Chapter – 23 Pandas Examples

Pandas ဥပမာများ

Python programming language မှ data များအတွက် အထူးပြုလုပ်ထားသည့် Pandas package မှ syntax များကို ဥပမာများဖြင့် တစ်ကြောင်းချင်းစီ ရှင်းပြထားသည်။ မိမိစက်တွင် install လုပ်ထားသည့် pandas version ကို သိလိုလျှင် **`pd.__version__`** ကို အသုံးပြုနိုင်သည်။

```
import pandas as pd
print(pd.__version__)
```

```
1.0.1
```

Pandas သုံးရန်အတွက် လိုအပ်သည့် အရာများ(dependencies) အားလုံး မိမိကွန်ပျူတာတွင် ရှိမရှိကို သိလိုလျှင် **`print(pd.show_versions(as_json=True))`** ဖြင့် ဖတ်ရှုနိုင်သည်။

```
# Print all pandas dependencies
print(pd.show_versions(as_json=True))
```

```
{'system': {'commit': None, 'python': '3.7.1.final.0', 'python-
bits': 64, 'OS': 'Windows', 'OS-release': '8.1', 'machine': 'AMD64',
'processor': 'Intel64 Family 6 Model 58 Stepping 9, GenuineIntel',
'byteorder': 'little', 'LC_ALL': 'None', 'LANG': 'None', 'LOCALE':
'None.None'}, 'dependencies': {'pandas': '0.23.4', 'pytest':
'4.0.2', 'pip': '18.1', 'setuptools': '40.6.3', 'Cython': '0.29.2',
'numpy': '1.15.4', 'scipy': '1.1.0', 'pyarrow': None, 'xarray':
None, 'IPython': '7.2.0', 'sphinx': '1.8.2', 'patsy': '0.5.1',
'dateutil': '2.7.5', 'pytz': '2018.7', 'blosc': None, 'bottleneck':
'1.2.1', 'tables': '3.4.4', 'numexpr': '2.6.8', 'feather': None,
'matplotlib': '3.0.2', 'openpyxl': '2.5.12', 'xlrd': '1.2.0',
'xlwt': '1.3.0', 'xlsxwriter': '1.1.2', 'lxml': '4.2.5', 'bs4':
'4.6.3', 'html5lib': '1.0.1', 'sqlalchemy': '1.2.15', 'pymysql':
None, 'psycopg2': None, 'jinja2': '2.10', 's3fs': None,
'fastparquet': None, 'pandas_gbq': None, 'pandas_datareader': None}}
None
```

ဥပမာ- **`'pymysql': None`** ။ 'pymysql' သည် None ဖြစ်နေသဖြင့် mysql နှင့်သက်ဆိုင်သည့်အရာများ pandas မှ မလုပ်ပေးနိုင်ပါ။ အသုံးပြုလိုပါက 'pymysql' ကို install လုပ်ရပါမည်။

23.1 List, Numpy Array သို့မဟုတ် Dict တစ်ခုကို Pandas Series တစ်ခုအဖြစ် ပြုလုပ်ခြင်း

```
import numpy as np
a_list = list("abcdefg")
numpy_array = np.arange(1, 10)
my_dictionary = {"A":  0, "B":1, "C":2, "D":3, "E":5}
```

`a_list = list("abcdefg")` ဖြင့် list တစ်ခု တည်ဆောက်သည်။

```
a_list                          # တည်ဆောက်ပြီးသည့် list ကို ကြည့်ရန်
```

```
['a', 'b', 'c', 'd', 'e', 'f', 'g']
```

numpy_array = np.arange(1, 10) ဖြင့် numpy array တစ်ခု တည်ဆောက်သည်။

```
numpy_array                    # တည်ဆောက်ပြီးသည့် numpy array ကို ကြည့်ရန်
```

```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

dictionary = {"A": 0, "B":1, "C":2, "D":3, "E":5} ဖြင့် dict တစ်ခု တည်ဆောက်သည်။

```
my_dictionary                  # တည်ဆောက်ပြီးသည့် dict ကို ကြည့်ရန်
```

```
{'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 5}
```

pd.Series(a_list) ဖြင့် list ကို pandas series တစ်ခုအဖြစ် ပြောင်းသည်။

```
series1 = pd.Series(a_list)        # pandas series တစ်ခုအဖြစ် ပြောင်းသည်။
print(series1)
```

```
0    a
1    b
2    c
3    d
dtype: object
```

pd.Series(numpy_array) ဖြင့် numpy_array ကို pandas series တစ်ခုအဖြစ် ပြောင်းသည်။

```
series2 = pd.Series(numpy_array)  # numpy_array ကို pandas series အဖြစ် ပြောင်းသည်။
print(series2)
```

```
0    1
1    2
2    3
dtype: int32
```

pd.Series(my_dictionary) ဖြင့် dictionary ကို pandas series တစ်ခုအဖြစ် ပြောင်းသည်။

```
series3 = pd.Series(my_dictionary) # dictionary ကို pandas series အဖြစ် ပြောင်းသည်။
print(series3)
```

```
A    0
B    1
C    2
```

```
D    3
E    5
dtype: int64
```

23.2 Series ၏ Index ကို Dataframe ၏ ကော်လံတစ်ခုအဖြစ် ပြောင်းခြင်း

Series တစ်ခု၏ index (index of a series)ကို dataframe ၏ ကော်လံတစ်ခုအဖြစ် ပြောင်းလိုလျှင် နည်းနှစ်နည်းကို အသုံးပြုနိုင်သည်။

ပထမဦးစွာ a မှ z အထိ အင်္ဂလိပ် အက္ခရာများ ပါဝင်သည့် list တစ်ခု ပြုလုပ်သည်။ ဒုတိယ 0 မှ 25 အထိ အစဉ်လိုက်ကိန်းများ ပါဝင်သည့် numpy array တစ်ခုအဖြစ် တည်ဆောက်သည်။ အင်္ဂလိပ်အက္ခရာများ ပါဝင်သည့် list နှင့် 0 မှ 25 အထိ အစဉ်လိုက်ကိန်းများ ပါဝင်သည့် numpy array ကို **zip()** ဖြင့် တွဲပြီး dictionary တစ်ခုအဖြစ် တည်ဆောက်သည်။ ထို dictionary ကို pandas series တစ်ခုအဖြစ် ပြောင်းသည်။

```
mylist = list('abcedfghijklmnopqrstuvwxyz')  # list
myarr = np.arange(26)                        # array
mydict = dict(zip(mylist, myarr))            # dict
ser = pd.Series(mydict)                      # pandas series
print(ser[:5])                               # ပထမဆုံး(၅)ခုကို ဖော်ပြရန်
```

```
a    0
b    1
c    2
e    3
d    4
dtype: int64
```

23.2.1 pd.DataFrame() ဖြစ် DataFrame တစ်ခု ပြုလုပ်ခြင်း

နည်း(၂)မျိုးဖြင့် DataFrame တစ်ခု ပြုလုပ်နိုင်သည်။ ပထမနည်းသည် **`pd.DataFrame()`** ကို သုံးသည့်နည်း ဖြစ်သည်။ Pandas series ကို DataFrame အဖြစ် ပြောင်းသည်။ ထို့နောက် index ကို reset လုပ်သည်။ **`.reset_index()`** မလုပ်ခင် ser_df ကို print ဖြင့် ကြည့်ပါ။

```
ser_df = pd.DataFrame(ser)          # using DataFrame
ser_df.reset_index()
```

	index	0
0	a	0
1	b	1
2	c	2
..	..	..
24	y	24
25	z	25

ဒုတိယနည်းသည် **`to_frame()`** ဖြင့် DataFrame အဖြစ် ပြောင်းသည်။ ထို့နောက် **`.reset_index()`** ဖြင့် index ကို reset လုပ်သည်။

```
ser_df = ser.to_frame().reset_index() # using pandas to_frame()
ser_df
```

	index	0
0	a	0
1	b	1
2	c	2
..	..	..
24	y	24
25	z	25

23.2.2 Series များကို ပေါင်း၍ DataFrame တစ်ခုအဖြစ် ပြောင်းလဲခြင်း

Series များကို ပေါင်း၍ DataFrame တစ်ခုအဖြစ် ပြောင်းသည့်နည်း(to combine many series to form a dataframe)ကို အောက်တွင် ဖော်ပြထားသည်။ ser1 နှင့် ser2 ကို ပေါင်း၍ dataframe တစ်ခု အဖြစ် ပြောင်းလဲပုံကို ဖော်ပြထားသည်။ ser1 နှင့် ser2 တည်ဆောက်ပုံကို အထက်တွင် ဖော်ပြပြီး ဖြစ်သည်။

```
ser1 = pd.Series(list('abcedfghijklmnopqrstuvwxyz'))
ser2 = pd.Series(np.arange(26))
```

ပထမနည်းသည် ser1 နှင့် ser2 ဆိုသည့် Pandas series နှစ်ခု တည်ဆောက်ပြီး DataFrame အဖြစ်သို့ ပြောင်းသည့်နည်း ဖြစ်သည်။ `pd.DataFrame()` ကိုသုံး၍ DataFrame တစ်ခုအဖြစ် ပြောင်းသည်။ ထို့နောက် index ကို reset လုပ်သည်။

```
ser_df = pd.DataFrame(ser1, ser2).reset_index()
ser_df.head()
```

	index	0
0	0	a
1	1	b
2	2	c
3	3	e
4	4	d

ဒုတိယနည်းသည် dictionary တစ်ခုအဖြစ် ပြုလုပ်ပြီးမှ DataFrame သို့ ပြောင်းသည့်နည်း ဖြစ်သည်။ Dictionary အဖြစ် တည်ဆောက်ချိန်တွင် ကော်လံများ၏ နာမည်ကို မိမိကြိုက်သည့် နာမည် ပေးနိုင်သည်။

```
ser_df = pd.DataFrame({"col1":ser1, "col2":ser2})
ser_df.head(5)
```

	col1	col2
0	a	0
1	b	1
2	c	2
3	e	3

4	d	4

တတိယနည်းသည် Pandas series နှစ်ခုကို ပေါင်းဆက်(concatenate လုပ်)၍ dataframe အဖြစ် ပြောင်းသည့်နည်း ဖြစ်သည်။

```
ser_df = pd.concat([ser1, ser2], axis = 1)  # using pandas concat
ser_df.head()
```

	0	1
0	a	0
1	b	1
2	c	2
3	e	3
4	d	4

23.3 Series တစ်ခု၏ Index အား နာမည် Assign လုပ်ခြင်း

List တစ်ခု ပြုလုပ်ပြီးနောက် pandas series တစ်ခုအဖြစ် ပြောင်းသည်။ Series ၏ index အမည်ကို ‘alphabets’ ဟု ပေးသည်။ (To assign name to the series’ index.)

```
ser = pd.Series(list('abcedfghijklmnopqrstuvwxyz'))
```

`ser.rename("alphabets")` ဖြင့် နာမည်ပြောင်းသည်။ `rename()` ကို သုံး၍ နာမည်ပြောင်းသည်။

```
ser.rename("alphabets")          # using series rename method
```

```
0     a
1     b
2     c
...
24    y
25    z
Name: alphabets, dtype: object
```

Series attribute ဖြစ်သည့် `.name` ကို အသုံးပြု၍ "other_name" ဟု နာမည်တစ်ခု assign လုပ်သည်။

```
ser.name = "other_name"         # using series attribute
ser
```

```
0     a
1     b
2     c
...
24    y
25    z
Name: other_name, dtype: object
```

ser2 ထဲတွင် မပါဝင်သည့် ser1 ထဲမှ item များ ကို ရွေးချယ်ပုံ(to get the items of series A not present in series B)ကို အောက်တွင် ဖော်ပြထားသည်။

```
ser1 = pd.Series([1, 2, 3, 4, 5])          # input ser1
ser2 = pd.Series([4, 5, 6, 7, 8])          # input ser2
ser1[~ser1.isin(ser2)]
```

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

23.4 Series နှစ်ခုစလုံးတွင် ဘုံအဖြစ် မပါဝင်သည့် ကိန်းများကို ရှာခြင်း

ser1 နှင့် ser2 နှစ်ခုစလုံးတွင် ဘုံအဖြစ် မပါဝင်သည့် ကိန်းများကို ရှာသည့် (to get the items not common to both series, ser1 and ser2?) ကုဒ်ကို အောက်တွင် ဖော်ပြထားသည်။ ser1 နှင့် ser2 တို့တွင် ရှိသည့် item များအနက် နှစ်ခုစလုံးတွင် ဘုံအဖြစ် မပါဝင်သည့် ser1 မှ item များ အားလုံးနှင့် ser2 မှ item များ အားလုံးကို ဖော်ပြရန် ဖြစ်သည်။

```
ser1 = pd.Series([1, 2, 3, 4, 5])      # input ser1
ser2 = pd.Series([4, 5, 6, 7, 8])      # input ser2
```

ပထမနည်း- pandas နည်း

```
a_not_b = ser1[~ser1.isin(ser2)]       # using pandas
b_not_a = ser2[~ser2.isin(ser1)]

a_not_b.append(b_not_a, ignore_index = True)
```

```
0    1
1    2
2    3
3    6
4    7
5    8
dtype: int64
```

ဒုတိယနည်းသည် numpy union နှင့် intersection ကို သုံးသည့်နည်း ဖြစ်သည်။ `np.union1d()` သည် နှစ်ခုထက် ပိုများသည့် array များကို one dimensional union လုပ်ခြင်းဖြစ်သည်။ Union လုပ်ချင်သည့် series နှစ်ခုကို ထည့်ပေးရသည်။ `np.intersect1d()` သည် နှစ်ခုထက် ပိုများသည့် array များကို one dimensional intersection လုပ်ခြင်း ဖြစ်သည်။

```
# using numpy union and intersection
ser_u = pd.Series(np.union1d(ser1, ser2))
ser_i = pd.Series(np.intersect1d(ser1, ser2))
ser_u[~ser_u.isin(ser_i)]
```

```
0    1
1    2
2    3
```

```
5    6
6    7
7    8
dtype: int64
```

23.5 Series ထဲရှိ ကိန်းတစ်လုံးစီ အကြိမ် အရေအတွက်ကို ရှာခြင်း

Series ထဲတွင် ပါဝင်နေသည့် ကိန်းတစ်လုံးစီ၏ ပါဝင်နေသည့် အကြိမ်အရေအတွက်ကို သိလိုလျှင် (to get frequency counts of unique items of a series) အောက်ပါအတိုင်း ရှာနိုင်သည်။ (Calculate the frequency counts of each unique value ser.)

```
ser = pd.Series(np.take(list('abcdefgh'),
                np.random.randint(8, size=30)))
ser
```

```
0     b
1     b
2     e
3     e
4     g
5     c
6     d
7     g
8     d
9     h
10    g
11    h
12    c
13    b
14    a
15    a
16    a
17    a
18    c
19    f
20    c
21    h
22    e
23    e
24    f
25    e
26    b
27    b
28    a
29    a
dtype: object
```

`.value_counts()` ကို အသုံးပြု၍ ဖော်ပြနိုင်သည်။

```
ser.value_counts()
```

```
a    6
e    5
b    5
c    4
g    3
h    3
f    2
d    2
dtype: int64
```

a (၆)ကြိမ် ပါဝင်သည်။

e (၅)ကြိမ် ပါဝင်သည်။

23.6 အများဆုံးပါဝင်သည့် ကိန်းကို ရှာခြင်း

အများဆုံးပါဝင်သည့် ကိန်းနှစ်လုံးကိုသာ ဖော်ပြပြီး ကျန်ကိန်းများကို 'Other' ဆိုသည် စကားလုံးဖြင့် အစားထိုးရန် အောက်ပါအတိုင်း လုပ်နိုင်သည်။ (To keep only top 2 most frequent values as it is and replace everything else as 'Other)

```
np.random.RandomState(100)
ser = pd.Series(np.random.randint(1, 5, [12]))
ser
```

```
0     1
1     4
2     1
3     2
4     2
5     1
6     4
7     2
8     2
9     2
10    2
11    1
dtype: int32
```

ကိန်းတစ်လုံးစီ၏ အကြိမ်အရေအတွက် မည်မျှပါသည်ကို သိရန် **value_counts** ကို သုံးပြီး အများဆုံး ကိန်း(၂)လုံးကို ယူသည်။ **~ser.isin** ကို သုံး၍ filter လုပ်သည်။ ကျန်ကိန်းများကို Other အဖြစ် assign လုပ်သည်။

```
ser.value_counts()
ser[~ser.isin(ser.value_counts().index[:2])] = 'Other'
ser
```

```
0         1
1     Other
2         1
3         2
4         2
5         1
6     Other
7         2
8         2
9         2
10        2
11        1
dtype: object
```

23.7 Numpy Array တစ်ခုကို DataFrame အဖြစ် ပြုလုပ်ခြင်း

Numpy array တစ်ခုကို မိမိ အလိုရှိသည့် shape အတိုင်း dataframe တည်ဆောက်လိုလျှင် (to convert a numpy array to a dataframe of given shape)

```
# input
ser3 = pd.Series(np.random.randint(1, 10, 35))
ser3
```

```
0     5
1     8
2     9
3     9
4     8
5     2
6     7
7     3
8     4
9     9
10    2
11    7
12    2
13    7
14    3
15    4
16    9
17    2
18    8
19    2
20    3
21    1
22    1
23    2
24    2
25    8
26    2
27    3
28    1
```

```
5     4          17    4          29    2
6     3          18    5          30    5
7     1          19    7          31    1
8     4          20    7          32    4
9     1          21    8          33    9
10    8          22    4          34    9
11    5          23    7          dtype: int32
```

Series တစ်ခုကို row 7 ခု နှင့် ကော်လံ 5 ပါသည့် dataframe အဖြစ် ပြောင်းလဲသည်။ (Reshape the series ser into a dataframe with 7 rows and 5 columns)

```
pd.DataFrame(np.array(ser3).reshape(7, 5))     # using numpy
```

	0	1	2	3	4
0	9	1	3	1	1
1	2	1	3	1	5
2	2	8	2	1	8
3	6	1	8	3	9
4	5	6	8	6	2
5	5	4	2	9	1
6	9	1	4	1	1

23.8 Series တစ်ခု၏ သတ်မှတ်ထားသည့်နေရာမှ Item များကို ထုတ်ယူခြင်း

Series တစ်ခု၏ သတ်မှတ်ထားသော နေရာမှ item များကို ထုတ်ယူသည်။ (to extract items at given positions from a series)။ သတ်မှတ်ထားသော နေရာများကို list တစ်ခု အဖြစ် ထည့်ပေးသည်။ ထိုသို့ ထည့်ပေးလိုက်သည့် list အတိုင်း item များကို ထုတ်ယူသည်။ Access လုပ်သည်။ (From a series, extract the items at positions in list pos.)

```
ser4 = pd.Series(list('abcdefghijklmnopqrstuvwxyz'))
pos_list = [0, 4, 8, 14, 20]                  # position များ ပါဝင်သည့် list
```

`.loc[]` (location)ကို သုံး၍ ထုတ်ယူသည်။ Access လုပ်သည်။

```
ser4.loc[pos_list]       # using loc
```

```
0     a
4     e
8     i
14    o
20    u
dtype: object
```

`.take()` ကို သုံး၍ ထုတ်ယူသည်။ Access လုပ်သည်။

```
ser4.take(pos_list)                    # using series take
```

```
0     a
4     e
8     i
14    o
20    u
dtype: object
```

23.9 Series နှစ်ခုကို DataFrame ပြုလုပ်ခြင်း

Series နှစ်ခုကို ဒေါင်လိုက်(vertically) ထပ်၍ dataframe ပြုလုပ်နိုင်သလို၊ အလျားလိုက် (horizontally) ဘေးချင်းယှဉ်၍ dataframe ပြုလုပ်နိုင်သည်။

```
ser1 = pd.Series(range(5))
ser2 = pd.Series(list('abcde'))
```

ser1.append(ser2) ser1 ထဲသို့ ser2 ကို ထပ်ထည့်ပြီး ပေါင်းသည်။ တစ်နည်းအားဖြင့် append လုပ်ပြီး ပေါင်းသည်။

```
ser1.append(ser2)  # ဒေါင်လိုက်(vertically) ထပ်၍ dataframe ပြုလုပ်သည်။
```

```
0    0                    1    b
1    1                    2    c
2    2                    3    d
3    3                    4    e
4    4                    dtype: object
0    a
```

ser1 နှင့် ser2 ကို ပေါင်း(concat လုပ်)သည်။ Concatenation လုပ်သည့်အခါ ပေါင်းလိုသည့် axis ကို ပြောပေးရသည်။ မသတ်မှတ်ပေးလျှင် default axis = 0 ဖြစ်သည်။ axis = 0 သည် ဒေါင်လိုက် ဖြစ်အောင် ပေါင်းသည်။ axis = 0, ဒေါင်လိုက် ဖြစ်အောင်ပေါင်းသည် ဆိုသည်မှာ data frame တွင် row အသစ်များ ထပ်ထိုးလာအောင် ပေါင်းခြင်း ဖြစ်သည်။

```
# using pandas concat and axis = 0
pd.concat([ser1, ser2], axis = 0)
```

```
0    0                    1    b
1    1                    2    c
2    2                    3    d
3    3                    4    e
4    4                    dtype: object
0    a
```

axis = 1 သည် အလျားလိုက် ဖြစ်အောင်ပေါင်းသည်။ axis = 1 အလျားလိုက် ဖြစ်အောင်ပေါင်းသည် ဆိုသည်မှာ data frame တွင် ကော်လံ အသစ်များ ထပ်ထိုးလာအောင် ပေါင်းခြင်း ဖြစ်သည်။

```
# axis = 1 သည် အလျားလိုက် ဖြစ်အောင်ပေါင်းသည်။ (horizontal)
```

```
pd.concat([ser1, ser2], axis = 1)
```

	0	1
0	0	a
1	1	b
2	2	c
3	3	d
4	4	e

23.10 Series နှစ်ခုမှ ပါဝင်သည့် Item များ၏ တည်ရှိရာနေရာကို ရှာခြင်း

ser1 နှင့် ser2 နှစ်ခုထဲမှ ser2 ထဲတွင် ပါဝင်သည့် ser1 item များ၏ တည်ရှိရာနေရာကို သိလိုလျှင် `.isin()` ကို အသုံးပြုနိုင်သည်။ (to get the positions of items of series ser1 in another series ser)

```
ser1 = pd.Series([10, 9, 6, 5, 3, 1, 12, 8, 13])
ser2 = pd.Series([1, 3, 10, 13, 7, 4])
```

ပထမ ser2 ထဲမှာ ပါသည့် ser1 ထဲမှ item ၏ နေရာများ (index နံပါတ်)ကို အောက်ပါအတိုင်း ရှာနိုင်သည်။

```
ser1[ser1.isin(ser2)]
```

```
0    10
4     3
5     1
8    13
dtype: int64
```

ser2 ထဲမှာ ပါသည့် ser1 ထဲမှ item ၏ index နံပါတ်များကိုသာ အောက်ပါအတိုင်း ထုတ်ယူနိုင်သည်။

```
# get's the index, but it's sorts the index
list(ser1[ser1.isin(ser2)].index)
```

```
[0, 4, 5, 8]
```

```
# using pandas Index and get location
[pd.Index(ser1).get_loc(i) for i in ser2]
```

```
KeyError                                  Traceback (most recent call last)
KeyError: 7
```

23.11 Series နှစ်ခု၏ Mean Squared Error တွက်ခြင်း

`truth` ဆိုသည့် series နှင့် `predicted` ဆိုသည့် series တို့၏ mean squared error တွက်ပုံ တွက်နည်းကို ဖော်ပြမည်။ ပထမဦးစွာ series နှစ်ခု တည်ဆောက်သည်။

```
truth = pd.Series(range(10))
pred = pd.Series(range(10)) + np.random.random(10)
```

ပထမနည်း- numpy နည်း

```
np.mean((truth-pred)**2)          # using numpy
```

```
0.26474101079782053
```

ဒုတိယနည်း - sklear metrics နည်း

```
# using sklear metrics
from sklearn.metrics import mean_squared_error
mean_squared_error(truth, pred)
```

```
0.26474101079782053
```

23.12 ပထမဆုံး အက္ခရာကို အကြီးစာလုံး(Upper Case)အဖြစ် ပြောင်းခြင်း

စာလုံးတစ်လုံးမှ ပထမဆုံး အက္ခရာကို အကြီးစာလုံးအဖြစ် ပြောင်းခြင်း(Change the first character of each word to upper case in each word of ser.)

```
ser = pd.Series(['just', 'a', 'random', 'list'])
ser
```

```
0      just
1         a
2    random
3      list
dtype: object
```

ပထမနည်း- python string method

Python မှ string method တစ်ခုဖြစ်သည့် `title()` ကို အသုံးပြုသည်။

```
[i.title() for i in ser]
```

```
['Just', 'A', 'Random', 'List']
```

ဒုတိယနည်း - lambda နည်း

```
ser.map(lambda x: x.title())       # using lambda
```

```
0      Just
1         A
2    Random
3      List
dtype: object
```

တတိယနည်း - `map()` နှင့် `upper()` ကို သုံးသည်။ ပထမဆုံး character ကို ရွေး၍ အကြီးစာလုံး အဖြစ် ပြောင်းသည်။

```
ser.map(lambda x: x[0].upper() + x[1:]) # other solution
```

```
0      Just
1         A
2    Random
```

```
3      List
dtype: object
```

23.13 စာလုံးတစ်လုံးတွင် ပါဝင်သည့် အက္ခရာ အရေအတွက်ကို သိလိုလျှင်

စာလုံးတစ်လုံး(word)တွင် ပါဝင်သည့် အက္ခရာ အရေအတွက်ကို သိနိုင်သည့်နည်း သုံးနည်း ရှိသည်။
(to calculate the number of characters in each word in a series?)

```
ser = pd.Series(['just', 'a', 'random', 'list'])
```

ပထမနည်း

```
[len(i) for i in ser]          # using list comprehension
```

```
[4, 1, 6, 4]
```

ဒုတိယနည်း - `.map(len)`

```
ser.map(len)                   # using series map
```

```
0    4
1    1
2    6
3    4
dtype: int64
```

တတိယနည်း - `.apply(len)`

```
ser.apply(len)                  # using series apply
```

```
0    4
1    1
2    6
3    4
dtype: int64
```

23.14 Pandas Timeseries

23.14.1 Date-Strings Series တစ်ခုကို Timeseries အဖြစ်သို့ ပြောင်းခြင်း

`ser` ဆိုသည့် date-strings series တစ်ခု ပြုလုပ်သည်။ ၄င်းကို timeseries အဖြစ်သို့ ပြောင်းသည်။

```
ser = pd.Series(['01 Jan 2010', '02-02-2011', '20120303',
                 '2013/04/04', '2014-05-05', '2015-06-06T12:20'])
```

```
0          01 Jan 2010
1           02-02-2011
2             20120303
3           2013/04/04
4           2014-05-05
5    2015-06-06T12:20
dtype: object
```

`.to_datetime()` ဖြင့် အောက်တွင် ဖော်ပြထားသည့် format အတိုင်း ပြောင်းနိုင်သည်။

```
pd.to_datetime(ser)                    # using pands to_datetime
```

```
0   2010-01-01 00:00:00
1   2011-02-02 00:00:00
2   2012-03-03 00:00:00
3   2013-04-04 00:00:00
4   2014-05-05 00:00:00
5   2015-06-06 12:20:00
dtype: datetime64[ns]
```

Date utility (`dateutil`)ကို သုံး၍ parse လုပ်သည့်နည်းဖြင့် timeseries အဖြစ်သို့ ပြောင်းသည်။

```
# using dateutil parse
from dateutil.parser import parse
ser.map(lambda x: parse(x))
```

```
0   2010-01-01 00:00:00
1   2011-02-02 00:00:00
2   2012-03-03 00:00:00
3   2013-04-04 00:00:00
4   2014-05-05 00:00:00
5   2015-06-06 12:20:00
dtype: datetime64[ns]
```

23.14.1 Date-Strings Series တစ်ခုကို တခြား Time Format များသို့ ပြောင်းခြင်း

Date-strings Series တစ်ခုကို နေ့(day of month) ၊ အပတ်(week number) ၊ day of year နှင့် day of week စသည်ဖြင့် ပြောင်းနိုင်သည်။

```
ser = pd.Series(['01 Jan 2010', '02-02-2011', '20120303', '2013/04/04',
                 '2014-05-05', '2015-06-06T12:20'])
```

`.to_datetime(ser)` ဖြင့် တစ်လ အတွင်းရှိ နေ့များ(day of month) အဖြစ်သို့ ပြောင်းနိုင်သည်။ (၁)ရက် မှ (၃၁)ရက် အတွင်း ဖြစ်နိုင်သည်။ List တစ်ခု အဖြစ် ပြန်ပေးသည်။

```
pd.to_datetime(ser).dt.day.to_list()
```

```
[1, 2, 3, 4, 5, 6]
```

```
pd.to_datetime(ser).dt.dayofyear.to_list()          # day of year
```

```
[1, 33, 63, 94, 125, 157]
```

Monday, Tuesday စသည့် ရက်နာမည်များဖြင့် ဖော်ပြရန်

```
# day of week in words
week_dict = {0:"Monday", 1:"Tuesday", 2:"Wednesday",
             3:"Thursday", 4:"Friday", 5:"Saturday", 6:"Sunday"}
pd.to_datetime(ser).dt.dayofweek.map(week_dict).to_list()
```

```
['Friday', 'Wednesday', 'Saturday', 'Thursday', 'Monday',
'Saturday']
```

23.15 Pandas မှ date_range() အကြောင်း

Time Series Analysis တွင် မသိမဖြစ် လုံးဝ အခြေခံကြသည့် **`date_range()`** အကြောင်းကို အကျဉ်းချုံး ဖော်ပြထားသည်။ **`.datetime.now()`** ဖြင့် ယခု လက်ရှိအချိန်(now)ကို ဖတ်သည်။

```
import datetime                 # datetime moduel ကို import လုပ်သည်။
datetime.datetime.now()
```

```
datetime.datetime(2020, 7, 4, 13, 53, 23, 97455)
```

23.15.1 ရက်စွဲနှစ်ခု အကြား ကြာမြင့်ခဲ့သည် ရက်ပေါင်းကို တွက်ခြင်း

```
delta = datetime.datetime(2020,5,30 - datetime.datetime(1938,6,3)
delta
```

```
datetime.timedelta(days=29947)
```

23.15.2 ရေးနိုင်သည့် ရက်စွဲပုံစံများ(Formats)

```
x = datetime.datetime(2020, 8, 17)
x
```

```
datetime.datetime(2020, 8, 17, 0, 0)
```

အောက်တွင် ရေးနိုင်သည့် ရက်စွဲပုံစံများ(format)ကို ဖော်ပြထားသည်။

```
'2016 Jul 1', '7/1/2016', '1/7/2016', 'July 1, 2016', '2016-07-01',
'2016/07/01'
```

```
('2016 Jul 1',
 '7/1/2016',
 '1/7/2016',
 'July 1, 2016',
 '2016-07-01',
 '2016/07/01')
```

```
pd.Timestamp('2020-07-10')          # နာရီ, မိနစ်, စက္ကန့် ထည့်သွင်း ဖော်ပြသည်။
```

```
Timestamp('2020-07-10 00:00:00')
```

```
# နာရီ, မိနစ်, စက္ကန့် ထည့်သွင်း ဖော်ပြသည်။
pd.Timestamp('2016-07-10 10')
```

```
Timestamp('2016-07-10 10:00:00')
```

```
# နာရီ, မိနစ်, စက္ကန့် ထည့်သွင်း ဖော်ပြသည်။
pd.Timestamp('2016-07-10 10:15')
```

```
Timestamp('2016-07-10 10:15:00')
```

23.16 Properties of Timestamps

```
t = pd.Timestamp('2016-07-10 10:15')
pd.Period('2016-01')                    # လ(month)ကိုသာ ဆိုလိုသည်။
```

```
Period('2016-01', 'M')
```

```
pd.Period('2016-01-02')                 # နေ့အထိ ဖော်ပြသည်။
```

```
Period('2016-01-02', 'D')
```

```
pd.Period('2016-01-01')                 # နေ့အထိ ဖော်ပြသည်။
```

```
Period('2016-01-01', 'D')
```

```
pd.Period('2016-01-01 10')              # နာရီအထိ ဖော်ပြသည်။
```

```
Period('2016-01-01 10:00', 'H')
```

```
pd.Period('2016-01-01 10:10')           # မိနစ်အထိ ဖော်ပြသည်။
```

```
Period('2016-01-01 10:10', 'T')
```

```
pd.Period('2016-01-01 10:10:10')        # စက္ကန့်အထိ ဖော်ပြသည်။
```

```
Period('2016-01-01 10:10:10', 'S')
```

```
pd.Timedelta('1 day')                   # Time offsets (ɔ)ရက်
```

```
Timedelta('1 days 00:00:00')
```

'2016-01-01 10:10' မှ (၁)ရက်တိတိ ကြာပြီးချိန်

```
pd.Period('2016-01-01 10:10') + pd.Timedelta('1 day')
```

```
Period('2016-01-02 10:10', 'T')
```

'2016-01-01 10:10' မှ (၁)ရက်တိတိ ကြာပြီးချိန်

```
pd.Timestamp('2016-01-01 10:10') + pd.Timedelta('1 day')
```

```
Timestamp('2016-01-02 10:10:00')
```

'2016-01-01 10:10' မှ 15 နာနို စက္ကန့်(nano sec)အကြာ

```
pd.Timestamp('2016-01-01 10:10') + pd.Timedelta('15 ns')
```

```
Timestamp('2016-01-01 10:10:00.000000015')
```

23.17 Date Rage

```
# notebook setting အတွက်
from IPython.core.interactiveshell import InteractiveShell

# notebook setting အတွက်
InteractiveShell.ast_node_interactivity = "all"
```

Pandas မှ `date_range()` ကို အသုံးပြု၍ ရက်စွဲများ၊ နေ့ရက်များ၊ အချိန်များကို နေ့အလိုက် ဖြစ်အောင် ပြင်ဆင်နိုင်သည်။ Pandas မှ `date_range()` ပုံစံ(format)သည် အောက်ပါအတိုင်း ဖြစ်သည်။

```
date_range(
    start=None,
    end=None,
    periods=None,
    freq=None,
    tz=None,
    normalize=False,
    name=None,
    closed=None,
    **kwargs,)
```

23.17.1 ၂၀၂၀ ခုနှစ် ဇူလိုင်လ (၁)ရက်နေ့(2020 Jul 1)မှ စ၍ နောက်ထပ် (၁၀)ရက်

၂၀၂၀ ခုနှစ် ဇူလိုင်လ (၁)ရက်နေ့(2020 Jul 1)မှ စ၍ နောက်ထပ် (၁၀)ရက်ကို ရက်အလိုက် ဖြစ်အောင် **date_range()** ဖြင့် ရေးသည်။

- စမည့်ရက်(start='2020 Jul 1')ထည့်ပေးရသည်။

- ကြာမည့် အချိန်ကာလ(periods = 10)ထည့်ပေးရသည်။ စုစုပေါင်း (၁၀)ရက်
- ပိုင်းခြားရမည့် ပမာဏ, စက္ကန့်တိုင်း, နာရီတိုင်း၊ နေ့စဉ်, လစဉ်, နှစ်စဉ်, ရုံဖွင့်ရက်တိုင်း စသည်ဖြင့် ထည့်ပေးရသည်။
- **`freq = 'D'`** သည် frequency = daliy ဖြစ်သည်။ D = daliy

```
rng = pd.date_range('2020 Jul 1', periods = 10, freq = 'D')
rng
```

```
DatetimeIndex(['2020-07-01', '2020-07-02', '2020-07-03', '2020-07-04',
               '2020-07-05', '2020-07-06', '2020-07-07', '2020-07-08',
               '2020-07-09', '2020-07-10'],
               dtype='datetime64[ns]', freq='D')
```

၂၀၂၀ ခုနှစ် ဇူလိုင်လ (၁)နေ့(2020 Jul 1)မှ နောက်ထပ် (၁၂)လကို လလိုက်ဖြစ်အောင် ထုတ်ပေးမည်ကုဒ်ကို အောက်တွင် ဖော်ပြထားသည်။

- စမည့်ရက်(start='2020 Jul 1')ထည့်ပေးရသည်။
- ကြာမည့်အချိန်ကာလ(periods = 12)ထည့်ပေးရသည်။ စုစုပေါင်း (၁၂)လ
- ပိုင်းခြားရမည့် ပမာဏ, စက္ကန့်တိုင်း, နာရီတိုင်း၊ နေ့စဉ်, လစဉ်, နှစ်စဉ်, ရုံဖွင့်ရက်တိုင်း စသည်ဖြင့် ထည့်ပေးရသည်။
- **`freq = 'M'`** သည် frequency = monthly ဖြစ်သည်။ M = monthly

```
m_range = pd.date_range('2020 Jul 1', periods = 12, freq = 'M')
m_range
```

```
DatetimeIndex(['2020-07-31', '2020-08-31', '2020-09-30',
               '2020-10-31','2020-11-30', '2020-12-31',
               '2021-01-31', '2021-02-28', '2021-03-31',
               '2021-04-30', '2021-05-31', '2021-06-30'],
               dtype='datetime64[ns]', freq='M')
```

စမည့်ရက် နှင့် နောက်ဆုံးရက် ထည့်ပေးလျှင်လည်းရသည်။ စမည့်ရက် နှင့် နောက်ဆုံးရက် ထည့်ပေးလျှင် **`freq = 'D'`** သည် daliy frequency သည် default ဖြစ်သည်။ D = daliy ဟု ဖော်ပြပေးသည်။

```
pd.date_range(start='1/1/2018', end='1/08/2018')
```

```
DatetimeIndex(['2018-01-01', '2018-01-02', '2018-01-03',
               '2018-01-04','2018-01-05', '2018-01-06',
               '2018-01-07', '2018-01-08'],
               dtype='datetime64[ns]', freq='D')
```

စမည့်ရက်(start='1/1/2018') နှင့် ကြာမည့်အချိန်ကာလ(periods = 8)ထည့်ပေးရသည်။ စုစုပေါင်း (၈) ရက်

```
pd.date_range(start='1/1/2018', periods=8)
```

```
DatetimeIndex(['2018-01-01', '2018-01-02', '2018-01-03',
               '2018-01-04','2018-01-05', '2018-01-06',
               '2018-01-07', '2018-01-08'],
               dtype='datetime64[ns]', freq='D')
```

နောက်ဆုံးရက်(end='2020 Jul 1')နှင့် ကြာမည့်အချိန်ကာလ(periods = 8)ထည့်ပေးရသည်။ စုစုပေါင်း (၈)ရက် ဖြစ်သည်။

```
pd.date_range(end='2020 Jul 1', periods=8)
```

```
DatetimeIndex(['2020-06-24', '2020-06-25', '2020-06-26',
               '2020-06-27', '2020-06-28', '2020-06-29',
               '2020-06-30', '2020-07-01'],
               dtype='datetime64[ns]', freq='D')
```

'2018-04-24'မှ စ၍ (၁)ရက်ခြားစီ ဆေးခန်း (၃)ကြိမ် ပြရမည့် ရက်များကို ဖော်ပြသည်။

```
pd.date_range(start='2018-04-24', periods=3, freq = '2D')
```

```
DatetimeIndex(['2018-04-24', '2018-04-26', '2018-04-28'],
              dtype='datetime64[ns]', freq='2D')
```

23.17.2 Multiples are Allowed

- ရက်တွေ, လတွေ, နှစ်တွေကို မြောက်ဖော်ကိန်းဖြင့် ရေးနိုင်သည်။
- 3M သည် ၃လ ဖြစ်သည်။ 7D သည် တစ်ပတ်ဖြစ်သည်။
- 1/7/2020 နေ့မှ စ၍ လကုန်ရက်တိုင်း (၃)လစာ ပင်စင်ထုတ်ရမည်။ တစ်နှစ်စာ ဖြစ်သည်။

```
pd.date_range(start='1/7/2020', periods=4, freq='3M') # 3M = ၃လ
```

```
DatetimeIndex(['2020-01-31', '2020-04-30', '2020-07-31',
               '2020-10-31'], dtype='datetime64[ns]', freq='3M')
```

```
pd.date_range(start='1/1/2018', periods=5, freq='3M')
```

```
DatetimeIndex(['2018-01-31', '2018-04-30', '2018-07-31',
               '2018-10-31', '2019-01-31'],
                dtype='datetime64[ns]', freq='3M')
```

23.17.3 Frequency ကို Offset လုပ်နိုင်သည်

`freq = pd.offsets.MonthEnd(3)` သည် (၃)လ တစ်ခါဖြစ်သည်။

```
pd.date_range(start='1/1/2018', periods=5,
              freq=pd.offsets.MonthEnd(3))
```

```
DatetimeIndex(['2018-01-31', '2018-04-30', '2018-07-31', '2018-10-31',
```

```
               '2019-01-31'], dtype='datetime64[ns]', freq='3M')
```

B သည် business day frequency ဖြစ်သည်။ ရုံးဖွင့်ရက်တိုင်း (စနေ, တနင်္ဂနွေ မပါ) **`freq = pd.offsets.BDay()`**)

```
pd.date_range(start='1/1/2018', end='1/10/2018', freq=pd.offsets.BDay())
```

```
DatetimeIndex(['2018-01-01', '2018-01-02', '2018-01-03', '2018-01-04',
               '2018-01-05', '2018-01-08', '2018-01-09', '2018-01-10'],
              dtype='datetime64[ns]', freq='B')
```

မိမိ လိုချင်သည့် timezone ကို လည်း **`tz`** ဖြင့် သတ်မှတ်ပေး(specify လုပ်) နိုင်သည်။

```
pd.date_range(start='1/1/2018', periods=5, tz='Asia/Singapore')
```

```
DatetimeIndex(['2018-01-01 00:00:00+08:00', '2018-01-02 00:00:00+08:00',
               '2018-01-03 00:00:00+08:00', '2018-01-04 00:00:00+08:00',
               '2018-01-05 00:00:00+08:00'],
              dtype='datetime64[ns, Asia/Singapore]', freq='D')
```

`closed` ဖြင့် အစဆုံးရက် နှင့် နောက်ဆုံးရက် ပါ, မပါ control လုပ်နိုင်သည်။ ဘာမှ မပြော ထားလျှင်(default) အစဆုံးရက်နှင့် နောက်ဆုံးရက်ကို ထည့်ထားသည်။ (The default includes boundary points on either end.)

```
pd.date_range(start='2017-01-01', end='2017-01-04', closed=None)
```

```
DatetimeIndex(['2017-01-01', '2017-01-02', '2017-01-03',
               '2017-01-04'], dtype='datetime64[ns]', freq='D')
```

`closed='left'` သည် **နောက်ဆုံး**ရက်ကို ချန်ထားသည်။ ထည့် မ,တွက်ပါ။ (to exclude **end** if it falls on the boundary.)

```
pd.date_range(start='2017-01-01', end='2017-01-04', closed='left')
```

```
DatetimeIndex(['2017-01-01', '2017-01-02', '2017-01-03'],
              dtype='datetime64[ns]', freq='D')
```

`closed='right'` ဟု ရေးလျှင် **အစဆုံး** ရက်ကို ချန်ထားသည်။ ထည့်မတွက်ပါ။ (to exclude **start** if it falls on the boundary.)

```
pd.date_range(start='2017-01-01', end='2017-01-04', closed='right')
```

```
DatetimeIndex(['2017-01-02', '2017-01-03', '2017-01-04'],
                           dtype='datetime64[ns]', freq='D')
```

ရုံးဖွင့်ရက်များသာ (Only want business days)

```
pd.period_range('2016-01-01 10:10', freq = 'B', periods = 10)
```

```
PeriodIndex(['2016-01-01', '2016-01-04', '2016-01-05', '2016-01-06',
             '2016-01-07', '2016-01-08', '2016-01-11', '2016-01-12',
             '2016-01-13', '2016-01-14'], dtype='period[B]', freq='B')
```

Frequency များကို ပေါင်းရေး နိုင်သည်။ ဥပမာ- (၂၅)နာရီကို (၁)ရက်နှင့် (၁)နာရီ ဟု ရေးနိုင်သည်။ (What if you want to advance by 25 hours each day.) `freq = '25H'` သည် (၂၅)နာရီ ဖြစ်သည်။

```
p1 = pd.period_range('2016-01-01 10:10', freq = '25H', periods = 10)
p1
```

```
PeriodIndex(['2016-01-01 10:00', '2016-01-02 11:00', '2016-01-03 12:00',
             '2016-01-04 13:00', '2016-01-05 14:00', '2016-01-06 15:00',
             '2016-01-07 16:00', '2016-01-08 17:00', '2016-01-09 18:00',
             '2016-01-10 19:00'], dtype='period[25H]', freq='25H')
```

```
p2 = pd.period_range('2016-01-01 10:10', freq ='1D1H', periods = 10)
p2
```

```
PeriodIndex(['2016-01-01 10:00', '2016-01-02 11:00',
             '2016-01-03 12:00', '2016-01-04 13:00',
             '2016-01-05 14:00', '2016-01-06 15:00',
             '2016-01-07 16:00', '2016-01-08 17:00',
             '2016-01-09 18:00', '2016-01-10 19:00'],
             dtype='period[25H]', freq='25H')
```

Time object များကိုလည်း indexing လုပ်နိုင်သည်။ Date range ကို သုံးထားသည်။

```
rng = pd.date_range('2020 Jul 1', periods = 10, freq = 'D')
rng
pd.Series(range(len(rng)), index = rng)
```

```
DatetimeIndex(['2020-07-01', '2020-07-02', '2020-07-03',
               '2020-07-04','2020-07-05', '2020-07-06',
               '2020-07-07', '2020-07-08','2020-07-09',
               '2020-07-10'],dtype='datetime64[ns]', freq='D')
2020-07-01    0
2020-07-02    1
2020-07-03    2
2020-07-04    3
2020-07-05    4
2020-07-06    5
2020-07-07    6
2020-07-08    7
2020-07-09    8
```

```
2020-07-10    9
Freq: D, dtype: int64
```

23.17.4 '07-10-16 8:00' မှ စ၍ (၁)နာရီခြား (၁၀) ကြိမ်

```
ts = pd.Series(range(10), pd.date_range('07-10-16 8:00',
                                        periods = 10, freq = 'H'))
ts
```

```
2016-07-10 08:00:00    0
2016-07-10 09:00:00    1
2016-07-10 10:00:00    2
2016-07-10 11:00:00    3
2016-07-10 12:00:00    4
2016-07-10 13:00:00    5
2016-07-10 14:00:00    6
2016-07-10 15:00:00    7
2016-07-10 16:00:00    8
2016-07-10 17:00:00    9
Freq: H, dtype: int64
```

```
ts_period = ts.to_period()
ts_period
```

```
2016-07-10 08:00    0
2016-07-10 09:00    1
2016-07-10 10:00    2
2016-07-10 11:00    3
2016-07-10 12:00    4
2016-07-10 13:00    5
2016-07-10 14:00    6
2016-07-10 15:00    7
2016-07-10 16:00    8
2016-07-10 17:00    9
Freq: H, dtype: int64
```

A number of string aliases are given to useful common time series frequencies. We will refer to these aliases as offset aliases.

Alias Description

B = business day frequency

C = custom business day frequency

D = calendar day frequency

W =weekly frequency

M = month end frequency

SM =semi-month end frequency (15th and end of month)

BM =business month end frequency

CBM= custom business month end frequency

MS= month start frequency

SMS= semi-month start frequency (1st and 15th)

BMS= business month start frequency

CBMS= custom business month start frequency

Q = quarter end frequency

BQ = business quarter end frequency

QS = quarter start frequency

BQS =business quarter start frequency

A, Y = year end frequency

BA, BY = business year end frequency

AS, YS = year start frequency

BAS, BYS = business year start frequency

BH= business hour frequency

H =hourly frequency

T, min = minutely frequency

S = secondly frequency

L, ms =milliseconds

U, us = microseconds

N = nanoseconds

```
import calendar as cal
pd.date_range(start='7/1/2012', end='7/10/2012',
                freq=pd.offsets.CDay(calendar=cal)).to_pydatetime()
```

```
array([datetime.datetime(2012, 7, 2, 0, 0),
       datetime.datetime(2012, 7, 3, 0, 0),
       datetime.datetime(2012, 7, 4, 0, 0),
       datetime.datetime(2012, 7, 5, 0, 0),
       datetime.datetime(2012, 7, 6, 0, 0),
       datetime.datetime(2012, 7, 9, 0, 0),
       datetime.datetime(2012, 7, 10, 0, 0)], dtype=object)
```

-**End**-