중요사항 정리

넘파이(Numpy)

리스트와 비슷한 기능을 하고 리스트보다 처리 속도가 훨씬 빠르다.
대용량의 배열과 행렬 연산이 가능하다.
넘파이의 핵심적인 개체는 다차원 배열, 각 요소는 인덱스, 차원은 축(axis)
벡터(1차원 배열) axis 0, 행렬matrix(2차원 배열) 세로축(axis 0) - 가로축(axis 1), Tensor(3차원 배열) 가로축(axis 0) - 세로축(axis 1) - 높이(axis 2) scalar는 인덱스가 하나인 경우(0차원)
import ~ as에서 as뒤에 나타나는 이름은 as 앞의 이름을 대체하는 별칭(alias)
넘파이 배열에는 수학적 연산자가 적용 가능하다.(브로드캐스팅)
a = np.array([1,2,3]), a = a+1 -> [2,3,4]
인덱싱과 슬라이싱이 가능하다.(논리적인 인덱싱 가능)

넘파이 속성

ndim: 배열 축 혹은 차원의 개수
shape: 배열의 차원 (m,n)형식의 튜플 형, m,n은 각 차원의 원소의 크기를 의미
size: 배열 원소의 개수, m * n
dtype: 배열내의 원소의 자료형을 기술
itemsize: 배열내의 원소의 크기를 바이트 단위로 기술

넘파이 함수

np.array(): 배열 생성
np.arange([start,]stop, [step]): range()함수와 같은 의미.
np.linspace(start, stop, num): 시작값부터 끝값까지 균일한 간격으로 지정된 개수만큼의 배열을 생성.
np.logspace(x, y, n): 로그 스케일로 수들을 생성. 10의 x승부터 10의 y승까지 n개의 수가 생성.
np.reshape(): 데이터의 개수는 유지하고 배열의 차원과 형태를 변경
np.flatten(): 고차원 배열을 1차원 배열로 만든다.

난수

np.random.seed(): 컴퓨터로 만드는 난수의 초기 입력값으로 사용
np.random.rand(a): 0과1 사이에서 랜덤으로 a만큼 수를 구한다.
np.random.randint(a,b): 정수 a와 정수 b사이의 난수를 생성하여 반환
np.random.randn(): 난수들의 평균은 0이고 표준편차는 1이 되는 표준정규분포가 되도록 한다.
np.mean(): 평균값
np.median(): 중앙값

상관관계

corrcoeff(x,y): 요소들의 상관관계를 계산한다. 두 요소들 사이의 관계가 얼마나 직선에 가까운 관계가 있는가를 나타낸다.

자기점검

numpy 자체를 새롭게 사용하다보니 함수들이나 슬라이싱 인덱싱 하는 것에 약간 어려움을 느껴서 반복적으로 공부하였다.

심화문제 풀기

10-1

In [3]:
```
import numpy as np
num_arr = np.array(range(1,21))
print(num_arr)
```

```
[ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20]
```

In [7]:
```
print(num_arr[::-1])
```

```
[20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1]
```

In [9]:
```
print('num_arr 내의 모든 원소의 합 :', sum(num_arr))
```

```
num_arr 내의 모든 원소의 합 : 210
```

In [11]:
```
print(num_arr.reshape(5,4))
```

```
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]
 [13 14 15 16]
 [17 18 19 20]]
```

10-2

In [12]:
```
import numpy as np
n_arr = np.array(range(0,25))
n_arr = n_arr.reshape(5,5)
print(n_arr)
```

```
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]
 [20 21 22 23 24]]
```

In [19]:
```
print('첫 원소: ', n_arr[0,0])
print('마지막 원소 :', n_arr[-1,-1])
```

```
첫 원소:  0
마지막 원소 : 24
```

In [25]:
```
print(n_arr[:2])
```

```
[[0 1 2 3 4]
 [5 6 7 8 9]]
```

In [27]:
```
print(n_arr[2:])
```

```
[[10 11 12 13 14]
 [15 16 17 18 19]
 [20 21 22 23 24]]
```

In [31]:
```
print(n_arr[:,::2])
```

```
[[ 0  2  4]
 [ 5  7  9]
 [10 12 14]
 [15 17 19]
 [20 22 24]]
```

In [32]:
```
print(n_arr[::2, ::2])
```

```
[[ 0  2  4]
 [10 12 14]
 [20 22 24]]
```

In [35]:
```
n_arr = n_arr[:2]
n_arr = n_arr.reshape(5,-1)
print(n_arr)
```

```
[[0 1]
 [2 3]
 [4 5]
 [6 7]
 [8 9]]
```

10-3

In [38]:
```
import numpy as np
np.random.seed(100)
a = np.random.rand(3,3,3)
print(a)
```

```
[[[0.54340494 0.27836939 0.42451759]
  [0.84477613 0.00471886 0.12156912]
  [0.67074908 0.82585276 0.13670659]]

 [[0.57509333 0.89132195 0.20920212]
  [0.18532822 0.10837689 0.21969749]
  [0.97862378 0.81168315 0.17194101]]

 [[0.81622475 0.27407375 0.43170418]
  [0.94002982 0.81764938 0.33611195]
  [0.17541045 0.37283205 0.00568851]]]
```

In [45]:
```
print('a의 원소들 중 최댓값: ', np.max(a))
```

```
a의 원소들 중 최댓값:  0.9786237847073697
```

In [46]:
```
print('a의 원소들 중 최댓값의 위치: ', np.argmax(a))
```

```
a의 원소들 중 최댓값의 위치:  15
```

10-4

In [56]:
```
import numpy as np
a = np.array([1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1]).reshape(5,5)
print(a)
```

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

In [57]:
```
print(a[:1] + 2)
```

```
[[3 2 3 2 3]]
```

10-5

In [79]:
```
import numpy as np
a = np.arange(0,32).reshape(4,4,2)
b = a.flatten()
print('10번째 원소 :', b[9])
print('20번째 원소 :', b[19])
```

```
10번째 원소 : 9
20번째 원소 : 19
```

10-6

In [84]:
```
x1 = [i for i in range(100)]
x2 = [i + np.random.randint(1,10) for i in range(100)]
x3 = [i + np.random.randint(1,50) for i in range(100)]
x4 = [np.random.randint(1,100) for i in range(100)]
result = np.corrcoef([x1, x2, x3, x4])
print(result)
print('x1과 가장 상관관계가 높은 리스트는 x2이고 낮은 리스트는 x4이다.')
```

```
[[ 1.          0.9962114   0.88635586 -0.08973075]
 [ 0.9962114   1.          0.88203841 -0.10189291]
 [ 0.88635586  0.88203841  1.         -0.06632723]
 [-0.08973075 -0.10189291 -0.06632723  1.        ]]
x1과 가장 상관관계가 높은 리스트는 x2이고 낮은 리스트는 x4이다.
```