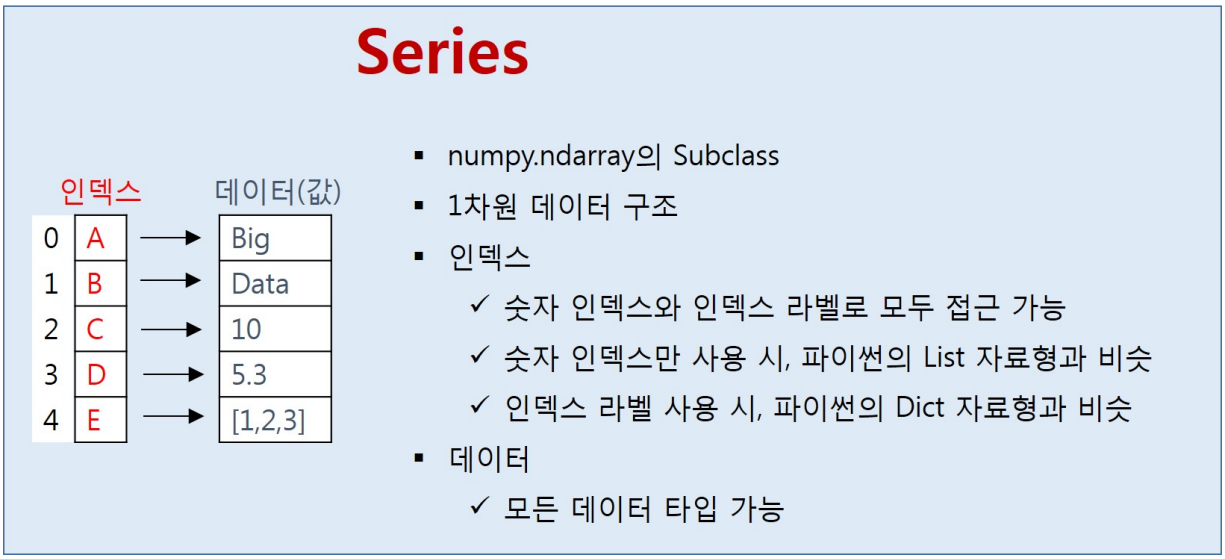
Pandas 주요 데이터 타입

Series (시리즈)



In [2]:
```
# python module import
import os
os.getcwd()

import pandas

# as를 통하여 module을 간단하게 접근한다
import pandas as pd

# pandas.Series == pd.Series
# pandas.Series()
# pd.Series()

# from module import module function
# from pandas import Series, DataFrame
```

Series 생성

(1) list로 생성

In [3]:
```
pd.Series([3, 5, 1, 2])
```

Out[3]:
```
0    3
1    5
2    1
3    2
dtype: int64
```

(2) dictionary로 생성

In [4]:
```
# 사전 타입의 데이터로 Series 생성
data2 = pd.Series({'a': 3, 'b': 15, 'c': 5, 'd': 35})
data2
```

Out[4]:
```
a     3
b    15
c     5
d    35
dtype: int64
```

In [5]:
```
# type(data) : data의 타입 확인
type(data2)
```

Out[5]:
```
pandas.core.series.Series
```

In [6]:
```
# dtypes : data 형태 반환
data2.dtypes
```

Out[6]:
```
dtype('int64')
```

In [7]:
```
# 3번째 데이터 접근
# python index는 0부터 시작
data2[2]
```

Out[7]:
```
5
```

In [8]:
```
# 값은 모든 데이터 타입 가능
data3 = pd.Series(['big data', 30, [1, 2, 3]])
data3
```

Out[8]:
```
0     big data
1           30
2    [1, 2, 3]
dtype: object
```

In [9]:
```
print(data2); print(data3)
```

```
a     3
b    15
c     5
d    35
dtype: int64
0     big data
1           30
2    [1, 2, 3]
dtype: object
```

In [10]:
```
# Series.index : series index
data3.index = ['수강명', '수강생수', '비고']
data3
```

Out[10]:
```
수강명       big data
수강생수           30
비고       [1, 2, 3]
dtype: object
```

Series 생성할때 2가지 방법

(1) list

Series([data value], index=[data index])

(2) dictionary

Series({data index : data value})

In [11]:
```
# index label로 Series 접근
data3['수강생수']
```

Out[11]:
```
30
```

In [12]:
```
# Series 생성할 때 index 같이 생성
data4 = pd.Series([3, 15, 5, 15], index = ['a', 'b', 'c', 'd'])
data4
```

Out[12]:
```
a     3
b    15
c     5
d    15
dtype: int64
```

In [13]:
```
# index 숫자로 접근
data4[2]
```

Out[13]:
```
5
```

In [14]:
```
# index name(label)으로 접근
data4['c']
```

Out[14]:
```
5
```

Series 색인

In [15]:
```
list('abcdefghij')
```

Out[15]:
```
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
```

In [16]:
```
data5 = pd.Series([x for x in range(10, 101, 10)],
                  index = list('abcdefghij'))
data5
```

Out[16]:
```
a     10
b     20
c     30
d     40
e     50
f     60
g     70
h     80
i     90
j    100
dtype: int64
```

In [17]:
```
# 숫자인덱스가 5인 항목 조회
data5[5]
```

Out[17]:
```
60
```

In [18]:
```
# 인덱스 라벨이 'f'인 항목 조회
data5['f']
```

Out[18]:
```
60
```

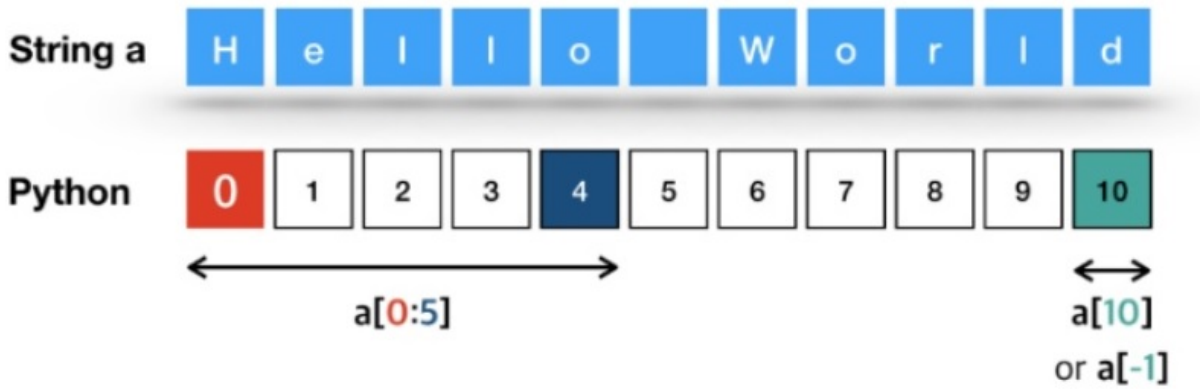
In [19]:
```
# 숫자인덱스가 1,3,5인 항목 조회
data5[[1, 3, 5]]
```

Out[19]:
```
b    20
d    40
f    60
dtype: int64
```

In [20]:
```
# 'a','c','f' 만 조회
data5[['a', 'c', 'f']]
```

Out[20]:
```
a    10
c    30
f    60
dtype: int64
```

인덱스 슬라이싱 (숫자 인덱스)



1) :을 기점으로 왼쪽에 start index, 오른쪽에 end index 지정

2) start index 보다 크거나 같고, end index보다 작은 항목을 선택

3) 예를 들어, a[2:5]는 a[[2,3,4]]와 동일

4) start index와 end index는 생략 가능하며, 생략 시 start index는 첫 인덱스, end index는 끝 인덱스로 지정됨.

5) 음수 인덱스는 끝에서 시작되는 역인덱스임.

In [21]:
```
# 숫자 인덱스가 2,3,4,5인 항목 선택
data5[[2,3,4,5]]
# 콜론(:)을 사용하면 list로 반환이 가능하므로 바로 출력 가능
data5[2:6]
```

Out[21]:
```
c    30
d    40
e    50
f    60
dtype: int64
```

In [22]:
```
# 맨 앞에 있는 5개의 항목 선택
data5[:5]
```

Out[22]:
```
a    10
b    20
c    30
d    40
e    50
dtype: int64
```

In [23]:
```
# 뒤에서 2번째 항목 선택
data5[-2]
```

Out[23]:
```
90
```

In [24]:
```
# 가장 뒤에 있는 3개의 항목 선택
data5[-3:]
```

Out[24]:
```
h     80
i     90
j    100
dtype: int64
```

In [25]:
```
# 뒤에서 3번째, 2번째 불러오기
data5[-3:-1]
```

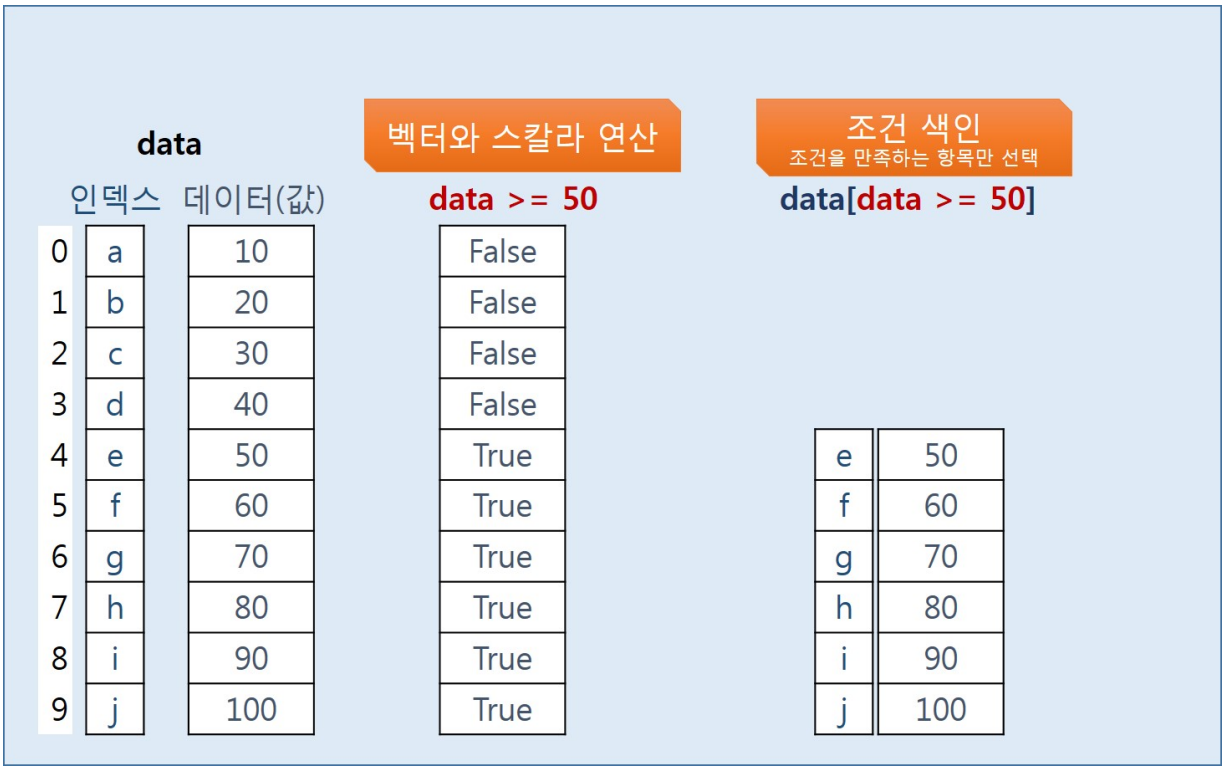
Out[25]:
```
h    80
i    90
dtype: int64
```

In [26]:
```
# 인덱스 라벨이 'c'부터 'g'까지 항목 선택
data5['c':'g']
```

Out[26]:
```
c    30
d    40
e    50
f    60
g    70
dtype: int64
```

조건을 활용한 인덱스 선택 (조건 색인)

데이터 분석에서 가장 중요한 부분이므로 반드시 기억해야함!



In [27]:
```
# 데이터 값이 70이상인 데이터만 불러오기
data5[data5 >= 70]
```

Out[27]:
```
g     70
h     80
i     90
j    100
dtype: int64
```

In [28]:
```
# 값이 30보다 크거나 같고, 70보다 작은 항목만 선택
data5[(data5 >= 30) & (data5 < 70)]
```

Out[28]:
```
c    30
d    40
e    50
f    60
dtype: int64
```

Series 속성 조회 및 주요 함수

가장 중요 최고로 중요

sample

A	10
B	-2
C	3.5
D	4
E	8

		설명	예제 혹은 결과
속성	Index	인덱스 정보 출력	['A','B','C','D','E']
	values	데이터 출력	[10, -2, 3.5, 4, 8]
	size	데이터 항목의 개수 출력	5
	loc	인덱스 라벨로 특정 항목 색인	sample.loc['C'] → 3.5
	iloc	숫자 인덱스로 특정 항목 색인	sample.iloc[4] → 8
함수()	abs()	각 항목별 절대값 출력	[10, **2**, 3.5, 4, 8]
	max()	크기가 가장 큰 항목 출력 (최소값(min), 평균(mean), 항목 개수 (count) 등도 있음)	10
	append()	새로운 항목 추가	sample.append(Series({'F':-15})) --> 인덱스 라벨과 값이 -15인 항목 추가
	add()	모든 항목에 값을 더함.	sample.add(7) → [17, 5, 10.5, 11, 15]
	describe()	통계 수치 제공 (평균, max/min 등)	

공식 문서: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.html

In [31]:
```
data6 = pd.Series([10, -2, 3.5, 4, 8],
                  index = list('ABCDE'))
data6
```

Out[31]:
```
A    10.0
B    -2.0
C     3.5
D     4.0
E     8.0
dtype: float64
```

속성 (Attributes)

In [32]:
```
# index
# 반환형이 list
data6.index
```

Out[32]:
```
Index(['A', 'B', 'C', 'D', 'E'], dtype='object')
```

In [33]:
```
# values
data6.values
```

Out[33]:
```
array([10. , -2. ,  3.5,  4. ,  8. ])
```

In [35]:
```
# dtype
data6.dtype
```

Out[35]:
```
dtype('float64')
```

In [36]:
```
# size
data6.size
```

Out[36]:
```
5
```

In [37]:
```
# loc
# 숫자 기입할 경우 에러
data6.loc['C']
```

Out[37]:
```
3.5
```

In [38]:
```
# iloc (i: index)
# 문자 기입할 경우 에러
data6.iloc[2]
```

Out[38]:
```
3.5
```

함수 (function)

In [39]:
```
# abs()
# data6의 값 자체는 변화X
data6.abs()
```

Out[39]:
```
A    10.0
B     2.0
C     3.5
D     4.0
E     8.0
dtype: float64
```

In [41]:
```
# max()
data6.max()
```

Out[41]:
```
10.0
```

In [42]:
```
# min()
data6.min()
```

Out[42]:
```
-2.0
```

In [43]:
```
# mean()
data6.mean()
```

Out[43]:
```
4.7
```

In [44]:
```
# append()
# list의 append: 값을 추가
data6.append(pd.Series({'F':2}))
```

Out[44]:
```
A    10.0
B    -2.0
C     3.5
D     4.0
E     8.0
F     2.0
dtype: float64
```

In [45]:
```
# add()
data6.add(100)
```

Out[45]:
```
A    110.0
B     98.0
C    103.5
D    104.0
E    108.0
dtype: float64
```

In [46]:
```
# describe()
data6.describe()
```

Out[46]:
```
count     5.000000
mean      4.700000
std       4.631414
min      -2.000000
25%       3.500000
50%       4.000000
75%       8.000000
max      10.000000
dtype: float64
```

In [85]:
```
# 0 ~100 Series만들기
```

Out[85]:
```
0         0
1         1
2         2
3         3
4         4
       ...
96       96
97       97
98       98
99       99
100     100
Length: 101, dtype: int64
```

In [86]:
```

```

Out[86]:
```
count    101.000000
mean      50.000000
std       29.300171
min        0.000000
25%       25.000000
50%       50.000000
75%       75.000000
max      100.000000
dtype: float64
```