

# DataFrame 클래스 속성 및 메소드 조사

## o. DataFrame 객체 생성

```
# Create DataFrame object
dataframe = DataFrame(data = [[1, 2, 3, 4], [5, 6, 7, 8]], index=['row1', 'row2'], columns=['column1', 'column2', 'column3', 'column4'])
print(dataframe)
```

	column1	column2	column3	column4
row1	1	2	3	4
row2	5	6	7	8

## 1. at[row label, column label]

### a. 설명

DataFrame 클래스의 속성 중 하나로 행과 열의 라벨을 이용하여 값에 접근한다. 만약 라벨이 존재하지 않는 경우 KeyError가 발생한다.

### b. 예제 코드 및 실행 결과

```
# 1. at - access a singlue value for a row/column label pair.

print(dataframe.at['row1', 'column1']) # 1
print(dataframe.at['row1', 'column5']) # Raise KeyError (label does not exist)

1
-----
KeyError                                                 Traceback (most recent call last)
/usr/local/lib/python3.7/dist-packages/pandas/core/indexes/base.py in get_loc(self, key, method, tolerance)
    3360         try:
    3361             return self._engine.get_loc(casted_key)
-> 3362         except KeyError as err:
    3363             raise KeyError(key) from err
    3364
    3365         if is_scalar(key) and isna(key) and not self.hasnans:
```

KeyError: 'column5'

The above exception was the direct cause of the following exception:

```
KeyError                                                 Traceback (most recent call last)
/usr/local/lib/python3.7/dist-packages/pandas/core/indexes/base.py in get_loc(self, key, method, tolerance)
    3361         return self._engine.get_loc(casted_key)
    3362     except KeyError as err:
-> 3363         raise KeyError(key) from err
    3364
    3365     if is_scalar(key) and isna(key) and not self.hasnans:
```

KeyError: 'column5'

## 2. axes

### a. 설명

DataFrame 클래스의 속성 중 하나로 행과 열의 라벨 정보를 반환한다.

### b. 예제 코드 및 실행 결과

```
[6] # 2. axes - return a list representing the axes of the DataFrame.  
print(dataframe.axes) # information of row axis labels and column axis labels.  
  
[Index(['row1', 'row2'], dtype='object'), Index(['column1', 'column2', 'column3', 'column4'], dtype='object')]
```

## 3. index

### a. 설명

DataFrame 클래스의 속성 중 하나로 행의 라벨 정보를 반환한다.

### b. 예제 코드 및 실행 결과

```
# 3. index - the index(row labels) of the DataFrame.  
print(dataframe.index) # information of row axis labels.  
  
Index(['row1', 'row2'], dtype='object')
```

## 4. columns

### a. 설명

DataFrame 클래스의 속성 중 하나로 열의 라벨 정보를 반환한다.

### b. 예제 코드 및 실행 결과

```
# 4. columns - the column labels of the DataFrame.  
print(dataframe.columns) # information of column axis labels.  
  
Index(['column1', 'column2', 'column3', 'column4'], dtype='object')
```

## 5. dtypes

### a. 설명

DataFrame 클래스의 속성 중 하나로 DataFrame의 데이터 타입 정보를 반환한다.

### b. 예제 코드 및 실행 결과

```
# 5. dtypes - return the dtypes in the DataFrame.  
print(dataframe.dtypes) # dtypes of each columns.
```

```
column1      int64  
column2      int64  
column3      int64  
column4      int64  
dtype: object
```

## 6. empty

### a. 설명

DataFrame 클래스의 속성 중 하나로 DataFrame이 빈 상태인지를 반환한다.

### b. 예제 코드 및 실행 결과

```
# 6. empty - indicator whether DataFrame is empty.  
print(dataframe.empty)
```

```
False
```

## 7. flags

### a. 설명

DataFrame 클래스의 속성 중 하나로 해당 판다스 객체와 관련있는 속성을 가져온다.

## b. 예제 코드 및 실행 결과

```
# 7. flags - get the properties associated with this pandas object.  
print(dataframe.flags)  
dataframe.flags.allow_duplicates = False  
print(dataframe.flags)  
  
<Flags(allow_duplicates=True)>  
<Flags(allow_duplicates=False)>
```

## 8. iat[row position(integer), column position(integer)]

### a. 설명

DataFrame 클래스의 속성 중 하나로 행과 열의 위치(정수)를 이용하여 값에 접근한다. 만약 위치가 존재하지 않는 경우 IndexError가 발생한다.

### b. 예제 코드 및 실행 결과

```
# 8. iat - access a single value for a row/column pair by integer position.  
print(dataframe.iat[0, 0])  
print(dataframe.iat[0, 5]) # Raise IndexError(index is out of bounds.)  
  
1  
-----  
IndexError Traceback (most recent call last)  
<ipython-input-48-eaa154409743> in <module>()  
  1 # 7. iat - access a single value for a row/column pair by integer position.  
  2 print(dataframe.iat[0, 0])  
----> 3 print(dataframe.iat[0, 5])  
  
-----  
/usr/local/lib/python3.7/dist-packages/pandas/core/indexes/base.py in __getitem__(self, key)  
  4602         if is_scalar(key):  
  4603             key = com.cast_scalar_indexer(key, warn_float=True)  
-> 4604             return getitem(key)  
  4605         if isinstance(key, slice):  
  
IndexError: index 5 is out of bounds for axis 0 with size 4
```

## 9. iloc

### a. 설명

DataFrame 클래스의 속성 중 하나로 위치(정수)를 선택하여 해당 위치에 존재하는 모든 값에 접근한다.

## b. 예제 코드 및 실행 결과

```
# 9. iloc - purely integer-location based indexing for selection by position.
print(dataframe.iloc[0]) # select 0th row
print(dataframe.iloc[1]) # select 1th row
# print(dataframe.iloc[2]) # out of bounds
print(dataframe.iloc[:, 0]) # select 0th column
print(dataframe.iloc[:, 1]) # select 1th column
# print(dataframe.iloc[:, 4]) # out of bounds
print(dataframe.iloc[:]) # select all
print(dataframe.iloc[[0]]) # select 0th row with rabel
print(dataframe.iloc[[0, 1]]) # select 0th row, 1th row with rabel
print(dataframe.iloc[[True, False]]) # select rows with boolean (True: select, False: not select),
print(dataframe.iloc[0, 0]) # select 0th row & 0th column's element
```

```
column1    1
column2    2
column3    3
column4    4
Name: row1, dtype: int64
column1    5
column2    6
column3    7
column4    8
Name: row2, dtype: int64
row1      1
row2      5
Name: column1, dtype: int64
row1      2
row2      6
Name: column2, dtype: int64
  column1  column2  column3  column4
row1        1        2        3        4
row2        5        6        7        8
  column1  column2  column3  column4
row1        1        2        3        4
  column1  column2  column3  column4
row1        1        2        3        4
row2        5        6        7        8
  column1  column2  column3  column4
row1        1        2        3        4
1
```

## 10. loc

### a. 설명

DataFrame 클래스의 속성 중 하나로 위치(라벨 또는 불리언 배열)를 선택하여 해당 위치에 존재하는 모든 값에 접근한다.

## b. 예제 코드 및 실행 결과

```
# 10. loc - access a group of rows and columns by label(s) or boolean array.  
print(dataframe.loc['row1']) # select 'row1' row  
print(dataframe.loc['row2']) # select 'row2' row  
# print(dataframe.loc['row3']) # keyerror  
print(dataframe.loc[:, 'column1']) # select 'column1' column  
print(dataframe.loc[:, 'column2']) # select 'column2' column  
# print(dataframe.loc[:, 'column5']) # keyerror  
print(dataframe.loc[:]) # select all  
print(dataframe.loc[['row1']]) # select 'row1' row with label  
print(dataframe.loc[['row1', 'row2']]) # select 'row1' row, 'row2' row with label  
print(dataframe.loc[[True, False]]) # select rows with boolean (True: select, False: not select),  
print(dataframe.loc['row1', 'column1']) # select 0th row & 0th column's element  
  
column1      1  
column2      2  
column3      3  
column4      4  
Name: row1, dtype: int64  
column1      5  
column2      6  
column3      7  
column4      8  
Name: row2, dtype: int64  
row1        1  
row2        5  
Name: column1, dtype: int64  
row1        2  
row2        6  
Name: column2, dtype: int64  
    column1  column2  column3  column4  
row1        1        2        3        4  
row2        5        6        7        8  
    column1  column2  column3  column4  
row1        1        2        3        4  
    column1  column2  column3  column4  
row1        1        2        3        4  
row2        5        6        7        8  
    column1  column2  column3  column4  
row1        1        2        3        4  
1
```

## 11. ndim

### a. 설명

DataFrame 클래스의 속성 중 하나로 해당 객체의 차원을 반환한다.

### b. 예제 코드 및 실행 결과

```
# 11. ndim - return an int representing the number of axes / array dimensions.  
print(dataframe.ndim)
```

## 12. shape

### a. 설명

DataFrame 클래스의 속성 중 하나로 해당 객체의 행과 열의 개수를 튜플로 반환한다.

### b. 예제 코드 및 실행 결과

```
# 12. shape - return a tuple representing the edimensionality of the DataFrame.  
print(dataframe.shape)  
  
(2, 4)
```

## 13. size

### a. 설명

DataFrame 클래스의 속성 중 하나로 해당 객체 내 요소의 개수를 반환한다.

### b. 예제 코드 및 실행 결과

```
# 13. size - return an int representing the number of elements in this object.  
print(dataframe.size)
```

8

## 14. style

### a. 설명

DataFrame 클래스의 속성 중 하나로 해당 객체에 대한 styler 객체를 반환한다.

### b. 예제 코드 및 실행 결과

```
# 14. style - returns a styler object.  
print(dataframe.style)  
  
<pandas.io.formats.style.Styler object at 0x7f5b4c24f790>
```

## 15. values

### a. 설명

DataFrame 클래스의 속성 중 하나로 해당 객체를 numpy 객체로 반환한다.

### b. 예제 코드 및 실행 결과

```
# 15. values - return a numpy representation of the DataFrame.  
print(dataframe.values)  
  
[[1 2 3 4]  
 [5 6 7 8]]
```

## 16. apply(func, axis=0, raw=False, result\_type=None, args=(), \*\*kwargs)

### a. 설명

DataFrame 클래스의 메소드 중 하나로 주어진 axis에 대하여 func를 적용한 결과를 반환한다. 이때 axis가 0 또는 ‘index’인 경우에는 각각의 열에 대해 적용하고, 1 또는 ‘columns’인 경우에는 각각의 행에 대해 적용한다.

### b. 예제 코드 및 실행 결과

```
# 16. apply(func, axis=0, raw=False, result_type=None, args=(), **kwargs) - apply a function along an axis of the DataFrame  
#   func - function to apply to each column or row.  
#   axis - axis along which the function is applied.  
#         - 0 or 'index': apply function to each column.  
#         - 1 or 'columns': apply function to each row.  
#   return - result of applying func along the given axis of the DataFrame  
print(dataframe.apply(np.sqrt))  
print(dataframe.apply(np.sum, axis='index'))  
print(dataframe.apply(np.sum, axis='columns'))  
  
          column1    column2    column3    column4  
row1  1.000000  1.414214  1.732051  2.000000  
row2  2.236068  2.449490  2.645751  2.828427  
column1      6  
column2      8  
column3     10  
column4     12  
dtype: int64  
row1      10  
row2      26  
dtype: int64
```

## 17. combine(other, func, fill\_value=None, overwrite=True)

### a. 설명

DataFrame 클래스의 메소드 중 하나로 다른 DataFrame과 column-wise 결합한 결과를 반환한다. 인자에 func에 대한 함수를 넣어 두 DataFrame을 결합하는 기준을 제공할 수 있고, fill\_value에 대한 값을 넣어 NaN인 요소를 대체할 값을 지정할 수 있고, overwrite에 대한 불리언 값을 넣어 요소가 존재하지 않는 경우 NaN을 넣을지를 지정할 수 있다.

### b. 예제 코드 및 실행 결과

```
# 17. combine(other, funct, fill_value=None, overwrite=True) - perform column-wise combine with another Dataframe.
#   other - the DataFrame to merge column-wise.
#   func - function that takes series as inputs and return a series or scalar. used to merge the two dataframes.
#   fill_value - the value to fill NaNs with prior to passing any column to the merge funct.
#   overwrite - if True, columns in self that do not exist in other will be overwritten with NaNs.
#   return - combination of the provided DataFrames.

df1 = DataFrame({'A': [0, 0], 'B': [None, 0]})
print(df1)
df2 = DataFrame({'B': [1, 1], 'C': [True, False]})
print(df2)

smaller = lambda s1, s2: s1 if (s1.sum() < s2.sum()) else s2

print(df1.combine(df2, smaller, fill_value=-1, overwrite=True))
print(df1.combine(df2, smaller, fill_value=-1, overwrite=False))

      A      B
0  0    NaN
1  0    0.0
      B      C
0  1    True
1  1   False
      A      B      C
0 -1.0 -1.0    True
1 -1.0  0.0   False
      A      B      C
0  0    -1.0   True
1  0    0.0   False
```

## 18. copy(deep=True)

### a. 설명

DataFrame 클래스의 메소드 중 하나로 주어진 DataFrame 객체를 복사한 DataFrame을 반환한다. 인자에 deep에 대한 불리언 값을 넣어 깊은 복사와 얕은 복사를 결정한다. deep이 True인 경우 깊은 복사를, False인 경우 얕은 복사를 한다.

## b. 예제 코드 및 실행 결과

```
# 18. copy(deep=True) - make a copy of this object's indices and data.  
#     if True, a new object will be created with a copy of the calling object's data and indices.  
#     Modifications to the data or indices of the copy will not be reflected in the original object  
#     if False, a new object will be created without copying the calling object's data or index.  
#     Any changes to the data of the original will be reflected in the shallow copy.  
#     return - copied object  
  
dataframe = DataFrame(data = [[1, 2, 3, 4], [5, 6, 7, 8]], index=['row1', 'row2'], columns=['column1', 'column2',  
deep = dataframe.copy(deep=True)  
shallow = dataframe.copy(deep=False)  
print(dataframe)  
print(deep)  
print(shallow)  
  
# modify original  
dataframe['column4'] = -1  
print(dataframe)  
print(deep)  
print(shallow)  
  
      column1  column2  column3  column4  
row1        1          2          3          4  
row2        5          6          7          8  
      column1  column2  column3  column4  
row1        1          2          3          4  
row2        5          6          7          8  
      column1  column2  column3  column4  
row1        1          2          3          4  
row2        5          6          7          8  
      column1  column2  column3  column4  
row1        1          2          3          -1  
row2        5          6          7          -1  
      column1  column2  column3  column4  
row1        1          2          3          4  
row2        5          6          7          8  
      column1  column2  column3  column4  
row1        1          2          3          -1  
row2        5          6          7          -1
```

## 19. dot(other)

### a. 설명

DataFrame 클래스의 메소드 중 하나로 두 DataFrame에 대한 행렬 곱 결과를 반환한다.

### b. 예제 코드 및 실행 결과

```
# 19. dot(other) - compute the matrix multiplication between DataFrame and other  
#     return - result  
  
dataframe = DataFrame([[1, 2, 3, 4], [5, 6, 7, 8]]) # n x k matrix  
dataframe2 = DataFrame([[-1, -1], [0, 0], [1, 1], [-1, -1]]) # k x m matrix  
print(dataframe.dot(dataframe2)) # n x m matrix  
  
      0   1  
0 -2 -2  
1 -6 -6
```

## 20. head(n=5)

### a. 설명

DataFrame 클래스의 메소드 중 하나로 주어진 객체의 상위 n개 행을 반환한다. 만약 n이 음수인 경우 상위 n번째 행을 제외한 모든 행을 반환한다.

### b. 예제 코드 및 실행 결과

```
# 20. head(n=5) - return the first n rows.
#     n's default value is 5 and if n is negative it will return all rows except the first n rows.
dataframe = DataFrame(data = [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]], index=['row1', 'row2', 'row3'],
print(dataframe.head(1)) # return first 1 row.
print(dataframe.head(-1)) # return all rows except first -1 row(=last 1 row).

      column1  column2  column3  column4
row1      1        2        3        4
      column1  column2  column3  column4
row1      1        2        3        4
row2      5        6        7        8
```

## 21. tail(n=5)

### a. 설명

DataFrame 클래스의 메소드 중 하나로 주어진 객체의 하위 n개 행을 반환한다. 만약 n이 음수인 경우 하위 n번째 행을 제외한 모든 행을 반환한다.

### b. 예제 코드 및 실행 결과

```
# 21. tail(n=5) - return the last n rows.
dataframe = DataFrame(data = [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]], index=['row1', 'row2', 'row3'],
print(dataframe.tail(1)) # return last 1 row.
print(dataframe.tail(-1)) # return all rows except last -1 row(=first 1 row).

      column1  column2  column3  column4
row3      9        10       11       12
      column1  column2  column3  column4
row2      5        6        7        8
row3      9        10       11       12
```

## 22. max(axis=NoDefault.no\_default,skipna=True, level=None, numeric\_only=None)

### a. 설명

DataFrame 클래스의 메소드 중 하나로 주어진 axis에 대한 최대값을 반환한다. axis에 0 또는 ‘index’를 넣어 각각의 column에 대한 최대값을, 1 또는 ‘columns’를 넣어 각각의 row에 대한 최대값을 찾도록 할 수 있다. 또한 인자에 skipna에 대한 불리언 값을 넣어 NaN 또는 null 값을 제외할지를 결정할 수 있다.

## b. 예제 코드 및 실행 결과

```
# 22. max(axis=NoDefault.no_default, skipna=True, level=None, numeric_only=None) - return the maximum of the values over the requested axis.  
#     axis - axis for the function to be applied on. (index(0), columns(1))  
#     skipna - exclude NA/null values when computing the result.  
#     level - if the axis is a MultiIndex, count along a particular level.  
#     numeric_only - include only float, int, boolean columns. If None, will attempt to use everything.  
#     return - Series or DataFrame(if level specified)  
  
print(dataframe)  
print(dataframe.max()) # axis='index'  
print(dataframe.max(axis='index')) # return the maximum value of each column.  
print(dataframe.max(axis='columns')) # return the maximum value of each index.  
  
      column1  column2  column3  column4  
row1       1       2       3       4  
row2       5       6       7       8  
row3       9      10      11      12  
column1      9  
column2     10  
column3     11  
column4     12  
dtype: int64  
column1      9  
column2     10  
column3     11  
column4     12  
dtype: int64  
row1       4  
row2       8  
row3      12  
dtype: int64
```

## 23. idxmax(axis=0, skipna=True)

### a. 설명

DataFrame 클래스의 메소드 중 하나로 주어진 axis에 대한 최대값의 위치를 반환한다. axis에 0 또는 ‘index’를 넣어 각각의 column에 대한 최대값을, 1 또는 ‘columns’를 넣어 각각의 row에 대한 최대값을 찾도록 할 수 있다. 또한 인자에 skipna에 대한 불리언 값을 넣어 NaN 또는 null 값을 제외할지를 결정할 수 있다.

### b. 예제 코드 및 실행 결과

```
# 23. idxmax(axis=0, skipna=True) - return the row label of the maximum value.  
#     axis - axis for the function to be applied on. (index(0), columns(1))  
#     skipna - exclude NA/null values. If the entire Series is NA, the result will be NA.  
#     return - label of the maximum value.  
  
print(dataframe)  
print(dataframe.idxmax()) # return the maximum value's index label of each column.  
print(dataframe.idxmax(1)) # return the maximum value's column label of each row.  
  
      column1  row3  
column2  row3  
column3  row3  
column4  row3  
dtype: object  
row1    column4  
row2    column4  
row3    column4  
dtype: object
```

## 24. min(axis=NoDefault,no\_default,skipna=True,level=None, numeric\_only=None)

### a. 설명

DataFrame 클래스의 메소드 중 하나로 주어진 axis에 대한 최소값을 반환한다. axis에 0 또는 ‘index’를 넣어 각각의 column에 대한 최소값을, 1 또는 ‘columns’를 넣어 각각의 row에 대한 최소값을 찾도록 할 수 있다. 또한 인자에 skipna에 대한 불리언 값을 넣어 NaN 또는 null 값을 제외할지를 결정할 수 있다.

### b. 예제 코드 및 실행 결과

```
# 24. min(axis=NoDefault.no_default, skipna=True, level=None, numeric_only=None) - return the minimum of the values over the requested axis.
#   axis - axis for the function to be applied on. (index(0), columns(1))
#   skipna - exclude NA/null values when computing the result.
#   level - if the axis is a MultiIndex, count along a particular level.
#   numeric_only - include only float, int, boolean columns. if None, will attempt to use everything.
#   return - Series or DataFrame(if level specified)

print(dataframe)
print(dataframe.min()) # axis='index'
print(dataframe.min(axis='index')) # return the minimum value of each column.
print(dataframe.min(axis='columns')) # return the minimum value of each index.

column1  column2  column3  column4
row1      1          2          3          4
row2      5          6          7          8
row3      9         10         11         12
column1  1
column2  2
column3  3
column4  4
dtype: int64
column1  1
column2  2
column3  3
column4  4
dtype: int64
row1     1
row2     5
row3     9
dtype: int64
```

## 25. idxmin(axis=0, skipna=True)

### a. 설명

DataFrame 클래스의 메소드 중 하나로 주어진 axis에 대한 최소값의 위치를 반환한다. axis에 0 또는 ‘index’를 넣어 각각의 column에 대한 최소값을, 1 또는 ‘columns’를 넣어 각각의 row에 대한 최소값을 찾도록 할 수 있다. 또한 인자에 skipna에 대한 불리언 값을 넣어 NaN 또는 null 값을 제외할지를 결정할 수 있다.

## b. 예제 코드 및 실행 결과

```
# 25. idxmin(axis=0, skipna=True) - return the row label of the minimum value.  
#     axis - axis for the function to be applied on. (index(0), columns(1))  
#     skipna - exclude NA/null values. If the entire Series is NA, the result will be NA.  
#     return - label of the minimum value.  
  
print(dataframe)  
print(dataframe.idxmin()) # return the minimum value's index label of each column.  
print(dataframe.idxmin(1)) # return the maximum value's column label of each row.  
  
      column1  column2  column3  column4  
row1      1        2        3        4  
row2      5        6        7        8  
row3      9       10       11       12  
column1    row1  
column2    row1  
column3    row1  
column4    row1  
dtype: object  
row1    column1  
row2    column1  
row3    column1  
dtype: object
```

## 26.sum(axis=None, skipna=True, level=None, numeric\_only=None, min\_count=0)

### a. 설명

DataFrame 클래스의 메소드 중 하나로 주어진 axis에 대한 합을 반환한다. axis에 0 또는 'index'를 넣어 각각의 column에 대한 합을, 1 또는 'columns'를 넣어 각각의 row에 대한 합을 찾도록 할 수 있다. 또한 인자에 skipna에 대한 불리언 값을 넣어 NaN 또는 null 값을 제외할지를 결정할 수 있다.

## b. 예제 코드 및 실행 결과

```
# 26. sum(axis=None, skipna=True, level=None, numeric_only=None, min_count=0) - return the sum of the values over the requested axis.  
#     axis - axis for the function to be applied on. (index(0), columns(1))  
#     skipna - exclude NA/null values when computing the result.  
#     level - if the axis is a multiindex, count along a particular level.  
#     numeric_only - include only float, int, boolean columns. if None, will attempt to use everything.  
#     min_count - the required number of valid values.  
  
print(dataframe)  
print(dataframe.sum('index')) # return the sum of each column.  
print(dataframe.sum('columns')) # return the sum of each index.  
  
      column1  column2  column3  column4  
row1      1        2        3        4  
row2      5        6        7        8  
row3      9       10       11       12  
column1    15  
column2    18  
column3    21  
column4    24  
dtype: int64  
row1     10  
row2     26  
row3     42  
dtype: int64
```

## 27. transpose()

### a. 설명

DataFrame 클래스의 메소드 중 하나로 주어진 객체에 대한 역치 행렬 객체를 반환한다.

### b. 예제 코드 및 실행 결과

```
# 27. transpose() - transpose index and columns
#      return - the transposed DataFrame
print(dataframe)
print(dataframe.T)
print(dataframe.transpose())
```

	column1	column2	column3	column4
row1	1	2	3	4
row2	5	6	7	8
row3	9	10	11	12
	row1	row2	row3	
column1	1	5	9	
column2	2	6	10	
column3	3	7	11	
column4	4	8	12	
	row1	row2	row3	
column1	1	5	9	
column2	2	6	10	
column3	3	7	11	
column4	4	8	12	

## 28.add\_prefix(prefix)

### a. 설명

DataFrame 클래스의 메소드 중 하나로, 주어진 객체가 DataFrame 객체인 경우 모든 column 라벨의 앞에 prefix를 붙이고 주어진 객체가 Series 객체인 경우 모든 row 라벨의 앞에 prefix를 붙인다.

### b. 예제 코드 및 실행 결과

```
# 28. add_prefix(prefix) - prefix labels with string prefix.  
#     prefix - the string to add before each column's label  
#             the string to add before each row's label if Series.  
#     return - new Series or DataFrame with updated labels.  
dataframe = DataFrame(data=[[0, 1, 2, 3, 4], [5, 6, 7, 8, 9]])  
print(dataframe)  
print(dataframe.add_prefix('column'))
```

```
0   1   2   3   4  
0   0   1   2   3   4  
1   5   6   7   8   9  
    column0  column1  column2  column3  column4  
0       0        1        2        3        4  
1       5        6        7        8        9
```

## 29.add\_suffix(suffix)

### a. 설명

DataFrame 클래스의 메소드 중 하나로, 주어진 객체가 DataFrame 객체인 경우 모든 column 라벨의 뒤에 suffix를 붙이고 주어진 객체가 Series 객체인 경우 모든 row 라벨의 뒤에 suffix를 붙인다.

### b. 예제 코드 및 실행 결과

```
# 29. add_suffix(suffix) - suffix labels with string suffix.  
#     suffix - the string to add after each column's label if DataFrame.  
#             the string to add after each row's label if Series.  
#     return - new Series or DataFrame with updated labels.  
dataframe = DataFrame(data=[[0, 1, 2, 3, 4], [5, 6, 7, 8, 9]])  
print(dataframe)  
print(dataframe.add_suffix(' column'))
```

```
0   1   2   3   4  
0   0   1   2   3   4  
1   5   6   7   8   9  
    0 column  1 column  2 column  3 column  4 column  
0       0        1        2        3        4  
1       5        6        7        8        9
```

## 30. equals(other)

### a. 설명

DataFrame 클래스의 메소드 중 하나로 주어진 두 객체가 동일한 요소를 갖는 객체인지를 반환한다.

### b. 예제 코드 및 실행 결과

```
# 30. equals(other) - test whether two objects contain the same elements.
#       other - the other Series or DataFrame to be compared with the first.
#       return - true if all elements are the same in both objects.
dataframe = DataFrame(data=[[0, 1, 2, 3, 4], [5, 6, 7, 8, 9]])
dataframe2 = DataFrame(data=[[0, 1, 2, 3, 4], [5, 6, 7, 8, 9]])
dataframe3 = DataFrame(data=[[0, 1, 2, 3, 4], [-1, -1, -1, -1, -1]])

print(dataframe.equals(dataframe2))
print(dataframe.equals(dataframe3))
```

True  
False