1. Appending & Concatenating Series

Pandas series is a One-dimensional ndarray with axis labels. The labels need not be unique but must be a hashable type. The object supports both integer- and label-based indexing and provides a host of methods for performing operations involving the index.

In [1]:
```
# importing pandas as pd
import pandas as pd

# Creating the first Series
sr1 = pd.Series(['New York', 'Chicago', 'Toronto', 'Lisbon', 'Rio'])

# Create the first Index
index_1 = ['City 1', 'City 2', 'City 3', 'City 4', 'City 5']

# set the index of first series
sr1.index = index_1

# Creating the second Series
sr2 = pd.Series(['Chicage', 'Shanghai', 'Beijing', 'Jakarta', 'Seoul'])

# Create the second Index
index_2 = ['City 6', 'City 7', 'City 8', 'City 9', 'City 10']

# set the index of second series
sr2.index = index_2

# Print the first series
print(sr1)

# Print the second series
print(sr2)
```

```
City 1     New York
City 2      Chicago
City 3      Toronto
City 4       Lisbon
City 5          Rio
dtype: object
City 6      Chicage
City 7     Shanghai
City 8      Beijing
City 9      Jakarta
City 10       Seoul
dtype: object
```

In [2]:
```
# append sr2 at the end of sr1
result = sr1.append(sr2)

# Print the result
print(result)
```

```
City 1      New York
City 2       Chicago
City 3       Toronto
City 4        Lisbon
City 5           Rio
City 6       Chicage
City 7      Shanghai
City 8       Beijing
City 9       Jakarta
City 10        Seoul
dtype: object
```

Use Series.append() function to append the passed series object at the end of this series object. Ignore the original index of the two series objects.

In [3]:
```
# importing pandas as pd
import pandas as pd

# Creating the first Series
sr1 = pd.Series(['New York', 'Chicago', 'Toronto', 'Lisbon', 'Rio'])

# Create the first Index
index_1 = ['City 1', 'City 2', 'City 3', 'City 4', 'City 5']

# set the index of first series
sr1.index = index_1

# Creating the second Series
sr2 = pd.Series(['Chicage', 'Shanghai', 'Beijing', 'Jakarta', 'Seoul'])

# Create the second Index
index_2 = ['City 6', 'City 7', 'City 8', 'City 9', 'City 10']

# set the index of second series
sr2.index = index_2

# Print the first series
print(sr1)

# Print the second series
print(sr2)
```

```
City 1     New York
City 2      Chicago
City 3      Toronto
City 4       Lisbon
City 5          Rio
dtype: object
City 6      Chicage
City 7     Shanghai
City 8      Beijing
City 9      Jakarta
City 10       Seoul
dtype: object
```

In [4]:
```
# append sr2 at the end of sr1
# ignore the index
result = sr1.append(sr2, ignore_index = True)

# Print the result
print(result)
```

```
0    New York
1     Chicago
2     Toronto
3      Lisbon
4         Rio
5     Chicage
6    Shanghai
7     Beijing
8     Jakarta
9       Seoul
dtype: object
```

concat series

In [5]:
```
s1 = pd.Series(['a', 'b'])
s2 = pd.Series(['c','d'])
```

In [6]:
```
pd.concat([s1, s2], ignore_index=True)
```

Out[6]:
```
0    a
1    b
2    c
3    d
dtype: object
```

In [7]:
```
pd.concat([s1, s2], keys=['s1', 's2',])
```

Out[7]:
```
s1  0    a
    1    b
s2  0    c
    1    d
dtype: object
```

In [8]:
```
pd.concat([s1, s2], keys=['s1', 's2'], names=['Series name', 'Row ID'])
```

Out[8]:
```
Series name  Row ID
s1           0         a
             1         b
s2           0         c
             1         d
dtype: object
```

2. Appending & Concatenating DataFrames

In [9]:
```
import numpy as np
import pandas as pd
```

In [10]:
```
# Python program to concatenate
# dataframes using Panda

# Creating first dataframe
df1 = pd.DataFrame({'A': ['A0', 'A1', 'A2', 'Nan'],
                    'B': ['B0', 'B1', 'B2', 'B3'],
                    'C': ['C0', 'C1', 'C2', 'C3'],
                    'D': ['D0', 'D1', 'D2', 'D3']},
                    index = [0, 1, 2, 3])

df1
```

Out[10]:

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	Nan	B3	C3	D3

In [11]:
```
# Importing pandas as pd
import pandas as pd

# Creating the first Dataframe using dictionary
df1 = df = pd.DataFrame({"a":[1, 2, 3, 4],
                         "b":[5, 6, 7, 8]})

# Creating the Second Dataframe using dictionary
df2 = pd.DataFrame({"a":[1, 2, 3],
                    "b":[5, 6, 7]})

# Print df1
print(df1, "\n")

# Print df2
df2
```

```
   a  b
0  1  5
1  2  6
2  3  7
3  4  8
```

Out[11]:

	a	b
0	1	5
1	2	6
2	3	7

In [12]:
```
# to append df2 at the end of df1 dataframe
df1.append(df2)
```

Out[12]:

	a	b
0	1	5
1	2	6
2	3	7
3	4	8
0	1	5
1	2	6
2	3	7

In [13]:
```
# A continuous index value will be maintained
# across the rows in the new appended data frame.
df1.append(df2, ignore_index = True)
```

Out[13]:

	a	b
0	1	5
1	2	6
2	3	7
3	4	8
4	1	5
5	2	6
6	3	7

In [14]:
```
# Importing pandas as pd
import pandas as pd

# Creating the first Dataframe using dictionary
df1 = pd.DataFrame({"a":[1, 2, 3, 4],
                    "b":[5, 6, 7, 8]})

# Creating the Second Dataframe using dictionary
df2 = pd.DataFrame({"a":[1, 2, 3],
                    "b":[5, 6, 7],
                    "c":[1, 5, 4]})

# for appending df2 at the end of df1
df1.append(df2, ignore_index = True)
```

Out[14]:

	a	b	c
0	1	5	NaN
1	2	6	NaN
2	3	7	NaN
3	4	8	NaN
4	1	5	1.0
5	2	6	5.0
6	3	7	4.0

3. Keys & MultiIndexes

In [15]:
```
import pandas, io
```

In [16]:
```
data = io.StringIO('''Fruit,Color,Count,Price
Apple,Red,3,$1.29
Apple,Green,9,$0.99
Pear,Red,25,$2.59
Pear,Green,26,$2.79
Lime,Green,99,$0.39
''')
df_unindexed = pandas.read_csv(data)
df_unindexed
```

Out[16]:

	Fruit	Color	Count	Price
0	Apple	Red	3	$1.29
1	Apple	Green	9	$0.99
2	Pear	Red	25	$2.59
3	Pear	Green	26	$2.79
4	Lime	Green	99	$0.39

Add a multi-index based on two columns

In [17]:
```
df = df_unindexed.set_index(['Fruit', 'Color'])
df
```

Out[17]:

Fruit	Color	Count	Price
Apple	Red	3	$1.29
	Green	9	$0.99
Pear	Red	25	$2.59
	Green	26	$2.79
Lime	Green	99	$0.39

In [18]:
```
import pandas as pd
import numpy as np
```

In [19]:
```
index = [('California', 2000), ('California', 2010),
         ('New York', 2000), ('New York', 2010),
         ('Texas', 2000), ('Texas', 2010)]
populations = [33871648, 37253956,
               18976457, 19378102,
               20851820, 25145561]
pop = pd.Series(populations, index=index)
pop
```

Out[19]:
```
(California, 2000)    33871648
(California, 2010)    37253956
(New York, 2000)      18976457
(New York, 2010)      19378102
(Texas, 2000)         20851820
(Texas, 2010)         25145561
dtype: int64
```

In [20]:
```
index = pd.MultiIndex.from_tuples(index)
index
```

Out[20]:
```
MultiIndex([('California', 2000),
            ('California', 2010),
            (  'New York', 2000),
            (  'New York', 2010),
            (     'Texas', 2000),
            (     'Texas', 2010)],
           )
```

In [21]:
```
pop = pop.reindex(index)
pop
```

Out[21]:
```
California  2000    33871648
            2010    37253956
New York    2000    18976457
            2010    19378102
Texas       2000    20851820
            2010    25145561
dtype: int64
```

In [22]:
```
pop[:, 2010]
```

Out[22]:
```
California    37253956
New York      19378102
Texas         25145561
dtype: int64
```

4. Merging DataFrames

In [23]:
```
import numpy as np
import pandas as pd
# Dataframe of number of sales made by an employee
sales = {'Tony': 103,
         'Sally': 202,
         'Randy': 380,
         'Ellen': 101,
         'Fred': 82
        }
# Dataframe of all employees and the region they work in
region = {'Tony': 'West',
          'Sally': 'South',
          'Carl': 'West',
          'Archie': 'North',
          'Randy': 'East',
          'Ellen': 'South',
          'Fred': np.nan,
          'Mo': 'East',
          'HanWei': np.nan,
         }
```

In [24]:
```
# Make dataframes
sales_df = pd.DataFrame.from_dict(sales, orient='index',
                                  columns=['sales'])
region_df = pd.DataFrame.from_dict(region, orient='index',
                                   columns=['region'])
```

In [25]:
```
joined_df_merge = region_df.merge(sales_df, how='left',
                                  left_index=True,
                                  right_index=True)
print(joined_df_merge)
```

```
       region  sales
Tony     West  103.0
Sally   South  202.0
Carl     West    NaN
Archie  North    NaN
Randy    East  380.0
Ellen   South  101.0
Fred      NaN   82.0
Mo       East    NaN
HanWei    NaN    NaN
```

In [26]:
```
grouped_df = joined_df_merge.groupby(by='region').sum()
grouped_df.reset_index(inplace=True)
print(grouped_df)
```

```
  region  sales
0   East  380.0
1  North    0.0
2  South  303.0
3   West  103.0
```

5. Joining DataFrames

In [27]:
```
# import the pandas library
import pandas as pd
left = pd.DataFrame({
   'id':[1,2,3,4,5],
   'Name': ['Alex', 'Amy', 'Allen', 'Alice', 'Ayoung'],
   'subject_id':['sub1','sub2','sub4','sub6','sub5']})
right = pd.DataFrame(
   {'id':[1,2,3,4,5],
   'Name': ['Billy', 'Brian', 'Bran', 'Bryce', 'Betty'],
   'subject_id':['sub2','sub4','sub3','sub6','sub5']})
print (left)
print (right)
```

```
   id    Name subject_id
0   1    Alex       sub1
1   2     Amy       sub2
2   3   Allen       sub4
3   4   Alice       sub6
4   5  Ayoung       sub5
   id   Name subject_id
0   1  Billy       sub2
1   2  Brian       sub4
2   3   Bran       sub3
3   4  Bryce       sub6
4   5  Betty       sub5
```

In [28]:
```
import pandas as pd
left = pd.DataFrame({
   'id':[1,2,3,4,5],
   'Name': ['Alex', 'Amy', 'Allen', 'Alice', 'Ayoung'],
   'subject_id':['sub1','sub2','sub4','sub6','sub5']})
right = pd.DataFrame({
    'id':[1,2,3,4,5],
   'Name': ['Billy', 'Brian', 'Bran', 'Bryce', 'Betty'],
   'subject_id':['sub2','sub4','sub3','sub6','sub5']})
print (pd.merge(left,right,on='id'))
```

```
   id  Name_x subject_id_x Name_y subject_id_y
0   1    Alex         sub1  Billy         sub2
1   2     Amy         sub2  Brian         sub4
2   3   Allen         sub4   Bran         sub3
3   4   Alice         sub6  Bryce         sub6
4   5  Ayoung         sub5  Betty         sub5
```

Here is a summary of the **how** options and their SQL equivalent names –

Merge Method	SQL Equivalent	Description
left	LEFT OUTER JOIN	Use keys from left object
right	RIGHT OUTER JOIN	Use keys from right object
outer	FULL OUTER JOIN	Use union of keys
inner	INNER JOIN	Use intersection of keys

In [29]:
```
import pandas as pd
left = pd.DataFrame({
   'id':[1,2,3,4,5],
   'Name': ['Alex', 'Amy', 'Allen', 'Alice', 'Ayoung'],
   'subject_id':['sub1','sub2','sub4','sub6','sub5']})
right = pd.DataFrame({
   'id':[1,2,3,4,5],
   'Name': ['Billy', 'Brian', 'Bran', 'Bryce', 'Betty'],
   'subject_id':['sub2','sub4','sub3','sub6','sub5']})
print (pd.merge(left, right, on='subject_id', how='left'))
```

```
   id_x  Name_x subject_id  id_y Name_y
0     1    Alex       sub1   NaN    NaN
1     2     Amy       sub2   1.0  Billy
2     3   Allen       sub4   2.0  Brian
3     4   Alice       sub6   4.0  Bryce
4     5  Ayoung       sub5   5.0  Betty
```

In [30]:
```
print (pd.merge(left, right, on='subject_id', how='right'))
```

```
   id_x  Name_x subject_id  id_y Name_y
0   2.0     Amy       sub2     1  Billy
1   3.0   Allen       sub4     2  Brian
2   NaN     NaN       sub3     3   Bran
3   4.0   Alice       sub6     4  Bryce
4   5.0  Ayoung       sub5     5  Betty
```

In [31]:
```
print (pd.merge(left, right, how='outer', on='subject_id'))
```

```
   id_x  Name_x subject_id  id_y Name_y
0   1.0    Alex       sub1   NaN    NaN
1   2.0     Amy       sub2   1.0  Billy
2   3.0   Allen       sub4   2.0  Brian
3   4.0   Alice       sub6   4.0  Bryce
4   5.0  Ayoung       sub5   5.0  Betty
5   NaN     NaN       sub3   3.0   Bran
```

In [32]:
```
print (pd.merge(left, right, on='subject_id', how='inner'))
```

```
   id_x  Name_x subject_id  id_y Name_y
0     2     Amy       sub2     1  Billy
1     3   Allen       sub4     2  Brian
2     4   Alice       sub6     4  Bryce
3     5  Ayoung       sub5     5  Betty
```

In [33]:
```
import pandas as pd
info = pd.DataFrame({'key': ['K0', 'K1', 'K2', 'K3', 'K4', 'K5'],
'A': ['A0', 'A1', 'A2', 'A3', 'A4', 'A5']})
x = pd.DataFrame({'key': ['K0', 'K1', 'K2'],
'B': ['B0', 'B1', 'B2']})
info.join(x, lsuffix='_caller', rsuffix='_x')
info.set_index('key').join(x.set_index('key'))
info.join(x.set_index('key'), on='key')
```

Out[33]:

	key	A	B
0	K0	A0	B0
1	K1	A1	B1
2	K2	A2	B2
3	K3	A3	NaN
4	K4	A4	NaN
5	K5	A5	NaN

6. merge_ordered

In [34]:
```
df1 = pd.DataFrame(
    {
        "key": ["a", "c", "e", "a", "c", "e"],
        "lvalue": [1, 2, 3, 1, 2, 3],
        "group": ["a", "a", "a", "b", "b", "b"]
    }
)
```

In [35]:
```
df1
```

Out[35]:

	key	lvalue	group
0	a	1	a
1	c	2	a
2	e	3	a
3	a	1	b
4	c	2	b
5	e	3	b

In [36]:
```
df2 = pd.DataFrame({"key": ["b", "c", "d"], "rvalue": [1, 2, 3]})
df2
```

Out[36]:

	key	rvalue
0	b	1
1	c	2
2	d	3

In [37]:
```
pd.merge_ordered(df1, df2, fill_method= "ffill", left_by="group")
```

Out[37]:

	key	lvalue	group	rvalue
0	a	1	a	NaN
1	b	1	a	1.0
2	c	2	a	2.0
3	d	2	a	3.0
4	e	3	a	3.0
5	a	1	b	NaN
6	b	1	b	1.0
7	c	2	b	2.0
8	d	2	b	3.0
9	e	3	b	3.0