파이썬 Pandas 패키지

2018. 1. 29

목차

Pandas?

- NumPy 기반에서 개발되었으며, 고수준의 자료 구조와 파이썬을 통한 데이터 분석 도구를 포함
  - Pandas를 설치하기 앞서 NumPy가 설치되어야 함
  - 아나콘다(Anaconda)를 설치하는 경우, 이미 설치 완료

Pandas 주요 기능

- 시계열 데이터와 비시계열 데이터를 함께 다룰 수 있는 통합 자료 구조 제공
- 누락된 데이터를 유연하게 처리
- SQL 같은 일반 데이터베이스 처럼 합치고 관계 연산을 수행

1.Pandas 자료 구조 소개

- Series
  - import pandas as pd
  - obj=pd.Series([4,7,-5,3])
  - obj

```
0    4
1    7
2   -5
3    3
dtype: int64
```

1.Pandas 자료 구조 소개

- Series
  - obj.values

```
array([ 4,  7, -5,  3], dtype=int64)
```

  - obj.index

```
RangeIndex(start=0, stop=4, step=1)
```

1.Pandas 자료 구조 소개

- Series
  - obj2 = pd.Series([4, 7, -5, 3], index=['d', 'b', 'a', 'c'])
  - obj2

```
d    4
b    7
a   -5
c    3
dtype: int64
```

  - obj2.index

```
Index(['d', 'b', 'a', 'c'], dtype='object')
```

1.Pandas 자료 구조 소개

- Series 색인
  - 데이터의 값을 선택하거나 대입할 때는 색인을 이용하여 접근
  - obj2[‘a’]
  - obj2[‘d’]=6
  - obj2[[‘c’, ‘a’, ‘d’]]

1.Pandas 자료 구조 소개

- Series
  - 고정 길이의 정렬된 사전형
  - ‘b’ in obj2

    True
  - ‘e’ in obj2

    False

1.Pandas 자료 구조 소개

- 사전형 데이터로 부터 Series 데이터 생성
  - sdata = {‘Ohio’:35000, ‘Texa’:71000, ‘Oregon’:16000, ‘Utah: 5000}
  - obj3 = pd.Series(sdata)

```
Ohio      35000
Oregon    16000
Texa      71000
Utah       5000
dtype: int64
```

1.Pandas 자료 구조 소개

- 사전형 데이터와 리스트 활용
  - states = {‘California’, ‘Ohio’, ‘Oregon’, ‘Texa’}
  - obj4 = pd.Series(sdata, index=states)

```
California         NaN
Oregon         16000.0
Texa           71000.0
Ohio           35000.0
dtype: float64
```

1. Pandas 자료 구조 소개

- Series 데이터 'NA' 값 확인
  - pd.isnull(obj4)

```
California      True
Oregon         False
Texa           False
Ohio           False
dtype: bool
```

  - pd.notnull(obj4)

```
California     False
Oregon          True
Texa            True
Ohio            True
dtype: bool
```

1. Pandas 자료 구조 소개

- 색인이 다른 Series 데이터의 산술 연산
  - print(obj3 + obj4)

```
California         NaN
Ohio           70000.0
Oregon         32000.0
Texa          142000.0
Utah               NaN
dtype: float64
```

1. Pandas 자료 구조 소개

- DataFrame
  - 스프레드시트 형식의 자료 구조
  - 여러 개의 열은 서로 다른 종류의 값(숫자, 문자, 불리언 등)을 가질 수 있음
  - DataFrame은 데이터를 내부적으로 2차원 형식으로 저장

1. Pandas 자료 구조 소개

- DataFrame
  - data = {'state':['Ohio','Ohio','Ohio','Nevada','Nevada']
  - ,'year':[2000,2001,2002,2001,2002],'pop':[1.5,1.7,3.6,2.4,2.9]}
  - frame=pd.DataFrame(data)
  - print(frame)

	pop	state	year
0	1.5	Ohio	2000
1	1.7	Ohio	2001
2	3.6	Ohio	2002
3	2.4	Nevada	2001
4	2.9	Nevada	2002

1. Pandas 자료 구조 소개

- DataFrame
  - data = {'state':['Ohio','Ohio','Ohio','Nevada','Nevada']
  - ,'year':[2000,2001,2002,2001,2002],'pop':[1.5,1.7,3.6,2.4,2.9]}

  - # 열은 정렬되어 저장되나, 원하는 순서대로 재지정 가능
  - print(pd.DataFrame(data, columns=['year', 'state', 'pop']))

1. Pandas 자료 구조 소개

- DataFrame
  - Serises와 동일하게 없는 값을 추가하면 NA 값이 저장
  - frame2= pd.DataFrame(data, columns=['year', 'state', 'pop', 'debt'],index=['one', 'two', 'three', 'four', 'five'])
  - print(frame2)

1. Pandas 자료 구조 소개

- DataFrame
  - 열은 Series 처럼 사전 형식 또는 속성 형식으로 접근
  - print(frame2['state'])
  - print(frame2.year)

1. Pandas 자료 구조 소개

- DataFrame
  - 행은 ix, loc 와 같은 메서드를 통하여 접근
  - print(frame2.ix[‘three’])
  - print(frame2.loc[‘three’])

1. Pandas 자료 구조 소개

- DataFrame
  - 열에는 특정 값이나 배열의 값을 대입할 수 있음
  - frame2.debt = 16.5
  - frame2[‘debt’]) = np.arange(5)
  - # 리스트나 배열의 값은 행의 수와 일치해야 함. 없는 색인에는 값이 대입되지 않음

1. Pandas 자료 구조 소개

- DataFrame
  - frame2['eastern'] = frame2.state =='Ohio'
  - frame2

	year	state	pop	debt	eastern
one	2000	Ohio	1.5	0	True
two	2001	Ohio	1.7	1	True
three	2002	Ohio	3.6	2	True
four	2001	Nevada	2.4	3	False
five	2002	Nevada	2.9	4	False

  - del frame2[‘eastern’] # 사전형의 삭제와 동일

1. Pandas 자료 구조 소개

- 중첩된 사전형 데이터 활용
  - pop = {'Nevada':{2001: 2.4, 2002: 2.9}, 'Ohio':{2000: 1.5, 2001:1.7, 2002:3.6}} # {키1: {키2: 값2}} 키1은 열, 키2는 행
  - frame3 = pd.DataFrame(pop)
  - frame3

	Nevada	Ohio
2000	NaN	1.5
2001	2.4	1.7
2002	2.9	3.6

2. Pandas 핵심 기능

- reindex
  - 새로운 색인에 맞도록 객체를 재생성
  - obj=pd.Series([4.5, 7.2,-5.3, 3.6], index=[' d ' , ' b ' , ' a ' , ' c '])
  - obj
  - obj2 = obj.reindex([‘a’, ‘b’, ‘c’, ‘d’, ‘e’])

```
d     4.5
b     7.2
a    -5.3
c     3.6
dtype: float64
```

```
a    -5.3
b     7.2
c     3.6
d     4.5
e     NaN
dtype: float64
```

2. Pandas 핵심 기능

- reindex
  - obj.reindex(['a','b','c','d','e'], fill_value=0)

```
a   -5.3
b    7.2
c    3.6
d    4.5
e    0.0
dtype: float64
```

2. Pandas 핵심 기능

- reindex
  - obj3 = pd.Series(['blue','purple','yelow'], index=[0,2,4])
  - obj3
  - obj3.reindex(range(6), method = ‘ffill’)
  - obj3.reindex(range(6), method = ‘bfill’)

```
0       blue
2     purple
4      yelow
dtype: object
```

```
0       blue
1       blue
2     purple
3     purple
4      yelow
5      yelow
dtype: object
```

2. Pandas 핵심 기능

- 하나의 행 또는 열 삭제하기
  - obj = pd.Series(np.arange(5), index=['a','b','c','d','e'])
  - obj
  - new_obj = obj.drop('d')
  - new_obj

```
a    0
b    1
c    2
e    4
```

2. Pandas 핵심 기능

- Series 색인하기, 선택하기, 거르기
  - obj = pd.Series(np.arange(4), index=['a','b','c','d'])
  - obj['b'] / obj[1]
  - obj[2:4] / obj['b', 'c', 'd']
  - obj[[1,3]]
  - obj[obj <2]
  - obj[‘b’:’c’]

2. Pandas 핵심 기능

- DataFrame 색인하기, 선택하기, 거르기
  - data = pd.DataFrame(np.arange(16).reshape((4,4)), index=['Ohio', 'Colorade','Utah','NewYork'], columns=['one', 'two', 'three', 'four'])
  - data

	one	two	three	four
Ohio	0	1	2	3
Colorade	4	5	6	7
Utah	8	9	10	11
NewYork	12	13	14	15

2. Pandas 핵심 기능

- DataFrame 색인하기, 선택하기, 거르기
  - data[‘two’]
  - data[[‘three’, ‘one’]]
  - data[:2]
  - data[data[‘three’]>5]

2. Pandas 핵심 기능

- DataFrame 색인하기, 선택하기, 거르기
  - print(data.loc[:,["one"]])
  - print(data.loc['Colorado'])
  - print(data.loc['Colorado',['two','three']])
  - print(data.loc[['Colorado','Utah']])
  - print(data.loc[['Colorado','Utah'],['two','three']])

2. Pandas 핵심 기능

- DataFrame 정렬하기
  - print(data.sort_index())
  - print(data.sort_index(axis=1))

	one	two	three	four
Colorado	4	5	6	7
NewYork	12	13	14	15
Ohio	0	1	2	3
Utah	8	9	10	11

	four	one	three	two
Ohio	3	0	2	1
Colorado	7	4	6	5
Utah	11	8	10	9
NewYork	15	12	14	13

2. Pandas 핵심 기능

- DataFrame 정렬하기 – by
  - print(data.sort_values(by=['one']))
  - print(data.sort_values(by=['one', 'two']))

2. Pandas 핵심 기능

	one	two	three	four
Ohio	0	1	2	3
Colorado	4	5	6	7
Utah	8	9	10	11
NewYork	12	13	14	15

- DataFrame 기술통계 계산과 요약
  - print(data.sum())
  - print(data.sum(axis=0))

```
one      24
two      28
three    32
four     36
dtype: int64
```

  - print(data.sum(axis=1))

```
Ohio         6
Colorado    22
Utah        38
NewYork     54
dtype: int64
```

2. Pandas 핵심 기능

- DataFrame 기술통계 계산과 요약
  - print(data.cumsum())
  - print(data.idxmax())
  - print(data.describe())

	one	two	three	four
count	4.000000	4.000000	4.000000	4.000000
mean	6.000000	7.000000	8.000000	9.000000
std	5.163978	5.163978	5.163978	5.163978
min	0.000000	1.000000	2.000000	3.000000
25%	3.000000	4.000000	5.000000	6.000000
50%	6.000000	7.000000	8.000000	9.000000
75%	9.000000	10.000000	11.000000	12.000000
max	12.000000	13.000000	14.000000	15.000000

3. 계층적 색인

- 축에 대한 다중(둘 이상) 색인 단계를 지정할 수 있음
  - 고차원 데이터를 낮은 차원의 형식으로 다룰 수 있게 해줌
  - 계층적으로 색인된 객체는 데이터의 부분 집합을 부분적 색인으로 접근하는 것이 가능

3. 계층적 색인

- data = pd.Series(np.random.randn(10), index=[['a','a','a','b','b','b','c','c','d','d'],[1,2,3,1,2,3,1,2,2,3,]])

```
a  1    1.306733
   2    0.385137
   3   -0.747420
b  1   -0.895881
   2   -1.215567
   3   -1.320514
c  1    0.012055
   2   -0.328112
d  2    2.452141
   3   -0.691429
```

3. 계층적 색인

- print(data.index)

```
MultiIndex(levels=[['a', 'b', 'c', 'd'], [1, 2, 3]],
           labels=[[0, 0, 0, 1, 1, 1, 2, 2, 3, 3], [0, 1, 2, 0, 1,
2, 0, 1, 1, 2]])
```

- print(data[‘b’])
- print(data[‘b’:’c’])
- print(data.loc[['b','d']])
- data[:,2]

3. 계층적 색인

- data.unstack()

	1	2	3
a	1.306733	0.385137	-0.747420
b	-0.895881	-1.215567	-1.320514
c	0.012055	-0.328112	NaN
d	NaN	2.452141	-0.691429

- data.unstack().stack()

```
a  1    1.306733
   2    0.385137
   3   -0.747420
b  1   -0.895881
   2   -1.215567
   3   -1.320514
c  1    0.012055
   2   -0.328112
d  2    2.452141
   3   -0.691429
```

4. 병합과 조인

- df1 = pd.DataFrame({'employee': ['Bob', 'Jake', 'Lisa', 'Sue'], 'group': ['Accounting', 'Engineering', 'Engineering', 'HR']})

- df2 = pd.DataFrame({'employee': ['Lisa', 'Bob', 'Jake', 'Sue'], 'hire_date': [2004, 2008, 2012, 2014]})

- print(df1, df2)

4. 병합과 조인

df1

	employee	group
0	Bob	Accounting
1	Jake	Engineering
2	Lisa	Engineering
3	Sue	HR

df2

	employee	hire_date
0	Lisa	2004
1	Bob	2008
2	Jake	2012
3	Sue	2014

4. 병합과 조인 - 일대일 조인

- df3 = pd.merge(df1, df2)
- df3

	employee	group	hire_date
0	Bob	Accounting	2008
1	Jake	Engineering	2012
2	Lisa	Engineering	2004
3	Sue	HR	2014

4. 병합과 조인 – 다대일 조인

- df4 = pd.DataFrame({'group': ['Accounting', 'Engineering', 'HR'], 'supervisor': ['Carly', 'Guido', 'Steve']})
- print(df3)
- print(df4)

df3

	employee	group	hire_date
0	Bob	Accounting	2008
1	Jake	Engineering	2012
2	Lisa	Engineering	2004
3	Sue	HR	2014

df4

	group	supervisor
0	Accounting	Carly
1	Engineering	Guido
2	HR	Steve

4. 병합과 조인 – 다대일 조인

- print(pd.merge(df3, df4))

```
pd.merge(df3, df4)
```

	employee	group	hire_date	supervisor
0	Bob	Accounting	2008	Carly
1	Jake	Engineering	2012	Guido
2	Lisa	Engineering	2004	Guido
3	Sue	HR	2014	Steve

4. 병합과 조인 – 다대다조인

- df5 = pd.DataFrame({'group': ['Accounting', 'Accounting', 'Engineering', 'Engineering', 'HR', 'HR'], 'skills': ['math', 'spreadsheets', 'coding', 'linux', 'spreadsheets', 'organization']})

df1

	employee	group
0	Bob	Accounting
1	Jake	Engineering
2	Lisa	Engineering
3	Sue	HR

df5

	group	skills
0	Accounting	math
1	Accounting	spreadsheets
2	Engineering	coding
3	Engineering	linux
4	HR	spreadsheets
5	HR	organization

4. 병합과 조인 – 다대다조인

- print(pd.merge(df1, df5))

```
pd.merge(df1, df5)
```

	employee	group	skills
0	Bob	Accounting	math
1	Bob	Accounting	spreadsheets
2	Jake	Engineering	coding
3	Jake	Engineering	linux
4	Lisa	Engineering	coding
5	Lisa	Engineering	linux
6	Sue	HR	spreadsheets
7	Sue	HR	organization

4. 병합과 조인 – 키를 활용한 조인

- pd.merge(df1, df2, on='employee')

df1

	employee	group
0	Bob	Accounting
1	Jake	Engineering
2	Lisa	Engineering
3	Sue	HR

df2

	employee	hire_date
0	Lisa	2004
1	Bob	2008
2	Jake	2012
3	Sue	2014

pd.merge(df1, df2, on='employee')

	employee	group	hire_date
0	Bob	Accounting	2008
1	Jake	Engineering	2012
2	Lisa	Engineering	2004
3	Sue	HR	2014

4. 병합과 조인 – 키를 활용한 조인

- df3 = pd.DataFrame({'name': ['Bob', 'Jake', 'Lisa', 'Sue'], 'salary': [70000, 80000, 120000, 90000]})
- pd.merge(df1, df3, left_on="employee", right_on="name")

df1

	employee	group
0	Bob	Accounting
1	Jake	Engineering
2	Lisa	Engineering
3	Sue	HR

df3

	name	salary
0	Bob	70000
1	Jake	80000
2	Lisa	120000
3	Sue	90000

4. 병합과 조인 – 키를 활용한 조인

- df3 = pd.DataFrame({'name': ['Bob', 'Jake', 'Lisa', 'Sue'], 'salary': [70000, 80000, 120000, 90000]})
- pd.merge(df1, df3, left_on="employee", right_on="name")

df1

	employee	group
0	Bob	Accounting
1	Jake	Engineering
2	Lisa	Engineering
3	Sue	HR

df3

	name	salary
0	Bob	70000
1	Jake	80000
2	Lisa	120000
3	Sue	90000

4. 병합과 조인 – 키를 활용한 조인

- ppd.merge(df1, df3, left_on="employee", right_on="name").drop('name', axis=1)

	employee	group	name	salary
0	Bob	Accounting	Bob	70000
1	Jake	Engineering	Jake	80000
2	Lisa	Engineering	Lisa	120000
3	Sue	HR	Sue	90000

	employee	group	salary
0	Bob	Accounting	70000
1	Jake	Engineering	80000
2	Lisa	Engineering	120000
3	Sue	HR	90000

4. 병합과 조인 – 키를 활용한 조인

- df1a = df1.set_index('employee')
- df2a = df2.set_index('employee')
- print(df1a, df2a)

df1a

	group
employee	
Bob	Accounting
Jake	Engineering
Lisa	Engineering
Sue	HR

df2a

	hire_date
employee	
Lisa	2004
Bob	2008
Jake	2012
Sue	2014

4. 병합과 조인 – 키를 활용한 조인

- pd.merge(df1a, df2a, left_index=True, right_index=True)

	group	hire_date
employee		
Lisa	Engineering	2004
Bob	Accounting	2008
Jake	Engineering	2012
Sue	HR	2014

4. 병합과 조인 – 키를 활용한 조인

- df6 = pd.DataFrame({'name': ['Peter', 'Paul', 'Mary'], 'food': ['fish', 'beans', 'bread']},columns=['name', 'food'])
- df7 = pd.DataFrame({'name': ['Mary', 'Joseph'], 'drink': ['wine', 'beer']}, columns=['name', 'drink'])
- pd.merge(df6, df7)

df6

	name	food
0	Peter	fish
1	Paul	beans
2	Mary	bread

df7

	name	drink
0	Mary	wine
1	Joseph	beer

pd.merge(df6, df7)

	name	food	drink
0	Mary	bread	wine

4. 병합과 조인 – 키를 활용한 조인

- pd.merge(df6, df7, how='inner')

	name	food	drink
0	Mary	bread	wine

4. 병합과 조인 – 키를 활용한 조인

- pd.merge(df6, df7, how='outer')

	name	food
0	Peter	fish
1	Paul	beans
2	Mary	bread

	name	drink
0	Mary	wine
1	Joseph	beer

	name	food	drink
0	Peter	fish	NaN
1	Paul	beans	NaN
2	Mary	bread	wine
3	Joseph	NaN	beer

4. 병합과 조인 – 키를 활용한 조인

- df8 = pd.DataFrame({'name': ['Bob', 'Jake', 'Lisa', 'Sue'], 'rank': [1, 2, 3, 4]})
- df9 = pd.DataFrame({'name': ['Bob', 'Jake', 'Lisa', 'Sue'], 'rank': [3, 1, 4, 2]})
- pd.merge(df8, df9, on="name")

	name	rank
0	Bob	1
1	Jake	2
2	Lisa	3
3	Sue	4

	name	rank
0	Bob	3
1	Jake	1
2	Lisa	4
3	Sue	2

	name	rank_x	rank_y
0	Bob	1	3
1	Jake	2	1
2	Lisa	3	4
3	Sue	4	2

4. 병합과 조인 – 키를 활용한 조인

- pd.merge(df8, df9, on="name", suffixes=["_L", "_R"])

	name	rank
0	Bob	1
1	Jake	2
2	Lisa	3
3	Sue	4

	name	rank
0	Bob	3
1	Jake	1
2	Lisa	4
3	Sue	2

	name	rank_L	rank_R
0	Bob	1	3
1	Jake	2	1
2	Lisa	3	4
3	Sue	4	2

5. 피봇 테이블&데이터 분석 예

- import numpy as np
- import pandas as pd
- import seaborn as sns
- titanic = sns.load_dataset('titanic')

5. 피봇 테이블&데이터 분석 예

- titanic.head()

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no

5. 피봇 테이블&데이터 분석 예

- titanic.groupby('sex')[['survived']].mean()

	survived
sex	
female	0.742038
male	0.188908

- titanic.groupby(['sex', 'class'])['survived'].aggregate('mean').unstack()

class	First	Second	Third
sex			
female	0.968085	0.921053	0.500000
male	0.368852	0.157407	0.135447

5. 피봇 테이블&데이터 분석 예

- titanic.pivot_table('survived', index='sex', columns='class')

class	First	Second	Third
sex			
female	0.968085	0.921053	0.500000
male	0.368852	0.157407	0.135447

5. 피봇 테이블&데이터 분석 예

- age = pd.cut(titanic['age'], [0, 18, 80])
- titanic.pivot_table('survived', ['sex', age], 'class')

	class	First	Second	Third
sex	age			
female	(0, 18]	0.909091	1.000000	0.511628
	(18, 80]	0.972973	0.900000	0.423729
male	(0, 18]	0.800000	0.600000	0.215686
	(18, 80]	0.375000	0.071429	0.133663

5. 피봇 테이블&데이터 분석 예

- fare = pd.qcut(titanic['fare'], 2)
- titanic.pivot_table('survived', ['sex', age], [fare, 'class'])

	fare	[0, 14.454]			(14.454, 512.329]		
	class	First	Second	Third	First	Second	Third
sex	age						
female	(0, 18]	NaN	1.000000	0.714286	0.909091	1.000000	0.318182
	(18, 80]	NaN	0.880000	0.444444	0.972973	0.914286	0.391304
male	(0, 18]	NaN	0.000000	0.260870	0.800000	0.818182	0.178571
	(18, 80]	0.0	0.098039	0.125000	0.391304	0.030303	0.192308

5. 피봇 테이블&데이터 분석 예

- titanic.pivot_table(index='sex', columns='class', aggfunc={'survived':sum, 'fare':mean})

	fare			survived		
class	First	Second	Third	First	Second	Third
sex						
female	106.125798	21.970121	16.118810	91.0	70.0	72.0
male	67.226127	19.741782	12.661633	45.0	17.0	47.0