Python数据可视化之matplotlib精进

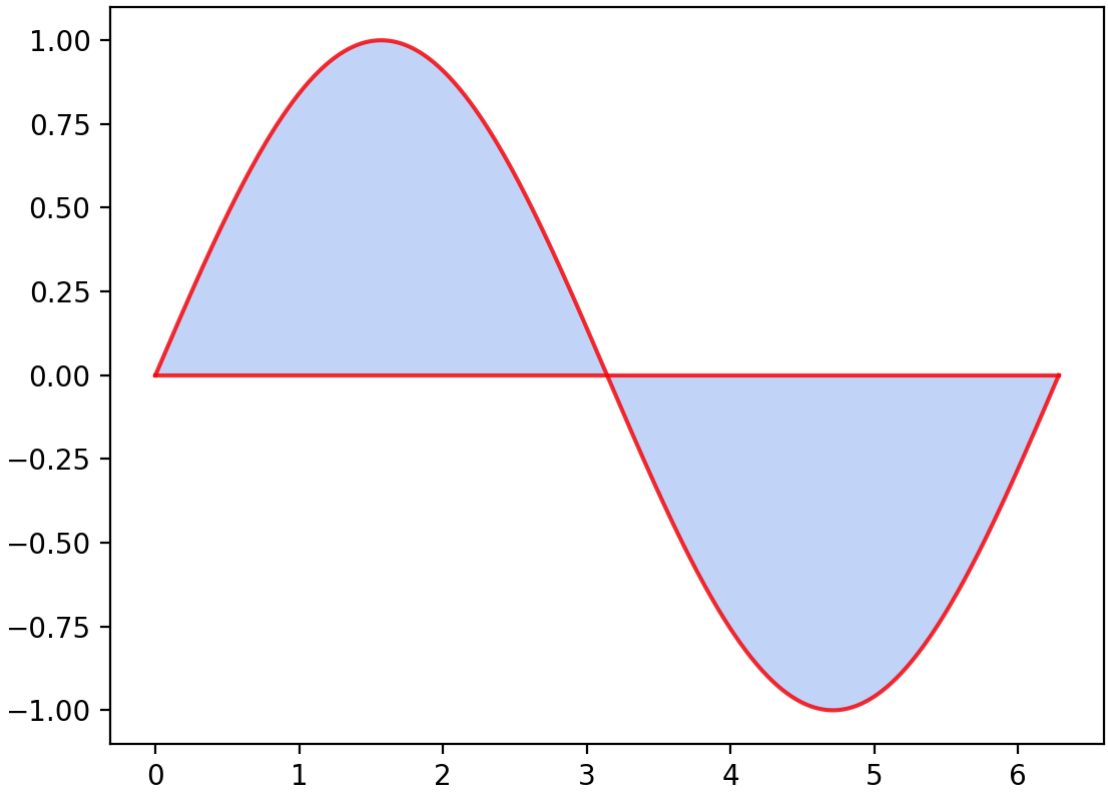
一、向图形填充颜色

1.1 fill()——多边形颜色填充

`plt.fill(x,y,color,alpha)`：填充函数曲线与坐标轴之间的区域

- x，y：是一个x，y的序列，每个多边形由其节点x和y的位置列表定义
- color：填充的颜色
- alpha：透明度

```
x = np.linspace(0,2*np.pi,500)
y = np.sin(x)

plt.fill(x,y,color="cornflowerblue",alpha=0.4)

plt.plot(x,y,color="r",alpha=0.8)
plt.plot([x[0],x[-1]],[y[0],y[-1]],color="r",alpha=0.8)
plt.show()
```



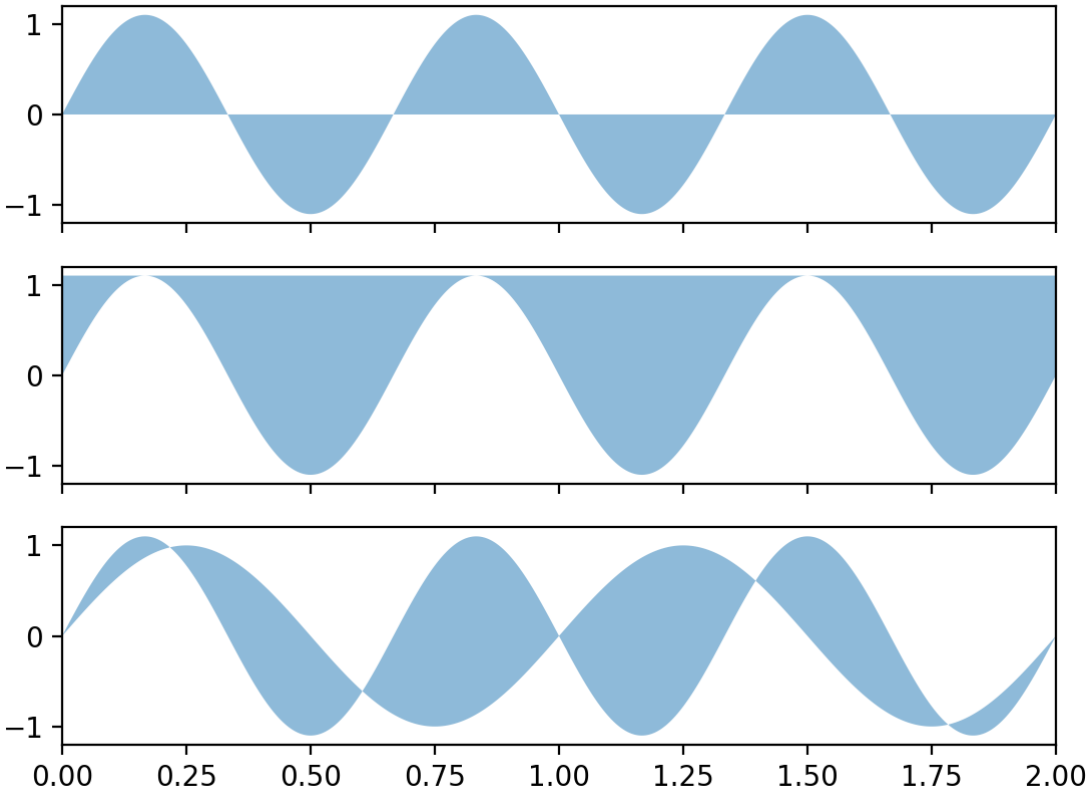
1.2 fill_between()——交叉颜色填充

`fill_between()`：填充两个函数曲线之间的部分，对应函数 `fill_betweenx()`

- x:一个序列，定义曲线的节点的x坐标
- y1:可以是一个序列，定义第一条曲线的y坐标
- y2:可以是一个序列，定义第二条曲线的y坐标
- where：一个可选参数，布尔值，y1>y2表示y1线在上时填充，y1<y2表示y2线在上时填充

- interpolate :看图中会发现有些填充的地方会有空白，此参数为Tur会自动填充
- facecolor：要填充的颜色

```
x = np.linspace(0,2,500)
y1 = np.sin(2*np.pi*x)
y2 = 1.1*np.sin(3*np.pi*x)
fig,ax = plt.subplots(3,1,sharex="all")

ax[0].fill_between(x,0,y2,alpha=0.5)

ax[1].fill_between(x,y2,1.1,alpha=0.5)

ax[2].fill_between(x,y1,y2,alpha=0.5)

plt.show()
```



二、使用patches绘制几何图形

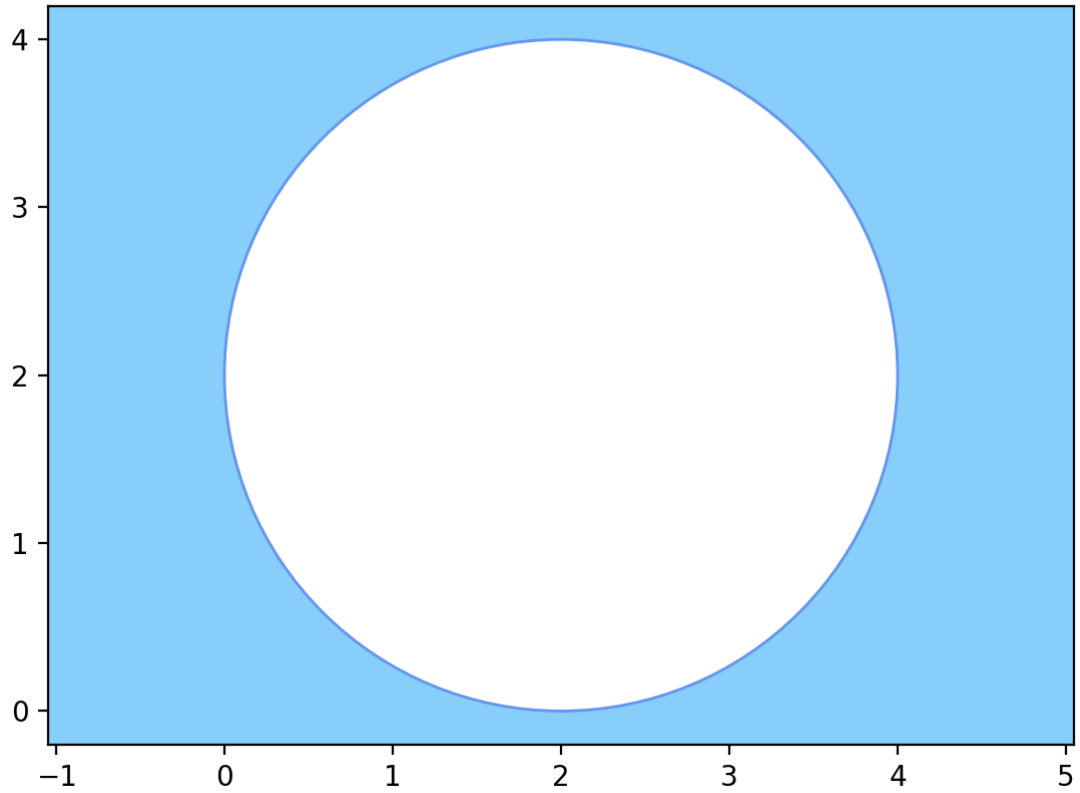
`from matplotlib.patches import Circle, Ellipse, Rectangle , Arc, Wedge`

2.1 Circle()——圆

`Circle((2,2),radius,facecolor,edgecolor)`

- (2,2)：圆的中心的坐标位置。
- radius：圆的半径大小。
- facecolor：圆的填充颜色。
- edgecolor：圆的轮廓的颜色。

```
fig,ax = plt.subplots(1,1)
rectangle = ax.patch
rectangle.set_facecolor("lightskyblue")

circle = Circle((2,2),radius=2,facecolor="white",edgecolor="cornflowerblue")

ax.add_patch(circle)
ax.axis([-1,5,-1,5])     #调整x轴y轴的坐标轴显示
ax.set_yticks(np.arange(-1,6,1))     #调整刻度线位置
ax.axis("equal")         #调整刻度线变化量
plt.subplots_adjust(left=0.1)

plt.show()
```

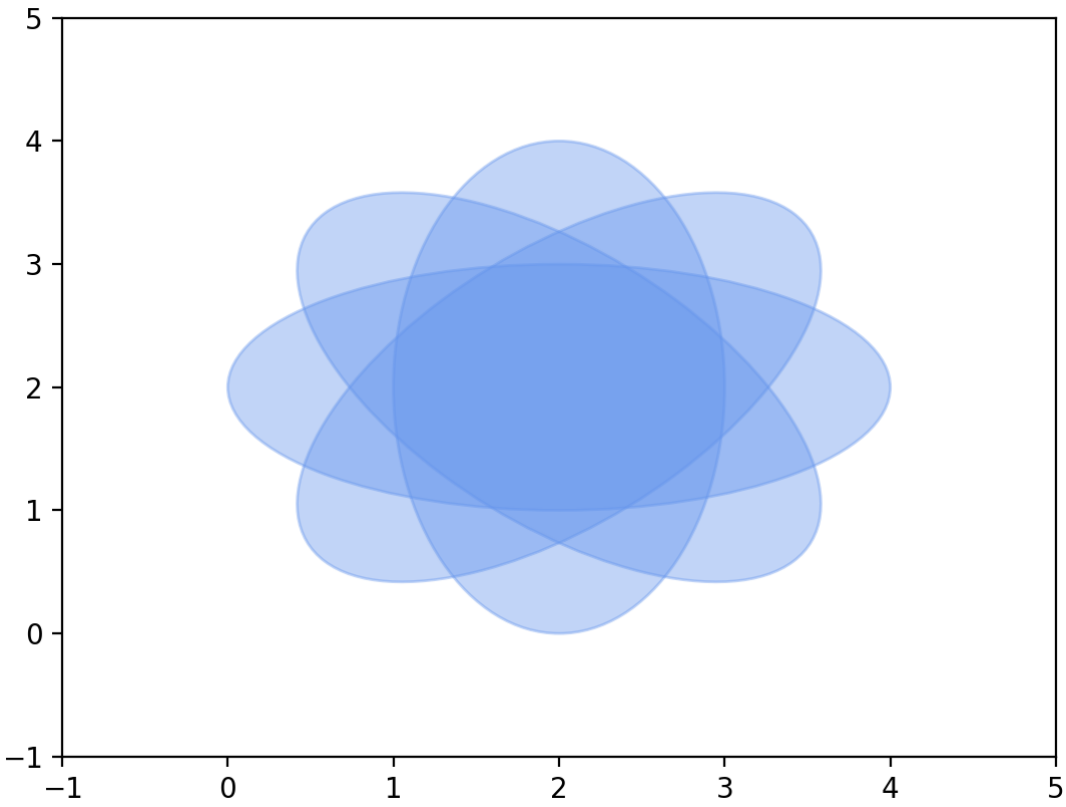


2.2 Ellipse()——椭圆

`Ellipse(xy, width, height, angle=0, **kwargs)`

- xy(float, float) , 椭圆中心的坐标
- width:float, 横坐标的长度
- height：float, 纵坐标的长度
- angle: float, 逆时针旋转的角度，默认值为0

```
fig,ax = plt.subplots(1,1,subplot_kw={"aspect":"equal"})

angles = np.linspace(0,135,4)

ellipse = [Ellipse((2,2),4,2,a) for a in angles]
for elle in ellipse:
    ax.add_patch(elle)
    elle.set_alpha(0.4)
    elle.set_color("cornflowerblue")

ax.axis([-1,5,-1,5])
plt.show()
```



2.3 Rectangle()——矩形

`Rectangle(xy=(0,0),width,height,angle,**kwargs)`

- xy(float, float) , 矩形左下角的坐标
- width:float, 横坐标的长度
- height：float, 纵坐标的长度
- angle: float, 逆时针旋转的角度，默认值为0

2.4 Arc()——圆弧和楔形

`Arc((2.5,1.1),width,height,angle,theta1,theta2,color,alpha)`

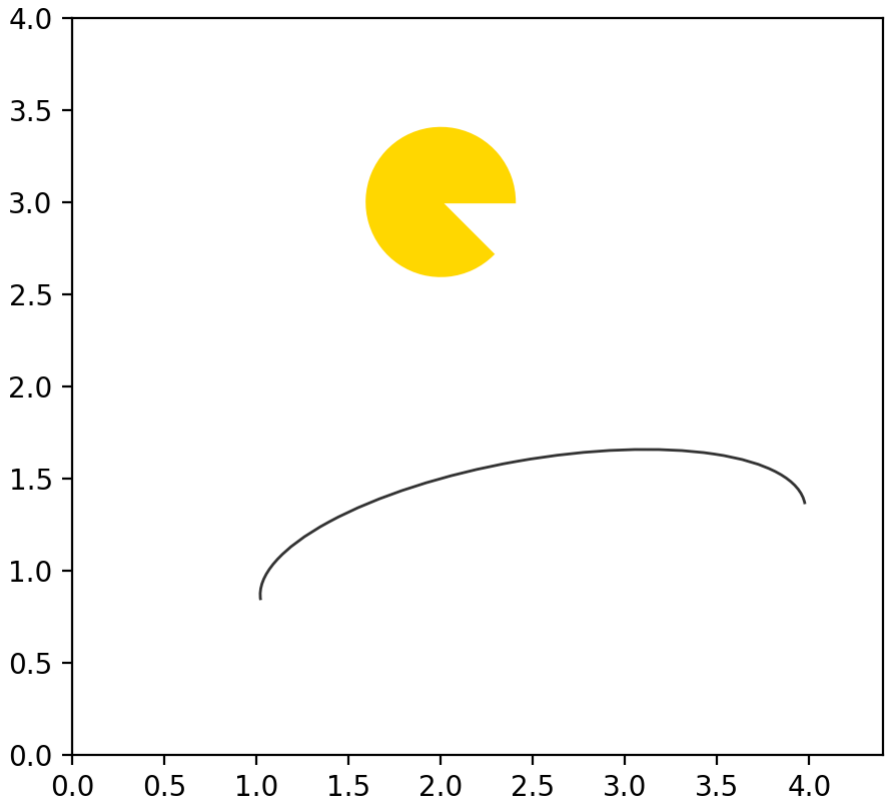
- (2.5,1.1)：圆弧的中心位置的坐标。
- width：圆弧的宽度。
- height：圆弧的高度。
- angle：圆弧的逆时针旋转的角度。
- theta1：圆弧起点处的角度。
- theta2：圆弧终点处的角度。

- color：圆弧的颜色。
- alpha：圆弧的透明度。

`Wedge((2,7),r,theta1,theta2,color)`

- (2,7)：楔形的中心位置的坐标。
- r：楔形的半径。
- theta1：楔形起始位置的角度（逆时针方向旋转）。
- theta2：楔形终止位置的角度（逆时针方向旋转）。
- color：楔形的填充区域颜色。
- width：可以绘制圆环，表示外圆和内圆的半径差

```
fig,ax = plt.subplots(subplot_kw={"aspect":"equal"})

base = Arc((2.5,1.1),3,1,angle=10,theta1=0,theta2=180,color="k",alpha=0.8)
left_arm_joinstyle1 =  Wedge((2,3),0.4,0,315,color="gold")

pol=[base,left_arm_joinstyle1]
for pln in pol:
    ax.add_patch(pln)
ax.axis([0,4.4,0,4])
plt.show()
```



三、组合展示统计图形

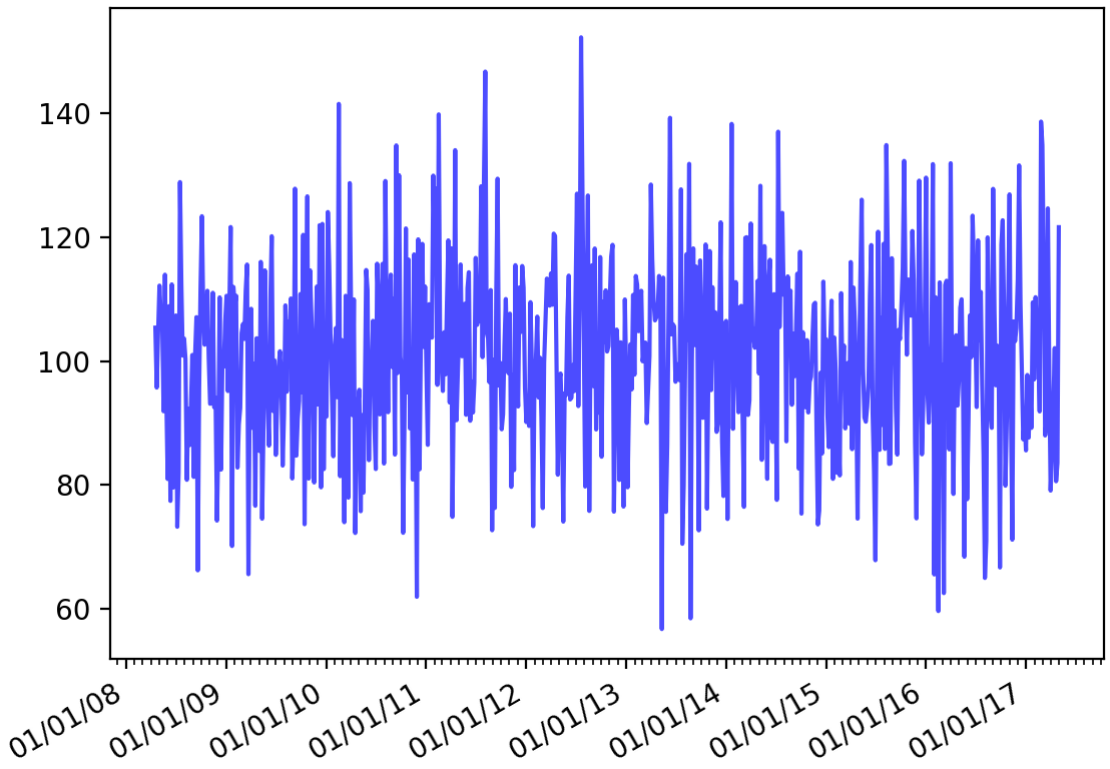
3.2 plot_date()日期型时间序列图

`plot_date()`

- dates：如果参数 xdate 的取值是 True，dates 就被理解成 matplotlib 的日期。
- y：对应 dates 的 *y* 轴数值。
- "b-"：折线图的线条样式和颜色。

- xdate：参数 xdate 的默认取值是 True，*x* 轴会被理解成 matplotlib 的日期。
- alpha：设置线条的颜色透明度。

```
import datetime
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import numpy as np
fig,ax = plt.subplots()
months = mdates.MonthLocator() # a Locator instance

dateFmt = mdates.DateFormatter("%m/%d/%y") # a Formatter instance

ax.xaxis.set_major_formatter(dateFmt)
ax.xaxis.set_minor_locator(months)
# set appearance parameters for ticks,ticklabels,and gridlines
ax.tick_params(axis="both",direction="out",labelsize=10)
date1 = datetime.date(2008, 4, 17)
date2 = datetime.date(2017, 5, 4)
delta = datetime.timedelta(days=5)
dates = mdates.drange(date1, date2, delta)
y = np.random.normal(100,15,len(dates))
ax.plot_date(dates,y,"b-",alpha=0.7)
fig.autofmt_xdate()
plt.show()
```



3.5 设置一般化的日期刻度线

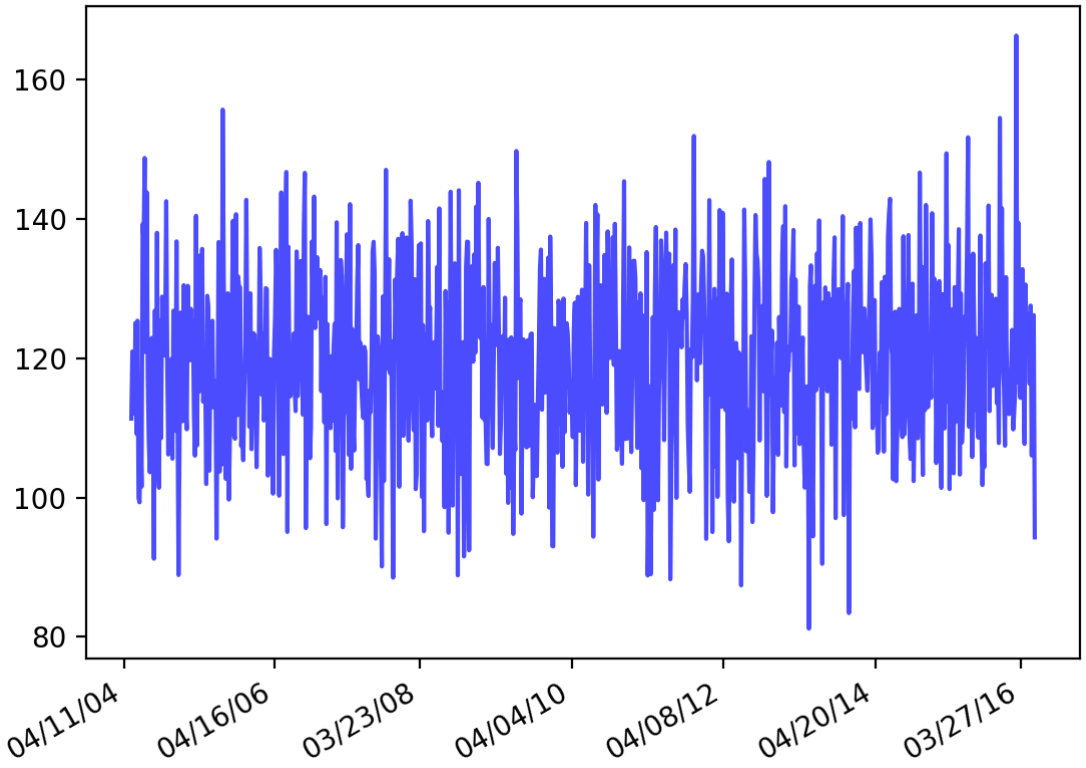
`import matplotlib.dates as mdates`

类 rrule构建的一个简单包装器，可以实现任意刻度线的定制化的目标。

- freq：可以取值 YEARLY、MONTHLY、WEEKLY、DAILY、HOURLY、MINUTELY 或SECONDLY，其中，YEARLY 的取值是 0。

- interval：每个 freq 下的间隔区间。如果使用 freq 中的 YEARLY，interval 的取值是 2，就表示以每两年作为年份的间隔区间。
- byeaster：复活节（周日）的滞后天数。如果传递参数值 0，就会产生复活节（周日）当天的日期。

```
import datetime
import matplotlib.dates as mdates

fig,ax = plt.subplots()

# tick every 5th easter
rule = mdates.rrulewrapper(mdates.YEARLY,byeaster=0,interval=2)
loc = mdates.RRuleLocator(rule) # a Locator instance
dateFmt = mdates.DateFormatter("%m/%d/%y") # a Formatter instance

# format the ticks
ax.xaxis.set_major_locator(loc)
ax.xaxis.set_major_formatter(dateFmt)

# set appearance parameters for ticks,ticklabels,and gridlines
ax.tick_params(axis="both",direction="out",labelsize=10)
date1 = datetime.date(2004, 5, 17)
date2 = datetime.date(2016, 6, 4)
delta = datetime.timedelta(days=5)
dates = mdates.drange(date1, date2, delta)
y = np.random.normal(120,12,len(dates))
ax.plot_date(dates,y,"b-",alpha=0.7)
fig.autofmt_xdate()
plt.show()
```

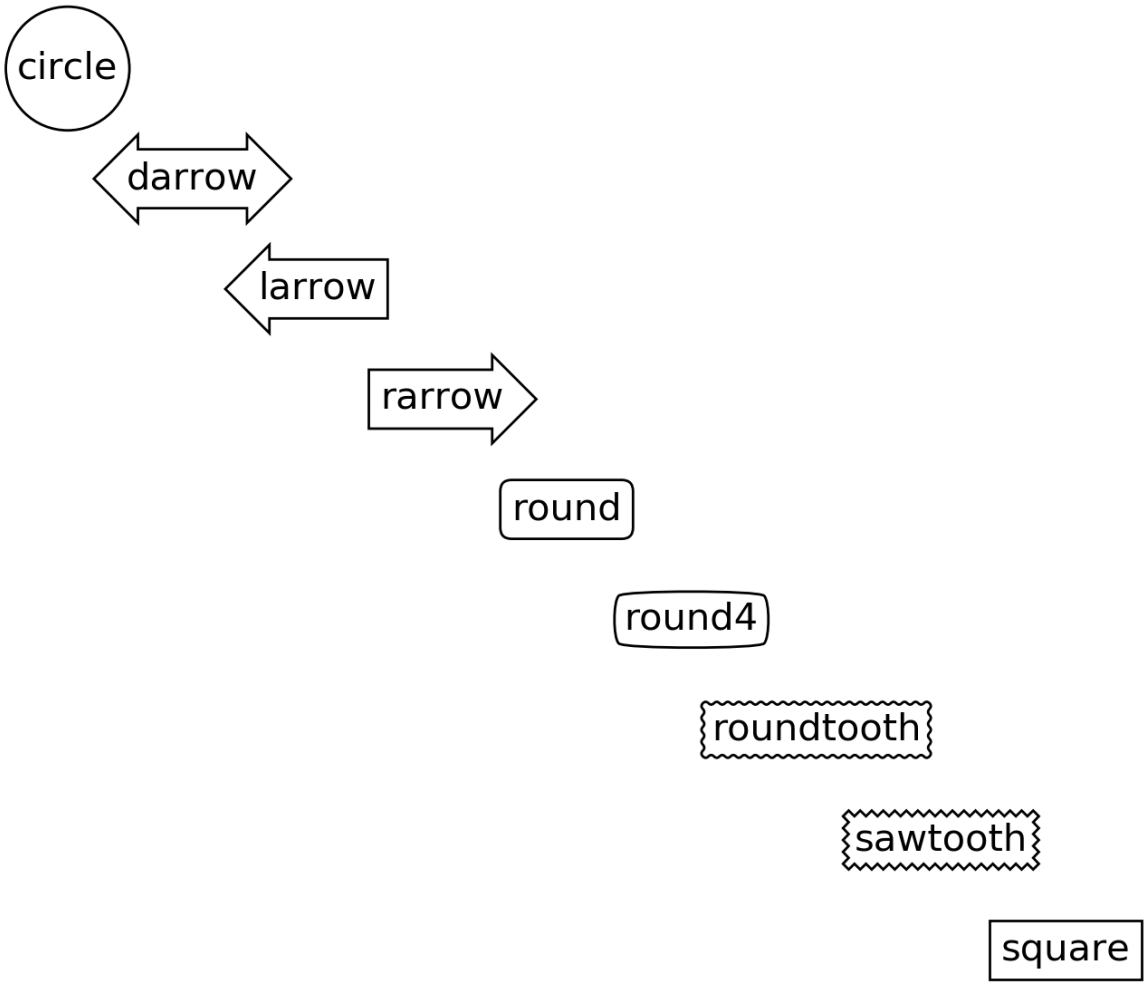


四、文本内容的样式

4.1.1 文本框样式

```
import matplotlib.patches as patches
import matplotlib.pyplot as plt
fig = plt.figure(1, figsize=(8,9), dpi=72)
fontsize = 0.5*fig.dpi
ax = fig.add_subplot(1,1,1,frameon=False,xticks=[],yticks=[])

boxStyles = patches.BoxStyle.get_styles()
boxStyleNames = list(boxStyles.keys())
boxStyleNames.sort()

for i,name in enumerate(boxStyleNames):
     ax.text(float(i+0.5)/len(boxStyleNames),
     (float(len(boxStyleNames))-0.5-i)/len(boxStyleNames),
     name,
     ha="center",
     size=fontsize,
     transform=ax.transAxes,
     bbox=dict(boxstyle=name,fc="w",ec="k"))
plt.show()
```

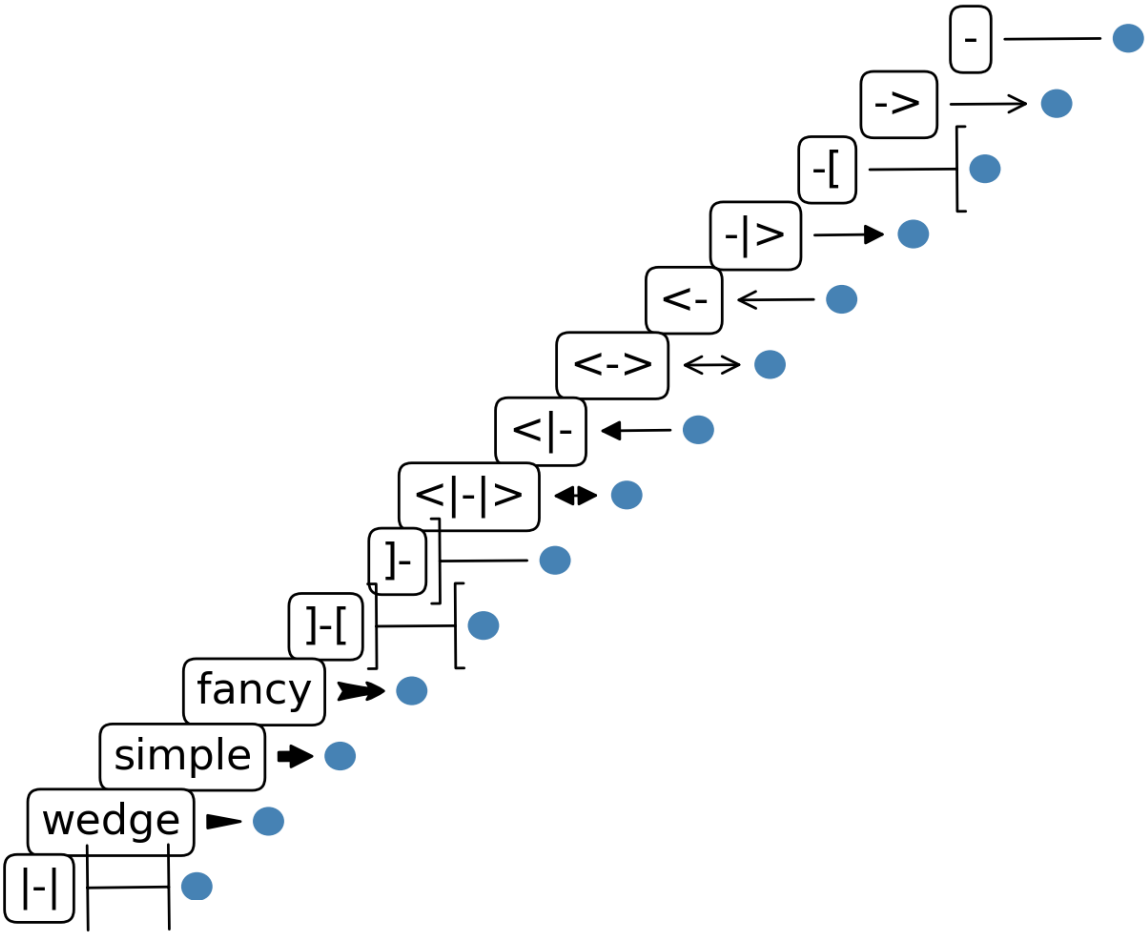
- 通过调用“patches.BoxStyle.get_styles()”语句，即调用类 BoxStyle 中的类方法 get_styles()，获得的返回值是可以使用的文本框样式的字典，其中键是文本框样式的名字。
- 参数 bbox 接收字典作为参数值，字典的键 boxstyle 对应的键值 name 就是文本框样式的名称。



4.1.2 文本注释箭头的样式

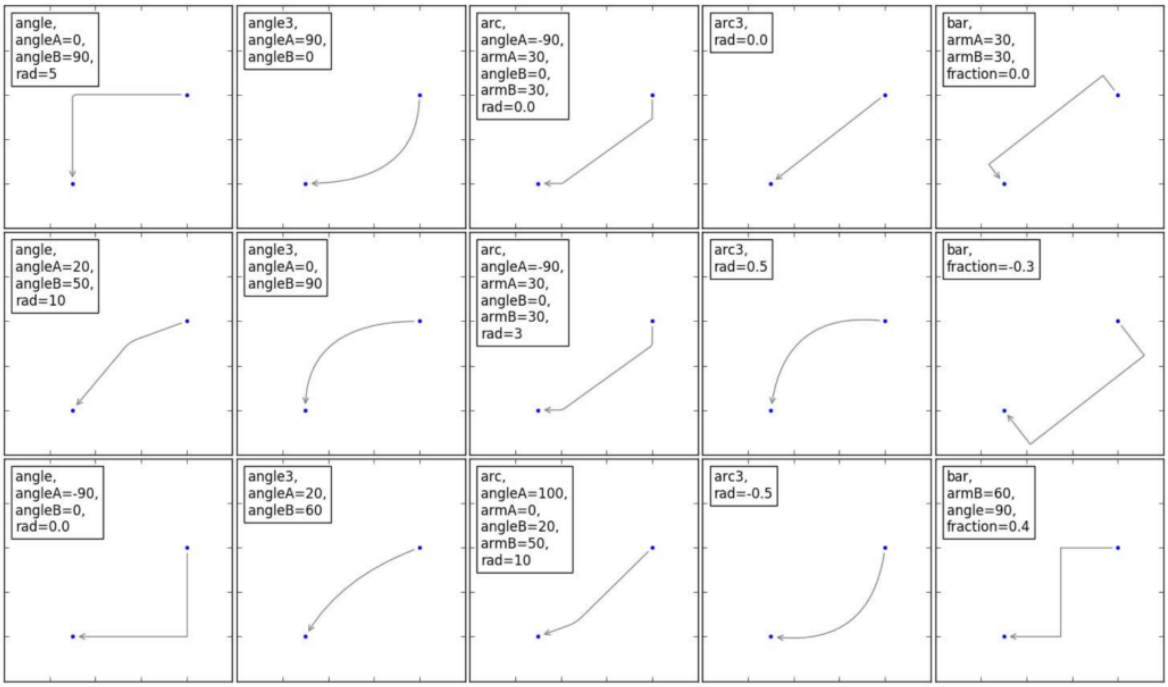
```
fig = plt.figure(1, figsize=(8,9))
fontsize = 0.4*40

ax = fig.add_subplot(1,1,1,frameon=False)
arrowStyles = patches.ArrowStyle.get_styles()
arrowStyleNames = list(arrowStyles.keys())
arrowStyleNames.sort()

ax.set_xlim(0,len(arrowStyleNames)+2.5)
ax.set_ylim(-2,len(arrowStyleNames))

for i,name in enumerate(arrowStyleNames):
    p = patches.Circle((float(len(arrowStyleNames))+1.2-i, float(len
                    (arrowStyleNames))-2.8-i),0.2,color="steelblue",alpha=1.0)
    ax.add_patch(p)
    ax.annotate(name,
            (float(len(arrowStyleNames))+1.2-i,float(len(arrowStyleNames))-2.8-
i),
            (float(len(arrowStyleNames))-1-i,float(len(arrowStyleNames))-3-i),
            xycoords="data",
            ha="center",
            size=fontsize,
            arrowprops=dict(arrowstyle=name,
                            facecolor="k",
                            edgecolor="k",
                            patchB=p,
                            shrinkA=5,
                            shrinkB=5,
                            connectionstyle="arc3"),
            bbox=dict(boxstyle="round",fc="w",ec="k"))

ax.xaxis.set_visible(False) # ax.set_xticks([])
ax.yaxis.set_visible(False) # ax.set_yticks([])
plt.show()
```

- 通过调用“patches.ArrowStyle.get_styles()”语句，即调用类 ArrowStyle 中的类方法 get_styles()，获得的返回值是可以应用的箭头样式的字典，其中键是箭头样式的名字。
- 调用实例方法 annotate()向坐标轴内迭代添加有指示的注解，参数 **arrowprops** 接收字典参数值
    - arrowstyle：是箭头样式的名称
    - facecolor：箭头填充颜色
    - edgecolor：轮廓颜色
    - shrinkA：控制箭头的起始端和注释内容的文本框的间隔距离
    - shrinkB：控制箭头的终止端和注释点补片 p 的轮廓线的间隔距离
    - patchB：指定注释点补片名称，
    - relpos：箭头起始点相对注释文本的位置，默认为 (0.5, 0.5)，即文本的中心。
    - connectionstyle：可以设置连接风格，“angle”、“angle3”、“arc”、“arc3”、“bar”，



文本注释箭头的连接风格

```
ax[0,1].annotate("",
            xy=(x1,y1),xycoords="data",
            xytext=(x2,y2),textcoords="data",
            arrowprops=dict(arrowstyle="->",
                            color="gray",
                            shrinkA=5,
                            shrinkB=5,
                            patchA=None,
                            patchB=None,
                            connectionstyle="angle3,angleA=90,angleB=0"))
```



4.2 text()——文本内容的布局

`text(x,y,s,ha="left",va="bottom",transform,**font_ style)`

- x,y：位置
- s：内容
- ha：“left","right","center"，文本内容水平对齐
- va：文本内容垂直对齐
  - "bottom"：文本内容的下边界放在补片 rect 的矩形底边框上
  - "center"：文本内容的中间位置放在矩形边框的中点上
  - baseline：以添加文本内容的坐标点的纵坐标作为水平线的对齐方式
- transform：
- wrap：为True时，控制文本自动换行
- rotation：旋转角度
- rotation_mode：旋转模式，
  - “None”或“default”：文本内容的旋转模式是首先旋转边界框，然后根据水平和垂直对齐方式进行对齐
  - “anchor”：文本内容的旋转模式就是首先将文本内容的边界框按照对齐方式进行对齐，然后对边界框根据指定的角度进行旋转
- multialignment：多行文本的对齐方式。“left","right","center"

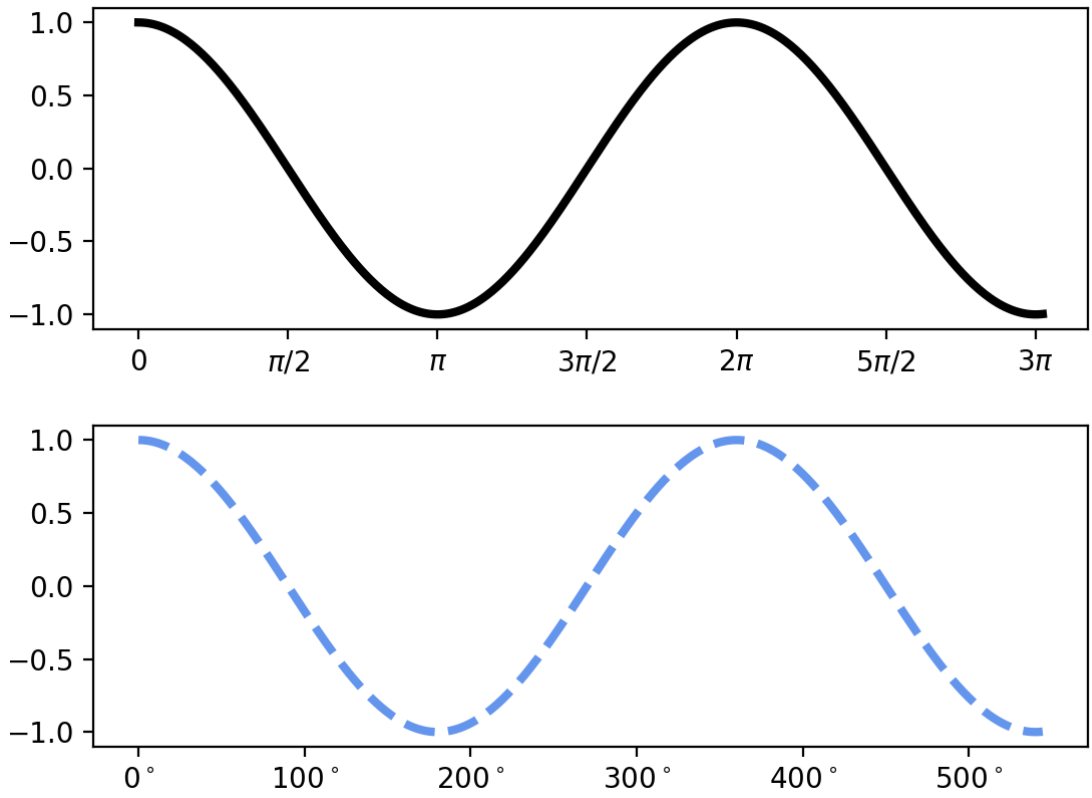
五、调整计量单位和计量方法

5.1.1 弧度和角度

```
from basic_units import radians,degrees,cos

x = np.linspace(0,9.5,500)
rad_x = [i*radians for i in x]
fig,ax = plt.subplots(2,1)
```

```
ax[0].plot(rad_x,cos(rad_x),ls="-",lw=3,color="k",xunits=radians)
ax[0].set_xlabel("")

ax[1].plot(rad_x,cos(rad_x),ls="--",lw=3,color="cornflowerblue",xunits=degrees)
ax[1].set_xlabel("")

fig.subplots_adjust(hspace=0.3)
plt.show()
```

- 弧度“xunits=radians”和角度“xunits=degrees”，对应的是yunits
- 调用实例方法 set_xlabel()隐藏 *x* 轴的轴标签。



5.1.2 厘米和英寸

`from basic_units import cm,inch`

xunits=cm, yunits=inch：改变参数xunits、 yunits的操作

5.1.3 秒、赫兹和分钟

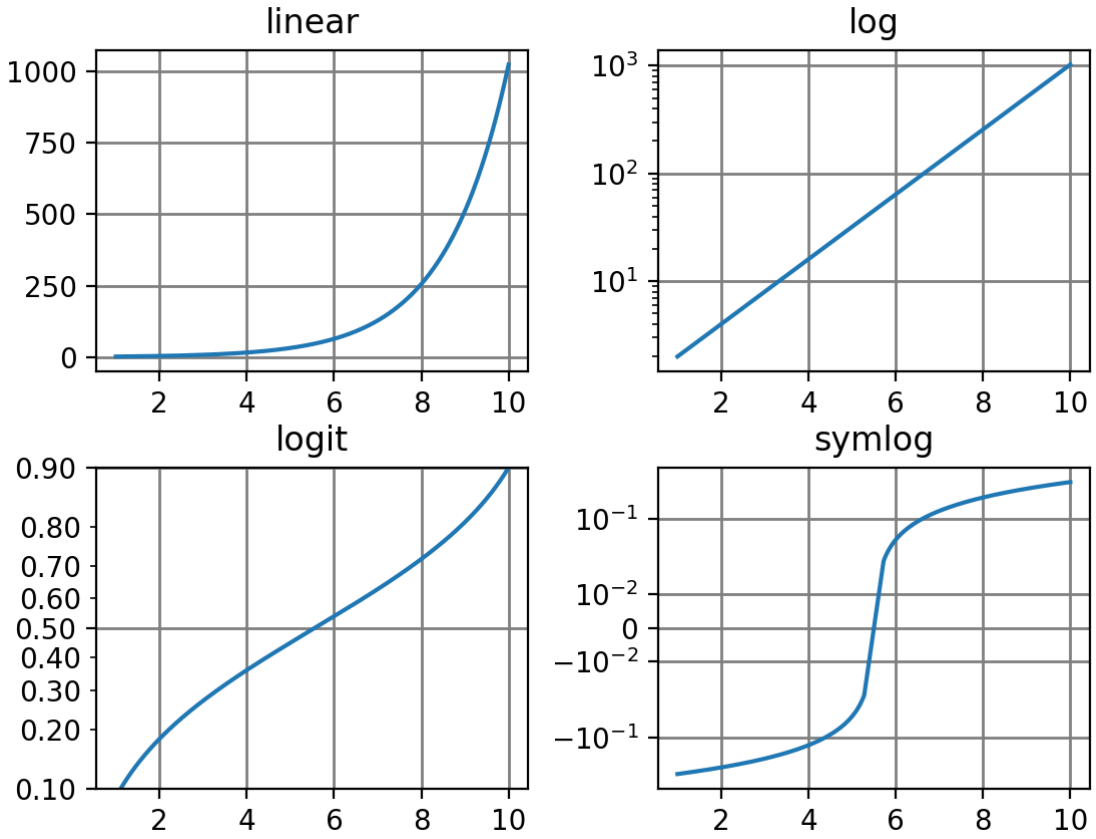
`from basic_units import secs,minutes,hertz`

5.2 不同计量方法

```
x = np.linspace(1,10,1000)
y1 = [2**j for j in x]
y2 = [0.09*j for j in x]
fig,ax = plt.subplots(2,2)

# linear
ax[0,0].plot(x,y1)
ax[0,0].set_yscale("linear")
```

```
ax[0,0].set_title("linear")
ax[0,0].grid(True,ls="-",lw=1,color="gray")

# log
ax[0,1].plot(x,y1)
ax[0,1].set_yscale("log")
ax[0,1].set_title("log")
ax[0,1].grid(True,ls="-",lw=1,color="gray")

# logit
ax[1,0].plot(x,y2)
ax[1,0].set_yscale("logit")
ax[1,0].set_title("logit")
ax[1,0].grid(True,ls="-",lw=1,color="gray")
ax[1,0].set_ylim(0.1,0.9)

# symlog
ax[1,1].plot(x,y2-np.average(y2))
ax[1,1].set_yscale("symlog",linthreshy=0.02)
ax[1,1].set_title("symlog")
ax[1,1].grid(True,ls="-",lw=1,color="gray")
fig.subplots_adjust(hspace=0.3,wspace=0.3)
plt.show()
```

- “ax[0,1].**set_yscale**("log")”语句，设置 *y* 轴的刻度采用对数计量方法
- logit 刻度与线性刻度之间的变换公式是$y_{\text{new}} = \log_{10}[y/(1-y)]$
- smylog 刻度的计量方法，使得曲线出现了对称的变化趋势，即 *y* 轴上的0<↵的刻度线之间的距离和0>↵的刻度线之间的距离完全相同



七、动画

7.1 animation模块

```
from matplotlib.animation import FuncAnimation

fig, ax = plt.subplots(1,1)
x = np.linspace(0,2*np.pi,5000)
y = np.exp(-x)*np.cos(2*np.pi*x)
line, = ax.plot(x,y,color="cornflowerblue",lw=3)
ax.set_ylim(-1.0,1.0)

# to clear current frame
def init():
    line.set_ydata([np.nan]*len(x))
    return line,

# to update the data
def animate(data):
     line.set_ydata(np.exp(-x)*np.cos(2*np.pi*x+float(data)/100))
     return line,

# to call class FuncAnimation which connects animate and init
ani = FuncAnimation(
                     fig,
                     animate,
                     init_func=init,
                     frames=200,
                     interval=2,
                     blit=True)
```

- 函数 init()的作用是在绘制下一帧动画画面之前清空画布窗口中的当前动画画面
- 函数 animate()的作用是绘制每帧动画画面
- 这两个函数的返回值“line”后面的符号“,”是不可以省略的，原因就是只有添加了符号“,”，才可以使得返回值是 Line2D 对象
- FuncAnimation 的构造函数主要接收的参数有 Figure 对象、函数 func、帧数 frames、帧与帧之间的间隔时间 interval。
    - `fig`：figure对象
    - `func`：不断更新图像的函数，生成新的xdata和ydata
    - `frames`：不断提供frame给updata用于生成新的xdata和ydata
    - `init_func=init`：初始化函数为init，自定义开始帧。
    - `interval=1`：时间间隔为1ms，interval的单位以ms计算。
    - `blit=True`：选择更新所有点，还是仅更新产生变化的点。应选择True，但mac用户请选择False，否则无法显示。

7.2 pyplot模块

```
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.patches import Circle
from warnings import filterwarnings

# ignore warning
filterwarnings("ignore",".*GUI is implemented.*")
```

```
# set several variables
word = "kaleidoscope"
row = int(len(word) / 4)
col = int(len(word) / 4)
num = int(len(word) / 4)
data = np.random.random((row, col, num))
colorMap = ["spring", "summer", "autumn", "winter"]
subplot_row = int(len(word) / 6)
subplot_col = int(len(word) / 6)
font = dict(family="monospace", weight="bold", style="italic", fontsize=10)
subplot_kw = dict(aspect="equal", frame_on=False, xticks=[], yticks=[])

# create subplots
fig, ax = plt.subplots(subplot_row, subplot_col, subplot_kw=subplot_kw)


# generate a subplot
def rowcolgenerator(r, c, season):
    index = colorMap.index(season)
    t = index * num
    subtitle = "No.{} '{}' Theme of the {}"
    for j in range(len(data)):
        ax[r, c].cla()
    collection = ax[r, c].pcolor(data[j, :], cmap=colorMap[index])
    patch = Circle((1.5, 1.5), radius=1.5, transform=ax[r, c].transData)
    collection.set_clip_path(patch)
    element = colorMap[index].capitalize()
    ax[r, c].set_title(subtitle.format((j + 1), word[t:t + 3], element),
                       **font)
    ax[r, c].set_axis_off()
    plt.pause(0.15)

# create animation
def animation():
    i = 0
    for r in range(subplot_row):
        for c in range(subplot_col):
            rowcolgenerator(r, c, colorMap[i])
    i += 1

    title = "Life Kaleidoscope Consists of Four Seasons"
    plt.suptitle(title, family="serif", weight="black", fontsize=20)

    plt.subplots_adjust(wspace=0.05, hspace=0.2)
    plt.show()

if __name__ == "__main__":
 animation()
```

- 调用模块 pyplot 中的函数 pause()，设置在执行下一句代码之前的延迟时间，
- 函数 pause()通常用来绘制简单的动画内容，模块 animation 通常用来绘制更加复杂的动画内容。当然，这里讲的简单与复杂是相对而言的。

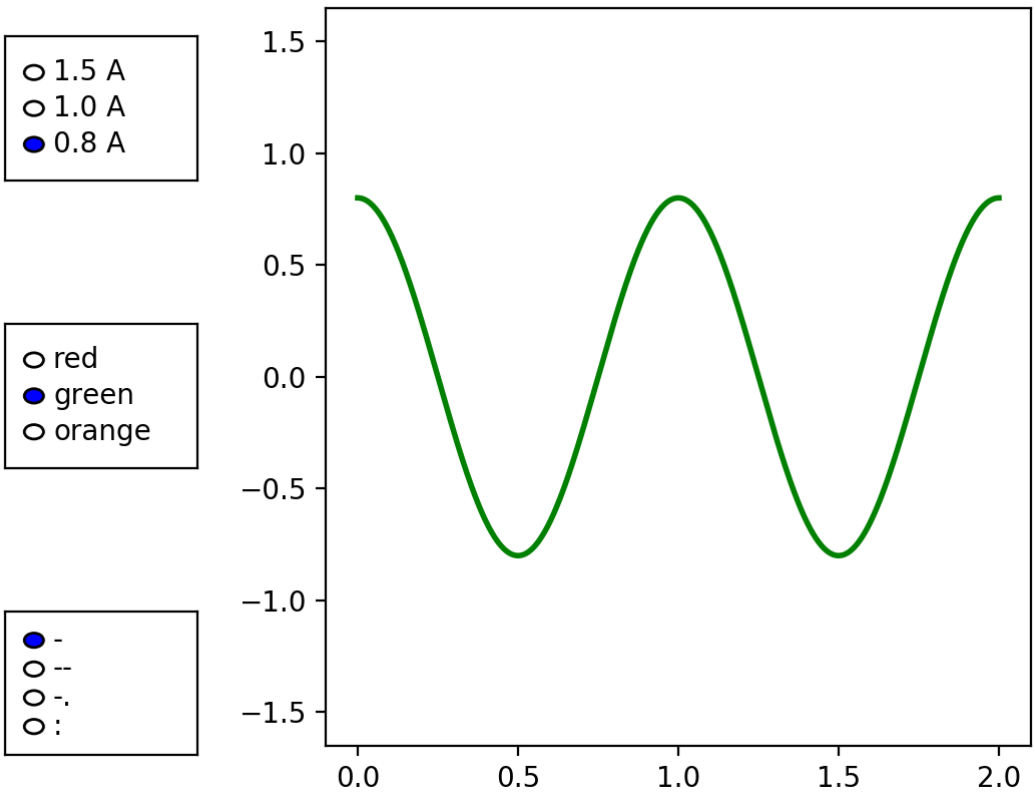
八、GUI效果widgets模块

8.1 RadioButtons

可以在画布中添加具备选择功能的按钮，通过按下按钮的操作过程，最终实现绘制内容的改变。

```
from matplotlib.widgets import RadioButtons

x = np.linspace(0.0,2.0,1000)
y1 = 1.5*np.cos(2*np.pi*x)
y2 = 1.0*np.cos(2*np.pi*x)
y3 = 0.8*np.cos(2*np.pi*x)

fig,ax = plt.subplots(1,1)
line, = ax.plot(x,y1,color="red",lw=2)
plt.subplots_adjust(left=0.35)
axesbgcolor = "cornflowerblue"

# a set of radionbuttons about amplitude
ax1 = plt.axes([0.1,0.7,0.15,0.15])
radio1 = RadioButtons(ax1,("1.5 A","1.0 A","0.8 A"))

def amplitudefunc(label):
     hzdict = {"1.5 A":y1,"1.0 A":y2,"0.8 A":y3}
     ydata = hzdict[label]
     line.set_ydata(ydata)
     plt.draw()
radio1.on_clicked(amplitudefunc)

# a set of radiobuttons about color
ax2 = plt.axes([0.1,0.4,0.15,0.15])
radio2 = RadioButtons(ax2,("red","green","orange"))

def colorfunc(label):
     line.set_color(label)
     plt.draw()
radio2.on_clicked(colorfunc)

# a set of radionbuttons about linestyle
ax3 = plt.axes([0.1,0.1,0.15,0.15])
radio3 = RadioButtons(ax3,("-","--","-.",":"))

def linestylefunc(label):
     line.set_linestyle(label)
     plt.draw()
radio3.on_clicked(linestylefunc)
plt.show()
```

- 类 RadioButtons 的构造函数中传递坐标轴实例 ax1 和按钮的标签内容，目的是向坐标轴中添加指定振幅大小的收音机按钮。
- 振幅函数 amplitudefunc()，在该函数中，调用函数 draw()更新单击了相应按钮后的画布内容。函数 draw()一般使用在交互模式下的画布内容的更新操作的过程里。
- 调用实例方法 on_clicked()，在振幅按钮被单击时，就会将振幅按钮的文本标签内容作为参数值传入函数 amplitudefunc()中，最终实现振幅函数 amplitudefunc()的调用目标



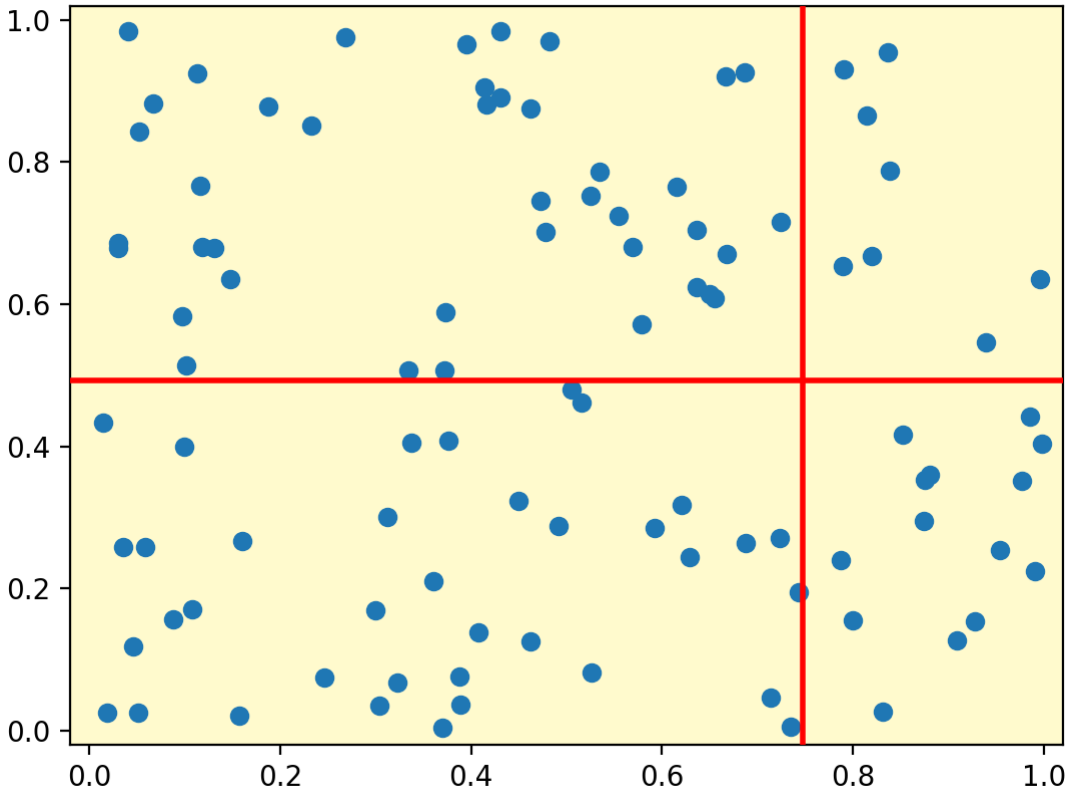
8.2 Cursor

向图形中添加一组横纵交叉的直线，从而实现图形界面中任何位置的数值定位的可视化效果

这种横纵交叉线又很像数值放大镜，可以清楚地显示任何位置的坐标数值。

```
from matplotlib.widgets import Cursor

lineprops=dict(color="red",lw=2)
fig,ax = plt.subplots(1,1)
x = np.random.random(100)
y = np.random.random(100)

ax.scatter(x,y,marker="o")
ax.set_xlim(-0.02,1.02)
ax.set_ylim(-0.02,1.02)

cursor = Cursor(ax,useblit=True,**lineprops)

plt.show()
```

- Cursor(ax,useblit=True,**lineprops)”语句，实现横纵交叉线的展示需求。同时，使用参数 lineprops 设置横纵交叉线的线条颜色和线条宽度等属性特征



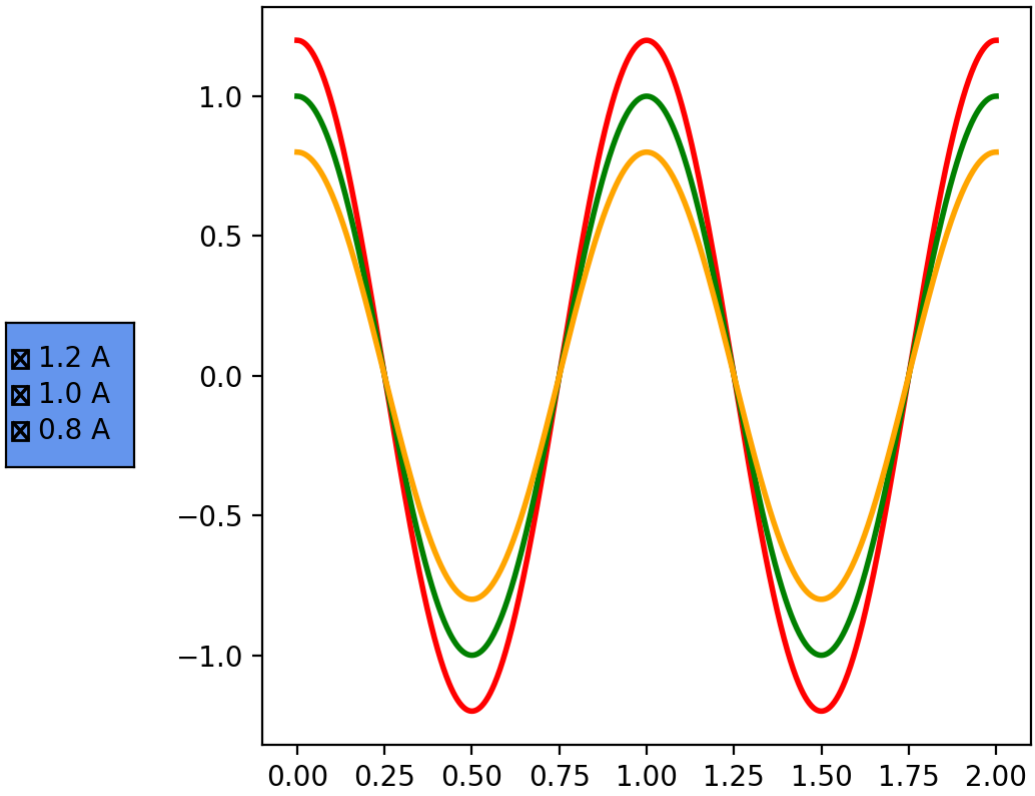
8.3 CheckButtons

可以在画布中添加点选按钮来实现网页上的点选框的 GUI 效果

```
from matplotlib.widgets import CheckButtons

x = np.linspace(0.0,2.0,1000)
y1 = 1.2*np.cos(2*np.pi*x)
y2 = 1.0*np.cos(2*np.pi*x)
y3 = 0.8*np.cos(2*np.pi*x)

fig,ax = plt.subplots(1,1)
line1, = ax.plot(x,y1,color="red",lw=2,visible=False,label="1.2 A")
line2, = ax.plot(x,y2,color="green",lw=2,label="1.0 A")
line3, = ax.plot(x,y3,color="orange",lw=2,label="0.8 A")
plt.subplots_adjust(left=0.30)
axesbgcolor = "cornflowerblue"

cax = plt.axes([0.1,0.4,0.1,0.15],facecolor=axesbgcolor)

lines = [line1,line2,line3]
labels = [str(line.get_label()) for line in lines]
visibility = [line.get_visible() for line in lines]
check = CheckButtons(cax,labels,visibility)

def func(label):
     index = labels.index(label)
     lines[index].set_visible(not lines[index].get_visible())
     plt.draw()
check.on_clicked(func)
plt.show()
```

- 向画布中添加一个坐标轴，得到坐标轴实例 cax

- 通过推导列表分别获得曲线的标签列表 labels 和可见情况列表 visibility。
- 调用类 CheckButtons 获得实例 check，在这个过程中，会将坐标轴实例 cax、标签列表 labels和可见情况列表 visibility 作为参数值传入类 CheckButtons 的构造函数中
- 个函数 func()，其一是将点选按钮与曲线的可见情况进行关联，这个功能是通过实例方法 set_visible()来完成的；其二是通过调用函数 draw()将点选后的画布内容进行更新，以显示出点选后的绘图内容。
- 调用实例方法 on_clicked()将点选动作和曲线显示联系起来



九、事件处理

9.1 关闭画布后出现事件结果提示

```
from __future__ import print_function
import matplotlib.pyplot as plt

def handle_event(close):
    print("Handling Event: Closed Figure!")

font_style = dict(family="serif",weight="black",size=50)
fig = plt.figure()
fig.canvas.mpl_connect("close_event", handle_event)
plt.text(0.15, 0.5,"close_event",**font_style)
plt.show()
```

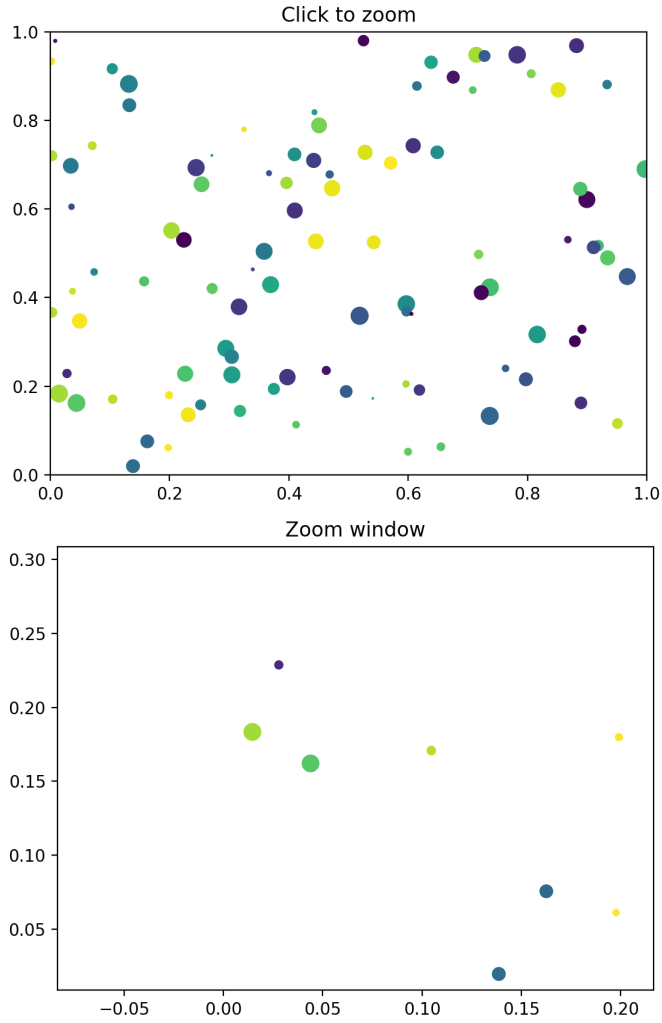
- `fig.canvas.mpl_connect("close_event", handle_event)`：将关闭画布事件和事件处理效果进行关联，从而使得关闭动作可以被有效地追踪和展示。
- 事件处理效果是在函数 handle_event()中进行设置的，也就是以文本形式展示事件处理效果。

9.2 局部放大效果

```
fig1,ax1 = plt.subplots(1,1)
fig2,ax2 = plt.subplots(1,1)

ax1.set_xlim(0,1)
ax1.set_ylim(0,1)
ax1.set_autoscale_on(False)
ax1.set_title("Click to zoom")

ax2.set_xlim(0.0,0.4)
ax2.set_ylim(0.0,0.4)
ax2.set_autoscale_on(False)
ax2.set_title("Zoom window")

x = np.random.rand(100)
y = np.random.rand(100)
s = np.random.rand(100)*100
c = np.random.rand(100)

ax1.scatter(x,y,s,c)
ax2.scatter(x,y,s,c)

def clicktozoom(event):
    if event.button != 1:
        return
    x,y = event.xdata,event.ydata
    ax2.set_xlim(x-0.15,x+0.15)
    ax2.set_ylim(y - 0.15, y + 0.15)
    fig2.canvas.draw()

fig1.canvas.mpl_connect("button_press_event", clicktozoom)
plt.show()
```

- `fig1.canvas.mpl_connect("button_press_event", clicktozoom)`，即将事件处理名称与事件处理效果进行有效关联，从而完成特定事件处理模式的执行任务。
- 函数 clicktozoom()的作用就是设置事件处理效果：

  如果在原始画布上单击散点图中的某一点或某一处，那么此处的数据点的横、纵坐标的数值就会被赋给变量 x 和 y，从而以 x 和 y 为圆心、以 0.15 为半径重新设置画布 2 中的版幅，进而实现放大画面的事件处理效果。



十、导入图像

10.1 外部图像的多样化展示

```
from matplotlib.cbook import get_sample_data
from matplotlib.patches import Circle

with get_sample_data("d://sunflower.png",asfileobj=True) as imageFile:
    imageArray = plt.imread(imageFile)

fig,ax = plt.subplots(1,1)
ai = ax.imshow(imageArray)
patch = Circle((605,360),radius=350,transform=ax.transData)
ai.set_clip_path(patch)

ax.set_axis_off()
plt.show()
```

- 借助关键字 with 和函数 get_sample_data()，同时使用关键字 as，将图片文件以 Python 文件对象形式存储在变量 imageFile 中。
- 使用函数 imread()，将文件对象 imageFile 转化成数组 array，进而将数组存储在变量 imageArray 中，
- 调用坐标轴实例 ax 的实例方法 imshow()，将以数组形式存储的图像 imageArray 加载到坐标轴上

10.3 热力图

```
golfer_stats = ["GIR", "Scrambling", "Bounce Back",
                "Ball Striking", "Sand Saves", "Birdie Conversion"]
golfer_names = ["Golfer %d" % i for i in range(1, 7)]

# we use the normalized percentages.
golfer_percentages = np.random.randn(6, 6)
shape = golfer_percentages.shape

fig, ax = plt.subplots()
im = ax.imshow(golfer_percentages, cmap="Greens")
colorbar = fig.colorbar(im, ax=ax)
colorbar.set_label("Golfer normalized percentages", rotation=-90, va="bottom")

ax.set_xticks(np.arange(0, shape[1], 1))
ax.set_yticks(np.arange(0, shape[0], 1))
ax.set_xticklabels(golfer_names)
ax.set_yticklabels(golfer_stats)

# add text to each area of heatmap
for i in range(len(golfer_stats)):
    for j in range(len(golfer_names)):
        text = ax.text(i, j, round(golfer_percentages[j, i], 1),
                       ha="center",
                       va="center",
                       color="w")

# turn tick line off and set tick label position
ax.tick_params(direction="out",
               bottom=False,
               right=False,
               labeltop=True,
               labelbottom=False)

# set tick label format
plt.setp(ax.get_xticklabels(),
         rotation=-30,
         ha="right",
         rotation_mode="anchor")

# turn spines off
spinesTupleList = list(ax.spines.items())
for i,each in enumerate(spinesTupleList):
     spine = each[1]
     spine.set_visible(False)

# set minor tick line position and create white grid
ax.set_xticks(np.arange(0.5,shape[1]-1,1),minor=True)
ax.set_yticks(np.arange(0.5,shape[0]-1,1),minor=True)
ax.grid(which="minor",color="w",linestyle="-",linewidth=3)

# turn minor tick line off
ax.tick_params(which="minor",top=False,bottom=False,left=False,right=False)
fig.tight_layout()
plt.show()
```

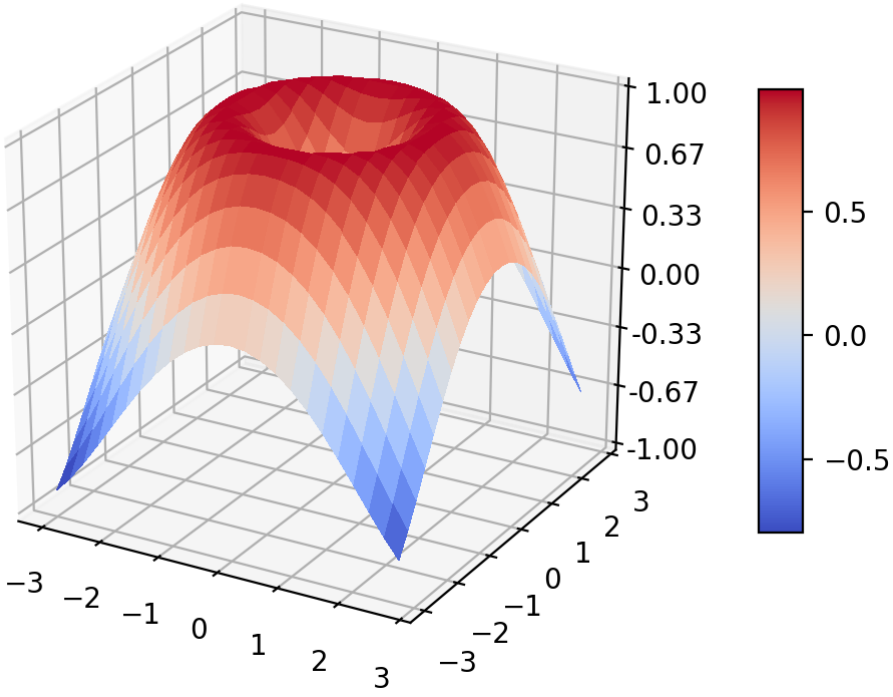
- im = ax.imshow(golfer_percentages,cmap="Greens")”语句。调用 Axes 的实例方法 imshow()，将二维数组 golfer_percentages 作为参数值传入实例方法 imshow()里，参数 cmap的参数值是“Greens”，这是一个不同饱和度的绿色的颜色映射表
- 调用“fig.colorbar(im,ax=ax)”语句，将二维数组的元素和颜色进行映射，这种映射关系就用颜色标尺来显示。
- ax.set_xticks(np.arange(0,shape[1],1))”和“ax.set_xticklabels(golfer_names)”语句，分别设置 *x* 轴的刻度线的位置和刻度线上的刻度标签的文本内容。

十一、绘制3D图形

11.1 绘制带颜色标尺的彩色曲面

```
from matplotlib import cm
from matplotlib.ticker import LinearLocator,FormatStrFormatter
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure()
ax = fig.add_subplot(1,1,1,projection="3d")

x = np.arange(-3,3,0.25)
y = np.arange(-3,3,0.25)
x,y = np.meshgrid(x,y)
r = np.sqrt(np.power(x,2)+np.power(y,2))
z = np.sin(r)

# plot 3d surface
surf = ax.plot_surface(x,y,z,
                       rstride=1,
                       cstride=1,
                       cmap=cm.coolwarm,
                       linewidth=0,
                       antialiased=False)

# customize the z axis
ax.set(zlim=(-1,1))
ax.zaxis.set_major_locator(LinearLocator(7))
ax.zaxis.set_major_formatter(FormatStrFormatter("%3.2f"))

# add a color bar mapping values to colors
fig.colorbar(surf,shrink=0.6,aspect=10)
plt.show()
```

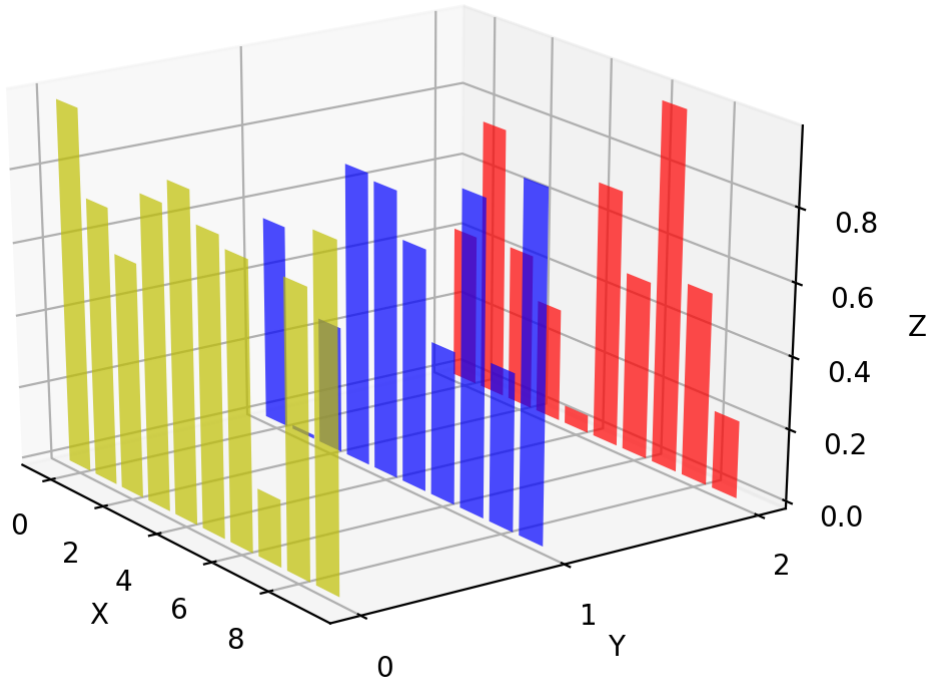
- 调用类 Axes3D 的实例方法 plot_surface()绘制曲面：通过参数 rstride 和 cstride 设置曲面上单位曲面的大小，参数 cmap 用于设置曲面补片的颜色映射表类型
- 为了使 *z* 轴的刻度线和刻度标签更加清晰和直观，使用一组代码对 *z* 轴的刻度线和刻度标签进行定制化设置，主要调整刻度线的数量和刻度标签的小数点位数。



10.2 在 3D 空间里分层展示投射到指定平面后的 2D 柱状图

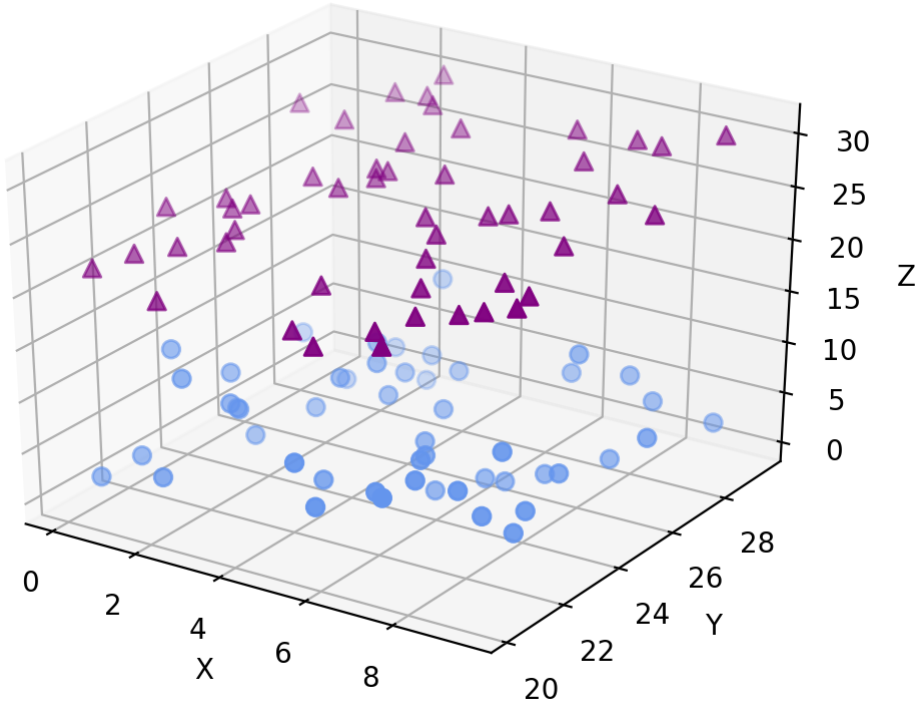
```
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure()
ax = fig.add_subplot(1,1,1,projection="3d")
colorsList = ["r","b","y"]
yLayersList = [2,1,0]

for color,layer in zip(colorsList,yLayersList):
    x = np.arange(10)
    y = np.random.rand(10)
    ax.bar(x, y, zs=layer, zdir="y", color=color, alpha=.7)

ax.set(xlabel="X", ylabel="Y", zlabel="Z", yticks=yLayersList)
plt.show()
```

- ax.bar(x,y,zs=layer,zdir="y",color=color,alpha=.7)的作用，也就是类 Axes3D 的实例方法 bar()的使用方法。
    - 参数 x 表示柱体左边位置的列表；
    - 参数 y 表示柱体高度的列表；
    - 参数 zs 是将柱状图进行投射的层次序号；
    - 参数 zdir 是将 *z* 轴用来表示柱体的高度，即将 *y* 轴设定成 *z* 轴；
    - 参数 color 用于设置柱体的颜色；
    - 参数alpha 用于设置柱体的透明度



11.3 在 3D 空间里绘制散点图

```
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure()
ax = fig.gca(projection="3d")

xs = np.random.rand(50)*10
ys = np.random.rand(50)*10+20
zs1 = np.random.rand(50)*10
zs2 = np.sqrt(xs**2+ys**2)

ax.scatter(xs,ys,zs=zs1,zdir="z",c="cornflowerblue",marker="o",s=40)
ax.scatter(xs,ys,zs=zs2,zdir="z",c="purple",marker="^",s=40)
ax.set(xlabel="X",ylabel="Y",zlabel="Z")
plt.show()
```



十五、 数学表达式

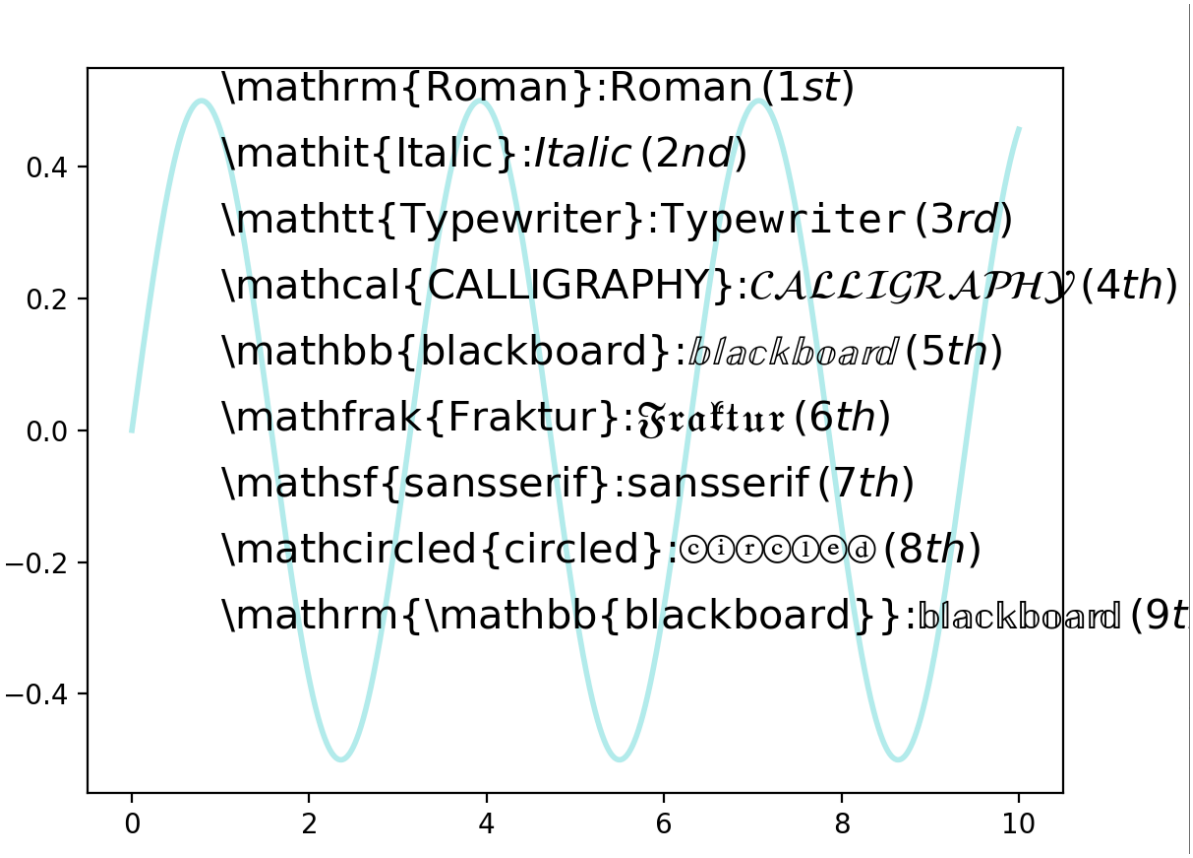
15.1 编辑字符串

`r"$Regular text\mathtext$"`

- 字母 r 表示原始字符串（Raw Strings）
- 文本字符串可以由普通文本 Regular text 和数学文本\mathtext 所组成，
- 其中数学文本\mathtext 用“\”开始的字符串“mathtext”来表示一个命令，命令中需要输出的内容放在一对花括号“{}”中

15.2 字体效果

```
x = np.linspace(0,10,10000)
y = np.sin(x)*np.cos(x)
plt.plot(x,y,ls="-",lw=2,color="c",alpha=0.3)

plt.text(1,0.5,r"\mathrm{Roman}:$\mathrm{Roman}\/(1st)$",fontsize=15)
plt.text(1,0.4,r"\mathit{Italic}:$\mathit{Italic}\/(2nd)$",fontsize=15)
plt.text(1,0.3,r"\mathtt{Typewriter}:$\mathtt{Typewriter}\/(3rd)$",fontsize=15)
plt.text(1,0.2,r"\mathcal{CALLIGRAPHY}:$\mathcal{CALLIGRAPHY}\/(4th)$",fontsize=
15)
plt.text(1,0.1,r"\mathbb{blackboard}:$\mathbb{blackboard}\/(5th)$",fontsize=15)
plt.text(1,0.0,r"\mathfrak{Fraktur}:$\mathfrak{Fraktur}\/(6th)$",fontsize=15)
plt.text(1,-0.1,r"\mathsf{sansserif}:$\mathsf{sansserif}\/(7th)$",fontsize=15)
plt.text(1,-0.2,r"\mathcircled{circled}:$\mathcircled{circled}\/(8th)$",fontsize
=15)
plt.text(1,-0.3,r"\mathrm{\mathbb{blackboard}}:$\mathrm{\mathbb{blackboard}}\/(9
th)$",fontsize=15)
plt.show()
```



15.3 数学公式

1. 基础数学

∞ \infty	≠ \neq	× \times	÷ \div	~ \sim
± \pm	∓ \mp	⩽ \leq	⩾ \geq	≈ \approx
% \%	∗ \ast	· \cdot	≡ \equiv	≅ \cong

∈ \in	∋ \ni	∃ \exists	∄ \nexists	∝ \propto
∪ \cup	∩ \cap	⊥ \perp	∥ \parallel	∘ \circ

2. 希腊字母

α \alpha	β \beta	γ \gamma	δ \delta	ϕ \phi
ϵ \epsilon	ζ \zeta	η \eta	θ \theta	ι \iota
κ \kappa	λ \lambda	μ \mu	ν \nu	ξ \xi
ρ \rho	σ \sigma	τ \tau	υ \upsilon	φ \varphi
χ \chi	ω \omega	π \pi	ψ \psi	Ξ \Xi
Γ \Gamma	Δ \Delta	Λ \Lambda	Ω \Omega	Φ \Phi
Ψ \Psi	Π \Pi	Σ \Sigma	Θ \Theta	Υ \Upsilon

3. 字母类符号

ℵ \aleph	ℶ \beth	ℸ \daleth	ℷ \gimel	∂ \partial
ⅎ \Finv	ℜ \Re	ℓ \ell	Å \AA	ð \eth
ℑ \Im	ℏ \hslash	∁ \complement	℘ \wp	∀ \forall

4. 运算符

∫ \int	∬ \iint	∭ \iiint	⨌ \oiiint	∑ \sum
∏ \prod	∐ \coprod	⋀ \bigwedge	⋁ \bigvee	⋂ \bigcap
⋃ \bigcup	⨀ \bigodot	⨁ \bigoplus	⨂ \bigotimes	⨄ \biguplus
\ \backslash	/ \slash	⋇ \divideontimes	⋆ \star	≀ \wr
△ \vartriangle	‡ \ddag	⋄ \diamond	† \dag	∧ \wedge
∨ \vee	⊗ \odot	⊙ \otimes	⊕ \oplus	⊖ \ominus
⋒ \Cup	⋓ \Cap	∔ \dotplus	⊺ \intercal	∸ \dotminus
∵ \because	∴ \therefore	∽ \backsim	⊎ \uplus	≃ \simeq

5. 箭头

← \leftarrow	→ \to	→ \rightarrow	↑ \uparrow	↓ \downarrow
⟵ \longleftarrow	⟶ \longrightarrow	⇐ \Leftarrow	⇒ \Rightarrow	↦ \mapsto
↔ \leftrightarrow	↚ \nleftarrow	↛ \nrightarrow	⇁ \rightharpoonup	↼ \leftharpoonup
↮ \nleftrightarrow	⟺ \Leftrightarrow	⇎ \nLeftrightarrow	⇍ \nLeftarrow	⇏ \nRightarrow

6. 几何学

∠ \angle	∢ \sphericalangle	∤ \nmid	∦ \nparallel	■ \blacksquare

7. 分数

（1）分数基本 TeX 符号编写规则：\frac{}{}。其中，第 1 对花括号中放入分子，第 2 对花括号中放入分母。

（2）样例是$\frac{dy}{dx}$，Python 代码是 r"$\frac{\mathrm{dy}}{\mathrm{dx}}$"。

8. 上下标

（1）上下标基本 TeX 符号编写规则：anytext_{downtext}^{uptext}。其中，“anytext”是需要添加上下标的文本内容，第 1 对花括号中放入下标内容，第 2 对花括号中放入上标内容，上下标的先后顺序可以颠倒前后位置，上下标可以选择其一添加，如 anytext_{downtext}或 anytext^{uptext}。

（2）上标样例是x^2，Python 代码是 r"$\mathrm{x^{2}}$。

（3）下标样例是x_2，Python 代码是 r"$\mathrm{x_{2}}$"。

9. 根式

（1）根式基本 TeX 符号编写规则：\sqrt[]{}。其中，方括号中放入开算术方根的次数，花括号中放入开算术方根的数值或字符。

（2）根式样例是$\sqrt{2}$，Python 代码是 r"$\sqrt{2}$"。

（3）根式样例是$\sqrt[2]{2}$，Python 代码是 r"$\sqrt[2]{2}$"。

（4）根式样例是$\sqrt[3]{2}$，Python 代码是 r"$\sqrt[3]{2}$"。

（5）对于开算术方根的数值或字符出现复杂嵌套的情形，不建议使用\sqrt[]{}，而应该使用上标的模式 anytext^{}，表示开算术方根的次数。

10. 积分

（1）积分基本 TeX 符号编写规则：\int_{downtext}^{uptext}。其中，第 1 对花括号中放入积分下限，第 2 对花括号中放入积分上限。

（2）积分样例是$\int_2^6 x^2 dx$，Python 代码是 r"$\mathrm{\int_2^6 x^2 dx}$"。

11. 大型运算符

（1）大型运算符 TeX 符号编写规则（以求积运算符为例）：\prod_{downtext}^{uptext}。

（2）大型运算符样例是$\prod_{k=1}^{n} A_k$，Python 代码是 r"$\mathrm{\prod_{k=1}^{n} A_{k}}$"。

12. 括号

(	)	[	]	\{
\}	\vert	\Vert	/	

13. 函数

Pr \Pr	sin \sin	cos \cos	exp \exp	det \det
lim \lim	ln \ln	log \log	max \max	min \min
tan \tan	arg \arg	lg \lg		

14. 强调符号

ȧ \dot a	ä \ddot a	â \hat a	ã \tilde a	ā \bar a
$\vec{a}$ \vec a	$\overline{xyz}$ \overline{xyz}	$\widehat{xyz}$ \widehat{xyz}	$\widetilde{xyz}$ \widetilde{xyz}	

15. 矩阵

⋯ \cdots	… \ldots	⋮ \vdots	⋱ \ddots	

16. 其他

white space \/	$ \$	` \backprime	′ \prime	