서울사이버대학교 데이터리터러시 11주차 실습 코드

numpy의 1차원 자료형 1

- 1차원적인 데이터 형식
- 파이썬의 리스트와 유사하나, 벡터 연산이 가능함

In [1]:
```
import numpy as np
```

In [2]:
```
a = [10, 20, 30, 40, 50]
```

In [3]:
```
a
```

Out[3]:
```
[10, 20, 30, 40, 50]
```

In [4]:
```
type(a)
```

Out[4]:
```
list
```

In [5]:
```
b = np.array(a)
```

In [6]:
```
b
```

Out[6]:
```
array([10, 20, 30, 40, 50])
```

In [7]:
```
print(b)
```

```
[10 20 30 40 50]
```

In [8]:
```
b.shape
```

Out[8]:
```
(5,)
```

In [9]:
```
c = np.array([5, 4, 3, 2, 1])
```

In [10]:
```
c
```

Out[10]:
```
array([5, 4, 3, 2, 1])
```

In [11]:
```
b + c
```

Out[11]:
```
array([15, 24, 33, 42, 51])
```

In [12]:
```
b + 5
```

Out[12]:
```
array([15, 25, 35, 45, 55])
```

In [13]:
```
a + 5
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-13-5cd40e0bb6cf> in <module>
----> 1 a + 5

TypeError: can only concatenate list (not "int") to list
```

In [14]:
```
a + [5, 4, 3, 2, 1]
```

Out[14]: [10, 20, 30, 40, 50, 5, 4, 3, 2, 1]

numpy의 1차원 자료형 2

- 자주 사용하는 numpy 기본 함수
  - sum(): 합계
  - mean(): 평균
  - max(): 최댓값
  - min(): 최솟값
  - round(): 반올림
  - arange(): 리스트의 range()와 유사
  - zeros(): 0으로 이루어진 벡터
  - ones(): 1로 이루어진 벡터
  - copy(): 복사

In [15]:
```
np.sum(b)
```

Out[15]: 150

In [16]:
```
np.mean(b)
```

Out[16]: 30.0

In [17]:
```
np.max(b)
```

Out[17]: 50

In [18]:
```
np.min(b)
```

Out[18]: 10

In [19]:
```
c = b / 3
```

In [20]:
```
c
```

Out[20]: array([3.33333333, 6.66666667, 10.   , 13.33333333, 16.66666667])

In [21]:
```
np.round(c, 1)
```

Out[21]: array([3.3, 6.7, 10. , 13.3, 16.7])

In [22]:
```
np.round(c, 3)
```

Out[22]: array([3.333, 6.667, 10.   , 13.333, 16.667])

In [23]:
```
np.round(c, 0)
```

Out[23]: array([3., 7., 10., 13., 17.])

In [24]:
```
b[0]
```

Out[24]: 10

In [25]:
```
b[-1]
```

Out[25]: 50

In [26]:
```
b[1:3]
```

Out[26]: array([20, 30])

In [27]:
```
b[2:]
```

Out[27]: array([30, 40, 50])

In [28]:
```
np.sum(b[2:])
```

Out[28]: 120

In [29]:
```
np.mean(b[2:])
```

Out[29]: 40.0

In [30]:
```
b[1] = 200
```

In [31]:
```
b
```

Out[31]: array([10, 200, 30, 40, 50])

In [32]:
```
d = np.arange(5)
```

In [33]:
```
d
```

Out[33]: array([0, 1, 2, 3, 4])

In [34]:
```
e = np.arange(3, 8)
```

In [35]:
```
e
```

Out[35]: array([3, 4, 5, 6, 7])

In [36]:
```
f = np.arange(3, 8, 2)
```

In [37]:
```
f
```

Out[37]: array([3, 5, 7])

In [38]:
```
g = np.zeros(5)
```

In [39]:
```
g
```

Out[39]: array([0., 0., 0., 0., 0.])

In [40]:
```
h = np.ones(5)
```

In [41]:
```
h
```

Out[41]: array([1., 1., 1., 1., 1.])

In [42]:
```
b
```

Out[42]: array([10, 200, 30, 40, 50])

In [43]:
```
bb = b
```

In [44]:
```
bb
```

Out[44]: array([10, 200, 30, 40, 50])

In [45]:
```
bb[0] = 100
```

In [46]:
```
bb
```

Out[46]: array([100, 200, 30, 40, 50])

In [47]:
```
b
```

Out[47]: array([100, 200, 30, 40, 50])

In [48]:
```
b == bb
```

Out[48]: array([True, True, True, True, True])

In [49]:
```
bb = b.copy()
```

In [50]:
```
bb
```

Out[50]: array([100, 200, 30, 40, 50])

In [51]:
```
b[2] = 300
```

In [52]:
```
b
```

Out[52]: array([100, 200, 300, 40, 50])

In [53]:
```
bb
```

Out[53]: array([100, 200, 30, 40, 50])

In [54]:
```
b == bb
```

numpy의 2차원 자료형

- 2차원적인 데이터 형식
- 행렬 연산이 가능함

In [1]:
```
import numpy as np
```

In [2]:
```
a = np.array([[10, 20, 30, 40], [50, 60, 70, 80]])
```

In [3]:
```
a
```

Out[3]:
```
array([[10, 20, 30, 40],
       [50, 60, 70, 80]])
```

In [4]:
```
a.shape
```

Out[4]:
```
(2, 4)
```

In [5]:
```
b, c = a.shape
```

In [6]:
```
b
```

Out[6]:
```
2
```

In [7]:
```
c
```

Out[7]:
```
4
```

In [8]:
```
a[0, 0]
```

Out[8]:
```
10
```

In [9]:
```
a[-1, -1]
```

Out[9]:
```
80
```

In [10]:
```
a[1, 1:]
```

Out[10]:
```
array([60, 70, 80])
```

In [11]:
```
a
```

Out[11]:
```
array([[10, 20, 30, 40],
       [50, 60, 70, 80]])
```

In [12]:
```
a[0, 1] = 2
```

In [13]:
```
a
```

Out[13]:
```
array([[10,  2, 30, 40],
       [50, 60, 70, 80]])
```

In [14]:
```
b = np.zeros((2, 4))
```

In [15]:
```
b
```

Out[15]:
```
array([[0., 0., 0., 0.],
```

```
       [0., 0., 0., 0.]])
```

In [16]:
```
c = np.ones((3, 5))
```

In [17]:
```
c
```

Out[17]:
```
array([[1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1.]])
```

In [18]:
```
c = np.arange(8)
```

In [19]:
```
c
```

Out[19]:
```
array([0, 1, 2, 3, 4, 5, 6, 7])
```

In [20]:
```
c.reshape(2, 4)
```

Out[20]:
```
array([[0, 1, 2, 3],
       [4, 5, 6, 7]])
```

In [21]:
```
c
```

Out[21]:
```
array([0, 1, 2, 3, 4, 5, 6, 7])
```

In [22]:
```
c = c.reshape(2, 4)
```

In [23]:
```
c
```

Out[23]:
```
array([[0, 1, 2, 3],
       [4, 5, 6, 7]])
```

In [24]:
```
c = c.reshape(4, 2)
```

In [25]:
```
c
```

Out[25]:
```
array([[0, 1],
       [2, 3],
       [4, 5],
       [6, 7]])
```

In [26]:
```
c.sum()
```

Out[26]:
```
28
```

In [27]:
```
d = c.copy()
```

In [28]:
```
d[0, 0] = 10
```

In [29]:
```
d
```

Out[29]:
```
array([[10,  1],
       [ 2,  3],
       [ 4,  5],
       [ 6,  7]])
```

In [30]:
```
c
```

Out[30]:
```
array([[0, 1],
       [2, 3],
```

```
       [4, 5],
       [6, 7]])
```

In [31]:
```
c == d
```

Out[31]:
```
array([[False,  True],
       [ True,  True],
       [ True,  True],
       [ True,  True]])
```

In [32]:
```
e = np.arange(49).reshape(7, 7)
```

In [33]:
```
e
```

Out[33]:
```
array([[ 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, 25, 26, 27],
       [28, 29, 30, 31, 32, 33, 34],
       [35, 36, 37, 38, 39, 40, 41],
       [42, 43, 44, 45, 46, 47, 48]])
```

In [34]:
```
e[:, 1:3]
```

Out[34]:
```
array([[ 1,  2],
       [ 8,  9],
       [15, 16],
       [22, 23],
       [29, 30],
       [36, 37],
       [43, 44]])
```

In [35]:
```
e[3:, :4]
```

Out[35]:
```
array([[21, 22, 23, 24],
       [28, 29, 30, 31],
       [35, 36, 37, 38],
       [42, 43, 44, 45]])
```

In [36]:
```
e[3:, :4].sum()
```

Out[36]:
```
528
```

In [37]:
```
f = e[3:, :4].copy()
```

In [38]:
```
f
```

Out[38]:
```
array([[21, 22, 23, 24],
       [28, 29, 30, 31],
       [35, 36, 37, 38],
       [42, 43, 44, 45]])
```

In [39]:
```
e[4, 0] = 100
```

In [40]:
```
e
```

Out[40]:
```
array([[  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,  25,  26,  27],
       [100,  29,  30,  31,  32,  33,  34],
       [ 35,  36,  37,  38,  39,  40,  41],
       [ 42,  43,  44,  45,  46,  47,  48]])
```

In [41]:
```
f
```

Out[41]:
```
array([[21, 22, 23, 24],
       [28, 29, 30, 31],
       [35, 36, 37, 38],
       [42, 43, 44, 45]])
```

In [42]:
```
f.sum()
```

Out[42]:
```
528
```

numpy의 기본 연산과 조건 처리 1

- numpy의 기본 연산
  - 사칙연산 (+, -, *, /, //, %)
  - 스칼라곱, 행렬곱

In [1]:
```
import numpy as np
```

In [2]:
```
a = np.arange(5, 45, 5).reshape(2, 4)
```

In [3]:
```
b = np.arange(20, 36, 2).reshape(2, 4)
```

In [4]:
```
a
```

Out[4]:
```
array([[ 5, 10, 15, 20],
       [25, 30, 35, 40]])
```

In [5]:
```
b
```

Out[5]:
```
array([[20, 22, 24, 26],
       [28, 30, 32, 34]])
```

In [6]:
```
a + b
```

Out[6]:
```
array([[25, 32, 39, 46],
       [53, 60, 67, 74]])
```

In [7]:
```
a - b
```

Out[7]:
```
array([[-15, -12,  -9,  -6],
       [ -3,   0,   3,   6]])
```

In [8]:
```
b - a
```

Out[8]:
```
array([[15, 12,  9,  6],
       [ 3,  0, -3, -6]])
```

In [9]:
```
a * b
```

Out[9]:
```
array([[ 100,  220,  360,  520],
       [ 700,  900, 1120, 1360]])
```

In [10]:
```
a / b
```

Out[10]:
```
array([[0.25      , 0.45454545, 0.625     , 0.76923077],
       [0.89285714, 1.        , 1.09375   , 1.17647059]])
```

In [11]:
```
a // b
```

Out[11]:
```
array([[0, 0, 0, 0],
       [0, 1, 1, 1]], dtype=int32)
```

In [12]:
```
a % b
```

Out[12]:
```
array([[ 5, 10, 15, 20],
       [25,  0,  3,  6]], dtype=int32)
```

In [13]:
```
a * 10
```

Out[13]:
```
array([[ 50, 100, 150, 200],
       [250, 300, 350, 400]])
```

In [14]:
```
a
```

Out[14]:
```
array([[ 5, 10, 15, 20],
       [25, 30, 35, 40]])
```

In [15]:
```
c = a * 10
```

In [16]:
```
c
```

Out[16]:
```
array([[ 50, 100, 150, 200],
       [250, 300, 350, 400]])
```

In [17]:
```
b
```

Out[17]:
```
array([[20, 22, 24, 26],
       [28, 30, 32, 34]])
```

In [18]:
```
bb = b.reshape(4, 2)
```

In [19]:
```
bb
```

Out[19]:
```
array([[20, 22],
       [24, 26],
       [28, 30],
       [32, 34]])
```

In [20]:
```
a @ bb
```

Out[20]:
```
array([[1400, 1500],
       [3480, 3740]])
```

numpy의 기본 연산과 조건 처리 2

- numpy의 조건 처리

In [21]:
```
a = np.arange(1, 15)
```

In [22]:
```
a
```

Out[22]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14])
```

In [23]:
```
a > 10
```

Out[23]:
```
array([False, False, False, False, False, False, False, False, False,
       False,  True,  True,  True,  True])
```

In [24]:
```
a == 5
```

Out[24]:
```
array([False, False, False, False,  True, False, False, False, False,
       False, False, False, False, False])
```

In [25]:
```
a[a > 5]
```

Out[25]:
```
array([ 6,  7,  8,  9, 10, 11, 12, 13, 14])
```

In [26]:
```
a
```

Out[26]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14])
```

In [27]:
```
a[a < 5] = 5
```

In [28]:
```
a
```

Out[28]:
```
array([ 5,  5,  5,  5,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14])
```

In [29]:
```
a[a > 10] = 10
```

In [30]:
```
a
```

Out[30]:
```
array([ 5,  5,  5,  5,  5,  6,  7,  8,  9, 10, 10, 10, 10, 10])
```