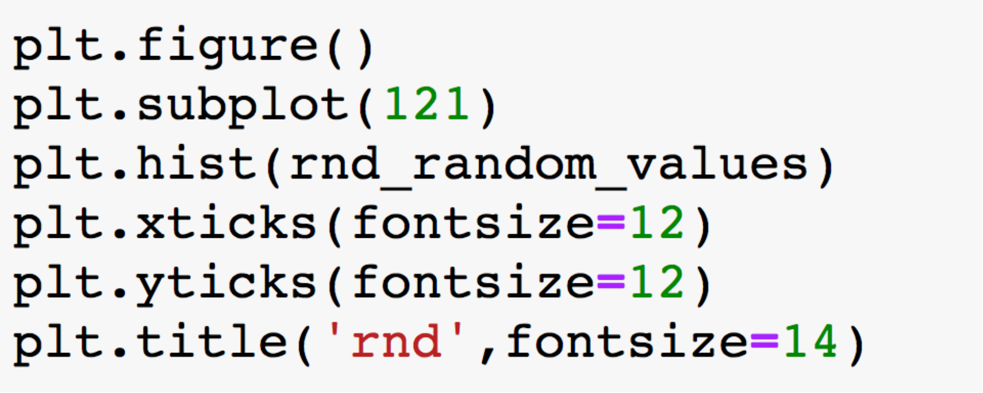


```python
array_size = 1000000
rnd_random_values = []
for ivalue in range(0,array_size):
    rnd_random_values.append(rnd.random())
```

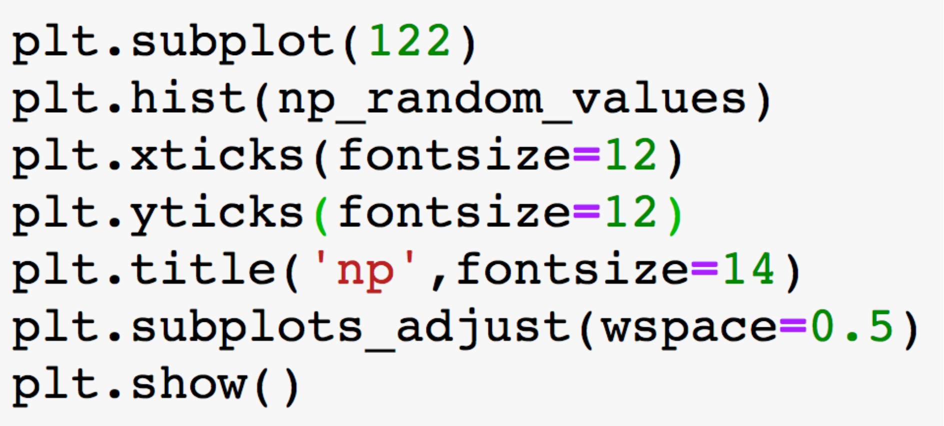


```python
np_random_values = np.random.random((array_size,1))
```

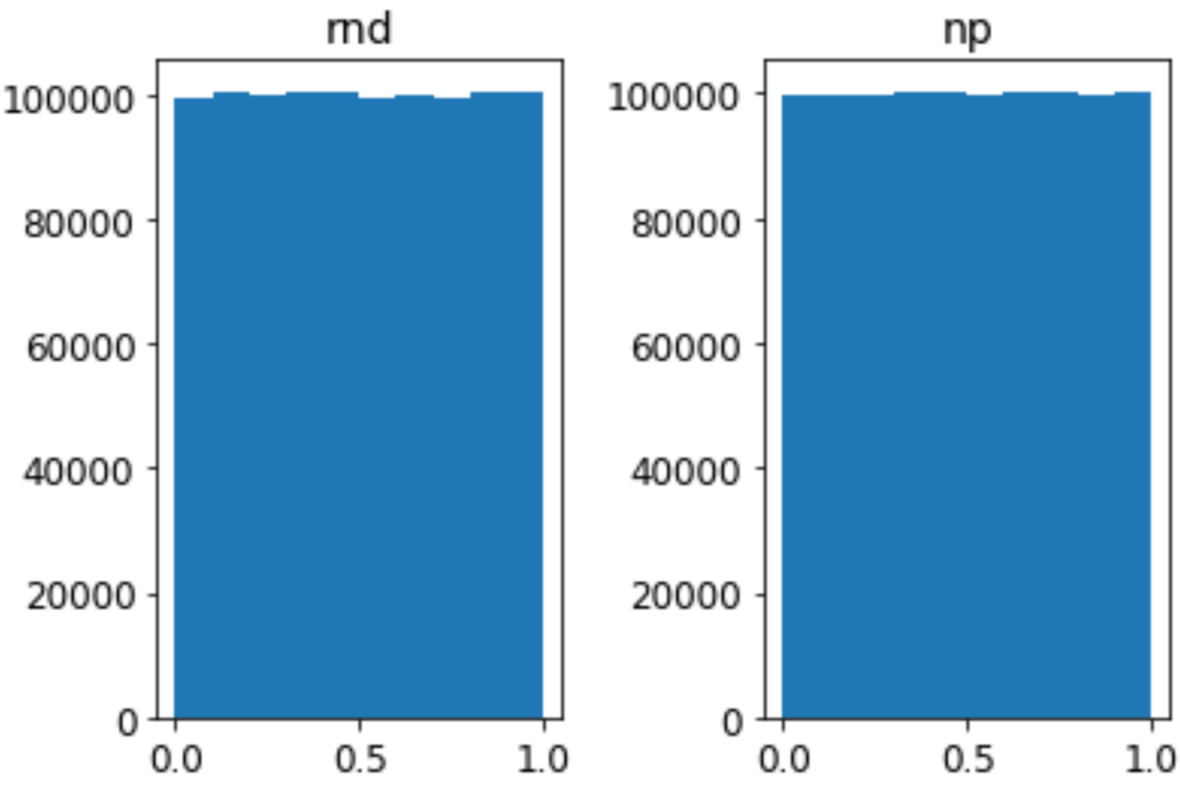
Random Numbers



```
plt.figure()
plt.subplot(121)
plt.hist(rnd_random_values)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.title('rnd',fontsize=14)
```

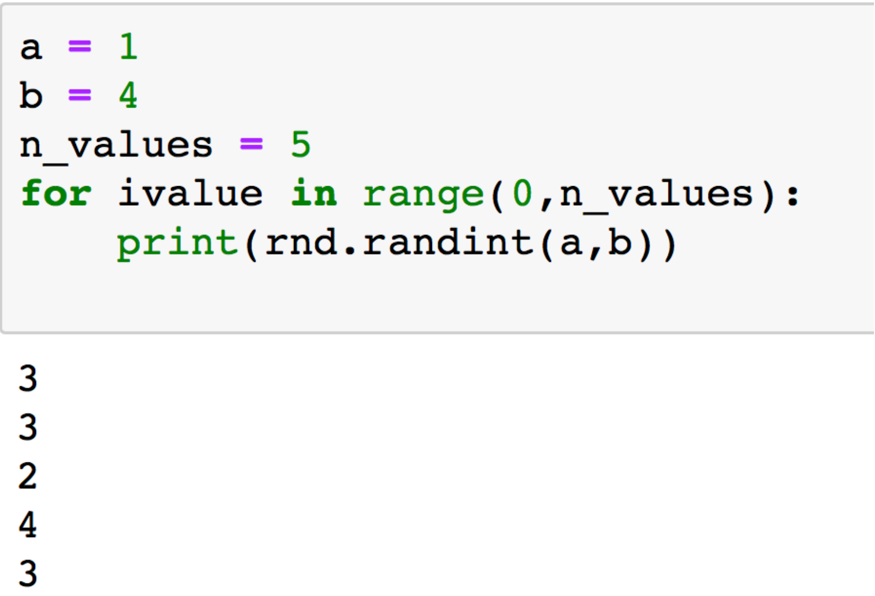


```
plt.subplot(122)
plt.hist(np_random_values)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.title('np',fontsize=14)
plt.subplots_adjust(wspace=0.5)
plt.show()
```



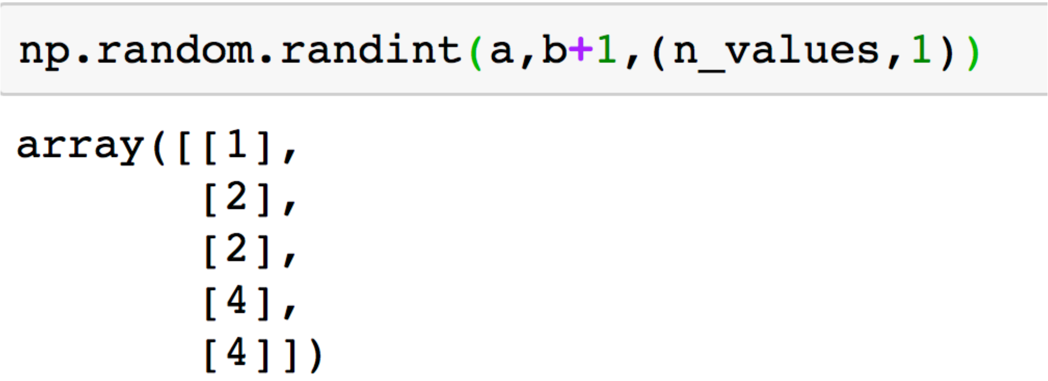
Random Numbers

- There are also functions to generate random integers



```
a = 1
b = 4
n_values = 5
for ivalue in range(0,n_values):
    print(rnd.randint(a,b))
```

```
3
3
2
4
3
```



```
np.random.randint(a,b+1,(n_values,1))
```

```
array([[1],
       [2],
       [2],
       [4],
       [4]])
```

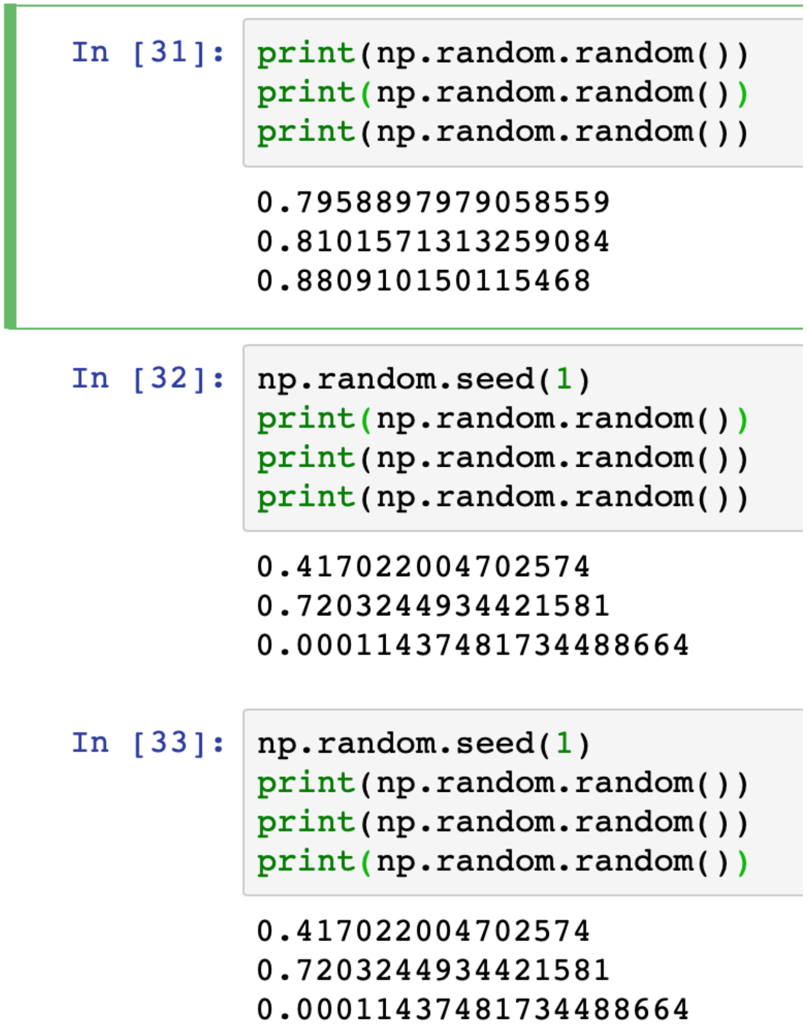
Random Numbers

- Random number generators are not actually random, but rely on algorithms that generate pseudorandom numbers

- The algorithms rely on the value of an initial **seed**

- Based on the value of the seed, the algorithm can generate sequences of numbers that appear random (but are actually deterministic)

Random Numbers



In [31]:
```
print(np.random.random())
print(np.random.random())
print(np.random.random())
```

```
0.7958897979058559
0.8101571313259084
0.880910150115468
```

In [32]:
```
np.random.seed(1)
print(np.random.random())
print(np.random.random())
print(np.random.random())
```

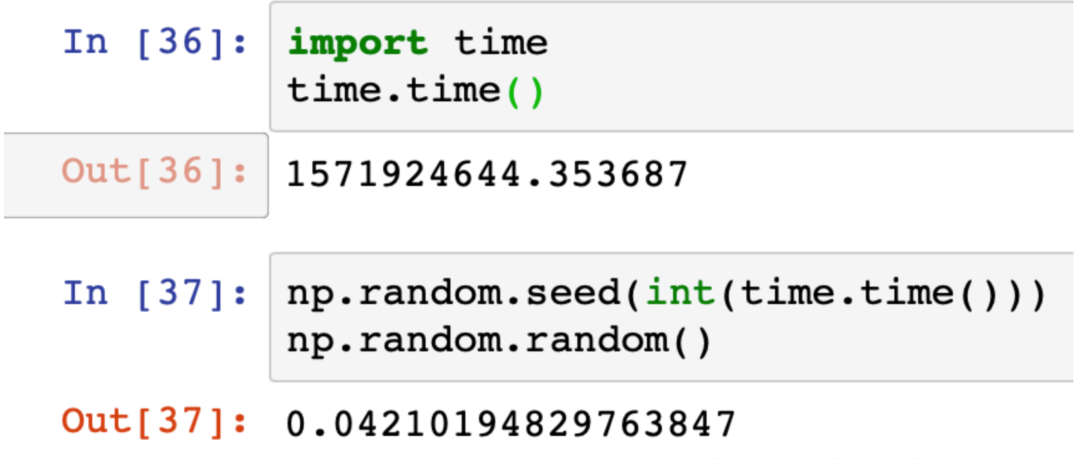
```
0.417022004702574
0.7203244934421581
0.00011437481734488664
```

In [33]:
```
np.random.seed(1)
print(np.random.random())
print(np.random.random())
print(np.random.random())
```

```
0.417022004702574
0.7203244934421581
0.00011437481734488664
```

Random Numbers

- You can seed the random number generator once with the current time at the start of your code



```
In [36]: import time
         time.time()

Out[36]: 1571924644.353687

In [37]: np.random.seed(int(time.time()))
         np.random.random()

Out[37]: 0.04210194829763847
```