

▼ Q. 파이썬 외장함수 random을 이용하여 아래사항들을 출력하세요.

- $0.0 \leq x < 1.0$ 사이의 float를 랜덤하게 선택
- 10에서 20사이의 정수중에서 난수값 선택
- 10보다 적은 int 선택
- 20과 30사이의 float 선택
- [6,7,8,9,10]리스트 내부에 있는 요소를 랜덤하게 선택
- [6,7,8,9,10]리스트 내부에 있는 요소중 k개 선택

```
1 import random
2
3 print (random.random())
4 print (random.randint(10, 20))
5 print (random.randrange(10))
6 print (random.uniform(20, 30))
7 print (random.choice([6,7,8,9,10]))
8 print (random.sample([6,7,8,9,10], k=2))
```

```
0.29852642093292314
12
5
21.89955645729318
6
[7, 9]
```

▼ Q. 주어진 데이터에서 랜덤으로 하나를 선택하여 꺼낸 다음 그 값을 출력하고 꺼내진 요소를 제외

data = [1,2,3,4,5,6,7,8,9,10]

```
1 data = [1,2,3,4,5,6,7,8,9,10]
2 random.shuffle(data)
3 print (data.pop())
4 print (sorted(data))
```

```
2
[1, 3, 4, 5, 6, 7, 8, 9, 10]
```

▼ Q. 넘파이 random을 이용하여 아래사항들을 출력하세요.

- 0부터 10까지 랜덤한 숫자 1개
- 0부터 1사이의 균일분포에서 난수 5개로 구성되는 배열 생성
- 평균 0, 표준편차 1의 가우시안 표준정규분포에서 난수 10개로 구성되는 배열 생성

```
1 import numpy as np
2 print (np.random.randint(1, 10))
3 print (np.random.rand(5))
4 print (np.random.randn(10))
```

```
9
[3.14217457e-04 7.68468347e-01 6.77429146e-01 4.95385723e-01
 8.91905984e-01]
[ 0.90552964 -0.74571398 -0.73277102  0.30040138  1.44176289  0.88948358
 1.23661487  0.93477923 -0.20275066  0.08665786]
```

▼ Q. 넘파이 random을 이용하여 아래사항들을 출력하세요.

- [6,7,8,9,10] 리스트의 데이터의 순서 바꾸기

```
1 arr = [6,7,8,9,10]
2 np.random.shuffle(arr)
3 print (arr)
```

```
☞ [9, 8, 6, 10, 7]
```

▼ Q. 넘파이 random을 이용하여 아래사항들을 출력하세요.

- 10보다 적은 정수에서 랜덤으로 중복되지 않게 5개를 선택 배열을 생성하세요.
- c1 = [6,7,8,9,10] 배열에서 랜덤으로 중복 선택이 가능하게 5개를 선택, 배열을 생성하세요.

```
1 print (np.random.choice(10, 5, replace=False))
2 c1 = [6,7,8,9,10]
3 print (np.random.choice(c1, 5))
```

```
☞ [6 3 4 9 0]
   [ 6  7 10  9  9]
```

▼ Q. 넘파이 random을 이용하여 2.0보다 크거나 3.0보다 작은 5개의 수를 출력하세요.

```
1 print (np.random.uniform(2.0, 3.0, 5))
```

```
☞ [2.78528131 2.94019464 2.30228088 2.08392709 2.98506673]
```

▼ Q. 넘파이 random을 이용하여 0.0보다 크거나 1.0보다 작은 (3,3) 2차원 배열을 2가지 방식으로

```
1 print (np.random.rand(3,3), '\n')
2 print (np.random.uniform(0.0,1.0,(3,3)), '\n')
3 print (np.random.random_sample((3,3)))
```

```
☞
```

```
[[0.12773351 0.9616301 0.39786691]
```

Q. 앞(head) 또는 뒤(tail) (n=1) 가 나올 확률이 각각 50%(p=0.5)인 동전 던지기를 20번(size=20) 요.

```
[0.3795929 0.56811987 0.90322109]
```

```
1 print (np.random.binomial(1, 0.5, size=20))
```

```
[1 0 1 1 1 0 1 0 1 1 1 1 1 1 0 1 1 1 1 0]
```

```
rn 31940104 n 49025229 n 9371230211
```

Q. (2,3,4) 3차원 배열 형태로 정규분포(np.random.normal)로 부터 무작위 샘플을 생성

```
1 print (np.random.normal(0.0, size=(2, 3, 4)))
```

```
[[-0.98563329 0.79109744 1.32862223 -0.46211299]
 [-0.19897257 -0.45616672 -0.37004468 -0.40040919]
 [ 0.28814635 -3.08512496 -0.6199405 -0.01724624]]
```

```
[[-1.11105286 0.64117183 0.13284961 0.15814423]
 [-0.77106216 -0.6751964 1.23089826 -0.67129015]
 [ 0.30528408 -0.93996078 -0.03648123 -0.06028254]]]
```

Q. [4 7 7 1 9]에서 중복되지 않는 원소 배열과 각 원소의 중복 개수 배열을 출력하세요.

```
1 qlist = [4,7,7,1,9]
2 alist, count = np.unique(qlist, return_counts=True)
3 print (alist)
4 print (count)
```

```
[1 4 7 9]
[1 1 2 1]
```

Q. [1, 1, 2, 2, 2, 3]에서 0 ~ 5에 해당하는 정수의 개수 배열을 출력하세요.

```
1 qlist = [1, 1, 2, 2, 2, 3]
2 print (np.bincount(qlist, minlength=5))
```


```
[0 2 3 1 0]
```

Q(응용) 'ID', 'Prod', 'Price' 3개의 column과 10개의 row으로 구성되는 데이터셋을 아래를 참조하

- ID는 1 ~ 10으로 적용
- Prod는 상품코드로서 np.random을 이용하여 10보다 적은 정수를 랜덤하게 선택
- Price는 가격으로서 np.random을 이용하여 1000 보다 적은 양의 소수를 랜덤하게 선택
- 3개의 컬럼을 결합하는 방법은 concat() 함수를 사용

```
1 import pandas as pd
2
3 ds_id = pd.DataFrame(np.arange(1, 11))
4 # print (ds_id, '\n')
```

```
5
6 ds_Prod = pd.DataFrame([np.random.randint(10) for i in range(10)])
7 # print (ds_Prod, '\n')
8 ds_Price = pd.DataFrame([np.random.uniform(1000) for i in range(10)])
9 # print (ds_Price, '\n')
10
11 ds = pd.concat([ds_id, ds_Prod, ds_Price], axis=1)
12 print (ds)
```



	0	0	0
0	1	7	667.943683
1	2	3	481.310368
2	3	6	621.221988
3	4	3	665.953255
4	5	6	123.335624
5	6	2	950.797822
6	7	1	531.907532
7	8	7	158.328058
8	9	6	139.105337
9	10	5	496.211149