IPA 주관 인공지능센터 기본(fundamental) 과정

- GitHub link: here
- E-Mail: windkyle7@gmail.com

이번에는 seaborn 모듈에서 제공하는 Tidy data 중 `tips` 데이터를 불러온다.

In [1]:
```
import seaborn as sns
tips = sns.load_dataset('tips')
```

불러온 데이터가 어떤 데이터인지 확인해본다.

In [2]:
```
tips.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 7 columns):
total_bill    244 non-null float64
tip           244 non-null float64
sex           244 non-null category
smoker        244 non-null category
day           244 non-null category
time          244 non-null category
size          244 non-null int64
dtypes: category(4), float64(2), int64(1)
memory usage: 7.2 KB
```

In [3]:
```
tips['time2'] = tips.time.astype('object')
```

`total_bill` 컬럼의 가장 앞의 5개 데이터는 다음과 같다.

In [4]:
```
tips['total_bill'].head()
```

Out[4]:
```
0    16.99
1    10.34
2    21.01
3    23.68
4    24.59
Name: total_bill, dtype: float64
```

이 데이터를 `bill` 이라는 식별자에 바인딩 하고 이 데이터를 예시로 설명을 해보고자 한다.

In [5]:
```
bill = tips[['total_bill']].head()
```

In [6]:
```
bill.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 1 columns):
total_bill    5 non-null float64
dtypes: float64(1)
memory usage: 120.0 bytes
```

위에서 살펴보면 `bill` 의 `dtype` 은 `float64` 라는 것을 알 수 있다.

In [7]:
```
bill.total_bill
```

Out[7]:
```
0    16.99
1    10.34
2    21.01
3    23.68
4    24.59
Name: total_bill, dtype: float64
```

만약 `16.99` 에 해당하는 데이터를 `0.5` 로 전부 바꾼다고 해보자.

In [8]:
```
[method for method in dir(bill.total_bill) if 'replace' in method or 'change' in method]
```

Out[8]:
```
['_maybe_cache_changed', 'pct_change', 'replace']
```

대충 살펴보니 `replace` 메소드가 보인다. 이 메소드를 사용하면 바뀌지 않을까?

In [9]:
```
bill.total_bill.replace('16.99', '0.5')
```

Out[9]:
```
0    16.99
1    10.34
2    21.01
3    23.68
4    24.59
Name: total_bill, dtype: float64
```

In [10]:
```
bill.total_bill.replace(16.99, 0.5)
```

Out[10]:
```
0     0.50
1    10.34
2    21.01
```

```
3    23.68
4    24.59
Name: total_bill, dtype: float64
```

값이 잘 바뀐것을 확인할 수 있다. 마찬가지로, 이번에는 `object` 타입을 바꿔보자.

In [11]:
```
times = tips[['time']].head()
times
```

Out[11]:

	time
0	Dinner
1	Dinner
2	Dinner
3	Dinner
4	Dinner

In [12]:
```
times.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 1 columns):
time    5 non-null category
dtypes: category(1)
memory usage: 181.0 bytes
```

`times.time` 타입은 `object`이다. 그럼 마찬가지로 `16.99`라는 값을 모두 `0.5`로 바꿔보자.

In [13]:
```
times['time'].replace('Dinner', 'Launch')
```

Out[13]:
```
0    Launch
1    Launch
2    Launch
3    Launch
4    Launch
Name: time, dtype: object
```

`Series.replace` vs `StringMethod.replace`

이는 `dtype`에 따라, `정규식(regex)`으로 문자열 데이터를 파싱하거나 매칭시켜 특정 값만을 뽑아내고 싶을 때 용례가 다르다.

자세한 사항은 여기를 참고하면 도움이 될 것이다.

In [14]:
```
time_str = times['time'].str
```

다음 `StringMethod.replace` 메소드의 설명을 한번 참고해보자.

In [15]:
```
?time_str.replace
```

```
    Replace occurrences of pattern/regex in the Series/Index with
    some other string. Equivalent to :meth:`str.replace` or
    :func:`re.sub`.
```

위 설명에도 나와있듯, 정규식을 사용한다는 차이점이 있다.

In [16]:
```
times['time'].str.replace('Dinner', 'Launch')
```

Out[16]:
```
0    Launch
1    Launch
2    Launch
3    Launch
4    Launch
Name: time, dtype: object
```

정규식 패턴을 사용해서 앞글자 `D`를 `K`로 바꿔본다.

In [17]:
```
time_str.replace('[D]', 'K')
```

Out[17]:
```
0    Kinner
1    Kinner
2    Kinner
3    Kinner
4    Kinner
Name: time, dtype: object
```

처음으로 돌아와서, `tips` 데이터를 다시 한번 살펴보자.

In [18]:
```
tips.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 8 columns):
total_bill    244 non-null float64
tip           244 non-null float64
sex           244 non-null category
```

```
smoker        244 non-null category
day           244 non-null category
time          244 non-null category
size          244 non-null int64
time2         244 non-null object
dtypes: category(4), float64(2), int64(1), object(1)
memory usage: 9.1+ KB
```

위에서 살펴보면 `day` 라는 컬럼이 있는 것을 확인했다.

In [19]:
```
tips.day.head()
```

Out[19]:
```
0    Sun
1    Sun
2    Sun
3    Sun
4    Sun
Name: day, dtype: category
Categories (4, object): [Thur, Fri, Sat, Sun]
```

이렇게 `category` 데이터로 되어있는데, 이 `day` 라는 데이터는 `시계열` 에 해당하는 데이터이다.

다음 장에서는 `시계열 (TimeSeries)` 이라는 것에 대해 다루게 될 것이다.