데이터처리

데이터 프레임 변경

- **lambda()**
  - 익명 함수로 간단하게 함수를 만들어서 사용할 때 사용
- **map()**
  - sequence형 데이터를 argument로 받아서 각 원소에 입력 받은 함수를 적용시키고 list로 반환
  - 어떤 값을 치환할 때 유용하게 사용
  - Series의 내장 메소드로 저장
  - Map은 각 Series 데이터에 적용
- **replace()**
  - map 함수의 기능 중에서 데이터를 변환하는 기능만 담당해 주는 특화된 메소드
- **apply()**
  - 전체 column에 해당 함수를 적용
  - Series, DataFrame모두 적용

데이터 프레임 변경

- **map() : Series에만 적용**

```
mapdata2 = data['지역'].map(lambda x : x + ':' + str(len(x)))
```

```
mapdata2
```

```
0          경상남도 거제시 :9
1          경상남도 거제시 :9
2          경상남도 거제시 :9
3          경상남도 거제시 :9
4          경상남도 거제시 :9
```

```
area = {'부산광역시':'busan'}
mapdata = data['지역'].map(area)
mapdata
```

```
0        NaN
1        NaN
2        NaN
3        NaN
4        NaN
5      busan
6      busan
7      busan
```

```
mapdata11 = data['지역'].replace(area)
```

```
mapdata11
```

```
0          경상남도 거제시
1          경상남도 거제시
2          경상남도 거제시
3          경상남도 거제시
4          경상남도 거제시
5              busan
6              busan
7              busan
```

데이터 프레임 변경

```
spdata = data['지역'].str.split(' ')
```

```
spdata
```

```
0          [경상남도, 거제시, ]
1          [경상남도, 거제시, ]
2          [경상남도, 거제시, ]
3          [경상남도, 거제시, ]
4          [경상남도, 거제시, ]
5               [부산광역시]
6               [부산광역시]
7               [부산광역시]
```

```
spdata = spdata.apply(lambda x : pd.Series(x))
```

```
spdata
```

	0	1	2	3
0	경상남도	거제시		NaN
1	경상남도	거제시		NaN
2	경상남도	거제시		NaN
3	경상남도	거제시		NaN
4	경상남도	거제시		NaN
5	부산광역시	NaN	NaN	NaN
6	부산광역시	NaN	NaN	NaN
7	부산광역시	NaN	NaN	NaN

1. 지역열의 문자열을 분리한 Series 생성
2. lambda함수 :
   생성된 Series의 각행에 있는 리스트를 Series로 만듬
3. apply함수:
   Series로 만들어진 것을 DataFrame으로 생성

결측 데이터 확인

- **.isnull() , .isna()**
  - 결측 데이터 확인
- **.isnull().sum()**
  - 결측 데이터 개수 확인

```
spdata.isnull()
```

	시	구
0	False	False
1	False	False
2	False	False
3	False	False
4	False	False
5	False	True
6	False	True
7	False	True

```
spdata.isnull().sum()
```

```
시      0
구     19
dtype: int64
```

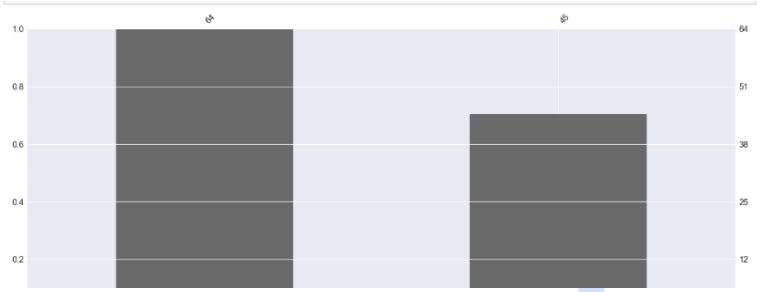
결측데이터 시각화

- missingno
  - pip install missingno

```
import missingno as msno

msno.matrix(spdata)
plt.show()
```

```
msno.bar(spdata)
plt.show()
```



결측 데이터 삭제

- **dropna()명령**
  - **thresh** 인수
    - 특정 갯수 이상의 데이터가 있는 행(열)삭제
  - **axis** 인자를 1
    - 누락데이터가 있는 열을 제거

```
spdata4 = spdata2.dropna(thresh = 44,axis=1)
```

```
spdata2.isnull().sum()
```

```
0     0
1    19
2    44
3    63
dtype: int64
```

```
spdata4
```

	0	1
0	경상남도	거제시
1	경상남도	거제시
2	경상남도	거제시
3	경상남도	거제시
4	경상남도	거제시
5	부산광역시	NaN
6	부산광역시	NaN
7	부산광역시	NaN

결측 데이터 변경

- **fillna()명령**
  - 결측데이터 값 변경

```
spdata4 = spdata4.fillna('*')
```

```
spdata4
```

	시	구
0	경상남도	거제시
1	경상남도	거제시
2	경상남도	거제시
3	경상남도	거제시
4	경상남도	거제시
5	부산광역시	*
6	부산광역시	*
7	부산광역시	*

데이터 프레임 값 변경

```
spdata4['지역']= spdata4.apply(lambda x : x.시 if x.구=='*' or x.구=='' else x.구 ,  axis='columns')
```

```
spdata4
```

	시	구	지역
0	경상남도	거제시	거제시
1	경상남도	거제시	거제시
2	경상남도	거제시	거제시
3	경상남도	거제시	거제시
4	경상남도	거제시	거제시
5	부산광역시	*	부산광역시
6	부산광역시	*	부산광역시
7	부산광역시	*	부산광역시

데이터 프레임 그룹별 개수

```
spdata4g = spdata4.groupby('지역')['시'].count()
```

```
spdata4g = pd.DataFrame(spdata4g)
```

```
spdata4g.head()
```

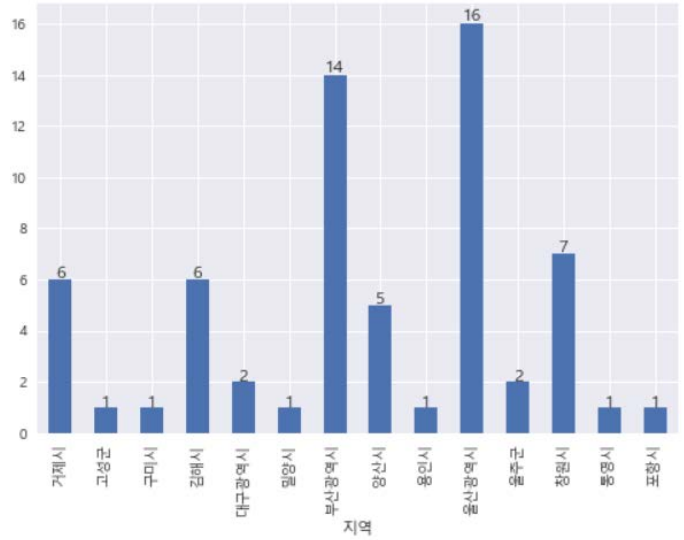
지역	시
거제시	6
고성군	1
구미시	1
김해시	6
대구광역시	2

판다스 그래프 값 표시

```
plt.rc('font', family='Malgun Gothic')
ax = spdata4g.plot(kind='bar', legend=False)

for p in ax.patches:
    left, bottom, width, height = p.get_bbox().bounds
    ax.annotate("%d"%(height), (left+width/2, height*1.01), ha='center')

plt.show()
```



- ax.patches
  ax가 가르키는 그래프에서, 막대들을 담고있는 리스트

- p.get_bbox().bounds
  해당 막대그래프의 정보
  왼쪽, 아래, 막대그래프의 폭, 높이에 대한 정보(튜플)

- ax.annotate(s, xy, *args, **kwargs)
  그래프 안에 특정 위치에 문자열 표시
  - s 인자에 문자열
  - xy 에는 튜플로 문자열이 들어갈 (x, y) 위치
  - ha horizontal align , center