PART1_1_32202546_안지성(2021.09.16)

September 16, 2021

1 Series

1.0.1 1-1 딕셔너리를 시리즈로 변환

k:v' 구조를 갖는 딕셔너리를 정의하여 변수 dict_data에 저장한다. 변수 dict_data에 저장되어 있는 딕셔너리를 Series() 함수의 인자로 전달하면, 시리즈로 변환한다. Series() 함수가 반환한 시리즈 객체를 변수 sr에 저장한다.

[45]:
```
# pandas 불러오기
import pandas as pd
```

[46]:
```
# key;value 쌍으로 딕셔너리를 만들고, 변수 dict_data에 저장
dict_data = {'a': 1, 'b': 2, 'c': 3}

# 판다스 Series() 함수로 dictionary를 Series로 변환, 변수 sr에 저장
sr = pd.Series(dict_data)

# sr의 자료형 출력
print(type(sr))
```

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

[47]:
```
# 변수 sr에 저장되어 있는 시리즈 객체들 출력
print(sr)
```

```
a    1
b    2
c    3
dtype: int64
```

1.0.2 1-2 리스트를 시리즈로 변환

① 시리즈 인덱스 파이썬 리스트를 시리즈로 변환해 본다. 단, 딕셔너리의 키처럼 인덱스로 변환될 값이 없다. 따라서, 인덱스를 별도로 정의하지 않으면, 디폴트로 정수형 위치 인덱스 (0,1,2, ...)가 자동 지정된다. 밑의 코드에서는 0 ~ 4 범위의 정수값이 인덱스로 지정된다.

[48]:
```
# 리스트를 시리즈로 변환하여 변수 sr에 저장
list_data = ['2021-09-16', 3.14, 'ABC', 100, True]

sr = pd.Series(list_data)
```

```
print(sr)
```

```
0    2021-09-16
1          3.14
2           ABC
3           100
4          True
dtype: object
```

② 인덱스 vs. 데이터 값 배열 확인하기 시리즈의 index 속성과 values 속성을 이용하면, 인덱스 배열과 데이터 값의 배열을 불러올 수 있다.

[49]:
```
idx = sr.index
val = sr.values
print(idx)
```

```
RangeIndex(start=0, stop=5, step=1)
```

[50]:
```
print(val)
```

```
['2021-09-16' 3.14 'ABC' 100 True]
```

1.0.3 1-3 튜플을 시리즈로 변환

①시리즈 생성 정수형 위치 인덱스 대신에 인덱스 이름을 따로 지정할 수 있다. Series() 함수의 index 옵션에 인덱스 이름을 리스트 형태로 직접 전달하는 방식이다.

[51]:
```
# 투플을 시리즈로 변환(인덱스 옵션 지정)
tup_data = ('지성', '2021-09-16', '남', True)

sr = pd.Series(tup_data, index = ['이름', '날짜', '성별', '학생여부'])

print(sr)
```

```
이름              지성
날짜      2021-09-16
성별               남
학생여부          True
dtype: object
```

② 원소 1개 선택

[52]:
```
print(sr[0])
```

```
지성
```

[53]:
```
print(sr['이름'])
```

```
지성
```

③ 여러 개의 원소를 선택(인덱스 리스트 활용) 정수형 위치 인덱스는 0부터 시작하기 때문에 2번째 인덱스 이름인 '생년월일'은 정수형 인덱스 1을 사용하고, 3번째 인덱스 이름인 '성별'은 정수형 인덱스 2를 사용한다.

[54]:
```
# 인덱스 리스트 활용
print(sr[[1,2]])
print(sr[['날짜', '성별']])
```

```
날짜    2021-09-16
성별            남
dtype: object
날짜    2021-09-16
성별            남
dtype: object
```

④ 여러 개의 원소를 선택(인덱스 범위 지정) sr[1:2]에서 정수형 위치 인덱스를 사용할 때는, 범위의 끝이 포함되지 않는다.('성별' 불포함) 그러나, sr['생년월일':'성별']과 같이, 인덱스 이름을 사용하면 범위의 끝('성별')이 포함된다.

[55]:
```
# 기본 슬라이싱은 뒤의 인덱스 값은 포함 X
print(sr[1:2])
```

```
날짜    2021-09-16
dtype: object
```

[56]:
```
# 인덱스 이름으로 쓰면 슬라이싱 할 때 뒤에 쓴 이름까지 포함
print(sr['날짜' : '성별'])
```

```
날짜    2021-09-16
성별            남
dtype: object
```

2 DataFrame

2.0.1 1-4 DataFrame 생성하기

원소 3개씩 담고 있는 리스트를 5개 만든다. 이들 5개의 리스트를 원소로 갖는 딕셔너리를 정의하고, 판다스 DataFrame() 함수에 전달하면 5개의 열을 갖는 데이터프레임을 만든다. 이때, 딕셔너리의 키(k)가 열 이름(c0 ~ c4)이 되고, 값(v)에 해당하는 각 리스트가 데이터프레임의 열이 된다. 행 인덱스에는 정수형 위치 인덱스(0, 1, 2)가 자동 지정된다.

[57]:
```
# 열이름을 key로 하고, 리스트를 value로 갖는 딕셔너리 정의(2차원 배열)
dict_data = {'c0':[1,2,3], 'c1':[4,5,6], 'c2':[7,8,9], 'c3':[10,11,12], 'c4':
 ↪[13,14,15]}
```

[58]:
```
df = pd.DataFrame(dict_data)

print(type(df))
print(df)
```

```
<class 'pandas.core.frame.DataFrame'>
   c0  c1  c2  c3  c4
0   1   4   7  10  13
1   2   5   8  11  14
2   3   6   9  12  15
```

2.0.2 1-5 행 인덱스/열 이름 지정하여 데이터 프레임 만들기

① 데이터프레임을 만들 때 설정 '3개의 원소를 갖는 리스트' 2개를 원소로 갖는 리스트(2차원 배열)로 데이터프레임을 만든다. 이때, 각 리스트가 행으로 변환되는 점에 유의한다. • index 옵션: ['준서', '예은'] 배열을 지정 • columns 옵션: ['나이', '성별', '학교'] 배열을 지정

[60]:
```
df = pd.DataFrame([[21, '남', '단국대'], [22, '여', '단국대']],
                  index=['지성', '다은'],
                  columns=['나이','성별','학교'])

print(df)
```

```
    나이 성별   학교
지성  21  남  단국대
다은  22  여  단국대
```

[61]:
```
print(df.index)
print(df.columns)
```

```
Index(['지성', '다은'], dtype='object')
Index(['나이', '성별', '학교'], dtype='object')
```

② 속성을 지정하여 변경하기 데이터프레임 df의 행 인덱스 배열을 나타내는 df.index와 열 이름 배열을 나타내는 df.columns에 새로운 배열을 할당하는 방식으로, 행 인덱스와 열 이름을 변경할 수 있다.

[63]:
```
df.index = ['std1', 'std2']

print(df)
```

```
      나이 성별   학교
std1  21  남  단국대
std2  22  여  단국대
```

[64]:
```
df.columns = ['연령', '남녀', '소속']

print(df)
```

```
      연령 남녀   소속
std1  21  남  단국대
std2  22  여  단국대
```

2.0.3 1-6

④ rename 메소드 사용 rename() 메소드를 적용하면, 행 인덱스 또는 열 이름의 일부를 선택하여 변경 가능. 단, 원본 객체를 직접 수정하는 것이 아니라 새로운 데이터프레임 객체를 반환하는 점 유의~! 원본 객체를 변경하려면, inplace=True 옵션을 사용.

[67]:
```
df.rename(columns={'연령':'나이', '남녀':'성별', '소속':'학교'}, inplace=True)

print(df)
```

```
      나이 성별   학교
std1  21  남   단국대
std2  22  여   단국대
```

2.1 행/열 삭제

2.1.1 1-7 행 삭제

[68]:
```
# DataFrame() 함수로 데이터프레임 변환, 변수 df에 저장
exam_data = {'수학' : [ 90, 80, 70 ], '영어' : [ 98, 89, 95 ],
             '음악' : [ 85, 95, 100 ], '체육' : [ 100, 90, 90 ]}
```

[70]:
```
df = pd.DataFrame(exam_data, index=['지성', '다은', '뽀송'])
print(df)
```

```
    수학  영어   음악   체육
지성  90  98   85  100
다은  80  89   95   90
뽀송  70  95  100   90
```

[71]:
```
df2 = df[:]
```

[72]:
```
print(df2)
```

```
    수학  영어   음악   체육
지성  90  98   85  100
다은  80  89   95   90
뽀송  70  95  100   90
```

[73]:
```
df2.drop('뽀송', inplace=True)
```

```
C:\Python39\lib\site-packages\pandas\core\frame.py:4906: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  return super().drop(
```

[74]:
```
df2.drop('지성')
```

[74]:
```
     수학  영어  음악  체육
다은  80  89  95  90
```

[75]:
```
print(df2)
```

```
    수학  영어  음악   체육
지성  90  98  85  100
다은  80  89  95   90
```

2.1.2 1-8 열 삭제

[76]:
```
df3 = df[:]
df4 = df[:]
```

[77]:
```
df3.drop('수학', axis=1, inplace=True)
```

```
C:\Python39\lib\site-packages\pandas\core\frame.py:4906: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  return super().drop(
```

[78]:
```
print(df3)
```

```
    영어   음악   체육
지성  98   85  100
다은  89   95   90
뽀송  95  100   90
```

[80]:
```
df4.drop(['영어','음악'], axis=1, inplace=True)
```

```
C:\Python39\lib\site-packages\pandas\core\frame.py:4906: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  return super().drop(
```

[81]:
```
print(df4)
```

```
    수학   체육
지성  90  100
다은  80   90
뽀송  70   90
```

2.1.3 1-9 행 선택

① 1개의 행 선택 데이터프레임의 첫 번째 행에는 '지성' 학생의 과목별 점수 데이터가 입력되어 있다. '지성' 학생의 과목별 점수 데이터를 행으로 추출하면 시리즈 객체가 반환된다. `loc`

인덱서를 이용하려면 '지성'이라는 인덱스 이름을 직접 입력하고, iloc을 이용할 때는 첫 번째 정수형 위치를 나타내는 0을 입력한다. 각각 반환되는 값을 label1 변수와 position1 변수에 저장, 출력하면 같은 결과를 갖는다.

[82]:
```
label1 = df.loc['지성']
position1 = df.iloc[0]
print(label1)
```

```
수학     90
영어     98
음악     85
체육    100
Name: 지성, dtype: int64
```

[84]:
```
print(position1)
```

```
수학     90
영어     98
음악     85
체육    100
Name: 지성, dtype: int64
```

② 여러 개의 행을 선택(인덱스 리스트 활용) 2개 이상의 행 인덱스를 배열로 입력하면, 매칭되는 모든 행 데이터를 동시에 추출한다. 데이터프레임 df의 첫번째와 두번째 행에 있는 '지성', '다은' 학생을 인덱싱으로 선택해 본다. loc 인덱서는 ['지성', '다은'] 과 같이 인덱스 이름을 배열로 전달하고, iloc을 이용할 때는 [0, 1] 과 같이 정수형 위치를 전달한다. 이때, label2 변수와 position2 변수에 저장된 값은 같다.

[85]:
```
# 행 인덱스를 사용하여 2개 이상의 행 선택

label2 = df.loc[['지성', '다은']]
position2 = df.iloc[[0,1]]

print(label2)
```

```
    수학  영어  음악   체육
지성  90  98  85  100
다은  80  89  95   90
```

[86]:
```
print(position2)
```

```
    수학  영어  음악   체육
지성  90  98  85  100
다은  80  89  95   90
```

③ 여러 개의 원소를 선택(인덱스 범위 지정) 행 인덱스의 범위를 지정하여 여러 개의 행을 동시에 선택하는 슬라이싱 기법을 사용한다. 단, 인덱스 이름을 범위로 지정한 label3의 경우에는 범위의 마지막 값인 '뽀송' 학생의 점수가 포함 되지만, 정수형 위치 인덱스를 사용한 position3에는 범위의 마지막 값인 '뽀송' 학생의 점수가 제외된다.

[87]:
```
# 행 인덱스의 범위를 지정하여 행 선택

label3 = df.loc['지성' : '뽀송']
position3 = df.iloc[0:2]

print(label3)
```

```
    수학  영어   음악   체육
지성  90  98   85  100
다은  80  89   95   90
뽀송  70  95  100   90
```

[88]:
```
print(position3)
```

```
    수학  영어  음악   체육
지성  90  98  85  100
다은  80  89  95   90
```

2.1.4 1-10 열 선택

① 1개의 열 선택 type( ) 함수를 사용하여, 데이터프레임에서 1개의 열을 선택할 때 반환되는 객체의 자료형을 확인하면 시리즈이다.

[89]:
```
# DataFrame() 함수로 데이터프레임 변환, 변수 df에 저장
exam_data = {'이름' : ['지성', '다은', '뽀송'],
             '수학' : [ 90, 80, 70 ],
             '영어' : [ 98, 89, 95 ],
             '음악' : [ 85, 95, 100 ],
             '체육' : [ 100, 90, 90 ]}
df = pd.DataFrame(exam_data)

print(df)
print(type(df))
```

```
   이름  수학  영어   음악   체육
0  지성  90  98   85  100
1  다은  80  89   95   90
2  뽀송  70  95  100   90
<class 'pandas.core.frame.DataFrame'>
```

[90]:
```
# '수학' 점수 데이터만 선택, 변수 math1에 저장
math1 = df['수학']
print(math1)
print(type(math1))
```

```
0    90
1    80
2    70
Name: 수학, dtype: int64
<class 'pandas.core.series.Series'>
```

[91]:
```
# '영어' 점수 데이터만 선택, 변수 english에 저장
eng = df.영어
print(eng)
print(type(eng))
```

```
0    98
1    89
2    95
Name: 영어, dtype: int64
<class 'pandas.core.series.Series'>
```

② n개의 열 선택(리스트 입력) 대괄호 안에 열 이름의 리스트를 입력하면 리스트의 원소인 열을 모두 선택하여 데이터프레임으로 반환한다. 리스트의 원소로 열 이름 한 개만 있는 경우에도, 2중 대괄호([[ ]])를 사용하는 것이 되어 반환되는 객체가 시리즈가 아니라 데이터프레임이 된다.

[93]:
```
# '음악', '체육' 점수 데이터를 선택. 변수 music_gym에 저장
music_gym = df[['음악', '체육']]
print(music_gym)
print(type(music_gym))
```

```
    음악   체육
0   85  100
1   95   90
2  100   90
<class 'pandas.core.frame.DataFrame'>
```

[94]:
```
# '수학' 점수 데이터만 선택. 변수 math2에 저장
math2 = df[['수학']]
print(math2)
print(type(math2))
```

```
   수학
0  90
1  80
2  70
<class 'pandas.core.frame.DataFrame'>
```

2.1.5 1-11 원소 선택

① 1개의 원소를 선택

[102]:
```
# DataFrame() 함수로 데이터프레임 변환, 변수 df에 저장
exam_data = {'이름' : ['지성', '다은', '뽀송'],
             '수학' : [ 90, 80, 70 ],
             '영어' : [ 98, 89, 95 ],
             '음악' : [ 85, 95, 100 ],
             '체육' : [ 100, 90, 90 ]}
df = pd.DataFrame(exam_data)
```

```
print(df)
```

```
   이름  수학  영어   음악   체육
0  지성  90  98   85  100
1  다은  80  89   95   90
2  뽀송  70  95  100   90
```

[103]:
```
df.set_index('이름')
```

[103]:
```
     수학  영어   음악   체육
이름
지성  90  98   85  100
다은  80  89   95   90
뽀송  70  95  100   90
```

[104]:
```
df.set_index('이름', inplace=True)
```

[105]:
```
print(df)
```

```
    수학  영어   음악   체육
이름
지성  90  98   85  100
다은  80  89   95   90
뽀송  70  95  100   90
```

[106]:
```
# 데이터프레임 df의 특정 원소 1개 선택('지성'의 '음악' 점수)
a = df.loc['지성', '음악']
print(a)
b = df.iloc[0, 2]
print(b)
```

```
85
85
```

② 2개 이상의 원소를 선택(시리즈)

[107]:
```
# 데이터프레임 df의 특정 원소 2개 이상 선택('지성'의 '음악', '체육' 점수)
c = df.loc['지성', ['음악', '체육']]
print(c)
d = df.iloc[0, [2, 3]]
print(d)
e = df.loc['지성', '음악':'체육']
print(e)
f = df.iloc[0, 2:]
print(f)
print(type(f))
```

```
음악    85
체육    100
```

```
Name: 지성, dtype: int64
음악     85
체육    100
Name: 지성, dtype: int64
음악     85
체육    100
Name: 지성, dtype: int64
음악     85
체육    100
Name: 지성, dtype: int64
<class 'pandas.core.series.Series'>
```

③ 2개 이상의 원소를 선택(데이터프레임)

[112]:
```
# df의 2개 이상의 행과 열로부터 원소 선택 ('지성', '다은'의 '음악', '체육' 점수)
g = df.loc[['지성', '다은'], ['음악', '체육']]
print(g)
h = df.iloc[[0, 1], [2, 3]]
print(h)
i = df.loc['지성':'다은', '음악':'체육']
print(i)
j = df.iloc[0:2, 2:]
print(j)
print(type(j))
```

```
    음악   체육
이름
지성  85  100
다은  95   90
    음악   체육
이름
지성  85  100
다은  95   90
    음악   체육
이름
지성  85  100
다은  95   90
    음악   체육
이름
지성  85  100
다은  95   90
<class 'pandas.core.frame.DataFrame'>
```

2.1.6 1-12 열 추가

다음 예제에서'국어'열을 새로 추가하는데, 모든 학생들의 국어 점수가 동일하게 80점으로 입력되는 과정을 보여준다.

[113]:
```
# 데이터프레임 df에 '국어' 점수 열(column) 추가. 데이터 값은 80 지정
df['국어'] = 80
print(df)
```

```
    수학  영어   음악   체육  국어
이름
지성  90  98   85  100  80
다은  80  89   95   90  80
뽀송  70  95  100   90  80
```

2.1.7 1-13 행 추가

[114]:
```
# DataFrame() 함수로 데이터프레임 변환, 변수 df에 저장
exam_data = {'이름' : ['지성', '다은', '뽀송'],
             '수학' : [ 90, 80, 70 ],
             '영어' : [ 98, 89, 95 ],
             '음악' : [ 85, 95, 100 ],
             '체육' : [ 100, 90, 90 ]}
df = pd.DataFrame(exam_data)

print(df)
```

```
   이름  수학  영어   음악   체육
0  지성  90  98   85  100
1  다은  80  89   95   90
2  뽀송  70  95  100   90
```

[115]:
```
# 새로운 행(row) 추가 - 같은 원소 값 입력
df.loc[3] = 0
print(df)
```

```
   이름  수학  영어   음악   체육
0  지성  90  98   85  100
1  다은  80  89   95   90
2  뽀송  70  95  100   90
3   0   0   0    0    0
```

[116]:
```
# 새로운 행(row) 추가 - 원소 값 여러 개의 배열 입력
df.loc[4] = ['사랑별', 90, 80, 70, 60]
print(df)
```

```
    이름  수학  영어   음악   체육
0   지성  90  98   85  100
1   다은  80  89   95   90
2   뽀송  70  95  100   90
3    0   0   0    0    0
4  사랑별  90  80   70   60
```

[117]:
```
# 새로은 행(row) 추가 - 기존 행 복사
df.loc['행5'] = df.loc[3]
print(df)
```

```
     이름  수학  영어   음악   체육
0    지성  90  98   85  100
1    다은  80  89   95   90
2    뽀송  70  95  100   90
3     0   0   0    0    0
4   사랑별  90  80   70   60
행5    0   0   0    0    0
```

2.1.8 1-14 원소 값 변경

① 1개의 원소를 변경 '지성' 학생의 '체육' 점수를 선택하는 여러 방법을 시도한다. 각 방법을 비교하기 위해, 각기 다른 점수를 새로운 값으로 입력하여 원소를 변경하였다. (변경된 값을 원으로 표시)

[118]:
```
# DataFrame() 함수로 데이터프레임 변환, 변수 df에 저장
exam_data = {'이름' : ['지성', '다은', '뽀송'],
             '수학' : [ 90, 80, 70 ],
             '영어' : [ 98, 89, 95 ],
             '음악' : [ 85, 95, 100 ],
             '체육' : [ 100, 90, 90 ]}
df = pd.DataFrame(exam_data)

print(df)
```

```
   이름  수학  영어   음악   체육
0  지성  90  98   85  100
1  다은  80  89   95   90
2  뽀송  70  95  100   90
```

[119]:
```
# '이름' 열을 새로운 인덱스로 지정하고, df 객체에 변경사항 반영
df.set_index('이름', inplace=True)
print(df)
```

```
    수학  영어   음악   체육
이름
지성  90  98   85  100
다은  80  89   95   90
뽀송  70  95  100   90
```

[120]:
```
# 데이터프레임 df의 특정 원소를 변경하는 방법: '지성'의 '체육' 정수
df.iloc[0][3] = 80
print(df)
```

```
    수학  영어   음악  체육
이름
지성  90  98   85  80
```

```
다은  80  89   95  90
뽀송  70  95  100  90
```

[121]:
```
df.loc['지성']['체육'] = 90
print(df)
```

```
    수학  영어   음악  체육
이름
지성  90  98   85  90
다은  80  89   95  90
뽀송  70  95  100  90
```

[122]:
```
df.loc['지성', '체육'] = 100
print(df)
```

```
    수학  영어   음악   체육
이름
지성  90  98   85  100
다은  80  89   95   90
뽀송  70  95  100   90
```

② 1개 이상의 원소를 변경 여러 개의 원소를 선택하여 새로운 값을 할당하면, 모든 원소를 한꺼번에 같은 값으로 변경 가능하다. 선택한 원소의 개수에 맞춰 각기 다른 값을 배열 형태로 입력할 수도 있다.

[123]:
```
# 데이터프레임 df의 원소 여러 개를 변경하는 방법: '지성'의 '음악', '체육' 점수
df.loc['지성', ['음악', '체육']] = 50
print(df)
```

```
    수학  영어   음악  체육
이름
지성  90  98   50  50
다은  80  89   95  90
뽀송  70  95  100  90
```

[124]:
```
df.loc['지성', ['음악', '체육']] = 100, 50
print(df)
```

```
    수학  영어   음악  체육
이름
지성  90  98  100  50
다은  80  89   95  90
뽀송  70  95  100  90
```

2.1.9 1-15 행, 열의 위치 바꾸기

[125]:
```
# 데이터프레임 df을 전치하기(메소드 활용)
df = df.transpose()
print(df)
```

```
이름   지성  다은   뽀송
수학   90  80   70
영어   98  89   95
음악  100  95  100
체육   50  90   90
```

[126]:
```
# 데이터프레임 df를 다시 전치하기(클래스 속성 활용)
df = df.T
print(df)
```

```
    수학  영어   음악  체육
이름
지성  90  98  100  50
다은  80  89   95  90
뽀송  70  95  100  90
```