IPA 주관 인공지능센터 기본(fundamental) 과정

- GitHub link: here
- E-Mail: windkyle7@gmail.com

Pandas

Why Pandas ?

`판다스(Pandas)` 를 사용하는 목적은 대표적으로 **두 가지** 이유가 있다.

- EDA(Exploratory data analysis)
  - 통계값 분석
  - Visualization
- Preprocessing

What is Pandas ?

쉽게 말하면 `판다스 = 엑셀` 이다. 다만 빅데이터를 다룰 때 엑셀로 데이터를 분석하려고 하면 엄청나게 느리고 메모리 리소스도 많이 잡아먹기 때문에 판다스를 사용하면 굉장히 빠르고 간편하게 사용할 수 있다.

- 왜 빠른가?? --> 판다스 역시 NumPy 기반으로 만들어졌기 때문에.

판다스 처음 사용해보기

먼저 `pandas` 를 임포트한다. 그리고 alias는 **관례적**으로 `pd` 를 사용한다.

In [1]:
```
import pandas as pd
```

현재 실습에서 사용하는 판다스 버전은 다음과 같다.

In [2]:
```
pd.__version__
```

Out[2]:
```
'0.24.2'
```

실습에 사용할 데이터는 공공데이터포털에서 제공하는 교통사고 데이터를 가지고 진행하고자 한다.

공공데이터포털은 여기로 들어가서 교통사고 데이터를 다운로드 받은 후에 `csv` 파일을 불러온다.

데이터를 불러올 시, `UnicodeDecodeError` 예외가 발생한다면 가장 먼저 `engine` 인자값에 `python` 을 먼저 넣어주고 그 다음에 인코딩을 시도해본다.

In [3]:
```
data = pd.read_csv('data/accident01.csv', engine='python', encoding='cp949')
```

그럼 불러온 데이터를 한번 확인해보자.

In [4]:
```
data[:10]
```

Out[4]:

	시도	시군구	월	발생건수	사망자수	부상자수	중상	경상	부상신고
0	서울	종로구	01월	103	0	133	23	90	20
1	서울	종로구	02월	84	1	111	20	77	14
2	서울	종로구	03월	101	0	149	34	91	24
3	서울	종로구	04월	93	0	114	27	78	9
4	서울	종로구	05월	102	1	143	24	96	23
5	서울	종로구	06월	96	1	123	32	84	7
6	서울	종로구	07월	89	0	128	28	83	17
7	서울	종로구	08월	90	0	127	29	85	13
8	서울	종로구	09월	91	1	133	24	96	13
9	서울	종로구	10월	113	0	166	30	109	27

불러온 데이터는 `데이터 프레임(DataFrame)` 타입이다.

In [5]:
```
type(data)
```

Out[5]:
```
pandas.core.frame.DataFrame
```

어떤 데이터들이 있는지 한번 확인해본다.

In [6]:
```
data.values
```

Out[6]:
```
array([['서울', '종로구', '01월', ..., 23, 90, 20],
```

```
Out[6]: array([['서울', '종로구', '01월', ..., 23, 90, 20],
               ['서울', '종로구', '02월', ..., 20, 77, 14],
               ['서울', '종로구', '03월', ..., 34, 91, 24],
               ...,
               ['세종', '세종', '10월', ..., 22, 66, 2],
               ['세종', '세종', '11월', ..., 17, 71, 3],
               ['세종', '세종', '12월', ..., 13, 69, 1]], dtype=object)
```

Tip. 경로를 모를 경우...

경로를 알기 위해 `os` 패키지에서 path 모듈을 사용한다.

```
In [7]: import os
```

OS마다 표현하는 방식이 다르기 때문에 `os.path.curdir` 로 현재 경로를 상대 경로로써 사용할 때 어떤식으로 표현하는지 알아본다.

```
In [8]: os.path.curdir
Out[8]: '.'
```

`os` 패키지에서 `abspath` 는 절대 경로를 가져온다. 현재 실습하고 있는 프로젝트의 절대 경로를 얻어온다.

```
In [9]: os.path.abspath('.')
Out[9]: '/home/user/workspace'
```

Seaborn

씨본(seaborn)은 `데이터 시각화(Data Visualization)` 를 위한 패키지이다.

씨본은 alias를 **관례적**으로 `sns` 를 사용한다.

```
In [10]: import seaborn as sns
```

tips 데이터로 분석해보기

간단한 실습을 위해 씨본에서 `Tidy-data` 를 불러오는데, 데이터는 `tips` 데이터를 사용하고자 한다.

```
In [11]: tips = sns.load_dataset('tips')
```

불러온 데이터가 어떤식으로 되어있는지 살펴본다.

```
In [12]: tips.shape
Out[12]: (244, 7)
```

```
In [13]: tips.dtypes
Out[13]: total_bill     float64
         tip            float64
         sex           category
         smoker        category
         day           category
         time          category
         size             int64
         dtype: object
```

불러온 데이터가 어떤식으로 구성되어 있는지 알기 위해서 NumPy처럼 위와 같이 `shape` 와 `dtypes` 를 사용할 수도 있지만, 판다스에서는 훨씬 더 많은 것들을 제공한다.

Pandas 데이터 분석 - 5총사

info

판다스에서는 다음과 같이 `info()` 메소드를 사용한다. 결과의 `RangeIndex` 란 판다스가 알아서 데이터에 인덱스를 붙여주는데, 이를 의미한다.

```
In [14]: tips.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 7 columns):
total_bill    244 non-null float64
tip           244 non-null float64
sex           244 non-null category
```

```
smoker        244 non-null category
day           244 non-null category
time          244 non-null category
size          244 non-null int64
dtypes: category(4), float64(2), int64(1)
memory usage: 7.2 KB
```

In [15]:
```
tips.info(verbose=False)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Columns: 7 entries, total_bill to size
dtypes: category(4), float64(2), int64(1)
memory usage: 7.2 KB
```

describe

통계값을 보기 위해서는 다음과 같이 `describe()` 메소드를 사용한다. `describe`는 기본적으로 숫자로 이루어진 데이터들에 대한 통계를 보애준다.

In [16]:
```
tips.describe()
```

Out[16]:

	total_bill	tip	size
count	244.000000	244.000000	244.000000
mean	19.785943	2.998279	2.569672
std	8.902412	1.383638	0.951100
min	3.070000	1.000000	1.000000
25%	13.347500	2.000000	2.000000
50%	17.795000	2.900000	2.000000
75%	24.127500	3.562500	3.000000
max	50.810000	10.000000	6.000000

`category` 데이터에 한정해서 빈도수 등을 알아보고자 할 때, 다음과 같이 `include`에 인자값을 넣어준다.

In [17]:
```
tips.describe(include='category')
```

Out[17]:

	sex	smoker	day	time
count	244	244	244	244
unique	2	2	4	2
top	Male	No	Sat	Dinner
freq	157	151	87	176

`float` 데이터에 대해 알아보고 싶다면 다음과 같이 `float`를 넣어준다.

In [18]:
```
tips.describe(include='float')
```

Out[18]:

	total_bill	tip
count	244.000000	244.000000
mean	19.785943	2.998279
std	8.902412	1.383638
min	3.070000	1.000000
25%	13.347500	2.000000
50%	17.795000	2.900000
75%	24.127500	3.562500
max	50.810000	10.000000

`int` 데이터도 마찬가지다.

In [19]:
```
tips.describe(include='int')
```

Out[19]:

	size
count	244.000000
mean	2.569672
std	0.951100
min	1.000000
25%	2.000000
50%	2.000000
75%	3.000000
max	6.000000

여러 데이터들을 알고 싶을 때, 다음과 같이 리스트 형태로 넣어준다.

In [20]:
```
tips.describe(include=['int', 'category'])
```

Out[20]:

	sex	smoker	day	time	size
count	244	244	244	244	244.000000
unique	2	2	4	2	NaN
top	Male	No	Sat	Dinner	NaN
freq	157	151	87	176	NaN
mean	NaN	NaN	NaN	NaN	2.569672
std	NaN	NaN	NaN	NaN	0.951100
min	NaN	NaN	NaN	NaN	1.000000
25%	NaN	NaN	NaN	NaN	2.000000
50%	NaN	NaN	NaN	NaN	2.000000
75%	NaN	NaN	NaN	NaN	3.000000
max	NaN	NaN	NaN	NaN	6.000000

head

`head` 는 가장 앞에서부터 5개까지 데이터를 보여준다.

In [21]:
```
tips.head()
```

Out[21]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

첫번째 인자값으로 가져오고 싶은 데이터의 개수를 입력해줄 수 있다.

In [22]:
```
tips.head(3)
```

Out[22]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3

tail

`tail` 은 가장 뒤에서부터 5개까지 데이터를 보여준다.

In [23]:
```
tips.tail()
```

Out[23]:

	total_bill	tip	sex	smoker	day	time	size
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

In [24]:
```
tips.tail(3)
```

Out[24]:

	total_bill	tip	sex	smoker	day	time	size
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

sample

`sample` 은 랜덤하게 데이터를 하나 뽑아서 보여준다.

In [25]:
```
tips.sample()
```

Out[25]:

	total_bill	tip	sex	smoker	day	time	size
50	12.54	2.5	Male	No	Sun	Dinner	2

마찬가지로 개수를 정해줄 수 있다. 물론 랜덤하게 가져온다.

In [26]:
```
tips.sample(5)
```

Out[26]:

	total_bill	tip	sex	smoker	day	time	size
173	31.85	3.18	Male	Yes	Sun	Dinner	2
128	11.38	2.00	Female	No	Thur	Lunch	2
186	20.90	3.50	Female	Yes	Sun	Dinner	3
84	15.98	2.03	Male	No	Thur	Lunch	2
172	7.25	5.15	Male	Yes	Sun	Dinner	2

다시 교통사고 데이터로..

이제 실제로 교통사고 데이터를 가지고 분석을 시도해본다.

먼저, 데이터를 불러온다.

In [27]:
```
data = pd.read_csv('data/accident01.csv', engine='python', encoding='cp949')
```

불러온 데이터를 앞서 언급했던 **5총사: info, describe, head, tail, sample**로 확인해본다.

In [28]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2745 entries, 0 to 2744
Data columns (total 9 columns):
시도       2745 non-null object
시군구      2745 non-null object
월        2745 non-null object
발생건수     2745 non-null int64
사망자수     2745 non-null int64
부상자수     2745 non-null int64
중상       2745 non-null int64
경상       2745 non-null int64
부상신고     2745 non-null int64
dtypes: int64(6), object(3)
memory usage: 193.1+ KB
```

In [29]:
```
data.describe(include='all')
```

Out[29]:

	시도	시군구	월	발생건수	사망자수	부상자수	중상	경상	부상신고
count	2745	2745	2745	2745.00000	2745.000000	2745.000000	2745.000000	2745.000000	2745.000000
unique	17	206	12	NaN	NaN	NaN	NaN	NaN	NaN
top	경기	동구	07월	NaN	NaN	NaN	NaN	NaN	NaN
freq	372	72	229	NaN	NaN	NaN	NaN	NaN	NaN
mean	NaN	NaN	NaN	79.10674	1.377413	117.681967	27.052095	82.881967	7.747905
std	NaN	NaN	NaN	77.92157	1.553821	117.356238	24.090962	87.130782	10.627361
min	NaN	NaN	NaN	1.00000	0.000000	1.000000	0.000000	0.000000	0.000000
25%	NaN	NaN	NaN	21.00000	0.000000	32.000000	10.000000	20.000000	1.000000
50%	NaN	NaN	NaN	57.00000	1.000000	84.000000	21.000000	55.000000	4.000000
75%	NaN	NaN	NaN	107.00000	2.000000	155.000000	36.000000	109.000000	11.000000
max	NaN	NaN	NaN	478.00000	12.000000	749.000000	173.000000	587.000000	84.000000

In [30]:
```
data.describe(include='object')
```

Out[30]:

	시도	시군구	월
count	2745	2745	2745
unique	17	206	12
top	경기	동구	07월
freq	372	72	229

In [31]:
```
data.sample(5)
```

Out[31]:

	시도	시군구	월	발생건수	사망자수	부상자수	중상	경상	부상신고
1295	충남	태안군	12월	15	0	27	11	16	0
2087	경북	경산시	02월	65	5	78	19	59	0
1628	전남	담양군	09월	28	2	44	16	28	0
893	강원	동해시	06월	34	0	51	6	45	0
1793	전남	완도군	06월	14	0	16	1	15	0

In [32]:
```
data.head()
```

Out[32]:

	시도	시군구	월	발생건수	사망자수	부상자수	중상	경상	부상신고
0	서울	종로구	01월	103	0	133	23	90	20

	시도	시군구	월	발생건수	사망자수	부상자수	중상	경상	부상신고
1	서울	종로구	02월	84	1	111	20	77	14
2	서울	종로구	03월	101	0	149	34	91	24
3	서울	종로구	04월	93	0	114	27	78	9
4	서울	종로구	05월	102	1	143	24	96	23

In [33]:
```
data.tail()
```

Out[33]:

	시도	시군구	월	발생건수	사망자수	부상자수	중상	경상	부상신고
2740	세종	세종	08월	52	1	85	26	58	1
2741	세종	세종	09월	50	4	61	14	46	1
2742	세종	세종	10월	65	0	90	22	66	2
2743	세종	세종	11월	65	0	91	17	71	3
2744	세종	세종	12월	57	0	83	13	69	1

원하는 dtype만 가져오기

원하는 타입만 가져오고 싶을 때, `select_dtypes`로 불러올 수 있다.

In [34]:
```
data.select_dtypes('object')[:5]
```

Out[34]:

	시도	시군구	월
0	서울	종로구	01월
1	서울	종로구	02월
2	서울	종로구	03월
3	서울	종로구	04월
4	서울	종로구	05월

tips 데이터도 마찬가지.

In [35]:
```
tips.select_dtypes('category')[:5]
```

Out[35]:

	sex	smoker	day	time
0	Female	No	Sun	Dinner
1	Male	No	Sun	Dinner
2	Male	No	Sun	Dinner
3	Male	No	Sun	Dinner
4	Female	No	Sun	Dinner

시·도별 평균 데이터를 알고싶을 때는 다음과 같이 그룹화해본다. 그룹화 할 때는 `groupby`를 이용하는데, 설명은 밑에서 다루고자 한다.

In [36]:
```
data.groupby('시도').mean()[:10]
```

Out[36]:

시도	발생건수	사망자수	부상자수	중상	경상	부상신고
강원	34.712963	0.912037	55.194444	12.310185	40.129630	2.754630
경기	143.677419	1.822581	218.209677	45.239247	154.008065	18.962366
경남	53.208333	1.541667	76.175926	26.912037	45.819444	3.444444
경북	50.970803	1.492701	76.346715	21.470803	51.310219	3.565693
광주	124.316667	1.266667	199.616667	24.733333	169.083333	5.800000
대구	137.489583	1.250000	201.375000	43.468750	141.812500	16.093750
대전	125.900000	1.550000	190.533333	33.950000	148.500000	8.083333
부산	62.171875	0.640625	85.770833	23.687500	55.140625	6.942708
서울	129.316667	1.013333	179.170000	37.940000	125.750000	15.480000
세종	66.250000	1.666667	95.166667	24.000000	68.666667	2.500000

다시 tips 데이터로..

indexing

`DataFrame`에서 인덱싱을 하는 방법은 여러 가지가 있다. 그 중 `iloc`를 사용해서 인덱싱하는 방법이 있다.

In [37]:
```
tips.iloc[:5]
```

Out[37]:

	total_bill	tip	sex	smoker	day	time	size

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

행과 열을 범위로 가져오고 싶을 때, `NumPy`의 `ix_` 처럼 사용하는 것도 가능하다.

In [38]:
```
tips.iloc[:5, :2]
```

Out[38]:

	total_bill	tip
0	16.99	1.01
1	10.34	1.66
2	21.01	3.50
3	23.68	3.31
4	24.59	3.61

인덱싱을 통해서 데이터를 바꿀 수 있다.

In [39]:
```
tips.head(1)
```

Out[39]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2

In [40]:
```
tips.iloc[0, 0]
```

Out[40]:
```
16.99
```

`tips` 데이터의 가장 첫번째 행의 `total_bill`은 `16.99`이다. 이를 `18.99`로 바꿔본다.

In [41]:
```
tips.iloc[0, 0] = 18.99
```

확인해보면 값이 바뀐 것을 알 수 있다.

In [42]:
```
tips.iloc[0, 0]
```

Out[42]:
```
18.99
```

In [43]:
```
tips.head(1)
```

Out[43]:

	total_bill	tip	sex	smoker	day	time	size
0	18.99	1.01	Female	No	Sun	Dinner	2

groupby

각 데이터를 그룹화시켜 분석하기 위해 사용한다.

성별과 팁에 대한 상관 관계를 알아보자.

In [44]:
```
x = tips.groupby('sex').mean()
```

In [45]:
```
x
```

Out[45]:

sex	total_bill	tip	size
Male	20.744076	3.089618	2.630573
Female	18.079885	2.833448	2.459770

각 데이터를 `Series` 타입으로 불러본다.

In [46]:
```
x.tip
```

Out[46]:
```
sex
Male      3.089618
Female    2.833448
Name: tip, dtype: float64
```

In [47]:
```
x.total_bill
```

Out[47]:
```
sex
Male      20.744076
Female    18.079885
Name: total_bill, dtype: float64
```

그룹에서 `tip`과 `total bill`의 비율을 가지는 새로운 행 `ratio`를 추가한 후 확인해본다.

In [48]:
```
x['ratio'] = x.tip / x.total_bill
```

In [48]:
```
x['ratio'] = x.tip / x.total_bill
```

In [49]:
```
x
```

Out[49]:

sex	total_bill	tip	size	ratio
Male	20.744076	3.089618	2.630573	0.148940
Female	18.079885	2.833448	2.459770	0.156718

여러 데이터를 한번에 그룹화할 때는 리스트안에 컬럼명을 넣어준다.

In [50]:
```
groups = tips.groupby(['tip', 'sex'])
```

In [51]:
```
groups.mean()[:5]
```

Out[51]:

tip	sex	total_bill	size
1.00	Male	12.600000	2.000000
	Female	5.356667	1.333333
1.01	Male	NaN	NaN
	Female	18.990000	2.000000
1.10	Male	NaN	NaN

In [52]:
```
groups = tips.groupby(['sex', 'day'])
```

In [53]:
```
groups.mean()[:10]
```

Out[53]:

sex	day	total_bill	tip	size
Male	Thur	18.714667	2.980333	2.433333
	Fri	19.857000	2.693000	2.100000
	Sat	20.802542	3.083898	2.644068
	Sun	21.887241	3.220345	2.810345
Female	Thur	16.715312	2.575625	2.468750
	Fri	14.145556	2.781111	2.111111
	Sat	19.680357	2.801786	2.250000
	Sun	19.983333	3.367222	2.944444

Fancy indexing

판다스는 NumPy 기반으로 만들어졌기 때문에 `팬시 인덱싱(Fancy indexing)` 이 가능하다.

In [54]:
```
groups[['tip']].mean()
```

Out[54]:

sex	day	tip
Male	Thur	2.980333
	Fri	2.693000
	Sat	3.083898
	Sun	3.220345
Female	Thur	2.575625
	Fri	2.781111
	Sat	2.801786
	Sun	3.367222

In [55]:
```
tips.iloc[:10][['tip', 'total_bill']]
```

Out[55]:

	tip	total_bill
0	1.01	18.99
1	1.66	10.34
2	3.50	21.01
3	3.31	23.68
4	3.61	24.59
5	4.71	25.29
6	2.00	8.77
7	3.12	26.88
8	1.96	15.04
9	3.23	14.78

데이터를 저장할 때...

저장하고 싶은 형식에 맞춰 `to_` 가 붙은 메소드를 사용한다.

In [56]:
```
[method for method in dir(pd.DataFrame) if 'to_' in method]
```

Out[56]:
```
['_to_dict_of_blocks',
 'to_clipboard',
 'to_csv',
 'to_dense',
 'to_dict',
 'to_excel',
 'to_feather',
 'to_gbq',
 'to_hdf',
 'to_html',
 'to_json',
 'to_latex',
 'to_msgpack',
 'to_numpy',
 'to_panel',
 'to_parquet',
 'to_period',
 'to_pickle',
 'to_records',
 'to_sparse',
 'to_sql',
 'to_stata',
 'to_string',
 'to_timestamp',
 'to_xarray']
```

In [57]:
```
tips.to_csv('tips.csv')
```

In [58]:
```
pd.read_csv('tips.csv').iloc[:10, 1:]
```

Out[58]:

	total_bill	tip	sex	smoker	day	time	size
0	18.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
5	25.29	4.71	Male	No	Sun	Dinner	4
6	8.77	2.00	Male	No	Sun	Dinner	2
7	26.88	3.12	Male	No	Sun	Dinner	4
8	15.04	1.96	Male	No	Sun	Dinner	2
9	14.78	3.23	Male	No	Sun	Dinner	2

Visualization

불러온 데이터를 알아보기 쉽게 시각화해본다.

시각화를 해주기 위해 먼저 `matplotlib` 을 임포트 해야한다. `matplotlib` 은 데이터 시각화를 위한 패키지이다.

In [59]:
```
import matplotlib.pyplot as plt
```

몽키 패칭을 통해 판다스 데이터 프레임의 데이터를 시각화할 수 있도록 해준다.

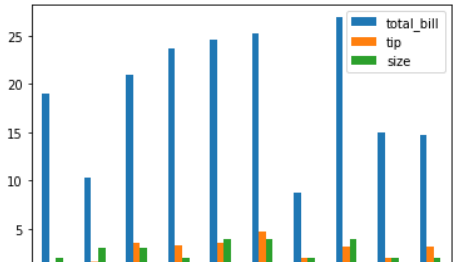
다음과 같이 바 그래프를 그려본다.

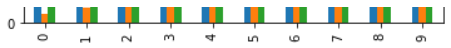
In [60]:
```
%matplotlib inline
```

만약 브라우저 상에서 그래프가 보이지 않을 경우, 위의 키워드를 입력해준다.

In [61]:
```
tips.iloc[:10].plot.bar()
```

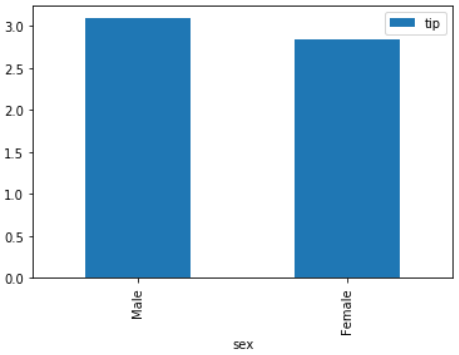
Out[61]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc350bc9e80>
```





In [62]:
```
tips.groupby('sex').mean()[['tip']].plot.bar()
```

Out[62]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc350b1c2b0>
```



미세먼지 데이터로 분석해보기

공공데이터포탈에서 미세먼지 데이터를 받은 후 분석을 시도해본다.

In [63]:
```
dust = pd.read_csv('data/dust.csv', engine='python', encoding='cp949')
```

In [64]:
```
dust.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 250 entries, 0 to 249
Data columns (total 3 columns):
날짜                  250 non-null object
 미세먼지농도 (㎍/㎥)     250 non-null int64
결과                  250 non-null object
dtypes: int64(1), object(2)
memory usage: 5.9+ KB
```

위에서 확인해보면 `날짜`에 해당하는 데이터 타입이 `object`라고 되어있다.

In [65]:
```
dust['날짜'][:10]
```

Out[65]:
```
0    2017-01-10
1    2017-01-11
2    2017-01-13
3    2017-01-14
4    2017-01-20
5    2017-01-21
6    2017-01-27
7    2017-01-28
8    2017-01-31
9    2017-02-01
Name: 날짜, dtype: object
```

날짜 데이터는 시간에 관한 데이터이므로, `시계열` 형식으로 바꿔주어야 한다.

아래와 같이 `to_datetime`으로 시계열로 바꾸어준다.

In [66]:
```
dust['date'] = pd.to_datetime(dust['날짜'])
```

`date`라는 컬럼을 새로 추가한 후, 시계열 데이터를 만들었다.

In [67]:
```
dust.columns
```

Out[67]:
```
Index(['날짜', ' 미세먼지농도 (㎍/㎥)', '결과', 'date'], dtype='object')
```

In [68]:
```
dust.head()
```

Out[68]:

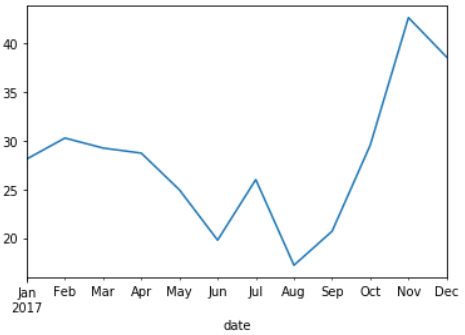
	날짜	미세먼지농도 (㎍/㎥)	결과	date
0	2017-01-10	19	좋음	2017-01-10
1	2017-01-11	20	좋음	2017-01-11
2	2017-01-13	29	좋음	2017-01-13
3	2017-01-14	40	보통	2017-01-14
4	2017-01-20	20	좋음	2017-01-20

미세먼지 평균 농도로 월·일별 그래프를 그려본다.

In [69]:
```
month = dust.set_index('date')[dust.columns[1]].resample('M')
```

In [70]:
```
month.mean().plot.line()
```

Out[70]: <matplotlib.axes._subplots.AxesSubplot at 0x7fc350cf00b8>



In [71]:
```
day = dust.set_index('date')[dust.columns[1]].resample('D')
```

In [72]:
```
day.mean().plot.line()
```

Out[72]: <matplotlib.axes._subplots.AxesSubplot at 0x7fc34ea232b0>