Pandas

- PANel DAta
- 구조화된 데이터의 처리를 지원하는 python 라이브러리
- numpy와 통합해 강력한 spreadsheet 처리 기능 제공
- 인덱싱, 연산용 함수, 전처리 함수, 데이터 처리 및 통계 분석에 용이
- tabular data
  - attribute(field, feature, column)
  - instance(tuple, row)
  - data(value)
- Series
  - DataFrame 중 하나의 column에 해당하는 데이터의 모음
  - column vector를 표현하는 object
  - subclass of numpy.ndarray
  - index, data, dtype
- DataFrame
  - numpy array-like
  - each column can have a different type
  - row and col index
  - series를 모아서 만든 data table
- loc, iloc
  - loc: locate index (name)
  - iloc: locate index number(0, 1, 2, ...)
- selection with column names

```
df[['account', 'street', 'state']].head().T
```

- map

```
def change_sex(x):
return 0 if x == "male" else 1

df.sex.map(change_sex)
```

- useful functions
  - unique()
  - describe(): (숫자형만) 통계치 한번에 출력
  - sort_values()
  - corr(): correlation
  - value_counts()
- Groupby
  - SQL groupby 명령어와 동일

- split - apply - combine 과정을 거쳐 연산

```
# groupby(묶음의 기준 컬럼)[적용받는 컬럼].적용받는 연산
df.groupby("Team")["Points"].sum()
```

- apply 유형
  - aggregation: 요약된 통계정보 추출 ex. sum, mean
  - transformation: 해당 정보 변환 ex. lambda
  - filtration: 특정 정보 제거 (필터링)

```
grouped.agg(sum)
grouped.agg(np.mean)

# normalize
score = lambda x: (x - x.mean()) / x.std()
grouped.transform(socore)

df.groupby("Team").filter(lambda x: x["Points"].max() > 700)
```

- Pivot table
  - index 축은 groupby와 동일
  - column에 추가로 labeling 값 추가
  - value에 numeric type 값을 aggregation

```
# 값: duration
df.pivot_table([duration],
               index=[df.month, df.item],
               columns=df.network,
               aggfunc="sum",
               fill_value=0)
```

- merge & concat

```
pd.merge(df_a, df_b, on='subject_id')
# pd.merge(df_a, df_b, left_on='subject_id', right_on='id', how='right')
# join: inner, left, right, full
pd.concat([df_a, df_b])
```

- persistence
  - db 연결 conn을 사용해 dataframe 생성

```
import sqlite3

conn = sqlite3.connet('./data.db')
cur = conn.cursor()
cur.execute('select * from airlines limit 5;')
res = cur.fetchall()
# tuple 형태로 나옴
df_airlines = pd.read_sql_query('select * from airlines;', conn)
```

- XLS
  - openpyxls, XlsxWrite

```
writer = pd.ExcelWriter('./df_routes.xlsx', engine='xlswriter')
df_routes.to_excel(writer, sheet_name='Sheet1')
df_routes.to_pickle('./df_routes.pickle')
```