pandas Pandas Basic Tutorial

 https://github.com/sohicode/cheatsheet/

SOHI CODE

Import

판다스 라이브러리를 활용

https://pandas.pydata.org/

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

Data Structure

판다스 자료구조

Series(index-value) 인덱싱된 1차원 자료구조

```
s = pd.Series(['apple', 'banana', 'cherry'],
              index=['a', 'b', 'c'])
```

DataFrame(index-column-value) 인덱싱된 2차원 자료구조

```
df = pd.DataFrame([['rex', 11, 140, 35],
                   ['dino', 9, 125, 30], ['mew', 13, 145, 40],
                   ['pengha', 8, 115, 20], ['kiki', 8, 130, 35]],
                  index = ['411', '211', '611', '111', '112'],
                  columns = ['name', 'age', 'height', 'weight'])
```

File

데이터 가져오기

```
df = pd.read_csv('file.csv')
df = pd.read_excel('file.xlsx', 'Sheet1')
```

데이터 저장하기

```
df.to_csv('result_file.csv')
df.to_excel('result_file.xlsx')
```

Summarize Data

데이터 정보 확인

```
df.shape       # (행, 열) 크기 확인
df.head()      # 첫 5행 출력(기본 5행)
df.tail(3)     # 마지막 3행 출력
df.info()      # 데이터 정보 확인
df.describe()  # 통계 요약 정보 확인 (NaN 제외)
df.index       # 인덱스 확인
df.columns     # 컬럼 확인
df.values      # 값 확인
df.dtypes      # 자료형 확인
df.nunique()   # 유일값 개수 확인
```

Sort

정렬

```
df.sort_index()                          # index 기준으로 오름차순 정렬
df.sort_values(by='height')              # 열 기준으로 오름차순 정렬
df.sort_values('weight')                 # 열 기준으로 오름차순 정렬
df.sort_values('weight', ascending=False) # 내림차순 정렬
```

Rank

순위

```
df.rank()                    # 순위 확인
df.rank(method='min')        # 동률일 때 최소값으로 순위 적용
df[df['height'].rank()==1.0] # 순위 1등 데이터 확인
```

Plotting

그래프

```
df['age'].plot.hist()        # index 기준으로 오름차순 정렬
df.plot.scatter(x=2, y=3)    # col1 열 기준으로 오름차순 정렬
```

Selection index

iloc 행 선택

```
df.iloc[0]       # 첫 번째 행 선택
df.iloc[1]       # 두 번째 행 선택
df.iloc[-1]      # 마지막 행 선택
df.iloc[[0, 2, 4]] # index가 0, 2, 4인 행 선택
df.iloc[0:2]     # index가 0인 행부터 2가 되기 전까지 슬라이싱
```

iloc 열 선택

```
df.iloc[:, 0]    # 첫 번째 열 선택
df.iloc[:, -1]   # 마지막 열 선택
df.iloc[0:3, 0:2] # index가 0~2, column이 0~1인 데이터 선택
```

Selection label

loc 행 선택

```
df.loc['611']      # index가 611인 행 선택
df.loc['411':'611'] # index가 411~611인 행 선택(611도 포함)
```

loc 열 선택

```
df.loc[:, 'name']          # coloum이 name인 열 선택
df.loc[:, ['name', 'age']] # coloum이 name, age인 열 선택
df.loc[df['age'] > 10 :, ['name', 'age']] # 나이가 10보다 큰 행
```

Condition

하나의 열 선택

```
df['name']   # 한개 열 선택
df.name      # 한개 열 선택
```

팬시 인덱싱으로 여러 열 선택

```
df[['name', 'age']]   # 여러개 열 선택
```

불리언 인덱싱으로 조건에 맞는 행 선택

```
df[df['height'] < 140]   # 키가 140보다 작은 데이터
df[(df['height'] > 120) & (df['height'] < 140)]
                         # 키가 120보다 크고 140보다 작은 데이터
                         # 여러조건인 경우 ()와 함께 &, |, ~ 사용
df[df['age'] == 8]       # 나이의 값이 8인 데이터
df[~(df['age'] == 8)]    # 나이의 값이 8이 아닌 데이터
```

Aggregate Function

집계 연산

```
df.sum()          # 합계 확인
df.count()        # 개수 확인
df.min()          # 최소값 확인
df['age'].idxmin() # 최소인덱스
df.std()          # 표준편차 확인
df.mean()         # 평균 확인
df.median()       # 중앙값 확인
df.max()          # 최대값 확인
df.age.idxmax()   # 최대인덱스
df.var()          # 분산 확인
df.value_counts()  # 유일값 개수 확인
df['age'].value_counts()  # 유일값 개수 확인
```

Missing Data

빈 값 확인 Na(Not Available)

```
df.isna()   # 결측치면 True, df.isnull()
df.notna()  # 결측치가 아니면 True, df.notnull()
```

데이터 처리

```
df.dropna()                  # df가 빈값이면 삭제
df.dropna(subset=['age'])    # age열이 빈값이면 삭제
df.fillna(value)             # 빈값에 값(value) 채워넣기
df.drop_duplicates()         # 중복값 삭제
```

pandas Pandas Advanced Tutorial

 https://github.com/sohicode/cheatsheet/

SOHI CODE

자료형 확인 — Data Type

```
df.dtypes
```

자료형 변경

```
df['weight'].astype('float')  # float 로 변경함
```

그룹 연산 — Group by

```
gdf = pd.DataFrame({'class': [1, 1, 1, 2, 2, 2],
                    'name' : ['rex', 'dino', 'kong', 'sand', 'sky', 'kiki'],
                    'gender' : ['m', 'm', 'f', 'm', 'f', 'f'],
                    'score' : [99, 88, np.nan, 79, 60, 90]})

gdf.groupby('class').count()     # class그룹의 데이터 개수
gdf.groupby('class')['score'].mean() # class그룹의 데이터 평균
gdf.groupby(['class', 'gender']).count() # class, gender그룹
gdf.groupby(['class', 'gender'])['score'].mean()
gdf.groupby(['class', 'gender'])['score'].agg([sum, max, min])
  # class, gender그룹의 score열에 대한 sum, max, min 집계 결과 확인
```

문자열 연산 — Series str

```
like = pd.DataFrame([
   ['rex', 11, 'climbing', 'apple', '$100', '2012.05.24.'],
   ['dino', 9, 'drawing', 'korean pear', '$200', '2014.04.18.'],
   ['mew', 13, 'traveling', 'blueberry', '$150', '2010.07.07.']],
   index = [1, 2, 3],
   columns = ['name', 'age', 'hobby', 'fruit', 'money', 'birthday'])

like['fruit'].str.replace('korean pear', 'pear') # 문자열 변경
like['birthday'].str.split('.', expand=True)[0] # 문자열 분할
like['money'].str.strip('$')   # '$'를 제거함
like['name'].str.upper()       # 대문자로 변환함
like['name'].str.islower()     # 소문자면 True
like['hobby'].str.len()        # 문자열 길이 확인
like['hobby'].str.findall('ing')  # 'ing'를 찾으면 True
```

데이터 연결 — concat

```
s1 = pd.Series(['a', 'b'], index=[1, 2])
s2 = pd.Series(['c', 'd'], index=[3, 4])
pd.concat([s1, s2])            # 행 방향으로 연결

df1 = pd.DataFrame({'a': ['a0', 'a1'], 'b': ['b0', 'b1']})
df2 = pd.DataFrame({'a': ['a2', 'a2'], 'b': ['b2', 'b3']},
                   index=[2, 3])
df3 = pd.DataFrame({'c': ['c1', 'c2'], 'd': ['d1', 'd2']},
                   index=[1, 2])
df4 = pd.concat([df1, df2])
df5 = pd.concat([df1, df3], axis=1)   # 열 방향으로 연결
```

df1

	a	b
0	a0	b0
1	a1	b1

df2

	a	b
2	a2	b2
3	a2	b3

df3

	c	d
1	c1	d1
2	c2	d2

df4

	a	b
0	a0	b0
1	a1	b1
2	a2	b2
3	a2	b3

df5

	a	b	c	d
0	a0	b0	NaN	NaN
1	a1	b1	c1	d1
2	NaN	NaN	c2	d2

데이터 합치기 — merge

```
shop =  pd.DataFrame({'shop_id' : ['s01', 's02', 's03', 's04'],
      'city' : ['Chennai', 'Madurai', 'Trichy', 'Coimbatore'],
      'zip_code' : [600001, 625001, 620001, 641001] } )
product = pd.DataFrame(
       {'shop_id' : ['s01', 's02', 's02', 's03', 's03', 's03', 's05'],
       'product_id' : ['p01', 'p02', 'p03', 'p01', 'p02', 'p03', 'p02'],
       'price' : [220, 500, 145, 225, 510, 150, 505] } )

pd.merge(shop, product, on='shop_id', how='inner')
  # shop_id를 기준으로 두DataFrame의 shop_id가 같은 행을 합침
pd.merge(shop, product, on='shop_id', how='left')
pd.merge(shop, product, on='shop_id', how='right')
pd.merge(shop, product, on='shop_id', how='outer')
```

결측 데이터 처리 — dropna, fillna

```
s = pd.Series([1, 2, np.nan, 'String', None])

s.isna()                 # 빈 값이면 True
s[s.notna()]             # 빈 값이 아닌 데이터 목록
s.dropna()               # 빈 값이면 삭제함
s.fillna(0)              # 빈 값이면 0을 채움
s.fillna(method='ffill')  # 빈 값이면 앞의 데이터를 채움
s.fillna(method='bfill')  # 빈 값이면 뒤의 데이터를 채움
```

값 치환 — replace

```
s = pd.Series([1., 2., -999., 3., -1000., 4.])

s.replace(-999, np.nan)  # -999 는 nan 값으로 변경함
s.replace([-999, -1000], np.nan)  #-999,-1000을nan값으로변경함
```

중복 제거 — drop duplicates

```
s = pd.Series([1, 2, 1, 2, 3, np.nan, 'String', None])

s.unique()               # 유일값 목록 확인
s.duplicated()           # 중복값이면 True, 아니면 False
s.drop_duplicates()      # 중복값이면 1개만 남기고 삭제함
```

행 삭제 — drop rows/columns

```
df.drop(['111'])  # 111 행을 삭제함
df.drop(['111'], inplace=True)   # 행 삭제한 것을 df에 적용함
```

열 삭제

```
df.drop('weight', axis=1)       # weight 컬럼을 삭제함
df.drop(columns=['weight'])   # weight 컬럼을 삭제함
df.drop(columns=['height', 'weight'])  # 여러 컬럼을 삭제함
```

인덱스, 컬럼명 변경 — rename index/columns

```
df.reset_index()          # 인덱스를 0부터 재설정
df.set_index('name')    # 컬럼을 인덱스로 설정함
df.rename(index = {'411': '4학년1반1번'}) # 인덱스명 변경
df.rename(columns = {'name':'이름', 'age':'나이'}) # 컬럼명 변경
```