Python class *2*

Pandas

이번에는 데이터 구조를 구성하고 데이터 조작과 분석을 해보는 시간을 갖도록 하겠습니다. 이를 위해 필요한 작업으로 `pandas`를 `import`해야 합니다.

```
import pandas as pd
```

위 코드는 단순히 import 하는 것이 아니라 `pd`라는 변수로서 `pandas`를 import하여 사용하겠다는 의미입니다.

Series & DataFrame

Pandas에서는 데이터를 효과적으로 다루기 위해 Series와 DataFrame을 제공합니다. Pandas의 Series는 1차원 데이터를 다루는 데 효과적인 자료구조이며, DataFrame은 행과 열로 구성된 2차원 데이터를 다루는 데 효과적인 자료구조입니다.

Series

아래 코드와 같이 `Series`를 통해 1차원 데이터를 선언하여 사용할 수 있습니다.

```
purchase_1 = pd.Series({'Name':'Chris','Item':'DogFood','Cost':22.50})
```

```
purchase_2 = pd.Series({'Name':'Michel','Item':'CatMilk','Cost':15.75})
```

```
purchase_3 = pd.Series({'Name':'Tim','Item':'Bread','Cost':8.65})
```

DataFrame

그리고 아래와 같이 `DataFrame`을 사용하여 2차원 데이터를 선언하여 사용할 수 있게 됩니다.

```
df=pd.DataFrame([purchase_1,purchase_2,purchase_3],
   index=['Store1','Store2','Store3'])
print(df)
```

	Cost	Item	Name
Store1	22.50	DogFood	Chris
Store2	15.75	CatMilk	Michel
Store3	8.65	Bread	Tim

Pandas.DataFrame.head()

그리고 DataFrame에는 `head()` 라는 함수가 있는데 이 함수는 방대한 자료 중에 앞부분에 위치한 일부만을 보여주는 기능을 합니다. 아래에서는 DataFrame에 들어 있는 자료가 크지 않아서 그냥 출력한 것과 동일한 결과를 얻게 되었습니다.

```
df.head()
```

	Cost	Item	Name
Store1	22.50	DogFood	Chris
Store2	15.75	CatMilk	Michel
Store3	8.65	Bread	Tim

Pandas.DataFrame.loc()

DataFrame의 `loc()` 이라는 함수는 특정 location의 데이터들을 발췌하여 리턴해주는 기능을 합니다.

```
df.loc['Store2']
```

```
Cost      15.75
Item    CatMilk
Name     Michel
Name: Store2, dtype: object
```

```
df.loc['Store1']
```

```
Cost       22.5
Item    DogFood
Name      Chris
Name: Store1, dtype: object
```

```
df.loc['Store1','Cost']
```

```
22.5
```

Pandas.DataFrame.T

그리고 DataFrame의 편리한 기능 중 하나를 소개하겠습니다. 2차원 자료구조를 다루다 보면 index와 columns들을 서로 바꿔서 자료구조를 만들고 싶을 때가 간혹 있습니다. 이럴때 아주 쉽고 편하게 바꾸어 주는 함수가 있습니다. 바로 ... `pandas.DataFrame.T` 입니다. 사용법은 아래 코드와 같이 이미 선언되어 있는 DataFrame에 `.T` 만 붙여주면 끝! 입니다.

```
df.T
```

	Store1	Store2	Store3
Cost	22.5	15.75	8.65
Item	DogFood	CatMilk	Bread
Name	Chris	Michel	Tim

혼자해보기

이제 잠시 여태 배운 내용들을 가지고 2차원 자료구조를 분석해도록 하겠습니다.

```
df['Cost']
```

```
Store1    22.50
Store2    15.75
Store3     8.65
Name: Cost, dtype: float64
```

```
df.loc['Store1']['Cost']
```

```
22.5
```

```
df.loc[:,['Name','Cost']]
```

	Name	Cost
Store1	Chris	22.50
Store2	Michel	15.75
Store3	Tim	8.65

Pandas.DataFrame.drop()

이제는 자료구조 자체를 수정하는 방법들을 배워보도록하겠습니다.

자료구조를 다루다 보면 불필요한 인덱스들을 없애고 싶을 때가 있을 겁니다. 이때 사용하면 되는 함수가 drop('인덱스이름')입니다.

```
df.drop('Store2')
```

	Cost	Item	Name
Store1	22.50	DogFood	Chris
Store3	8.65	Bread	Tim

Pandas.DataFrame.copy()

DataFrame을 복사하고 싶을 때에는 `copy()` 라는 함수를 drop과 마찬가지로 사용하면 됩니다.

```
copy_df=df.copy()
print(copy_df)
```

	Cost	Item	Name
Store1	22.50	DogFood	Chris
Store2	15.75	CatMilk	Michel
Store3	8.65	Bread	Tim

컬럼 추가하기

DataFrame에 columns을 추가 하고 싶을 때는 아래와 같이 그냥 원래 있던것 처럼 선언하면 됩니다.

```
df['Location']=None
print(df)
```

	Cost	Item	Name	Location
Store1	22.50	DogFood	Chris	None
Store2	15.75	CatMilk	Michel	None

Store3	8.65	Bread	Tim	None

컬럼 삭제하기

방금 추가한 컬럼을 삭제하는 방법은 아래 코드와 같이 삭제할 컬럼을 명시하고 앞에 `del`을 선언해주면 됩니다.

```
del df['Location']
print(df)
```

	Cost	Item	Name
Store1	22.50	DogFood	Chris
Store2	15.75	CatMilk	Michel
Store3	8.65	Bread	Tim

컬럼,인덱스 단위로 데이터 수정하기

데이터를 수정할 때 개별로 수정할 때도 있지만 컬럼이나 인덱스 단위로 한번에 수정을 하고 싶을때가 있습니다. 그럴 때는 수정할 컬럼이나 인덱스를 선언한뒤 수정할 내용을 덧붙이기만 하면 됩니다. 아래 코드는 Integer 타입의 Cost라는 컬럼을 한번에 +2 해주는 코드입니다.

```
df['Cost']+2
```

```
Store1    24.50
Store2    17.75
Store3    10.65
Name: Cost, dtype: float64
```