

Chapter. 02

데이터 크롤링

I 데이터 전처리

FASTCAMPUS

ONLINE

금융공학/퀀트 I

강사. 서찬웅

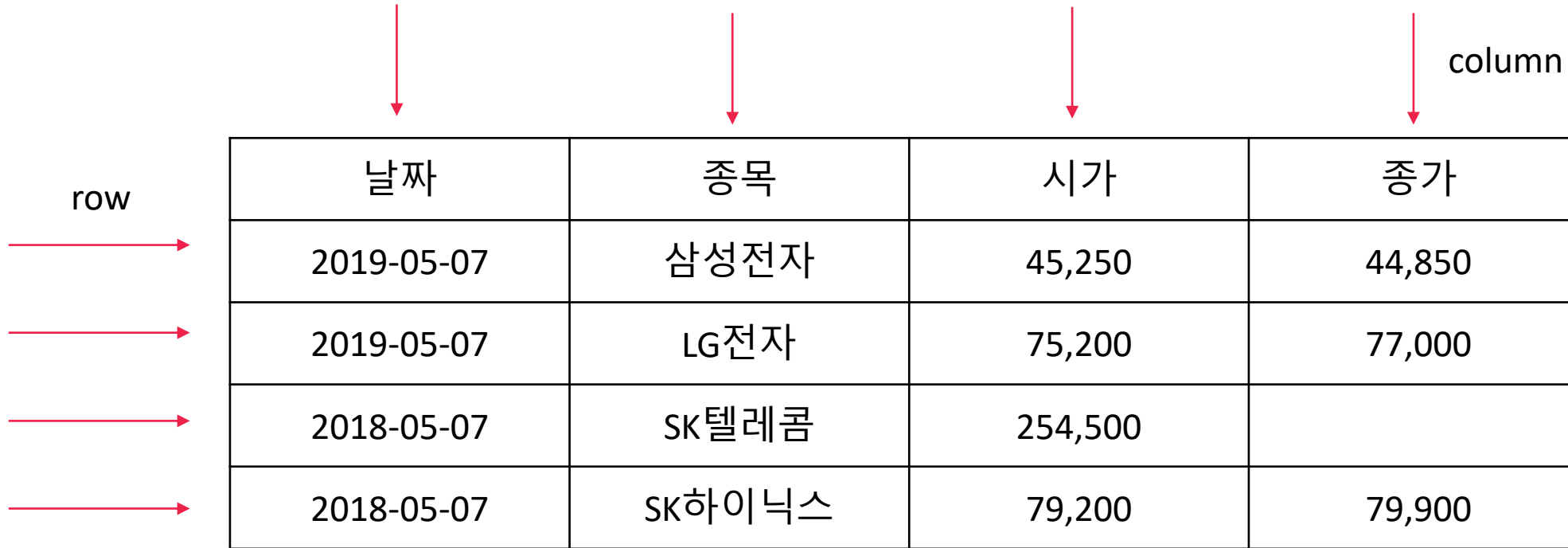
I 이번 시간에 배울 내용

강의내용

1. pandas 데이터 타입 변환과 전처리
2. 문자열과 수치 타입간 변환
3. 날짜 다루기

I DataFrame에 대해서 조금 더 자세히 알아보겠습니다.

- 지난 고급주제에서 DataFrame을 언급했습니다. 앞으로 최종적으로 분석을 진행할 데이터 형식은 DataFrame입니다. 이번 시간에는 DataFrame의 구조에 대해서 간단히 공부하겠습니다.
- DataFrame은 아래 그림처럼 생긴 데이터 집합입니다.
 - row : 4, column : 4개가 모여 있는 데이터 입니다.



The diagram shows a table with 4 rows and 4 columns. Red arrows point to the labels 'row' and 'column' on the left and top of the table, respectively. The table contains the following data:

날짜	종목	시가	종가
2019-05-07	삼성전자	45,250	44,850
2019-05-07	LG전자	75,200	77,000
2018-05-07	SK텔레콤	254,500	
2018-05-07	SK하이닉스	79,200	79,900

I DataFrame를 생성하겠습니다.

- 아래 컬럼들의 데이터는 눈으로 보기엔 날짜 및 숫자처럼 보일 수 있지만 사실 아래 데이터들은 문자열 데이터입니다. 우리가 숫자라고 생각하고 각각의 컬럼에 수치연산 함수를 적용하게 된다면 오류가 발생합니다.
- 이럴 때는 우리는 데이터 타입을 변경할 필요가 있습니다.
- 또한 데이터 양이 커지면 같은 int 형 데이터라도 int32, int64의 때문에 메모리에 차지하고 낭비되는 용량이 발생합니다. 이럴 경우에도 데이터 타입을 적당한 타입으로 변경해야 합니다.
- 아래 예제를 통해서 확인을 해보겠습니다.

```
stock_dict = [{'날짜': '2019-05-07',
               '종목': '삼성전자',
               '시가': '45,250',
               '종가': '44,850'},
              {'날짜': '2019-05-07',
               '종목': 'LG전자',
               '시가': '75,200',
               '종가': '77,000'},
              {'날짜': '2019-05-07',
               '종목': 'SK텔레콤',
               '시가': '254,500'},
              {'날짜': '2019-05-07',
               '종목': 'SK하이닉스',
               '시가': '79,200',
               '종가': '79,900'}]
```

- DataFrame() 함수 안에 dict형태의 데이터를 전달하면 DataFrame 형식 데이터가 생성됩니다.
- 컬럼 순서가 눈에 보이는 순서가 아니기 때문에 columns 매개변수에 컬럼 순서를 지정하여 전달합니다.

```
data = pd.DataFrame(stock_dict, columns=['날짜', '종목', '시가', '종가'])
```

	날짜	종목	시가	종가
0	2019-05-07	삼성전자	45,250	44,850
1	2019-05-07	LG전자	75,200	77,000
2	2019-05-07	SK텔레콤	254,500	NaN
3	2019-05-07	SK하이닉스	79,200	79,900

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4 entries, 0 to 3
Data columns (total 4 columns):
날짜      4 non-null object
종목      4 non-null object
시가      4 non-null object
종가      3 non-null object
dtypes: object(4)
memory usage: 104.0+ bytes
```

I DataFrame의 결측치 처리에 대해서 알아보겠습니다.

- 앞의 DataFrame에서 NaN으로 표시되는 값은 데이터가 없다는 뜻으로 결측치를 의미합니다.
- 결측치가 있을 경우 DataFrame에서는 `dropna()` 메소드와 `fillna()` 메소드를 제공하여 처리할 수 있습니다.
 - `dropna()` : DataFrame에서 결측치가 포함된 row를 삭제(해당하는 row의 전체 값이 삭제가 됩니다.)
 - `fillna()` : 매개변수로 전달하는 값으로 NaN을 대체합니다. 특정 숫자로 채우고 싶다면 매개변수로 특정 숫자를 넣거나, 시계열 데이터의 경우 `method` 매개변수의 값으로 `ffill/bfill`를 전달하면 앞/뒤 값으로 NaN을 대체합니다. (앞뒤 데이터로 데이터를 채우는 것은 시계열 데이터일 때 권장합니다.)

```
data.dropna()
```

	날짜	종목	시가	종가
0	2019-05-07	삼성전자	45,250	44,850
1	2019-05-07	LG전자	75,200	77,000
3	2019-05-07	SK하이닉스	79,200	79,900

```
data.fillna(0)
```

	날짜	종목	시가	종가
0	2019-05-07	삼성전자	45,250	44,850
1	2019-05-07	LG전자	75,200	77,000
2	2019-05-07	SK텔레콤	254,500	0
3	2019-05-07	SK하이닉스	79,200	79,900

```
data.fillna(method='ffill')
```

	날짜	종목	시가	종가
0	2019-05-07	삼성전자	45,250	44,850
1	2019-05-07	LG전자	75,200	77,000
2	2019-05-07	SK텔레콤	254,500	77,000
3	2019-05-07	SK하이닉스	79,200	79,900

```
data.fillna(method='bfill')
```

	날짜	종목	시가	종가
0	2019-05-07	삼성전자	45,250	44,850
1	2019-05-07	LG전자	75,200	77,000
2	2019-05-07	SK텔레콤	254,500	79,900
3	2019-05-07	SK하이닉스	79,200	79,900

I DataFrame의 데이터형 변환에 대해서 알아보겠습니다.

- '시가', '종가'의 데이터는 문자열이기 때문에 int형으로 변경해야 쿼트에서 사용할 수 있습니다. 이제 각 컬럼의 형태를 변환하는 방법에 대해서 알아보겠습니다.
- apply() : 각각의 컬럼에 함수(사용자 함수포함) 를 적용할 수 있게 만들어주는 메소드입니다.

```
data2 = data.copy()
```

```
data2.dropna(inplace=True)
```

```
data2['시가'].apply(lambda x : int(x.replace(",","")))
```

```
0    45250
1    75200
3    79200
Name: 시가, dtype: int64
```

```
data2['시가'] = data2['시가'].apply(lambda x : int(x.replace(",","")))
```

```
data2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3 entries, 0 to 3
Data columns (total 4 columns):
날짜      3 non-null object
종목      3 non-null object
시가      3 non-null int64
종가      3 non-null object
dtypes: int64(1), object(3)
memory usage: 84.0+ bytes
```

한 컬럼에 적용

```
data3 = data.copy()
```

```
data3.dropna(inplace=True)
```

```
data3[['시가', '종가']] = data3[['시가', '종가']].applymap(lambda x : int(x.replace(",","")))
```

```
data3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3 entries, 0 to 3
Data columns (total 4 columns):
날짜      3 non-null object
종목      3 non-null object
시가      3 non-null int64
종가      3 non-null int64
dtypes: int64(2), object(2)
memory usage: 96.0+ bytes
```

여러 컬럼에 적용

DataFrame은 = 연산자를 사용하여 값을 할당하면 얇은 복사가 되기 때문에 복사된 값을 변경하면 원본도 같이 변경됩니다.
copy() 메소드를 사용하여 깊은 복사를 진행합니다.

I DataFrame에서 날짜 데이터로 변환을 해보겠습니다.

- 날짜 데이터를 포함하여 형 변환을 해주는 메소드에 대해서 알아 보겠습니다.
 - `astype()` : 매개변수로 지정한 데이터로 형변환을 해주는 메소드
 - `to_datetime()` : 문자열 시간 데이터를 `datetime` 객체로 형 변환 해주는 메소드
- 아래는 `to_datetime()` 메소드를 실행하여 날짜 객체로 변환하는 예제 및 int 64 데이터를 `astype()` 메소드를 사용하여 int32로 변경하는 예제입니다.

```
data3['날짜'] = pd.to_datetime(data3['날짜'], format='%Y-%m-%d %H:%M:%S')
```

```
data3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3 entries, 0 to 3
Data columns (total 4 columns):
날짜      3 non-null datetime64[ns]
종목      3 non-null object
시가      3 non-null int64
종가      3 non-null int64
dtypes: datetime64[ns](1), int64(2), object(1)
memory usage: 108.0+ bytes
```

```
data3['종가'] = data3['종가'].astype('int32')
```

```
data3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3 entries, 0 to 3
Data columns (total 4 columns):
날짜      3 non-null datetime64[ns]
종목      3 non-null object
시가      3 non-null int64
종가      3 non-null int32
dtypes: datetime64[ns](1), int32(1), int64(1), object(1)
memory usage: 96.0+ bytes
```

- 형 변환을 사용하여 메모리 사용량이 변화하는 것을 확인할 수 있습니다.

I 정리

- DataFrame 데이터 타입 변환 및 전처리, 결측치 처리
- 문자열을 수치형으로 변환하는 과정
- 날짜 데이터 변환하기
- int64형 데이터를 int32로 변경하여 메모리 사용량 줄이기

감사합니다