1. GİRİŞ

Veri Bilimi, karmaşık problemleri çözmek için yapılandırılmış ve yapılandırılmamış verilerden bilgi ve öngörü elde etmek için bilimsel yöntemleri, süreçleri, algoritmaları ve sistemleri kullanan çok disiplinli bir alandır. Bu süreçte, bilimsel problem çözme teknikleri, istatistik, bilimsel yöntemler, yapay zeka (AI) ve veri analizi dahil olmak üzere birçok alan bir araya getirilip, birlikte kullanır.

Veri Bilimi temizleme, toplama ve gelişmiş veri analizine uygun hale getirme işleme dahil olmak üzere verileri analize hazırlamayı kapsar. Bu işlemler için Python'da kullanılan **NumPy**, **Pandas** ve veriyi daha iyi gözlemleyip anlamak için **Matplotlib** kütüphanesi ile veri görselleştirmeye değinilecektir.

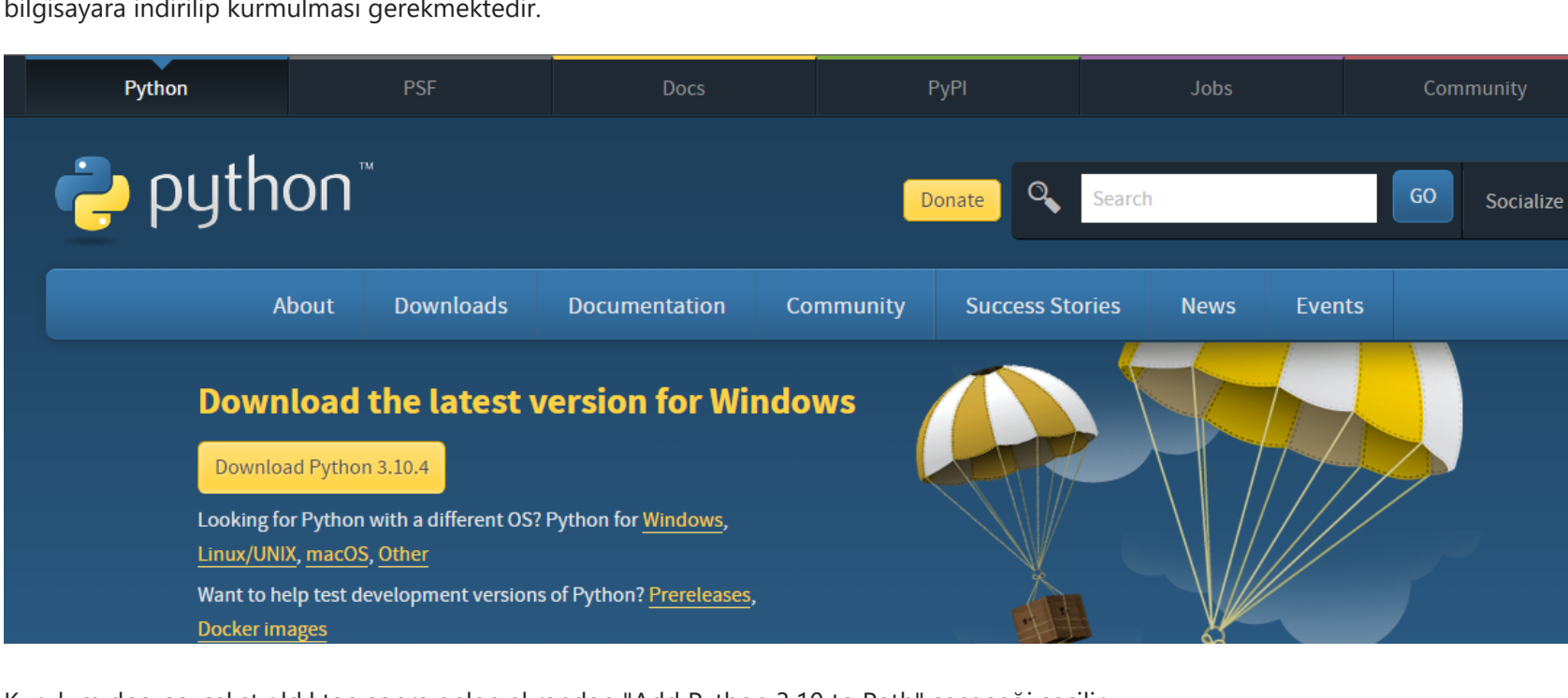
2. KULLANILAN PROGRAM VE KÜTÜPHANELERİN TANITILMASI

Python'un kolaylıkla verileri analiz edebilme yeteneği ve kullanışlı arayüzü, tercih edilmesinin nedenleri arasındadır. Burada Python arayüz olarak Jupyter Lab kullanılacaktır. Tercihe bağlı olarak jupyter notbook veya başka arayüzler kullanılabilir.

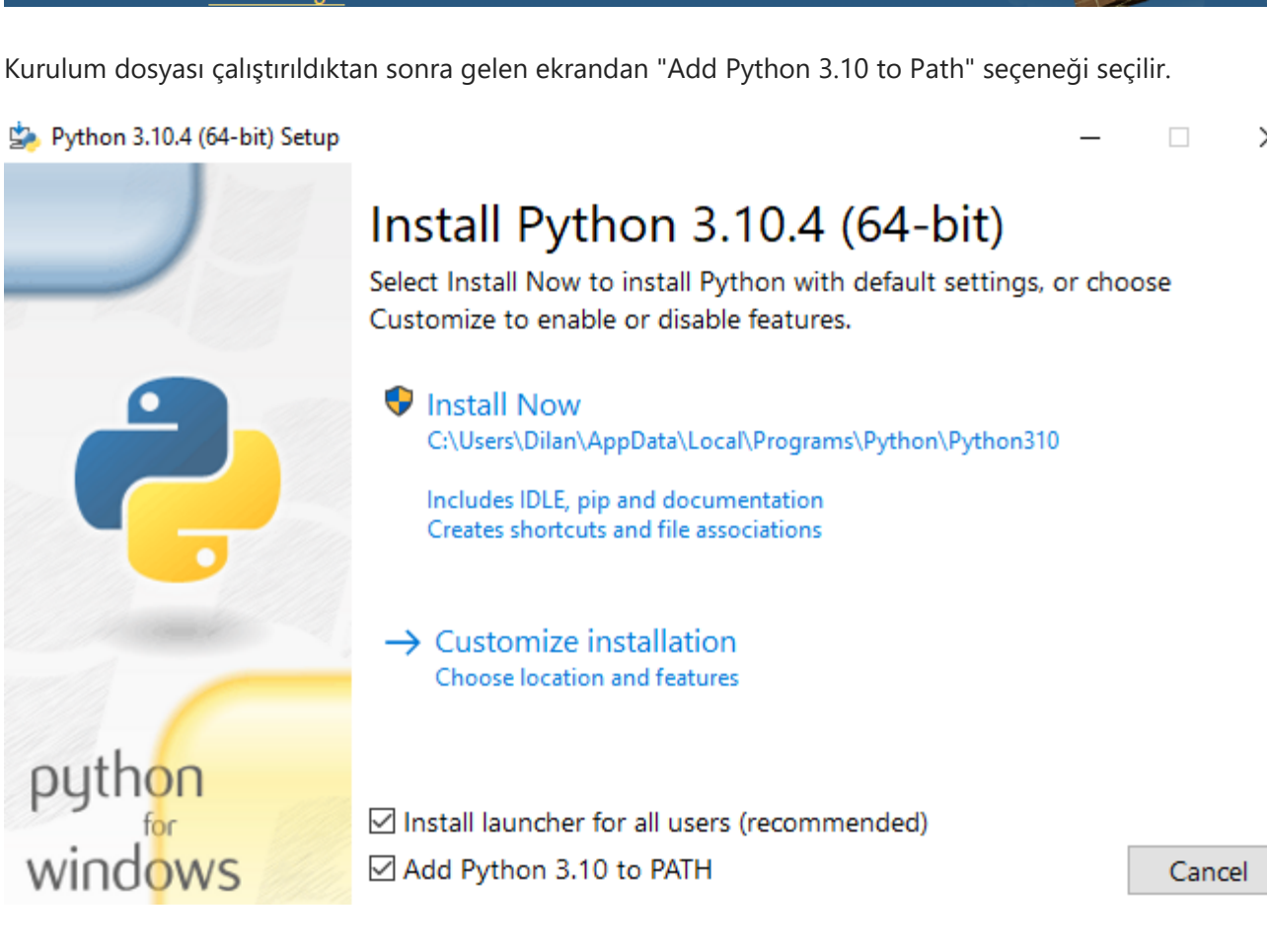
Jupyter notbook; not defterlerini oluşturmak, düzenlemek ve çalıştırmak için sanal ortamın oluşturulmasının yanı sıra basit bir dosya tarayıcısı hizmeti de sağlar, Jupyter Lab ise daha karmaşık bir yapıya sahiptir ve bir kullanıcı ara yüzü ile birlikte entegre edilebilen bir ortamda kullanılır. Ayrıca Jupyter Lab, Jupyter Notbook'un sahip olduğu tüm özelliklere sahiptir, bu yüzden burada Jupyter Lab üzerinden işlemler gerçekleştirilecektir.

Python Kurulumu

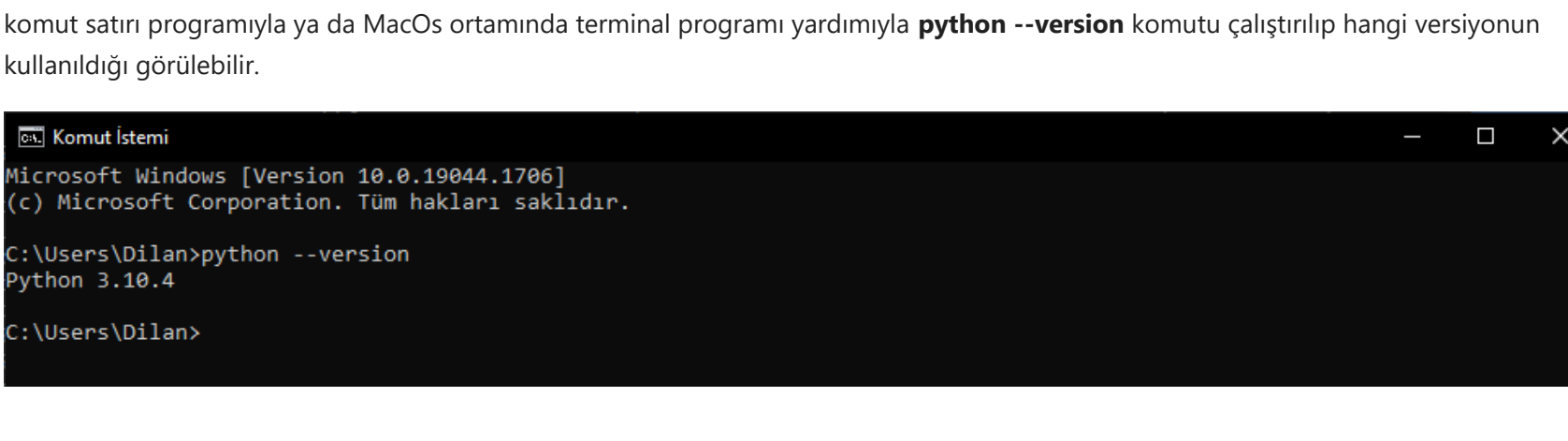
Python kurulumu için **www.python.org** sitesini ziyaret edilir. Download linkinden kullanılan işletim sistemine göre Python derleyicisinin bilgisayara indirilip kurmulması gerekmektedir.



Kurulum dosyası çalıştırıldıktan sonra gelen ekrandan "Add Python 3.10 to Path" seçeneği seçilir.

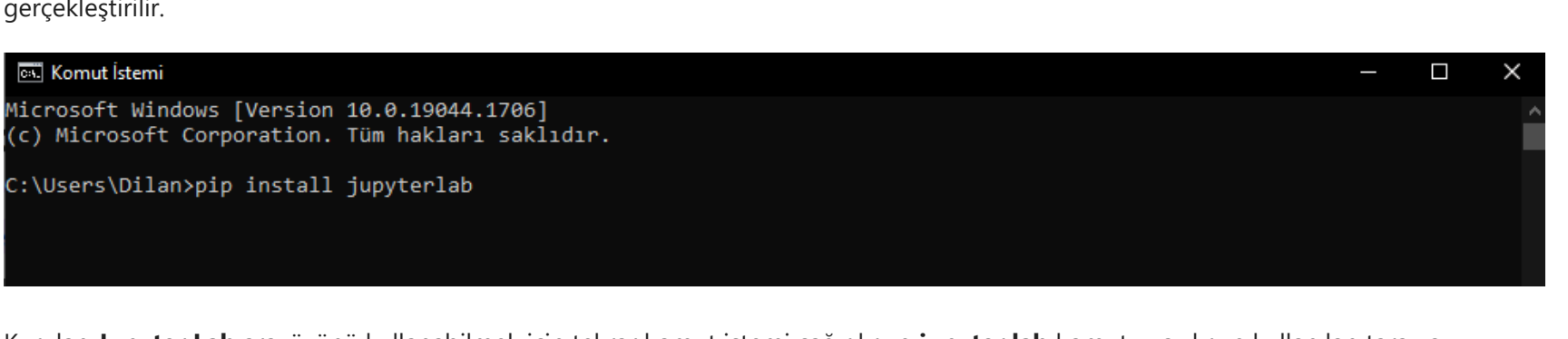


Python kurulumundan sonra python'ın başarılı bir şekilde kurulup kurulmadığı kontrol edilebilir. Windows' da cmd ya da powershell komut satırı programıyla ya da MacOs ortamında terminal programı yardımıyla **python --version** komutu çalıştırılıp hangi versiyonun kullanıldığı görülebilir.



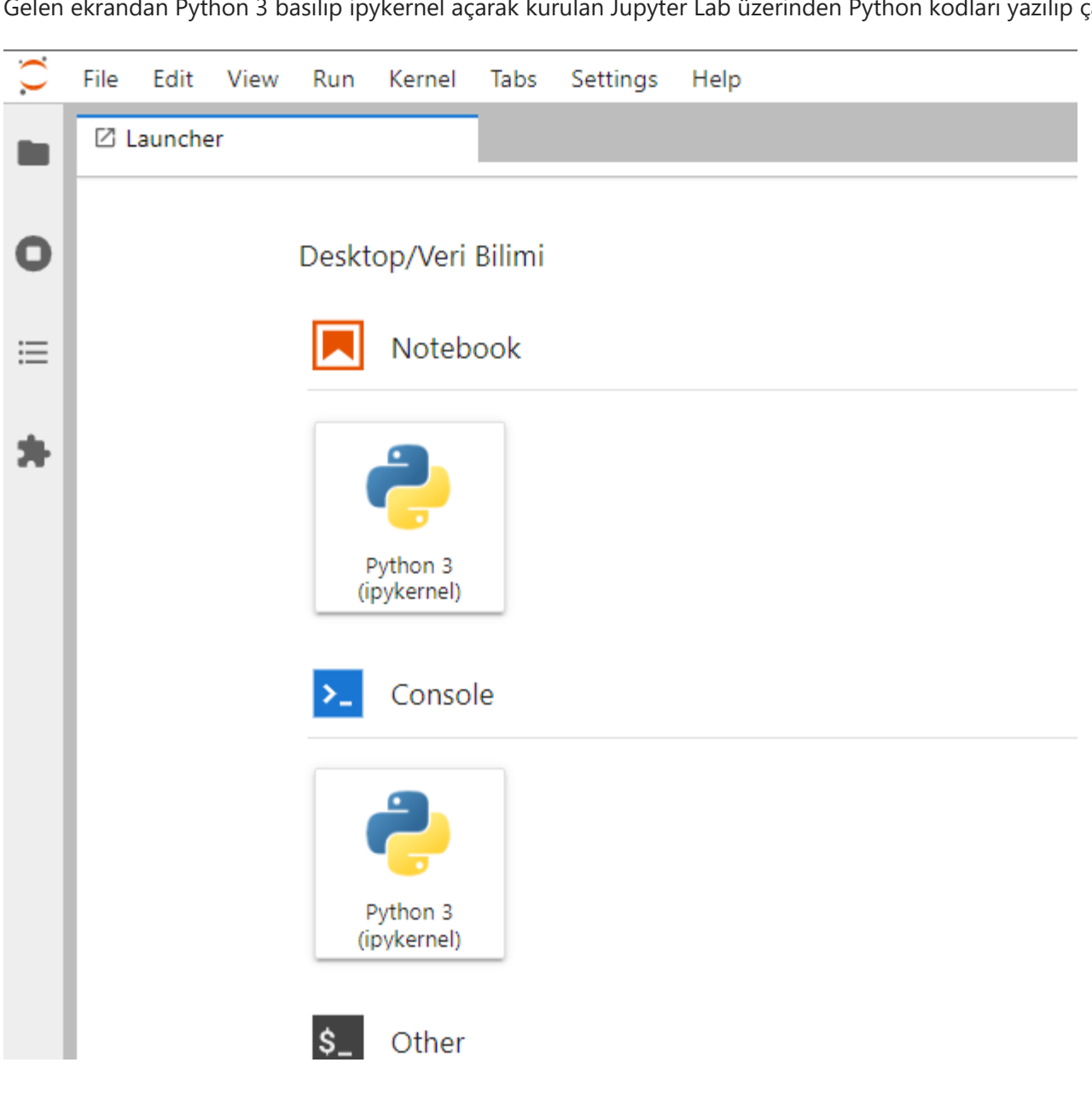
Jupyter Lab Kurulumu

Python kurulduktan sonra komut istemi çağırılarak **pip install jupyterlab** komutu yazılıp enter tuşuna basılarak jupyterlab kurulumu gerçekleştirilir.



Kurulan **Jupyter Lab** arayüzünü kullanabilmek için tekrar komut istemi çağırılır ve **jupyter lab** komutu yazılır ve kullanılan tarayıcı tarafından Jupyter ekranına yönlendirilir.

Gelen ekrandan Python 3 basılıp ipykernel açarak kurulan Jupyter Lab üzerinden Python kodları yazılıp çalıştırılabilir.



Kullanılan Paketlerin Kurulması

Komut istemi çağırılarak **pip install numpy**, **pip install pandas** ve **pip install matplotlib** yazılarak bu paketler kurulur. Kurulan paketleri kullanabilmek için **import numpy**, **import pandas** ve **import matplotlib** yazılarak içeri aktarılmalıdır.

3. NUMPY PAKETİ

NumPy veri analizi alandaki başlıca paketlerden, kütüphanelerden biridir. Paket adı Numeric Python kemelerinin kısaltmalarında oluşmaktadır. Büyük boyutlu ve çok boyutlu dizileri ve matrisleri işlemek için tasarlanmıştır. NumPy Paketi listelerde matematiksel işlem yapabilmek için geliştirilmiştir. Yüksek düzeyde matematiksel fonksiyonlar yardımıyla bu nesnelerle çeşitli işlemlerin gerçekleştirilmesini sağlar.

Bu pakette temel veri yapısı listeler yerine NumPy dizileridir (NumPy array). NumPy dizileri sadece tek türden veriler içerebilmektedir. Bir NumPy dizisinde farklı türden veri tanımlanırsa bu farklı veriler aynı veri tiplerine dönüştürülür. Örneğin diziye sayısal ve metin verileri tanımlanırsa tüm sayısal veriler metin veri tipine çevrilir.

NumPy paketini kullanabilmek için **import numpy** yazılarak paket içeri aktarılmalıdır. Paketi içeri aktardıktan sonra pakette yer alan fonksiyon ve metodları kullanabilmek için paket ismi ve sonrasında nokta (numpy.) yazılmalıdır. Her defasında NumPy yazmak zor geliyorsa **as np** olarak kısaltılabilmektedir.

```
In [2]: import numpy as np
        dizi = np.array(["Regresyon", "Optimizasyon", 1,2,3])
        print(dizi)

        ['Regresyon' 'Optimizasyon' '1' '2' '3']
```

Hem sayısal hem de metin veri tipinde oluşturulan dizinin tüm sayısal verileri görüldüğü üzere metin veri tipine dönüştürüldü.

Python'da oluşturulan herhangi bir liste **.array()** metodu kullanarak NumPy dizisine dönüştürülebilir.

```
In [3]: import numpy as np
        liste = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]
        print(liste)
        type(liste)

        [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
Out[3]: list

In [4]: liste_np = np.array(liste)
        print(liste_np)
        type(liste_np)

        [ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20]
Out[4]: numpy.ndarray
```

Liste dizisi NumPy array veri tipine dönüştürüldü. Python'daki diziler üzerinden işlem yapılmak istenirse, bu diziler NumPy array veri tipine dönüştürülmelidir. Dönüştürme yapmadan da bu diziler üzerinden işlemler gerçekleştirilebilir, fakat Python dizileri ve NumPy dizileri arasında bazı farklar mavcuttur. Örneğin Python'daki herhangi iki diziyi birleştirme işlemi "+" işlemi ile yapılabilir, fakat NumPy dizisine "+" işlemi uygulanırsa matematiksel toplama anlamına gelir ve iki dizinin aynı sıradaki elemanları aritmetik olarak toplanır. NumPy dizilerinde işlemler eleman bazında gerçekleşmektedir.

```
In [5]: liste_1 = [50,45,65,75,80,25,90,30]
        liste_2 = [45,78,64,95,50,30,70,75]
        liste_1 + liste_2

Out[5]: [50, 45, 65, 75, 80, 25, 90, 30, 45, 78, 64, 95, 50, 30, 70, 75]

In [6]: import numpy as np
        liste_1 = [50,45,65,75,80,25,90,30]
        liste_2 = [45,78,64,95,50,30,70,75]
        liste_1_np = np.array(liste_1)
        liste_2_np = np.array(liste_2)
        liste_1_np + liste_2_np

Out[6]: array([ 95, 123, 129, 170, 130,  55, 160, 105])
```

NumPy dizilerindende herhangi bir eleman seçme normal listelerle aynıdır. Aynı zamanda istenen şartları taşıyan elemanları seçtiirmek de mümkündür. Bunun için mantıksal operatörlerden faydalanılabilir.

```
In [7]: liste_1 = [50,45,65,75,80,25,90,30]
        liste_2 = [45,78,64,95,50,30,70,75]
        liste_1_np = np.array(liste_1)
        liste_2_np = np.array(liste_2)
        ortalama = (liste_1_np + liste_2_np)/2
        print(ortalama)

        [47.5 61.5 64.5 85.  65.  27.5 80.  52.5]
```

Hesaplanılan bu ortalama dizisinden sadece 60'ın altındaki değerler listelenmek istenirse **ortalama[ortalama < 60]** yazılabilir.

```
In [8]: ortalama[ortalama < 60]

Out[8]: array([47.5, 27.5, 52.5])
```

NumPy paketinde aynı zamanda for döngülerindeki gibi sayı dizileri oluşturulabilir. Bunun için **.arange(), .linspace() ve .logspace()** metodları kullanılabilir.

```
In [9]: np.arange(1,15)  # 1'den 14'e kadar olan sayıları yazar.

Out[9]: array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14])

In [10]: np.arange(1,15,3)  # 1'den 14'e kadar olan sayıları 3'er arttırarak yazar.

Out[10]: array([ 1,  4,  7, 10, 13])

In [11]: np.linspace(-1,5,10)  # -1 ve 5 arasında 10 eşit aralıklı sayı üretir.

Out[11]: array([-1.        , -0.33333333,  0.33333333,  1.        ,  1.66666667,
                 2.33333333,  3.        ,  3.66666667,  4.33333333,  5.        ])

In [12]: np.logspace(0,3,4)  # dizideki sayıları 10'nun üssüolarak yazar.

Out[12]: array([   1.,   10.,  100., 1000.])
```

Çok Boyutlu NumPy Dizileri

NumPy dizileri homojen,tümü aynı tipte çok boyutlu dizilerdir. NumPy'de boyutlara eksen denir. Veriler iki veya üç boyutlu olabileceği gibi çok daha fazla boyuta da sahip olabilir. Veri nesnesindeki herhangi bir veriye erişebilmek için boyutunun bilinmesi gerekmektedir. Örneğin bir listede herhangi bir listeye ulaşmak için listedeki sıra numarasını bilmek yeterlidir, fakat iki boyutlu bir veri setinde verilere erişebilmek için o verinin hangi satır ve sütunda olduğu bilinmelidir. Üç boyutlu bir veride bura ek olarak yüksekiğinin de bilinmesi gerekmektedir.

NumPy dizilerinde eleman seçimi **dizi[başlangıç:bitiş:adım]** şeklinde yapılır. Bu değerlerden herhangi biri girilmezse varsayılan olarak başlangıç için 0, bitiş için dizideki son endeks numarası ve adım için 1 değerini alır.

```
In [13]: dizi = np.arange(20)
         print(dizi)

         [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19]

In [14]: dizi[1:10:2]

Out[14]: array([1, 3, 5, 7, 9])

In [15]: dizi[:5]  # ilk 5 eleman alınır.

Out[15]: array([0, 1, 2, 3, 4])

In [16]: dizi[10:] # 10. endeks numarasından sonraki elemanları yazar.

Out[16]: array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19])

In [17]: dizi[10:15]  # 10. endeks numarasından başlayıp 15.endeks numarasına kadar olan değerleri yazar.

Out[17]: array([10, 11, 12, 13, 14])

In [18]: dizi[::2]  # baştan sona kadar ikişer atlayarak tüm elemanları yazar.

Out[18]: array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18])
```

Çok boyutlu dizilerde aynı kural , her boyut için yazılan eksenlerin arasına virgül koyularak seçilebilir.

```
In [19]: import numpy as np
         dizi = dizi.reshape(4,5)
         dizi

Out[19]: array([[ 0,  1,  2,  3,  4],
                [ 5,  6,  7,  8,  9],
                [10, 11, 12, 13, 14],
                [15, 16, 17, 18, 19]])

In [20]: dizi[2:3, 3:4] #ikinci satır ve üçüncu sütun

Out[20]: array([[13]])

In [21]: dizi[:3, :3] #üçüncü satır üçüncu sütuna kadar

Out[21]: array([[ 0,  1,  2],
                [ 5,  6,  7],
                [10, 11, 12]])
```

NumPy Dizileri ile İşlemler

İşlemleri yapabilmek için öncelikle bir dizi tanımlayalım.

```
In [22]: import numpy as np
         dizi = np.array([[[1,2,3,4],[5,6,7,8],[9,10,11,12]],
                          [[13,14,15,16],[17,18,19,20],[1,2,3,4]]])
         print(dizi)

         [[[ 1  2  3  4]
           [ 5  6  7  8]
           [ 9 10 11 12]]

          [[13 14 15 16]
           [17 18 19 20]
           [ 1  2  3  4]]]

In [23]: dizi.shape # dizinin boyutunu verir.

Out[23]: (2, 3, 4)
```

Üç boyutlu dizinin yüksekliği 2, satır sayısı 3 ve sütun sayısının 4 olduğu görülmektedir. NumPy dizisinin farklı eksenleri üzerindeki toplamları bulmak için **.sum** metodu kullanılır. Toplamı bulmak için istediğimiz eksen numarasını parantez içinde belirtmeliyiz. İki boyutlu dizilerde sütun 0, satır 1 endeksi ile gösterilir. Üç boyutlu dizilerde ise yükseklik 0, sütun 1 ve satır 2 endeks numarası ile gösterilir. Birikimli sıklıkları bulmak için ise **.cumsum** metodu kullanılır.

```
In [24]: dizi.sum(0)  # yükseklik toplamları

Out[24]: array([[14, 16, 18, 20],
                [22, 24, 26, 28],
                [10, 12, 14, 16]])

In [25]: dizi.sum(1)   # sütun toplamları

Out[25]: array([[15, 18, 21, 24],
                [31, 34, 37, 40]])

In [26]: dizi.sum(2)   # satır toplamları

Out[26]: array([[10, 26, 42],
                [58, 74, 10]])
```

Verilen eksendeki...

dizi.max() : maksimum değeri

dizi.argmax() : maksimum değerlerin sıra numarası

dizi.min() : minimum değer

dizi.argmin() : minimum değerlerin sıra numarası

dizi.ptp() : maksimum ve minimum değerler arsındaki fark

dizi.mean() : ortalama değerleri hesaplar

dizi.var() : varyans değerlerini hesaplar

dizi.std() : standart sapma değerlerini hesaplar

dizi.prod() : sayıların çarpımını hesaplar

dizi.sort() : verilerin sıralanmış halini iletir

dizi.cumprod(): sayıların birikimli çarpımlarını hesaplar

dizi.round(a) : dizide yer alan sayıları a ondalık sayıya yuvarlar

dizi.trace() : dizideki köşegenler toplamını hesaplar

dizi.clip(min,max) : dizide yer alan min ve max değerleri aynen, min'den düşük değerleri min, max'tan büyük değerleri ise max olarak iletir.

```
In [27]: dizi = np.array([[1.3,4.5],[2.2,8.1],[1.1,3.3],[4.2,5.1],[3.2,6.3]])
         dizi

Out[27]: array([[1.3, 4.5],
                [2.2, 8.1],
                [1.1, 3.3],
                [4.2, 5.1],
                [3.2, 6.3]])

In [28]: np.sum([dizi[:,1]]) # ikinci sütundaki sayıların toplamı

Out[28]: 27.3

In [29]: np.mean([dizi[:,1]]) # ikinci sütundaki sayıların ortalaması

Out[29]: 5.46

In [30]: np.median([dizi[1,:]]) # ikinci satırdaki sayıların medaynı

Out[30]: 5.15

In [31]: np.std(dizi[:,1]) # ikinci sütun standart sapma

Out[31]: 1.6365818036383026

In [32]: np.corrcoef(dizi[:,0],dizi[:,1]) # birinci ve ikinci sütunların korelasyon katsayıları

Out[32]: array([[1.        , 0.35158017],
                [0.35158017, 1.        ]])
```

NumPy dizilerinde aritmetik işlemler de yapılabilir.

```
In [33]: x = np.array([[2,4,6,8],[10,12,14,16],[3,4,5,6]])
         y = np.array([[3,6,4,5],[1,2,3,7],[20,21,22,23]])
         x + y

Out[33]: array([[ 5, 10, 10, 13],
                [11, 14, 17, 23],
                [23, 25, 27, 29]])

In [34]: x-y
         x/y
         x*y

Out[34]: array([[  6,  24,  24,  40],
                [ 10,  24,  42, 112],
                [ 60,  84, 110, 138]])
```

NumPy dizileri matris çarpımları için **.npmatmul()** metodu uygulanabilir.

```
In [35]: x = np.array([[2,4,6,8],[10,12,14,16],[3,4,5,6]])
         y = np.array([[3,6,4,5],[1,2,3,7],[20,21,22,23],[1,2,3,4]])
         np.matmul(x,y)

Out[35]: array([[138, 162, 176, 208],
                [338, 410, 432, 520],
                [119, 143, 152, 182]])
```

NumPy Metodları

np.reshape NumPy dizisinde dizinin boyutlarını yeniden belirlemek için kullanılır. **np.amax()** bir NumPy dizisinde istenen eksendeki maksimum değerleri iletir. **np.amin()** bir NumPy dizisinde istenen eksendeki minimum değerleri iletir.

```
In [36]: x = np.random.uniform(low=1,high=50,size=16).reshape(4,4)
         x
         np.max(x, axis=0) # her sütundaki max değeri verir.

Out[36]: array([47.45697462, 46.87293946, 46.25062578, 39.75680164])

In [37]: np.max(x, axis=1) # her satırdaki max değeri verir.

Out[37]: array([44.08929608, 45.63611741, 47.45697462, 46.87293946])
```

np.argsort() bir dizinin elemanları,dizi küçükten büyüğe sıralandığında kaçıncı sırada olacağını belirtir.

```
In [38]: y = np.array([5,4,8,10,1,6,3,0,9])
         y
         y.argsort()

Out[38]: array([7, 4, 6, 1, 0, 5, 2, 8, 3], dtype=int64)
```

np.concatenate() iki Numpy dizisini istenen eksenlerden birleştirmek için kullanılır. Örneğin iki boyutlu iki diziyi birleştirirken axis=0 seçilirse satırları yani alt alta, axis=1 seçilirse sütunları yan yana birleştirme işlemi gerçekleştirilir.

```
In [39]: x = np.arange(2,8).reshape(3,2)
         x

Out[39]: array([[2, 3],
                [4, 5],
                [6, 7]])

In [40]: y = np.arange(6,12).reshape(3,2)
         y

Out[40]: array([[ 6,  7],
                [ 8,  9],
                [10, 11]])

In [41]: np.concatenate([x,y], axis=0)

Out[41]: array([[ 2,  3],
                [ 4,  5],
                [ 6,  7],
                [ 6,  7],
                [ 8,  9],
                [10, 11]])

In [42]: np.concatenate([x,y], axis=1)

Out[42]: array([[ 2,  3,  6,  7],
                [ 4,  5,  8,  9],
                [ 6,  7, 10, 11]])
```

np.full() istenen boyutta, değerde ve veri türünde NumPy dizisi oluşturur. Veri türü argümanı (dtype) seçime bağlıdır.

```
In [43]: np.full((3,2), 7.7)

Out[43]: array([[7.7, 7.7],
                [7.7, 7.7],
                [7.7, 7.7]])
```

np.intersect1d() iki veya daha fazla NumPy dizisindeki ortak elemanları alır.

```
In [44]: x = [1,2,3,4,5]
         y = [5,6,1,3,8]
         np.intersect1d(x,y)

Out[44]: array([1, 3, 5])
```

np.isin() aranan elemanların bir liste veya dizide olup olmadığını gösterir. Listede olanları True, olmayanları False olarak gösterir.

```
In [45]: x = [1,2,3,4,5,10,14,78,36,25,9]
         liste = [5,6,1,3,8]
         np.isin(x,liste)

Out[45]: array([ True, False,  True, False,  True, False, False, False, False,
                False, False])
```

np.isnan() bir dizideki verilerin NaN olup olmadığını göstermek için kullanılır (Not a Number : bir sayı değil).

x = np.log([2,4,-1]) x

np.setdiff1d() bir dizide olup ikinci dizede olmayan elemanları seçmek için kullanılabilir.

```
In [46]: x = [1,2,3,4,5,10,9]
         liste = [5,6,1,3,8]
         np.setdiff1d(x,y)

Out[46]: array([ 2,  4,  9, 10])
```

np.zeros() istenen boyutta sıfırlardan meydana gelen NumPy dizisi oluşturulabilir.

```
In [47]: np.zeros(5)

Out[47]: array([0., 0., 0., 0., 0.])
```

np.ones() istenen boyutta birlerden meydana gelen NumPy dizisi oluşturulabilir.

```
In [48]: np.ones(5)

Out[48]: array([1., 1., 1., 1., 1.])
```

np.unique() bir dizideki tekil değerleri gösterir.

```
In [49]: x = [1,2,3,4,5,10,14,78,36,25,9,1,6,4,5,2]
         np.unique(x)

Out[49]: array([ 1,  2,  3,  4,  5,  6,  9, 10, 14, 25, 36, 78])
```

np.repeat() tekrar eden elemanlardan bir dizi oluşturur.

```
In [50]: np.repeat(1,3)

Out[50]: array([1, 1, 1])
```

np.where() bir NumPy dizisinde istenen şartları sağlayan elemanları seçer.

```
In [51]: dizi = np.arange(1,100)
         np.where(dizi % 10 == 0)

Out[51]: (array([ 9, 19, 29, 39, 49, 59, 69, 79, 89], dtype=int64),)
```

Bunlar dışında çok sayıda NumPy metodu bulunmaktadır, https://numpy.org/learn/ adresinden daha detaylı incelenebilir.

4. PANDAS PAKETİ

NumPy Paketi veri saklama açısından hızlı ve etkin bir paket, fakat farklı türde verilerin kullanılması ve verilerden seçim yapma konusunda yetersiz kalabilmeltedir. Pandas Paketi ise hem yüksek performanslı hem de esnek bir veri analizi bakımında oldukca kullanışlıdır.

Pandas'ta iki temel veri nesnesi esas alınmaktadır. Bunlar seriler ve veri çerçeveleridir.

Pandas Serileri

Pandas serileri endekslenmiş tek boyutlu diziler olarak düşünülebilir. Pandas paketini kullanabilmek için **import pandas** yazılarak paket içeri aktarılmalıdır. Paketi içeri aktardıktan sonra pakette yer alan fonksiyon ve metodları kullanabilmek için paket ismi ve sonrasında nokta yazılmalıdır. Aynı şekilde her defasında Pandas yazmak zor geliyorsa **as pd** olarak kısaltılabilir.

```
In [52]: import pandas as pd
         pandas_seri = pd.Series([10,6,1,2,8,3])
         pandas_seri

Out[52]: 0    10
         1     6
         2     1
         3     2
         4     8
         5     3
         dtype: int64

In [53]: type(pandas_seri)

Out[53]: pandas.core.series.Series
```

Pandas Serilerinin endeksleri varsayılan olarak 0'dan başlayan ardışık tamsayılardır, fakat isteğe göre farklı endeks isimleri de kullanılabilmektedir.

```
In [54]: pandas_seri = pd.Series([1,2,3,4,5],
                                 index = ['i','i+1','i+2','i+3','i+4'])
         pandas_seri

Out[54]: i      1
         i+1    2
         i+2    3
         i+3    4
         i+4    5
         dtype: int64
```

Serilerde arka arkaya gelen verileri seçmek de mümkündür.

```
In [55]: pandas_seri['i+1':'i+3']

Out[55]: i+1    2
         i+2    3
         i+3    4
         dtype: int64
```

Pandas Veri Çerçeveleri

Pandas paketinde iki boyutlu veri kümeleri "DataFrame" (veri çerçevesi) olarak adlandırılmaktadır. Bu veri çerçeveleri Python'da veri analizini kolaylaştıran kullanışlı veri yapılarıdır. Veri çerçeveleri iki boyutlu veri kümesi özelliklerini taşımaktadır. Her satırda bir gözlem, her sütunda ise farklı değişken yer almaktadır. Bir sütunda yer alan tüm veriler NumPy'dakinin aksine farklı veri tiplerinden oluşabilir. Bu yönüyle veri çerçeveleri ile betimsel ve çıkarımsal veri analizi yapmak, modeller kurmak ve verileri görselleştirmek daha kolaydır. Bu nedenle veri biliminde bu paketin kullanımı oldukca yaygındır.

Pandas DataFrame oluşturmak için farklı yöntemler kullanılmaktadır. Bunlardan yaygın olarak kullanılan sözlük veri yapısıdır. Örneğin bir okuldaki öğrencilerle ilgili bilgiler aşağıdaki şekilde girilebilir.

```
In [56]: Ogrenci_dict = {
             'Mat_Notları':[35,45,80,50,70,15,30,60,95,80,85],
             'Fen_Notları':[50,78,45,65,95,90,90,70,74,78,96],
             'Türkce_Notları':[80,75,78,74,20,56,48,95,62,36,90],
             'Sınıflar' : ['1.sınıf','2.sınıf','3.sınıf','1.sınıf','1.sınıf','3.sınıf','1.sınıf','2.sınıf','3.sınıf','3.
         }
         import pandas as pd
         Notlar = pd.DataFrame(Ogrenci_dict)
         print(Notlar)

            Mat_Notları  Fen_Notları  Türkce_Notları Sınıflar
         0           35           50              80  1.sınıf
         1           45           78              75  2.sınıf
         2           80           45              78  3.sınıf
         3           50           65              74  1.sınıf
         4           70           95              20  1.sınıf
         5           15           90              56  3.sınıf
         6           30           90              48  1.sınıf
         7           60           70              95  2.sınıf
         8           95           74              62  3.sınıf
         9           80           78              36  3.sınıf
         10          85           96              90  2.sınıf
```

Aynı şekilde liste yapıları da Pandas veri çerçevesine dönüştürülebilir, fakat bu biraz daha uzun sürmektedir. Çünkü bu işlemi yapmak için öncelikle veri çerçevesi sütunları listeler olarak tanımlanır, sütun isimleri ve sütunları zip fonksiyonunu kullanılarak tuple verilerden oluşan bir listeye çevrilir, daha sonra zip veri yapısı sözlüğe ve en son olarak da sözlük pandas veri çerçevesine dönüştürülür.

```
In [57]: Mat_Notları = [35,45,80,50,70,15,30,60,95,80,85]
         Fen_Notları = [50,78,45,65,95,90,90,70,74,78,96]
         Türkce_Notları = [80,75,78,74,20,56,48,95,62,36,90]
         Sınıflar = ['1.sınıf','2.sınıf','3.sınıf','1.sınıf','1.sınıf','3.sınıf','1.sınıf','2.sınıf','3.sınıf','3.sınıf'
         list_name = ['Mat_Notları','Fen_Notları','Türkce_Notları','Sınıflar']
         sutunlar = [Mat_Notları,Fen_Notları,Türkce_Notları]
         zip_veri = list(zip(list_name, sutunlar))
         veri = dict(zip_veri)
         print(veri)

         {'Mat_Notları': [35, 45, 80, 50, 70, 15, 30, 60, 95, 80, 85], 'Fen_Notları': [50, 78, 45, 65, 95, 90, 90, 70, 7
         4, 78, 96], 'Türkce_Notları': [80, 75, 78, 74, 20, 56, 48, 95, 62, 36, 90]}

In [58]: veri_df = pd.DataFrame(veri)
         print(veri_df)

            Mat_Notları  Fen_Notları  Türkce_Notları
         0           35           50              80
         1           45           78              75
         2           80           45              78
         3           50           65              74
         4           70           95              20
         5           15           90              56
         6           30           90              48
         7           60           70              95
         8           95           74              62
         9           80           78              36
         10          85           96              90
```

DataFrame yeni bir sütun eklenebilir.

```
In [59]: veri_df['Dönem'] = 'Güz'
         print(veri_df)

            Mat_Notları  Fen_Notları  Türkce_Notları Dönem
         0           35           50              80   Güz
         1           45           78              75   Güz
         2           80           45              78   Güz
         3           50           65              74   Güz
         4           70           95              20   Güz
         5           15           90              56   Güz
         6           30           90              48   Güz
         7           60           70              95   Güz
         8           95           74              62   Güz
         9           80           78              36   Güz
         10          85           96              90   Güz
```

Gündelik hayatta kullanılan veri setleri genelde excel, csv formatlarda gelir. Csv formatındaki veri setini pandas formatında okutmak için pandas paketindeki **.read_csv()** metodu kullanılmaktadır. Veri seti okutulurken satır isimleri için "index_col=0" yazılmalıdır.

```
In [60]: import pandas as pd
         pokemon = pd.read_csv("C:/Users/Dilan/Desktop/pokemon.csv")
         pokemon
```

Out[60]:

	#	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	1	Bulbasaur	Grass	Poison	45	49	49	65	65	45	1	False
1	2	Ivysaur	Grass	Poison	60	62	63	80	80	60	1	False
2	3	Venusaur	Grass	Poison	80	82	83	100	100	80	1	False
3	4	Mega Venusaur	Grass	Poison	80	100	123	122	120	80	1	False
4	5	Charmander	Fire	NaN	39	52	43	60	50	65	1	False
...	...	...	...	...	...	...	...	...	...	...	...	...
795	796	Diancie	Rock	Fairy	50	100	150	100	150	50	6	True
796	797	Mega Diancie	Rock	Fairy	50	160	110	160	110	110	6	True
797	798	Hoopa Confined	Psychic	Ghost	80	110	60	150	130	70	6	True
798	799	Hoopa Unbound	Psychic	Dark	80	160	60	170	130	80	6	True
799	800	Volcanion	Fire	Water	80	110	120	130	90	70	6	True

800 rows × 12 columns

Aynı ekilde bu veri setindeki değişkenler kullanılarak Pandas Serisi ve DataFrame oluşturulabilir.

```
In [61]: series = pokemon["Defense"]
         print(type(series))

         <class 'pandas.core.series.Series'>

In [62]: series = pokemon["Defense"]
         print(type(series))

         <class 'pandas.core.series.Series'>

In [63]: data_frame = pokemon[["Defense"]]
         print(type(data_frame))

         <class 'pandas.core.frame.DataFrame'>
```

Veri seti **.melt()** metodu ile yeniden düzenlenebilir (pandas.melt(frame, id_vars=None, value_vars=None, var_name=None, value_name='value', col_level=None, ignore_index=True)).

```
In [64]: pokemon_melt = pd.melt(frame = pokemon, id_vars = "Name", value_vars=["Attack", "Defense"]) # Veriyi belirlenen
         pokemon_melt
```

Out[64]:

	Name	variable	value
0	Bulbasaur	Attack	49
1	Ivysaur	Attack	62
2	Venusaur	Attack	82
3	Mega Venusaur	Attack	100
4	Charmander	Attack	52
...	...	...	...
1595	Diancie	Defense	150
1596	Mega Diancie	Defense	110
1597	Hoopa Confined	Defense	60
1598	Hoopa Unbound	Defense	60
1599	Volcanion	Defense	120

1600 rows × 3 columns

```
In [65]: type(pokemon_melt)

Out[65]: pandas.core.frame.DataFrame
```

Veri çerçevesinin hangi sütunlardan oluştuğunu görmek için **.columns** kullanılabilir.

```
In [66]: pokemon_melt.columns

Out[66]: Index(['Name', 'variable', 'value'], dtype='object')
```

Veri çerçevesindeki bu sütun isimleri isteğe bağlı olarak değiştirilebilir.

```
In [67]: pokemon_melt.columns = ['Pokemon_Adı','Saldırı_Türü','Puan']
         print(pokemon_melt)

                  Pokemon_Adı Saldırı_Türü  Puan
         0          Bulbasaur       Attack    49
         1            Ivysaur       Attack    62
         2           Venusaur       Attack    82
         3      Mega Venusaur       Attack   100
         4         Charmander       Attack    52
         ...              ...          ...   ...
         1595         Diancie      Defense   150
         1596    Mega Diancie      Defense   110
         1597  Hoopa Confined      Defense    60
         1598   Hoopa Unbound      Defense    60
         1599       Volcanion      Defense   120

         [1600 rows x 3 columns]
```

Pokemon_melt veri çerçevesinin yapısını incelelemek için **.head()** metodu kullanılabilir. Bu metodta argüman olarak görmek istenilen satır sayısı yazılır.

```
In [68]: pokemon_melt.head(8)
```

Out[68]:

	Pokemon_Adı	Saldırı_Türü	Puan
0	Bulbasaur	Attack	49
1	Ivysaur	Attack	62
2	Venusaur	Attack	82
3	Mega Venusaur	Attack	100
4	Charmander	Attack	52
5	Charmeleon	Attack	64
6	Charizard	Attack	84
7	Mega Charizard X	Attack	130

Son satırları görmek için **.tail()** metodu kullanılır. Bu metoda da aynı şekilde sondan kaç tane satır görmek isteniliyorsa girilir.

In [69]:
```
pokemon_melt.tail(8)
```

Out[69]:

	Pokemon_Adı	Saldırı_Türü	Puan
1592	Xerneas	Defense	95
1593	Yveltal	Defense	95
1594	Zygarde Half Forme	Defense	121
1595	Diancie	Defense	150
1596	Mega Diancie	Defense	110
1597	Hoopa Confined	Defense	60
1598	Hoopa Unbound	Defense	60
1599	Volcanion	Defense	120

Veri çerçevesine ait bilgiler **.info()** metodu ile incelenebilir. Bu metod istatistiksel özellikler yerine sütunların veri tipi, veri çerçevesindeki satır sayısı ve her bir sütundaki veri sayısı gibi değerleri verir.

In [70]:
```
pokemon_melt.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1600 entries, 0 to 1599
Data columns (total 3 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Pokemon_Adı   1598 non-null   object
 1   Saldırı_Türü  1600 non-null   object
 2   Puan          1600 non-null   int64
dtypes: int64(1), object(2)
memory usage: 37.6+ KB
```

Veri çerçevelerini değiştirmek istemiyorsak başka bir veri çerçevesine kopyalanarak aktaralabilir.

In [71]:
```
pokemon_new = pokemon_melt.copy()
pokemon_new
```

Out[71]:

	Pokemon_Adı	Saldırı_Türü	Puan
0	Bulbasaur	Attack	49
1	Ivysaur	Attack	62
2	Venusaur	Attack	82
3	Mega Venusaur	Attack	100
4	Charmander	Attack	52
...	...	...	...
1595	Diancie	Defense	150
1596	Mega Diancie	Defense	110
1597	Hoopa Confined	Defense	60
1598	Hoopa Unbound	Defense	60
1599	Volcanion	Defense	120

1600 rows × 3 columns

Aynı şekilde veri çerçevelerini istersek **.values** özellği kullanarak NumPy dizisine çevirebiliriz. Pandas paketi NumPy paketi üzerinden geliştirildiğinden, NumPy metodları pandas veri çerçevlerine de uygulanabilmektedir.

In [72]:
```
np_dizi = pokemon_melt.values
np_dizi
```

Out[72]:
```
array([['Bulbasaur', 'Attack', 49],
       ['Ivysaur', 'Attack', 62],
       ['Venusaur', 'Attack', 82],
       ...,
       ['Hoopa Confined', 'Defense', 60],
       ['Hoopa Unbound', 'Defense', 60],
       ['Volcanion', 'Defense', 120]], dtype=object)
```

İki veri çerçevesini anahtar bir sütun kullanarak birleştirmek için **.merge()** fonksiyonu kullanılabilir.

In [73]:
```
veri_1 = np.array([['Dilan', 52], ['Sena', 65], ['Abuzettin', 70], ['Ahmet', 66]])
sutun_isimleri = ['uzunluk', 'agirlik']
df1 = pd.DataFrame(data=veri_1, columns=sutun_isimleri)
print(df1)
```

```
     uzunluk agirlik
0      Dilan      52
1       Sena      65
2  Abuzettin      70
3      Ahmet      66
```

In [74]:
```
veri_2 = np.array([['Dilan', 52], ['Sema', 55], ['Buğra', 70], ['Ahmet', 76]])
sutun = ['uzunluk', 'agirlik']
df2 = pd.DataFrame(data=veri_2, columns=sutun)
print(df2)
```

```
  uzunluk agirlik
0   Dilan      52
1    Sema      55
2   Buğra      70
3   Ahmet      76
```

İki veri çerçevesi birleştirilirken **'left'**, **'right'**, **'inner'** ve **'outer'** yöntemleri kullanabilir. Veri çerçevesi birleştirilirken hangi sütunun dikkate alınacağı On ile belirtilir. Veri çerçevesinden birinde olmayan değerler için NaN atanır.

In [75]:
```
df3 = pd.merge(df1, df2, how='left', on='uzunluk')    # önce yazılan veri çerçevesi esas alınır. Burada df1 esa
df3
```

Out[75]:

	uzunluk	agirlik_x	agirlik_y
0	Dilan	52	52
1	Sena	65	NaN
2	Abuzettin	70	NaN
3	Ahmet	66	76

In [76]:
```
df4 = pd.merge(df1, df2, how='right', on='uzunluk')
df4
```

Out[76]:

	uzunluk	agirlik_x	agirlik_y
0	Dilan	52	52
1	Sema	NaN	55
2	Buğra	NaN	70
3	Ahmet	66	76

how='inner seçildiğinde birleştirilecek sütunun sadece her iki veri çerçevesinde de yer alan satırları yani kesişim kümesi dikkate alınır

In [77]:
```
df5 = pd.merge(df1, df2, how='inner', on='uzunluk')   # önce yazılan veri çerçevesi esas alınır. Burada df1 esa
df5
```

Out[77]:

	uzunluk	agirlik_x	agirlik_y
0	Dilan	52	52
1	Ahmet	66	76

how='outer' seçildiğinde birleştirilecek sütundaki tüm satırlar yani bileşim kümesi dikkate alınır.

In [78]:
```
df5 = pd.merge(df1, df2, how='outer', on='uzunluk')   # önce yazılan veri çerçevesi esas alınır. Burada df1 esa
df5
```

Out[78]:

	uzunluk	agirlik_x	agirlik_y
0	Dilan	52	52
1	Sena	65	NaN
2	Abuzettin	70	NaN
3	Ahmet	66	76
4	Sema	NaN	55
5	Buğra	NaN	70

Çok Boyutlu Pandas Veri Yapıları

Buraya kadarki gördüğümüz pandas seri ve veri çerçeveleri bir veya iki boyuttan oluşmaktaydı, fakat pandas verilerinde çoklu indeks eklenerek üç ve daha fazla boluytlu veri yapılarılarıda oluşturulabilmektedir.

In [79]:
```
# Pandas serisine çoklu indeks ekleme
import pandas as pd
indeks = [('218A',2018),('218A',2022),
          ('218B',2018),('218B',2022),
          ('218C',2018),('218C',2022),
          ('218D',2018),('218D',2022),
          ('218E',2018),('218E',2022),
          ('218F',2018),('218F',2022)]
fiyat = [150,180,100,195,200,500,120,300,450,600,350,700]
ürünler = pd.Series(fiyat, index=indeks)
ürünler
```

Out[79]:
```
(218A, 2018)    150
(218A, 2022)    180
(218B, 2018)    100
(218B, 2022)    195
(218C, 2018)    200
(218C, 2022)    500
(218D, 2018)    120
(218D, 2022)    300
(218E, 2018)    450
(218E, 2022)    600
(218F, 2018)    350
(218F, 2022)    700
dtype: int64
```

İndeks ekleyerek seride seçim yapılabilir.

In [80]:
```
ürünler[('218B',2018) : ('218E',2022)]
```

Out[80]:
```
(218B, 2018)    100
(218B, 2022)    195
(218C, 2018)    200
(218C, 2022)    500
(218D, 2018)    120
(218D, 2022)    300
(218E, 2018)    450
(218E, 2022)    600
dtype: int64
```

Çoklu indeks kullanımı aynı zamanda **MultiIndex** ile daha kolay yapılabilir.

In [81]:
```
indeks = pd.MultiIndex.from_tuples(indeks)
indeks
```

Out[81]:
```
MultiIndex([('218A', 2018),
            ('218A', 2022),
            ('218B', 2018),
            ('218B', 2022),
            ('218C', 2018),
            ('218C', 2022),
            ('218D', 2018),
            ('218D', 2022),
            ('218E', 2018),
            ('218E', 2022),
            ('218F', 2018),
            ('218F', 2022)],
           )
```

Satırlar için oluşturulan indeksler sütunlar için de oluşturulabilir.

In [82]:
```
indeks = pd.MultiIndex.from_product([['218A','218B','218C'], [2018,2022]])
sütunlar = pd.MultiIndex.from_product([['stok_1','stok_2','stok_3','stok_4'], ['Stok_kodu','fiyat']])
veri = np.array(['A123',150,'B123',180,'C123',100,'D123',195,
                 'E123',200,'F123',500,'A124',120,'B124',300,
                 'C124',450,'D124',600,'E124',350,'F124',700,
                 'A123',150,'B123',180,'C123',100,'D123',195,
                 'E123',200,'F123',500,'A124',120,'B124',300,
                 'C124',450,'D124',600,'E124',350,'F124',700])
veri = np.reshape(veri,(6,8))
ürünler = pd.DataFrame(veri, index=indeks, columns=sütunlar)
ürünler
```

Out[82]:

		stok_1		stok_2		stok_3		stok_4	
		Stok_kodu	fiyat	Stok_kodu	fiyat	Stok_kodu	fiyat	Stok_kodu	fiyat
218A	**2018**	A123	150	B123	180	C123	100	D123	195
	2022	E123	200	F123	500	A124	120	B124	300
218B	**2018**	C124	450	D124	600	E124	350	F124	700
	2022	A123	150	B123	180	C123	100	D123	195
218C	**2018**	E123	200	F123	500	A124	120	B124	300
	2022	C124	450	D124	600	E124	350	F124	700

Veri Çerçevesinde Seçim Yapma

Veri çerçevesindeki herhangi bir sütunu seçmek için **veri_cercevesi[sutun_ismi]** formatı kullanılır.

In [83]:
```
import pandas as pd
import numpy as np
dates = pd.date_range("20130101", periods=6)
df = pd.DataFrame(np.random.randn(6,4), index=dates, columns=list('ABCD'))
df.index = ['1.gün','2.gün','3.gün','4.gün','5.gün','6.gün']
df
```

Out[83]:

	A	B	C	D
1.gün	0.713797	1.463918	0.691080	-0.221787
2.gün	0.413574	1.820148	0.297391	-0.206780
3.gün	-1.530161	-0.385489	1.192284	0.882360
4.gün	-0.084747	1.051510	1.361572	1.258304
5.gün	-1.513685	-0.697736	0.594432	0.022762
6.gün	0.679573	-0.140734	0.054095	-0.841819

In [84]:
```
df['A']
```

Out[84]:
```
1.gün    0.713797
2.gün    0.413574
3.gün   -1.530161
4.gün   -0.084747
5.gün   -1.513685
6.gün    0.679573
Name: A, dtype: float64
```

In [85]:
```
type(df['A'])
```

Out[85]:
```
pandas.core.series.Series
```

Tek boyutlu seri olarak seçilmek isteniyorsa sütun ismi bir köşeli parantez içine alınır. Çekilen sütunun veri çerçevesi olarak alınması isteniyorsa sütun ismi iki köşeli parantez içinde yazılmalıdır.

In [86]:
```
df[['A']]
```

Out[86]:

	A
1.gün	0.713797
2.gün	0.413574
3.gün	-1.530161
4.gün	-0.084747
5.gün	-1.513685
6.gün	0.679573

In [87]:
```
type(df[['A']])
```

Out[87]:
```
pandas.core.frame.DataFrame
```

Birden fazla sütun çekmek istenirse iki adet köşeli parantez kullanılmalıdır.

In [88]:
```
df[['A','C']]
```

Out[88]:

	A	C
1.gün	0.713797	0.691080
2.gün	0.413574	0.297391
3.gün	-1.530161	1.192284
4.gün	-0.084747	1.361572
5.gün	-1.513685	0.594432
6.gün	0.679573	0.054095

Veri çerçevesindeki belirli satırlar seçilmek istenirse, istenilen satır numaralarını arada iki nokta olacak şekilde yazılır. Seçme işlemi yapılırken Python'da indeks numarsının sıfırdan başlandığı unutulmamalıdır.

In [89]:
```
df[2:4] # 2 indeks numarası ile başlayan ve 3 indeks numarsı ile biten saırlar seçilir.
```

Out[89]:

	A	B	C	D
3.gün	-1.530161	-0.385489	1.192284	0.882360
4.gün	-0.084747	1.051510	1.361572	1.258304

Pandas'da satır ve sütunları kolayca seçebilmek için **loc** ve **iloc** fonksiyonları kullanılabilir. Loc, location yani konum ve iloc ise inter location yani sayısal konum anlamına gelmektedir.

Satır isimleri ile seçim yapmak için **.loc[]** fonksiyonu kullanılır.

In [90]:
```
df.loc[['1.gün']]  # Burda da seçim yapılırken DataFrame formatında olması için çift köşeli parantez içine alın
```

Out[90]:

	A	B	C	D
1.gün	0.713797	1.463918	0.69108	-0.221787

Birden fazla satır da seçilebilir.

In [91]:
```
df.loc[['1.gün','3.gün']]
```

Out[91]:

	A	B	C	D
1.gün	0.713797	1.463918	0.691080	-0.221787
3.gün	-1.530161	-0.385489	1.192284	0.882360

Bir sütun ve satırdaki değeri almak için sütun ve satır isimlerini sırayla köşeli parantez içine alınmalıdır.

In [92]:
```
df['A']['3.gün']  # A sütunundaki 3.güne ait değeri verir.
```

Out[92]:
```
-1.5301613131590623
```

Aynı seçim farklı şekilde de yapılabilir.

In [93]:
```
df.A['3.gün']
```

Out[93]:
```
-1.5301613131590623
```

Birden fazla satır ve sütun seçimi yapmak da mümkündür. Bunun için istenilen satır ve sütunlar virgülle ayrılmış iki ayrı liste olarak iletilmelidir.

In [94]:
```
df.loc[["1.gün","3.gün"], ["A","C"]]
```

Out[94]:

	A	C
1.gün	0.713797	0.691080
3.gün	-1.530161	1.192284

Belirli değişkenlere ait tüm satırları seçmek için satır ismi yazmayıp bunun yerine **:** yazılması yeterlidir.

In [95]:
```
df.loc[:, ["A","B"]]
```

Out[95]:

	A	B
1.gün	0.713797	1.463918
2.gün	0.413574	1.820148
3.gün	-1.530161	-0.385489
4.gün	-0.084747	1.051510
5.gün	-1.513685	-0.697736
6.gün	0.679573	-0.140734

Satır ve sütun isimlerini bilinmiyorsa indeks numaraları ile seçim yapılabilir. Bunun için **iloc** fonksiyonu kullanılır. Buradaki tek fark satır ve sütunların isimleri yerine indeks numaraları ile seçim yapılır.

In [96]:
```
df.iloc[1]
```

Out[96]:
```
A    0.413574
B    1.820148
C    0.297391
D   -0.206780
Name: 2.gün, dtype: float64
```

In [97]:
```
df.iloc[[1]]
```

Out[97]:

	A	B	C	D
2.gün	0.413574	1.820148	0.297391	-0.20678

In [98]:
```
df.iloc[[1,2,3]]
```

Out[98]:

	A	B	C	D
2.gün	0.413574	1.820148	0.297391	-0.206780
3.gün	-1.530161	-0.385489	1.192284	0.882360
4.gün	-0.084747	1.051510	1.361572	1.258304

Burda da iki köşeli parantez kullanıldığına dikkat edilmelidir. Birden fazla satır ve sütunda seçim yapmak için loc fonksiyonundaki gibi fakat indeks numaraları yazarak seçim yapılır.

In [99]:
```
df.iloc[[1,2,3], [2,3]]
```

Out[99]:

	C	D
2.gün	0.297391	-0.206780
3.gün	1.192284	0.882360
4.gün	1.361572	1.258304

Sütun ismi ve indeks numaralarını birlikte kullanılabilmektedir.

In [100…
```
df['A'][0:3]
```

Out[100]:
```
1.gün    0.713797
2.gün    0.413574
3.gün   -1.530161
Name: A, dtype: float64
```

Satır veya sütunlar ters sırada yazdırılabilmektedir.

In [101…
```
df.loc['5.gün':'2.gün':-1]
```

Out[101]:

	A	B	C	D
5.gün	-1.513685	-0.697736	0.594432	0.022762
4.gün	-0.084747	1.051510	1.361572	1.258304
3.gün	-1.530161	-0.385489	1.192284	0.882360
2.gün	0.413574	1.820148	0.297391	-0.206780

Pandas veri çerçevesindeki bilgilere erişmek için pandas paketinde yer alan **.iterrows()** metodu kullanılabilir. Bu metodu for döngüsüyle kullanarak veri çerçevesindeki bilgilere erişilebilir.Sadece belirli sütunlardaki bilgilere de erişilebilir, ayrıca yeni sütunlar eklenebilir.

In [102…
```
for gun, satir in df.iterrows():
    print(gun)
    print(satir)
```

```
1.gün
A    0.713797
B    1.463918
C    0.691080
D   -0.221787
Name: 1.gün, dtype: float64
2.gün
A    0.413574
B    1.820148
C    0.297391
D   -0.206780
Name: 2.gün, dtype: float64
3.gün
A   -1.530161
B   -0.385489
C    1.192284
D    0.882360
Name: 3.gün, dtype: float64
4.gün
A   -0.084747
B    1.051510
C    1.361572
D    1.258304
Name: 4.gün, dtype: float64
5.gün
A   -1.513685
B   -0.697736
C    0.594432
D    0.022762
Name: 5.gün, dtype: float64
6.gün
A    0.679573
B   -0.140734
C    0.054095
D   -0.841819
Name: 6.gün, dtype: float64
```

Veri Analizi

Bir veri seti ile çalışılırken öncelikle veri seti anlamaya çalışılmalı, tanımlayıcı istatistikler incelenmelidir. Pandas paketinde bulunan **.describe()** metodu ile veri incelenebilir.

In [103…
```
pokemon.describe()
```

Out[103]:

	#	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation
count	800.0000	800.000000	800.000000	800.000000	800.000000	800.000000	800.000000	800.00000
mean	400.5000	69.258750	79.001250	73.842500	72.820000	71.902500	68.277500	3.32375
std	231.0844	25.534669	32.457366	31.183501	32.722294	27.828916	29.060474	1.66129
min	1.0000	1.000000	5.000000	5.000000	10.000000	20.000000	5.000000	1.00000
25%	200.7500	50.000000	55.000000	50.000000	49.750000	50.000000	45.000000	2.00000
50%	400.5000	65.000000	75.000000	70.000000	65.000000	70.000000	65.000000	3.00000
75%	600.2500	80.000000	100.000000	90.000000	95.000000	90.000000	90.000000	5.00000
max	800.0000	255.000000	190.000000	230.000000	194.000000	230.000000	180.000000	6.00000

Her bir sütun için yukarıdan aşağıya count veri sayısı, mean ortalama, std standart sapma, min en düşük değer, 25% (%25 yüzdelik), 50% medyan, 75% (%75 yüzdelik) ve max en yüksek değer verilir. Sayısal bilgilere ilişkin özet bilgiler ayrı ayrı hesaplamak da mümkündür.

In [104…
```
pokemon.count()
```

Out[104]:
```
#             800
Name          799
Type 1        800
Type 2        414
HP            800
Attack        800
Defense       800
Sp. Atk       800
Sp. Def       800
Speed         800
Generation    800
Legendary     800
dtype: int64
```

Sadece istenilen sütuna ait değerler de seçilebilir.

In [105…
```
pokemon[['Attack','Defense']].count()
```

Out[105]:
```
Attack     800
Defense    800
dtype: int64
```

In [106…
```
pokemon[['Attack','Defense']].std()
```

Out[106]:
```
Attack     32.457366
Defense    31.183501
dtype: float64
```

In [107…
```
pokemon[['Attack','Defense']].quantile(0.25)
```

Out[107]:
```
Attack     55.0
Defense    50.0
Name: 0.25, dtype: float64
```

In [108…
```
pokemon[['Attack','Defense']].quantile([0.25,0.75])
```

Out[108]:

	Attack	Defense
0.25	55.0	50.0
0.75	100.0	90.0

Burada sadece sayısal verilere ilişkin özet bilgiler verilir. Sayısal olmayan sütunlar için sütun ismi belirtilmelidir. Örneğin pokemon veri setindeki **Legendary** sütunu için özet bilgiler aşağıdaki şekilde görülebilir.

In [109…
```
pokemon['Legendary'].describe()
```

Out[109]:
```
count       800
unique        2
top       False
freq        735
Name: Legendary, dtype: object
```

Legendary sütununda 2 farklı değer bulunmaktadır, bunlardan en sık görülen False olduğu görülmektedir (top False). Freq ise en sık görülen False değerine ait 735 gözlem olduğunu belirtmektedir.

Sayısal olmayan değişkenler sınıf, tür, kategori belrten değişkenlere kategorik değişkenler denir. Kategorik değişkenleri görmek için **.unique()** metodu kullanılabilir.

In [110…
```
pokemon['Legendary'].unique()
```

Out[110]:
```
array([False,  True])
```

Sütunlara uygulanan metodlar satırlara da uygulanabilmektedir. Yani satır bazında min,max,ortalama vb. gibi değerler hesaplanabilir. Bunun için uygulanılan metod için **axis='columns'** argümanı belirtilmelidir.

In [111…
```
pokemon.count(axis='columns')
```

Out[111]:
```
0      12
1      12
2      12
3      12
4      11
       ..
795    12
796    12
797    12
798    12
799    12
Length: 800, dtype: int64
```

Kategorik değişkendeki her türe ait istatistiksel deeğerler ayrı ayrı incelenebilir. Bunun için mantıksal seçim kuralları uygulanabilir.

In [112…
```
pokemon.rename(columns ={'Type 1':'Type_1',
                         'Type 2':'Type_2'},inplace=True)    # Type 1 ve Type 2 isimleri rename ile Type_1 ve Typ

# Type_1'i Rock olanları seçer
veri = pokemon['Type_1']== 'Rock'
type(veri)
veri
```

Out[112]:
```
0      False
1      False
2      False
3      False
4      False
       ...
795     True
796     True
797    False
798    False
799    False
Name: Type_1, Length: 800, dtype: bool
```

In [113…
```
Rock = pokemon.loc[veri,:]
Rock.head()
```

Out[113]:

	#	Name	Type_1	Type_2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
80	81	Geodude	Rock	Ground	40	80	100	30	30	20	1	False
81	82	Graveler	Rock	Ground	55	95	115	45	45	35	1	False
82	83	Golem	Rock	Ground	80	120	130	55	65	45	1	False
103	104	Onix	Rock	Ground	35	45	160	30	45	70	1	False
149	150	Omanyte	Rock	Water	35	40	100	90	55	35	1	False

In [114…
```
Rock.describe()
```

Out[114]:

	#	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation
count	44.000000	44.000000	44.000000	44.000000	44.000000	44.000000	44.000000	44.000000
mean	432.840909	65.363636	92.863636	100.795455	63.340909	75.477273	55.909091	3.454545
std	237.258758	20.833884	35.325458	36.447209	28.249670	32.265965	29.903580	1.848375
min	81.000000	30.000000	40.000000	40.000000	10.000000	25.000000	10.000000	1.000000
25%	249.750000	50.000000	59.750000	71.500000	45.000000	50.000000	35.000000	2.000000
50%	398.000000	68.500000	95.000000	100.000000	60.000000	70.000000	50.000000	3.000000
75%	628.250000	80.000000	120.250000	120.500000	74.250000	90.500000	70.000000	5.000000
max	797.000000	123.000000	165.000000	200.000000	160.000000	150.000000	150.000000	6.000000

Metin tipi sütunlarda seçim yapabilmek için farklı bir yol olarak **.str** metodu da uygulanabilir.

In [115…
```
veri = pokemon['Type_1'].str.contains('Rock') # .contains('Rock'), Rock içerenleri almak için yazılır.
Rock = pokemon[veri]
Rock.head()
```

Out[115]:

	#	Name	Type_1	Type_2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
80	81	Geodude	Rock	Ground	40	80	100	30	30	20	1	False
81	82	Graveler	Rock	Ground	55	95	115	45	45	35	1	False
82	83	Golem	Rock	Ground	80	120	130	55	65	45	1	False
103	104	Onix	Rock	Ground	35	45	160	30	45	70	1	False
149	150	Omanyte	Rock	Water	35	40	100	90	55	35	1	False

Farklı koşullar veya birden fazla koşul kullanarak çoklu seçimler yapılabilir.

In [116…
```
pokemon[pokemon.Attack > 170]
```

Out[116]:

	#	Name	Type_1	Type_2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
163	164	Mega Mewtwo X	Psychic	Fighting	106	190	100	154	100	130	1	True
232	233	Mega Heracross	Bug	Fighting	80	185	115	40	105	75	2	False
424	425	Primal Groudon	Ground	Fire	100	180	160	150	90	90	3	True
426	427	Mega Rayquaza	Dragon	Flying	105	180	100	180	100	115	3	True
429	430	DeoxysAttack Forme	Psychic	NaN	50	180	20	180	20	150	3	True

In [117…
```
pokemon.Defense[pokemon.Attack > 170]
```

Out[117]:
```
163    100
232    115
424    160
426    100
429     20
Name: Defense, dtype: int64
```

Pandas paketinde bunlardan farklı metodlar da mevcuttur. Örneğin bir veri çerçevesinde hiç sıfır içermeyen sütunlar **.all**, sıfırdan faklı değerler içeren stunları görmek için **.any()**, NaN değeri içeren sütunları görmek için **.isnull()**, içermeyenleri görmek için **.notnull()** metodları mevcuttur. Daha ayrıntılı bilgi için https://pandas.pydata.org adresine bakılabilir.

5. VERİ GÖRSELLEŞTİRME

Veri ile çalışmanın en önemli bölümlerden birisi de verinin görselleştirilmesidir. Yapılan çalışmalarda kullanılan veriyi anlamak ve anlamlandırmak için öncelikle veriyi özetlemek ve görselleştirmek çok önemlidir. Verinin görselleştirilmesi yapılan çalışmaları başkalarına aktarmanın en etkili yoludur.

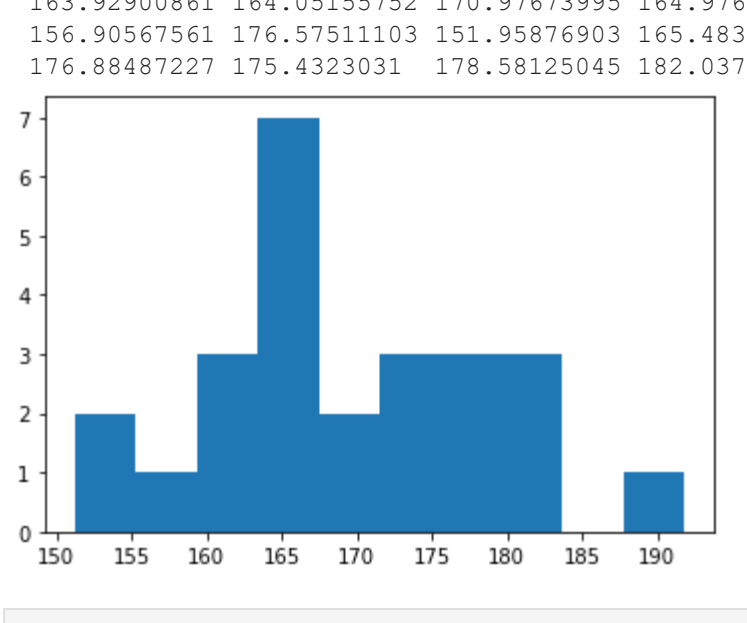
Python'da veri görselleştirmek için kullanılan en önemli paketlerden biri **Matplotlib** paketidir. Bu paketten yararlanarak veri görselleştirmesi için histogram, box plot, violin, çizgi, nokta grafikleri, iki değişkenli grafikler ve daha pek çok grafik çizimleri yapılabilmektedir.

Histogram

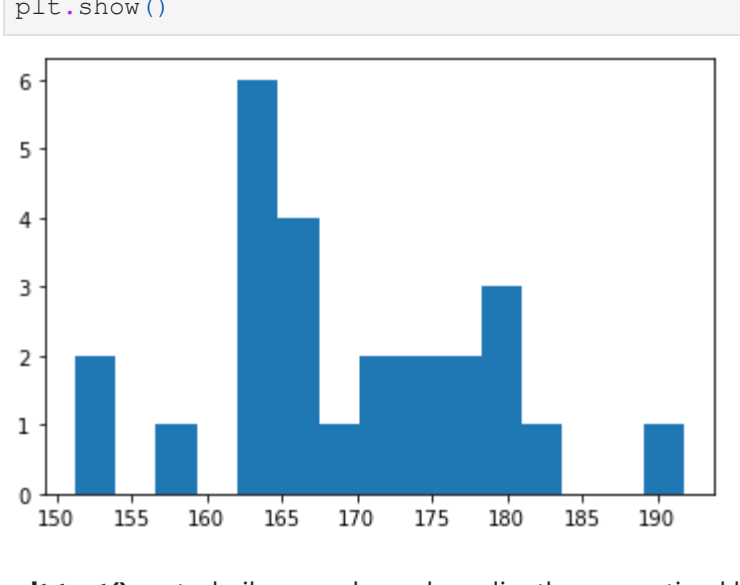
Histogram, frekans dağılımlarını gösteren bir grafiktir. Verilen her aralıktaki gözlem sayısını gösteren bir grafiktir. Python'da histogram çizdirmek için matplot.pyplot paketinde yer alan **.hist()** metodu kullanılır. Bu metodda histogram çizdirilirken **bins=** argümanı ile veriler istenilen aralığa ayrılabilmektedir.

In [118…
```
import matplotlib.pyplot as plt
import numpy as np

x = np.random.normal(170, 10, 25)
print(x)

plt.hist(x)
plt.show()
```

```
[171.59260373 166.22630707 151.19434959 163.12122394 191.80837739
 162.87445406 180.56683521 165.27438858 162.04877125 173.27595519
 163.92900861 164.05155752 170.97673995 164.97686752 164.15697273
 156.90567561 176.57511103 151.95876903 165.48347434 179.67609839
 176.88487227 175.4323031  178.58125045 182.03745178 168.53374378]
```

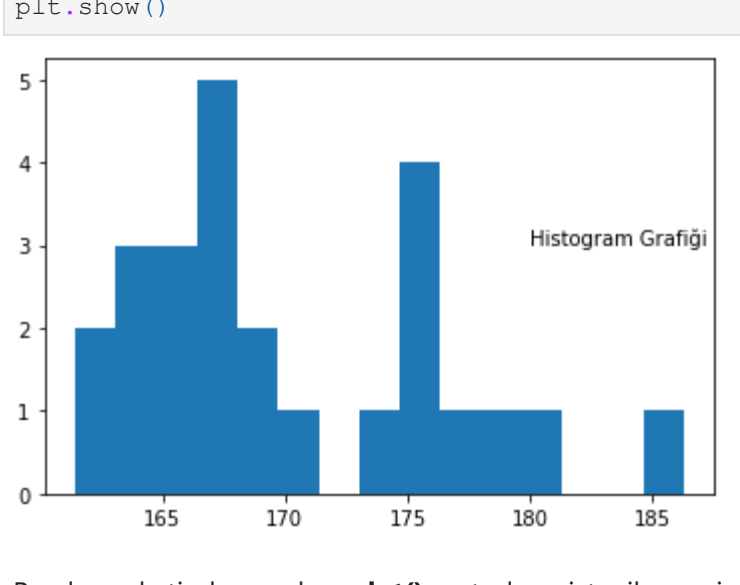


In [119…
```
plt.hist(x, bins=15)
plt.show()
```



plt.text() metodu ile sırasıyla x-y koordinatları ve metin eklenebilir. Aynı zamanda **plt.annotate()** metodu ile grafiğe metin ekenebilir.

In [120…
```
import matplotlib.pyplot as plt
import numpy as np

x = np.random.normal(170, 10, 25)

plt.hist(x, bins=15)
plt.text(180,3, 'Histogram Grafiği')
plt.show()
```



Pandas paketinde yer alan **.plot()** metodunu istenilen veri çerçevesine uygulayarak histogram çizdirilebilir.

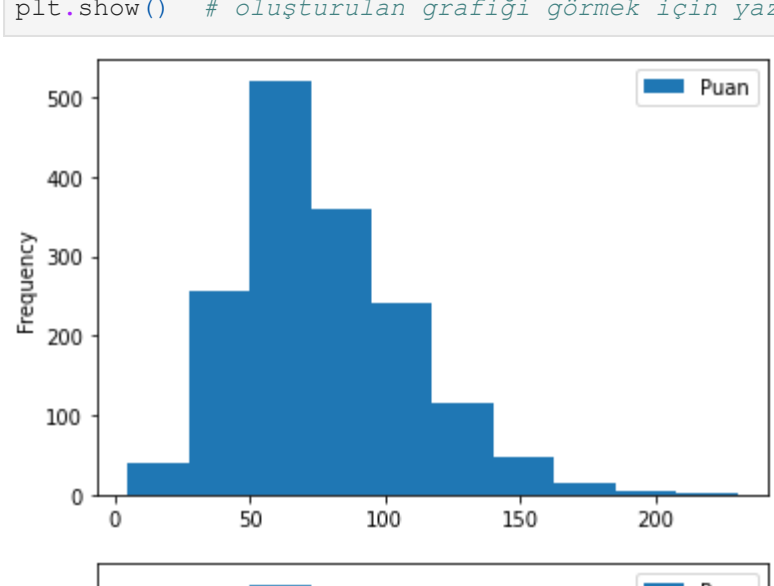
In [121…
```
import pandas as pd
pokemon = pd.read_csv("C:/Users/Dilan/Desktop/pokemon.csv")
pokemon_melt = pd.melt(frame = pokemon, id_vars = "Name", value_vars=["Attack", "Defense"])
pokemon_melt.columns = ['Pokemon_Adı','Saldırı_Türü','Puan']
pokemon_melt
```

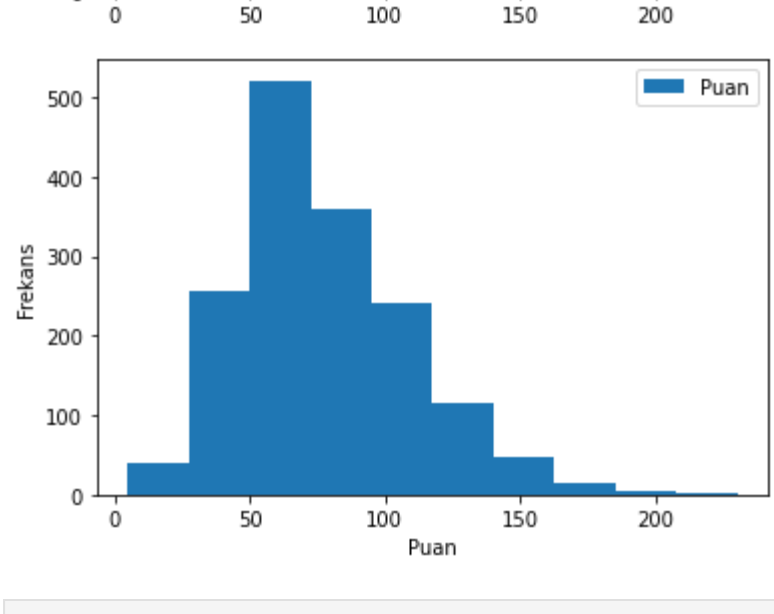
Out[121]:

	Pokemon_Adı	Saldırı_Türü	Puan
0	Bulbasaur	Attack	49
1	Ivysaur	Attack	62
2	Venusaur	Attack	82
3	Mega Venusaur	Attack	100
4	Charmander	Attack	52
...	...	...	...
1595	Diancie	Defense	150
1596	Mega Diancie	Defense	110
1597	Hoopa Confined	Defense	60
1598	Hoopa Unbound	Defense	60
1599	Volcanion	Defense	120

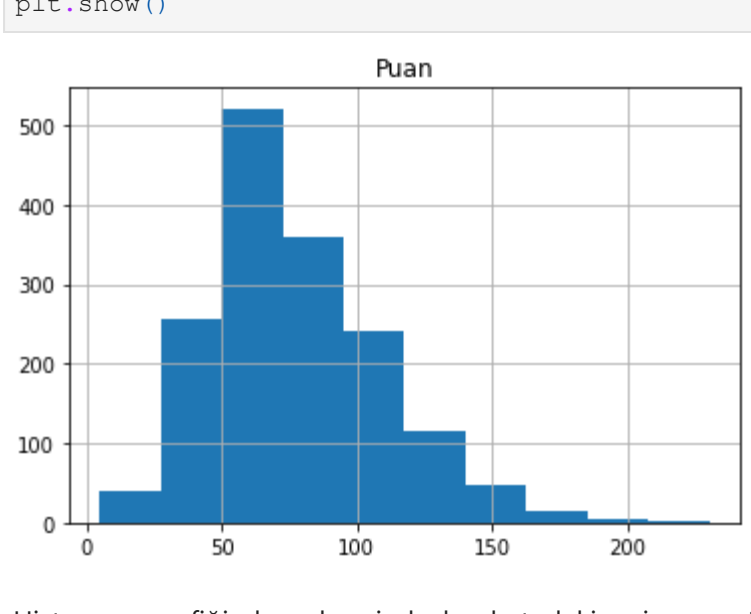
1600 rows × 3 columns

In [122...
```
#Her iki yöntemle de çizdirilebilir.
pokemon_melt.plot(y='Puan', kind='hist')  # 1.histogram
pokemon_melt.plot.hist(y='Puan')          # 2.histogram
plt.xlabel('Puan')      # x eksenine etiket ekler
plt.ylabel('Frekans')   # y eksenine etiket ekler
plt.show()  # oluşturulan grafiği görmek için yazılmalıdır
```



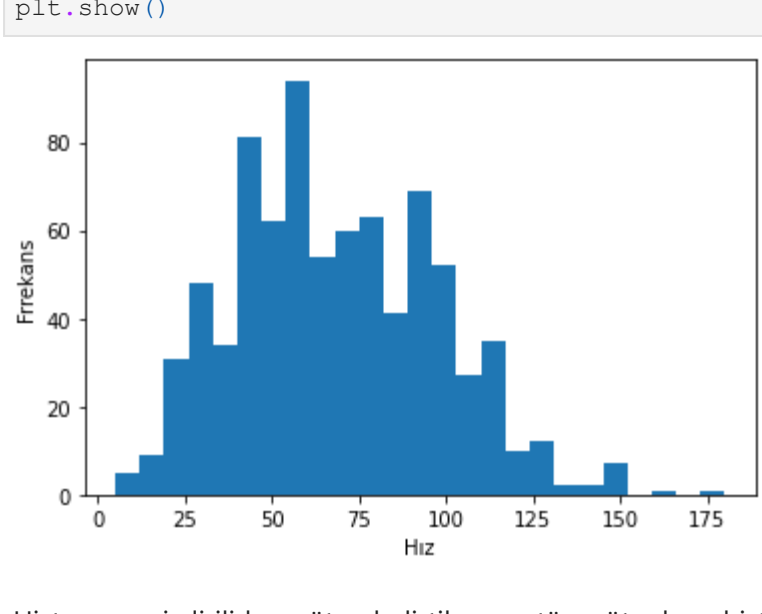


In [123...
```
pokemon_melt.hist(column='Puan')
plt.show()
```



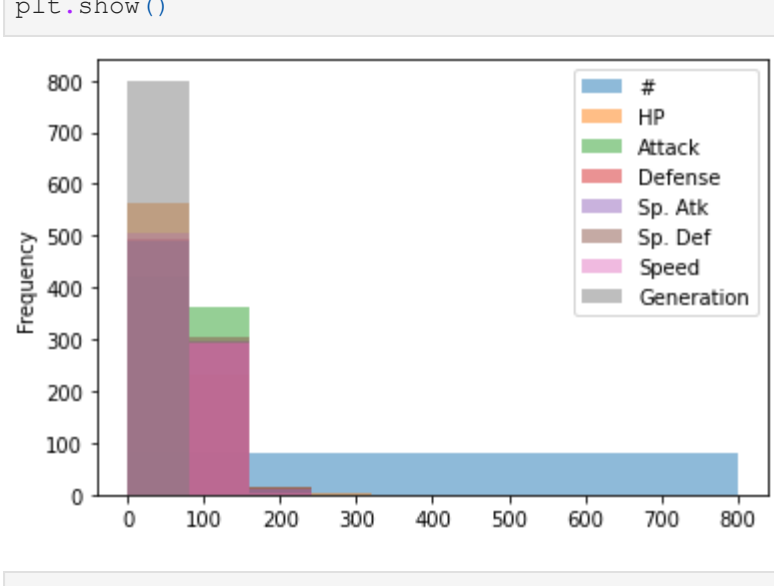
Histogram grafiğinde y ekseninde, her kutudaki veri sayısı gösterilir. Kutulardaki veri sayısı yerine toplam veri sayısına oranı görmek için **normed=True** yazılmalıdır (plt.hist(pokemon_melt['Puan'], normed=True)). Histogram çiziminde veriler istenilen kutulara bölünebilmektedir.

In [124...
```
import matplotlib.pyplot as plt
plt.hist(pokemon['Speed'], bins=25)
plt.xlabel('Hız')
plt.ylabel('Frrekans')
plt.show()
```

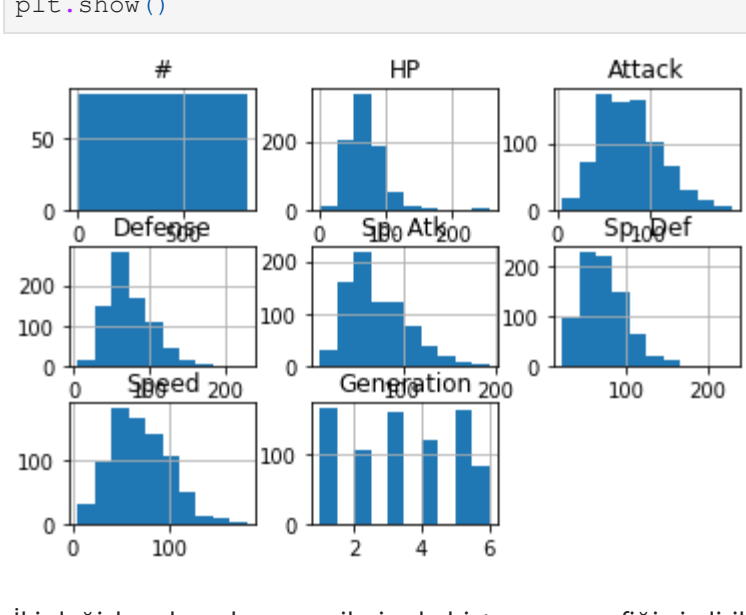


Histogram çizdirilirken sütun belirtilmezse tüm sütunların histogram grafiği bir arada çizdirilir.

In [125...
```
pokemon.plot.hist(alpha=0.5)
plt.show()
```

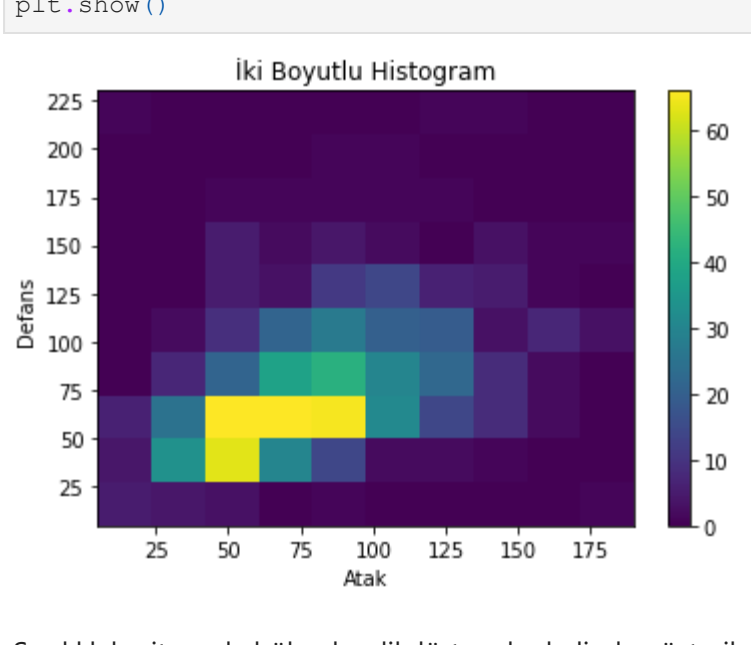


In [126...
```
# Farklı çerçevelerde çizdirilmek istenirse
pokemon.hist()
plt.show()
```



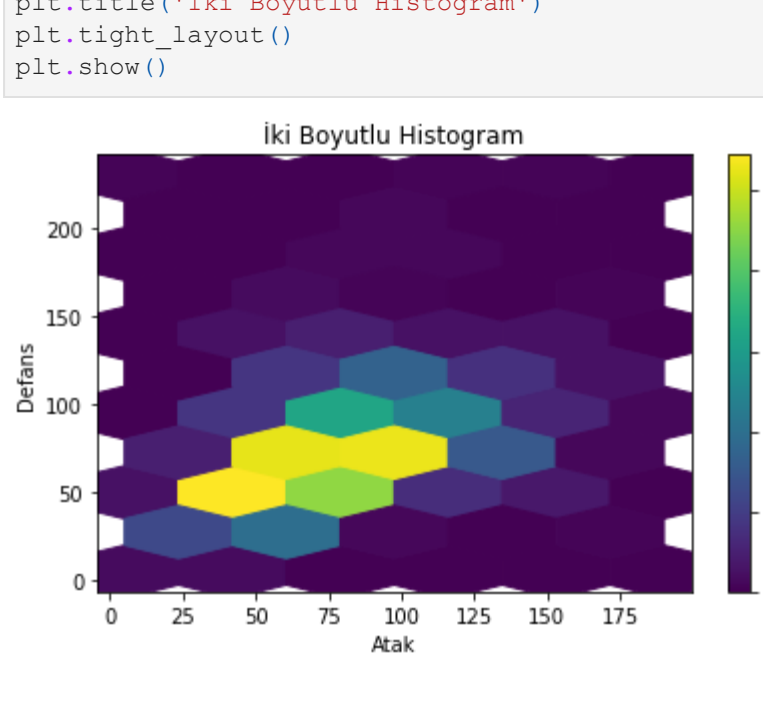
İki değişkenden oluşan verilerin de histogram grafiği çizdirilebilir. Bu grafikler sıcaklık haritası olarak da adlandırılmaktadır. İki değişkenli histogram grafiği çizdirmek için **plt.hist2d()** fonksiyonu kullanılır.

In [127...
```
import matplotlib.pyplot as plt
X = pokemon['Attack']
Y = pokemon['Defense']
plt.hist2d(X,Y)
plt.colorbar()
plt.xlabel('Atak')
plt.ylabel('Defans')
plt.title('İki Boyutlu Histogram')  # grafiğe başlık eklemek için kullanılır.
plt.show()
```



Sıcaklık haritasında bölmeler dikdörtgenler halinde gösterilmekte. Bu bölmeler altıgen olarak da çizdirilebilmektedir. Bunun için **plt.hexbin()** komutu kullanılır. Gridsize argümanı ile sırasıyla x ve y eksenlerdeki altıgen sayısı belirtilir.

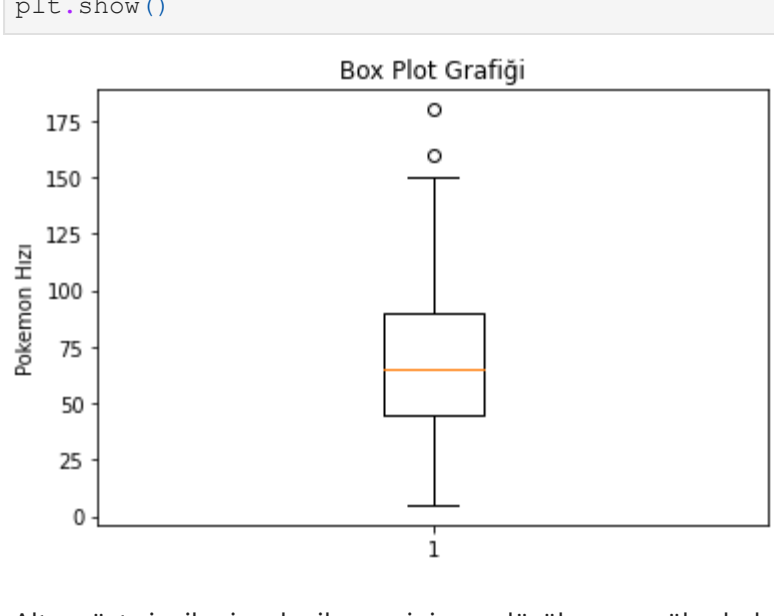
In [128...
```
plt.hexbin(X,Y, gridsize=(5,5))
plt.colorbar()
plt.xlabel('Atak')
plt.ylabel('Defans')
plt.title('İki Boyutlu Histogram')
plt.tight_layout()
plt.show()
```



Box Plot

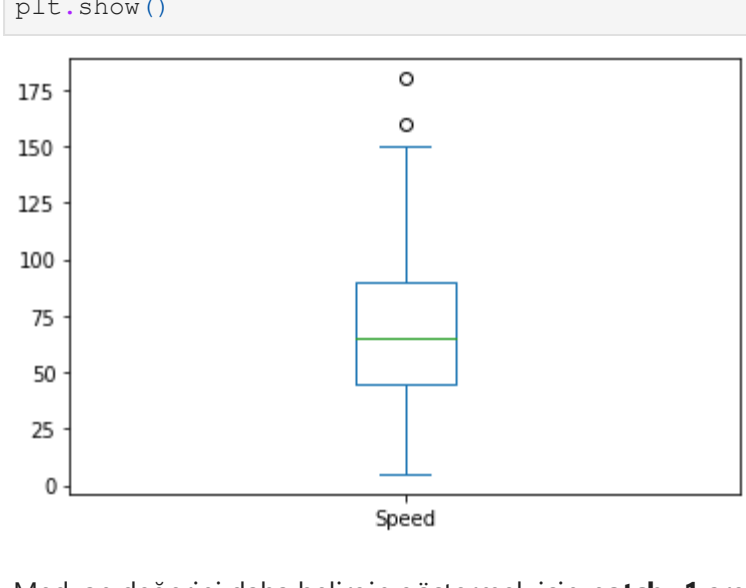
Box Plot yani Kutu grafiği bir verinin minimum, maksimum, medyan değerleri ile 25 ve 75 yüzdelik dilimlerini gösteren bir grafiktir. İncelenen verinin hangi aralıklarda dağıldığını görebilmek için kullanılır.

In [129...
```
import matplotlib.pyplot as plt
import pandas as pd
plt.boxplot(pokemon['Speed'])
plt.ylabel('Pokemon Hızı')
plt.title('Box Plot Grafiği')
plt.show()
```



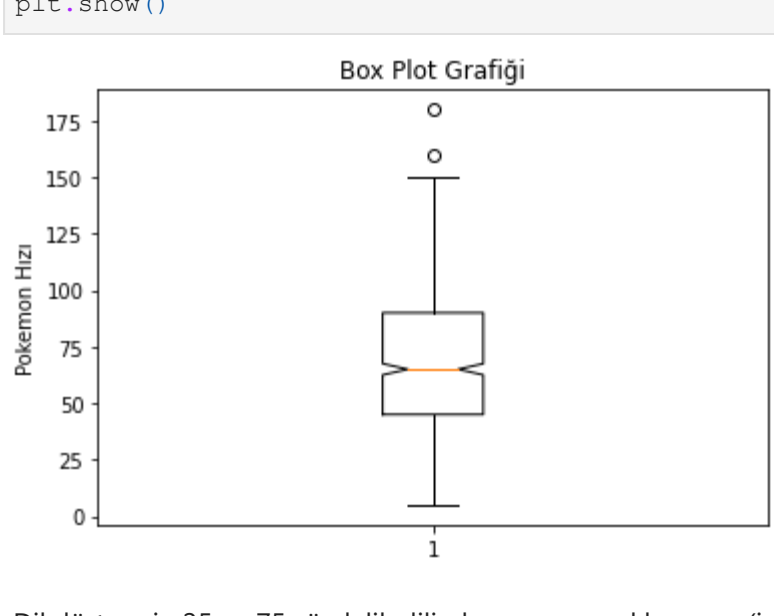
Alt ve üst çizgiler incelenilen verinin en düşük ve en yüksek değerlerini gösterir. Dikdörtgenin alt ve üst kenarları sırasıyla %25 ve %75'lik dilimleri, dikdörtgenin içindeki çizgi ise verinin medyanını yani ortadaki değeri gözsterir. Bu grafik pandas modülünde tanımlı pokemon **.plot(y='Speed', kind='box')** komutu ile de çizdirilebilir.

In [130...
```
pokemon.plot(y='Speed', kind='box')
plt.show()
```



Medyan değerini daha belirgin göstermek için **notch=1** argümanı yazılabilir.

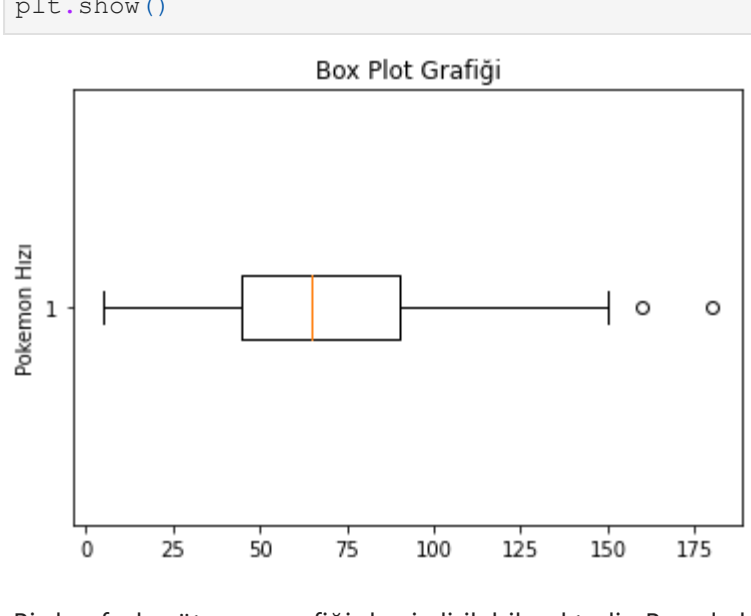
In [131...
```
plt.boxplot(pokemon['Speed'], notch=1)
plt.ylabel('Pokemon Hızı')
plt.title('Box Plot Grafiği')
plt.show()
```



Dikdörtgenin 25 ve 75.yüzdelik dilimler arası çeyrekler arası (interquartile range, IOR) olarak adlandırılmaktadır. Aşağıda ve yukardaki çizgiler aşırı (outlier) olmayan değerlerin alt ve üst sınırlarını belirtir. Grafiğin dışında kalan üst yuvarlak değerler ise aykırı değerlerdir. Bu grafikte iki tane aykırı değerin olduğu görülmektedir.

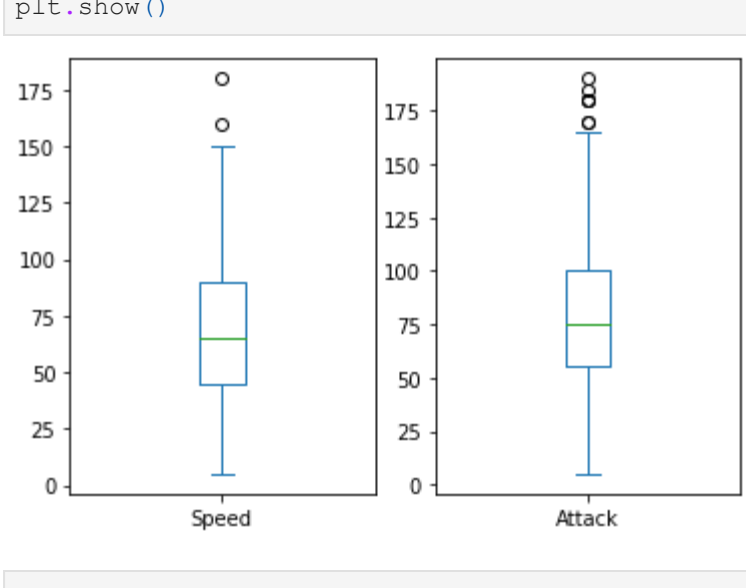
Box Plot yatay olarak da çizdirilebilmektedr. Bunun için vert (yani dikey) argümanına False değri atanmalıdır. Bu argüman varsayılan olarak True değerini almaktadır, değiştirilmediğinde grafik dikey olarak çizdirilir.

In [132...
```
plt.boxplot(pokemon['Speed'], vert=False)
plt.ylabel('Pokemon Hızı')
plt.title('Box Plot Grafiği')
plt.show()
```

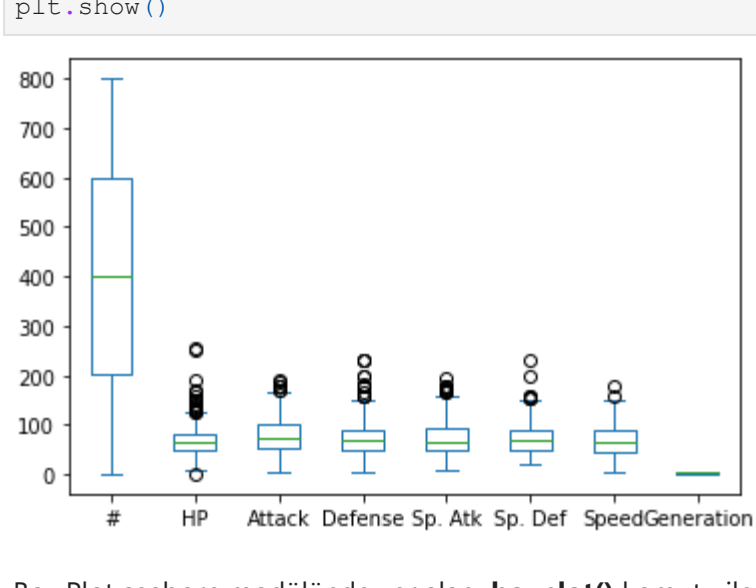


Birden fazla sütunun grafiği de çizdirilebilmektedir. Burada herhangi bir seçim yapılmazsa bütün sütunların kutu grafiği bir arada çizdirilir.

In [133...
```
s = ['Speed', 'Attack']
pokemon[s].plot(kind='box', subplots=True)
plt.show()
```

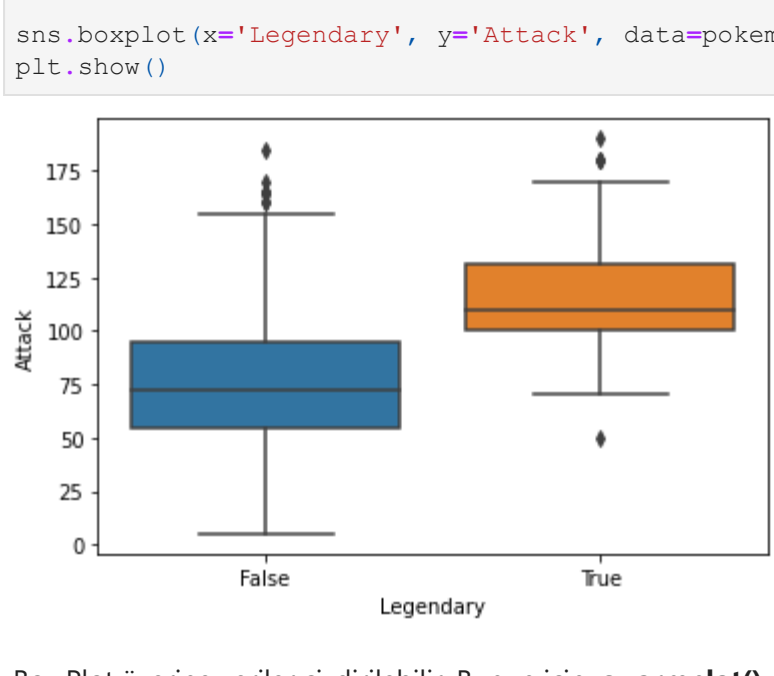


In [134...
```
pokemon.plot(kind='box')
plt.show()
```



Box Plot seaborn modülünde yer alan **.boxplot()** komutu ile de çizdirilebilmektedir. Grafik varsayılan olarak dikey çizdirilir, yatay çizdirilmek isteniyorsa x ve y eksenlerinin yerleri değiştirilmelidir.

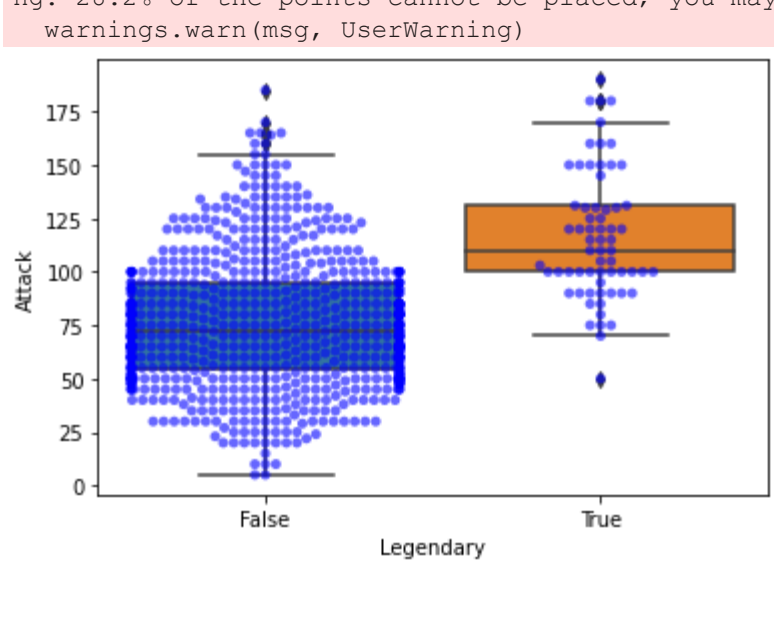
In [135...
```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

sns.boxplot(x='Legendary', y='Attack', data=pokemon)
plt.show()
```



Box Plot üzerine veriler çizdirilebilir. Bunun için **.swarmplot()** metodu kullanılır. Bu metodla aynı değerlere sahip veriler yan yana çizdirilerek dağılımları hakkında fikir verir.

In [136...
```
sns.boxplot(x='Legendary', y='Attack', data=pokemon)
sns.swarmplot(x='Legendary', y='Attack', data=pokemon, color='b', alpha=0.6)
plt.show()
```

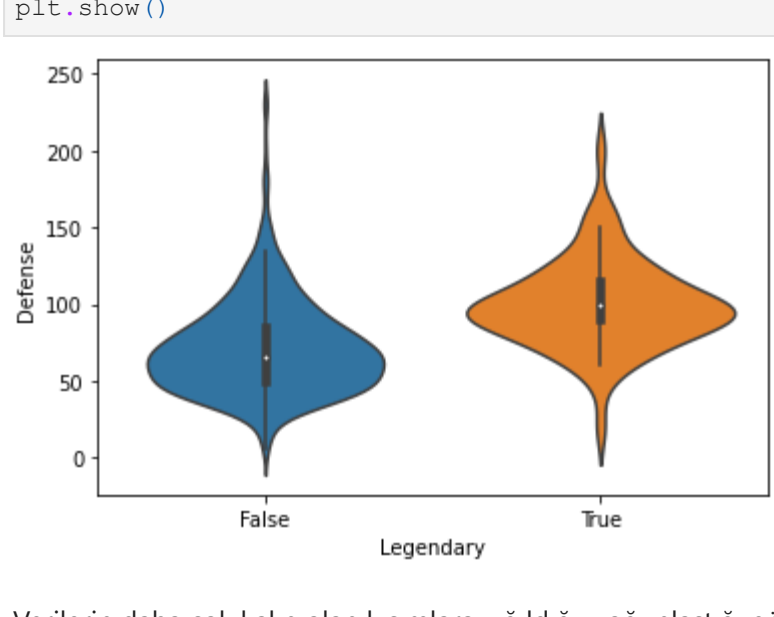
```
C:\Users\Dilan\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categorical.py:1296: UserWarning: 28.2% of the points cannot be placed; you may want to decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
```



Keman (Violin) Grafiği

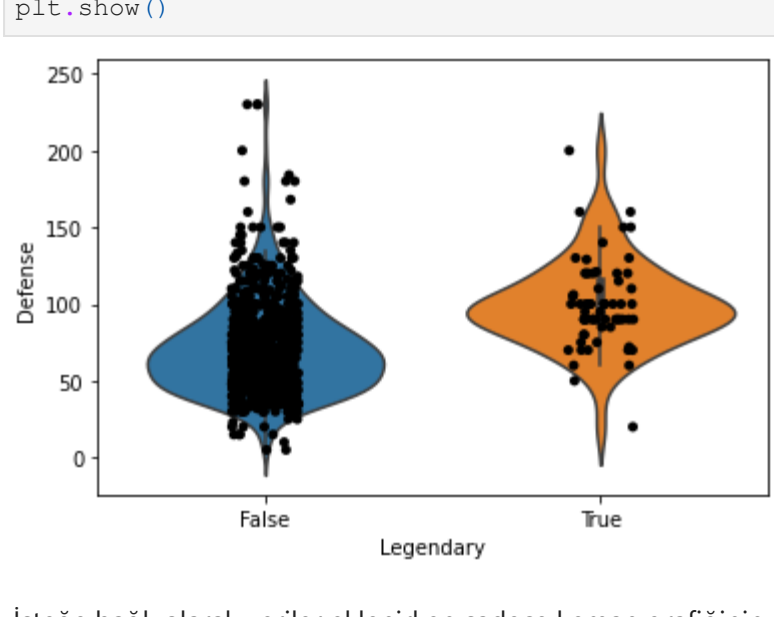
Violin grafiği, Box plot grafiğine benzemektedir. Dışarıdan bir keman gövedesini andırdığı için bu isim verilmiştir. Bu grafik de incelenilen verinin dağlımı hakkında bilgi verir. Violin grafiğini çizdirmek için seaborn paketinde yer alan **violinplot()** metodu kullanılır. Seaborn violin grafiğinde box plot grafiği ile birlikte verilir.

In [137...
```
sns.violinplot(x='Legendary', y='Defense', data=pokemon)
plt.show()
```



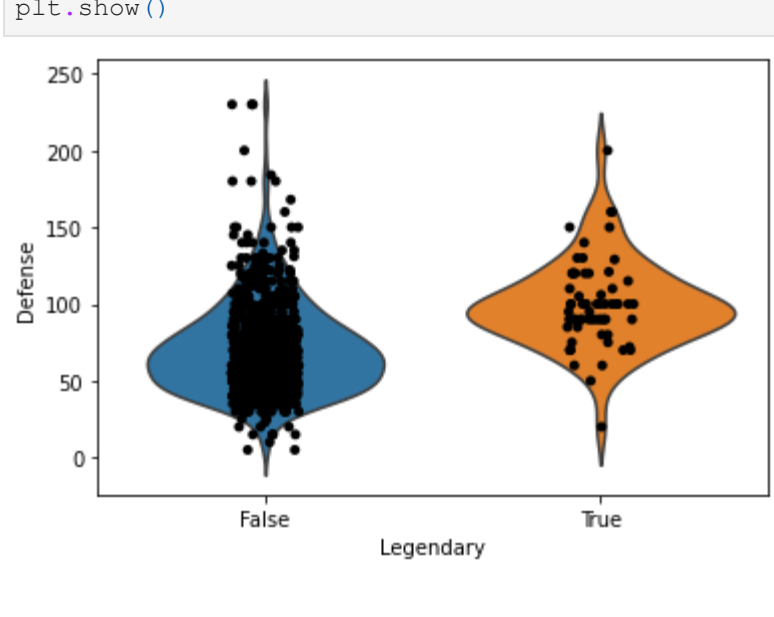
Verilerin daha çok kalın olan kısımlara yığıldığı, yoğunlaştığı görülmektedir. Bu grafikte de veriler grafiğin üzerine eklenebilir. Bunun için .swarmplot yerine stripplot kullanılanılabilir. Verilerin sağa sola yayılması isteniyorsa **jitter=True** argümanı yazılabilir.

In [138...
```
sns.violinplot(x='Legendary', y='Defense', data=pokemon)
sns.stripplot(x='Legendary', y='Defense', data=pokemon, jitter=True, color='black')
plt.show()
```



İsteğe bağlı olarak veriler eklenirken sadece keman grafiğinin çizdirilip kutu garfiğinin kaldırılması isteniyorsa **inner=None** yazılabilir.

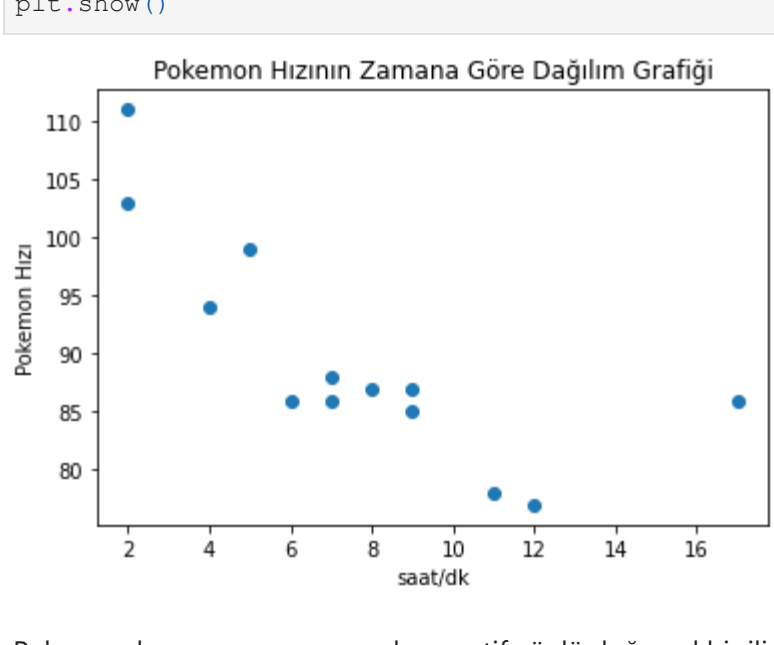
In [139...
```
sns.violinplot(x='Legendary', y='Defense', data=pokemon, inner=None)
sns.stripplot(x='Legendary', y='Defense', data=pokemon, jitter=True, color='black')
plt.show()
```



Nokta Grafik (Scatter Plot)

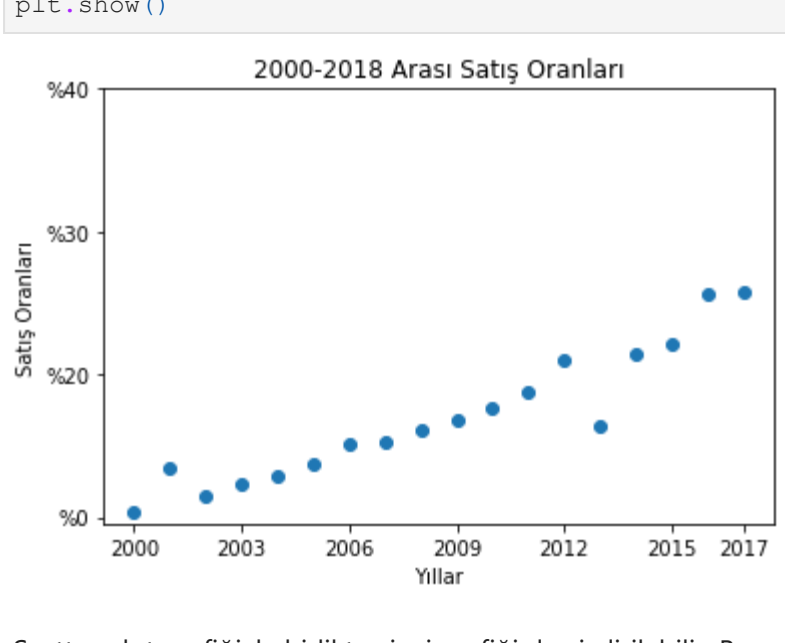
Her bir veri noktasını tek başına olarak nokta şeklinde gösteren grafiklere Scatter Plot denir. Bu grafik her bir gözlem için bir nokta çizer ve iki sayısal değişken arasındaki ilişkileri gözlemlemek ve göstermektir. Scatter Plot ile verinin dağılımı incelenebilmektedir. Dağılım grafiği çizdirmek için **.scatter()** metodu kullanılmaktadır.

In [140...
```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])

plt.scatter(x, y)
plt.xlabel('saat/dk')
plt.ylabel('Pokemon Hızı')
plt.title('Pokemon Hızının Zamana Göre Dağılım Grafiği')
plt.show()
```



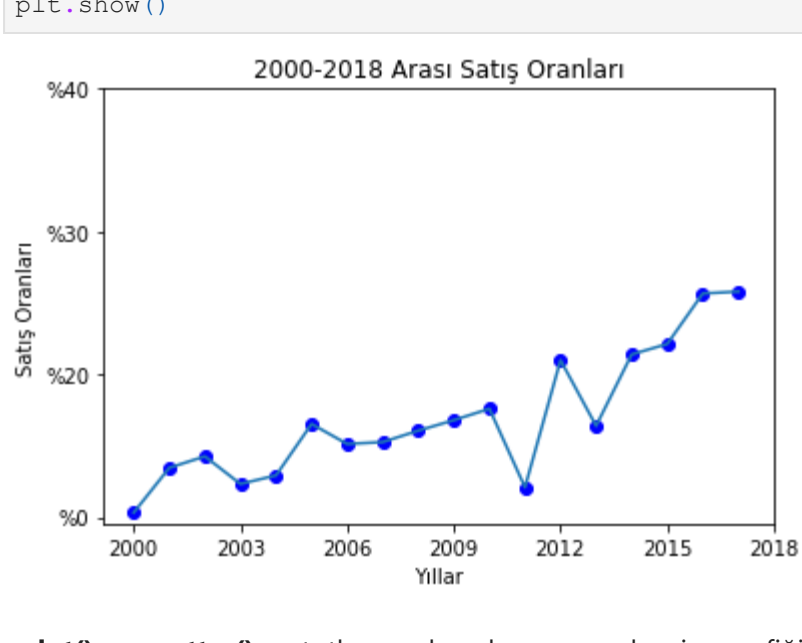
Pokemon hızı ve zaman arasında negatif yönlü doğrusal bir ilişkinin olduğu görülmektedir. Yani pokemon hızı azaldıkça zaman da artmaktadır. Ekenlerde yazılan değerleri belirlemek için **plt.yticks()** ve **plt.xticks()** kullanılır.

In [141...
```
import matplotlib.pyplot as plt
import numpy as np

yillar = range(2000,2018) #2000'i içerir fakat 2018'i içermez
satis = [.5,5.2,2.3,3.5,4.4,5.6,7.7,7.9,9.1,10.2,11.4,13.15,16.5,9.6,17.1,18.2,23.5,23.7]

plt.scatter(yillar, satis)
plt.xticks([2000,2003,2006,2009,2012,2015,2017])
plt.yticks([0,15,30,45], ['%0','%20','%30','%40'])
plt.xlabel('Yıllar')
plt.ylabel('Satış Oranları')
plt.title('2000-2018 Arası Satış Oranları')
plt.show()
```



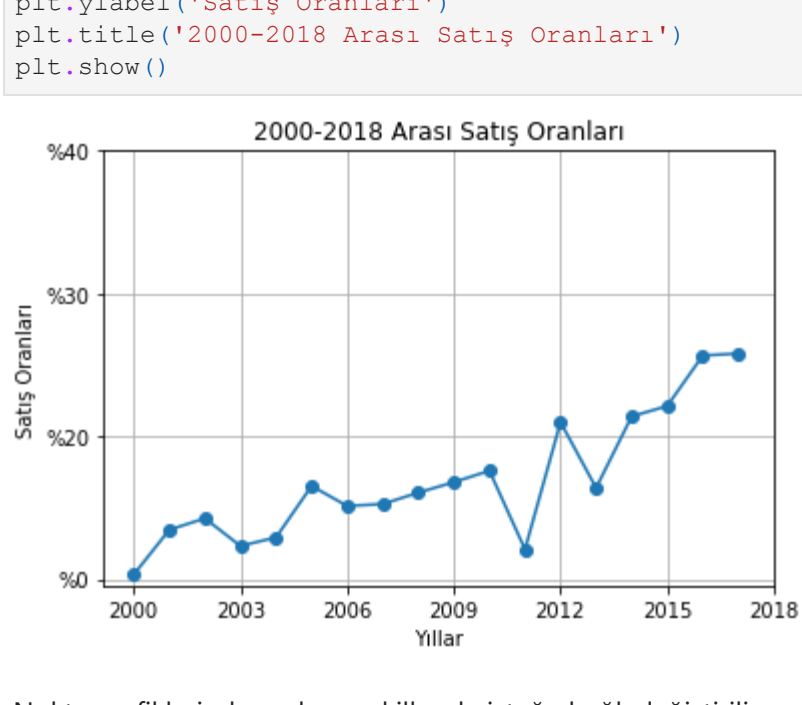
Scatter plot grafiğiyle birlikte çizgi grafiği de çizdirilebilir. Bunun için **.plot()** ve **.scatter()** metotları bir arada kullanılmalı.

In [142...
```
import matplotlib.pyplot as plt
import numpy as np

yillar = range(2000,2018)
satis = [.5,5.2,6.4,3.5,4.4,9.8,7.7,7.9,9.1,10.2,11.4,3.15,16.5,9.6,17.1,18.2,23.5,23.7]

plt.scatter(yillar, satis, color='blue')
plt.plot(yillar, satis)
plt.xticks([2000,2003,2006,2009,2012,2015,2018])
plt.yticks([0,15,30,45], ['%0','%20','%30','%40'])
plt.xlabel('Yıllar')
plt.ylabel('Satış Oranları')
plt.title('2000-2018 Arası Satış Oranları')
plt.show()
```



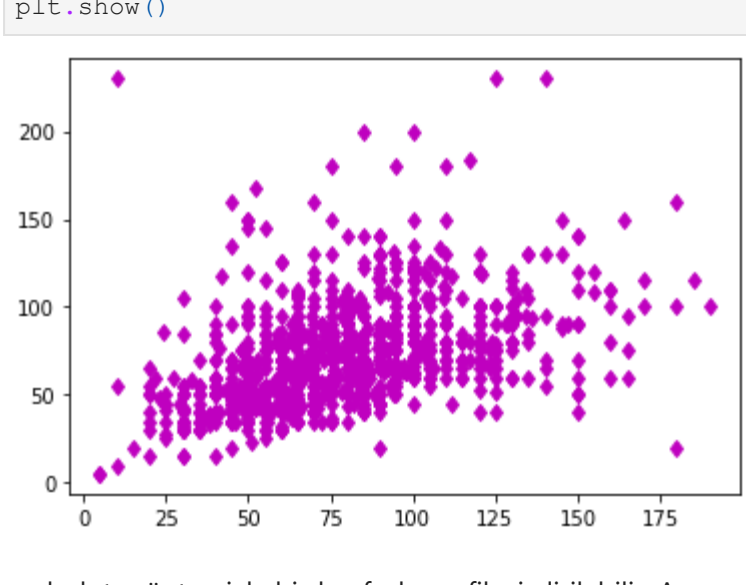
.plot() ve **.scatter()** metotlarını arka arkaya yazmak yeine grafiği tek bir komut içinde çizdirilebilmektedir. **plt.plot(yillar, satis, '-o')** yazılarak da aynı grafik elde edilebilir. Burada '-' çizgi ve 'o' nokta grafiğini belirtmektedir. Buna ek olarak isteğe bağlı seviye çizgileri için plt.grid(True) eklenebilir.

In [143...
```
import matplotlib.pyplot as plt
import numpy as np

yillar = range(2000,2018)
satis = [.5,5.2,6.4,3.5,4.4,9.8,7.7,7.9,9.1,10.2,11.4,3.15,16.5,9.6,17.1,18.2,23.5,23.7]

plt.plot(yillar, satis, '-o')
plt.grid(True)
plt.xticks([2000,2003,2006,2009,2012,2015,2018])
plt.yticks([0,15,30,45], ['%0','%20','%30','%40'])
plt.xlabel('Yıllar')
plt.ylabel('Satış Oranları')
plt.title('2000-2018 Arası Satış Oranları')
plt.show()
```



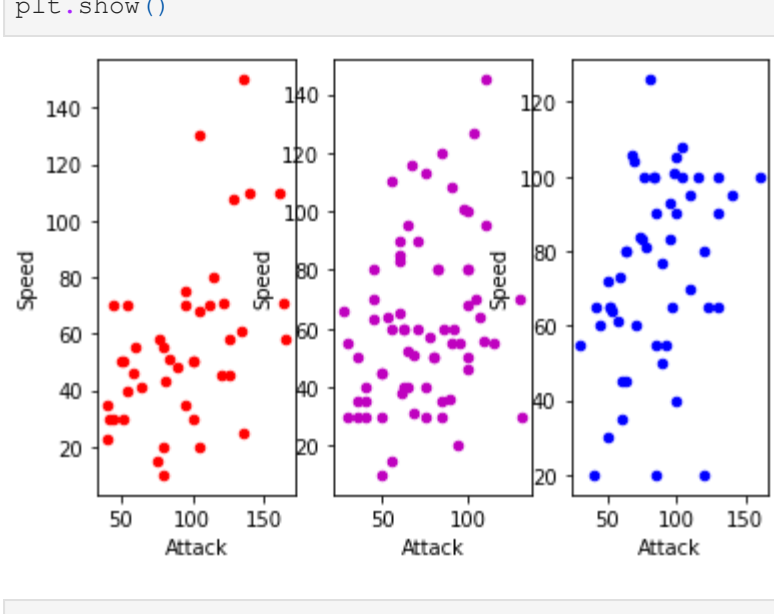
Nokta grafiklerinde renk ve şekiller de isteğe bağlı değiştirilip, seçilebilir.

In [144...
```
plt.plot(pokemon['Attack'], pokemon['Defense'], 'md')  # m mor ve d elmas şeklini temsil etmektedir.
plt.show()
```

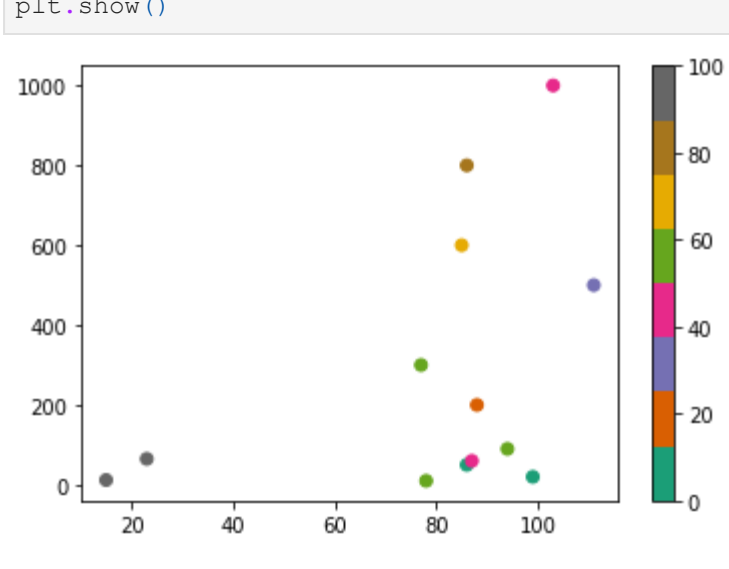


.subplots yöntemiyle birden fazla grafik çizdirilebilir. Ayrıca matplotlib modülünde bulunan **style.use()** komutu ile grafikler farklı temalarda çizdirilebilir. Örneğin **plt.style('ggplot')** komutunu kullanarak grafik ggplot temasıyla çizdirilebilir.

In [145...
```
figur, eksen=plt.subplots(nrows=1,ncols=3) # Tek satıra 3 sütun şeklinde 3 grafik çizdirebilmek için
pokemon[pokemon['Type 1']=='Rock'].plot(ax=eksen[0], x='Attack', y='Speed', kind='scatter', color='r') #1.grafi
pokemon[pokemon['Type 1']=='Grass'].plot(ax=eksen[1],x='Attack', y='Speed', kind='scatter', color='m') #2.grafi
pokemon[pokemon['Type 1']=='Fire'].plot(ax=eksen[2], x='Attack', y='Speed', kind='scatter', color='b') #3.grafi
plt.show()
```

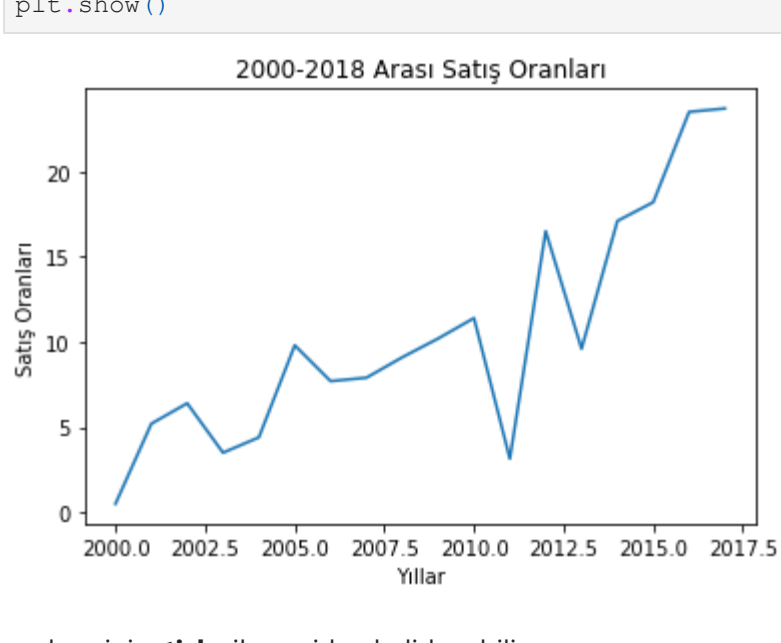


In [146...
```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([99,86,88,111,103,87,94,78,77,85,86,23,15])
y = np.array([20,50,200,500,1000,60,90,10,300,600,800,65,12])
colors = np.array([0, 10, 20, 30, 40, 45, 50, 55, 60, 70, 80, 90, 100])

plt.scatter(x, y, c=colors, cmap= 'Dark2')
plt.colorbar()
plt.show()
```



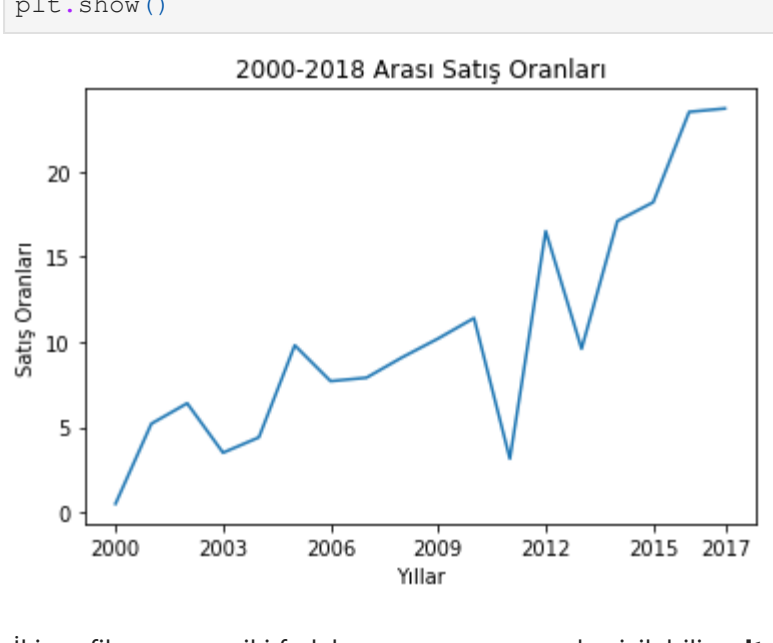
Çizgi Grafik (Line Plot)

In [147...
```
import matplotlib.pyplot as plt
import numpy as np

yillar = range(2000,2018)
satis = [.5,5.2,6.4,3.5,4.4,9.8,7.7,7.9,9.1,10.2,11.4,3.15,16.5,9.6,17.1,18.2,23.5,23.7]

plt.plot(yillar, satis)

plt.xlabel('Yıllar')
plt.ylabel('Satış Oranları')
plt.title('2000-2018 Arası Satış Oranları')
plt.show()
```



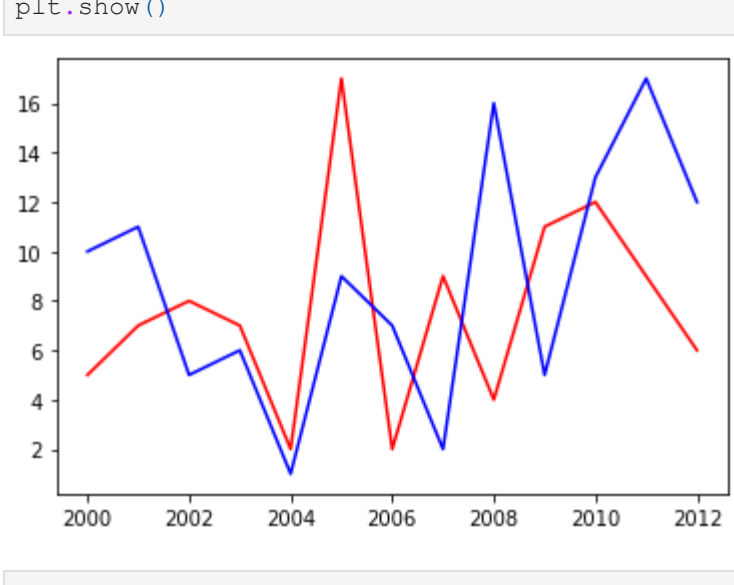
x eksenini **.xticks** ile yeniden belirlenebilir.

In [148...
```
import matplotlib.pyplot as plt
import numpy as np

yillar = range(2000,2018)
satis = [.5,5.2,6.4,3.5,4.4,9.8,7.7,7.9,9.1,10.2,11.4,3.15,16.5,9.6,17.1,18.2,23.5,23.7]

plt.plot(yillar, satis)
plt.xticks([2000,2003,2006,2009,2012,2015,2017])
plt.xlabel('Yıllar')
plt.ylabel('Satış Oranları')
plt.title('2000-2018 Arası Satış Oranları')
plt.show()
```

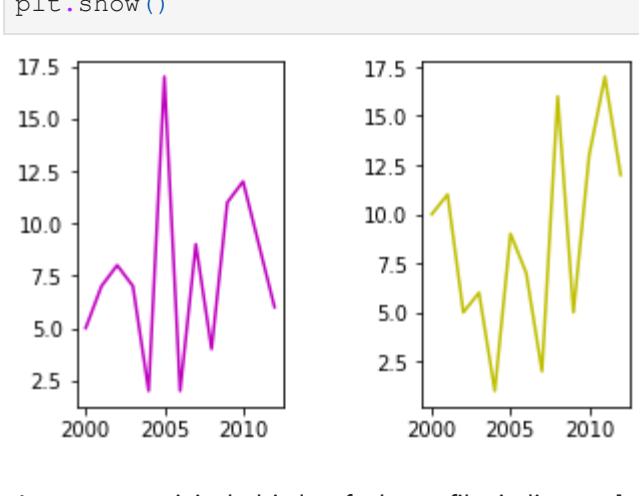


İki grafik yan yana iki farklı veya aynı çerçevede çizilebilir. **plt.axes()** ile yan yana grafikler çizilebilir.

In [149...
```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([10,11,5,6,1,9,7,2,16,5,13,17,12])
yillar = np.arange(2000,2013)

plt.plot(yillar, x, 'r')
plt.plot(yillar, y, 'b')
plt.show()
```

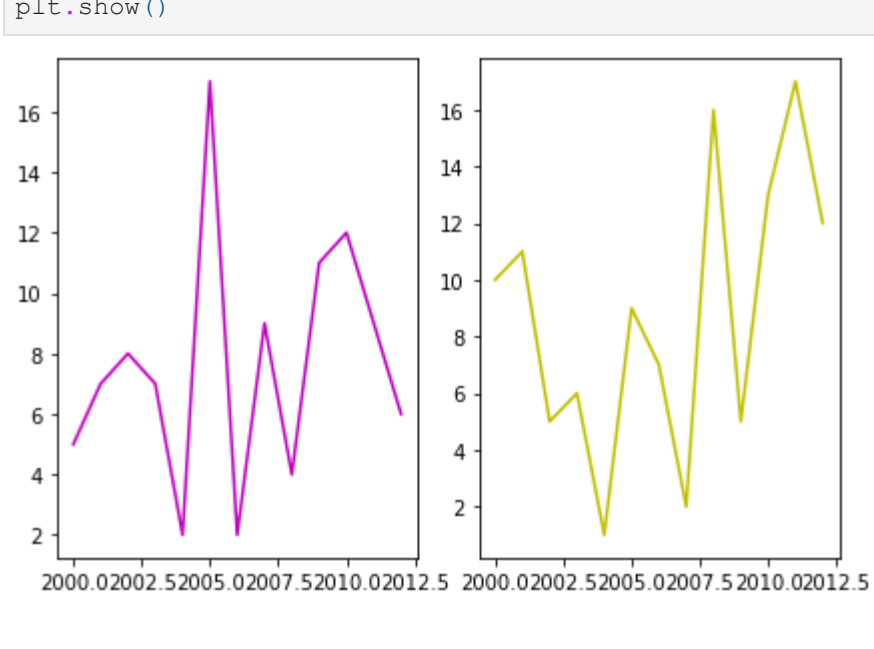


In [150...
```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([10,11,5,6,1,9,7,2,16,5,13,17,12])
yillar = np.arange(2000,2013)

plt.axes([0.25,0.25,0.24,0.6])  # 1.grafiğin konumu için
plt.plot(yillar, x, 'm')
plt.axes([0.65,0.25,0.24,0.6])  # 2.grafiğin konumu için
plt.plot(yillar, y, 'y')
plt.show()
```



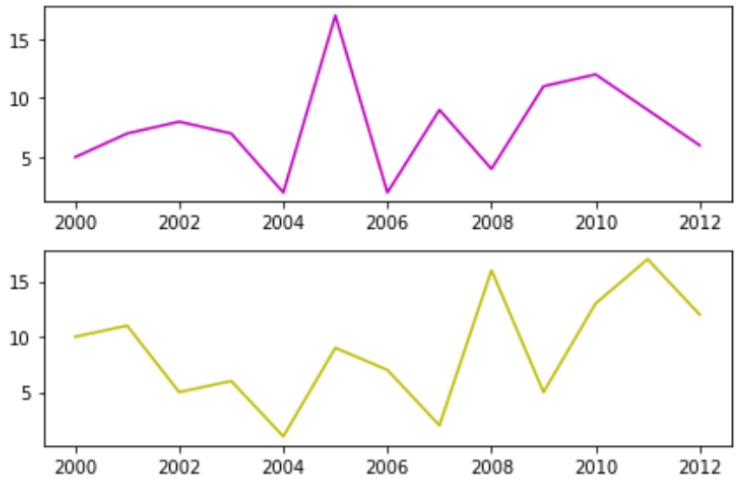
Aynı çerçeve içinde birden fazla grafik çizdirme **plt.subplot()** yöntemiyle de yapılabilir. Alt grafiklerin çok sıkışık şekilde gösterilmemesi için **plt.tight_layout()** kullanılabilir.

In [151...
```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([10,11,5,6,1,9,7,2,16,5,13,17,12])
yillar = np.arange(2000,2013)

plt.subplot(1,2,1)          # 1 satır ve 2 sütundan oluşan çerçeve çizilir ve 1.grafiği aktif hale getirir.
plt.plot(yillar, x, 'm')    # 1.grafik
plt.subplot(1,2,2)          # 1 satır ve 2 sütundan oluşan çerçeve çizilir ve 2.grafiği aktif hale getirir.
plt.plot(yillar, y, 'y')    # 2.grafiği
plt.tight_layout()
plt.show()
```



In [152…

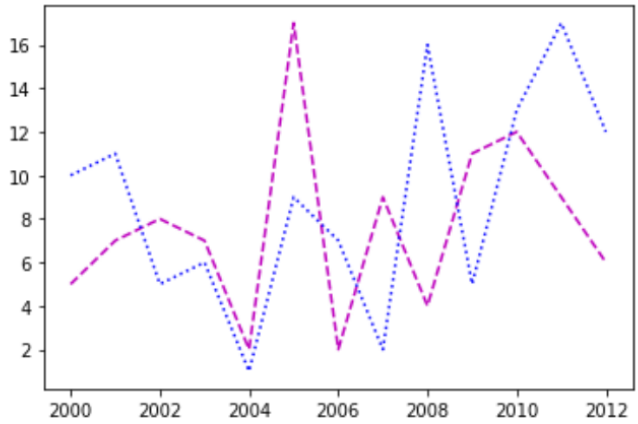
```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([10,11,5,6,1,9,7,2,16,5,13,17,12])
yillar = np.arange(2000,2013)

plt.subplot(2,1,1)        # 2 satır ve 1 sütundan oluşan çerçeve çizilir ve 1.grafiği aktif hale getirir.
plt.plot(yillar, x, 'm')  # 1.grafik
plt.subplot(2,1,2)        # 2 satır ve 1 sütundan oluşan çerçeve çizilir ve 2.grafiği aktif hale getirir.
plt.plot(yillar, y, 'y')  # 2.grafik
plt.tight_layout()
plt.show()
```



Bir arada çizdirilen grafikleri ayırt etmek için plt.label ile etiket eklenebilir. Etiket yeri belirtmek için **loc** argümanı kullanılır. Grafiğe metin eklemek için **plt.annotate()** metodu da kulanılabilir.

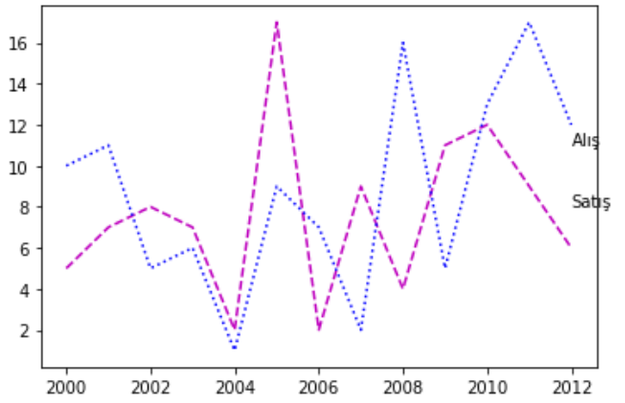
In [153…

```
x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([10,11,5,6,1,9,7,2,16,5,13,17,12])
yillar = np.arange(2000,2013)

plt.plot(yillar, x, 'm--', label='Alış')
plt.plot(yillar, y, 'b:' , label='Satış')
plt.show()
```



In [154…

```
x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([10,11,5,6,1,9,7,2,16,5,13,17,12])
yillar = np.arange(2000,2013)

plt.plot(yillar, x, 'm--')
plt.plot(yillar, y, 'b:')
plt.annotate('Alış', xy=(2012,11))
plt.annotate('Satış', xy=(2012,8))
plt.show()
```



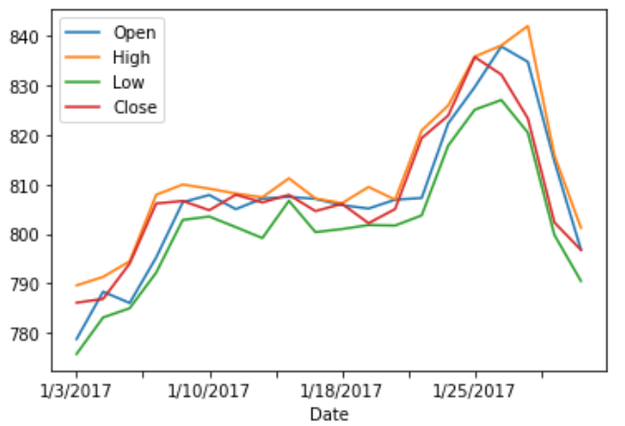
Burda da aynı şekide pandas modülündeki **.plot()** metoduyla çizgi grafiği çizdirilebilir.

In [155…
```
import pandas as pd
Google = pd.read_csv("C:/Users/Dilan/Desktop/Google.csv",index_col='Date')
Google
```

Out[155]:

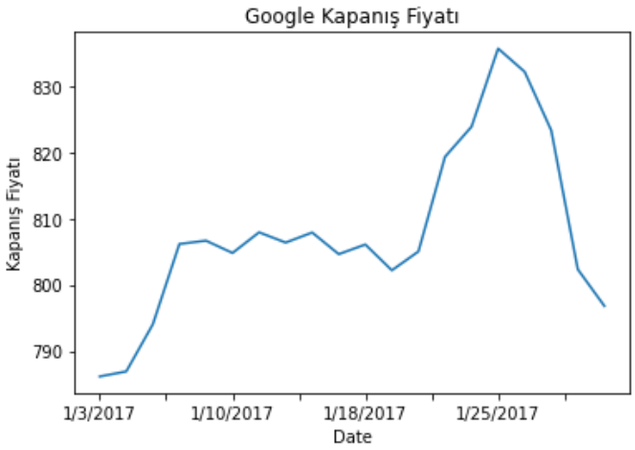
	Open	High	Low	Close	Volume
Date					
1/3/2017	778.81	789.63	775.80	786.14	1,657,300
1/4/2017	788.36	791.34	783.16	786.90	1,073,000
1/5/2017	786.08	794.48	785.02	794.02	1,335,200
1/6/2017	795.26	807.90	792.20	806.15	1,640,200
1/9/2017	806.40	809.97	802.83	806.65	1,272,400
1/10/2017	807.86	809.13	803.51	804.79	1,176,800
1/11/2017	805.00	808.15	801.37	807.91	1,065,900
1/12/2017	807.14	807.39	799.17	806.36	1,353,100
1/13/2017	807.48	811.22	806.69	807.88	1,099,200
1/17/2017	807.08	807.14	800.37	804.61	1,362,100
1/18/2017	805.81	806.21	800.99	806.07	1,294,400
1/19/2017	805.12	809.48	801.80	802.17	919,300
1/20/2017	806.91	806.91	801.69	805.02	1,670,000
1/23/2017	807.25	820.87	803.74	819.31	1,963,600
1/24/2017	822.30	825.90	817.82	823.87	1,474,000
1/25/2017	829.62	835.77	825.06	835.67	1,494,500
1/26/2017	837.81	838.00	827.01	832.15	2,973,900
1/27/2017	834.71	841.95	820.44	823.31	2,965,800
1/30/2017	814.66	815.84	799.80	802.32	3,246,600
1/31/2017	796.86	801.25	790.52	796.79	2,160,600

In [156…
```
Google.plot()
plt.show()
```



In [157…
```
Google['Close'].plot(title = 'Google Kapanış Fiyatı')  # Sadece Close sütununa ait çizgi grafiği
plt.ylabel('Kapanış Fiyatı')  # y eksenine bilgi ekleme
```

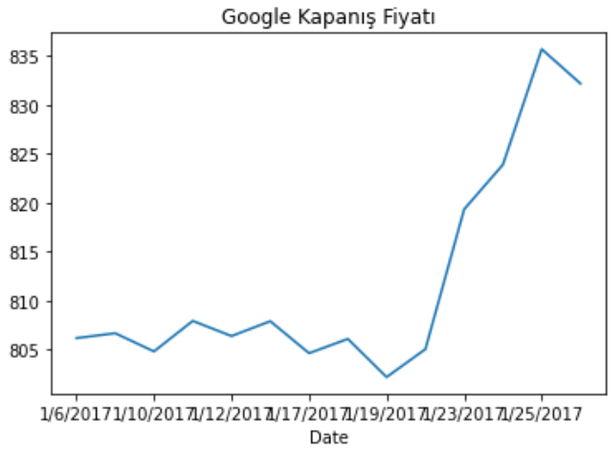
```
plt.show()
```



Sadece belli bir tarih aralığının grafiği de çizdirilebilir.

In [158…

```
Google.loc['1/6/2017':'1/26/2017','Close'].plot(title = 'Google Kapanış Fiyatı')
plt.show()
```



Daha fazlası için https://www.w3schools.com/python/matplotlib_pyplot.asp adresine bakılabilir.

6. KAYNAKÇA

1. ARSLAN, "Python ile Veri Bilim", İstanbul 2019

2. NumPy, "Learning Numpy", Erişim: 8 Nisan 2022, https://numpy.org/learn/

3. Pandas, "Pandas", Erişim: 9 Nisan 2022, https://pandas.pydata.org

4. Matplotlib, "Matplotlib Pyplot", Erişim: 15 Nisan 2022, https://www.w3schools.com/python/matplotlib_pyplot.asp

5. Kaggle, "Pokemon- Weedle's Cave", Erişim: 9 Nisan 2022, https://www.kaggle.com/datasets/terminus7/pokemon-challenge?select=pokemon.csv

6. Kaggle, "RNN - Google Stocks - JMA", Erişim: 15 Nisan 2022, https://www.kaggle.com/code/ukveteran/rnn-google-stocks-jma/data