Pandas Modülü

Python ile very analizi yaparken bilmemiz gereken bir başka önemli modül pandas'tır.

```
import pandas as pd
```

Komutu ile öncelikle pandas'ı çağırmamız gerekir.

Pandas modulu ile numpy modulu birbirini tamamlarlar. Bunun icin genelde bu iki modul birlikte cagirilrlar.

```
import pandas as pd
import numpy as np
```

Pandas'ta öğrenmemiz gereken iki önemli veri yapısı vardır: Series ve DataFrame. Genel olarak series, tek boyutlu veri il , DataFrame çok boyutlu veri ile çaılışılırken kullanılır.

1. Series

Pandas'taki series yapısı numpy'daki array'a (np.array) benzer; fakat bazı ekstra özelliklere sahiptir. Series, pd.Series() ile oluşturulur.

```
s=pd.Series([12,-4,7,9])
print(s)
0      12
1      -4
2      7
3      9
```

Eğer indis'I belirtmezsek, np.array'de yada list'de olduğu gibi indisler 0, 1, … olur. Farklı olarak Series'de Series'in elemanlarının idislerini özelleştirebilir (etiketleyebiliriz).

```
s=pd.Series([12,-4,7,9], index=['a','b','c','d'])
print(s)
a      12
b      -4
c      7
d      9
```

Şimdi series'in elemanlarını yeni indislerle çağıralım:

```
s['c']
7
```

```
notlar=pd.Series([62,90,100], index=['Ebru','Murat','Firat'])
notlar[Firat']
```

```
100
```

Eğer direkt tüm değerleri görmek istiyorsak, indislerle hic ilgilenmeden series adından sonra `.values` yazabiliriz:

```
notlar.values
array([ 62,  90, 100], dtype=int64)
```

Benzer şekilde yalnızca indisleri çağırmak istiyorsak `.index` yazarız:

```
notlar.index
Index(['Ebru', 'Murat', 'Firat'], dtype='object')
```

```
notlar[:2] #ilk iki notu getir
Ebru     62
Murat    90
dtype: int64
notlar[2:] #uçuncu nottan sonrasini göster
Firat    100
dtype: int64
```

Pandas Series'e Yeni Bir Eleman Eklemek

Oluşturduğumuz pd.series'e yeni bir eleman eklemek istersek

`.set_value('indis_adı', deger)` komutunu kullanırız.

Aşağıdaki örnekte halihazırda var olan `notlar` adlı series'e Hasan adlı öğrencinin 80 olan notunu ekleyelim.

```
notlar.set_value('Hasan', 80)
print(notlar)
Ebru     62
Murat    90
Firat    100
Hasan    80
dtype: int64
```

** Dikkat edilirse series oluşturulurken önce değer sonra indis beilirtiliyordu. .set_values() ile yeni bir eleman eklenirken ise once indis daha sonra deger giriliyor.

Pandas Series'ten Bir Eleman Çıkarmak

Oluşturduğumuz pd.series'ten bir eleman çıkarmak istersek bunu .drop komutu ile yaparız. Drop edeceğimiz (çıkaracağımız) elemanı indis'I ile yani label'I ile çağırırız. Elemanı çıkarttıktan sonra yeni series'te bu değişikliğin kalıcı olmasını istersek, bir de opsiyon olarak "inplace=True"ekleriz.

Genel format:

```
seriesAdı.drop(labels=['indisAdı',inplace=True]
```

```
notlar.drop(labels=['Hasan'], inplace=True)
Ebru     62
Murat    90
Firat    100
dtype: int64
```

Numpy Array'i Pandas Series Dönüştürmek

Var olan numpy array'leri pd.Series() icinde yazarak series formatına dönüştürebiliriz.

```
npArrayim=np.array([1,2,3,4])
pdSeriesim=pd.(npArrayim)
```

Temel Matematik Işlemleri

Numpy'daki temel matematik fonksiyonlari: np.log, np.sqrt, np.power, np.floor, np.ceil, np.sum, np.prod , np.log2.. pandas series'lerede uygulanabilir.

(Numpy'daki matematiksel fonksiyonlarin tamamı icin:

https://docs.scipy.org/doc/numpy-1.13.0/reference/routines.math.html

adresine gidiniz).

Not: Bu matematiksel fonksiyonlar, series'teki tüm elemanlara uygulanır.

```
yeniSeri=pd.Series([1.9, -2.8, -3,4])
np.ceil(yeniSeri) #her ondalıklı elemani kendinden buyuk en yakın
f                  tamsayıya götürür.
0    2.0
1   -2.0
2   -3.0
3    4.0
dtype: float64
np.log(yeniSeri)
0    0.641854
1         NaN
2         NaN
3    1.386294
dtype: float64
np.sum(yeniSeri)
0.0000001
```

Bir Series'deki Farklı Elemanları Saymak

.value_counts(), bir series'de her bir farklı elemanın kaç defa olduğunu sayar:

```
s=pd.Series(['a', 'b', 'a', 'c', 'f', 'c', 'c', 'b', 'b'])
```

```
s.value_counts()
c    3
b    3
a    2
f    1
dtype: int64
```

NaN Değerler

Yukarıda negative sayılarin logaritmasini alriken NaN değer bulduk (NaN: Not a Number). Python ile veri analizi yaparken bir çok kez NaN değerle karşılaşırız; özellikle kayıp veri NaN olarak kaydedilir. .isnull() bir series'deki nan değerleri bulur, notnull() ise nan olamayn degerleri bulur.

```
a=np.log(yeniSeri)
print(a)
0    0.641854
1         NaN
2         NaN
3    1.386294
dtype: float64
a.isnull()
0    False
1     True
2     True
3    False
dtype: bool
a.notnull()
0     True
1    False
2    False
3     True
dtype: bool
```

Series'leri Dictionary gibi Düşünmek

Her bir anahtarda (key'de) tek bir değerin olduğu dictionary'ler direkt Series gibi düşünülebilir. Tek yapmamız gereken bu sekildeki bir dictionary'i pd.Series()'in içine koymaktır.

```
vize={'aybuke':100, 'berfin':50, 'ayse':75}
vizeSeri=pd.Series(vize)
print(vizeSeri)
aybuke    100
ayse       75
berfin     50
dtype: int64
```

2. DataFrame

Pandas'taki bir diğer önemli very yapısı DataFrame'dir. DataFrame, yapı olarak Series'e benzer fakat buradaki temel fark, DataFrame'in iki boyutlu olması, böylece tablo oluşturmaya müsait olmasıdır. Aynı zamanda bir veriyi excel'e çıkarmak, yada bir excel dosyasını okumak için de DataFrame'den faydalanırız. Ayrıca DataFrame sütunların farklı tipte (sayısal, string) olmasına izein verir.

Örnek olarak şöyle bir tabloyu DataFrame yardımıyla oluşturalım:

No	Doğum Tarihi	Isim	Bölüm
2018123001	1999	Fırat	Bilgisayar
2018123002	1998	Acun	Radyo-TV
2018123003	2000	Şeyma	Moda
2018123004	1999	Murat	Müzik

Bunun için tabloyu öncelikle dictionary formatında kaydederiz:

```
temp={'No':[2018123001, 2018123002,2018123003,2018123004], 'Doğum
Tarihi':[1999,1998,2000,1999], 'Isim':['Fırat','Acun', 'Şeyma',
'Murat'], 'Bölüm':['Bilgisayar', 'Radyo-TV', 'Moda', 'Müzik']}
```

Daha sonra bu dictionary yapısını pd.DataFrame() ile DataFrame'e dönüştürüz.

```
tabloDF=pd.DataFrame(temp)
print(tabloDF)
         Bölüm  Doğum Tarihi  Isim          No
0  Bilgisayar          1999  Fırat  2018123001
1    Radyo-TV          1998   Acun  2018123002
2        Moda          2000  Şeyma  2018123003
3       Müzik          1999  Murat  2018123004
```

Eğer burada kolon isimlerini açıkça yazarsak , kolon isimleri otomatik olarak kendiliginden alfabetik sıralanmaz:

```
tabloDF=pd.DataFrame(temp,columns=['No','Doğum Tarihi','Isim','Bolum'
])
```

Yada pratik şöyle düşünebiliriz. DataFrame'deki column adları, bu DataFrame'in oluşturulduğu dictionary'deki key'lerdi. O halde .keys() komutu ile key adlarını getirir; bu gelen keyleri list() komutuyla liste haline getirebiliriz.

```
tabloDF=pd.DataFrame(temp,columns=list(temp.keys()))
print(tabloDF)
           No  Doğum Tarihi   Isim       Bölüm
0  2018123001          1999  Fırat  Bilgisayar
1  2018123002          1998   Acun    Radyo-TV

2  2018123003          2000  Şeyma        Moda
```

```
3  2018123004          1999  Murat       Müzik
```

DataFrame'in Kolonlarını Getirmek

Bir kolonun tamamını getirmek istersek, o kolonun adı `[' ']` içinde yazılır.

```
tabloDF['Bölüm']
0    Bilgisayar
1      Radyo-TV
2          Moda
3         Müzik
Name: Bölüm, dtype: object
```

Not: Çağırdığımız kolon pd.Series tipindedir.

Alternatif olarak çağırmak istediğimiz kolonun adını .'dan sonar yazarak da kolonu çağırabilirz.

```
tabloDF.Bölüm
0    Bilgisayar
1      Radyo-TV
2          Moda
3         Müzik
Name: Bölüm, dtype: object
```

DataFrame'in Satırlarını Getirmek

Bir DataFrame'in satırları . loc[] ile getirilir.

.loc[]'un içine çağırmak istediğimiz satırın indeksini yazarak o satırı çağırabiliriz.

```
tabloDF.loc[0]
No              2018123001
Doğum Tarihi          1999
Isim                 Fırat
Bölüm           Bilgisayar
Name: 0, dtype: object
```

Eğer birden fazla satır çağırmak istiyorsak loc'u kullanmayabiliriz.

```
tabloDF[0:2]
           No  Doğum Tarihi   Isim       Bölüm
0  2018123001          1999  Fırat  Bilgisayar
1  2018123002          1998   Acun    Radyo-TV
```

Bir Excel Dosyasını DataFrame Olarak Okumak

pd.read_excel() komutu ile directory'mizdeki excel dosyalarını okuyabiliriz.

Dieylimki data adlı bir excel dosyamız var.

```
df=pd.read_excel('data.xlsx')
```

Bir DataFrame'i Olarak Excel Dosyası Olarak Kaydetmek

tabloDF adlı dataFrame'I bir excel dosyası olarak kaydedelim.

Bunun için önce bir writer oluşturuz:

```
writer = pd.ExcelWriter('boylar2.xlsx', engine='xlsxwriter')
```

Bu writer'i .to_excel(writer) komutuyla dataframe'imiz olan tabloDF'ye ekleriz.

```
tabloDF.to_excel(writer)
```

Daha sonra bu writer'i .save() komutuyla kaydederiz:

```
writer.save()
```

```
tabloDF.isin([1999, 'Dogum Yılı'])
```

```
del tabloDF['Bölüm']
```