MYONGJI UNIVERSITY

Artificial Intelligence

Lecture 3. Python Review II

Spring 2022

Prof. Jonghoon Chun, Ph.D.

E-mail : jchun@mju.ac.kr
Lecture Note : http://lms.mju.ac.kr

MYONGJI UNIVERSITY

PYTHON 기초 REVIEW II

MYONGJI UNIVERSITY

If 문의 구조

```
if 조건문:
    수행할 문장1
    수행할 문장2
    ...
else:
    수행할 문장A
    수행할 문장B
    ...
```

- if 조건문: 바로 아래 문장부터 if문에 속하는 모든 문장에 들여쓰기(indentation)
- 다음은 오류

```
if 조건문:
    수행할 문장1
수행할 문장2
    수행할 문장3
```

비교연산자 와 논리연산자

비교연산자	설명
x < y	x가 y보다 작다
x > y	x가 y보다 크다
x == y	x와 y가 같다
x != y	x와 y가 같지 않다
x >= y	x가 y보다 크거나 같다
x <= y	x가 y보다 작거나 같다

연산자	설명
x or y	x와 y 둘중에 하나만 참이면 참이다
x and y	x와 y 모두 참이어야 참이다
not x	x가 거짓이면 참이다

If문 예제

In [1]:
```
money = 2000
card = 1
if money >= 3000 or card:
    print("택시를 타고 가라")
else:
    print("걸어가라")
```

택시를 타고 가라

x in s, x not in s

in	not in
x in 리스트	x not in 리스트
x in 튜플	x not in 튜플
x in 문자열	x not in 문자열

In [4]:
```
print(1 in [1, 2, 3])
print(1 not in [1, 2, 3])
print('a' in ('a', 'b', 'c'))
print('j' not in 'python')
```

```
True
False
True
True
```

조건문에서 아무 일도 하지 않게 설정하고 싶다면?

```
In [6]: pocket = ['paper', 'money', 'cellphone']
        if 'money' in pocket:
            pass
        else:
            print("카드를 꺼내라")
```

다중 조건문

```
If <조건문>:
    <수행할 문장1>
    <수행할 문장2>
    ...
elif <조건문>:
    <수행할 문장1>
    <수행할 문장2>
    ...
elif <조건문>:
    <수행할 문장1>
    <수행할 문장2>
    ...
...
else:
   <수행할 문장1>
   <수행할 문장2>
   ...
```

```
In [7]: pocket = ['paper', 'cellphone']
        card = 1
        if 'money' in pocket:
              print("택시를 타고가라")
        elif card:
              print("택시를 타고가라")
        else:
              print("걸어가라")
```

택시를 타고가라

While문

```
while <조건문>:
    <수행할 문장1>
    <수행할 문장2>
    <수행할 문장3>
    ...
```

```
In [8]: treeHit = 0
        while treeHit < 10:
            treeHit = treeHit +1
            print("나무를 %d번 찍었습니다." % treeHit)
            if treeHit == 10:
                print("나무 넘어갑니다.")
```

```
나무를 1번 찍었습니다.
나무를 2번 찍었습니다.
나무를 3번 찍었습니다.
나무를 4번 찍었습니다.
나무를 5번 찍었습니다.
나무를 6번 찍었습니다.
나무를 7번 찍었습니다.
나무를 8번 찍었습니다.
나무를 9번 찍었습니다.
나무를 10번 찍었습니다.
나무 넘어갑니다.
```

break와 continue는 여타 프로그래밍 언어와 사용법이 동일

For문

```
for 변수 in 리스트(또는 튜플, 문자열):
    수행할 문장1
    수행할 문장2
    ...
```

In [10]:
```
test_list = ['one', 'two', 'three']
for i in test_list:
    print(i)
```

```
one
two
three
```

In [11]:
```
a = [(1,2), (3,4), (5,6)]
for (first, last) in a:
    print(first + last)
```

```
3
7
11
```

For문

```
In [13]: marks = [90, 25, 67, 45, 80]

         number = 0
         for mark in marks:
             number = number +1
             if mark >= 60:
                 print("%d번 학생은 합격입니다." % number)
             else:
                 print("%d번 학생은 불합격입니다." % number)
```

```
1번 학생은 합격입니다.
2번 학생은 불합격입니다.
3번 학생은 합격입니다.
4번 학생은 불합격입니다.
5번 학생은 합격입니다.
```

for와 함께 자주 사용하는 range함수

In [18]:
```
# range(10)은 0부터 10 미만(0~9)의 숫자를 포함하는 range  객체를 생성
a = range(10)
print(a)

a = range(1, 10)
print(a)
```

```
range(0, 10)
range(1, 10)
```

In [16]:
```
sum = 0
for i in range(1, 11):
      sum = sum + i
print(sum)
```

```
55
```

In [17]:
```
marks = [90, 25, 67, 45, 80]
for number in range(len(marks)):
    if marks[number] < 60: continue
    print("%d번 학생 축하합니다. 합격입니다." % (number+1))
```

```
1번 학생 축하합니다. 합격입니다.
3번 학생 축하합니다. 합격입니다.
5번 학생 축하합니다. 합격입니다.
```

리스트 안에 for문 포함하기

In [21]:
```
a = [1,2,3,4]
result = []
for num in a:
     result.append(num*3)
print(result)


# 리스트 안에 for문 포함하기
a = [1,2,3,4]
result = [num * 3 for num in a]
print(result)
```

```
[3, 6, 9, 12]
[3, 6, 9, 12]
```

MYONGJI UNIVERSITY

함수

함수의 구조

```
def 함수이름(입력인수):
    <수행할 문장>
    ...
    return 결과값
```

In [22]:
```
def sum(a, b):
    return a+b

a = 3
b = 4
c = sum(a, b)
print(c)
```

```
7
```

In [23]:
```
# 입력값이 없는 경우
def say():
    return 'Hi'

a = say()
print(a)

# return값이 없는 경우
def sum(a, b):
    print("%d, %d의 합은 %d입니다." % (a, b, a+b))

sum(2, 4)
```

```
Hi
2, 4의 합은 6입니다.
```

여러 개의 입력값을 받는 함수

In [27]:
```
def sum_many(*args):
        sum = 0
        for i in args:
            sum = sum + i
        return sum

result = sum_many(1,2,3)
print(result)
result = sum_many(1,2,3,4,5,6,7,8,9,10)
print(result)
```

```
6
55
```

- *args와 같이 변수명 앞에 *를 사용하면 입력. 값들을 전부 모아서 tuple 로 생성함
- args 이외에 임의의 identifier(변수명) 사용 가능

여러 개의 값을 return

In [29]:
```
#  여러개의 값을 리턴할 경우, tuple 형태로 리턴함

def sum_and_mul(a,b):
     return a+b, a*b

result = sum_and_mul(3,4)
print(result)
print(type(result))
```

```
(7, 12)
<class 'tuple'>
```

입력 인수에 초깃값 미리 설정하기

In [30]:
```
# 인수의 초깃값(default) 설정
def say_myself(name, old, man=True):
    print("나의 이름은 %s 입니다." % name)
    print("나이는 %d살입니다." % old)
    if man:
        print("남자입니다.")
    else:
        print("여자입니다.")

say_myself("박응용", 27)
say_myself("박응용", 27, True)
```

```
나의 이름은 박응용 입니다.
나이는 27살입니다.
남자입니다.
나의 이름은 박응용 입니다.
나이는 27살입니다.
남자입니다.
```

사용자 입력

- 사용자가 입력한 값을 어떤 변수에 대입

In [32]:
```
# 사용자 입력
a = input()
print(a)
```

```
Life is too short, you need python
Life is too short, you need python
```

In [33]:
```
# 프롬프트를 띄워서 사용자 입력 받기
number = input("숫자를 입력하세요: ")
print(number)
```

```
숫자를 입력하세요: 3
3
```

print()

In [34]:
```
# 큰따옴표(")로 둘러싸인 문자열은 + 연산과 동일하다
print("life" "is" "too short")
print("life"+"is"+"too short")

# 문자열 띄어쓰기는 콤마로 한다
print("life", "is", "too short")

# 한 줄에 결과값 출력하기
for i in range(10):
    print(i, end=' ')
```

```
lifeistoo short
lifeistoo short
life is too short
0 1 2 3 4 5 6 7 8 9
```

파일 입출력

- 파일 생성 및 오픈

```
f = open("새파일.txt", 'w')
f.close()
```

파일열기모드	설명
r	읽기모드 - 파일을 읽기만 할 때 사용
w	쓰기모드 - 파일에 내용을 쓸 때 사용
a	추가모드 - 파일의 마지막에 새로운 내용을 추가 시킬 때 사용

- 파일 출력

In [35]:
```
f = open("new.txt", 'w')
for i in range(1, 11):
    data = "%d번째 줄입니다.\n" % i
    f.write(data)
f.close()
```

파일 입력

- readline() 함수
  - 라인단위로 읽기

In [37]:
```
f = open("new.txt", 'r')
while True:
    line = f.readline()
    if not line: break
    print(line)
f.close()
```

```
1번째 줄입니다.

2번째 줄입니다.

3번째 줄입니다.

4번째 줄입니다.

5번째 줄입니다.

6번째 줄입니다.

7번째 줄입니다.

8번째 줄입니다.

9번째 줄입니다.

10번째 줄입니다.
```

파일 입력

- readlines() 함수
  - 모든 라인을 읽어서 각각의 라인을 요소로 갖는 리스트로 리턴

```
In [38]: f = open("new.txt", 'r')
         lines = f.readlines()
         for line in lines:
             print(line)
         f.close()
```

```
1번째 줄입니다.

2번째 줄입니다.

3번째 줄입니다.

4번째 줄입니다.

5번째 줄입니다.

6번째 줄입니다.
```

파일 입력

- read() 함수
  - f.read()는 파일의 내용 전체를 문자열로 리턴

```
In [39]: f = open("new.txt", 'r')
         data = f.read()
         print(data)
         f.close()
```

```
1번째 줄입니다.
2번째 줄입니다.
3번째 줄입니다.
4번째 줄입니다.
5번째 줄입니다.
6번째 줄입니다.
7번째 줄입니다.
8번째 줄입니다.
9번째 줄입니다.
10번째 줄입니다.
```

with문과 함께 사용하기

```
with open("foo.txt", "w") as f:
    f.write("Life is too short, you need python")
```

- with 블록을 벗어나는 순간 열린 파일 객체 f가 자동으로 close

```
In [116]: with open("new.txt", "r") as f:
              data = f.read()
              print(data)
```

```
1번째 줄입니다.
2번째 줄입니다.
3번째 줄입니다.
4번째 줄입니다.
5번째 줄입니다.
6번째 줄입니다.
7번째 줄입니다.
8번째 줄입니다.
9번째 줄입니다.
10번째 줄입니다.
```

MYONGJI UNIVERSITY

CLASS

MYONGJI UNIVERSITY

클래스 정의

```
class 클래스이름[(상속 클래스명)]:
    <클래스 변수 1>
    <클래스 변수 2>
    ...
    def 메서드1(self[, 인수1, 인수2,,,]):
        <수행할 문장 1>
        <수행할 문장 2>
        ...
    def 메서드2(self[, 인수1, 인수2,,,]):
        <수행할 문장1>
        <수행할 문장2>
        ...
    ...
```

In [1]:

```
class Calculator:
    def __init__(self):
        self.result = 0

    def adder(self, num):
        self.result += num
        return self.result

cal1 = Calculator()
cal2 = Calculator()

print(cal1.adder(3))
print(cal1.adder(4))
print(cal2.adder(3))
print(cal2.adder(7))
```

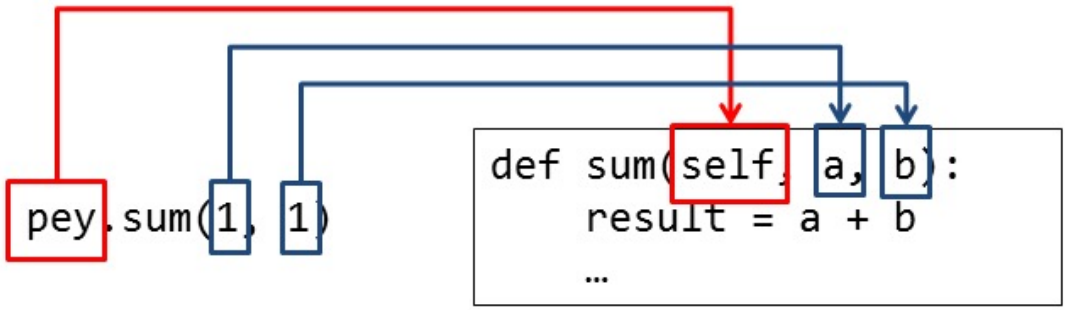
```
3
7
3
10
```

self 인자

In [2]:
```
class Service:
    secret = "영구는 외계인이다."     # 클래스 변수
    def sum(self, a, b):          # 더하기 서비스
        result = a + b
        print("%s + %s = %s입니다." % (a, b, result))

pey = Service()
pey.sum(1,1)
```

```
1 + 1 = 2입니다.
```



"파이썬은 객체를 통해 클래스의 함수를 호출할 때 호출한 객체 자신이 호출한 클래스 함수의 첫번째 입력 인수로 전달된다."

클래스 변수와 객체 변수

In [5]:
```
class Service:
    secret = "영구는 외계인이다."    # 클래스 변수
    def setname(self, name):
        self.name = name           # 객체변수
    def sum(self, a, b):
        result = a + b
        print("%s님 %s + %s = %s입니다." % (self.name, a, b, result))

pey = Service()
pey.setname("홍길동")
pey.sum(1, 1)
print(pey.secret)

pey2 = Service()
pey2.setname("김만식")
pey2.sum(2, 2)
print(pey2.secret)
```

```
홍길동님 1 + 1 = 2입니다.
영구는 외계인이다.
김만식님 2 + 2 = 4입니다.
영구는 외계인이다.
```

__init__

- 객체가 생성될 때 실행되는 method

In [6]:
```
class Service:
    def __init__(self, name):
        self.name = name
    def sum(self, a, b):
        result = a + b
        print("%s님 %s + %s = %s입니다." % (self.name, a, b, result))

pey = Service("홍길동")
pey.sum(1, 1)
```

```
홍길동님 1 + 1 = 2입니다.
```

MYONGJI UNIVERSITY

NUMPY와 PANDAS LIBRARY

MYONGJI UNIVERSITY

PIP (or pip3)

- Python 패키지 S/W를 설치 관리하는 프로그램

- 명령(cmd 창에서)
  - pip list
  - pip install –upgrade pip
  - pip install 패키지이름
  - pip uninstall 패키지이름

- 만약 설치가 되지 않을 경우 패키지를 직접 다운받아 pip 명령으로 설치

Numpy

- 수치계산을 효율적으로 지원하기 위한 라이브러리
- 다차원 배열과 고차원 수학함수 제공

- 설치
  - pip install numpy

참고: http://www.numpy.org/
http://aikorea.org/cs231n/python-numpy-tutorial/

Numpy array (배열)

In [10]:
```
import numpy as np

a = np.array([1, 2, 3])   # rank=1 shape = (3,)
print (type(a))
print (a.shape)
print (a[0], a[1], a[2])
a[0] = (5)                 # 요소를 변경
print (a)

b = np.array([[1,2,3],[4,5,6]])    # rank=2 shape = (2, 3)
print (b.shape)
print (b[0, 0], b[0, 1], b[1, 0])
```

```
<class 'numpy.ndarray'>
(3,)
1 2 3
[5 2 3]
(2, 3)
1 2 4
```

ndim*: 차원 shape: array의 크기

*Rank라는 용어를 썼으나 현재는 deprecated 되어 더 이상 사용되지 않음

특정 값의 배열 생성

In [13]:
```
import numpy as np

a = np.zeros((2,2))   # 모든 값이 0인 배열 생성
print (a)               # 출력 "[[ 0.  0.]
                      #       [ 0.  0.]]"

b = np.ones((1,2))    # 모든 값이 1인 배열 생성
print (b)               # 출력 "[[ 1.  1.]]"

c = np.full((2,2), 7) # 모든 값이 특정 상수인 배열 생성
print (c)                # 출력 "[[ 7.  7.]
                       #       [ 7.  7.]]"

d = np.eye(2)         # 2x2 단위행렬 생성
print (d)               # 출력 "[[ 1.  0.]
                      #       [ 0.  1.]]"

e = np.random.random((2,2)) # (0~1)사이의 임의의 값으로 채워진 배열 생성
print (e)                   # 임의의 값 출력 "[[ 0.91940167  0.08143941]
                            #                 [ 0.68744134  0.87236687]]"
```

```
[[ 0.  0.]
 [ 0.  0.]]
[[ 1.  1.]]
[[7 7]
 [7 7]]
[[ 1.  0.]
 [ 0.  1.]]
[[ 0.88387271  0.20806176]
 [ 0.44524173  0.10710617]]
```

Indexing과 slicing

In [15]:
```
import numpy as np

# Create the following rank 2 array with shape (3, 4)
# [[ 1  2  3  4]
#  [ 5  6  7  8]
#  [ 9 10 11 12]]
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])

# 슬라이싱을 이용하여 첫 두 행과 1열, 2열로 이루어진 부분배열
# b는 shape가 (2,2)인 배열
# [[2 3]
#  [6 7]]
b = a[:2, 1:3]
print(b)

# 슬라이싱된 배열은 원본 배열과 같은 데이터를 참조. 즉 슬라이싱된 배열을 수정하면
# 원본 배열 역시 수정됨
print (a[0, 1])   # 출력 "2"
b[0, 0] = 77    # b[0, 0]은 a[0, 1]과 같은 데이터
print (a[0, 1])   # 출력 "77"
```

```
[[2 3]
 [6 7]]
2
77
```

Indexing과 slicing

In [16]:
```
import numpy as np

# 아래와 같은 요소를 가지는 rank가 2이고 shape가 (3, 4)인 배열 생성
# [[ 1  2  3  4]
#  [ 5  6  7  8]
#  [ 9 10 11 12]]
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])

# 배열의 중간 행에 접근하는 두 가지 방법이 있습니다.
# 정수 인덱싱과 슬라이싱을 혼합해서 사용하면 낮은 rank의 배열이 생성되지만,
# 슬라이싱만 사용하면 원본 배열과 동일한 rank의 배열이 생성.
row_r1 = a[1, :]    # 배열a의 두 번째 행을 rank가 1인 배열로
row_r2 = a[1:2, :]  # 배열a의 두 번째 행을 rank가 2인 배열로
print (row_r1, row_r1.shape)  # 출력 "[5 6 7 8] (4,)"
print (row_r2, row_r2.shape) # 출력 "[[5 6 7 8]] (1, 4)"

# 행이 아닌 열의 경우에도 마찬가지
col_r1 = a[:, 1]
col_r2 = a[:, 1:2]
print (col_r1, col_r1.shape)  # 출력 "[ 2  6 10] (3,)"
print (col_r2, col_r2.shape)  # 출력 "[[ 2]
                              #       [ 6]
                              #       [10]] (3, 1)"
```

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

Data type(자료형)

- 명시적으로 특정 자료형을 지정 가능

In [17]:
```
import numpy as np

x = np.array([1, 2])  # Numpy가 자료형을 추측해서 선택
print (x.dtype)           # 출력 "int64"

x = np.array([1.0, 2.0])  # Numpy가 자료형을 추측해서 선택
print (x.dtype)             # 출력 "float64"

x = np.array([1, 2], dtype=np.int64)  # 특정 자료형을 명시적으로 지정
print (x.dtype)                         # 출력 "int64"
```

```
int32
float64
int64
```

In [151]:
```
x = np.array([1.3, 2.7], dtype=np.int64)
print(x.dtype)
print(x)
```

```
int64
[1 2]
```

배열 연산

In [19]:
```
import numpy as np

x = np.array([[1,2],[3,4]], dtype=np.float64)
y = np.array([[5,6],[7,8]], dtype=np.float64)

# [[ 6.0  8.0]
#  [10.0 12.0]]
print (x + y)
print (np.add(x, y))

# [[-4.0 -4.0]
#  [-4.0 -4.0]]
print (x - y)
print (np.subtract(x, y))

# [[ 5.0 12.0]
#  [21.0 32.0]]
print (x * y)
print (np.multiply(x, y))

# [[ 0.2         0.33333333]
#  [ 0.42857143  0.5       ]]
print (x / y)
print (np.divide(x, y))

# [[ 1.          1.41421356]
#  [ 1.73205081  2.        ]]
print (np.sqrt(x))
```

```
[[  6.   8.]
 [ 10.  12.]]
[[  6.   8.]
 [ 10.  12.]]
[[-4. -4.]
 [-4. -4.]]
[[-4. -4.]
 [-4. -4.]]
[[  5.  12.]
 [ 21.  32.]]
[[  5.  12.]
 [ 21.  32.]]
[[ 0.2         0.33333333]
 [ 0.42857143  0.5       ]]
[[ 0.2         0.33333333]
 [ 0.42857143  0.5       ]]
[[ 1.          1.41421356]
 [ 1.73205081  2.        ]]
```

배열 연산

In [21]:
```
import numpy as np

x = np.array([[1,2],[3,4]])

print (np.sum(x))  # 모든 요소를 합한 값을 연산; 출력 "10"
print (np.sum(x, axis=0))  # 각 열에 대한 합을 연산; 출력 "[4 6]"
print (np.sum(x, axis=1))  # 각 행에 대한 합을 연산; 출력 "[3 7]"
```

```
10
[4 6]
[3 7]
```

sum이외에도 mean, var, std, median 등 함수가 가능

Transpose와 broadcasting

In [25]:
```
import numpy as np

x = np.array([[1,2], [3,4]])
print (x)     # 출력 "[[1 2]
              #          [3 4]]"
print (x.T)   # 출력 "[[1 3]
              #          [2 4]]"

# rank 1인 배열을 전치할 경우 아무 일도 일어나지 않음
v = np.array([1,2,3])
print (v)     # 출력 "[1 2 3]"
print (v.T)   # 출력 "[1 2 3]"

# 2차원 배열
v = np.array([[1,2,3]])
print (v)     # 출력 "[1 2 3]"
print (v.T)   # 출력 "[1 2 3]"
```

x.T 대신에 np.transpose(x)도 가능

```
[[1 2]
 [3 4]]
[[1 3]
 [2 4]]
[1 2 3]
[1 2 3]
[[1 2 3]]
[[1]
 [2]
 [3]]
```

In [27]:
```
x = np.array([1, 2, 3, 4])
y = 3

print(x * 3)   # broadcasting
print(x + 3)
print(x / 3)
```

```
[ 3  6  9 12]
[4 5 6 7]
[ 0.33333333  0.66666667  1.          1.33333333]
```

Reshape (차원변경)

In [152]:
```
# 차원 변경 함수 (reshape)
x = np.array([[1, 2, 3, 4], [5,6,7,8]])
y = x.reshape((4, 2)) # 4 X 2 배열로 reshape

print(x)
print(y)
```

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

Pandas

- Python Data Analysis Library
- 데이터 분석 기능을 제공하는 라이브러리
- csv 파일 처리도 가능
- http://pandas.pydata.org/

- 기본적인 데이터 형식은 series(1차원)와 dataframe(2차원)
  - 2차원 데이터를 다룰 때 편리함

Series

- Series는 1차원 배열과 같은 자료 구조

```
In [65]: import pandas as pd

kakao = pd.Series([92600, 92400, 92100, 94300, 92300])
print(kakao)

print(kakao[0])
print(kakao[2])
print(kakao[4])
```

```
0    92600
1    92400
2    92100
3    94300
4    92300
dtype: int64
92600
92100
92300
```

```
In [159]: mine = pd.Series([10, 20, 30], index=['naver', 'sk', 'kt'])

for idx in mine.index:
    print(idx)

for price in mine.values:
    print(price)

print(mine)
```

```
naver
sk
kt
10
20
30
naver    10
sk       20
kt       30
dtype: int64
```

DataFrame

- 여러 개의 칼럼(Column)으로 구성된 2차원 형태의 자료구조
- 주로 파이썬 딕셔너리를 이용하여 정의
  - Key-value pair가 각 컬럼을 구성

In [69]:
```
import pandas as pd

raw_data = {'col0': [1, 2, 3, 4],
            'col1': [10, 20, 30, 40],
            'col2': [100, 200, 300, 400]}

data = pd.DataFrame(raw_data)
print(data)
print(data['col1'])
```

```
   col0  col1  col2
0     1    10   100
1     2    20   200
2     3    30   300
3     4    40   400
0    10
1    20
2    30
3    40
Name: col1, dtype: int64
```

In [70]:
```
print(type(data['col0']))  # dataframe의 각 column은 series 객체
```

```
<class 'pandas.core.series.Series'>
```

DataFrame

In [73]:
```
import pandas as pd
# 키, 몸무게, 유형 데이터프레임 생성하기
tbl = pd.DataFrame({
    "weight": [ 80.0, 70.4, 65.5, 45.9, 51.2 ],
    "height": [ 170,  180,  155,  143,  154  ],
    "type":   [ "f", "n", "n", "t", "t"]
})
# 몸무게 목록 추출하기
print("몸무게 목록")
print(tbl["weight"])
# 몸무게와 키 목록 추출하기
print("몸무게와 키 목록")
print(tbl[["weight","height"]])  # 2개 이상의 column을 추출할 경우 리스트를 이용
```

```
몸무게 목록
0    80.0
1    70.4
2    65.5
3    45.9
4    51.2
Name: weight, dtype: float64
몸무게와 키 목록
   weight  height
0    80.0     170
1    70.4     180
2    65.5     155
3    45.9     143
4    51.2     154
```

DataFrame slicing

In [170]:
```
import pandas as pd

tbl = pd.DataFrame({
    "weight": [80.0, 70.4, 65.5, 45.9, 51.2],
    "height": [170, 180, 155, 143, 154],
    "type": ["f", "n", "n", "t", "t"]
})

print(tbl[2:4]) # row 조건
print(tbl.iloc[2:4, :]) # 위와 동일한 결과(iloc)
print("\n")
print(tbl["weight"][2:4]) # Column과 row조건이 모두 포함
                          # tbl[2:4]["weight"]와 같이 기술해도 결과는 동일
```

```
   weight  height type
2    65.5     155    n
3    45.9     143    t
   weight  height type
2    65.5     155    n
3    45.9     143    t


2    65.5
3    45.9
Name: weight, dtype: float64
```

DataFrame row 조건검색

In [79]:
```
import pandas as pd
# 키, 몸무게, 유형 데이터프레임 생성하기
tbl = pd.DataFrame({
    "weight": [ 80.0, 70.4, 65.5, 45.9, 51.2, 72.5 ],
    "height": [ 170,  180,  155,  143,  154,  160  ],
    "gender": [ "f",  "m",  "m",  "f",  "f",  "m"  ]
})
print("--- height가 160 이상인 것")
print(tbl[tbl.height >= 160])
print("--- gender가 m 인 것")
print(tbl[tbl.gender == "m"])
```

```
--- height가 160 이상인 것
  gender  height  weight
0      f     170    80.0
1      m     180    70.4
5      m     160    72.5
--- gender가 m 인 것
  gender  height  weight
1      m     180    70.4
2      m     155    65.5
5      m     160    72.5
```

DataFrame 정렬

In [177]:
```
import pandas as pd
# 키, 몸무게, 유형 데이터프레임 생성하기

tbl = pd.DataFrame({
    "weight":[80.0, 70.4, 65.5, 45.9, 51.2, 72.5],
    "height":[170, 180, 155, 143, 154, 160],
    "gender":["f", "m", "m", "f", "f", "m"]
})

print("--- 키로 정렬")
print(tbl.sort_values(by="height"))
print("--- 몸무게로 정렬")
print(tbl.sort_values(by="weight", ascending=False))
```

```
--- 키로 정렬
   weight  height gender
3    45.9     143      f
4    51.2     154      f
2    65.5     155      m
5    72.5     160      m
0    80.0     170      f
1    70.4     180      m
--- 몸무게로 정렬
   weight  height gender
0    80.0     170      f
5    72.5     160      m
1    70.4     180      m
2    65.5     155      m
4    51.2     154      f
3    45.9     143      f
```

Column 추가

In [4]:
```
import pandas as pd

tbl = pd.DataFrame({
    "weight": [ 80.0, 70.4, 65.5, 45.9, 51.2, 72.5 ],
    "height": [ 170,  180,  155,  143,  154,  160  ],
    "gender": [ "f",  "m",  "m",  "f",  "f",  "m"  ]
})

tbl["eye"] = [1, 2, 3, 4, 5, 6]

print(tbl)
```

```
  gender  height  weight  eye
0      f     170    80.0    1
1      m     180    70.4    2
2      m     155    65.5    3
3      f     143    45.9    4
4      f     154    51.2    5
5      m     160    72.5    6
```

Column 순서 변경

In [181]:
```
import pandas as pd

tbl = pd.DataFrame({
    "weight":[80.0, 70.4, 65.5, 45.9, 51.2, 72.5],
    "height":[170, 180, 155, 143, 154, 160],
    "gender":["f", "m", "m", "f", "f", "m"]
})

print(tbl)

tbl = pd.DataFrame({
    "weight":[80.0, 70.4, 65.5, 45.9, 51.2, 72.5],
    "height":[170, 180, 155, 143, 154, 160],
    "gender":["f", "m", "m", "f", "f", "m"]},
    columns = ["gender", "height", "weight"]
)
print(tbl)
```

```
   weight  height gender
0    80.0     170      f
1    70.4     180      m
2    65.5     155      m
3    45.9     143      f
4    51.2     154      f
5    72.5     160      m
  gender  height  weight
0      f     170    80.0
1      m     180    70.4
2      m     155    65.5
3      f     143    45.9
4      f     154    51.2
5      m     160    72.5
```

Normalization (min max 활용)

In [81]:
```
import pandas as pd

tbl = pd.DataFrame({
    "weight": [ 80.0, 70.4, 65.5, 45.9, 51.2, 72.5 ],
    "height": [ 170,  180,  155,  143,  154,  160  ],
    "gender": [ "f",  "m",  "m",  "f",  "f",  "m"  ]
})
# 키와 몸무게 정규화하기
# 최댓값과 최솟값 구하기
def norm(tbl, key):
    c = tbl[key]
    v_max = c.max()
    v_min = c.min()
    print(key, "=", v_min, "-", v_max)
    tbl[key] = (c - v_min) / (v_max - v_min)
norm(tbl, "weight")
norm(tbl, "height")
print(tbl)
```

```
weight = 45.9 - 80.0
height = 143 - 180
  gender    height    weight
0      f  0.729730  1.000000
1      m  1.000000  0.718475
2      m  0.324324  0.574780
3      f  0.000000  0.000000
4      f  0.297297  0.155425
5      m  0.459459  0.780059
```

Numpy 객체 변환

- Pandas를 지원하지 않는 라이브러리의 경우 numpy로 변환 필요

In [10]:
```
import pandas as pd

tbl = pd.DataFrame({
    "x": [1, 2, 3],
    "y": [4, 5, 6],
    "z": [7, 8, 9]
})

n = tbl.to_numpy()

print(n)
```

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

MYONGJI UNIVERSITY

END

MYONGJI UNIVERSITY