

Python

시작하기

주의) ">>>"는 직접 타이핑 하지 않는다. 프롬프트를 의미한다.
'#'은 코멘트를 의미한다. 직접 입력할 필요는 없다.

```
>>> print('hello') >>> a = 'hello'
>>> print(a)
>>> b = 10
>>> c = 10.5
>>> print(b, c)
>>> d = [10, 20, 30] # list
>>> print(d)
>>> e = (10, 20, 30) # tuple
>>> print(e)
>>> print(type(d)) # list
>>> print(type(e)) # tuple
```

list (리스트) – 여러 element의 목록 (array)

* python의 list에서는 slice(콜론) 기능을 아는 것이 중요하다 !

```
>>> a = [0, 1, 2, 3, 4]
```

```
>>> print(a[0]) # 첫번째 방은 0번방. index는 0부터.
```

```
>>> print(a[-1]) # -1은 끝
```

```
>>> a.append(5) # element를 제일 뒤에 하나 추가.
```

```
>>> print(a)
```

```
>>> print(a[-1])
```

```
>>> print(a[-2]) # 끝에서 두번째
```

slice – 콜론(':')으로 index의 start,end를 지정함. end는 포함하지 않음

```
>>> print(a[1:3]) # 1번index(2번방)에서 3번index직전까지
```

```
>>> print(a[:3]) # start를 지정하지 않으면 처음부터.
```

```
>>> print(a[3:]) # end를 지정하지 않으면 끝까지.
```

```
>>> print(a[:-1]) # 처음부터 끝 직전까지.
```

```
>>> a.append('hello')
```

```
>>> print(a) * python의 list는 문자열, 숫자 등이 혼합되어 구성될 수 있다.
```

python 기본함수: **sum()**, **len()**

```
a = [4, 5, 6]
```

```
print(sum(a))    # 합계 구하기 → 15
```

```
print(len(a))    # 리스트길이(갯수) → 3
```

```
print(sum(a)/len(a))    # 평균값(mean) 구하기
```

for loop & range() – 가장 많이 쓰는 기능

* range()에도 list의 slice(콜론) 기능과 유사한 것이 있다.

```
>>> a = list(range(5)) # a = [0, 1, 2, 3, 4]와 유사한 효과
```

```
>>> for i in range(5): # for 루프
```

```
    print(i, 'Love') # for 라인 다음 라인 시작할 때 공백 2개 이상 주어야 함.
```

- range(*end*) – 전체 개수 의미. end 불포함.
- range(*start*, *end*) – start에서 end까지. end 불포함.
- range(*start*, *end*, *step*) – start에서 end까지. step만큼 간격주면서. end 불포함.

```
In [4]: print( list( range(5) ) )
```

```
[0, 1, 2, 3, 4]
```

```
In [3]: print( list( range(1,6) ) )
```

```
[1, 2, 3, 4, 5]
```

```
In [5]: print( list( range(1,6,2) ) )
```

```
[1, 3, 5]
```

String 다루기

주의) ">>>"는 직접 타이핑 하지 않는다. 프롬프트를 의미한다.

```
>>> s1 = 'aaa bbb ccc'
```

```
>>> s2 = 'AAA,BBB,CCC'
```

```
>>> s1.split()
```

```
['aaa', 'bbb', 'ccc']
```

```
>>> s2.split(',')
```

```
['AAA', 'BBB', 'CCC']
```

```
>>> s1.upper()
```

```
'AAA BBB CCC'
```

```
>>> s2.lower()
```

```
'aaa,bbb,ccc'
```

```
>>> s1.endswith('ccc')
```

```
True
```

```
>>> s2.startswith('ccc')
```

```
False
```

String – split(), join()

```
s = 'abc,def,ghi'
flds = s.split(',')
print(flds)
['abc', 'def', 'ghi']
```

```
s1 = '-'.join(flds)
print(s1)
'abc-def-ghi'
```

```
vlist = [1.234, 5.332342, 6.234324]
p = ['%.2f' % v for v in vlist]
print(', '.join(p))
```

tuple 다루기

(x,y) 같은 좌표데이터는 튜플(tuple)임. 함수를 호출할때 파라미터도 튜플임.

튜플에서는 unpack, pack 개념이해가 제일 중요함 !

튜플값을 여러개의 변수로 분리해서 받는 것을 unpack이라고 함.

```
>>> p= 10, 20  # 좌표 (10, 20)
```

```
>>> p[0]
```

```
10
```

```
>>> p[1]
```

```
20
```

```
#
```

```
>>> x, y = p    # unpacking
```

```
>>> print(x)    # 10
```

```
>>> print(y)    # 20
```

```
>>> pp = x,y    # packing
```

```
>>> print(type(pp), pp) # pp == (10, 20)
```


sort(), sorted() 차이

데이터 자기자신이 변하는가? 새로운 데이터가 생성되는가?

```
a = [9, 5, 6, 2]  
a.sort()  
print(a)
```

→
[2, 5, 6, 9]

```
b = [9, 5, 6, 2]  
sorted(b)  
print(b)
```

→
[9, 5, 6, 2]

```
b = sorted(b)  
print(b)
```

→
[2, 5, 6, 9]

List Comprehension

```
>>> a = range(10)
```

```
>>> [ x*x for x in a]
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

```
>>> [ x*x for x in a if x%2 == 1]
```

```
[1, 9, 25, 49, 81]
```

```
>>> [ x*x for x in a if x%2]
```

```
>>> [ x*x for x in a if not x%2]
```

List Comprehension

```
X = [i*0.1 for i in range(50)]
```

```
Y = [x*2 + 1 for x in X]
```

```
print(X)
```

```
print(Y)
```

zip()

a=[1,2,3]

b=[4,5,6]

list(zip(a, b))

[(1, 4), (2, 5), (3, 6)]

for x,y in zip(a,b):

print(x,y)

(2,12) ← zip

1	11
2	12
3	13
4	14
5	15
6	16
7	17
8	18
9	19

enumerate()

몇번째 데이터인지, 몇번째 루프인지 index값이 필요할 때 사용하면 좋다

```
ylist = [4,5,6]
```

```
for i,y in enumerate(ylist):
```

```
    print('[%02d] %d' % (i, y))
```

lambda – anonymous function

lambda 를 이용한 간단한 함수 정의

```
a = [1, 2, 3, 5, 10]
```

```
# inline, anonymous function ==> lambda
```

```
b2 = list(map(lambda x: x*x, a))
```

```
print(b2)
```

```
b3 = list(filter(lambda y: y > 50, b2))
```

```
print(b3)
```

List Comprehension & **numpy**

```
X = [i*0.1 for i in range(50)]
```

```
w = 2.5
```

```
b = 5
```

```
Y = [x*w + b for x in X]
```

```
print(Y)
```

```
import numpy as np
```

```
X = np.array(X)
```

```
Y = np.array(Y)
```

```
print(Y)
```

List Comprehension – Nested (중첩, for가 여러개)

```
>>> a = [1, 2, 3]
```

```
>>> b = [10, 20, 30]
```

```
>>> [i * j for i in a for j in b]
```

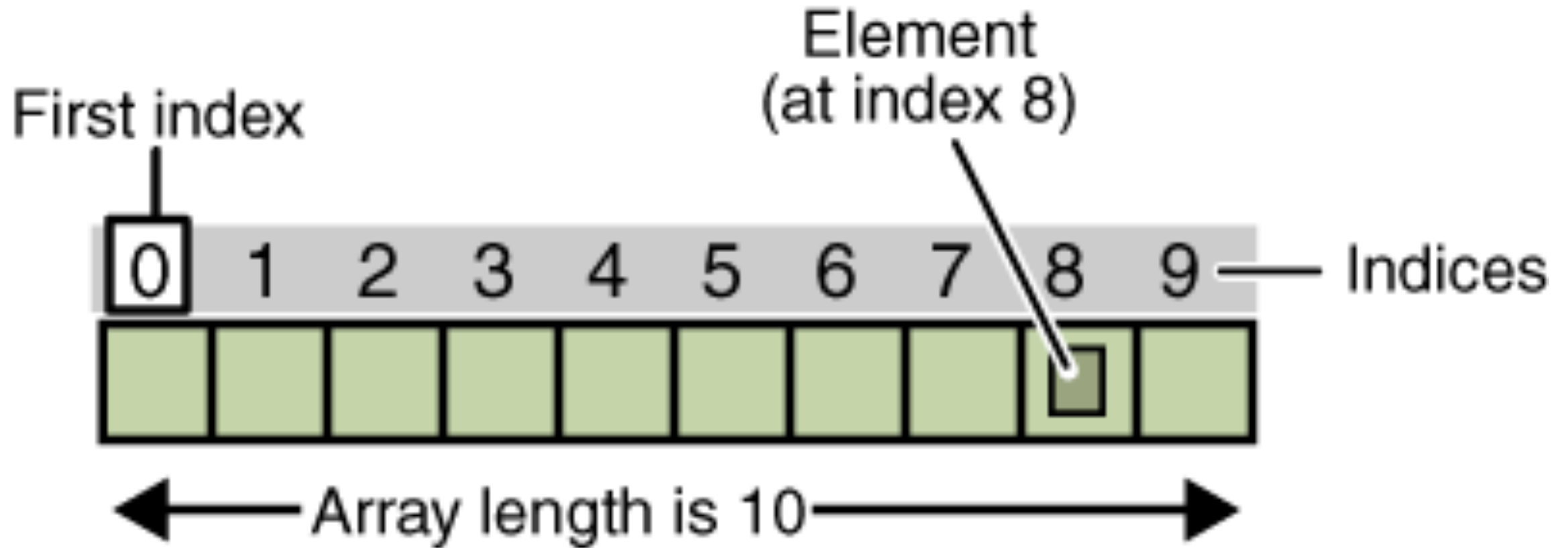
```
[10, 20, 30, 20, 40, 60, 30, 60, 90]
```

```
>>> ['i=%d, j=%d' % (i, j) for i in a for j in b]
```

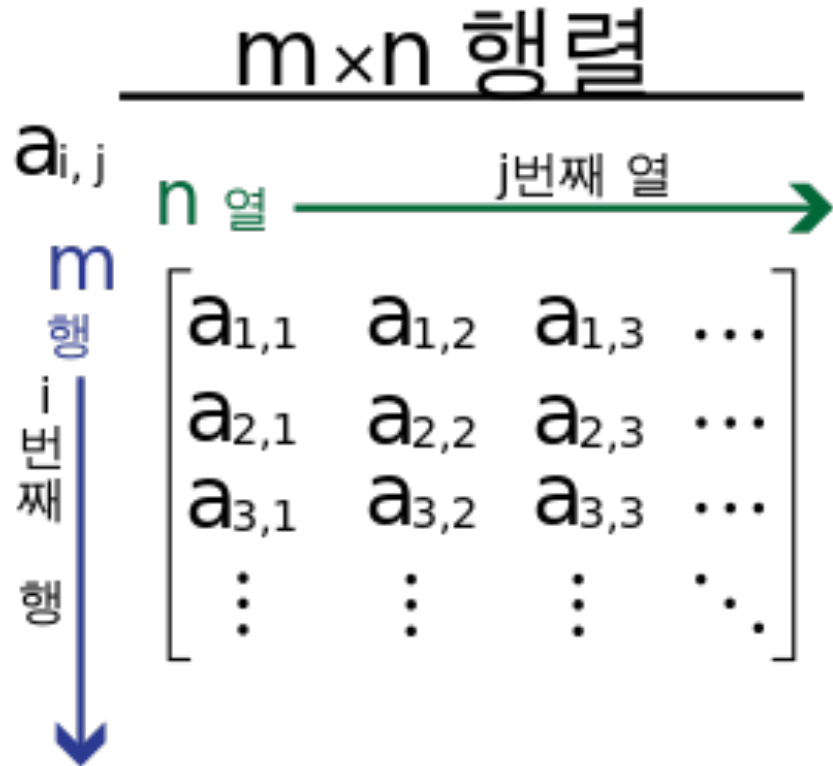
```
>>> ['i=%d, j=%d' % (i, j) for j in b for i in a ]
```


Array

Array (Sequece)



Matrix, 행렬

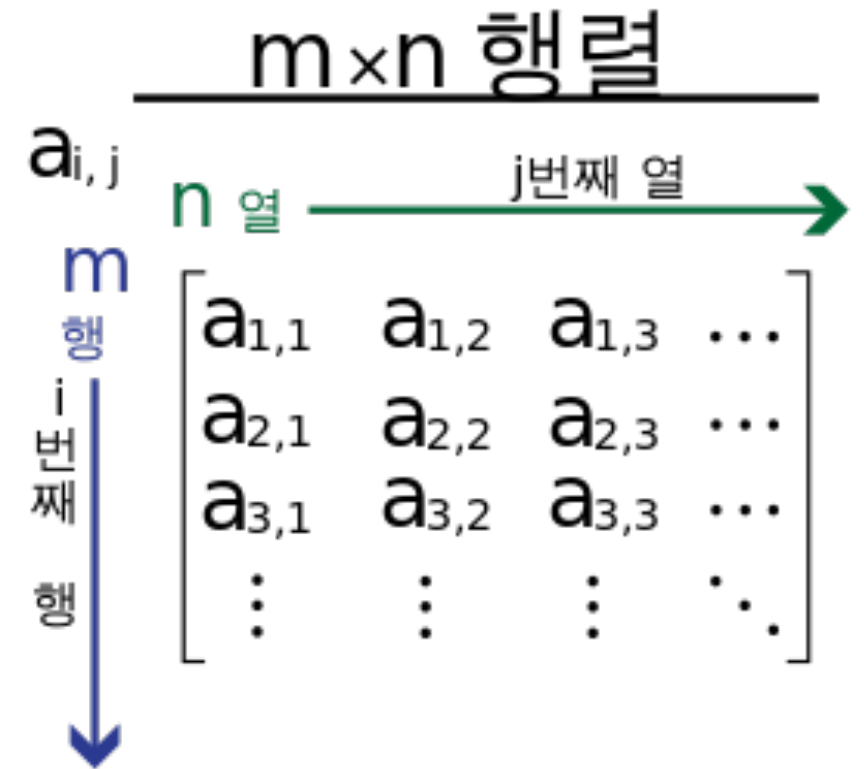


2 x 3 행렬

$$\begin{bmatrix} 1 & 9 & -13 \\ 20 & 5 & -16 \end{bmatrix}$$

Multi-dimensional Array

	Column 0	Column 1	Column 2	Column 3
Row 0	$a[0][0]$	$a[0][1]$	$a[0][2]$	$a[0][3]$
Row 1	$a[1][0]$	$a[1][1]$	$a[1][2]$	$a[1][3]$
Row 2	$a[2][0]$	$a[2][1]$	$a[2][2]$	$a[2][3]$



Python vs. Math

```
>>> a1=[1,9,-13]
>>> a2=[20,5,-16]
>>> A=[a1, a2]
>>> A
[[1, 9, -13], [20, 5, -16]]
>>> A[1]
[20, 5, -16]
>>> A[1][1]
5
```

2 x 3 행렬

$$\begin{bmatrix} 1 & 9 & -13 \\ 20 & 5 & -16 \end{bmatrix}$$

$$A_{2,2} = 5$$