The Series Data Structure

In [1]:

```
import pandas as pd    #판다스를 pd라는 이름으로 불러옴
pd.Series?
```

In [2]:

```
animals = ['Tiger', 'Bear', 'Moose']
animals
```

Out[2]:

```
['Tiger', 'Bear', 'Moose']
```

In [3]:

```
pd.Series(animals)
```

Out[3]:

```
0    Tiger
1     Bear
2    Moose
dtype: object
```

In [4]:

```
numbers = [1, 2, 3]
pd.Series(numbers)
```

Out[4]:

```
0    1
1    2
2    3
dtype: int64
```

In [5]:

```
animals = ['Tiger', 'Bear', None]
pd.Series(animals)
```

Out[5]:

```
0    Tiger
1     Bear
2     None
dtype: object
```

In [6]:

```
numbers = [1, 2, None]
pd.Series(numbers)    #NaN = Not a Number
```

Out[6]:

```
0    1.0
1    2.0
2    NaN
dtype: float64
```

In [7]:

```
import numpy as np
```

In [8]:

```
np.nan == None    #nan이 None이냐 는 것은 false
```

Out[8]:

```
False
```

In [9]:

```
np.nan == np.nan    #같은 데이턴지 알 수 없기 때문에 비교불가
```

Out[9]:

```
False
```

In [10]:

```
np.isnan(np.nan)    #넘버냐 아니냐
```

Out[10]:

```
True
```

In [11]:

```
sports = {'Archery' : 'Bhutan',
        'Golf' : 'Scotland',
        'Sumo' : 'Japan',
        'Taekwondo' : 'South Korea'}
s = pd.Series(sports)
```

In [12]:

```
s    #인덱스가 0, 1, 2 가 아닌 키값으로
```

Out[12]:

```
Archery           Bhutan
Golf            Scotland
Sumo               Japan
Taekwondo    South Korea
dtype: object
```

In [13]:

```
s.index
```

Out[13]:

```
Index(['Archery', 'Golf', 'Sumo', 'Taekwondo'], dtype='object')
```

In [14]:

```
s1 = pd.Series(['Tiger', 'Bear', 'Moose'], index = ['Indea', 'America', 'Canada'])
s1    #인덱스 직접 설정하여 집어넣음
```

Out[14]:

```
Indea       Tiger
America      Bear
Canada      Moose
dtype: object
```

In [16]:

```
s2 = pd.Series(sports, index = ['Golf', 'Sumo'])
s2    #선택한 것만 인덱스 집어넣음
```

Out[16]:

```
Golf    Scotland
Sumo       Japan
dtype: object
```

Querying a Series Data Structure

In [17]:

```
s    #순서는 있는것 0, 1, 2, 3 순
```

Out[17]:

```
Archery           Bhutan
Golf            Scotland
Sumo               Japan
Taekwondo    South Korea
dtype: object
```

In [18]:

```
s.iloc[2]    #순서로 접근할 때 iloc 이용
```

Out[18]:

```
'Japan'
```

In [19]:

```
s.loc['Taekwondo']    #그 키값에 접근
```

Out[19]:

```
'South Korea'
```

In [20]:

```
s[2]
```

Out[20]:

```
'Japan'
```

In [21]:

```
s['Taekwondo']
```

Out[21]:

```
'South Korea'
```

In [25]:

```
sports = {5 : 'Bhutan',
          6 : 'Scotland',
          7 : 'Japan',
          8 : 'South Korea'}
s3 = pd.Series(sports)
s3
```

Out[25]:

```
5         Bhutan
6       Scotland
7          Japan
8    South Korea
dtype: object
```

In [27]:

```
s3.iloc[0]    #iloc사용하는게 좋음
```

Out[27]:

```
'Bhutan'
```

In [31]:

```
s3[0]    #숫자 인덱스 0이 없어서 에러 s3[0]==s3.loc[0]
```

```
---------------------------------------------------------------------------
-
KeyError                                  Traceback (most recent call last)
~Wanaconda3WlibWsite-packagesWpandasWcoreWindexesWbase.py in get_loc(self,
 key, method, tolerance)
   2894            try:
-> 2895                return self._engine.get_loc(casted_key)
   2896            except KeyError as err:

pandasW_libsWindex.pyx in pandas._libs.index.IndexEngine.get_loc()

pandasW_libsWindex.pyx in pandas._libs.index.IndexEngine.get_loc()

pandasW_libsWhashtable_class_helper.pxi in pandas._libs.hashtable.Int64HashTabl
e.get_item()

pandasW_libsWhashtable_class_helper.pxi in pandas._libs.hashtable.Int64HashTabl
e.get_item()

KeyError: 0

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
<ipython-input-31-e43b00e01dff> in <module>
----> 1 s3[0]    #숫자 인덱스 0이 없어서 에러 s3[0]==s3.loc[0]

~Wanaconda3WlibWsite-packagesWpandasWcoreWseries.py in __getitem__(self, ke
y)
    880
    881        elif key_is_scalar:
--> 882            return self._get_value(key)
    883
    884        if is_hashable(key):

~Wanaconda3WlibWsite-packagesWpandasWcoreWseries.py in _get_value(self, lab
el, takeable)
    987
    988        # Similar to Index.get_value, but we do not fall back to pos
itional
--> 989        loc = self.index.get_loc(label)
    990        return self.index._get_values_for_loc(self, loc, label)
    991

~Wanaconda3WlibWsite-packagesWpandasWcoreWindexesWbase.py in get_loc(self,
 key, method, tolerance)
   2895                return self._engine.get_loc(casted_key)
   2896            except KeyError as err:
-> 2897                raise KeyError(key) from err
   2898
   2899        if tolerance is not None:

KeyError: 0
```

In [32]:

```
s4 = pd.Series([100, 120, 101, 3])
s4
```

Out[32]:

```
0    100
1    120
2    101
3      3
dtype: int64
```

In [33]:

```
total = 0    #s4에 있는 item 다 더하기
for item in s4 :
    total += item
print(total)
```

```
324
```

In [34]:

```
total = np.sum(s4)    #넘파이 함수 활용해서 합 구함
print(total)
```

```
324
```

In [35]:

```
s5 = pd.Series(np.random.randint(0, 1000, 10000)) #0~1000 중에 10000개 생성
s5
```

Out[35]:

```
0        838
1        580
2        552
3        330
4        518
       ...
9995     133
9996     782
9997     118
9998     365
9999     538
Length: 10000, dtype: int32
```

In [36]:

```
s5.head()    #맨 앞에 5개만 보여줌
```

Out[36]:

```
0    838
1    580
2    552
3    330
4    518
dtype: int32
```

In [37]:

```
%%timeit -n 100    #for문으로 합 구하는데 걸린 시간
total = 0
for item in s5 :
    total += item
```

```
2.63 ms ± 163 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

In [38]:

```
%%timeit -n 100    #numpy로 합 구하는데 걸린 시간 넘파이가 for문보다 빠름
total = np.sum(s5)
```

```
112 µs ± 19.2 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

In [39]:

```
#Add data
s6 = pd.Series([1, 2, 3])
s6
```

Out[39]:

```
0    1
1    2
2    3
dtype: int64
```

In [40]:

```
s6.loc['Animal'] = 'Bears'
s6    #dtype object로 바뀜
```

Out[40]:

```
0            1
1            2
2            3
Animal    Bears
dtype: object
```

In [41]:

```
s
```

Out[41]:

```
Archery           Bhutan
Golf            Scotland
Sumo               Japan
Taekwondo     South Korea
dtype: object
```

In [42]:

```
cricket_countries = pd.Series(['Australia', 'Pakistan', 'England'], index = ['Cricket', 'Cricket', '
cricket_countries
```

Out[42]:

```
Cricket    Australia
Cricket     Pakistan
Cricket      England
dtype: object
```

In [43]:

```
all_countries = s.append(cricket_countries)    #append 말단에 데이터 붙임
all_countries
```

Out[43]:

```
Archery           Bhutan
Golf            Scotland
Sumo               Japan
Taekwondo     South Korea
Cricket         Australia
Cricket          Pakistan
Cricket           England
dtype: object
```

In [44]:

```
s
```

Out[44]:

```
Archery           Bhutan
Golf            Scotland
Sumo               Japan
Taekwondo     South Korea
dtype: object
```

In [45]:

```
all_countries.loc['Cricket']    #loc사용해서 querying 가능
```

Out[45]:

```
Cricket    Australia
Cricket     Pakistan
Cricket      England
dtype: object
```

In [46]:

```
all_countries.loc['Sumo']
```

Out[46]:

```
'Japan'
```

The DataFrame Data Structure

In [47]:

```
purchase_1 = pd.Series({'Name' : 'Chris',
                        'Item Purchased' : 'Dog Food',
                        'Cost' : 22.50})
purchase_2 = pd.Series({'Name' : 'Kevyn',
                        'Item Purchased' : 'Kitty Litter',
                        'Cost' : 2.50})
purchase_3 = pd.Series({'Name' : 'Chris',
                        'Item Purchased' : 'Bird Seed',
                        'Cost' : 5.0})
```

In [48]:

```
purchase_1
```

Out[48]:

```
Name                 Chris
Item Purchased    Dog Food
Cost                  22.5
dtype: object
```

In [49]:

```
df = pd.DataFrame([purchase_1, purchase_2, purchase_3], index = ['Store 1', 'Store 1', 'Store 2'])
df.head()
```

Out[49]:

	Name	Item Purchased	Cost
Store 1	Chris	Dog Food	22.5
Store 1	Kevyn	Kitty Litter	2.5
Store 2	Chris	Bird Seed	5.0

In [50]:

```
df.loc['Store 2']    #data 1개라 seriese 로 보여줌
```

Out[50]:

```
Name                  Chris
Item Purchased    Bird Seed
Cost                      5
Name: Store 2, dtype: object
```

In [51]:

```
df.loc['Store 1']    #data 2개라 dataframe 형태로 보여줌
```

Out[51]:

	Name	Item Purchased	Cost
Store 1	Chris	Dog Food	22.5
Store 1	Kevyn	Kitty Litter	2.5

In [52]:

```
df.loc['Store 1']['Cost']    #dataframe에서 cost만 읽어옴
```

Out[52]:

```
Store 1    22.5
Store 1     2.5
Name: Cost, dtype: float64
```

In [53]:

```
df.loc['Store 1', 'Cost']    #이렇게도 가능
```

Out[53]:

```
Store 1    22.5
Store 1     2.5
Name: Cost, dtype: float64
```

In [54]:

```
df.T    #행 열 바꿈
```

Out[54]:

	Store 1	Store 1	Store 2
Name	Chris	Kevyn	Chris
Item Purchased	Dog Food	Kitty Litter	Bird Seed
Cost	22.5	2.5	5

In [55]:

```
df.T.loc['Cost']
```

Out[55]:

```
Store 1    22.5
Store 1     2.5
Store 2       5
Name: Cost, dtype: object
```

In [56]:

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

Out[56]:

```
Store 1    22.5
Store 1     2.5
Store 2     5.0
Name: Cost, dtype: float64
```

In [57]:

```
df.loc[:, ['Name', 'Cost']]
#==SQL : select Name, Cost from df
```

Out[57]:

	Name	Cost
Store 1	Chris	22.5
Store 1	Kevyn	2.5
Store 2	Chris	5.0

In [58]:

```
df.drop('Store 1')    #삭제된 모습만 보여줌 실제로 삭제된것 아님
```

Out[58]:

	Name	Item Purchased	Cost
Store 2	Chris	Bird Seed	5.0

In [59]:

```
df    #직접적으로 데이터를 바꾼게 아니기 때문에 그대로 있음
```

Out[59]:

	Name	Item Purchased	Cost
Store 1	Chris	Dog Food	22.5
Store 1	Kevyn	Kitty Litter	2.5
Store 2	Chris	Bird Seed	5.0

In [60]:

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

In [62]:

```
copy_df = df.drop('Store 1')
copy_df
```

Out[62]:

	Name	Item Purchased	Cost
Store 2	Chris	Bird Seed	5.0

In [63]:

```
df
```

Out[63]:

	Name	Item Purchased	Cost
Store 1	Chris	Dog Food	22.5
Store 1	Kevyn	Kitty Litter	2.5
Store 2	Chris	Bird Seed	5.0

In [64]:

```
del df['Name']    #컬럼 삭제 진짜로 삭제됨
```

In [65]:

```
df
```

Out[65]:

	Item Purchased	Cost
Store 1	Dog Food	22.5
Store 1	Kitty Litter	2.5
Store 2	Bird Seed	5.0

In [66]:

```
df['Location'] = None    #컬럼 집어넣음
df
```

Out[66]:

	Item Purchased	Cost	Location
Store 1	Dog Food	22.5	None
Store 1	Kitty Litter	2.5	None
Store 2	Bird Seed	5.0	None