Pandas

In [1]:

```
import pandas as pd
```

In [2]:

```
#first column is index
#second column value
pd.Series([10,88,3,4,5])
```

Out[2]:

```
0    10
1    88
2     3
3     4
4     5
dtype: int64
```

In [3]:

```
seri = pd.Series([10,88,3,4,5])
type(seri)
```

Out[3]:

```
pandas.core.series.Series
```

In [4]:

```
#The index structure of the series is accessed.
seri.axes
```

Out[4]:

```
[RangeIndex(start=0, stop=5, step=1)]
```

In [5]:

```
seri.ndim
```

Out[5]:

```
1
```

In [6]:

```
seri.dtype
```

Out[6]:

```
dtype('int64')
```

In [7]:

```
seri.size
```

Out[7]:

```
5
```

In [8]:

```
seri.values
```

Out[8]:

```
array([10, 88,  3,  4,  5], dtype=int64)
```

array([10, 88, 3, 4, 5], dtype=int64)

In [9]:

```
#return the first 5 row
seri.head()
```

Out[9]:

```
0    10
1    88
2     3
3     4
4     5
dtype: int64
```

In [10]:

```
seri.head(3)
```

Out[10]:

```
0    10
1    88
2     3
dtype: int64
```

In [11]:

```
#return the last 5 row
seri.tail(3)
```

Out[11]:

```
2    3
3    4
4    5
dtype: int64
```

In [12]:

```
seri1 = pd.Series([99,23,76,2323,98], index = [1,3,5,7,9])
seri1
```

Out[12]:

```
1      99
3      23
5      76
7    2323
9      98
dtype: int64
```

In [13]:

```
seri2 = pd.Series([99,23,76,2323,98], index = ["a","b","c","d","e"])
seri2
```

Out[13]:

```
a      99
b      23
c      76
d    2323
e      98
dtype: int64
```

In [14]:

```
seri2["a"]
```

Out[14]:

```
99
```

In [15]:

```
seri2["a":"c"]
```

Out[15]:

```
a    99
b    23
c    76
dtype: int64
```

In [16]:

```
#Create a dictinary
dic1 = {"reg":10, "log":11,"cart":12}
```

In [17]:

```
series = pd.Series(dic1)
```

In [18]:

```
series
```

Out[18]:

```
reg     10
log     11
cart    12
dtype: int64
```

In [19]:

```
#concatenation
pd.concat([series,series])
```

Out[19]:

```
reg     10
log     11
cart    12
reg     10
log     11
cart    12
dtype: int64
```

Indexing and Slicing

In [20]:

```
import numpy as np
a = np.array([1,2,33,444,75], dtype = "int64")
seri = pd.Series(a)
seri
```

Out[20]:

```
0      1
1      2
2     33
3    444
4     75
dtype: int64
```

In [21]:

```
seri[0]
```

Out[21]:

```
1
```

In [22]:

```
#slicing
seri[0:3]
```

Out[22]:

```
0     1
1     2
2    33
dtype: int64
```

In [23]:

```
seri = pd.Series([121,200,150,99], index = ["reg","loj","cart","rf"])
seri
```

Out[23]:

```
reg     121
loj     200
cart    150
rf       99
dtype: int64
```

In [24]:

```
#this method just uses to access indexes.
seri.index
```

Out[24]:

```
Index(['reg', 'loj', 'cart', 'rf'], dtype='object')
```

In [25]:

```
#this method just uses to access keys.
seri.keys
```

Out[25]:

```
<bound method Series.keys of reg     121
loj     200
cart    150
rf       99
dtype: int64>
```

In [26]:

```
#it can be used like dictionary method.
list(seri.items())
```

Out[26]:

```
[('reg', 121), ('loj', 200), ('cart', 150), ('rf', 99)]
```

In [27]:

```
seri.values
```

Out[27]:

```
array([121, 200, 150,  99], dtype=int64)
```

In [28]:

```
"reg" in seri
```

Out[28]:

```
True
```

In [29]:

```
"a" in seri
```

Out[29]:

```
False
```

In [30]:

```
seri["reg"]
```

Out[30]:

```
121
```

In [31]:

```
#fancy
seri[["rf","reg"]]
```

Out[31]:

```
rf      99
reg    121
dtype: int64
```

In [32]:

```
seri["reg"] = 130
seri["reg"]
```

Out[32]:

```
130
```

In [33]:

```
seri["reg":"loj"]
```

Out[33]:

```
reg    130
loj    200
dtype: int64
```

In [34]:

```
seri["reg"] in seri
```

Out[34]:

```
False
```

In [35]:

```
"reg" in seri
```

Out[35]:

```
True
```

In [36]:

```
130 in seri.items()
```

Out[36]:

```
False
```

Creating DataFrame

In [37]:

```
#NumPy cannot keep categorical and numeric data together. That's why we need a Pandas.
import pandas as pd
l = [1,2,23,345,7,8,3]
l
```

Out[37]:

```
[1, 2, 23, 345, 7, 8, 3]
```

In [38]:

```
pd.DataFrame(l,columns = ["degisken_isimleri"])
```

Out[38]:

	degisken_isimleri
0	1
1	2
2	23
3	345
4	7
5	8
6	3

In [39]:

```
import numpy as np
m = np.arange(1,10).reshape((3,3))
m
```

Out[39]:

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

In [40]:

```
pd.DataFrame(m, columns=["var1","var2","var3"])
```

Out[40]:

	var1	var2	var3
0	1	2	3
1	4	5	6
2	7	8	9

In [41]:

```
#dataframe renaming
df =pd.DataFrame(m, columns=["var1","var2","var3"])
df.head()
```

Out[41]:

	var1	var2	var3
0	1	2	3
1	4	5	6
2	7	8	9

In [42]:

```
df.columns
```

Out[42]:

```
Index(['var1', 'var2', 'var3'], dtype='object')
```

In [43]:

```
df.columns = ["deg1","deg2","deg3"]
```

In [44]:

```
df
```

Out[44]:

	deg1	deg2	deg3
0	1	2	3
1	4	5	6
2	7	8	9

In [45]:

```
df.index
```

Out[45]:

```
RangeIndex(start=0, stop=3, step=1)
```

In [46]:

```
df
```

Out[46]:

	deg1	deg2	deg3
0	1	2	3
1	4	5	6
2	7	8	9

In [47]:

```
df.describe()
```

Out[47]:

	deg1	deg2	deg3
count	3.0	3.0	3.0
mean	4.0	5.0	6.0
std	3.0	3.0	3.0
min	1.0	2.0	3.0
25%	2.5	3.5	4.5
50%	4.0	5.0	6.0
75%	5.5	6.5	7.5
max	7.0	8.0	9.0

In [48]:

```
df.T
```

Out[48]:

	0	1	2
deg1	1	4	7
deg2	2	5	8
deg3	3	6	9

In [49]:

```
type(df)
```

Out[49]:

```
pandas.core.frame.DataFrame
```

In [50]:

```
df.axes
```

Out[50]:

```
[RangeIndex(start=0, stop=3, step=1),
 Index(['deg1', 'deg2', 'deg3'], dtype='object')]
```

In [51]:

```
df.shape
```

Out[51]:

```
(3, 3)
```

In [52]:

```
df.ndim
```

Out[52]:

```
2
```

In [53]:

```
df.size
```

Out[53]:

```
9
```

In [54]:

```
df.values
```

Out[54]:

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

In [55]:

```
type(df.values)
```

Out[55]:

```
numpy.ndarray
```

In [56]:

```
df.head()
```

Out[56]:

	deg1	deg2	deg3

	deg1	deg2	deg3
0	1	2	3
1	4	5	6
2	7	8	9

In [57]:

```
df.tail(1)
```

Out[57]:

	deg1	deg2	deg3
2	7	8	9

In [58]:

```
a = np.array([1,2,3,4,5])
pd.DataFrame(a, columns =["deg1"])
```

Out[58]:

	deg1
0	1
1	2
2	3
3	4
4	5

In [59]:

```
import numpy as np
s1 = np.random.randint(10,size = 5)
s2 = np.random.randint(10,size = 5)
s3 = np.random.randint(10,size = 5)
```

In [60]:

```
dic1 = {"var1":s1, "var2":s2, "var3":s3}
dic1
```

Out[60]:

```
{'var1': array([1, 0, 2, 3, 0]),
 'var2': array([7, 8, 5, 0, 4]),
 'var3': array([4, 5, 1, 4, 9])}
```

In [61]:

```
df = pd.DataFrame(dic1)
df
```

Out[61]:

	var1	var2	var3
0	1	7	4
1	0	8	5
2	2	5	1
3	3	0	4
4	0	4	9

In [62]:

```
df[0:1]
```

Out[62]:

	var1	var2	var3
0	1	7	4

In [63]:

```
df[0:2]
```

Out[63]:

	var1	var2	var3
0	1	7	4
1	0	8	5

In [64]:

```
df.index = ["a","b","c","d","e"]
df
```

Out[64]:

	var1	var2	var3
a	1	7	4
b	0	8	5
c	2	5	1
d	3	0	4
e	0	4	9

In [65]:

```
df["c":"e"]
```

Out[65]:

	var1	var2	var3
c	2	5	1
d	3	0	4
e	0	4	9

In [66]:

```
df.drop("a", axis = 0)
```

Out[66]:

	var1	var2	var3
b	0	8	5
c	2	5	1
d	3	0	4
e	0	4	9

In [67]:

```
df
```

Out[67]:

	var1	var2	var3
a	1	7	4
b	0	8	5
c	2	5	1
d	3	0	4
e	0	4	9

In [68]:

```
#inplace = If we make it true, the drop will be done permanently.
df.drop("a", axis = 0, inplace = True)
```

In [69]:

```
df
```

Out[69]:

	var1	var2	var3
b	0	8	5
c	2	5	1
d	3	0	4
e	0	4	9

In [70]:

```
#fancy
l = {"c","e"}
df.drop(l, axis = 0)
```

Out[70]:

	var1	var2	var3
b	0	8	5
d	3	0	4

In [71]:

```
"var1" in df
```

Out[71]:

```
True
```

In [72]:

```
l = ["var1","var4","var2"]
for i in l:
    print(i in df)
```

```
True
False
True
```

In [73]:

```
l = ["var1","var2"]
df.drop(l, axis = 1)
```

Out[73]:

	var3 var3
b	5
c	1
d	4
e	9

loc & iloc

In [74]:

```
import numpy as np
import pandas as pd
m = np.random.randint(1,30, size = (10,3))
df = pd.DataFrame(m, columns=["var1","var2","var3"])
df
```

Out[74]:

	var1	var2	var3
0	5	22	10
1	5	19	26
2	3	26	2
3	24	27	11
4	10	18	28
5	29	28	16
6	6	5	2
7	12	24	28
8	18	3	7
9	22	18	29

In [75]:

```
df.loc[0:3]
```

Out[75]:

	var1	var2	var3
0	5	22	10
1	5	19	26
2	3	26	2
3	24	27	11

In [76]:

```
df.iloc[0:3]
```

Out[76]:

	var1	var2	var3
0	5	22	10
1	5	19	26
2	3	26	2

In [77]:

```
df.iloc[0,0]
```

Out[77]:

```
5
```

In [78]:

```
df.iloc[:3,:2]
```

Out[78]:

	var1	var2
0	**5**	**22**
1	**5**	**19**
2	**3**	**26**

In [79]:

```
df.loc[0:3,"var3"]
```

Out[79]:

```
0    10
1    26
2     2
3    11
Name: var3, dtype: int32
```

In [80]:

```
df.iloc[0:3]["var3"]
```

Out[80]:

```
0    10
1    26
2     2
Name: var3, dtype: int32
```

In [81]:

```
df[0:2]
```

Out[81]:

	var1	var2	var3
0	**5**	**22**	**10**
1	**5**	**19**	**26**

In [82]:

```
df.iloc[0:2]
```

Out[82]:

	var1	var2	var3
0	**5**	**22**	**10**
1	**5**	**19**	**26**

In [83]:

```
df.loc[0:2]
```

Out[83]:

	var1	var2	var3
0	5	22	10
1	5	19	26
2	3	26	2

In [84]:

```
df.iloc[:,:2]
```

Out[84]:

	var1	var2
0	5	22
1	5	19
2	3	26
3	24	27
4	10	18
5	29	28
6	6	5
7	12	24
8	18	3
9	22	18

In [85]:

```
df.iloc[:,0]
```

Out[85]:

```
0     5
1     5
2     3
3    24
4    10
5    29
6     6
7    12
8    18
9    22
Name: var1, dtype: int32
```

In [86]:

```
df.loc[0:3,"var3"]
```

Out[86]:

```
0    10
1    26
2     2
3    11
Name: var3, dtype: int32
```

In [87]:

```
df.iloc[0:3]["var3"]
```

Out[87]:

```
0    10
1    26
2     2
Name: var3, dtype: int32
```

Conditional Operations

In [88]:

```
df[0:2][["var1","var2"]]
```

Out[88]:

	var1	var2
0	5	22
1	5	19

In [89]:

```
df.var1 > 15
```

Out[89]:

```
0    False
1    False
2    False
3     True
4    False
5     True
6    False
7    False
8     True
9     True
Name: var1, dtype: bool
```

In [90]:

```
df[df.var1 > 15]["var2"]
```

Out[90]:

```
3    27
5    28
8     3
9    18
Name: var2, dtype: int32
```

In [91]:

```
df[(df.var1 >10) & (df.var3 < 8)]
```

Out[91]:

	var1	var2	var3
8	18	3	7

In [92]:

```
df.loc[(df.var1 >10),["var1","var2"]]
```

Out[92]:

	var1	var2
3	24	27
5	29	28
7	12	24
8	18	3
9	22	18

In [93]:

```
df[(df.var1 >10)][["var1","var2"]]
```

Out[93]:

	var1	var2
3	24	27
5	29	28
7	12	24
8	18	3
9	22	18

Join

In [94]:

```
import numpy as np
import pandas as pd
m = np.random.randint(1,30, size = (10,3))
df1 = pd.DataFrame(m, columns=["var1","var2","var3"])
df1
```

Out[94]:

	var1	var2	var3
0	10	12	12
1	26	4	17
2	10	29	3
3	20	22	9
4	2	27	18
5	17	5	25
6	13	21	10
7	3	19	2
8	15	22	5
9	16	6	17

In [95]:

```
df2 = df1 + 99
df2
```

Out[95]:

	var1	var2	var3
0	109	111	111
1	125	103	116
2	109	128	102
3	119	121	108
4	101	126	117
5	116	104	124
6	112	120	109
7	102	118	101

8	var1	var2	var3
9	115	105	116

In [96]:

```
pd.concat([df1,df2])
```

Out[96]:

	var1	var2	var3
0	10	12	12
1	26	4	17
2	10	29	3
3	20	22	9
4	2	27	18
5	17	5	25
6	13	21	10
7	3	19	2
8	15	22	5
9	16	6	17
0	109	111	111
1	125	103	116
2	109	128	102
3	119	121	108
4	101	126	117
5	116	104	124
6	112	120	109
7	102	118	101
8	114	121	104
9	115	105	116

In [97]:

```
pd.concat([df1,df2], ignore_index=True)
```

Out[97]:

	var1	var2	var3
0	10	12	12
1	26	4	17
2	10	29	3
3	20	22	9
4	2	27	18
5	17	5	25
6	13	21	10
7	3	19	2
8	15	22	5
9	16	6	17
10	109	111	111
11	125	103	116

12	[illegible]	[illegible]	[illegible]
13	119	121	108
14	101	126	117
15	116	104	124
16	112	120	109
17	102	118	101
18	114	121	104
19	115	105	116

In [98]:

```
df1.columns
```

Out[98]:

```
Index(['var1', 'var2', 'var3'], dtype='object')
```

In [99]:

```
df2.columns = ["var1","var2","deg3"]
df2
```

Out[99]:

	var1	var2	deg3
0	109	111	111
1	125	103	116
2	109	128	102
3	119	121	108
4	101	126	117
5	116	104	124
6	112	120	109
7	102	118	101
8	114	121	104
9	115	105	116

In [100]:

```
df1
```

Out[100]:

	var1	var2	var3
0	10	12	12
1	26	4	17
2	10	29	3
3	20	22	9
4	2	27	18
5	17	5	25
6	13	21	10
7	3	19	2
8	15	22	5
9	16	6	17

In [101]:

```
df2
```

Out[101]:

	var1	var2	deg3
0	109	111	111
1	125	103	116
2	109	128	102
3	119	121	108
4	101	126	117
5	116	104	124
6	112	120	109
7	102	118	101
8	114	121	104
9	115	105	116

In [102]:

```
pd.concat([df1,df2])
```

```
c:\users\omer\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launcher
.py:1: FutureWarning: Sorting because non-concatenation axis is not aligned. A future ver
sion
of pandas will change to not sort by default.

To accept the future behavior, pass 'sort=False'.

To retain the current behavior and silence the warning, pass 'sort=True'.

  """Entry point for launching an IPython kernel.
```

Out[102]:

	deg3	var1	var2	var3
0	NaN	10	12	12.0
1	NaN	26	4	17.0
2	NaN	10	29	3.0
3	NaN	20	22	9.0
4	NaN	2	27	18.0
5	NaN	17	5	25.0
6	NaN	13	21	10.0
7	NaN	3	19	2.0
8	NaN	15	22	5.0
9	NaN	16	6	17.0
0	111.0	109	111	NaN
1	116.0	125	103	NaN
2	102.0	109	128	NaN
3	108.0	119	121	NaN
4	117.0	101	126	NaN
5	124.0	116	104	NaN
6	109.0	112	120	NaN
7	101.0	102	118	NaN

	deg3	var1	var2	var3
8	104.0	114	121	NaN
9	116.0	115	105	NaN

In [103]:

```
pd.concat([df1,df2], join="inner", ignore_index=True)
```

Out[103]:

	var1	var2
0	10	12
1	26	4
2	10	29
3	20	22
4	2	27
5	17	5
6	13	21
7	3	19
8	15	22
9	16	6
10	109	111
11	125	103
12	109	128
13	119	121
14	101	126
15	116	104
16	112	120
17	102	118
18	114	121
19	115	105

In [104]:

```
?pd.concat
```

In [105]:

```
pd.concat([df1,df2], join_axes = [df2.columns], ignore_index= True)
```

```
c:\users\omer\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launcher
.py:1: FutureWarning: The join_axes-keyword is deprecated. Use .reindex or .reindex_like
on the result to achieve the same functionality.
  """Entry point for launching an IPython kernel.
```

Out[105]:

	var1	var2	deg3
0	10	12	NaN
1	26	4	NaN
2	10	29	NaN
3	20	22	NaN
4	2	27	NaN
5	17	5	NaN

	var1	var2	deg3
6	13	24	NaN
7	3	19	NaN
8	15	22	NaN
9	16	6	NaN
10	109	111	111.0
11	125	103	116.0
12	109	128	102.0
13	119	121	108.0
14	101	126	117.0
15	116	104	124.0
16	112	120	109.0
17	102	118	101.0
18	114	121	104.0
19	115	105	116.0

In [106]:

```
df1.columns
```

Out[106]:

```
Index(['var1', 'var2', 'var3'], dtype='object')
```

In [107]:

```
df2.columns
```

Out[107]:

```
Index(['var1', 'var2', 'deg3'], dtype='object')
```

Concatenation

In [108]:

```
import pandas as pd

df1 = pd.DataFrame({'Worker':['John','Doe','Mehmet','Jeff'],
                    'Positions':['HR','Engineering','AI','Accounting']
                   })
df1
```

Out[108]:

	Worker	Positions
0	John	HR
1	Doe	Engineering
2	Mehmet	AI
3	Jeff	Accounting

In [109]:

```
df2 = pd.DataFrame({'Worker':['John','Doe','Mehmet','Jeff'],

                    'Date_Of_starting_work':[2012,'2018','2015','2017']
                   })
df2
```

Out[109]:

	Worker	Date_Of_starting_work
0	John	2012
1	Doe	2018
2	Mehmet	2015
3	Jeff	2017

In [110]:

```
pd.merge(df1,df2)
```

Out[110]:

	Worker	Positions	Date_Of_starting_work
0	John	HR	2012
1	Doe	Engineering	2018
2	Mehmet	AI	2015
3	Jeff	Accounting	2017

In [111]:

```
#many to one
df3 = pd.merge(df1,df2, on = "Worker")
df3
```

Out[111]:

	Worker	Positions	Date_Of_starting_work
0	John	HR	2012
1	Doe	Engineering	2018
2	Mehmet	AI	2015
3	Jeff	Accounting	2017

In [112]:

```
df4 = pd.DataFrame({'Positions':['Accounting',"Engineering",'HR'],
                    'Mudur': ['Caner','Mustafa','Berkcan']
                   })
pd.merge(df3,df4)
```

Out[112]:

	Worker	Positions	Date_Of_starting_work	Mudur
0	John	HR	2012	Berkcan
1	Doe	Engineering	2018	Mustafa
2	Jeff	Accounting	2017	Caner

In [113]:

```
pd.merge(df3,df4)
```

Out[113]:

	Worker	Positions	Date_Of_starting_work	Mudur
0	John	HR	2012	Berkcan
1	Doe	Engineering	2018	Mustafa
2	Jeff	Accounting	2017	Caner

In [114]:

```
#many to many
df5 = pd.DataFrame({'Positions':['Accounting','Accounting',"Engineering","Engineering",'
HR','HR'],
                    'Ability': ['Math','Excel','Coding','Linux','Excel','Management']
                   })

df5
```

Out[114]:

	Positions	Ability
0	Accounting	Math
1	Accounting	Excel
2	Engineering	Coding
3	Engineering	Linux
4	HR	Excel
5	HR	Management

In [115]:

```
df1
```

Out[115]:

	Worker	Positions
0	John	HR
1	Doe	Engineering
2	Mehmet	AI
3	Jeff	Accounting

In [116]:

```
pd.merge(df1,df5)
```

Out[116]:

	Worker	Positions	Ability
0	John	HR	Excel
1	John	HR	Management
2	Doe	Engineering	Coding
3	Doe	Engineering	Linux
4	Jeff	Accounting	Math
5	Jeff	Accounting	Excel