

데이터처리프로그래밍

데이터분석도구(Pandas)



강원대학교 교육혁신원 송혜정
<hjsong@kangwon.ac.kr>



Pandas

✓ 학습목표

- 빅데이터 분석 도구인 파이썬 라이브러리 pandas를 이해한다.

✓ 학습내용

- Pandas 개요
- DataFrame
- Indexing / Selection / Viewing
- Setting / Addition / Deletion



강의에 앞서서..

- 본 강의자료는 아래의 자료들을 참고하여 만들어 졌음을 알립니다

1. Pandas (<https://pandas.pydata.org/pandas-docs/stable/index.html>)

Pandas

- 데이터 분석을 위한 파이썬 라이브러리
- pandas는 numpy 라이브러리 기반으로 개발
- 사용 가능한 데이터 유형
 - SQL 테이블 또는 Excel 스프레드 시트에서와 같이 서로 다른 데이터 형으로 열을 구성하는 테이블 형식의 데이터
 - 행, 열의 레이블이 포함 행렬 데이터
 - 정렬되거나 정렬되지 않은 시계열 데이터.
 - 서로 다른 형태의 관찰 또는 통계 데이터 집합
- pandas 데이터 구조
 - Series : 같은 자료형으로 구성된 1 차원 배열 구조
 - DataFrame : 서로 다른 데이터 형으로 구성 가능한 2 차원 테이블 구조 (R의 DataFrame이 제공하는 기능과 유사)

Pandas

- 기능
 - missing data (NaN)
 - inserted and deleted from DataFrame
 - group by
 - easy to convert
 - slicing, fancy indexing, and subsetting
 - merging and joining
 - reshaping
 - saving / loading data : flat files (CSV), Excel files, databases
 - Time series data processing

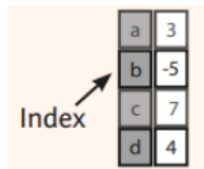
Series

1. 데이터 객체 생성

1) Series

모든 데이터 유형 (정수, 문자열, 부동 소수점 수, 파이썬 객체 등)을 포함할 수 있는 일차원 레이블 배열

s = pd.Series(data, index=index)



a	3
b	-5
c	7
d	4

```
#사용할 라이브러리 import
import numpy as np
import pandas as pd
from IPython.display import display
```

#Series, python list로 생성, 인덱스 자동생성(순번)

```
s1 = pd.Series([10,20,30,40,50])
display("s1=", s1)
print("index=", s1.index)
print("Values=", s1.values)
```

#Series, python list로 생성, 인덱스(레이블) 포함

```
s2 = pd.Series([10,20,30,40,50], index=['a', 'b', 'c', 'd', 'e'])
display("s2=", s2)
print("index=", s2.index)
print("values=", s2.values)
```

's1='

```
0    10
1    20
2    30
3    40
4    50
```

dtype: int64

```
index= RangeIndex(start=0, stop=5, step=1)
Values= [10 20 30 40 50]
```

's2='

```
a    10
b    20
c    30
d    40
e    50
```

dtype: int64

```
index= Index(['a', 'b', 'c', 'd', 'e'], dtype='object')
values= [10 20 30 40 50]
```

Series

#Series, numpy ndarray로 생성

```
s1 = pd.Series(np.random.randn(5))
s2 = pd.Series(np.random.randn(5), index=['a', 'b', 'c', 'd', 'e'])
display("s1=", s1)
display("s2=", s2)
```

#Series, 파이썬 dicts, KEY-VALUE 로 생성

```
d = {'b': 1, 'a': 0, 'c': 2}
s3 = pd.Series(d) #Key는 레이블로 처리
display("s3=", s3)
```

#Series, 하나의 값으로 생성, scalar value

```
s4 = pd.Series(5., index=['a', 'b', 'c', 'd', 'e'])
display("s4=", s4)
```

#Series, numpy의 ndarray와 유사한 속성으로 차원, 형태, 타입 확인

```
print("s1.ndim=", s1.ndim)
print("s1.shape=", s1.shape)
print("s1.dtype=", s1.dtype)
```

's1='

```
0    0.488914
1    0.326131
2   -1.158671
3   -0.805848
4   -1.305248
dtype: float64
```

's2='

```
a   -1.028073
b    0.926129
c    0.144403
d   -1.204910
e   -0.262983
dtype: float64
```

's3='

```
b    1
a    0
c    2
dtype: int64
```

's4='

```
a    5.0
b    5.0
c    5.0
d    5.0
e    5.0
dtype: float64
```

```
s1.ndim= 1
s1.shape= (5,)
s1.dtype= float64
```

Series

```
#Series, numpy의 ndarray와 유사한 인덱싱, 슬라이싱
display("s1=", s1)
display("s1[0]=", s1[0])           #index 0에 해당하는 값
display("s1[:2]=", s1[:2])         #처음부터 index (2-1)
display("s1[1:5:2]=", s1[1:5:2])  #1부터 index (5-1), step 2
display("s1[3:]=", s1[3:])         #3부터 끝까지
display("s1[-2:]=", s1[-2:])      #끝에서 두번째부터 끝까지
```

```
#Series, 파이썬 dicts와 유사한 인덱싱
print("s3['a']=", s3['a'])          #'a' 인덱스의 값 추출
print("s3.get('a')=", s3.get('a'))
print("'c' in s3 = ", 'c' in s3)  #s3 인덱스에 'c' 존재여부 판단
```

```
's1='
0    0.488914
1    0.326131
2   -1.158671
3   -0.805848
4   -1.305248
dtype: float64

's1[0]='
0.488914437652313

's1[:2]='
0    0.488914
1    0.326131
dtype: float64
```

```
's1[1:5:2]='
1    0.326131
3   -0.805848
dtype: float64

's1[3:]='
3   -0.805848
4   -1.305248
dtype: float64

's1[-2:]='
3   -0.805848
4   -1.305248
dtype: float64

s3['a'] = 0
s3.get('a') = 0
'c' in s3 = True
```


DataFrame

2)DataFrame

- 서로 다른 자료형으로 열을 구성할 수 있는 2 차원 데이터 구조
- 행의 레이블 index, 열의 레이블 columns 사용
- 엑셀 스프레드시트, SQL 테이블, Series 객체의 dict 구조

Columns		Country	Capital	Population
Index	0	Belgium	Brussels	11190846
	1	India	New Delhi	1303171035
	2	Brazil	Brasília	207847528

DataFrame

(1) DataFrame 생성

```
#DataFrame 생성
#DataFrame: data + index + columns
#DataFrame, From ndarrays, list
data = np.random.randint(8, size=(5, 3)) #random number
idx = np.arange(5)
col = list('ABC')
df = pd.DataFrame(data, index=idx, columns=col) #data, 행, 열 레이블 설정
display(df) #DataFrame 확인
```

	A	B	C
0	5	5	2
1	0	5	6
2	3	2	0
3	2	2	1
4	0	6	1

DataFrame

(1) DataFrame 생성

```
#DataFrame 속성 확인
print("df.dtypes = ", df.dtypes)    #데이터 타입들
print("df.index = ", df.index)      #행 레이블
print("df.columns = ", df.columns)  #열 레이블
print("df.values=", df.values)      #데이터값들
```

```
df.dtypes = A      int32
B      int32
C      int32
dtype: object
df.index = Int64Index([0, 1, 2, 3, 4], dtype='int64')
df.columns = Index(['A', 'B', 'C'], dtype='object')
df.values= [[5 5 2]
 [0 5 6]
 [3 2 0]
 [2 2 1]
 [0 6 1]]
```

DataFrame

(1) DataFrame 생성

```
#DataFrame 생성, From ndarrays, list
data = np.random.randint(10, size=(5,4)) #random number
col = list('ABCD')
dates = pd.date_range('20200101', periods=5)
df2 = pd.DataFrame(data) #data만으로 생성
df2.index = dates #행 레이블 설정
df2.columns = col #열 레이블 설정
display("df2=", df2)

print("df2.dtypes = ", df2.dtypes) #데이터 타입들
print("df2.index = ", df2.index) #행 레이블
print("df2.columns = ", df2.columns) #열 레이블
display("df2.values=", df2.values) #데이터값들
```

'df2='

	A	B	C	D
2020-01-01	4	4	5	4
2020-01-02	8	0	3	7
2020-01-03	9	6	3	2
2020-01-04	0	3	9	7
2020-01-05	7	6	7	0

```
df2.dtypes = A      int32
B      int32
C      int32
D      int32
dtype: object
df2.index = DatetimeIndex(['2020-01-01', '2020-01-02', '2020-01-03', '2020-01-04',
                           '2020-01-05'],
                           dtype='datetime64[ns]', freq='D')
df2.columns = Index(['A', 'B', 'C', 'D'], dtype='object')

'df2.values='
array([[4, 4, 5, 4],
       [8, 0, 3, 7],
       [9, 6, 3, 2],
       [0, 3, 9, 7],
       [7, 6, 7, 0]])
```

DataFrame

(1) DataFrame 생성

#DataFrame 생성, From dict of ndarrays / lists

```
d = {'one': [1., 2., 3., 4.],
     'two': [4., 3., 2., 1.]}
df3 = pd.DataFrame(d)
df4 = pd.DataFrame(d, index=['a', 'b', 'c', 'd'])
display("df3=", df3)
display("df4=", df4)
```

#DataFrame 생성, From dict of Series or dicts

```
d = {'one': pd.Series([1., 2., 3.], index=['a', 'b', 'c']),
     'two': pd.Series([1., 2., 3., 4.], index=['a', 'b', 'c', 'd'])}
df5 = pd.DataFrame(d)
display("df5=", df5)
```

#서로 다른 자료형으로 구성된 데이터프레임 생성

```
df6 = pd.DataFrame({'A': 1.,
                    'B': pd.Timestamp('20200101'),
                    'C': pd.Series(1, index=[0,1,2,3], dtype='float32'),
                    'D': np.array([3,4,5,6], dtype='int32'),
                    'E': pd.Categorical(["test", "train", "test", "train"]),
                    'F': 'foo'})
display("df6=", df6)
print("df6.dtypes=", df6.dtypes)
```

'df3='

	one	two
0	1.0	4.0
1	2.0	3.0
2	3.0	2.0
3	4.0	1.0

'df4='

	one	two
a	1.0	4.0
b	2.0	3.0
c	3.0	2.0
d	4.0	1.0

'df5='

	one	two
a	1.0	1.0
b	2.0	2.0
c	3.0	3.0
d	NaN	4.0

'df6='

	A	B	C	D	E	F
0	1.0	2020-01-01	1.0	3	test	foo
1	1.0	2020-01-01	1.0	4	train	foo
2	1.0	2020-01-01	1.0	5	test	foo
3	1.0	2020-01-01	1.0	6	train	foo

```
df6.dtypes= A          float64
B      datetime64[ns]
C          float32
D          int32
E          category
F          object
dtype: object
```

DataFrame

(2)DataFrame 확인(Viewing)

```
#DataFrame 생성
data = np.random.randint(10, size=(7, 3))
col = list('XYZ')
dates = pd.date_range('20200101', periods=7)
df = pd.DataFrame(data, index=dates, columns=col)
display(df)
```

```
#DataFrame display
print("df=", df) #전체 출력
print("df.head()=", df.head()) #상단행 5개
print("df.tail(3)=", df.tail(3)) #하단행 3개
```

```
#DataFrame 속성
print("df.dtypes =", df.dtypes) #데이터 타입들
print("df.index =", df.index) #행 레이블
print("df.columns =", df.columns) #열 레이블
print("df.values =", df.values) #데이터
```

	X	Y	Z
2020-01-01	6	7	3
2020-01-02	9	8	6
2020-01-03	3	0	0
2020-01-04	0	0	2
2020-01-05	5	0	1
2020-01-06	2	5	4
2020-01-07	6	7	1

```
df=
      X  Y  Z
2020-01-01  6  7  3
2020-01-02  9  8  6
2020-01-03  3  0  0
2020-01-04  0  0  2
2020-01-05  5  0  1
2020-01-06  2  5  4
2020-01-07  6  7  1
df.head()=
      X  Y  Z
2020-01-01  6  7  3
2020-01-02  9  8  6
2020-01-03  3  0  0
2020-01-04  0  0  2
2020-01-05  5  0  1
df.tail(3)=
      X  Y  Z
2020-01-05  5  0  1
2020-01-06  2  5  4
2020-01-07  6  7  1
```

```
df.dtypes = X      int32
            Y      int32
            Z      int32
dtype: object
df.index = DatetimeIndex(['2020-01-01', '2020-01-02', '2020-01-03', '2020-01-04',
                           '2020-01-05', '2020-01-06', '2020-01-07'],
                           dtype='datetime64[ns]', freq='D')
df.columns = Index(['X', 'Y', 'Z'], dtype='object')
df.values = [[6 7 3]
             [9 8 6]
             [3 0 0]
             [0 0 2]
             [5 0 1]
             [2 5 4]
             [6 7 1]]
```

DataFrame

(2)DataFrame 확인(Viewing)

```
#통계요약
print("df.describe()=", df.describe())

#Sorting by an axis:
dfsi = df.sort_index(ascending=False) #인덱스 역순으로 정렬
display(dfsi)

#Sorting by values:
dfvy = df.sort_values(by='Y') # 'Y'열을 기준으로 정렬
display(dfvy)
```

```
df.describe()=
```

		X	Y	Z
count	7.000000	7.000000	7.000000	
mean	4.428571	3.857143	2.428571	
std	2.992053	3.716117	2.070197	
min	0.000000	0.000000	0.000000	
25%	2.500000	0.000000	1.000000	
50%	5.000000	5.000000	2.000000	
75%	6.000000	7.000000	3.500000	
max	9.000000	8.000000	6.000000	

	X	Y	Z
2020-01-07	6	7	1
2020-01-06	2	5	4
2020-01-05	5	0	1
2020-01-04	0	0	2
2020-01-03	3	0	0
2020-01-02	9	8	6
2020-01-01	6	7	3

	X	Y	Z
2020-01-03	3	0	0
2020-01-04	0	0	2
2020-01-05	5	0	1
2020-01-06	2	5	4
2020-01-01	6	7	3
2020-01-07	6	7	1
2020-01-02	9	8	6

QUIZ!

1. 데이터프레임 생성 연습

1) 5명의 성적 데이터 생성 (성명, 입학년도, 학점(0~4.5), 성별(M,F))

열이름 : Name, Adyear, grade, sex

행인덱스 : 순번

2) 인덱스, 칼럼, 값들, 데이터타입 확인

3) 통계요약 확인

4) 성명순으로 정렬

5) 성적 역순으로 정렬

	Name	Adyear	Grade	Sex
0	kim	2017	3.5	M
1	lee	2017	2.0	M
2	min	2018	4.5	F
3	jin	2019	2.2	F
4	park	2018	4.0	M

- 데이터 프레임 생성에 대한 학습활동 결과는 e-루리에 제출바랍니다.
- 제목 : Report11. pandas 생성 연습문제
- 제출내용 : pandas 생성 연습문제를 해결하여
“Report11_성명_pandas.ipynb” 파일을 제출
- 제출기한 : 2021년 6월 8일 오후 11:30