설정변경코드

- 변수 명이 두번 이상 출력되어도 모두 콘솔에서 보여줄 것
- from IPython.core.interactiveshell import InteractiveShell
  InteractiveShell.ast_node_interactivity="all"
- InteractiveShell.ast_node_interactivity : 'all' | 'last' | 'last_expr' | 'none' (기본값은 'last_expr')

In [1]:

```
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity="all"
```

데이터 프레임 인덱서 : loc, iloc

- Pandas는 numpy행렬과 같이 쉼표를 사용한 (행 인덱스, 열 인덱스) 형식의 2차원 인덱싱을 지원
  - 특별한 인덱서(indexer) 속성을 제공
- loc : 라벨값 기반의 2차원 인덱싱
- iloc : 순서를 나타내는 정수 기반의 2차원 인덱싱

In [2]:

```
import pandas as pd
import numpy as np
```

행과 열을 동시에 인덱싱 하는 구조는 기본 자료구조 인덱스와 차이가 있음

- df['열']
- df[:'행'] 슬라이싱이 반드시 필요
- df['열'][:'행']

데이터 프레임에서 인덱서 사용

loc, iloc 속성을 사용하는 인덱싱

pandas 패키지는 [행번호,열번호] 인덱싱 불가

```
- iloc 속성 사용하면 가능
    - iloc[행번호,열번호] - 가능
    - loc[행제목,열제목] -가능
```

loc 인덱서 : 행 우선 인덱서

- df[열이름값] # 기본 인덱싱, 열우선
- df.loc[행인덱싱 값] #행우선 인덱서
- df.loc[행인덱싱 값,열인덱싱 값]

인덱싱 값

1. 인덱스 데이터(index name, column name)
2. 인덱스 데이터 슬라이스
3. 같은 행 인덱스를 갖는 불리언 시리즈(행 인덱싱인 경우)
    - 조건으로 추출 가능
4. 위 값을 반환하는 함수

In [4]:

```
# 예제 DF  생성
# 10-21 범위의 숫자를  value로 갖는 3행 4열의 df
df = pd.DataFrame(np.arange(10,22).reshape(3,4),
                  index=['a','b','c'],
                  columns = ["A","B","C","D"])
df
```

Out[4]:

	A	B	C	D
a	10	11	12	13
b	14	15	16	17
c	18	19	20	21

In [5]:

```
# loc 인덱서 사용
# 인덱스 값을 하나만 받는 경우
# a행의 모든 열 추출
# 시리즈로 반환
df.loc['a']
```

Out[5]:

```
A    10
B    11
C    12
D    13
Name: a, dtype: int32
```

loc 인덱서에서는 열 단독 인덱싱은 불가능 함

In [6]:

```
# loc인덱서를 이용해서 A열의 데이터를 추출하시오
# df.loc['A'] #KeyError
```

In [7]:

```
# 여러행 추출 -  슬라이싱 가능 (df반환)
df.loc['b':'c']
df['b':'c']
```

Out[7]:

	A	B	C	D
b	14	15	16	17
c	18	19	20	21

Out[7]:

	A	B	C	D
b	14	15	16	17
c	18	19	20	21

In [8]:

```
# 비연속적인 행을 추출 - 인덱스값으로 list 사용
df.loc[['b','c']]
```

Out[8]:

	A	B	C	D
b	14	15	16	17
c	18	19	20	21

In [9]:

```
# df.loc['b','c'] #  b행의 c열을 참조의미 c열이 없으므로 에러가 발생
```

In [10]:

```
# 하나의 행을 추출할때 반환 결과 타입을 다르게
df.loc["b"] # 시리즈 반환
df.loc[["b"]] # df 형태로 반환
```

Out[10]:

```
A    14
B    15
C    16
D    17
Name: b, dtype: int32
```

Out[10]:

	A	B	C	D
b	14	15	16	17

In [11]:

```
# df[["b","c"]]
# 위 인덱싱 출력 결과 및 이유를 쓰시오
# key error 발생, b와 c를 컬럼명에서 찾고 있음
df.loc[['b','c']]

df[["B","C"]] # 열우선인덱싱
```

Out[11]:

	A	B	C	D
b	14	15	16	17
c	18	19	20	21

Out[11]:

	B	C
a	11	12
b	15	16
c	19	20

boolean selection으로 row 선택하기

In [12]:

```
df
```

Out[12]:

	A	B	C	D
a	10	11	12	13
b	14	15	16	17
c	18	19	20	21

In [13]:

```
df.A # 시리즈 반환
df.A > 15 # 결과는 불리언시리즈
```

Out[13]:

```
a    10
b    14
c    18
Name: A, dtype: int32
```

Out[13]:

```
a    False
b    False
c     True
Name: A, dtype: bool
```

In [14]:

```
# df의 A열의 값이 15보다 큰 행을 추출
# 조건식을 인덱스로 사용
df.loc[df.A>15]
```

Out[14]:

	A	B	C	D
c	18	19	20	21

불리언 시리즈를 반환하는 함수를 인덱스로 사용

In [15]:

```
# 불리언 시리즈를 반환하는 함수 작성
# 함수명  : sel_row(df)
# df를 전달받아서 df의 A 컬럼 데이터에 대해 15보다 큰지 조건검사 후 결과를 반환
def sel_row(df) :
    return df.A>15
```

In [16]:

```
df
sel_row(df)
# 함수로 결과값을 전달받아 인덱스로 사용
df.loc[sel_row(df)]
```

Out[16]:

	A	B	C	D
a	10	11	12	13
b	14	15	16	17
c	18	19	20	21

Out[16]:

```
a    False
b    False
c     True
Name: A, dtype: bool
```

Out[16]:

	A	B	C	D
c	18	19	20	21

loc 인덱서 슬라이싱

In [17]:

```
#예제 df 생성
df2 = pd.DataFrame(np.arange(10,26).reshape(4,4),
                   columns=['a','b','c','d'])
df2 # 행인덱스는 지정하지 않아서 0부터 1씩 증가되는 정수 인덱스 자동 생성
# np.arange(10,26).reshape(4,4)
```

Out[17]:

	a	b	c	d
0	10	11	12	13
1	14	15	16	17
2	18	19	20	21
3	22	23	24	25

In [18]:

```
# loc의 슬라이싱 [초기인덱스값:마지막인덱스값]
# !!!! 주의

df2.loc[1:2]
df2[1:2] # 위치 인덱싱
```

Out[18]:

	a	b	c	d
1	14	15	16	17
2	18	19	20	21

Out[18]:

	a	b	c	d
1	14	15	16	17

loc 인덱서 사용 요소 값 접근

- 인덱싱으로 행과 열을 모두 받는 경우
- 문법 : df.loc[행인덱스,열인덱스] - 라벨(문자열)인덱스 사용

In [19]:

```
df2
df2.loc[0,'a']
```

Out[19]:

	a	b	c	d
0	10	11	12	13
1	14	15	16	17
2	18	19	20	21
3	22	23	24	25

Out[19]:

10

In [20]:

```
df
```

Out[20]:

	A	B	C	D
a	10	11	12	13
b	14	15	16	17
c	18	19	20	21

In [21]:

```
# df의 a행의 A열 값을 추출
df.loc['a','A']
```

Out[21]:

```
10
```

In [22]:

```
# loc인덱서를 사용한 원소값을 변경
# df.loc[행,열] = 값
df.loc['a','A'] = 50
df
```

Out[22]:

	A	B	C	D
a	50	11	12	13
b	14	15	16	17
c	18	19	20	21

In [23]:

```
df.loc[['a','b']]['A'] # 시리즈
df.loc[['a','b'],'A'] # 시리즈
df.loc[['a','b'],['A']] # df 반환
```

Out[23]:

```
a    50
b    14
Name: A, dtype: int32
```

Out[23]:

```
a    50
b    14
Name: A, dtype: int32
```

Out[23]:

	A
a	50
b	14

loc를 이용한 indexing 정리

In [24]:

```
# a행의 모든열 추출
df.loc['a'] # a행 모든열 추출, 시리즈로 반환
df.loc[['a']] # a행의 모든 열 추출, df 반환
df.loc['a',:] # a행의 모든 열 추출, 시리즈
df.loc[['a'],:] # a행의 모든 열 추출, df 반환
```

Out[24]:

```
A    50
B    11
C    12
D    13
Name: a, dtype: int32
```

Out[24]:

	A	B	C	D
a	50	11	12	13

Out[24]:

```
A    50
B    11
C    12
D    13
Name: a, dtype: int32
```

Out[24]:

	A	B	C	D
a	50	11	12	13

In [25]:

```
#a행의 B,C열을 추출하시오
df.loc['a'] # a행의 모든열 시리즈로 반환
df.loc['a', 'B':'C'] # 시리즈로 반환
df.loc[['a']] # df로 반환
df.loc[['a'],"B":"C"] # df로 반환

df.loc['a', ['B','C']] # 시리즈로 반환
```

Out[25]:

```
A    50
B    11
C    12
D    13
Name: a, dtype: int32
```

Out[25]:

```
B    11
C    12
Name: a, dtype: int32
```

Out[25]:

	A	B	C	D
a	50	11	12	13

Out[25]:

	B	C
a	11	12

Out[25]:

```
B    11
C    12
Name: a, dtype: int32
```

In [26]:

```
# B행부터 모든행의 A열을 추출
df.loc['b':] #b행부터 모든행
df.loc['b':,'A'] # 시리즈 반환
df.loc['b':]['A'] #시리즈 반환
df.loc['b':][['A']] # df반환
df.loc['b':,['A']] # df반환
df.loc['b':,'A':'A']
```

Out[26]:

	A	B	C	D
b	14	15	16	17
c	18	19	20	21

Out[26]:

```
b    14
c    18
Name: A, dtype: int32
```

Out[26]:

```
b    14
c    18
Name: A, dtype: int32
```

Out[26]:

	A
b	14
c	18

Out[26]:

	A
b	14
c	18

Out[26]:

	A
b	14
c	18

In [27]:

```
# a,b 행의 B,D열을 데이터 프레임으로 반환
df.loc['a':'b']
df.loc[['a','b']]
df.loc[['a','b']][['B','D']]
df.loc[['a','b'],['B','D']]
```

Out[27]:

	A	B	C	D
a	50	11	12	13
b	14	15	16	17

Out[27]:

	A	B	C	D
a	50	11	12	13
b	14	15	16	17

Out[27]:

	B	D
a	11	13
b	15	17

Out[27]:

	B	D
a	11	13
b	15	17

iloc 인덱서(위치 인덱스)

- 라벨(name)이 아닌 위치를 나타내는 **정수** 인덱스만 받는다.
- 위치 정수값은 0부터 시작
- 데이터프레임.iloc[행,열]

In [28]:

```
df # 컬럼명과 로우명이 문자형으로 설정됨
df.iloc[0,1]
```

Out[28]:

	A	B	C	D
a	50	11	12	13
b	14	15	16	17
c	18	19	20	21

Out[28]:

11

In [29]:

```
# iloc에 슬라이싱 적용 # 행과 열 모두 슬라이싱 적용 - df
df.iloc[0:2,1:2]
```

Out[29]:

	B
a	11
b	15

In [30]:

```
df.iloc[0:2]
df.iloc[0:2,1] #시리즈 반황
```

Out[30]:

	A	B	C	D
a	50	11	12	13
b	14	15	16	17

Out[30]:

```
a    11
b    15
Name: B, dtype: int32
```

In [31]:

```
df.iloc[2]
df.iloc[2,1:2]
```

Out[31]:

```
A    18
B    19
C    20
D    21
Name: c, dtype: int32
```

Out[31]:

```
B    19
Name: c, dtype: int32
```

In [32]:

```
# df 형태 추출 : df.iloc[행슬라이싱, 열슬라이싱]
df.iloc[2:3,1:2]
```

Out[32]:

	B
c	19

In [33]:

```
# 0행 데이터에서 끝에서 두번째 열부터 끝까지 반환
df
df.iloc[0:1,-2:] # df
df.iloc[0,-2:] # 시리즈
```

Out[33]:

	A	B	C	D
a	50	11	12	13
b	14	15	16	17
c	18	19	20	21

Out[33]:

	C	D
a	12	13

Out[33]:

```
C    12
D    13
Name: a, dtype: int32
```

In [34]:

```
df.iloc[[0,1],[1,2]]
```

Out[34]:

	B	C
a	11	12
b	15	16

- 원소들을 분류하여 갯수를 세는 함수 : value_counts()

In [35]:

```
df = pd.DataFrame({'num_legs': [2, 4, 4, 6],
                   'num_wings': [2, 0, 0, 0]},
                  index=['falcon', 'dog', 'cat', 'ant'])
df
```

Out[35]:

	num_legs	num_wings
falcon	2	2
dog	4	0
cat	4	0
ant	6	0

In [36]:

```
df.num_legs
df.num_legs.value_counts()
```

Out[36]:

```
falcon    2
dog       4
cat       4
ant       6
Name: num_legs, dtype: int64
```

Out[36]:

```
4    2
2    1
6    1
Name: num_legs, dtype: int64
```

In [37]:

```
# 동일한 값을 갖는 행의 수를 반환
df.value_counts()
```

Out[37]:

```
num_legs  num_wings
4         0           2
2         2           1
6         0           1
dtype: int64
```

In [ ]:

```

```

In [ ]:

```

```