데이터분석을위한 PANDAS

BASIC INDEX

인덱스활용

인덱스활용

◆ 데이터프레임(DataFrame)

▪ 특정 열 ➔ 행 인덱스로 설정

```
DataFrame.set_index(  keys,                  # 새로운 행 인덱스 컬럼
                      drop=True,             # 지정된 컬럼 삭제 여부
                      append=False,          # 기존 행인덱스에 추가
                      inplace=False,         # 원본 수정 여부 [기 False]
                      verify_integrity=False #새 인덱스에 중복 값 여부 검사
                      )
```

인덱스활용

◆ 데이터프레임(DataFrame)

열/컬럼 ➔ 행인덱스 설정

```
## DataFrame 생성
data = {  "date": ["2025-10-20", "2025-10-21", "2025-10-21"],
          "city": ["Seoul", "Busan", "Seoul"],
          "value": [10, 20, 30] }
dataDF = pd.DataFrame(data)
```

	date	city	value
0	2025-10-20	Seoul	10
1	2025-10-21	Busan	20
2	2025-10-21	Seoul	30

```
##-> 1) 단일 열을 인덱스로
##->    drop=True 기본 → 'date' 열 본문에서 사라짐
df1 = dataDF.set_index("date")
```

	city	value
date		
2025-10-20	Seoul	10
2025-10-21	Busan	20
2025-10-21	Seoul	30

인덱스활용

◆ 데이터프레임(DataFrame)

열/컬럼 ➔ 행인덱스 설정

```
##-> 2) 열을 인덱스로 올리되 열도 유지
##->    본문에 'date' 열이 그대로 남아 있음
df2 = dataDF.set_index("date", drop=False)
```

	date	city	value
date			
2025-10-20	2025-10-20	Seoul	10
2025-10-21	2025-10-21	Busan	20
2025-10-21	2025-10-21	Seoul	30

```
##-> 3) 여러 열로 MultiIndex 만들기
##->    2계층 인덱스
df3 = dataDF.set_index(["date", "city"])
```

		value
date	**city**	
2025-10-20	Seoul	10
2025-10-21	Busan	20
	Seoul	30

인덱스활용

◆ 데이터프레임(DataFrame)

▪ 행 인덱스 재배열 → 미존재 행인덱스 값은 NaN

```
DataFrame/Series.reindex(   labels=None,        # 새로운 행인덱스/열이름
                            index=None,         # 새로운 행인덱스
                            columns=None,       # 새로운 열이름
                            axis=None,          # 행/열 방향지정, labels와 함께
                            method=None,        # 미 존재 데이터 채울 방법 설정
                            level=None,
                            fill_value=nan,     # 미 존재 데이터 채울 값
                            limit=None,         # 연속해서 채울 수 있는 최대 갯수
                            tolerance=None      # 허용 오차
                        )
```

인덱스활용

◆ 데이터프레임(DataFrame)

열인덱스 재배열

```
##-> 1) 데이터프레임 생성
data={'이름':['홍길동','베트맨','마징가'],
      '국어':[90, 89, 100],
      '수학':[78, 100, 90],
      '음악':[88, 99, 77],
      '과학':[90, 76, 65]}

dataDF=pd.DataFrame(data)
print(dataDF)
```

	이름	국어	수학	음악	과학
0	홍길동	90	78	88	90
1	베트맨	89	100	99	76
2	마징가	100	90	77	65

인덱스활용

◆ 데이터프레임(DataFrame)

열인덱스 재배열

```
## ----------------------------------------
## 컬럼을 행인덱스로 설정
## - 메서드: DF변수명.set_index(컬럼명)
##          inplace 매개변수 = False
## ----------------------------------------
## 이름 컬럼 ==> 행 인덱스로 설정
dataDF2=dataDF.set_index('이름')
```

	이름	국어	수학	음악	과학
0	홍길동	90	78	88	90
1	베트맨	89	100	99	76
2	마징가	100	90	77	65

	국어	수학	음악	과학
이름				
홍길동	90	78	88	90
베트맨	89	100	99	76
마징가	100	90	77	65

인덱스활용

◆ 데이터프레임(DataFrame)

열인덱스 재배열

```
## => 기존 행인덱스 + 새로운 행인덱스
## fill_value 매개변수 설정
dataDF3=dataDF2.reindex(new_index, fill_value=0)

display(dataDF3)
```

	이름	국어	수학	음악	과학
0	홍길동	90	78	88	90
1	베트맨	89	100	99	76
2	마징가	100	90	77	65

재배열

이름	국어	수학	음악	과학
홍길동	90	78	88	90
베트맨	89	100	99	76
마징가	100	90	77	65
원더우먼	0	0	0	0

인덱스활용

◆ 데이터프레임(DataFrame)

▪ 행 인덱스 초기화 → 정수형 인덱스로 초기화

→ 기존 행인덱스는 컬럼에 추가

```
DataFrame/Series.reset_index( level=None,          # MultiIndex 전용) 해제할 인덱스 레벨
                              drop=False,          # 현재 인덱스 삭제 여부
                              inplace=False,       # 원본 적용 여부 설정
                              col_level=0,         # 열이 MultiIndex일 때 설정
                              col_fill='',         # 열이 MultiIndex일 때 설절
                              allow_duplicates=<no_default>, # 중복 열 여부
                              names=None           # 컬럼으로 들어오는 인덱스 이름
                            )
```

인덱스활용

◆ 데이터프레임(DataFrame)

인덱스 초기화

```
##-> 1) 데이터프레임 생성
dataDF = pd.DataFrame([ ('bird', 389.0),
                        ('bird', 24.0),
                        ('mammal', 80.5),
                        ('mammal', np.nan)],
                       index=['falcon', 'parrot', 'lion', 'monkey'],
                       columns=('class', 'max_speed'))

display(dataDF)
```

	class	max_speed
falcon	bird	389.0
parrot	bird	24.0
lion	mammal	80.5
monkey	mammal	NaN

인덱스활용

◆ 데이터프레임(DataFrame)

인덱스 초기화

```
# (2) 행인덱스 초기화
dataDF.reset_index()
```

	class	max_speed
falcon	bird	389.0
parrot	bird	24.0
lion	mammal	80.5
monkey	mammal	NaN

초기화

	index	class	max_speed
0	falcon	bird	389.0
1	parrot	bird	24.0
2	lion	mammal	80.5
3	monkey	mammal	NaN

인덱스활용

◆ 데이터프레임(DataFrame)

인덱스 초기화

```
## (2) 행인덱스 초기화
dataDF2=dataDF.reset_index()
dataDF2.reset_index()
```

	index	class	max_speed
0	falcon	bird	389.0
1	parrot	bird	24.0
2	lion	mammal	80.5
3	monkey	mammal	NaN

초기화

	level_0	index	class	max_speed
0	0	falcon	bird	389.0
1	1	parrot	bird	24.0
2	2	lion	mammal	80.5
3	3	monkey	mammal	NaN

인덱스활용

◆ 데이터프레임(DataFrame)

인덱스 초기화

```
## ------------------------------------------
## 열인덱스/열이름 초기화
## ------------------------------------------
## (1) columns 속성 설정
dataDF2.columns = pd.RangeIndex(dataDF2.shape[1])
display(dataDF2)

## (2) 축 설정 메서드 : set_axis(labels, axis=0)
dataDF2.set_axis(range(dataDF2.shape[1]), axis=1)
display(dataDF2)
```

	class	max_speed
falcon	bird	389.0
parrot	bird	24.0
lion	mammal	80.5
monkey	mammal	NaN

초기화

	0	1
falcon	bird	389.0
parrot	bird	24.0
lion	mammal	80.5
monkey	mammal	NaN

인덱스활용

◆ 데이터프레임(DataFrame)

▪ 행/열 인덱스기준 정렬

```
DataFrame/Series.sort_index(    axis=0,                 # 정렬 방향 설정
                                ascending=True,         # 정렬 방식 [기: 오름차순]
                                inplace=False,          # 원본 적용 여부
                                kind='quicksort',       # 정렬 알고리즘
                                na_position='last',     # 결측치 순서
                                ignore_index=False      # 기존 인덱스 무시 여부
                            )
```

인덱스활용

◆ 데이터프레임(DataFrame)

인덱스 정렬

```
##-> 1) 데이터프레임 생성
dataDF = pd.DataFrame(  [[1, 2, 3, 4, 5],
                         [11,22,33,44,55]] ,
                        index=[100, 29],
                        columns=['A','D','E','B','C'])

display(dataDF)
```

	A	D	E	B	C
100	1	2	3	4	5
29	11	22	33	44	55

인덱스활용

◆ 데이터프레임(DataFrame)

인덱스 정렬

```
##-> 2) 행 인덱스 정렬
##->    [기] 오름차순 정렬
dataDF.sort_index()

##->    [변] 내림차순 정렬
dataDF.sort_index(ascending=False)
```

	A	D	E	B	C
100	1	2	3	4	5
29	11	22	33	44	55

	A	D	E	B	C
100	1	2	3	4	5
29	11	22	33	44	55

인덱스활용

◆ 데이터프레임(DataFrame)

▪ 데이터/값 기준 정렬

```
DataFrame/Series.sort_values(    by,                 # 정렬할 컬럼/행 이름
                                 axis=0,             # 정렬 방향 설정
                                 ascending=True,     # 정렬 방식 [기: 오름차순]
                                 inplace=False,      # 원본 적용 여부
                                 kind='quicksort',   # 정렬 알고리즘
                                 na_position='last', # 결측치 순서
                                 ignore_index=False  # 기존 인덱스 무시 여부
)
```

인덱스활용

◆ 데이터프레임(DataFrame)

데이터 정렬

```
##-> 1) 데이터프레임 생성
dataDF = pd.DataFrame(  [[1, 22, 39, 4, 85],
                         [11,22, 53, 7,55]] ,
                        index=[100, 29],
                        columns=['A','D','E','B','C'])

display(dataDF)
```

	A	D	E	B	C
100	1	22	39	4	85
29	11	22	53	7	55

인덱스활용

◆ 데이터프레임(DataFrame)

데이터 정렬

```
##-> C 컬럼 기준 오름차순 정렬
dataDF.sort_values(by=['C'])

##-> C 컬럼 기준 내림차순 정렬
dataDF.sort_values(by=['C'], ascending=False)
```

	A	D	E	B	C
29	11	22	53	7	55
100	1	22	39	4	85

	A	D	E	B	C
100	1	22	39	4	85
29	11	22	53	7	55

인덱스활용

◆ 데이터프레임(DataFrame)

데이터 정렬

```
##-> C,B 컬럼 기준 오름차순 정렬
dataDF.sort_values(by=['C','B'])

##-> B,C 컬럼 기준 오름차순 정렬
dataDF.sort_values(by=['B','C'])
```

	A	D	E	B	C
100	1	22	39	4	85
29	11	22	53	7	55

C, B

	A	D	E	B	C
29	11	22	53	7	55
100	1	22	39	4	85

B, C

	A	D	E	B	C
100	1	22	39	4	85
29	11	22	53	7	55