|  |  |
| --- | --- |
| 교육 제목 | **데이터 프레임의 다양한 활용** |
| 교육 일시 | 20211111 09:00 ~ 18:00 |
| 교육 장소 | 자택 온라인 수업 |
| **교육 내용** | |
| 오전 | # 개별 원소에 함수 매핑 : 시리즈 객체.apply(함수명)  def add\_10(n):  return n+10  sr1 = df['age'].apply(add\_10)  sr1.head  # sr1 = df['age'].apply(add\_10)  sr1 = df['age'].apply(lambda x: x+10)  sr1.head  sr2  sr2 = df['age'].apply(add\_two\_10, b=20)  ㅇef max\_min(x):  return x.max() -x.min()  result = df.apply(lambda x: x.max() - xmin(), axis = 1)  print(result)  df['sum'] = df.apply(lambda x:x.sum(), axis = 1)  ad.head()  # 열 분리 추가 : stock.xlsx  # 데이터 프레임의 다양한 응용  mask1 = (titanic.age >= 10) & (titanic.age <20)  # df\_teenage = titanic.loc[mask1, :]  df\_teenage = titanic.loc[(titanic.age >= 10) & (titanic.age < 20), :]  df\_teenage  # df\_female -> 10대 미만이고 여상만 필터링  df\_female = titanic.loc[(titanic.age<10) & (titanic.sex == "female"), :]  df\_female  # df\_1 : 10대 미만이거나 60대 이상인 데이터의 'age', 'sex', 'class' 만 검색  mask2 = (titanic.age < 10) | (titanic.age >= 60)  df\_1 = titanic.loc[mask2, ['age', 'sex', 'class']]  df\_1  pdf  pdf1 = pd.pivot\_table(df,  index=['class', 'sex'],  column = 'survived',  values = ['age', 'fare'],  aggfunc = ['min', 'max'])  pdf1  pdf1.xs(1, level = 'survived', axis=1) |
| 오후 | pdf  pdf1 = pd.pivot\_table(df,  index=['class', 'sex'],  column = 'survived',  values = ['age', 'fare'],  aggfunc = ['min', 'max'])  pdf1  pdf1.xs(1, level = 'survived', axis=1) |