

빅데이터 실습

4주차 1차시

데이터프레임을 내 마음대로 변경하기

..... DataFrame 변경하기



새로운 컬럼 추가

- ① 고정 값 할당하기
- ② 기존 컬럼에 함수를 적용한 결과 할당하기
- ③ 기존 컬럼들의 산술 연산 결과 할당하기



불필요한 컬럼 or 데이터 삭제하기(Drop())

실습을 통한 컬럼 추가·삭제 이해하기

01

새로운 컬럼 추가 ①

[고정 값 할당하기]



새로운 컬럼 추가하기

① 컬럼 추가 후 값을 할당하는 방법

✓ 내가 추가하고자 하는 컬럼의 이름을 지정해 준 후 데이터 값을 할당해 줌

```
In [157]: # 컬럼 추가하기 1  
# limitStudent 컬럼(정원)을 추가하고, 값을 모두 30으로 저장  
df4['limitStudent'] = 30
```


Out[157]:

	Class	Year	Price	Location
C01	IoT	2018	100	Korea
C02	Network	2017	125	Korea
C03	Economy	2018	132	Korea
C04	Big Data	2018	312	US
C05	Cloud	2019	250	Korea

새로운 컬럼 추가하기

① 컬럼 추가 후 값을 할당하는 방법

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3



```
# limitStudent 컬럼(정원)을 추가하고, 값을 모두 30으로 저장
df4['limitStudent'] = 30
df4
```

Out[159]:

	Class	Year	Price	Location	limitStudent
C01	IoT	2018	100	Korea	30
C02	Network	2017	125	Korea	30
C03	Economy	2018	132	Korea	30
C04	Big Data	2018	312	US	30
C05	Cloud	2019	250	Korea	30

limitStudent라고 하는 컬럼을 추가하고, 추가된 컬럼의 값을 모두 30으로 할당하라는 의미가 됩니다.

In []: # 컬럼 추가하기 ?

새로운 컬럼 추가하기

① 컬럼 추가 후 값을 할당하는 방법

✓ 할당 값을 변경하고 싶을 때 동일한 구문을 똑같이 작성해 주면 됨

C04	Big Data	2018	312	US	30
C05	Cloud	2019	250	Korea	30

```
In [160]: df4['limitStudent'] = 40
df4
```

Out[160]:

	Class	Year	Price	Location	limitStudent
C01	IoT	2018	100	Korea	40
C02	Network	2017	125	Korea	40

데이터프레임의 특정 컬럼에 값을 할당하는 구문이 존재하는 컬럼이라면, 값이 업데이트되고, 존재하지 않는 컬럼이라면, 새로운 컬럼으로 추가하고 값을 할당됩니다.

새로운 컬럼 추가하기

② 추가된 컬럼에 서로 다른 값을 할당하는 방법

✓ 리스트의 형태로 값을 할당

C04	Big Data	2018	312	US	40
C05	Cloud	2019	250	Korea	40

```
In [161]: # 컬럼 추가하기 2.  
# numStudent 컬럼(수강학생수)을 추가하고, 값을 25,30,10,23,17로 저장  
df4
```

Out[161]:

	Class	Year	Price	Location	limitStudent
C01	IoT	2018	100	Korea	40
C02	Network	2017	125	Korea	40
C03	Economy	2018	132	Korea	40
C04	Big Data	2018	312	US	40

새로운 컬럼 추가하기

② 추가된 컬럼에 서로 다른 값을 할당하는 방법

✓ 값의 개수가 맞지 않을 경우 에러가 발생함

주의하기

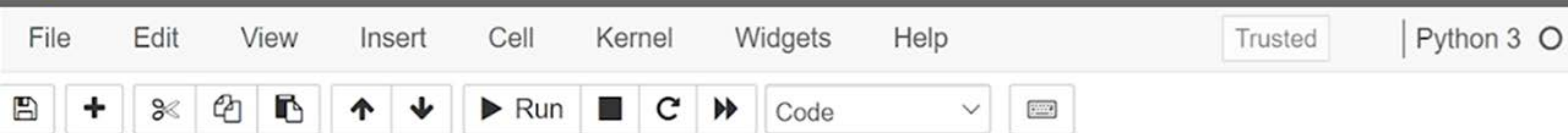
✓ 새로운 컬럼을 추가할 경우,
원본 데이터프레임의 행(row)의 개수와 신규로 할당되는 값의 개수가 같아야 함

```
ValueError                                Traceback (most recent call last)
<ipython-input-164-b5dc2d820b25> in <module>
      1 # 컬럼 추가하기 2.
      2 # numStudent 컬럼(수강학생수)을 추가하고, 값을 25,30,10,23,17로 저장
----> 3 df4['numStudent'] = [25, 30, 10, 23]
      4 df4
```

```
C:\WProgramData\Anaconda3\lib\site-packages\pandas\core\frame.py in __setitem__(self,
```

새로운 컬럼 추가하기

③ 기존 컬럼의 값을 활용하여 새로운 컬럼을 추가하기



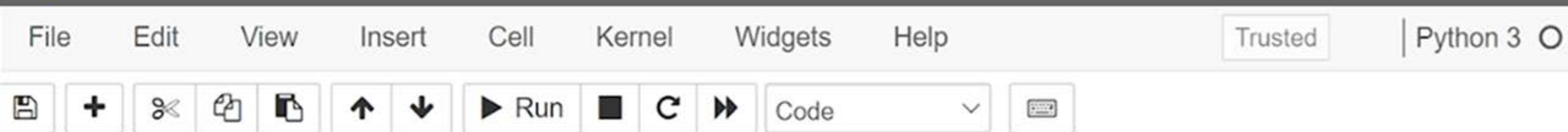
```
In [166]: # 컬럼 추가하기 3 (기존 컬럼에 함수를 적용한 결과로 값을 할당하기)
# priceLevel 컬럼을 추가하고,
# Price가 200과 같거나 크면, High, 200보다 작으면 Low
df4
```

Out [166]:

	Class	Year	Price	Location	limitStudent	numStudent
C01	IoT	2018	100	Korea	40	25
C02	Network	2017	125	Korea	40	30
C03	Economy	2018	132	Korea	40	10
C04	Big Data	2018	312	US	40	23
C05	Cloud	2019	250	Korea	40	17

새로운 컬럼 추가하기

③ 기존 컬럼의 값을 활용하여 새로운 컬럼을 추가하기



```
In [166]: # 컬럼 추가하기 3 (기존 컬럼에 함수를 적용한 결과로 값을 할당하기)
# priceLevel 컬럼을 추가하고,
# Price가 200과 같거나 크면, High, 200보다 작으면 Low
df4
```

Out [166]:

	Class	Year	Price	Location	limitStudent	numStudent
C01	IoT	2018	100	Korea	40	25
C02	Network	2017	125	Korea	40	30
C03	Economy	2018	132	Korea	40	10
C04	Big Data	2018	312	US	40	23

priceLevel에 대한 컬럼은 이 Price라고 하는 값을 기준으로 자동으로 변환이 되도록 하고 싶은 것입니다.

새로운 컬럼 추가하기

③ 기존 컬럼의 값을 활용하여 새로운 컬럼을 추가하기

- ✓ High, Low의 값으로 변환해주는 함수 정의하기
- ✓ 이 함수를 기존 컬럼에 적용한 결과로 새로운 컬럼 추가하기

```
In [166]: # 컬럼 추가하기 3 (기존 컬럼에 함수를 적용한 결과로 값을 할당하기)
# priceLevel 컬럼을 추가하고,
# Price가 200과 같거나 크면, High, 200보다 작으면 Low
df4
```

Out [166]:

	Class	Year	Price	Location	limitStudent	numStudent
C01	IoT	2018	100	Korea	40	25
C02	Network	2017	125	Korea	40	30
C03	Economy	2018	132	Korea	40	10
C04	Big Data	2018	312	US	40	23
C05	Cloud	2019	250	Korea	40	17

02

새로운 컬럼 추가 ②

[기존 컬럼에 함수를 적용한 결과 할당하기]



컬럼에 함수 적용하기

◎ 함수 생성 방법

Python 문법에 따라 def 키워드를 사용하여 함수명을 작성하면 됨

Trusted

Python 3

```
In [166]: # 컬럼 추가하기 3 (기존 컬럼에 함수를 적용한 결과로 값을 할당하기)
# priceLevel 컬럼을 추가하고,
# Price가 200과 같거나 크면, High, 200보다 작으면 Low

# 함수 생성
def get_plevel|
```

Out [166]:

	Class	Year	Price	Location	limitStudent	numStudent
C01	IoT	2018	100	Korea	40	25
C02	Network	2017	125	Korea	40	30
C03	Economy	2018	132	Korea	40	10
C04	Big Data	2018	312	US	40	23

컬럼에 함수 적용하기

◎ 함수 생성 방법

➤ Get plevel 함수 생성

```
# 함수 생성  
def get_plevel(X):  
    if X >= 200:  
        return 'High'  
    else:  
        return 'Low'
```

Get_plevel(250) ← High로 리턴

Get_plevel(150) ← Low로 리턴

컬럼에 함수 적용하기

◎ 함수 생성 방법

02. 함수의 input 값, 인자값을 작성함

```
if X >= 200:  
    return 'High'  
else:  
    return 'Low'
```

In [170]: df4

Out[170]:

	Class	Year	Price	Location	limitStudent	numStudent
C01	IoT	2018	100	Korea	40	25
C02	Network	2017	125	Korea	40	30
C03	Economy	2018	132	Korea	40	10

이러한 함수를 만든 이유는 Price 컬럼의 각각의 값에 함수를 적용하고 싶기 때문입니다.

컬럼에 함수 적용하기

◎ apply() 함수 적용하기

apply() 함수

함수를 적용하기 위한 함수

```
return 'High'  
else:  
    return 'Low'
```

In [170]: `# apply() 함수 --> 함수를 적용하기 위한 함수`

Out[170]:

	Class	Year	Price	Location	limitStudent	numStudent
C01	IoT	2018	100	Korea	40	25
C02	Network	2017	125	Korea	40	30
C03	Economy	2018	132	Korea	40	10
C04	Big Data	2018	312	US	40	23

컬럼에 함수 적용하기

◎ apply() 함수 적용하기

예

df4에 sum() 함수 실행하기

```
In [171]: # apply() 함수 --> 함수를 적용하기 위한 함수  
df4.sum()
```

```
Out[171]: Class          IoTNetworkEconomyBig DataCloud  
Year              1          10090  
Price              919  
Location          KoreaKoreaKoreaUSKorea  
limitStudent      200  
numStudent         105  
dtype: object
```

sum() 함수

더하기 함수

각 컬럼별로 더하기
연산을 수행함

sum()를 통해 더하기 연산이 수행된 문자열이 출력됨

컬럼에 함수 적용하기

◎ apply() 함수 적용하기

예

df4에 sum() 함수 실행하기

```
In [171]: # apply() 함수 -> 함수를 적용하기 위한 함수  
df4.sum()
```

```
Out[171]: Class      IoTNetworkEconomyBig DataCloud  
Year              10090  
Price              010  
Location      KoreaKoreaKoreaUSKorea  
limitStudent      200  
numStudent         105  
dtype: object
```

문자열(String)의 더하기 연산 = 문자열 연결(concatenate)

컬럼에 함수 적용하기

◎ apply() 함수 적용하기

✓ 컬럼 단위로 이 sum() 함수가 적용이 된 것임

```
return 'Low'
```

```
In [171]: # apply() 함수 -> 함수를 적용하기 위한 함수  
df4.sum()
```

```
Out[171]: Class          IoTNetworkEconomyBig DataCloud  
Year                  10090  
Price                  919  
Location              KoreaKoreaKoreaUSKorea  
limitStudent          200  
numStudent             105  
dtype: object
```

```
In [ ]:
```


컬럼에 함수 적용하기

◎ apply() 함수 적용하기

✓ sum() 함수를 적용하기 위한 apply() 함수를 쓸 수 있음

```
return 'Low'
```

```
In [173]: # apply() 함수 --> 함수를 적용하기 위한 함수  
#df4.sum()  
df4.apply('sum')
```

```
Out[173]: Class          IoTNetworkEconomyBig DataCloud  
Year                10090  
Price                919  
Location            KoreaKoreaKoreaUSKorea  
limitStudent        200  
numStudent           105  
dtype: object
```

sum() 함수를 적용해도 되지만 apply() 함수를 사용해도 동일하게 동작되는 방식입니다.

```
In [ ]:
```

컬럼에 함수 적용하기

◎ apply() 함수 적용하기

✓ df4에 apply()한 후 실행하면 컬럼 단위로 sum() 함수가 적용됨

```
return 'Low'
```

```
In [174]: # apply() 함수 --> 함수를 적용하기 위한 함수
#df4.sum()
#df4.apply('sum')
df4.apply(lambda X: X.sum()) ← X값을 인자로 받아 X에 sum()를 적용한 것임
```

```
Out[174]: Class          IoTNetworkEconomyBig DataCloud
Year                    10090
Price                   919
Location                KoreaKoreaKoreaUSKorea
limitStudent            200
numStudent               105
```

각각의 컬럼들을 순회하면서 각각의 컬럼들에 sum()이라고 하는 함수를 적용하는 것입니다.

컬럼에 함수 적용하기

◎ apply() 함수 적용하기

✓ 컬럼별로 순회하면서 sum() 함수를 수행하게 됨

```
return 'Low'
```

```
In [174]: # apply() 함수 --> 함수를 적용하기 위한 함수  
#df4.sum()  
#df4.apply('sum')  
df4.apply(lambda X: X.sum())
```

```
Out[174]: Class          IoTNetworkEconomyBig DataCloud  
Year                    10090  
Price                    919  
Location                KoreaKoreaKoreaUSKorea  
limitStudent            200  
numStudent               105  
dtype: object
```

컬럼에 함수 적용하기

◎ Price에 컬럼에 get_plevel() 함수를 적용

✓ [df4.apply('get_plevel')]을 적용하면 에러가 발생함

```
Price          919
Location      KoreaKoreaKoreaUSKorea
limitStudent   200
numStudent     105
dtype: object
```

```
In [176]: df4.apply('get_plevel')
```

AttributeError

Traceback (most recent call last)

```
<ipython-input-176-393fe2929ebc> in <module>
----> 1 df4.apply('get_plevel')
```

```
C:\WProgramData\Anaconda3\lib\site-packages\pandas\core\frame.py in apply(self, func, axis, raw, result_type, args, **kwargs)
    7516         kwds=kwargs
```


컬럼에 함수 적용하기

◎ Price에 컬럼에 get_plevel() 함수를 적용

```
In [174]: # apply() 함수 --> 함수를 적용하기 위한 함수  
#df4.sum()  
#df4.apply('sum')  
df4.apply(lambda X: X.sum())
```

컬럼 단위로 순회하면서 lambda 함수의 입력값이 됨

컬럼에 함수 적용하기

◎ Price에 컬럼에 get_plevel() 함수를 적용

✓ 어떤 특정 컬럼의 각각의 값에 새로운 값들을 할당하고 싶을 경우 익명 함수를 적용하는 것임

```
Out[174]: Class          IoTNetworkEconomyBig DataCloud
Year                  10090
Price                 919
Location              KoreaKoreaKoreaUSKorea
limitStudent          200
numStudent            105
dtype: object
```

```
In [176]: df4.apply(lambda X: pr|
```

AttributeError

Traceback (most recent call last)

<ipython-input-176-393fe2929ebc> in <module>

----> 1 df4.apply('get_plevel')

컬럼에 함수 적용하기

◎ Price에 컬럼에 get_plevel() 함수를 적용

✓ 컬럼 단위로 X값이 할당되어 순회함

```
limitStudent      200
numStudent         105
dtype: object
```

```
In [177]: df4.apply(lambda X: print(X))
```

```
C01      IoT
C02    Network
C03    Economy
C04    Big Data
C05      Cloud
Name: Class, dtype: object
C01      2018
C02      2017
C03      2018
C04      2018
```

컬럼에 함수 적용하기

◎ Price에 컬럼에 get_plevel() 함수를 적용

✓ 로우 단위로 순회하고 싶은 경우, axis 인자를 활용함

```
#df4.apply('sum')  
df4.apply(lambda X: X.sum())
```

```
Out[174]: Class          IoTNetworkEconomyBig DataCloud  
Year                10090  
Price                919  
Location            KoreaKoreaKoreaUSKorea  
limitStudent        200  
numStudent          105  
dtype: object
```

```
In [178]: df4.apply(lambda X: print(X), axis = 1)
```

```
Class          IoT  
Year           2018  
Price          100
```


컬럼에 함수 적용하기

◎ Price에 컬럼에 get_plevel() 함수를 적용

✓ 예 [axis = 1]의 경우 로우 단위로 순회함

```
limitStudent 200
numStudent    105
dtype: object
```

```
In [178]: df4.apply(lambda X: get_plevel(), axis = 1)
```

```
Class      IoT
Year      2018
Price      100
Location   Korea
limitStudent 40
numStudent 25
Name: C01, dtype: object
```

```
Class      Network
Year      2017
Price      125
```

← 첫 번째 X

컬럼에 함수 적용하기

◎ Price에 컬럼에 get_plevel() 함수를 적용

✓ X의 Price 값을 인자로 전달함

```
limitStudent      200  
numStudent         105  
dtype: object
```

```
In [179]: df4.apply(lambda X: get_plevel(X.Price), axis = 1)
```

```
Out[179]: C01      Low  
          C02      Low  
          C03      Low  
          C04     High  
          C05     High  
          dtype: object
```

```
In [ ]:
```


컬럼에 함수 적용하기

◎ Price에 컬럼에 get_plevel() 함수를 적용

✓ 이에 대한 결과를 df4의 새로운 컬럼으로 추가하면 됨

In [181]: df4

Out[181]:

	Class	Year	Price	Location	limitStudent	numStudent	priceLevel
C01	IoT	2018	100	Korea	40	25	Low
C02	Network	2017	125	Korea	40	30	Low
C03	Economy	2018	132	Korea	40	10	Low
C04	Big Data	2018	312	US	40	23	High
C05	Cloud	2019	250	Korea	40	17	High

In []: # 컬럼 추가하기 4 (기존 컬럼을 이용하여 새 컬럼 추가하기)

03

새로운 컬럼 추가 ③

[기존 컬럼들의 산술 연산 결과 할당하기]



새로운 컬럼 추가하기

© income(수입) 컬럼 추가하기

✓ 가격 (Price)와 학생수(numStudent)를 곱한 값을 할당

Trusted

Python 3

C05	Cloud	2019	250	Korea	40	17	High
-----	-------	------	-----	-------	----	----	------

```
In [182]: # 컬럼 추가하기 4 (기존 컬럼을 이용하여 새 컬럼 추가하기)
# income 컬럼을 추가하고, Price와 numStudent의 값을 곱한 값으로 할당
df4
```

Out [182]:

	Class	Year	Price	Location	limitStudent	numStudent	priceLevel
C01	IoT	2018	100	Korea	40	25	Low
C02	Network	2017	125	Korea	40	30	Low
C03	Economy	2018	132	Korea	40	10	Low
C04	Big Data	2018	312	US	40	23	High
C05	Cloud	2019	250	Korea	40	17	High

산술 연산 이해하기 두 값의 연산 수행

Jupyter [W3] 기초 2. Pandas 개요 및 데이터 타입 소개 ... (unsaved changes) Python 3 Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted



Python 3

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

C02	Network	2017	125	Korea	40	30	Low	3750
C03	Economy	2018	132	Korea	40	10	Low	1320
C04	Big Data	2018	312	US	40	23	High	7176
C05	Cloud	2019	250	Korea	40	17	High	4250

In [186]: # 산술 연산

In [187]: #1 두 값의 연산

5+10

File "<ipython-input-187-744ab3892b9a>", line 2

5+10

^

IndentationError: unexpected indent

◎ 시리즈 생성 후 100을 더할 경우

✓ 시리즈의 각 항목 값에 100씩 더해짐

Out[188]: 15

```
In [189]: #2. 1차원 데이터와 값 간의 연산  
sr = Series([3,5,7,9])
```

```
In [190]: sr + 100
```

```
Out[190]: 0    103  
         1    105  
         2    107  
         3    109  
         dtype: int64
```

```
In [ ]:
```

sr2020

삼성	4500
롯데	2800
LG	3000

+

sr2021

삼성	4800
롯데	3500
SK	2500

=

삼성	9300
롯데	6300
LG	NaN
SK	NaN

1차원 데이터들 간의 산술 연산은
같은 인덱스 라벨의 값들끼리 산술연산이 수행됩니다.

sr2020

삼성	4500
롯데	2800
LG	3000

+

sr2021

삼성	4800
롯데	3500
SK	2500

=

삼성	9300
롯데	6300
LG	NaN
SK	NaN

add() 함수를 통한 연산 수행 + fill_value 인자 사용
= sr2020.add(sr2021, fill_value = 0)

한쪽이 없는 경우 NaN값이 아니라 0으로 처리해서 산술 연산을 수행할 수가 있습니다.

산술 연산 이해하기 1차원 데이터들 간의 연산

Jupyter [W3] 기초 2. Pandas 개요 및 데이터 타입 소개 ... (unsaved changes) Logout

시리즈 간의 산술연산을 수행할 경우

인덱스 라벨이 같은 항목들끼리 산술 연산이 수행됨

```
sr2021 = Series([4500, 3500, 2500], index = [ '삼성' , '롯데' , 'SK' ])
```

```
In [196]: #sr2020 + sr2021  
sr2020.add(sr2021, fill_value = 0)
```

```
Out[196]: LG      3000.0  
          SK       2500.0  
          롯데    6300.0  
          삼성    9000.0  
          dtype: float64
```

```
In [193]: sr2021
```

```
Out[193]: 삼성      4500  
          롯데      3500  
          SK        2500
```


➤ 산술 연산 함수

+	add()
-	sub()
X	mul()
/	div()

산술 연산 실습하기

◎ 총원율을 계산해 새로운 컬럼에 추가하기

✓ [limitStudent]와 [numStudent]의 기존 컬럼 값을 가지고 할당할 수 있음

```
SK      NaN
롯데   700.0
삼성     0.0
dtype: float64
```

```
In [198]: # 실습. rate 컬럼을 추가하고, 총원율을 계산하여 할당.
# 총원율은 정원(limitStudent) 대비 수강학생수(numStudent) 비율
df4
```

Out[198]:

	Class	Year	Price	Location	limitStudent	numStudent	priceLevel	income
C01	IoT	2018	100	Korea	40	25	Low	2500
C02	Network	2017	125	Korea	40	30	Low	3750
C03	Economy	2018	132	Korea	40	10	Low	1320

산술 연산 실습하기

◎ 총원율을 계산해 새로운 컬럼에 추가하기

✓ [numStudent]의 컬럼 값을 [limitStudent]로 나누면 됨

```
df4['rate'] = df4.numStudent/df4.limitStudent*100
```

각각의 강의에 해당하는

정원수와 수강 학생 수를 기반으로 계산된 것

In [202]: df4

Out[202]:

	Class	Year	Price	Location	limitStudent	numStudent	priceLevel	income	rate
C01	IoT	2018	100	Korea	40	25	Low	2500	62.5
C02	Network	2017	125	Korea	40	30	Low	3750	75.0
C03	Economy	2018	132	Korea	40	10	Low	1320	25.0
C04	Big Data	2018	312	US	40	23	High	7176	57.5
C05	Cloud	2019	250	Korea	40	17	High	4250	42.5

산술 연산 실습하기

◎ 총원율을 계산해 새로운 컬럼에 추가하기

✓ 총원율을 반올림 하고 싶은 경우 [round() 함수] 사용

```
df4['rate'] = df4.numStudent/df4.limitStudent*100
```

In [202]: df4

Out[202]:

	Class	Year	Price	Location	limitStudent	numStudent	priceLevel	income	rate
C01	IoT	2018	100	Korea	40	25	Low	2500	62.5
C02	Network	2017	125	Korea	40	30	Low	3750	75.0
C03	Economy	2018	132	Korea	40	10	Low	1320	25.0
C04	Big Data	2018	312	US	40	23	High	7176	57.5
C05	Cloud	2019	250	Korea	40	17	High	4250	42.5

0이라고 하는 것은 소수점 첫째 자리에서 반올림해서 정수의 형태로 변환을 해 달라고 하는 것입니다.

산술 연산 실습하기

◎ 총원율을 계산해 새로운 컬럼에 추가하기

✓ 소수점 몇 번째 자리에서 반올림할 것인지를 인자값으로 주면 됨

```
df4['rate'] = df4.numStudent / df4.limitStudent * 100
```

```
In [203]: df4.round(0)
```

Out[203]:

	Class	Year	Price	Location	limitStudent	numStudent	priceLevel	income	rate
C01	IoT	2018	100	Korea	40	25	Low	2500	62.0
C02	Network	2017	125	Korea	40	30	Low	3750	75.0
C03	Economy	2018	132	Korea	40	10	Low	1320	25.0
C04	Big Data	2018	312	US	40	23	High	7176	57.0
C05	Cloud	2019	250	Korea	40	17	High	4250	42.0

산술 연산 실습하기

◎ 데이터프레임의 rate라고 하는 컬럼만 소수점 몇 번째 자리로 하고 싶을 경우

✓ key value의 형태로 지정해주면 됨

```
df4['rate'] = df4.numStudent / df4.limitStudent * 100
```

```
In [204]: #df4.round(0)
df4.round({'rate': 0})
```

Out[204]:

	Class	Year	Price	Location	limitStudent	numStudent	priceLevel	income	rate
C01	IoT	2018	100	Korea	40	25	Low	2500	62.0
C02	Network	2017	125	Korea	40	30	Low	3750	75.0
C03	Economy	2018	132	Korea	40	10	Low	1320	25.0
C04	Big Data	2018	312	US	40	23	High	7176	57.0
C05	Cloud	2019	250	Korea	40	17	High	4250	42.0

04

불필요한 컬럼 or 데이터 삭제하기(Drop())



데이터 삭제하기

drop() 함수

- ◎ 첫 번째 인자: 삭제하고자 하는 인덱스
- ◎ 두 번째 인자: 축(0 or 1)



2. 불필요한 컬럼이나 데이터 삭제하기

```
In [ ]: # drop()  
# 첫번째 인자: 삭제하고자 하는 인덱스명  
# 두번째 인자: axis (0 | )
```

```
In [ ]:
```


① C05 과목 삭제하기

- ✓ 첫번째 인자: 'C05' 인덱스 지정
- ✓ 두번째 인자: 0으로 지정 (로우 인덱스를 의미)

```
# 첫번째 인자: 삭제하고자 하는 인덱스명  
# 두번째 인자: axis (0 or 1)
```

```
In [205]: # C05 강의 삭제  
df4.drop('C05', 0)
```

Out[205]:

	Class	Year	Price	Location	limitStudent	numStudent	priceLevel	income	rate
C01	IoT	2018	100	Korea	40	25	Low	2500	62.5
C02	Network	2017	125	Korea	40	30	Low	3750	75.0
C03	Economy	2018	132	Korea	40	10	Low	1320	25.0
C04	Big Data	2018	312	US	40	23	High	7176	57.5
C05	Cloud	2019	250	Korea	40	17	High	4250	42.5

② priceLevel 컬럼 삭제하기

- ✓ 첫번째 인자: priceLevel' 인덱스 지정
- ✓ 두번째 인자: 1로 지정 (컬럼 인덱스를 의미)

	Class	Year	Price	Location	limitStudent	numStudent	priceLevel	income	rate
C01	IoT	2018	100	Korea	40	25	Low	2500	62.5
C02	Network	2017	125	Korea	40	30	Low	3750	75.0
C03	Economy	2018	132	Korea	40	10	Low	1320	25.0
C04	Big Data	2018	312	US	40	23	High	7176	57.5

```
In [ ]: # priceLevel 컬럼 삭제  
df4.drop('priceLevel', 1)
```


데이터 삭제하기

drop() 함수

◎ 함수를 실행한 결과가 output에 출력이 됨

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

OUT[206]:

	Class	Year	Price	Location	limitStudent	numStudent	priceLevel	income	rate
C01	IoT	2018	100	Korea	40	25	Low	2500	62.5
C02	Network	2017	125	Korea	40	30	Low	3750	75.0
C03	Economy	2018	132	Korea	40	10	Low	1320	25.0
C04	Big Data	2018	312	US	40	23	High	7176	57.5

```
In [207]: # priceLevel 컬럼 삭제  
df4.drop('priceLevel', 1)
```

Out[207]:

	Class	Year	Price	Location	limitStudent	numStudent	income	rate
C01	IoT	2018	100	Korea	40	25	2500	62.5

데이터 삭제하기

drop() 함수

◎ 출력은 되지만 실제로 원본 데이터에는 함수에 대한 결과가 반영되지 않음

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Code

C04 Big Data 2018 312 US 40 23 7176 57.5

```
In [207]: # priceLevel 컬럼 삭제  
df4.drop('priceLevel', 1)
```

Out[207]:

	Class	Year	Price	Location	limitStudent	numStudent	income	rate
C01	IoT	2018	100	Korea	40	25	2500	62.5
C02	Network	2017	125	Korea	40	30	3750	75.0
C03	Economy	2018	132	Korea	40	10	1320	25.0
C04	Big Data	2018	312	US	40	23	7176	57.5
C05	Cloud	2019	250	Korea	40	17	4250	42.5

◎ 원본을 변경하는 방법

01. 인자를 하나 더 추가하기

✓ inplace 인자를 False → True로 변경해주면 원본 데이터에 반영됨

Out[207]:

	Class	Year	Price	Location	limitStudent	numStudent	income	rate
C01	IoT	2018	100	Korea	40	25	2500	62.5
C02	Network	2017	125	Korea	40	30	3750	75.0
C03	Economy	2018	132	Korea	40	10	1320	25.0
C04	Big Data	2018	312	US	40	23	7176	57.5
C05	Cloud	2019	250	Korea	40	17	4250	42.5

```
In [ ]: df4.drop('priceLevel', 1, inplace = |)
```

◎ 원본을 변경하는 방법

02. 원본에 함수 결과를 할당하기

```
In [208]: df4.drop('priceLevel', 1, inplace = True)
```

```
In [209]: df4.drop('income', 1)
```

```
Out[209]:
```

	Class	Year	Price	Location	limitStudent	numStudent	income	rate
C01	IoT	2018	100	Korea	40	25	2500	62.5
C02	Network	2017	125	Korea	40	30	3750	75.0
C03	Economy	2018	132	Korea	40	10	1320	25.0
C04	Big Data	2018	312	US	40	23	7176	57.5

◎ 동시에 여러 개를 삭제하는 방법

✓ 인덱스를 첫 번째 인자로 지칭해줄 때, 리스트로 담아서 전달해주면 됨

Trusted

Python 3

```
In [208]: df4.drop('priceLevel', 1, inplace = True)
```

```
In [211]: df4 = df4.drop('income', 1)
```

```
In [212]: df4
```

Out[212]:

	Class	Year	Price	Location	limitStudent	numStudent	rate
C01	IoT	2018	100	Korea	40	25	62.5
C02	Network	2017	125	Korea	40	30	75.0
C03	Economy	2018	132	Korea	40	10	25.0
C04	Big Data	2018	312	US	40	23	57.5

데이터 삭제하기

수강생이 가장 작은 클래스를 찾아 제거하기

◎ 조건 색인을 통해 삭제하는 구문 실행하기

✓ 가장 작은 클래스가 무엇인지 알고 있을 경우 [axis = 0]

Trusted



Python 3

C05	Cloud	2019	250	Korea	40	17	42.5
-----	-------	------	-----	-------	----	----	------

```
In [213]: # 실습. 수강생이 가장 작은 클래스를 찾아서 삭제  
df4.drop('C03', )
```

Out [213]:

	Class	Year	Price	Location	limitStudent	numStudent	rate
C01	IoT	2018	100	Korea	40	25	62.5
C02	Network	2017	125	Korea	40	30	75.0
C03	Economy	2018	132	Korea	40	10	25.0
C04	Big Data	2018	312	US	40	23	57.5
C05	Cloud	2019	250	Korea	40	17	42.5

데이터 삭제하기

수강생이 가장 작은 클래스를 찾아 제거하기

◎ 조건 색인을 통해 삭제하는 구문 실행하기

✓ 가장 작은 클래스를 모를 경우 [df4.numStudent.min()]

```
In [218]: ##### 실습. 수강생이 가장 작은 클래스를 찾아서 삭제
df4[df4.numStudent == df4.numStudent.min()]
```

Out[218]:

	Class	Year	Price	Location	limitStudent	numStudent	rate	
	C03	Economy	2018	132	Korea	40	10	25.0

```
In [ ]: df4.drop('C03', 0)
```

del_class

내가 삭제하고자 하는 Class에 대한 정보를 찾을 수 있음

◎ 수강생이 가장 작은 클래스를 찾아 제거하기

✓ del_class에 index를 실행해보면 C03번이라는 것만 얻을 수가 있음

```
In [219]: ##### 실습. 수강생이 가장 작은 클래스를 찾아서 삭제
del_class = df4[df4.numStudent == df4.numStudent.min()]
del_class
```

Out[219]:

	Class	Year	Price	Location	limitStudent	numStudent	rate
C03	Economy	2018	132	Korea	40	10	25.0

```
In [ ]: df4.drop('C03', 0)
```


데이터 삭제하기

수강생이 가장 작은 클래스를 찾아 제거하기

© del_class.index를 C03번 대신에 바꿔 주면 동일하게 C03번을 삭제하게 됨

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3



Run



Code



del_class.index

Out[220]: Index(['C03'], dtype='object')

In [221]: df4.drop(del_class.index, 0)

Out[221]:

	Class	Year	Price	Location	limitStudent	numStudent	rate
C01	IoT	2018	100	Korea	40	25	62.5
C02	Network	2017	125	Korea	40	30	75.0
C04	Big Data	2018	312	US	40	23	57.5
C05	Cloud	2019	250	Korea	40	17	42.5

..... DataFrame 변경하기



새로운 컬럼 추가

- ① 고정 값 할당하기
- ② 기존 컬럼에 함수를 적용한 결과 할당하기
- ③ 기존 컬럼들의 산술 연산 결과 할당하기



불필요한 컬럼 or 데이터 삭제하기(Drop())



수고하셨습니다!