

# 파이썬 머신러닝 판다스 데이터분석

## Lecture (8)



Dr. Heesuk Kim

Part 0. 개발환경 준비

**Part 1. 판다스 입문**

Part 2. 데이터 입출력

Part 3. 데이터 살펴보기

Part 4. 시각화 도구

Part 5. 데이터 사전처리

Part 6. 데이터프레임의 다양한 응용

Part 7. 머신러닝 데이터 분석



# Part 1. 판다스 입문

---

1. 데이터과학자가 판다스를 배우는 이유
2. 판다스 자료구조
  - 2-1. 시리즈
  - 2-2. 데이터프레임
3. 인덱스 활용
4. 산술 연산
  - 4-1. 시리즈 연산
  - 4-2. 데이터프레임 연산



# Part 1. 판다스 입문

## 3. 인덱스 활용

### • 행 인덱스 초기화

- reset\_index() 메소드로 정수형 위치 인덱스로 초기화.  
기존 행 인덱스는 열로 이동. 새로운 데이터프레임 객체를 반환.

정수형 위치 인덱스로 초기화: DataFrame 객체.reset\_index( )

#### 예제 1-18

정수형 위치 인덱스로 초기화

(File: example/part1/1.18\_reset\_index.py)

```
1 # -*- coding: utf-8 -*-
2
3 import pandas as pd
4
5 # 딕셔너리 정의
6 dict_data = {'c0':[1,2,3], 'c1':[4,5,6], 'c2':[7,8,9], 'c3':[10,11,12], 'c4':[13,14,15]}
7
8 # 딕셔너리를 데이터프레임으로 변환. 인덱스를 [r0, r1, r2]로 지정
9 df = pd.DataFrame(dict_data, index=['r0', 'r1', 'r2'])
10 print(df)
11 print('\n')
12
13 # 행 인덱스를 정수형으로 초기화
14 ndf = df.reset_index()
15 print(ndf)
```

〈실행 결과〉 코드 전부 실행

	c0	c1	c2	c3	c4
r0	1	4	7	10	13
r1	2	5	8	11	14
r2	3	6	9	12	15

	index	c0	c1	c2	c3	c4
0	r0	1	4	7	10	13
1	r1	2	5	8	11	14
2	r2	3	6	9	12	15



Spyder (Python 3.7)

File Edit Search Source Run Debug Consoles Projects Tools View Help

Editor - D:\W\>>>2020년\_1학기\_LECTURE\의료정보처리(1)\source\part1\1.18\_reset\_index.py

1,16\_set\_index.py\* 1,17\_reindex.py\* 1,18\_reset\_index.py\*

```

1 |
2 |
3 import pandas as pd
4 |
5 # 딕셔너리를 정의
6 dict_data = {'c0':[1,2,3], 'c1':[4,5,6], 'c2':[7,8,9], 'c3':[10,11,12], 'c4':[13,14,15]}
7 |
8 # 딕셔너리를 데이터프레임으로 변환. 인덱스를 [r0, r1, r2]로 지정
9 df = pd.DataFrame(dict_data, index=['r0', 'r1', 'r2'])
10 print(df)
11 print('\n')
12 |
13 # 행 인덱스를 정수형으로 초기화
14 ndf = df.reset_index()
15 print(ndf)

```

Variable explorer

Name	Type	Size	Value
------	------	------	-------

IPython console

Console 11/A

```

Python 3.7.3 (default, Apr 24 2019, 15:29:51) [MSC v.
1915 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more
information.

IPython 7.6.1 -- An enhanced Interactive Python.

In [1]:

```

Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 1 Column: 1 Memory: 80 %

# Part 1. 판다스 입문

## 3. 인덱스 활용

### • 행 인덱스를 기준으로 데이터프레임 정렬

- sort\_index() 메소드로 행 인덱스를 기준으로 정렬.
- ascending 옵션을 사용하여 오름차순 또는 내림차순 설정.
- 새롭게 정렬된 데이터프레임 객체를 반환.

행 인덱스 기준 정렬: DataFrame 객체.sort\_index( )

#### 예제 1-19

데이터프레임 정렬

(File: example/part1/1.19\_sort\_index.py)

```
1  # -*- coding: utf-8 -*-
2
3  import pandas as pd
4
5  # 딕셔너리 정의
6  dict_data = {'c0':[1,2,3], 'c1':[4,5,6], 'c2':[7,8,9], 'c3':[10,11,12], 'c4':[13,14,15]}
7
8  # 딕셔너리를 데이터프레임으로 변환. 인덱스를 [r0, r1, r2]로 지정
9  df = pd.DataFrame(dict_data, index=['r0', 'r1', 'r2'])
10 print(df)
11 print('\n')
12
13 # 내림차순으로 행 인덱스 정렬
14 ndf = df.sort_index(ascending=False)
15 print(ndf)
```

〈실행 결과〉 코드 전부 실행

	c0	c1	c2	c3	c4
r0	1	4	7	10	13
r1	2	5	8	11	14
r2	3	6	9	12	15

	c0	c1	c2	c3	c4
r2	3	6	9	12	15
r1	2	5	8	11	14
r0	1	4	7	10	13



Spyder (Python 3.7)

File Edit Search Source Run Debug Consoles Projects Tools View Help

Editor - D:\W\>>>2020년\_1학기\_LECTURE\의료정보처리(1)\source\part1\19\_sort\_index.py

1,16\_set\_index.py 1,17\_reindex.py 1,18\_reset\_index.py 1,19\_sort\_index.py

```

1
2
3 import pandas as pd
4
5 # 딕셔너리를 정의
6 dict_data = {'c0':[1,2,3], 'c1':[4,15,6], 'c2':[7,8,9], 'c3':[10,11,12], 'c4':[13,14,15]}
7
8 # 딕셔너리를 데이터프레임으로 변환. 인덱스를 [r0, r1, r2]로 지정
9 df = pd.DataFrame(dict_data, index=['r0', 'r1', 'r2'])
10 print(df)
11 print('\n')
12
13 # 내림차순으로 행 인덱스 정렬
14 ndf = df.sort_index(ascending=False)
15 print(ndf)
16
17 |

```

Variable explorer

Name	Type	Size	Value
------	------	------	-------

IPython console

Console 15/A

Python 3.7.3 (default, Apr 24 2019, 15:29:51) [MSC v.1915 64 bit (AMD64)]  
Type "copyright", "credits" or "license" for more information.

IPython 7.6.1 -- An enhanced Interactive Python.

In [1]:

Permissions: RW End-of-lines: CRLF Encoding: UTF-8-GUESSED Line: 17 Column: 1 Memory: 80 %

### Part 1. 판다스 입문

---

- 1. 데이터과학자가 판다스를 배우는 이유
- 2. 판다스 자료구조
  - 2-1. 시리즈
  - 2-2. 데이터프레임
- 3. 인덱스 활용
- 4. 산술 연산
  - 4-1. 시리즈 연산
  - 4-2. 데이터프레임 연산





# Part 1. 판다스 입문

## 4. 산술연산

- 판다스 객체의 산술연산은 내부적으로 **3단계 프로세스**를 거친다.

- ① 행/열 인덱스를 기준으로 모든 원소를 정렬한다.
- ② 동일한 위치에 있는 원소끼리 일대일로 대응시킨다.
- ③ 일대일 대응이 되는 원소끼리 연산을 처리한다. 이때, 대응되는 원소가 없으면 NaN으로 처리한다.

### 4-1. 시리즈 연산

- 시리즈 vs. 숫자

시리즈와 숫자 연산: **Series객체** + 연산자(+, -, \*, /) + 숫자

#### 예제 1-21

시리즈를 숫자로 나누기

(File: example/part1/1.21\_series\_to\_number.py)

```
1 # -*- coding: utf-8 -*-
2
3 # 라이브러리 불러오기
4 import pandas as pd
5
6 # 딕셔너리 데이터로 판다스 시리즈 만들기
7 student1 = pd.Series({'국어':100, '영어':80, '수학':90})
8 print(student1)
9 print('\n')
10
11 # 학생의 과목별 점수를 200으로 나누기
12 percentage = student1/200
13
14 print(percentage)
15 print('\n')
16 print(type(percentage))
```

<실행 결과> 코드 전부 실행

```
국어    100
영어     80
수학     90
dtype: int64

국어    0.50
영어    0.40
수학    0.45
dtype: float64

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

시리즈 객체에 어떤 숫자를 더하면, 시리즈의 개별 원소에 각각 숫자를 더하고 계산한 결과를 시리즈 객체로 반환한다. 덧셈, 뺄셈, 곱셈, 나눗셈 모두 가능하다. 앞의 예제에서는 시리즈 객체의 각 원소를 200으로 나누는 과정을 살펴본다.



The screenshot shows the Spyder Python IDE interface. The main editor window displays a Python script with the following code:

```

1
2
3 # 라이브러리 불러오기
4 import pandas as pd
5
6 # 딕셔너리 데이터로 판다스 시리즈 만들기
7 student1 = pd.Series({'국어':100, '영어':80, '수학':90})
8 print(student1)
9 print('\n')
10
11 # 학생의 과목별 점수를 200으로 나누기
12 percentage = student1 / 200
13
14 print(percentage)
15 print('\n')
16 print(type(percentage))
    
```

The Variable explorer on the right shows no variables. The IPython console at the bottom displays the following output:

```

Python 3.7.3 (default, Apr 24 2019, 15:29:51) [MSC v.1915 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.6.1 -- An enhanced Interactive Python.

In [1]:
    
```

The status bar at the bottom indicates: Permissions: RW, End-of-lines: CRLF, Encoding: UTF-8, Line: 16, Column: 24, Memory: 85 %.



# Part 1. 판다스 입문

## 4-1. 시리즈 연산

### • 시리즈 vs. 시리즈

- 시리즈와 시리즈 사이에 사칙연산 처리.
- 같은 인덱스를 가진 원소끼리 계산.
- 해당 인덱스에 연산 결과를 매칭하여 새 시리즈를 반환.

시리즈와 시리즈 연산: Series1 + 연산자(+, -, \*, /) + Series2

#### 예제 1-22

시리즈 사칙연산

(File: example/part1/1.22\_series\_to\_series.py)

```
1 # -*- coding: utf-8 -*-
2
3 # 라이브러리 불러오기
4 import pandas as pd
5
6 # 딕셔너리 데이터로 판다스 시리즈 만들기
7 student1 = pd.Series({'국어':100, '영어':80, '수학':90})
8 student2 = pd.Series({'수학':80, '국어':90, '영어':80})
9
10 print(student1)
11 print('\n')
12 print(student2)
13 print('\n')
14
15 # 두 학생의 과목별 점수로 사칙연산 수행
16 addition = student1 + student2          # 덧셈
17 subtraction = student1 - student2       # 뺄셈
18 multiplication = student1 * student2    # 곱셈
19 division = student1 / student2          # 나눗셈
20 print(type(division))
21 print('\n')
22
23 # 사칙연산 결과를 데이터프레임으로 합치기 (시리즈 -> 데이터프레임)
24 result = pd.DataFrame([addition, subtraction, multiplication, division],
25                        index=['덧셈', '뺄셈', '곱셈', '나눗셈'])
26 print(result)
```

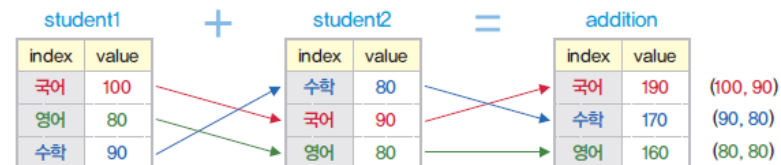
<실행 결과> 코드 전부 실행

```
국어    100
영어     80
수학     90
dtype: int64
```

```
수학     80
국어     90
영어     80
dtype: int64
```

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

	국어	수학	영어
덧셈	190.000000	170.000	160.0
뺄셈	10.000000	10.000	0.0
곱셈	9000.000000	7200.000	6400.0
나눗셈	1.111111	1.125	1.0



[그림 1-17] 시리즈의 산술연산(덧셈)

The screenshot shows the Spyder Python IDE interface. The main editor window displays a Python script with the following code:

```

1 # -*- coding: utf-8 -*-
2
3 # 라이브러리 불러오기
4 import pandas as pd
5
6 # 딕셔너리 데이터로 판다스 시리즈 만들기
7 student1 = pd.Series({'국어':100, '영어':80, '수학':90})
8 student2 = pd.Series({'수학':80, '국어':90, '영어':80})
9
10 print(student1)
11 print('\n')
12 print(student2)
13 print('\n')
14
15 # 두 학생의 과목별 점수로 사칙연산 수행
16 addition = student1 + student2           # 덧셈
17 subtraction = student1 - student2        # 뺄셈
18 multiplication = student1 * student2     # 곱셈
19 division = student1 / student2           # 나눗셈
20 print(type(division))
21 print('\n')
22
23 # 사칙연산 결과를 데이터프레임으로 합치기 (시리즈 -> 데이터프레임)
24 result = pd.DataFrame([addition, subtraction, multiplication, division],
25                        index=['덧셈', '뺄셈', '곱셈', '나눗셈'])
26 print(result)
    
```

The Variable explorer on the right shows no variables. The IPython console at the bottom displays the following output:

```

Python 3.7.3 (default, Apr 24 2019, 15:29:51) [MSC v.1915 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.6.1 -- An enhanced Interactive Python.

In [1]:
    
```

The status bar at the bottom indicates: Permissions: RW, End-of-lines: CRLF, Encoding: UTF-8, Line: 1, Column: 1, Memory: 85 %.

# Part 1. 판다스 입문

## 4-1. 시리즈 연산

### • 시리즈 vs. 시리즈 (계속)

- 두 시리즈의 원소 개수가 다르거나, 혹은 시리즈의 크기가 같더라도 인덱스 값이 다를 수 있다. 이런 경우, 유효한 값이 존재하지 않는다는 의미로 NaN으로 처리한다.
- 한편, 같은 인덱스가 양쪽에 모두 존재하여 서로 대응되어도 어느 한 쪽의 데이터 값이 NaN인 경우가 있다. 이때, 연산의 대상인 데이터가 존재하지 않기 때문에, NaN으로 처리한다.

sr1		+	sr2		=	sr3	
index	value		index	value		index	value
A	10		?	?		A	NaN
B	20		B	10		B	30
C	30		C	20		C	50
D	NaN		D	30		D	NaN
?	?		E	40		E	NaN

[그림 1-18] NaN값이 있는 시리즈의 산술연산(덧셈)

### 예제 1-23

NaN값이 있는 시리즈 연산

(File: example/part1/1.23\_series\_to\_series2.py)

```
1 # -*- coding: utf-8 -*-
2
3 # 라이브러리 불러오기
4 import pandas as pd
5 import numpy as np
6
7 # 딕셔너리 데이터로 판다스 시리즈 만들기
8 student1 = pd.Series({'국어':np.nan, '영어':80, '수학':90})
9 student2 = pd.Series({'수학':80, '국어':90})
10
```

```
11 print(student1)
12 print('\n')
13 print(student2)
14 print('\n')
15
16 # 두 학생의 과목별 점수로 사칙연산 수행 (시리즈 vs 시리즈)
17 addition = student1 + student2      # 덧셈
18 subtraction = student1 - student2   # 뺄셈
19 multiplication = student1 * student2 # 곱셈
20 division = student1/student2        # 나눗셈
21 print(type(division))
22 print('\n')
23
24 # 사칙연산 결과를 데이터프레임으로 합치기 (시리즈 -> 데이터프레임)
25 result = pd.DataFrame([addition, subtraction, multiplication, division],
26                        index=['덧셈', '뺄셈', '곱셈', '나눗셈'])
27 print(result)
```

<실행 결과> 코드 전부 실행

```
국어      NaN
영어      80.0
수학      90.0
dtype: float64

수학      80
국어      90
dtype: int64
```

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

	국어	수학	영어
덧셈	NaN	170.000	NaN
뺄셈	NaN	10.000	NaN
곱셈	NaN	7200.000	NaN
나눗셈	NaN	1.125	NaN

The screenshot shows the Spyder Python IDE interface. The main editor window displays a Python script with the following code:

```

1 # -*- coding: utf-8 -*-
2
3 # 라이브러리 불러오기
4 import pandas as pd
5 import numpy as np
6
7 # 딕셔너리 데이터로 판다스 시리즈 만들기
8 student1 = pd.Series({'국어':np.nan, '영어':80, '수학':90})
9 student2 = pd.Series({'수학':80, '국어':90})
10
11 print(student1)
12 print('\n')
13 print(student2)
14 print('\n')
15
16 # 두 학생의 과목별 점수로 사칙연산 수행 (시리즈 vs. 시리즈)
17 addition = student1 + student2          #덧셈
18 subtraction = student1 - student2       #뺄셈
19 multiplication = student1 * student2    #곱셈
20 division = student1 / student2          #나눗셈
21 print(type(division))
22 print('\n')
23
24 # 사칙연산 결과를 데이터프레임으로 합치기 (시리즈 -> 데이터프레임)
25 result = pd.DataFrame([addition, subtraction, multiplication, division],
26                        index=['덧셈', '뺄셈', '곱셈', '나눗셈'])
27 print(result)
    
```

The Variable explorer on the right shows an empty table with columns Name, Type, Size, and Value.

The IPython console at the bottom shows the following output:

```

Python 3.7.3 (default, Apr 24 2019, 15:29:51) [MSC v.1915 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.6.1 -- An enhanced Interactive Python.

In [1]:
    
```

The status bar at the bottom indicates: Permissions: RW, End-of-lines: CRLF, Encoding: UTF-8, Line: 1, Column: 13, Memory: 84 %.

# Any Question?

Thank you.

