Matplotlib

Matplotlib low-level bir grafik kütüphanesidir.

Görselleştirmeleri temel anlamda gösterip çeşitlendirebilmemiz ve veriyi güzel ifade edebilmemiz için birçok araç sunar.

Matplotlib ile ilgili birçok özellik **pyplot** submodule altında bulunur.

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

xpoints = np.array([0, 6])
ypoints = np.array([0, 250])

plt.plot(xpoints, ypoints)
plt.show()
```

x ve y noktalarını görselleştirmek

plot() fonksiyonunu belirli noktaları grafik üzerinde belirlemek için kullanırız.

Varsayılan ayarı gereğince, **plot()** fonksiyonu noktalar arasında bir çizim işlemi yapar.

plot() fonksiyonundaki birinci parametre, x eksenindeki noktaları temsil ederken; ikinci parametre y eksenindeki noktaları temsil eder.

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

xpoints = np.array([1, 8])
ypoints = np.array([3, 10])

plt.plot(xpoints, ypoints)
plt.show()
```

Çizgi Olmadan Grafik Oluşturma

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

xpoints = np.array([1, 8])
ypoints = np.array([3, 10])

plt.plot(xpoints, ypoints, 'o') #Marker ekledik.
plt.show()
```

Birden Fazla Nokta Belirlemek

!?

Birden fazla nokta belirlediğimizde grafikte neyin değişmesini bekleriz?

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

xpoints = np.array([1, 2, 6, 8])
ypoints = np.array([3, 8, 1, 10])

plt.plot(xpoints, ypoints)
plt.show()
```

Default olarak gelen X noktaları

Eğer x noktaları için bir parametre girilmez ise plot fonksiyonu x değerleri için 0, 1, 2, 3 gibi değerler girerek grafiği tamamlayacaktır.

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

ypoints = np.array([3, 8, 1, 10, 5, 7])

plt.plot(ypoints)
plt.show()
```

Markers

Grafik üzerindeki her bir noktayı işaretleyici noktalarla ifade edebiliriz.

Sembolleri özelleştirerek elimizdeki görselleştirmeyi daha iyi tasarlayabiliriz.

Eğer nokta olarak görünmesini istersek:

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

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, marker = 'o')
plt.show()
```

Ya da bir yıldız:

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

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, marker = 'x')
plt.show()
```

Diğer marker türlerini inceleyelim:

'o' Circle
'*' Star
'.' Point
',' Pixel
'x' X
'X' X (filled)
'+' Plus
'P' Plus (filled)
's' Square

Markerları aynı zamanda renk seçeneklerimiz ile de özelleştirebiliriz.

Hatta markerların çerçevelerinin dahi renklerini belirleyebiliyoruz.

mec = markeredgecolor

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

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, marker = 'o', ms = 20, mec = 'r')
plt.show()
```

Matplotlibin bize sunduğu renk seçenekleri dışında da renk kodları ile istediğimiz renkleri kullanabiliriz.

Renk seçimi neden önemlidir?

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

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, marker = 'o', ms = 20, mec = '#4CAF50', mfc = '#4CAF50')

plt.show()
```

Grafiklerde sadece markerların değil çizgilerin de renklerini, stillerini ve genişliklerini değiştirmek mümkündür.

```
plt.plot(ypoints, ls = ':')
```

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

```
ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, color = 'r')
plt.show()
```

```
plt.plot(ypoints, c = '#4CAF50')
```

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

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, linewidth = '20.5')
plt.show()
```

Eğer bir grafik içinde birden fazla çizgi grafiği kullanmak istersek grafikleri ayrı ayrı birer değişken içine atayarak bunu yapabiliriz.

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

y1 = np.array([3, 8, 1, 10])
y2 = np.array([6, 2, 7, 11])
```

```
plt.plot(y1)
plt.plot(y2)

plt.show()
```

Bunu x ve y eksenlerini ayrıca belirterek de yapabiliriz.

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

x1 = np.array([0, 1, 2, 3])
y1 = np.array([3, 8, 1, 10])
x2 = np.array([0, 1, 2, 3])
y2 = np.array([6, 2, 7, 11])

plt.plot(x1, y1, x2, y2)
plt.show()
```

Etiketler ve Başlıklar

xlabel() ve ylabel() fonksiyonlarını kullanarak x ve y eksenlerinde görünecek etiketleri belirtebiliriz.

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

x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])

plt.plot(x, y)

plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burnage")
```

```
plt.show()
```

title() özelliğini kullanarak bir başlık oluşturabiliriz.

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

x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])

plt.plot(x, y)

plt.title("Sports Watch Data")
plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burnage")

plt.show()
```

Eğer başlık ve etiketler için özellikleri belirlemek istersek dictionary yapısından faydalanabiliriz.

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

```
x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])

font1 = {'family':'serif','color':'blue','size':20}
font2 = {'family':'serif','color':'darkred','size':15}

plt.title("Sports Watch Data", fontdict = font1)
plt.xlabel("Average Pulse", fontdict = font2)
plt.ylabel("Calorie Burnage", fontdict = font2)

plt.plot(x, y)
plt.show()
```

Başlıklara özellikler eklemenin yanı sıra, nerede bulunacaklarını özelleştirebiliriz.

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

x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])

plt.title("Sports Watch Data", loc = 'left')
plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burnage")

plt.plot(x, y)
plt.show()
```

Subplot

subplot() fonksiyonunu kullanarak aynı anda iki grafiği birden görselleştirebiliriz.

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

#plot 1:
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])

plt.subplot(1, 2, 1)
plt.plot(x,y)

#plot 2:
x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])

plt.subplot(1, 2, 2)
plt.plot(x,y)

plt.show()
```

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

#plot 1:
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])
```

```
plt.subplot(2, 1, 1)
plt.plot(x,y)

#plot 2:
x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])

plt.subplot(2, 1, 2)
plt.plot(x,y)

plt.show()
```

⁉

İki grafik arasındaki farkı bulalım.

Grafik Çeşitleri

Elimizdeki verileri görselleştirme araçları ile birçok şekilde ifade edebiliriz. Bunlardan en çok kullanılanları:

-Scatter Plot(Noktalı Grafik)

-Bar Plot (Bar grafiği)

-Histogramlar

-Pie Charts (Pasta Grafikleri)

Scatter Plot

```
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])
sizes = np.array([20,50,100,200,500,1000,60,90,10,300,600,800,75])

plt.scatter(x, y, s=sizes)

plt.show()
```

```
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])
colors = np.array([0, 10, 20, 30, 40, 45, 50, 55, 60, 70, 80, 90, 100])

plt.scatter(x, y, c=colors, cmap='viridis')

plt.colorbar()

plt.show()
```

Bar Plot

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

x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])

plt.bar(x,y)
plt.show()
```

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

x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])

plt.bar(x, y, color = "#4CAF50")
plt.show()
```

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

x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])
```

```
plt.bar(x, y, width = 0.1)
```

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

x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])

plt.barh(x, y, height = 0.1)
plt.show()
```

Histogram

Histogram bir frekans grafiğidir. Sıklık grafiği olarak da ifade edebiliriz.

Hangi aralıklarda hangi değerler var görmek istiyorsak histogram uygun bir görselleştirme seçeneğidir.

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

x = np.random.normal(170, 10, 250) #Random fonksiyonu ne yapar? İçine aldığı parametreler

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

Pie Charts (Pasta Grafikleri)

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

y = np.array([35, 25, 25, 15])

plt.pie(y)
plt.show()
```

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

y = np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]

plt.pie(y, labels = mylabels, startangle = 90)
plt.show()
```

Eğer bir dilimi diğerlerinden ayırıp öne çıkarmak istersek **explode** özelliğini kullanabiliriz.

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

y = np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]
myexplode = [0.2, 0, 0, 0]

plt.pie(y, labels = mylabels, explode = myexplode)
plt.show()
```

Eğer lejant eklemek istersek **legend()** özelliğini kullanırız.

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

y = np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]

plt.pie(y, labels = mylabels)
plt.legend()
plt.show()
```

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

y = np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]
mycolors = ["black", "hotpink", "b", "#4CAF50"]

plt.pie(y, labels = mylabels, colors = mycolors)
plt.show()
```