Python programming language မှ pandas ဥပမာများ

Python programming language မှ Data များအတွက် အထူးပြုလုပ်ထားသည် pandas package မှ syntax များကို ဥပမာများ ဖြင့် တစ်ကြောင်းချင်း ရှင်းပြထားသည်။

မိမိစက်တွင် install လုပ်ထားသည့် pandas version ကို သိလိုလျှင်

In [1]:
```
import pandas as pd
print(pd.__version__)
```

```
0.24.2
```

pandas သုံးရန်အတွက် လိုအပ်သည့် အရာများ (dependencies) အားလုံးကို ဖတ်ရန် print လုပ်လိုလျှင်

In [2]:
```
# Print all pandas dependencies
print(pd.show_versions(as_json=True))
```

```
{'system': {'commit': None, 'python': '3.7.3.final.0', 'python-bit
s': 64, 'OS': 'Windows', 'OS-release': '7', 'machine': 'AMD64', 'p
rocessor': 'Intel64 Family 6 Model 78 Stepping 3, GenuineIntel', '
byteorder': 'little', 'LC_ALL': 'None', 'LANG': 'None', 'LOCALE':
'None.None'}, 'dependencies': {'pandas': '0.24.2', 'pytest': '4.3.
1', 'pip': '19.0.3', 'setuptools': '40.8.0', 'Cython': '0.29.6', '
numpy': '1.16.2', 'scipy': '1.2.1', 'pyarrow': None, 'xarray': Non
e, 'IPython': '7.4.0', 'sphinx': '1.8.5', 'patsy': '0.5.1', 'dateu
til': '2.8.0', 'pytz': '2018.9', 'blosc': None, 'bottleneck': '1.
2.1', 'tables': '3.5.1', 'numexpr': '2.6.9', 'feather': None, 'mat
plotlib': '3.0.3', 'openpyxl': '2.6.1', 'xlrd': '1.2.0', 'xlwt': '
1.3.0', 'xlsxwriter': '1.1.5', 'lxml.etree': '4.3.2', 'bs4': '4.7.
1', 'html5lib': '1.0.1', 'sqlalchemy': '1.3.1', 'pymysql': None, '
psycopg2': None, 'jinja2': '2.10', 's3fs': None, 'fastparquet': No
ne, 'pandas_gbq': None, 'pandas_datareader': None, 'gcsfs': None}}
None
```

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

list တစ်ခုမှ ၊ numpy array တစ်ခုမှ သို့မဟုတ် dict တစ်ခုမှ pandas series တစ်ခု ပြုလုပ်လိုလျှင်

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

`a_list = list("abcdefg")` ဖြင့် *list* တစ်ခုတည်ဆောက်သည်။
`numpy_array = np.arange(1, 10)` ဖြင့် *numpy array* တစ်ခုတည်ဆောက်သည်။
`dictionary = {"A": 0, "B":1, "C":2, "D":3, "E":5}` ဖြင့် *dict* တစ်ခုတည်ဆောက်သည်။

```
In [4]: a_list
Out[4]: ['a', 'b', 'c', 'd', 'e', 'f', 'g']

In [5]: numpy_array
Out[5]: array([1, 2, 3, 4, 5, 6, 7, 8, 9])

In [6]: dictionary
Out[6]: {'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 5}
```

`pd.Series(a_list)` ဖြင့် list ကို pandas series တစ်ခု အဖြစ် ပြောင်းသည်။

```
In [7]: series1 = pd.Series(a_list)
        print(series1)

0    a
1    b
2    c
3    d
4    e
5    f
6    g
dtype: object
```

`pd.Series(numpy_array)` ဖြင့် numpy_array ကို pandas series တစ်ခု အဖြစ် ပြောင်းသည်။

```
In [8]: series2 = pd.Series(numpy_array)
        print(series2)

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

pd.Series(dictionary) ဖြင့် dictionary ကို pandas series တစ်ခု အဖြစ် ပြောင်းသည်။

In [9]:
```
series3 = pd.Series(dictionary)
print(series3)
```

```
A    0
B    1
C    2
D    3
E    5
dtype: int64
```

series တစ်ခု၏ index များကို (index of a series) ကို dataframe၏ ကော်လံတစ်ခုအဖြစ် ပြောင်းလဲလိုလျှင်

(To convert the index of a series into a column of a dataframe)

ပထမ ဦးစွာ a မှ z အထိ အင်္ဂလိပ် အက္ခရာများ ပါဝင်သည့် list တစ်ခု ပြုလုပ်သည်။
ဒုတိယ 0 မှ 25 အထိအစဉ်လိုက်ကိန်းများ ပါဝင်သည့် numpy array တစ်ခုတည်ဆောက်သည်။
အင်္ဂလိပ် အက္ခရာများ ပါဝင်သည့် list နှင့် 0 မှ 25 အထိအစဉ်လိုက်ကိန်းများ ပါဝင်သည့် numpy array ကို zip() ဖြင့် တွဲပြီး dictionary တစ်ခုတည်ဆောက်သည်။ ထို dictionary ကို pandas series တစ်ခု အဖြစ် ပြောင်းသည်။

In [10]:
```
# input
mylist = list('abcedfghijklmnopqrstuvwxyz')
myarr = np.arange(26)
mydict = dict(zip(mylist, myarr))
ser = pd.Series(mydict)
print(ser[:5])
```

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

နည်း(၂)မျိုးဖြင့် ပြုလုပ်နိုင်သည်။ ပထမနည်း DataFrame ကို သုံးသည့်နည်း

pandas series ကို DataFrame အဖြစ် ပြောင်းသည်။ ထို့နောက် index ကို reset လုပ်သည်။
`.reset_index()` မလုပ်ခင် ser_df ကို print ဖြင့် ကြည့်ပါ။

In [11]:
```
# solution 1 using DataFrame
ser_df = pd.DataFrame(ser)
ser_df.reset_index()
```

Out[11]:

	index	0
0	a	0
1	b	1
2	c	2
3	e	3
4	d	4
5	f	5
6	g	6
7	h	7
8	i	8
9	j	9
10	k	10
11	l	11
12	m	12
13	n	13
14	o	14
15	p	15
16	q	16
17	r	17
18	s	18
19	t	19
20	u	20
21	v	21
22	w	22
23	x	23
24	y	24
25	z	25

ဒုတိယနည်း: ser ကို `to_frame()` ဖြင့် DataFrame အဖြစ်ပြောင်းသည်။ ထိုနောက် `.reset_index()` ဖြင့် index ကို reset လုပ်သည်။

In [12]:
```
# using pandas to_frame()
ser_df = ser.to_frame().reset_index()
ser_df
```

Out[12]:

	index	0
0	a	0
1	b	1
2	c	2
3	e	3
4	d	4
5	f	5
6	g	6
7	h	7
8	i	8
9	j	9
10	k	10
11	l	11
12	m	12
13	n	13
14	o	14
15	p	15
16	q	16
17	r	17
18	s	18
19	t	19
20	u	20
21	v	21
22	w	22
23	x	23
24	y	24
25	z	25

series များကို ပေါင်း၍ DataFrame တစ်ခုအဖြစ်ပြောင်းလဲခြင်း(to combine many series to form a dataframe)

ser1 နှင့် ser2 ကို dataframe တစ်ခု အဖြစ်ပြောင်းလဲပုံကို ဖော်ပြထားသည်။ ser1 နှင့် ser2 တည်ဆောက်ပုံကို အထက်တွင် ဖော်ပြပြီးဖြစ်သည်။

In [13]:
```
# input
ser1 = pd.Series(list('abcedfghijklmnopqrstuvwxyz'))
ser2 = pd.Series(np.arange(26))
```

ပထမနည်း- DataFrame ကို သုံးသည့်နည်း ser1 နှင့် ser2ကို DataFrame အဖြစ် ပြောင်းသည်။ ထို့နောက် index ကို reset လုပ်သည်။

In [14]:
```
# using pandas DataFrame
ser_df = pd.DataFrame(ser1, ser2).reset_index()
ser_df.head()
```

Out[14]:

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

ဒုတိယနည်း dictionary တစ်ခုအဖြစ် ပြုလုပ်ပြီးမှ DataFrame သို့ ပြောင်းသည့်နည်း။
dictionaryတစ်ခုအဖြစ် တည်ဆောက်ချိန်တွင် မိမိကြိုက်သည့် ကော်လံ နာမည်များကို ပေးနိုင်သည်။

In [15]:
```
# using pandas DataFrame with a dictionary, gives a specific name to
the column
ser_df = pd.DataFrame({"col1":ser1, "col2":ser2})
ser_df.head(5)
```

Out[15]:

	col1	col2
0	a	0
1	b	1
2	c	2
3	e	3
4	d	4

တတိယနည်း - Pandas series နှစ်ခုကို concatenate လုပ်၍ dataframe အဖြစ်ပြောင်းသည့်နည်း။

In [16]:
```
# using pandas concat
ser_df = pd.concat([ser1, ser2], axis = 1)
ser_df.head()
```

Out[16]:

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

series တစ်ခုအား နာမည် assign လုပ်လိုလျှင် (To assign name to the series' index)

Give a name to the series ser calling it 'alphabets'.

In [17]:
```
# input
ser = pd.Series(list('abcedfghijklmnopqrstuvwxyz'))
```

`ser.rename("alphabets")` ဖြင့် နာမည်ပြောင်းသည်။

In [18]:
```
# using series rename method
ser.rename("alphabets")
```

Out[18]:
```
0     a
1     b
2     c
3     e
4     d
5     f
6     g
7     h
8     i
9     j
10    k
11    l
12    m
13    n
14    o
15    p
16    q
17    r
18    s
19    t
20    u
21    v
22    w
23    x
24    y
25    z
Name: alphabets, dtype: object
```

series attribute ကို အသုံးပြု၍ တခြားနာမည်တစ်ခုကို assign လုပ်သည်။

In [19]:
```
# using series attribute
ser.name = "other_name"
ser
```

Out[19]:
```
0     a
1     b
2     c
3     e
4     d
5     f
6     g
7     h
8     i
9     j
10    k
11    l
12    m
13    n
14    o
15    p
16    q
17    r
18    s
19    t
20    u
21    v
22    w
23    x
24    y
25    z
Name: other_name, dtype: object
```

ser2 ထဲတွင် မပါဝင်သည့် ser1 ထဲမှ item များ ကို ရွေးချယ်လိုလျှင် to get the items of series A not present in series B

In [20]:
```
# input
ser1 = pd.Series([1, 2, 3, 4, 5])
ser2 = pd.Series([4, 5, 6, 7, 8])
```

In [21]:
```
ser1[~ser1.isin(ser2)]
```

Out[21]:
```
0    1
1    2
2    3
dtype: int64
```

ser1 နှင့် ser2 နှစ်ခုစလုံးတွင် ဘုံအဖြစ် မပါဝင်သည့် ကိန်းများကို သိလိုလျှင်

to get the items not common to both series A and series B? ser1 နှင့် ser2 တို့တွင် ရှိသည့် item များ အနက် နှစ်ခုစလုံးတွင် ဘုံအဖြစ် မပါဝင်သည့် ser1 မှ item များ အားလုံးနှင့် ser2 မှ item များ အားလုံးကို ဖော်ပြရန် (Get all items of ser1 and ser2 not common to both)

In [22]:
```
# input
ser1 = pd.Series([1, 2, 3, 4, 5])
ser2 = pd.Series([4, 5, 6, 7, 8])
```

ပထမနည်း- pandas နည်း

In [23]:
```
# using pandas
a_not_b = ser1[~ser1.isin(ser2)]
b_not_a = ser2[~ser2.isin(ser1)]

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

Out[23]:
```
0    1
1    2
2    3
3    6
4    7
5    8
dtype: int64
```

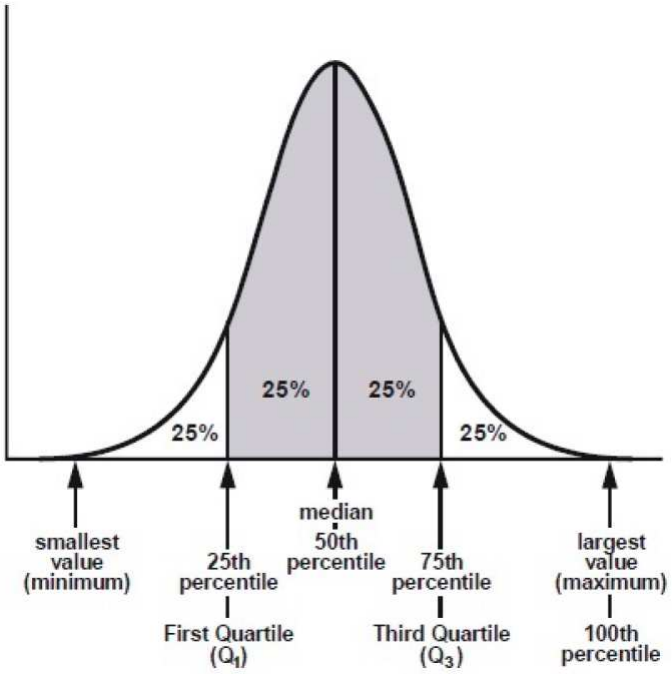
ဒုတိယနည်း - numpy union and intersection

In [24]:
```
# 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)]
```

Out[24]:
```
0    1
1    2
2    3
5    6
6    7
7    8
dtype: int64
```

numeric series တစ်ခုမှ minimum, 25th percentile, median, 75th, and max ရှာဖွေနည်း

(to get the minimum, 25th percentile, median, 75th, and max of a numeric series)




```
In [25]: # input
state = np.random.RandomState(100)
ser = pd.Series(state.normal(10, 5, 25))
```

ပထမနည်း - pandas နည်း

```
In [26]: # using pandas
ser.describe()
```

```
Out[26]: count    25.000000
mean     10.435437
std       4.253118
min       1.251173
25%       7.709865
50%      10.922593
75%      13.363604
max      18.094908
dtype: float64
```

ဒုတိယနည်း - numpy နည်း ၏ အားသာချက်မှာ မိမိကြိုက်နှစ်သက်သည့် percentile ကို ဖော်ပြနိုင်သည်။

```
In [27]: # or using numpy
np.percentile(ser, q = [0, 25, 50, 75, 100])
```

```
Out[27]: array([ 1.25117263,  7.70986507, 10.92259345, 13.36360403, 18.0949
083 ])
```

series ထဲတွင် ပါဝင်နေသည့် ကိန်းတစ်လုံးစီ၏ ပါဝင်သည့် အကြိမ် အရေအတွက်ကို သိလိုလျှင်(to get frequency counts of unique items of a series)

Calculate the frequency counts of each unique value ser.

In [28]:
```
# input
ser = pd.Series(np.take(list('abcdefgh'), np.random.randint(8, size=30)))
ser
```

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

.value_counts() ကို အသုံးပြု၍ ဖော်ပြနိုင်သည်။

In [29]:
```
ser.value_counts()
```

Out[29]:
```
f    6
h    4
c    4
b    4
a    4
g    3
d    3
e    2
dtype: int64
```

အများဆုံးပါဝင်သည့် ကိန်းနှစ်လုံးကိုသာ ဖော်ပြပြီး ကျန်ကိန်းများကို 'Other' ဆိုသည့် စကားလုံးဖြင့် အစားထိုး လိုလျှင်

To keep only top 2 most frequent values as it is and replace everything else as 'Other. From ser, keep the top 2 most frequent items as it is and replace everything else as 'Other'.

In [30]:
```
# input
np.random.RandomState(100)
ser = pd.Series(np.random.randint(1, 5, [12]))
ser
```

Out[30]:
```
0     3
1     3
2     1
3     4
4     2
5     3
6     2
7     2
8     3
9     4
10    1
11    3
dtype: int32
```

ကိန်းတစ်လုံးစီ၏ အကြိမ်အ ရေအတွက် မည်မျှ ပါသည်ကို သိရန် value_counts ကိုသုံးပြီး အများဆုံးကိန်း ၂လုံးကိုယူသည်။ ~ser.isin ကို သုံး၍ filter လုပ်သည်။ ကျန်ကိန်းများကို Other အဖြစ် assign လုပ်သည်။

In [31]:
```
ser.value_counts()
ser[~ser.isin(ser.value_counts().index[:2])] = 'Other'
ser
```

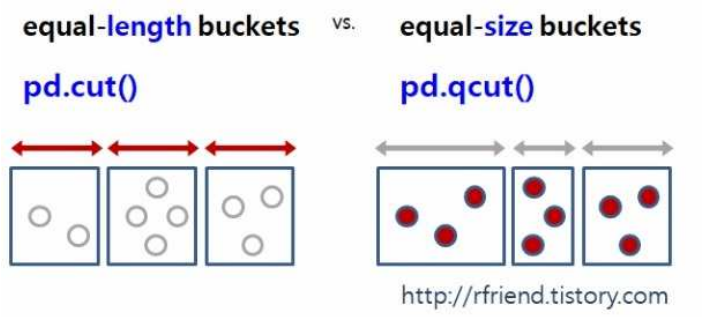
Out[31]:
```
0          3
1          3
2      Other
3      Other
4          2
5          3
6          2
7          2
8          3
9      Other
10     Other
11         3
dtype: object
```

numeric series တစ်ခုအတွင်းရှိ ကိန်းများကို အညီအမျှ (၁၀)စုခွဲလိုလျှင်

numeric series တစ်ခုအတွင်းရှိ ကိန်းများကို အညီအမျှ (၁၀)စုခွဲပြီး အစုနာမည်ပေးသည်။
Bin the series ser into 10 equal deciles and replace the values with the bin name.

In [32]:
```
# input
ser = pd.Series(np.random.random(20))
ser
```

Out[32]:
```
0     0.163777
1     0.677096
2     0.818741
3     0.288003
4     0.206756
5     0.256575
6     0.028243
7     0.038997
8     0.450959
9     0.569541
10    0.213448
11    0.749240
12    0.988077
13    0.665560
14    0.939130
15    0.792236
16    0.336676
17    0.971509
18    0.207075
19    0.927101
dtype: float64
```



In [33]:
```
pd.qcut(ser, q = 10)
# we can also pass labels
q = [0, .10, .20, .30, .40, .50, .60, .70, .80, .90, 1]
labels=['1st', '2nd', '3rd', '4th', '5th', '6th', '7th', '8th', '9th
', '10th']
pd.qcut(ser, q , labels)
```

Out[33]:
```
0      2nd
1      7th
2      8th
3      4th
4      2nd
5      4th
6      1st
7      1st
8      5th
9      6th
10     3rd
11     7th
12    10th
13     6th
14     9th
15     8th
16     5th
17    10th
18     3rd
19     9th
dtype: category
Categories (10, object): [1st < 2nd < 3rd < 4th ... 7th < 8th < 9t
h < 10th]
```

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

In [34]:
```
# input
ser = pd.Series(np.random.randint(1, 10, 35))
ser
```

Out[34]:
```
0     9
1     2
2     4
3     1
4     7
5     7
6     6
7     4
8     9
9     3
10    8
11    1
12    2
13    3
14    6
15    6
16    4
17    4
18    1
19    8
20    7
21    5
22    7
23    9
24    3
25    5
26    4
27    5
28    8
29    8
30    7
31    9
32    9
33    3
34    1
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
numpy နည်း

In [35]:
```
# using numpy
pd.DataFrame(np.array(ser).reshape(7, 5))
```

Out[35]:

	0	1	2	3	4
0	9	2	4	1	7
1	7	6	4	9	3
2	8	1	2	3	6
3	6	4	4	1	8
4	7	5	7	9	3
5	5	4	5	8	8
6	7	9	9	3	1

pandas နည်း

series တစ်ခုထဲမှာ မြှောက်ဖော်ကိန်း (၃)ဖြစ်သည့် item ၏ နေရာကို သိလိုလျှင်

(to find the positions of numbers that are multiples of 3 from a series)

In [36]:
```
# input
np.random.RandomState(100)
ser = pd.Series(np.random.randint(1, 5, 10))
ser
```

Out[36]:
```
0    2
1    2
2    4
3    2
4    2
5    2
6    4
7    3
8    1
9    2
dtype: int32
```

In [37]:
```
# using the where clause
ser.where(lambda x: x%3 == 0).dropna()
```

Out[37]:
```
7    3.0
dtype: float64
```

In [38]:
```
# using numpy and reshape to get a pandas series
#pd.Series(np.argwhere(ser%3 == 0).reshape(4))
np.argwhere(ser%3 == 0)
```

```
C:\Users\cnyuntk\AppData\Local\Continuum\anaconda3\lib\site-packag
es\numpy\core\fromnumeric.py:56: FutureWarning: Series.nonzero() i
s deprecated and will be removed in a future version.Use Series.to
_numpy().nonzero() instead
  return getattr(obj, method)(*args, **kwds)
```

Out[38]:
```
array([[7]], dtype=int64)
```

series တစ်ခု ၏ သတ်မှတ်ထားသည့်နေရာမှ item များကို ထုတ်ယူလိုလျှင်

(to extract items at given positions from a series) ser မှ နေရာများပါဝင်သည့် list တစ်ခုဖြင့် item များကို ထုတ်ယူသည်။ Access လုပ်သည်။ (From ser, extract the items at positions in list pos.)

In [39]:
```
# input
ser = pd.Series(list('abcdefghijklmnopqrstuvwxyz'))
pos = [0, 4, 8, 14, 20]
```

loc(location) ကို သုံး၍ ထုတ်ယူသည်။ Access လုပ်သည်။

In [40]:
```
# using loc
ser.loc[pos]
```

Out[40]:
```
0     a
4     e
8     i
14    o
20    u
dtype: object
```

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

In [41]:
```
# using series take
ser.take(pos)
```

Out[41]:
```
0     a
4     e
8     i
14    o
20    u
dtype: object
```

series နှစ်ခုကို ဒေါင်လိုက်(vertically) ထပ်၍ dataframe ပြုလုပ်လိုလျှင် ၊ အလျားလိုက်(horizontally)ထပ်၍ dataframe ပြုလုပ်လိုလျှင်

In [42]:
```
# input
ser1 = pd.Series(range(5))
ser2 = pd.Series(list('abcde'))
```

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

```
In [43]: # vertical
         ser1.append(ser2)
```

```
Out[43]: 0    0
         1    1
         2    2
         3    3
         4    4
         0    a
         1    b
         2    c
         3    d
         4    e
         dtype: object
```

ser1 နှင့် ser2 ကို concat ပြီး ပေါင်းသည်။ ser1 နှင့် ser2 ကို concat ပြီး ပေါင်းသည်။ concat လုပ်သည့်အခါ ပေါင်းလိုသည့် axis ကို ပြောပေးရသည်။ မသတ်မှတ်ပေးလျှင် default axis = 0 ဖြစ်သည်။ axis = 0 သည် ဒေါင်လိုက် ဖြစ်အောင်ပေါင်းသည်။
axis = 0 ဒေါင်လိုက် ဖြစ်အောင်ပေါင်းသည် ဆိုသည်မှာ data frame တွင် row အသစ်များ ထပ်ထိုးလာအောင် ပေါင်းသည်။

```
In [44]: # or using pandas concat and axis = 0
         pd.concat([ser1, ser2], axis = 0)
```

```
Out[44]: 0    0
         1    1
         2    2
         3    3
         4    4
         0    a
         1    b
         2    c
         3    d
         4    e
         dtype: object
```

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

In [45]:
```
# horizontal
pd.concat([ser1, ser2], axis = 1)
```

Out[45]:

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

ser1 နှင့် ser2 နှစ်ခုရှိတဲ့ အနက် ser2 ထဲ တွင် ပါဝင်သည့် ser1 item များ၏ တည်ရှိရာနေရာကို သိလိုလျှင်

to get the positions of items of series ser1 in another series ser

In [46]:
```
# input
ser1 = pd.Series([10, 9, 6, 5, 3, 1, 12, 8, 13])
ser2 = pd.Series([1, 3, 10, 13, 7, 4])
```

ပထမ ser2 ထဲမှာ ပါတဲ့ ser1 item များ

In [47]:
```
ser1[ser1.isin(ser2)]
```

Out[47]:
```
0    10
4     3
5     1
8    13
dtype: int64
```

ser2 ထဲမှာ ပါတဲ့ ser1 item တွေရဲ့ တည်ရှိရာနေရာကို သိလိုလျှင်

In [48]:
```
# get's the index, but it's sorts the index
list(ser1[ser1.isin(ser2)].index)
```

Out[48]:
```
[0, 4, 5, 8]
```

In [49]:
```
# input
truth = pd.Series(range(10))
pred = pd.Series(range(10)) + np.random.random(10)
```

ပထမနည်း- numpy နည်း

In [50]:
```
# using numpy
np.mean((truth-pred)**2)
```

Out[50]:
```
0.46142319402234194
```

ဒုတိယနည်း - sklear metrics နည်း

```
In [51]: # using sklear metrics
         from sklearn.metrics import mean_squared_error
         mean_squared_error(truth, pred)
```

```
Out[51]: 0.46142319402234194
```

စာလုံးတစ်လုံးမှ ပထမဆုံး အက္ခရာကို အကြီးစာလုံးအဖြစ် ပြောင်းလိုလျှင်

Change the first character of each word to upper case in each word of ser.

```
In [52]: # input
         ser = pd.Series(['just', 'a', 'random', 'list'])
         ser
```

```
Out[52]: 0       just
         1          a
         2     random
         3       list
         dtype: object
```

ပထမနည်း python string method

```
In [53]: # using python string method title() Assumes we only encounter strin
         g in the list
         [i.title() for i in ser]
```

```
Out[53]: ['Just', 'A', 'Random', 'List']
```

ဒုတိယနည်း - lambda နည်း

```
In [54]: # using lambda
         ser.map(lambda x: x.title())
```

```
Out[54]: 0       Just
         1          A
         2     Random
         3       List
         dtype: object
```

တတိယနည်း

```
In [55]: # other solution
ser.map(lambda x: x[0].upper() + x[1:])
```

```
Out[55]: 0       Just
1          A
2     Random
3       List
dtype: object
```

စာလုံးတစ်လုံးတွင် ပါဝင်သည့် အက္ခရာ အရေအတွက်ကို သိလိုလျှင်

to calculate the number of characters in each word in a series?

```
In [56]: # input
ser = pd.Series(['just', 'a', 'random', 'list'])
```

ပထမနည်း

```
In [57]: # using list comprehension
[len(i) for i in ser]
```

```
Out[57]: [4, 1, 6, 4]
```

ဒုတိယနည်း

```
In [58]: # using series map
ser.map(len)
```

```
Out[58]: 0    4
1    1
2    6
3    4
dtype: int64
```

တတိယနည်း

```
In [59]: # using series apply
ser.apply(len)
```

```
Out[59]: 0    4
1    1
2    6
3    4
dtype: int64
```

အနီးစပ်ဆုံးကိန်း(၂)လုံး၏ ခြားနားချက်။ ထိုခြားနားချက်ကိန်း၏ နောက်ထပ် ခြားနားချက်

Difference of differences between the consequtive numbers of ser.

In [60]:
```
# input
ser = pd.Series([1, 3, 6, 10, 15, 21, 27, 35])

# Desired Output
# [nan, 2.0, 3.0, 4.0, 5.0, 6.0, 6.0, 8.0]
# [nan, nan, 1.0, 1.0, 1.0, 1.0, 0.0, 2.0]
```

In [61]:
```
# using pandas diff()
ser.diff(periods = 1).tolist()
ser.diff(periods = 1).diff(periods = 1).tolist()
```

Out[61]:
```
[nan, nan, 1.0, 1.0, 1.0, 1.0, 0.0, 2.0]
```

series of date-strings တစ်ခုကို timeseries အဖြစ်သို့ ပြောင်းလဲလိုလျှင်

to convert a series of date-strings to a timeseries

In [62]:
```
# input
ser = pd.Series(['01 Jan 2010', '02-02-2011',
                 '20120303', '2013/04/04', '2014-05-05', '2015-06-06
T12:20'])

# Desired Output
# 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
```

In [63]:
```
# using pands to_datetime
pd.to_datetime(ser)

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

Out[63]:
```
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]
```

series of date strings မှ day of month ၊ week number ၊ day of year နှင့် day of week စသည်တို့ ပြောင်းလိုလျှင်

to get the day of month, week number, day of year and day of week from a series of date strings
Get the day of month, week number, day of year and day of week from ser.

In [64]:
```
# input
ser = pd.Series(['01 Jan 2010', '02-02-2011', '20120303', '2013/04/0
4',
                 '2014-05-05', '2015-06-06T12:20'])

# Desired output
# Date:  [1, 2, 3, 4, 5, 6]
# Week number:  [53, 5, 9, 14, 19, 23]
# Day num of year:  [1, 33, 63, 94, 125, 157]
# Day of week:  ['Friday', 'Wednesday', 'Saturday', 'Thursday', 'Mon
day', 'Saturday']
```

In [65]:
```
# day
pd.to_datetime(ser).dt.day.to_list()
```

Out[65]:
```
[1, 2, 3, 4, 5, 6]
```

In [66]:
```
# week
pd.to_datetime(ser).dt.week.to_list()
```

Out[66]:
```
[53, 5, 9, 14, 19, 23]
```

In [67]:
```
# another method
```

In [68]:
```
# another method
pd.to_datetime(ser).dt.weekofyear.to_list()
```

Out[68]:
```
[53, 5, 9, 14, 19, 23]
```

In [69]:
```
# day of year
pd.to_datetime(ser).dt.dayofyear.to_list()
```

Out[69]:
```
[1, 33, 63, 94, 125, 157]
```

In [70]:
```
# 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()
```

Out[70]:
```
['Friday', 'Wednesday', 'Saturday', 'Thursday', 'Monday', 'Saturda
y']
```

In [71]:
```
# another method
pd.to_datetime(ser).dt.weekday_name.to_list()
```

Out[71]:
```
['Friday', 'Wednesday', 'Saturday', 'Thursday', 'Monday', 'Saturda
y']
```

1. How to convert year-month string to dates corresponding to the 4th day of the month?

Change ser to dates that start with 4th of the respective months.

In [72]:
```
# input
ser = pd.Series(['Jan 2010', 'Feb 2011', 'Mar 2012'])

'''
Desired Output

0   2010-01-04
1   2011-02-04
2   2012-03-04
dtype: datetime64[ns]

'''
```

Out[72]:
```
'\nDesired Output\n\n0   2010-01-04\n1   2011-02-04\n2   2012-03-0
4\ndtype: datetime64[ns]\n\n'
```

In [73]:
```
# solution using parser
from dateutil.parser import parse
ser.map(lambda x: parse('04 ' + x))

# another solution

from dateutil.parser import parse
# Parse the date
ser_ts = ser.map(lambda x: parse(x))

# Construct date string with date as 4
ser_datestr = ser_ts.dt.year.astype('str') + '-' + ser_ts.dt.month.a
stype('str') + '-' + '04'

# Format it.
[parse(i).strftime('%Y-%m-%d') for i in ser_datestr]
```

Out[73]:
```
['2010-01-04', '2011-02-04', '2012-03-04']
```

1. How to filter words that contain atleast 2 vowels from a series?

From ser, extract words that contain atleast 2 vowels.

In [74]:
```
# input
ser = pd.Series(['Apple', 'Orange', 'Plan', 'Python', 'Money'])

'''
Desired Output


0     Apple
1    Orange
4     Money
dtype: object
'''
```

Out[74]:
```
'\nDesired Output\n\n\n0     Apple\n1    Orange\n4     Money\ndtyp
e: object\n'
```

In [75]:
```
# using nested loops
vowels = list("aeiou")
list_ = []
for w in ser:
    c = 0
    for l in list(w.lower()):
        if l in vowels:
            c += 1
    if c >= 2:
        print(w)
        list_.append(w)

ser[ser.isin(list_)]

# another solution using counter

from collections import Counter
mask = ser.map(lambda x: sum([Counter(x.lower()).get(i, 0) for i in
list('aeiou')]) >= 2)
ser[mask]
```

```
Apple
Orange
Money
```

Out[75]:
```
0     Apple
1    Orange
4     Money
dtype: object
```

1. How to filter valid emails from a series?

Extract the valid emails from the series emails. The regex pattern for valid emails is provided as reference.

In [76]:
```
# input
emails = pd.Series(['buying books at amazom.com', 'rameses@egypt.com
', 'matt@t.co', 'narendra@modi.com'])
pattern ='[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\\.[A-Za-z]{2,4}'

'''
Desired Output

1    rameses@egypt.com
2            matt@t.co
3    narendra@modi.com
dtype: object
'''
```

Out[76]:
```
'\nDesired Output\n\n1    rameses@egypt.com\n2            matt@t.c
o\n3    narendra@modi.com\ndtype: object\n'
```

In [77]:
```
# using powerful regex
import re
re_ = re.compile(pattern)
emails[emails.str.contains(pat = re_, regex = True)]

# other solutions
pattern ='[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\\.[A-Za-z]{2,4}'
mask = emails.map(lambda x: bool(re.match(pattern, x)))
emails[mask]

# using str.findall
emails.str.findall(pattern, flags=re.IGNORECASE)

# using list comprehension
[x[0] for x in [re.findall(pattern, email) for email in emails] if l
en(x) > 0]
```

Out[77]:
```
['rameses@egypt.com', 'matt@t.co', 'narendra@modi.com']
```

grouped by series ဖြင့် series များကို ခွဲခြားပြီး mean တန်ဖိုးရှာခြင်း

to get the mean of a series grouped by another series
'apple'၊ 'banana'၊ 'carrot' စသည်တို့ကို grouped by series ဖြင့် series များကို ခွဲခြားပြီး တစ်ခုချင်းစီ၏ mean တန်ဖိုးရှာခြင်း

In [78]:
```
# doesn't incluide the upper limit
fruit = pd.Series(np.random.choice(['apple', 'banana', 'carrot'], 1
0))
fruit
weights = pd.Series(np.linspace(1, 10, 10))
weights
#print(weights.tolist())
#print(fruit.tolist())

'''
Desired output

# values can change due to randomness
apple     6.0
banana    4.0
carrot    5.8
dtype: float64
'''
```

Out[78]:
```
'\nDesired output\n\n# values can change due to randomness\napple
6.0\nbanana    4.0\ncarrot    5.8\ndtype: float64\n'
```

In [79]:
```
# using pandas groupby
df = pd.concat([fruit, weights], axis = 1)
df
df.groupby(0).mean()

# use one list to calculate a kpi from another
weights.groupby(fruit).mean()
```

Out[79]:
```
apple     3.0
banana    4.0
carrot    7.2
dtype: float64
```

series နှစ်ခု မှ euclidean distance တွက်ခြင်း:

to compute the euclidean distance between two series Compute the euclidean distance between series (points) p and q, without using a packaged formula.
series တစ်ခု တွင် point p တန်ဖိုးပါဝင်သည်။ နောက် series တစ်ခု တွင် point q တန်ဖိုးပါဝင်သည်။ point p နှင့် q တို့၏ euclidean distance ကို packaged များမှ formula ကို မသုံးဘဲ တွက်ခြင်း

In [80]:
```
# Input
p = pd.Series([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
q = pd.Series([10, 9, 8, 7, 6, 5, 4, 3, 2, 1])
'''
Desired Output

18.165
'''
```

Out[80]:
```
'\nDesired Output\n\n18.165\n'
```

In [81]:
```
# using list comprehension
suma = np.sqrt(np.sum([(p - q)**2 for p, q in zip(p, q)]))
suma

# using series one to one operation
sum((p - q)**2)**.5

# using numpy
np.linalg.norm(p-q)
```

Out[81]:
```
18.16590212458495
```

numeric series တစ်ခုမှာ local maxima (or peaks) များကို ရှာဖွေခြင်း:

to find all the local maxima (or peaks) in a numeric series?

In [82]:
```
# input
ser = pd.Series([2, 10, 3, 4, 9, 10, 2, 7, 3])

'''
Desired output

array([1, 5, 7])
'''
```

Out[82]:
```
'\nDesired output\n\narray([1, 5, 7])\n'
```

In [83]:
```
# using pandas shift
local_max = ser[(ser.shift(1) < ser) & (ser.shift(-1) < ser)]
local_max.index

# using numpy
dd = np.diff(np.sign(np.diff(ser)))
dd
peak_locs = np.where(dd == -2)[0] + 1
peak_locs
```

Out[83]:
```
array([1, 5, 7], dtype=int64)
```

string တစ်ခုအတွင်း ပျောက်နေသည့် (missing)spaces များအား အနည်းဆုံးပါဝင်သည့် အက္ခရာဖြင့် အစားထိုးခြင်း

to replace missing spaces in a string with the least frequent character? Replace the spaces in my_str with the least frequent character.

In [84]:
```
# input
my_str = 'dbc deb abed ggade'

'''
Desired Output

'dbccdebcabedcggade'  # least frequent is 'c'
'''
```

Out[84]:
```
"\nDesired Output\n\n'dbccdebcabedcggade'  # least frequent is 'c
'\n"
```

In [85]:
```
# using Counter
from collections import Counter
my_str_ = my_str
Counter_ = Counter(list(my_str_.replace(" ", "")))
Counter_
minimum = min(Counter_, key = Counter_.get)

print(my_str.replace(" ", minimum))

# using pandas
ser = pd.Series(list(my_str.replace(" ", "")))
ser.value_counts()
minimum = list(ser.value_counts().index)[-1]
minimum
print(my_str.replace(" ", minimum))
```

```
dbccdebcabedcggade
dbccdebcabedcggade
```

‘2000-01-01’ ရက်မှ စသည့် TimeSeries တစ်ခုဖန်တီးပြီး စနေနေ့များအတွက် random number များ သတ်မှတ်ပေးခြင်း:

to create a TimeSeries starting ‘2000-01-01’ and 10 weekends (saturdays) after that having random numbers as values

In [86]:
```
'''
Desired Output
values can be random

2000-01-01    4
2000-01-08    1
2000-01-15    8
2000-01-22    4
2000-01-29    4
2000-02-05    2
2000-02-12    4
2000-02-19    9
2000-02-26    6
2000-03-04    6
'''
```

Out[86]:
```
'\nDesired Output\nvalues can be random\n\n2000-01-01    4\n2000-0
1-08    1\n2000-01-15    8\n2000-01-22    4\n2000-01-29    4\n2000
-02-05    2\n2000-02-12    4\n2000-02-19    9\n2000-02-26    6\n20
00-03-04    6\n'
```

In [87]:
```
dti = pd.Series(pd.date_range('2000-01-01', periods=10, freq='W-SAT
'))
random_num = pd.Series([np.random.randint(1, 10) for i in range(1
0)])


df = pd.concat({"Time":dti, "Numbers":random_num}, axis = 1)
df

# for more about time series functionality
# https://pandas.pydata.org/pandas-docs/stable/user_guide/timeserie
s.html#timeseries-offset-aliases

# another solution just using pandas Series
ser = pd.Series(np.random.randint(1,10,10), pd.date_range('2000-01-0
1', periods=10, freq='W-SAT'))
ser
```

Out[87]:
```
2000-01-01    9
2000-01-08    3
2000-01-15    2
2000-01-22    8
2000-01-29    8
2000-02-05    5
2000-02-12    7
2000-02-19    4
2000-02-26    9
2000-03-04    7
Freq: W-SAT, dtype: int32
```