



Matplotlib

GRAPH PLOTTING

با نحوه نصب کتابخانه ها

و استفاده از آنها آشنایی داریم

```
pip install matplotlib
```

```
import matplotlib.pyplot as plt
```

بسیاری از کاربرد های مورد استفاده این کتابخانه در ماژولی به نام pyplot قرار دارد

plot()

اولین تابعی که به معرفی آن می‌پردازیم تابع plot می‌باشد
این تابع دو پارامتر دریافت می‌کند که هر دو آنها لیست یا آرایه ای از اعداد می‌باشند
سپس نقاط متناظر این دو آرایه را دو به دو روی محور مختصات رسم می‌کند

آرایه اول نقاط روی محور x و آرایه دوم نقاط روی محور y به حساب می‌آیند

در تمام برنامه ها برای نمایش نمودار های رسم شده با plot باید از دستور show() استفاده کرد
تا تمام نمودار ها نمایش داده شوند

مثال و توضیح بیشتر:
برنامه plot.py

نکته:

- علاوه بر نمودار خطی، می‌توان نمودار را به صورت نقطه به نقطه و گسسته نیز نمایش داد
برای این کار 'o' را به عنوان پارامتر سوم به تابع plot می‌دهیم

نکته:

- محدودیتی در اعداد وارد شده به عنوان داده وجود ندارد تنها نکته این است که تعداد اعداد در هر دو محور x و y برابر باشند در غیر این صورت با خطا مواجه خواهیم شد
- در صورت نیاز می‌توان برای محور x داده ای در نظر نگرفت در این صورت به صورت خودکار به داده های محور y اعداد $(0,1,2,3,4,5,6, \dots)$ نسبت داده خواهد شد

FMT

FMT یا (format string) ابزاریست برای تغییر ظاهر و رنگ نمودار و نشان دادن نقاط مهم

برای اعمال این تغییرات سه فاکتور مهم وجود دارند

- | | |
|-----------|-------------------------------|
| 1. Marker | (برای نشان دادن نقاط مهم) |
| 2. Line | (برای ویرایش نوع خطوط نمودار) |
| 3. color | (برای ویرایش رنگ نمودار) |

که این سه فاکتور به این فرم بدون فاصله کنار هم استفاده می‌شوند










```
plt.plot(x , y , 'MarkerLineColor')
```

FMT

سوال:

مقادیر این سه فاکتور چه چیز هایی می‌تواند باشد؟

Marker

o	
*	
D	
x	
^	
+	
s	
p	
h	

برای اطلاع از نشانگر های بیشتر به سایت رسمی matplotlib مراجعه کنید

[matplotlib.markers — Matplotlib 3.3.3 documentation](https://matplotlib.org/3.3.3/using_matplotlib/markers.html)

Line

:	نقطه چین	
-	خط	
--	خط چین	
-.	نقطه چین و خط چین	

color

b	آبی	m	بنفش
g	سبز	y	زرد
r	قرمز	k	مشکی
c	فیروزه ای	w	سفید

mec, mfc, ms

ms (marker size)

با استفاده از ms میتوان اندازه نقاط روی نمودار را تغییر داد
مقدار ms میتواند هر عدد دلخواهی باشد (اگر مقدار مورد نظر منفی یا صفر باشد نقطه ای نمایش داده نمیشود

`plt.plot(x , y , ms = a number)`

mec (marker edge color)

این آرگومان رنگ حاشیه نقاط را تعیین میکند
از جدول رنگ های صفحه قبل میتوان برای این مورد نیز استفاده کرد

mfc (marker face color)

این آرگومان نیز رنگ داخل نقاط روی نمودار را تعیین میکند

برچسب ها و عناوین نمودار

گاهها نیاز است تا خود یا محور های نمودار نامگذاری شوند. حال به بررسی روند انجام این کار می پردازیم

Function	Syntax	Example
xlabel	Plt.xlabel("str")	Plt.xlabel("Average Pulse")
ylabel	Plt.ylabel("str")	Plt.ylabel("Calorie Burnage")
title	Plt.title("str")	Plt.title("Calorie data")

Grid

با استفاده از این روش میتوان خط هایی عمودی، افقی یا هر دو را بر روی نمودار کشید تا نمودار خوانا تر و مفهوم تر شود

<code>plt.grid(axis = 'both')</code>	عمودی و افقی
<code>plt.grid(axis = 'y')</code>	عمود بر محور y
<code>plt.grid(axis = 'x')</code>	عمود بر محور x
<code>plt.grid(color = 'color')</code>	تعیین رنگ grid
<code>plt.grid(linestyle = '--')</code>	تعیین نوع خطوط
<code>plt.grid(linewidth = number)</code>	تعیین ضخامت خطوط

می‌توان از تمام آرگومان های این تابع همزمان استفاده کرد

```
plt.grid(axis = 'both' , color = 'r', linestyle = ':', linewidth = 0.2)
```

نمایش نمودار های متعدد

subplot

گاهها نیاز است تا به منظور مقایسه یا اهداف دیگر چندین نمودار در کنار هم رسم شوند: بدین منظور شکل خروجی باید به قطعاتی تقسیم شوند و در هر قطعه یک نمودار نمایش داده شود

برای این عمل، تابع subplot سه آرگومان دریافت میکند که نشان دهنده شمایل شکل خروجی است و شکل را مطابق آن تقسیم بندی میکند

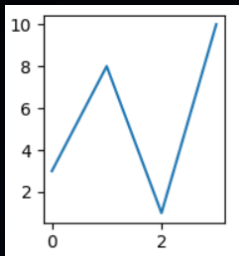
`Plt.subplot (rows , columns , the current plot)`

مکان نمودار فعلی تعداد ستون ها تعداد سطر های شکل

نکته:

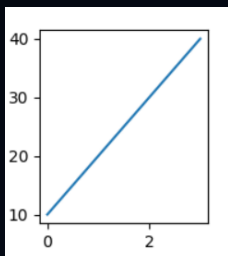
در استفاده از این روش باید ابتدا نقاط محور های مختصات تعریف شوند سپس از این تابع استفاده شده و در انتها از تابع plot استفاده کرد و سپس به سراغ نمودار بعدی رفت

در غیر این صورت نمودار ها یا با هم ترکیب میشوند یا نمودار های دیگر خالی میماند



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

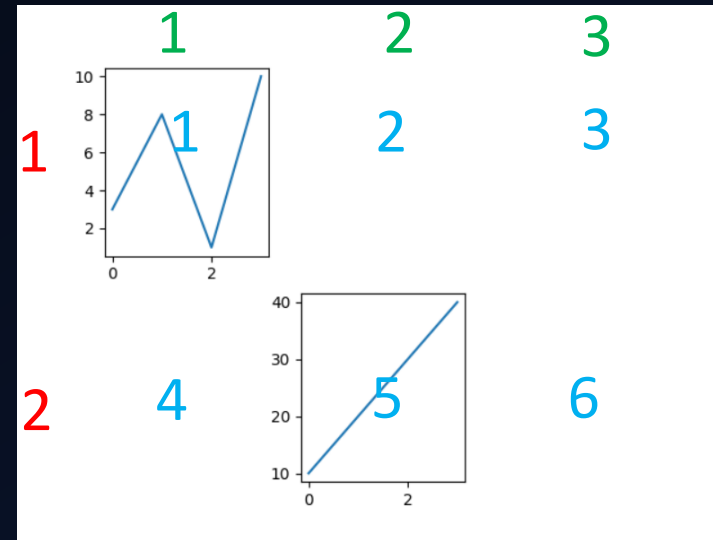
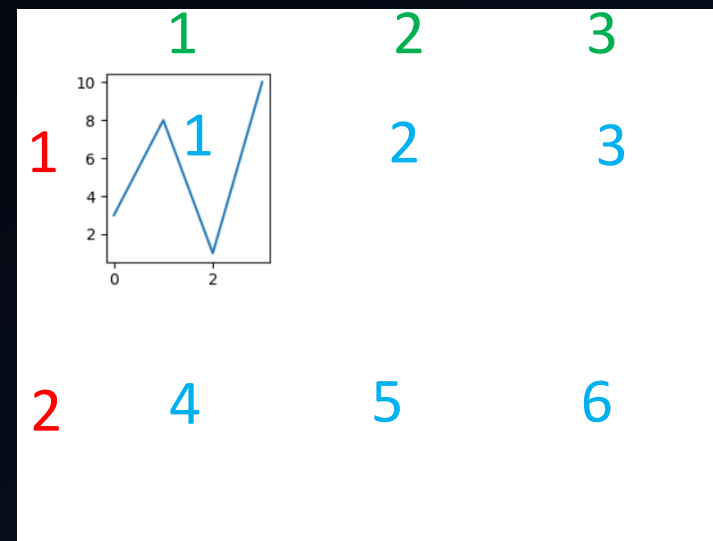
```
plt.subplot(2,3,1)
plt.plot(x,y)
```



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

```
plt.subplot(2,3,5)
plt.plot(x,y)
```

مثال:



در صورت نیاز میتوان به هر کدام از نمودار ها برچسب و عنوان داد

#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)
plt.title("....")
plt.xlabel("... ")
plt.ylabel("...")
```

#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.title("....")
plt.xlabel(".... ")
plt.ylabel(".... ")
```

#plot 3 , 4 , 5 ,6 ,

plt.show()

1. برای این کار ابتدا باید نقاط را مشخص کرد

2. سپس از تابع subplot استفاده کرد

3. بعد از آن تابع را رسم کرد (استفاده از تابع plot)

4. و در انتها برچسب ها (label) و عنوان (title) هر نمودار را مشخص کرد

5. سپس همین شرایط را برای رسم نمودار بعدی تکرار کرد

6. و پس از پایان با تابع show نمودار ها را به نمایش گذاشت

همچنین میتوان برای کل شکل یک عنوان کلی استفاده کرد (plt.suptitle("str") (مکان نوشتن کد اهمیتی ندارد)

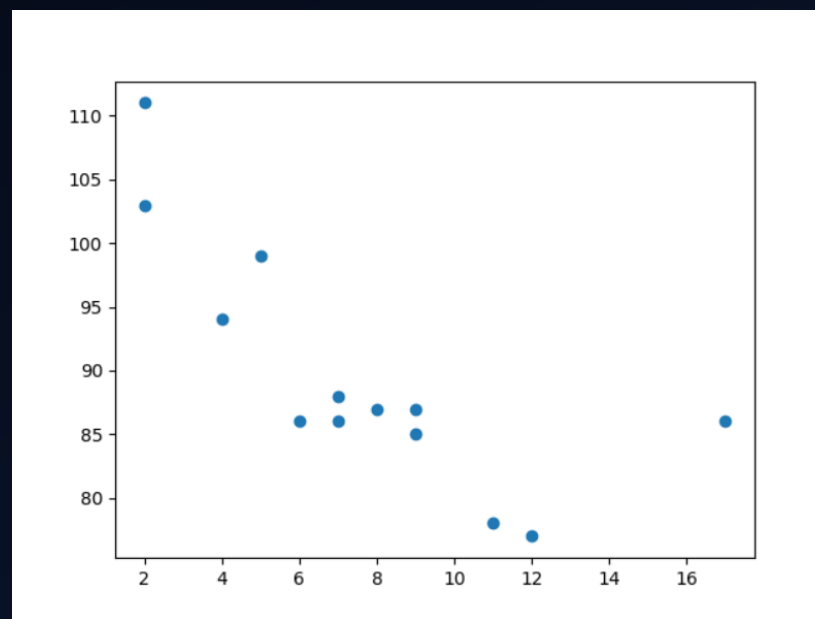
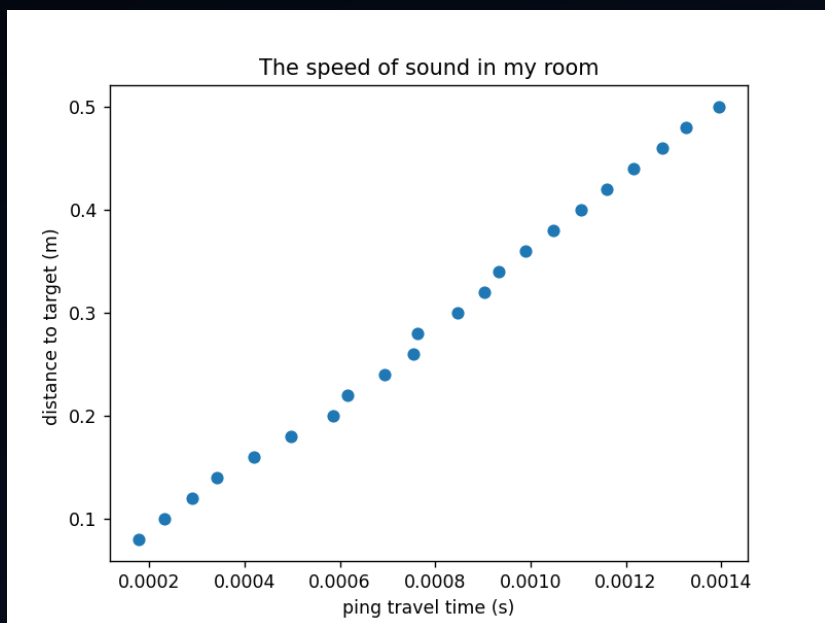
scatter

تا به حال با تابع plot آشنا شدیم
حال دومین تابعی که کار ترسیم داده ها را بر عهده دارد بررسی میکنیم

هنگامی که داده ها پیرو توابع ریاضی نیستند یا کاملاً تصادفی هستند
یا خود ما میلی به پیوسته نشان دادن آنها نداریم از این تابع استفاده میکنیم

`plt.scatter(x,y)`

که x و y آرایه هایی هم اندازه در تعداد هستند



با استفاده از c میتوان رنگ یک یا همه نقاط را تغییر داد

```
plt.scatter(x , y , c ="color")
```

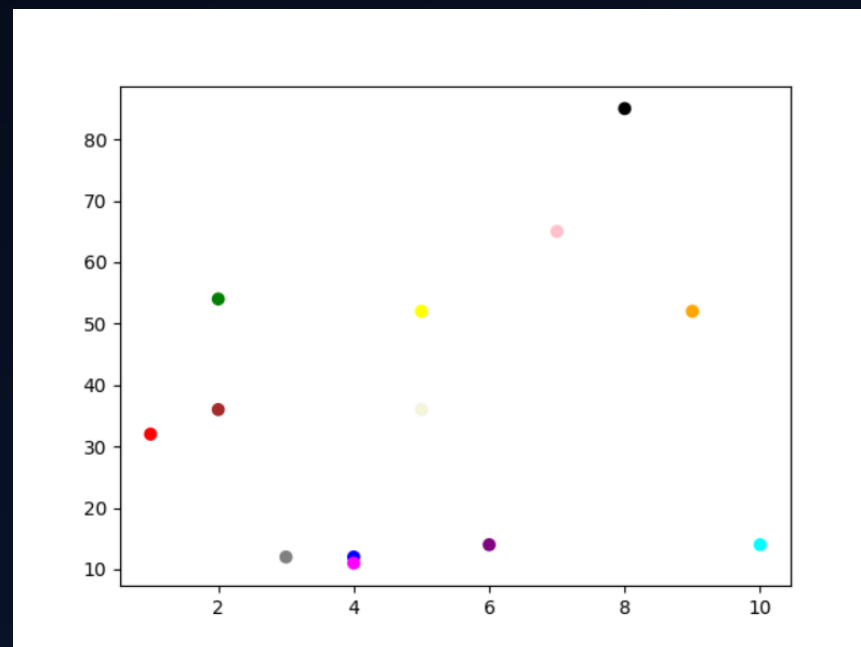
در صورتی که نیاز باشد تا هر نقطه ای رنگ مخصوص خود را داشته باشد باید آرایه ای هم اندازه از نظر تعداد ایندکس ها با آرایه های x و y از رنگ های مورد نیاز ساخت و در نهایت در تابع scatter وارد کنیم

```
x = np.array([1,2,4,5,7,8,9,6,5,2,3,10,4])
```

```
y = np.array([32,54,12,52,65,85,52,14,36,36,12,14,11])
```

```
colors = np.array(["red","green","blue","yellow","pink","black","orange","purple","beige","brown","gray","cyan","magenta"])
```

```
plt.scatter(x, y, c = colors)
```



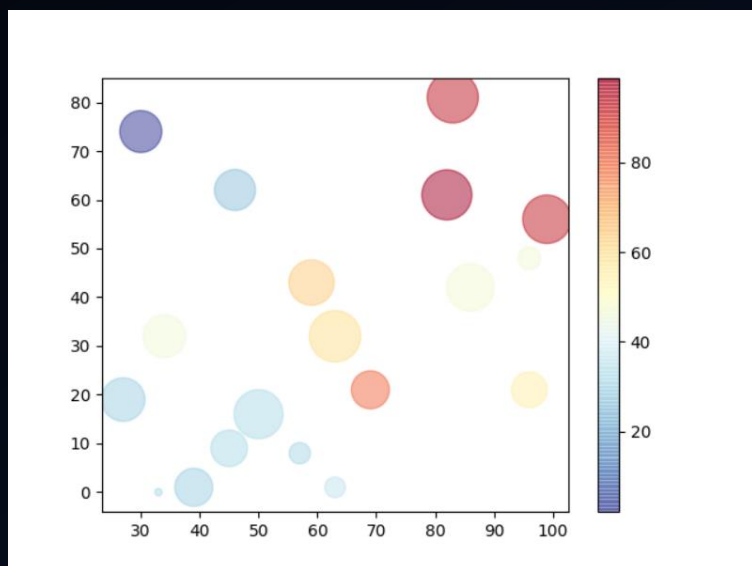
Color map

روشی است برای انتصاب رنگ به داده ها به طور منظم و با قاعده در این روش نیازی به وارد کردن دستی نام رنگ ها وجود ندارد

برای مثال

میخواهیم یک سیستم را به دفعات زیاد در شرایط مختلف آزمایش کنیم فرض کنیم نتیجه این آزمایش به عنوان خروجی برای هر دفعه آزمایش دو کمیت است (دما، احتمال، انرژی و....) حال با این روش میتوان یکی از کمیت ها (مثل دما، چگالی، تندی، لختی دورانی و) را با یک طیف رنگی و کمیت دیگر را (مثل اندازه حرکت، جرم، احتمال، انرژی) توسط اندازه نقاط نمایش داد

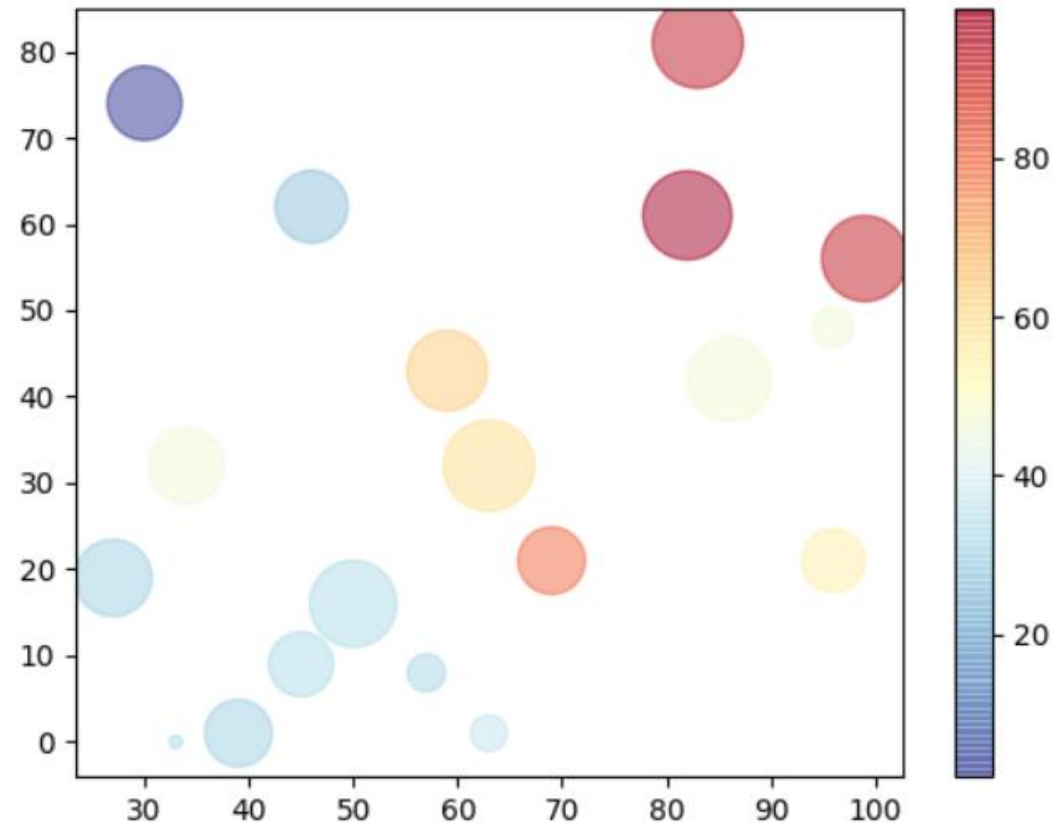
در این روش به هر داده عددی حسابی $[0, \infty)$ نسبت داده میشود و آن عدد تبدیل به رنگی مشخص میشود و بدین صورت هر داده رنگ مختص خود را میگیرد



Color map

برای مثال

میخواهیم یک سیستم را به دفعات زیاد در شرایط مختلف آزمایش کنیم
فرض کنیم نتیجه این آزمایش به عنوان خروجی برای هر دفعه آزمایش دو کمیت است (دما، احتمال، انرژی و....)
حال با این روش میتوان یکی از کمیت ها (مثل دما، چگالی، تندی، لختی دورانی و) را با یک طیف رنگی
و کمیت دیگر را (مثل اندازه حرکت، جرم، احتمال، انرژی) توسط اندازه نقاط نمایش داد



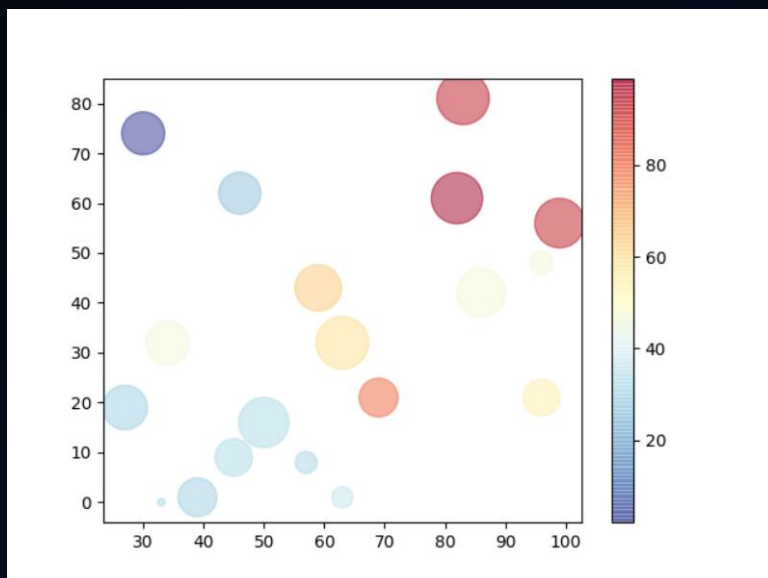
Color map

روشی است برای انتساب رنگ به داده ها به طور منظم و با قاعده در این روش نیازی به وارد کردن دستی نام رنگ ها وجود ندارد

برای مثال

میخواهیم یک سیستم را به دفعات زیاد در شرایط مختلف آزمایش کنیم فرض کنیم نتیجه این آزمایش به عنوان خروجی برای هر دفعه آزمایش دو کمیت است (دما، احتمال، انرژی و....) حال با این روش میتوان یکی از کمیت ها (مثل دما، چگالی، تندی، لختی دورانی و) را با یک طیف رنگی و کمیت دیگر را (مثل اندازه حرکت، جرم، احتمال، انرژی) توسط اندازه نقاط نمایش داد

در این روش به هر داده عددی حسابی $[0, \infty)$ نسبت داده میشود و آن عدد تبدیل به رنگ و اندازه ای مشخص میشود و بدین صورت هر داده رنگ مختص خود را میگیرد



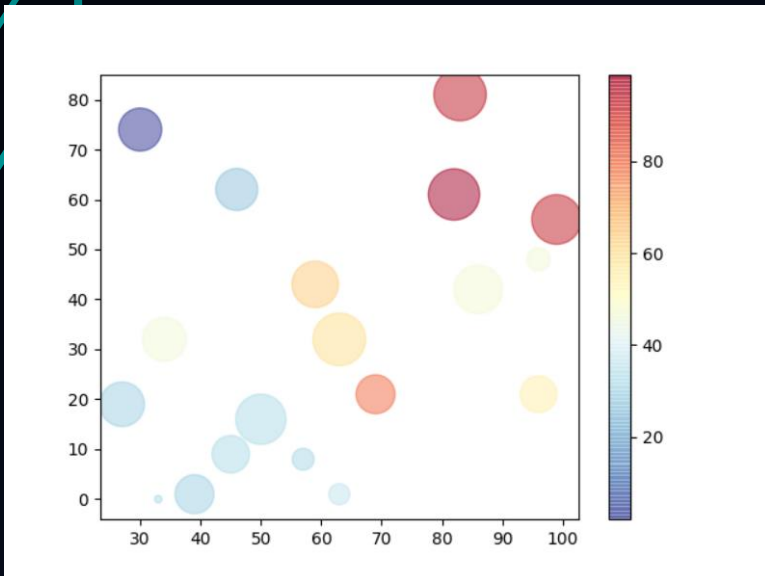
مثال:

```
x = [34 ,39, 96, 57, 63, 69, 27, 86, 96, 45, 99, 46, 33, 82, 30, 63, 59, 50, 86, 83]
y = [32 ,1, 21, 8, 1, 21, 19, 75, 48, 9, 56, 62, 0 ,61 ,74 ,32 ,43 ,16 ,42 ,81]
colors = [46, 29, 55, 31, 36, 80, 29, 17, 46, 32, 93, 26, 32, 99, 2 ,60, 64, 32, 47, 94]
sizes = [680 ,540, 480, 170, 160, 540, 710, 0 ,190 ,500 ,880 ,630, 20 ,940 ,660 ,990 ,770 ,900
,840, 980]
plt.scatter(x, y, c=colors, s=sizes, alpha=0.5, cmap='RdYlBu_r')
plt.colorbar()
plt.show()
```

در این روش c کلمه اختصاری color و s مختصر size میباشد
در ابتدا به ازای هر نقطه یک عدد به طور متناظر به عنوان رنگ و به عدد به عنوان اندازه به هر نقطه نظیر میشود
آرگومان alpha نیز به شفافیت این نقاط مربوط است (گاهی نقاط روی هم قرار می‌گیرند و اگر نقاط شفاف نباشد ممکن است
نقاط زیر یکدیگر پنهان شوند)

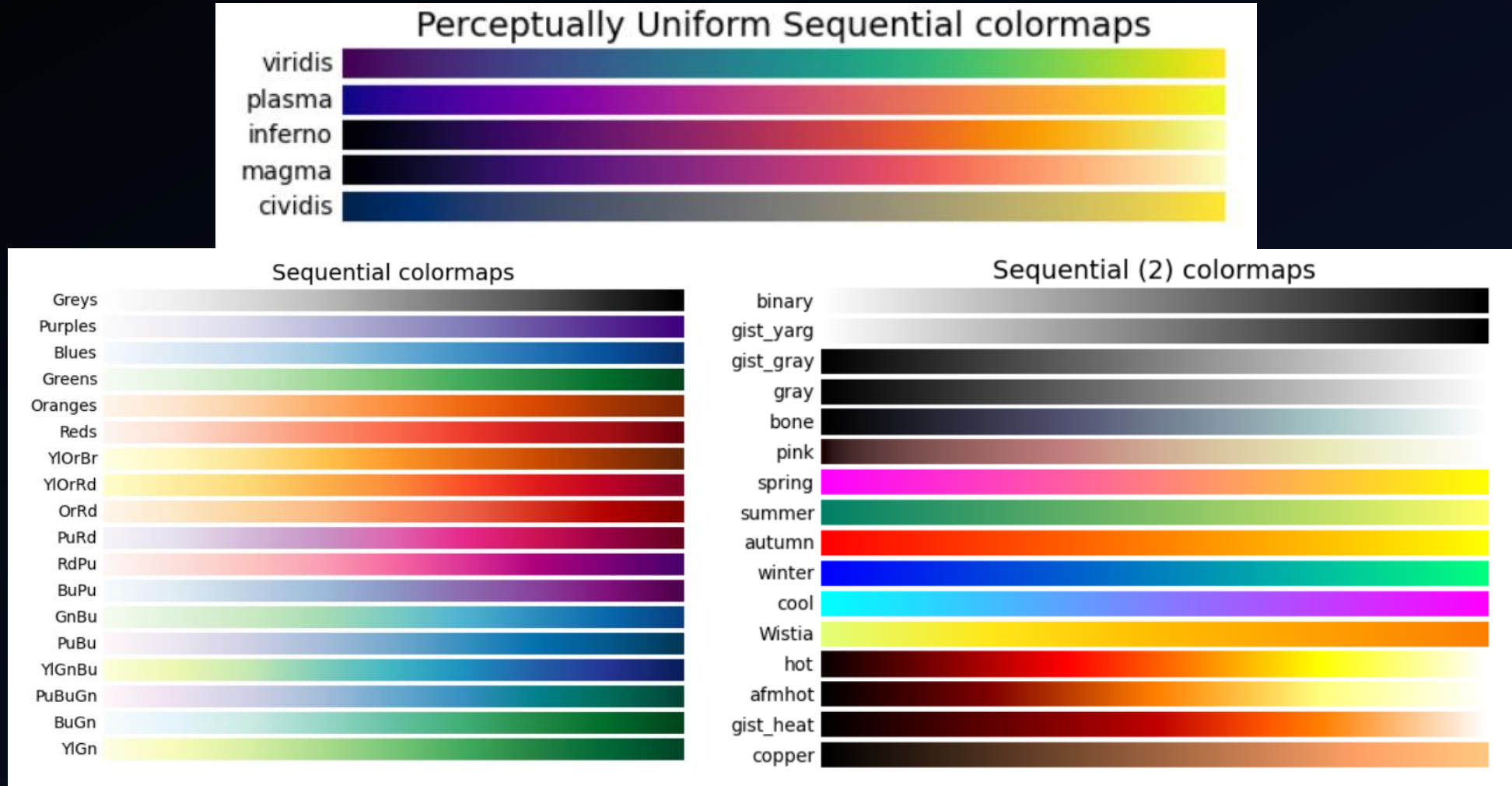
آرگومان cmap نیز که به معنای colormap است پالت رنگ های طیف رنگی ما را تعیین میکند

plt.colorbar() نیز برای نمایش نقشه رنگی به کار میرود



مثال:

آرگومان cmap نیز که به معنای colormap است پالت رنگ های طیف رنگی ما را تعیین میکند

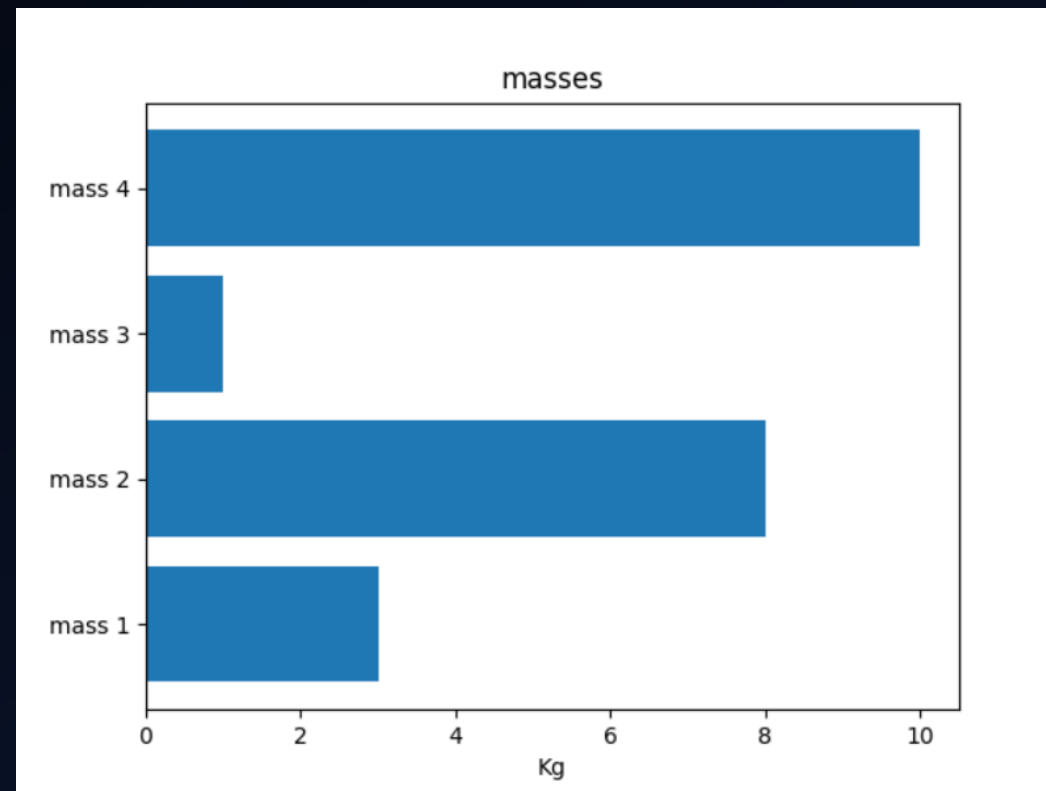
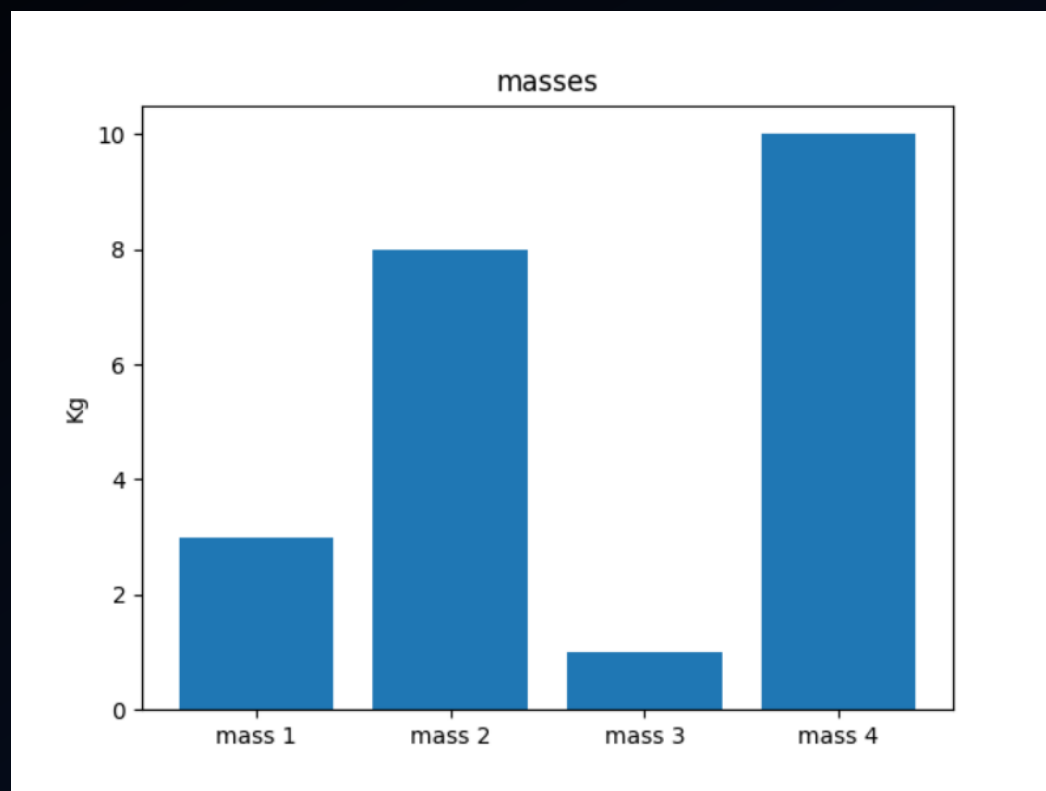


[Choosing Colormaps in Matplotlib — Matplotlib 3.5.0 documentation](#)

اطلاعات بیشتر

bar

سومین ابزار رسم بعد از plot و scatter است
با این تفاوت که محور x یا y لزوماً نباید عدد باشد و میتواند str هم باشد

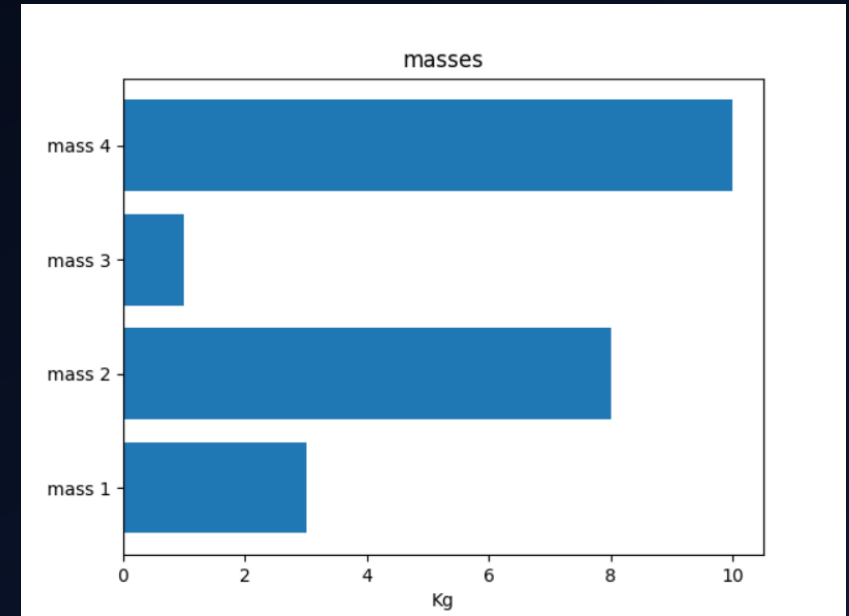
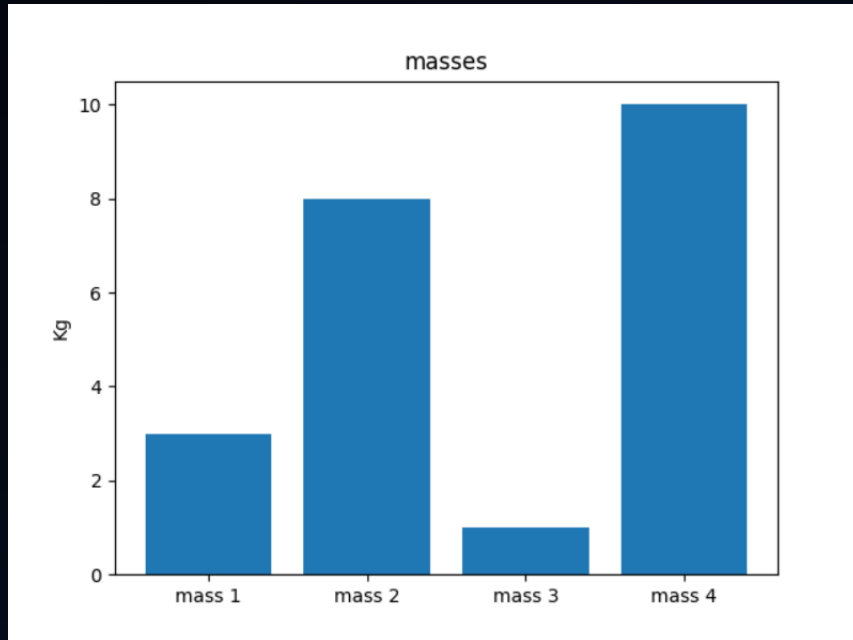


طرز کار این تابع نیز مشابه توابع قبل می‌باشد

```
x = np.array(["mass 1", "mass 2", "mass 3", "mass 4"])
y = np.array([3, 8, 1, 10])
plt.ylabel("Kg")
plt.title("masses")
plt.bar(x,y)
plt.show()
```

1. ابتداه محور های x و y را مشخص میکنیم
2. سپس تابع `bar` را فراخوانی میکنیم
3. و در نهایت نمودار را رسم میکنیم

در صورتی که مایلید تا نمودار را افقی رسم کنید باید از تابع `barh()` استفاده کنید
(Horizontal bar)



رنگ و ضخامت ستون ها

برای تغییر رنگ میتوان از آرگومان color در تابع bar استفاده کرد
لیست کامل رنگ های پشتیبانی شده :

[HTML Color Names \(w3schools.com\)](https://www.w3schools.com/html/html_color_names.asp)

تعیین ضخامت ستون ها:

1. برای ستون های عمودی میتوان از آرگومان width استفاده کرد و آن را برابر با یک عدد قرار داد

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

2. برای ستون های افقی از hieght استفاده میکنیم

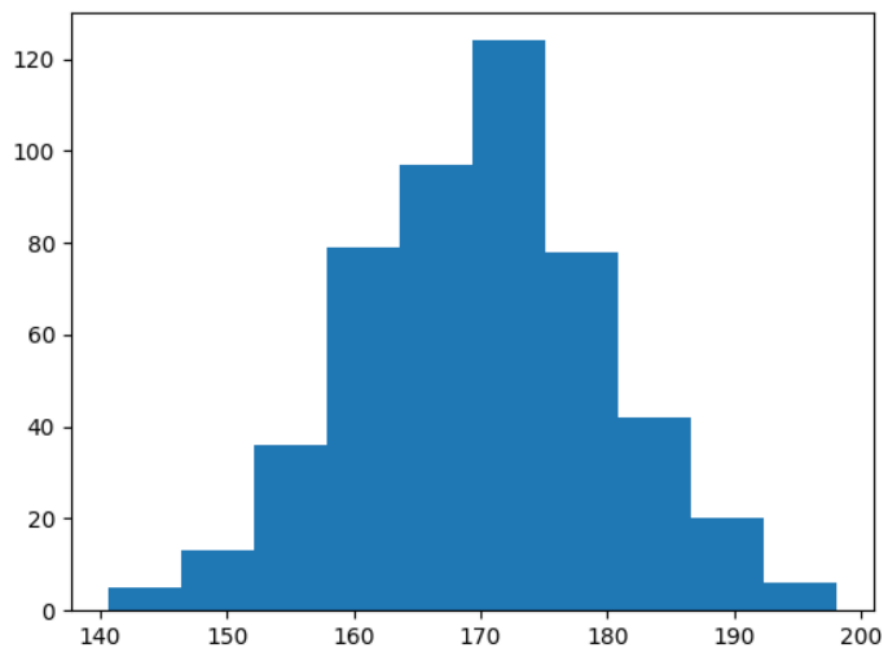
```
plt.barh(x, y, height = number)
```

histogram

چهارمین نوع رسم مورد بررسی ما histogram ها میباشند

هیستوگرام ها نمودار هایی هستند که توزیع تکرار یک کمیت را نشان میدهد

برای مثال توزیع قد 500 نفر که بین 140cm و 200cm هستند طبق داده های تصادفی بدین شرح است



برای رسم این نوع نمودار از $\text{hist}(x)$ استفاده میکنیم
که در آن x داده های ما هستند

نکته:

برای ساخت داده های تصادفی میتوان از کتابخانه numpy استفاده کرد
برای مثال برای ساخت داده های نرمال (توزیع گاوسی) از این روش میتوان استفاده کرد

```
data = np.random.normal(x , y , z)
```

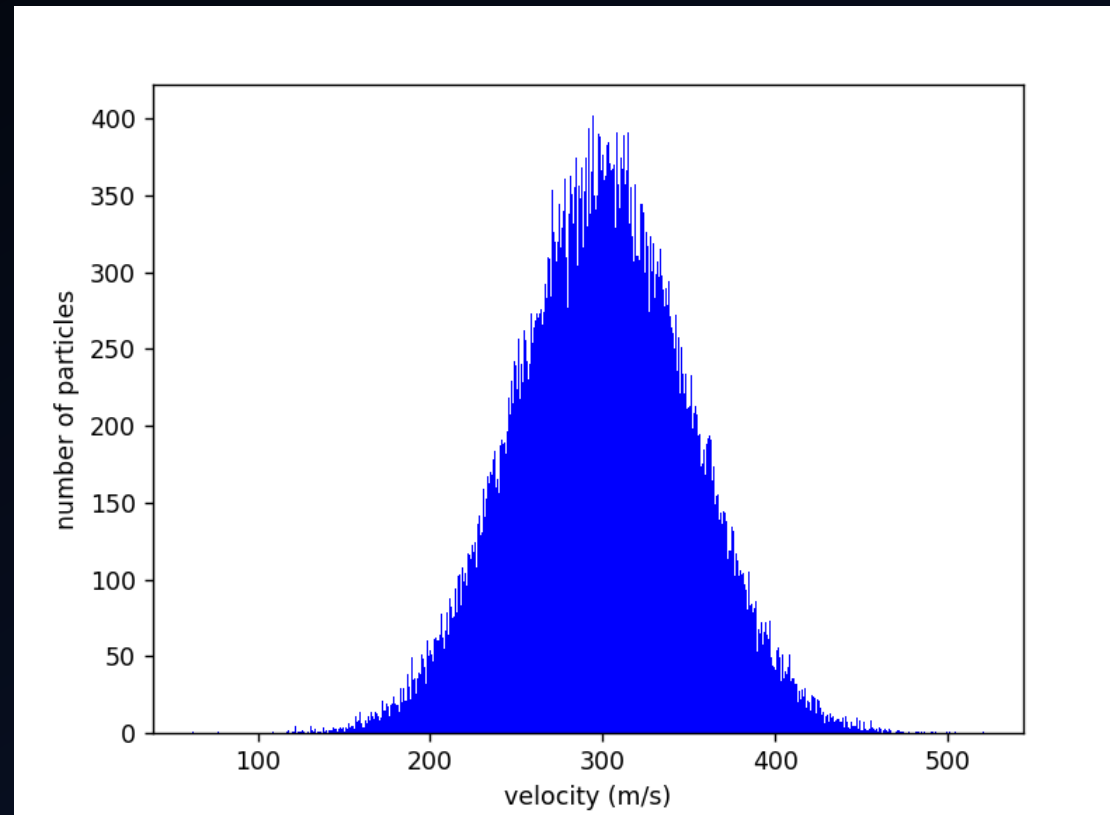
x میانگین داده ها (بیشتر داده های تولید شده نزدیک این عدد هستند)
y انحراف معیار میباشد
Z تعداد داده های تولید شده

مثال:

فرض کنید 100000 مولکول داریم که هر کدام سرعت خود را دارند و این سرعت ها تابع توزیع گاوسی هستند
برای نمایش نموداری این سیستم بدین صورت عمل میکنیم

```
x = np.random.normal(300,50,100000)
plt.hist(x,1000,color="blue")
plt.xlabel("velocity (m/s)")
plt.ylabel("number of particles")
plt.show()
```

عدد 1000 در تابع hist نشان دهنده تعداد ستون هاست



pie

و در آخر تابع pie که برای نمایش نمودار های دایره ای استفاده میشود

همانند توابع دیگر ابتدا آرایه اعداد مورد نظر را میسازیم
سپس از تابع pie استفاده میکنیم

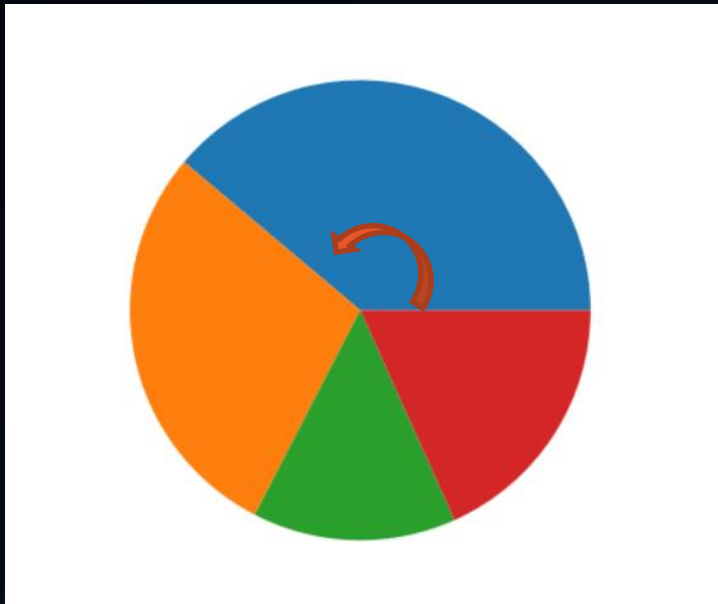
```
y = np.array([68,50,25,32])
```

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

اگر مساحت کل دایره ۱ باشد آنگاه :
مساحت و زاویه اشغال شده توسط هر داده روی نمودار طبق فرمول های زیر محاسبه میشود

$$area = \frac{data\ value}{sum\ of\ all\ values} \leq 1$$

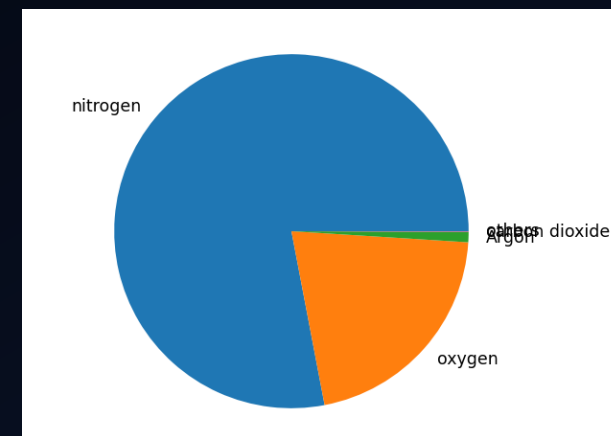
$$angel = \frac{data\ value \times 360}{sum\ of\ all\ values} \leq 360^{\circ}$$



رسم تکه های نمودار از $\theta = 0$ شروع شده و در جهت پادساعتگرد ادامه میابد

در صورت نیاز برای برچسب زدن بر قسمت های نمودار از پارامتر label استفاده میکنیم

```
y = np.array([78,21,0.93,0.04,0.029])  
my_labels = ["nitrogen","oxygen","Argon","carbon dioxide","others"]  
plt.pie(y,labels=my_labels)  
plt.show()
```



در شرایطی که عناوین و برچسب ها ناخوانا باشند از تابع legend استفاده میکنیم

```
y = np.array([78,21,0.93,0.04,0.029])  
my_labels = ["nitrogen","oxygen","Argon","carbon dioxide","others"]  
plt.pie(y)  
plt.legend(my_labels, title = "air compositions",loc = "best")
```

