파이썬 머신러닝 판다스 데이터분석

Lecture (8)

Dr. Heesuk Kim

목차

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)

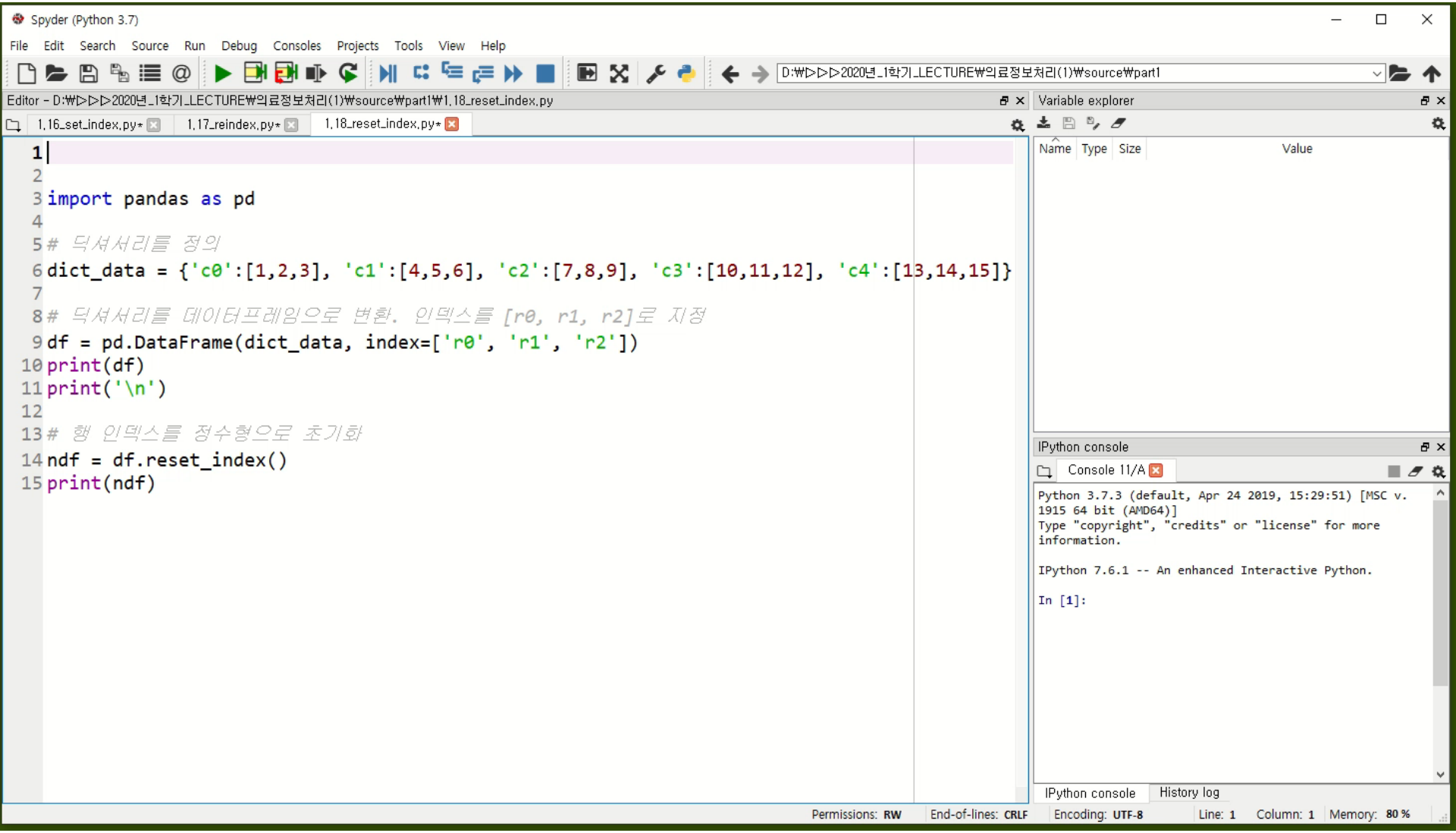
```
1  # -*- coding: utf-8 -*-
2
3  import pandas as pd
4
5  # 딕셔너리 정의
6  dict_data = {'c0':[1,2,3], 'c1':[4,5,6], 'c2':[7,8,9], 'c3':[10,11,12], 'c4':[13,14,15]}
7
8  # 딕셔너리를 데이터프레임으로 변환. 인덱스를 [r0, r1, r2]로 지정
9  df = pd.DataFrame(dict_data, index=['r0', 'r1', 'r2'])
10 print(df)
11 print('\n')
12
13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

Part 1. 판다스 입문 [화면 녹화]

예제 1-18



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)

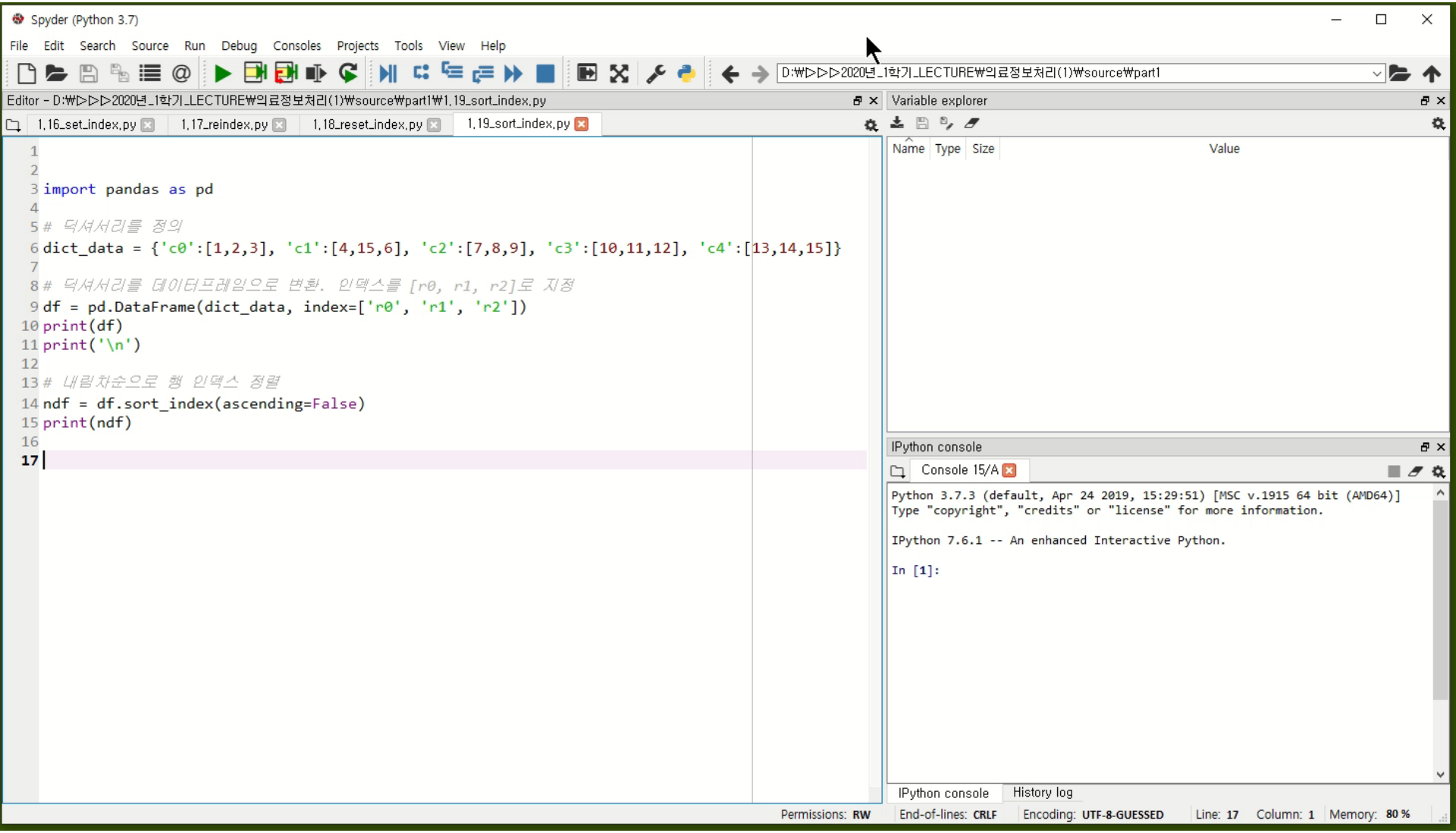
```
1  # -*- coding: utf-8 -*-
2
3  import pandas as pd
4
5  # 딕셔너리 정의
6  dict_data = {'c0':[1,2,3], 'c1':[4,5,6], 'c2':[7,8,9], 'c3':[10,11,12], 'c4':[13,14,15]}
7
8  # 딕셔너리를 데이터프레임으로 변환. 인덱스를 [r0, r1, r2]로 지정
9  df = pd.DataFrame(dict_data, index=['r0', 'r1', 'r2'])
10 print(df)
11 print('\n')
12
13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

Part 1. 판다스 입문 [화면 녹화]

예제 1-19



Part 1. 판다스 입문

4. 산술연산

- **판다스 객체의 산술연산은 내부적으로 3단계 프로세스를 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))
```

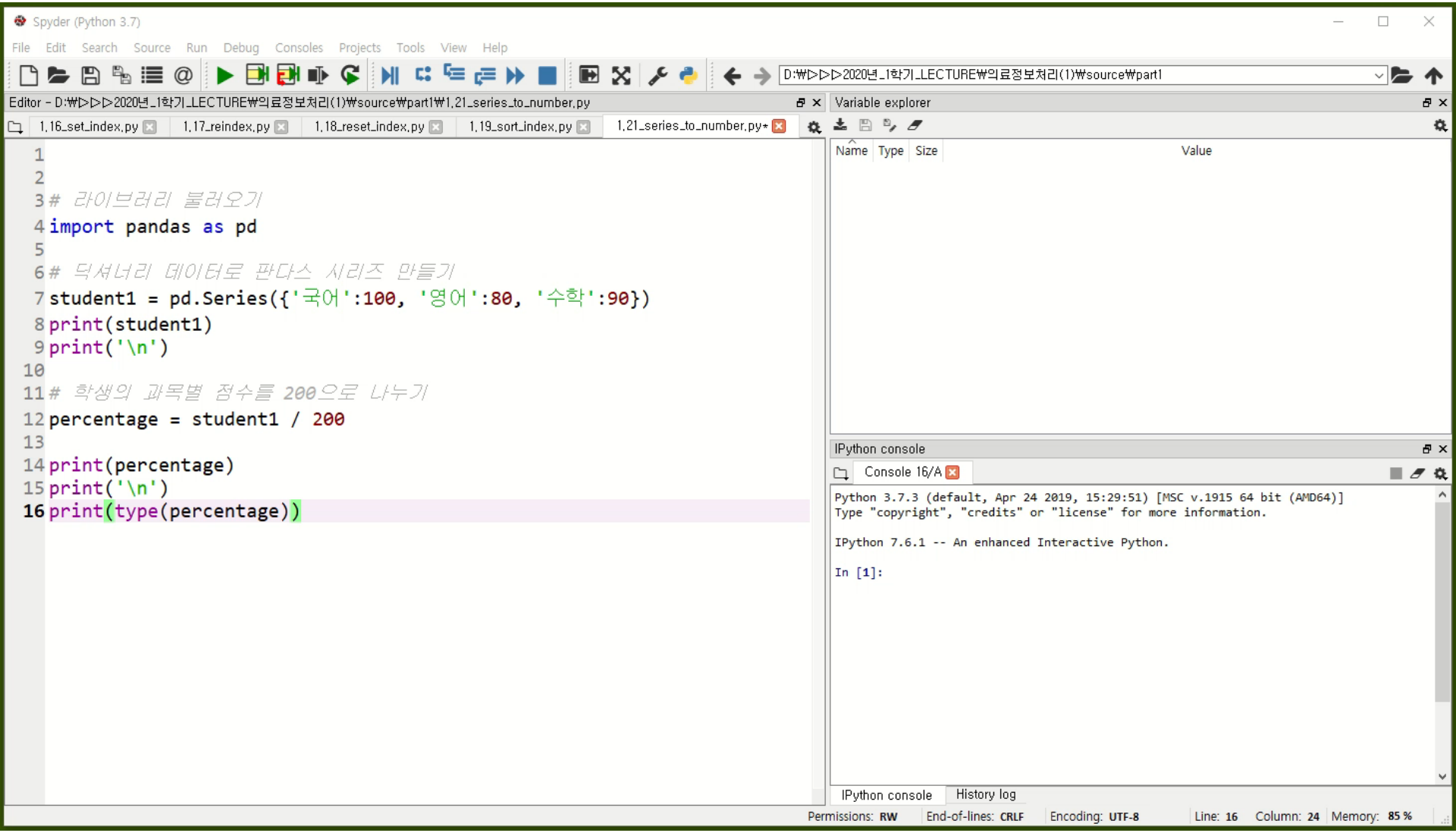
〈실행 결과〉 코드 전부 실행

```
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.

Part 1. 판다스 입문 [화면 녹화]

예제 1-21



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.

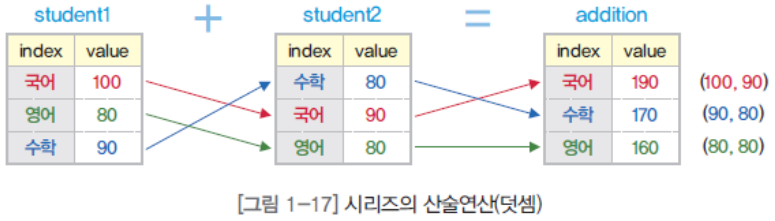
시리즈와 시리즈 연산: `Series1 + 연산자(+, -, *, /) + Series2`

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

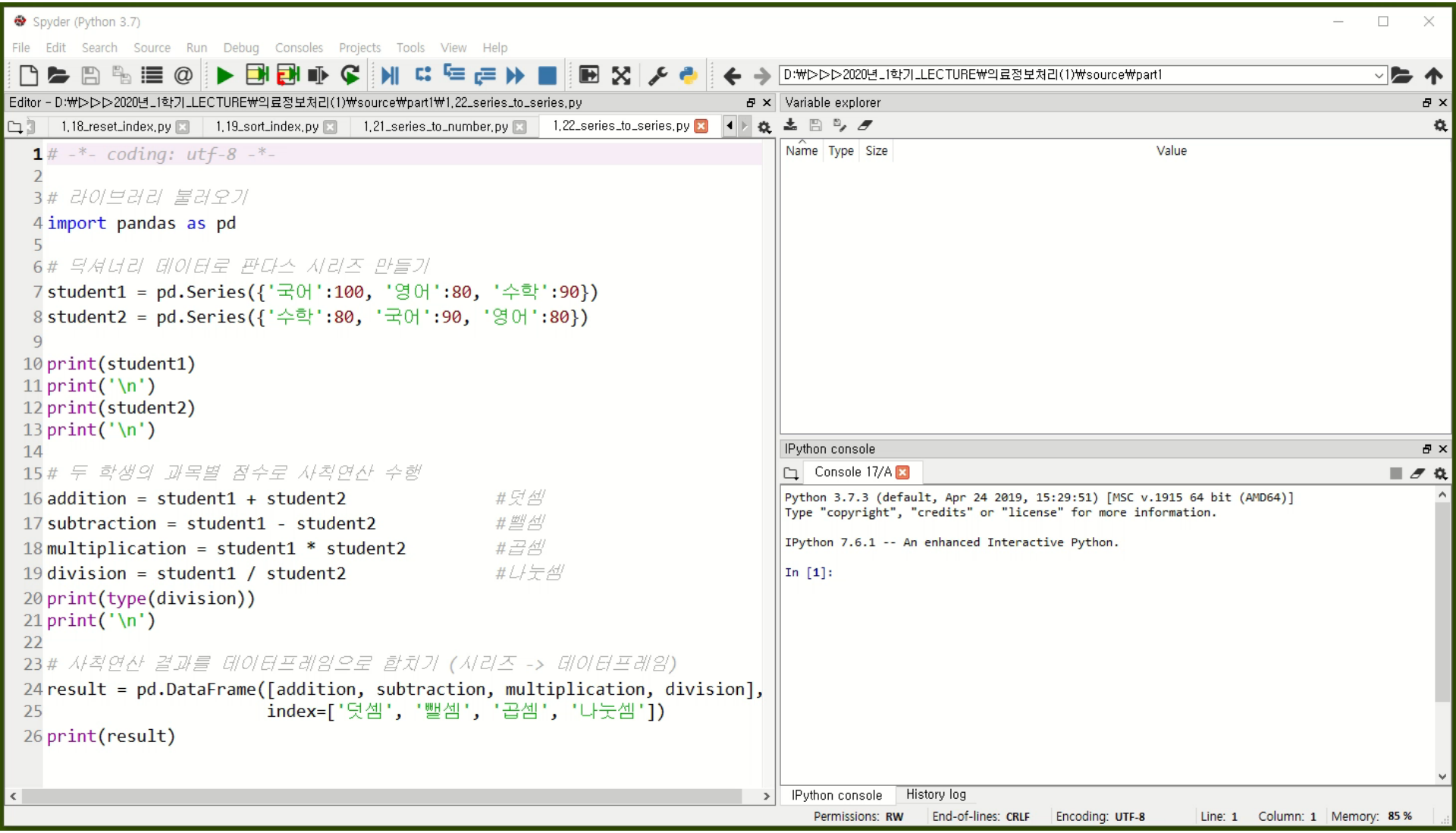
```
국어    100
영어     80
수학     90
dtype: int64


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)

Part 1. 판다스 입문 [화면 녹화]

예제 1-22

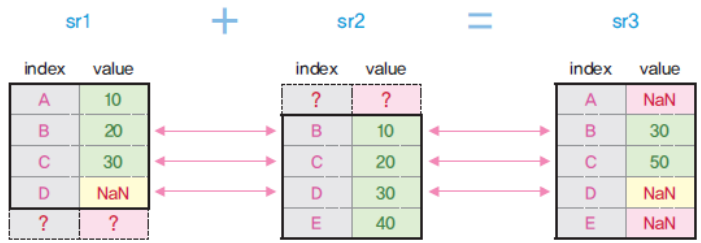


4-1. 시리즈 연산

- **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.



[그림 1-18] NaN값이 있는 시리즈의 산술연산(덧셈)

예제 1-23 NaN값이 있는 시리즈 연산 (File: example/part1/1.23_series_to_series2.py)

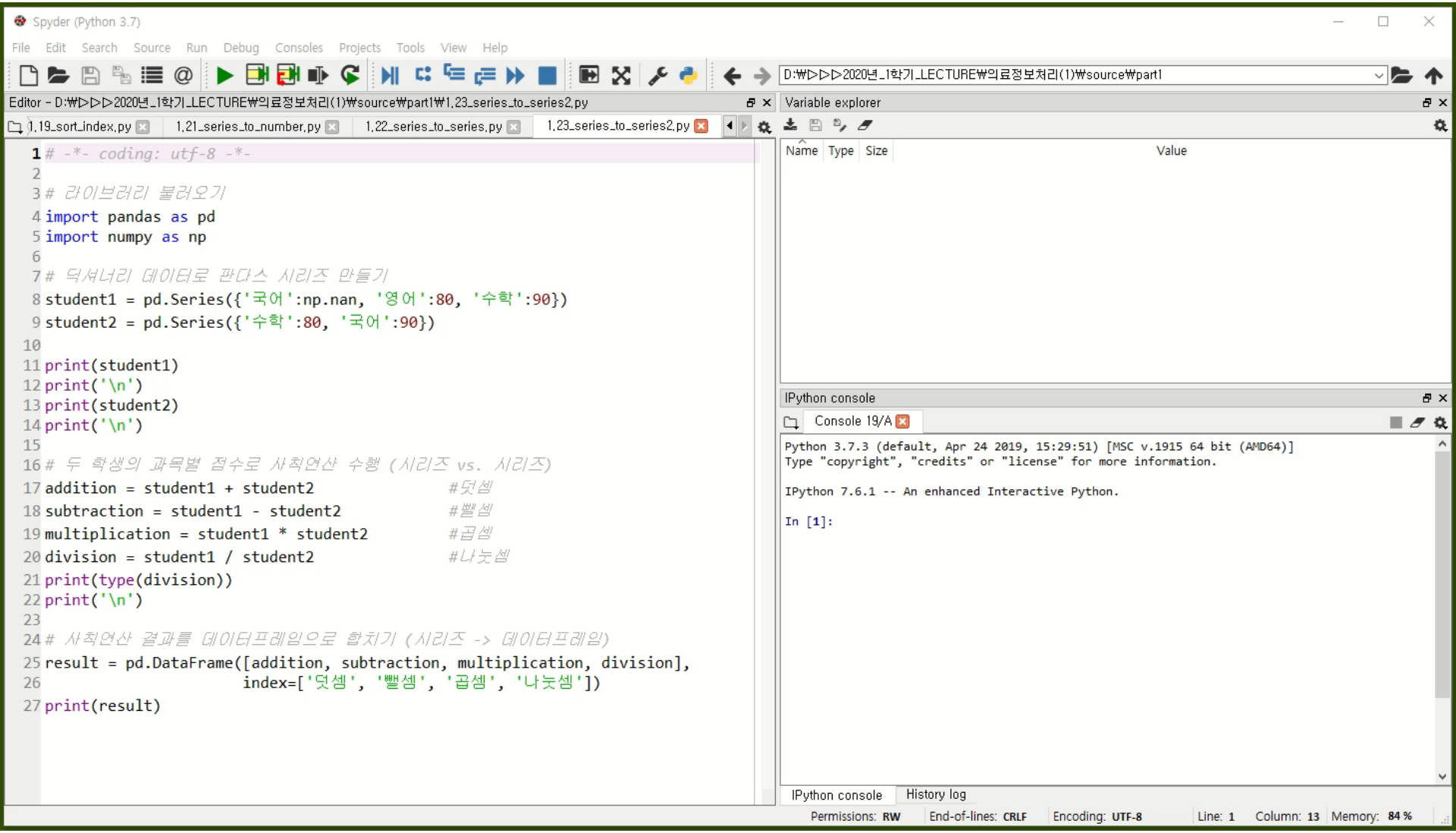
```
# -*- coding: utf-8 -*-

# 라이브러리 불러오기
import pandas as pd
import numpy as np

# 딕셔너리 데이터로 판다스 시리즈 만들기
student1 = pd.Series({'국어':np.nan, '영어':80, '수학':90})
student2 = pd.Series({'수학':80, '국어':90})

print(student1)
print('\n')
print(student2)
print('\n')

# 두 학생의 과목별 점수로 사칙연산 수행(시리즈 vs 시리즈)
addition = student1 + student2                # 덧셈
subtraction = student1 - student2             # 뺄셈
multiplication = student1 * student2          # 곱셈
division = student1/student2                  # 나눗셈
print(type(division))
print('\n')

# 사칙연산 결과를 데이터프레임으로 합치기(시리즈 -> 데이터프레임)
result = pd.DataFrame([addition, subtraction, multiplication, division],
                      index=['덧셈', '뺄셈', '곱셈', '나눗셈'])
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
```

Part 1. 판다스 입문 [화면 녹화]

예제 1-23



Any Question?

Thank you.