02 Python 2D绘图

任务目标

1、掌握Matplotlib的绘图原理

2、通过Matplotlib绘制常见的统计图形

3、通过函数完成自定义图形的绘制

Matplotlib是基于Numpy的一套Python工具包，Matplotlib提供了丰富的数据绘图工具，可用于绘制常见的统计图形，也可以通过自定义函数绘制统计图形。

10.1 Python 绘图的原理

10.1.1 Python绘图原理

二维图形的绘制离不开点、线、面等基本的几何图形，Python绘制2D图形，需要安装Matplotlib绘图库，安装Matplotlib绘图库的命令如下。

脚本 10-1-1 安装 Matplotlib 绘图库

```
pip install matplotlib –i  https://pypi.tuna.tsinghua.edu.cn/simple
```

1、使用np.linspace分配向量

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

if __name__ == "__main__":
    x = np.linspace(0,10,100);
    y = np.sin(x);
    plt.plot(x,y);
    plt.show();
```

2、使用list列表分配向量

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

if __name__ == "__main__":
    x = [];
    y = [];
    s = 0;
    for i in range(100):
        s+=0.1;
        x.append(s);
        y.append(0);
    for i  in range(len(x)):
        y[i] = math.sin(x[i])
    plt.plot(x,y);
    plt.show();
```

3、使用arange分配向量

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

if __name__ == "__main__":
    x =np.arange(0,10,0.1);
    y = np.sin(x);
    plt.plot(x,y);
    plt.show();
```

Matplotlib.pyplot是绘图接口，主要用于交互式绘图和编程绘图，pyplot是类似MATLAB的绘图接口。Matplotlib的pyplot、path、patches是主要的绘图包，points是点的坐标，lines是路径的方法，Path.MOVETO表示点移动，Path.LINETO表示在两点间进行画线，Path.CLOSEPOLY表示路径闭合。将点points和线lines组合成路径path，然后将path转化为图形patch，其中patches.PathPatch(path,facecolor,linewidth)的三个参数分别是路径、颜色、线宽等。fig表示绘图的图形，ax表示图形中的数轴，将patch添加到数轴ax上，显示图形。

1. 自定义红色柱状图

程序 10-1-1 绘制红色柱状图

```
import matplotlib.pyplot as plt;
from matplotlib.path import Path;
import matplotlib.patches as patches;
points = [(0., 0.), (0., 5.), (1., 5.),(1., 0.), (0., 0.)];  # 五个坐标点
lines = [Path.MOVETO,Path.LINETO,Path.LINETO,Path.LINETO,Path.CLOSEPOLY];
#点之间的连接方式
path = Path(points, lines);            #点和线形成路径
fig = plt.figure();                    #获取图形
ax = fig.add_subplot(111);             #获取数轴
patch = patches.PathPatch(path, facecolor='red', lw=2); #获取图形补丁
ax.add_patch(patch);                                  #将图形补丁添加到数轴中
ax.spines['right'].set_color('none')                        #将图像右边的轴设为透明
ax.spines['top'].set_color('none')                         #将图像上面的轴设为透明
ax.xaxis.set_ticks_position('bottom')                 #将x轴刻度设在下面的坐标轴上
ax.yaxis.set_ticks_position('left')                  #将y轴刻度设在左边的坐标轴上
ax.spines['bottom'].set_position(('data', 0))    #将两个坐标轴的位置设在原点
ax.spines['left'].set_position(('data', 0))
ax.set_xlim(-2,6);                             #设定x轴范围
ax.set_ylim(-2,6);                             #设定y轴范围
plt.show();                                  #显示图形
```

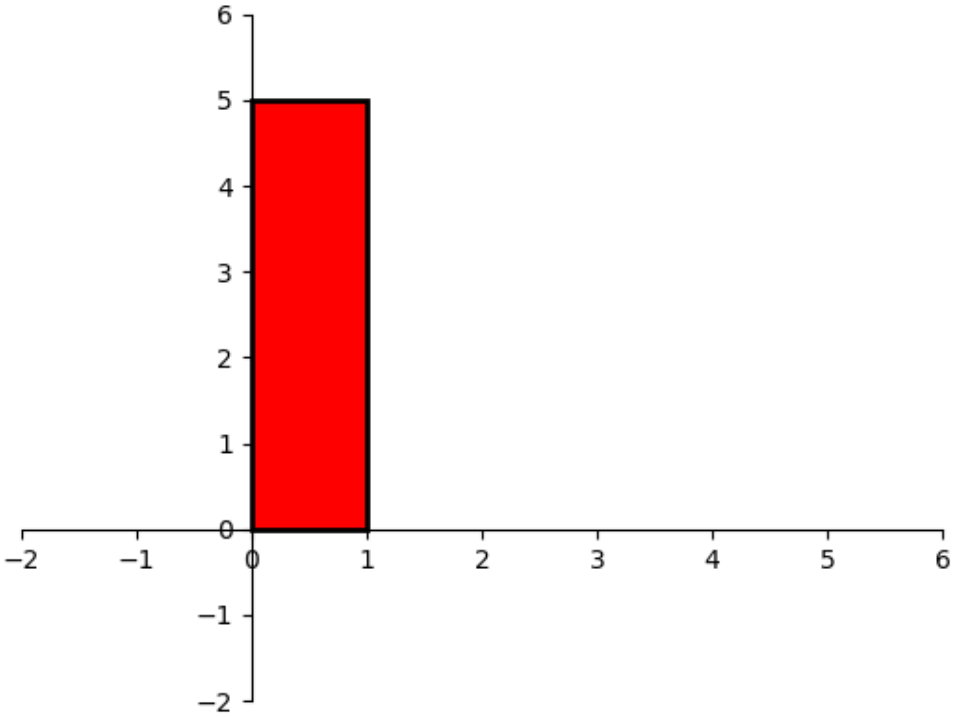

图10-1-1 绘制图形补丁

2. 复杂图形的绘制-基于面向对象设计方法

matplotlib.pyplot 是 matplotlib 的一个基于状态的接口。它提供了一种类似MATLAB的绘图方式。pyplot主要用于交互式绘图和简单的程序化绘图，对于更复杂的绘图，推荐使用面向对象的API。构建一个含有简单修改正弦波按钮的GUI，Next和Previous按钮可以动态更新不同频率的正弦波图形。

程序 10-1-2 绘制变频率的正弦波图形

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.widgets import Button
freqs = np.arange(2, 20, 3)
fig, ax = plt.subplots()
plt.subplots_adjust(bottom=0.2)
t = np.arange(0.0, 1.0, 0.001)
s = np.sin(2*np.pi*freqs[0]*t)
l, = plt.plot(t, s, lw=2)
class Index:
    ind = 0
    def next(self, event):
        self.ind += 1
        i = self.ind % len(freqs)
        ydata = np.sin(2*np.pi*freqs[i]*t)
        l.set_ydata(ydata)
        plt.draw()
    def prev(self, event):
        self.ind -= 1
        i = self.ind % len(freqs)
        ydata = np.sin(2*np.pi*freqs[i]*t)
        l.set_ydata(ydata)
```

```
        plt.draw()
callback = Index()
axprev = plt.axes([0.7, 0.05, 0.1, 0.075])
axnext = plt.axes([0.81, 0.05, 0.1, 0.075])
bnext = Button(axnext, 'Next')
bnext.on_clicked(callback.next)
bprev = Button(axprev, 'Previous')
bprev.on_clicked(callback.prev)
plt.show()
```

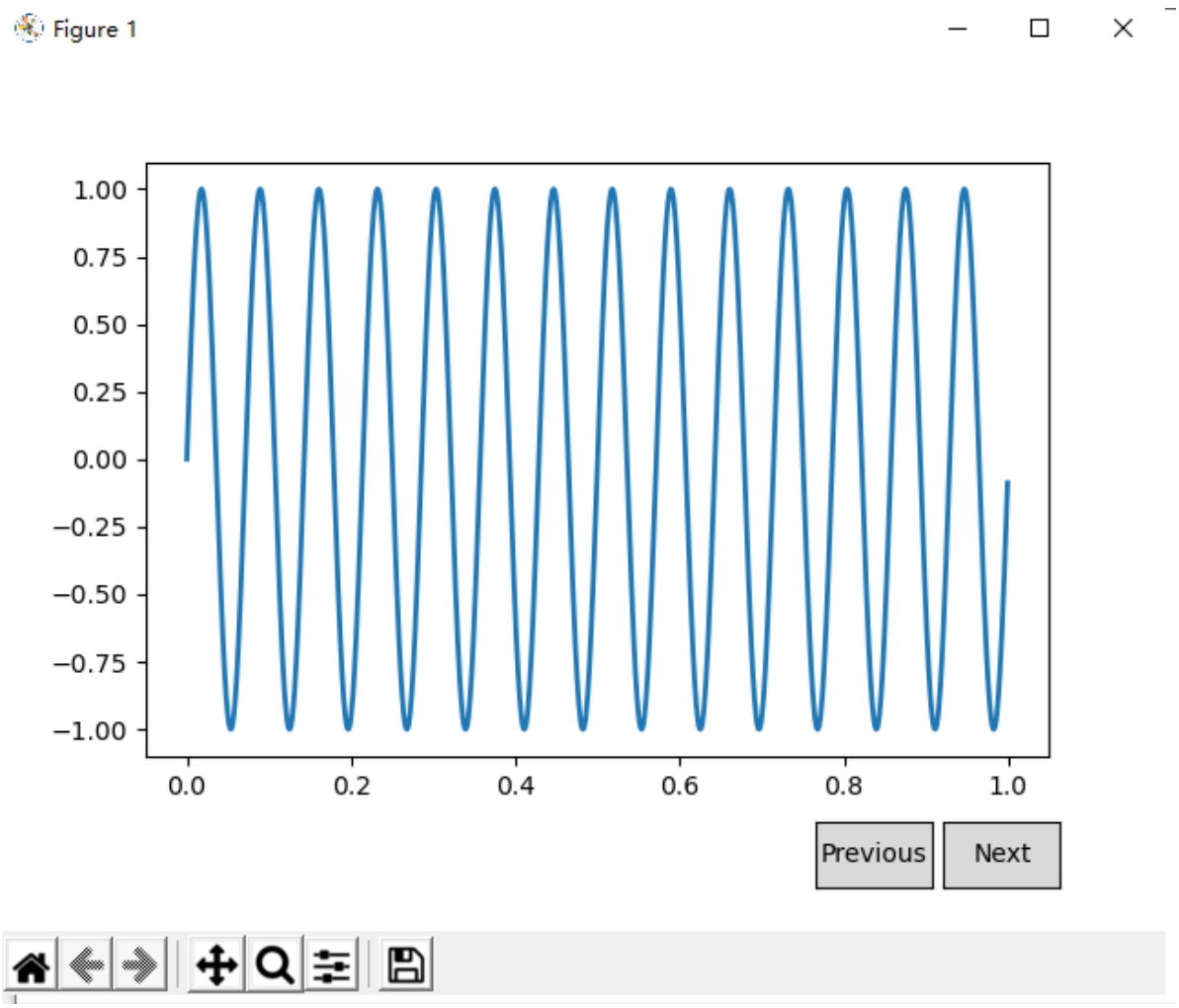

图 10-1-2 变频率的正弦波图形

10.1.2 Matplotlib函数

二维图形的绘制离不开点、线、面等基本的几何图形，Python绘制2D图形，需要安装Matplotlib绘图库，安装Matplotlib绘图库的命令如下。

3. 绘制折线图

matplotlib.pyplot.plot是折线图绘制函数，折线图绘制函数的原型如下：

```
matplotlib.pyplot.plot(*args,  scalex=True, scaley=True, data=None, **kwargs)
plot([x], y, [fmt], *,  data=None, **kwargs)
plot([x], y, [fmt], [x2], y2,  [fmt2], ..., **kwargs)
```

在plot函数中，x和y分别表示坐标系中点的位置，[fmt]用于定义基本的格式，如颜色、标记和线条样式。它是一种快捷的字符串符号，fmt包括颜色、标记和线三个部分组成。

```
fmt = '[marker][line][color]'
```

fmt每一项都是可选的，如果没有选择，fmt的值选择周期样式，如果给定线的样式，但是没有标记类型，则显示的数据只有线段，而没有标记。fmt也支持也支持其他组合，如[color][marker][line]，但要注意它们的解析可能有歧义。

Markers

character	description
`'.'`	point marker
`','`	pixel marker
`'o'`	circle marker
`'v'`	triangle_down marker
`'^'`	triangle_up marker
`'<'`	triangle_left marker
`'>'`	triangle_right marker
`'1'`	tri_down marker
`'2'`	tri_up marker
`'3'`	tri_left marker
`'4'`	tri_right marker
`'8'`	octagon marker
`'s'`	square marker
`'p'`	pentagon marker
`'P'`	plus (filled) marker
`'*'`	star marker
`'h'`	hexagon1 marker
`'H'`	hexagon2 marker
`'+'`	plus marker
`'x'`	x marker
`'X'`	x (filled) marker
`'D'`	diamond marker
`'d'`	thin_diamond marker
`'|'`	vline marker
`'_'`	hline marker

线的样式

character	description
`'-'`	solid line style
`'--'`	dashed line style
`'-.'`	dash-dot line style
`':'`	dotted line style

颜色的类型

character	color
`'b'`	blue
`'g'`	green
`'r'`	red
`'c'`	cyan
`'m'`	magenta
`'y'`	yellow
`'k'`	black
`'w'`	white

标记和颜色组合样式

character	描述
`'b'`	蓝色标记，默认形状
`'or'`	红色圆圈
`'-g'`	绿色实线
`'--'`	默认颜色虚线。
`'^k:'`	黑色虚线，上三角标记

如果颜色是格式字符串中的唯一部分，可以使用任何matplotlib.colors规范中的颜色，例如('green')或十六进制字符串('#008000')，如果还包含标记类型和线的样式，则不能使用颜色的全名或十六进制字符串。

程序 10-1-3 绘制正弦、余弦曲线。

```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(-np.pi,np.pi,256,endpoint=True)
y=np.cos(x)
z=np.sin(x)
fig = plt.figure(figsize=(5,5))
plt.plot(x,y)
plt.plot(x,z)
plt.show()
```

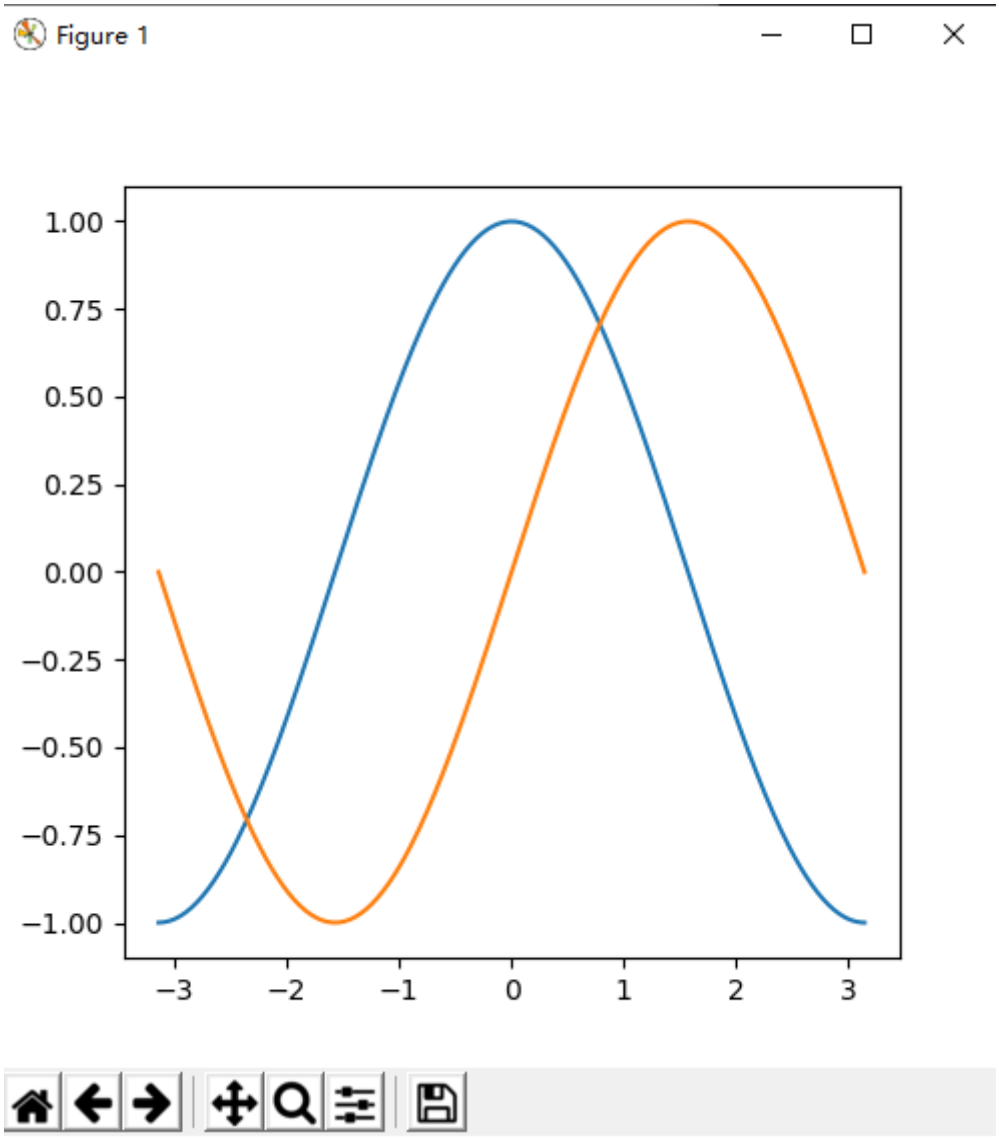

图 10-1-3 双色正弦、余弦曲线

4. 为折线图添加$Latex$标题

在图形中添加Latex数学符号，需要使用包含数学符号，如\sin(x)$。下图绘制了sin(x)和cos(x)两条曲线。如果折线颜色没有设定，则matplotlib采用默认属性周期的索引rcParams["axes. prop_cycle"]设定线段的颜色，(rcParams["axes.prop_cycle"] (default: cycler('color', ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b', '#e377c2', '#7f7f7f', '#bcbd22', '#17becf']))。其中第一种颜色是'#1f77b4'，第二种颜色是'#ff7f0e'。以下的颜色依次类推。

matplotlib.pyplot.title(label, fontdict=None, loc=None, pad=None, *, y=None, **kwargs)为坐标轴设置一个标题。坐标轴标题可以放置在以下三个位置，图形上方的中心位置、与图形左边缘齐平、与图形右边缘齐平。

matplotlib.pyplot.xlim(*args, **kwargs) 获取或设置当前轴的x-limits。
matplotlib.pyplot.ylim(*args, **kwargs) 获取或设置当前轴的y-limits。
matplotlib.pyplot.xticks(ticks=None, labels=None, **kwargs) 获取或设置当前X轴的刻度线位置和标签。
matplotlib.pyplot.yticks(ticks=None, labels=None, **kwargs) 获取或设置当前Y轴的刻度线位置和标签。

程序 10-1-3 绘制带数学符号的正弦、余弦曲线

```
import matplotlib.pyplot as plt
import numpy as np;
x = np.linspace(-np.pi,np.pi,256,endpoint=True);
y = np.cos(x);
z = np.sin(x);
plt.plot(x,y);
plt.plot(x,z);
plt.title("Function $\sin(x)$ and $\cos(x)$",loc='left');
plt.xlim(-3.0,3.0);
plt.ylim(-1.0,1.0);
plt.xticks([-np.pi,-np.pi/2,0,np.pi/2,np.pi],[r'$-\pi$',r'$-
\pi/2$',r'$0$',r'$+\pi/2$',r'$+\pi$']);
plt.yticks([-1,0,+1],[r'$-1$',r'$0$',r'$+1$']);
plt.show();
```

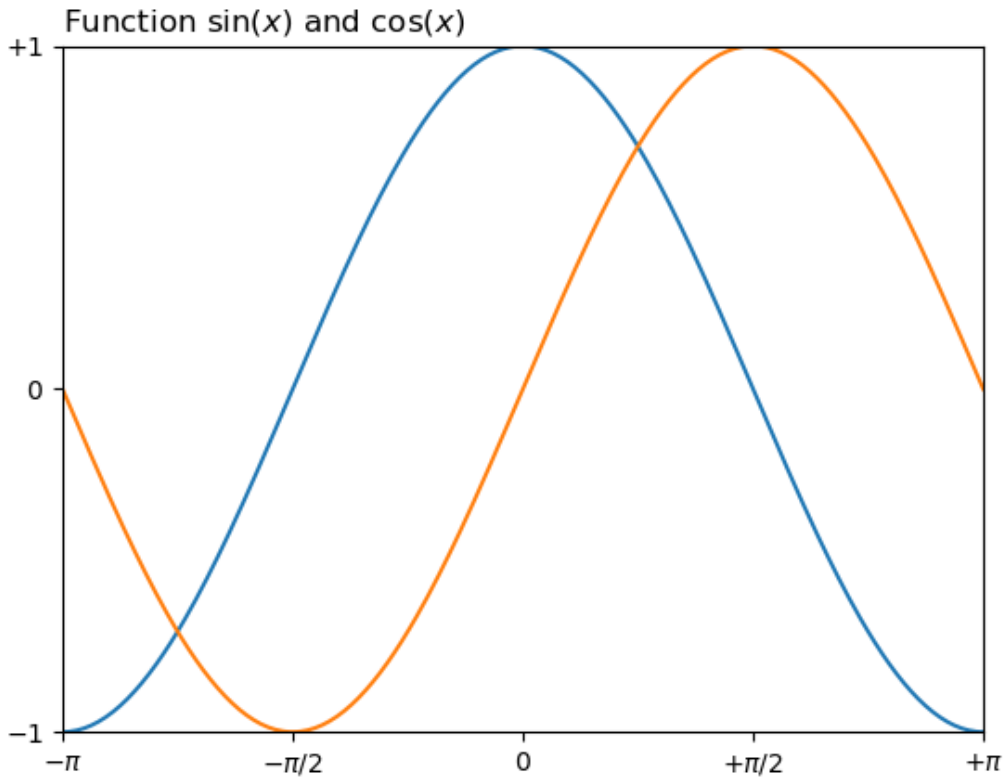

图10-1-4 添加数学符号的正弦-余弦函数

习 题

1. 利用Path、Patch绘制多个柱状图，其中第一个柱状图的高度为5，宽度为1，起始点为（0,0）点，第二个柱状图的高度为3，宽度为1，起始点为（3,0）点。

```
import matplotlib.pyplot as plt;
from matplotlib.path import Path;
import matplotlib.patches as patches;
points = [(0., 0.), (0., 5.), (1., 5.),(1., 0.), (0., 0.)];  # 五个坐标点
lines = [Path.MOVETO,Path.LINETO,Path.LINETO,Path.LINETO,Path.CLOSEPOLY];

points1 = [(3., 0.), (3., 2.), (4., 2.),(4., 0.), (3., 0.)];  # 五个坐标点

#点之间的连接方式
path = Path(points, lines);            #点和线形成路径
path1 = Path(points1,lines);
fig = plt.figure();                    #获取图形
ax = fig.add_subplot(111);             #获取数轴
patch = patches.PathPatch(path, facecolor='red', lw=2); #获取图形补丁
patch1 = patches.PathPatch(path1, facecolor='green', lw=2); #获取图形补丁
ax.add_patch(patch);
ax.add_patch(patch1);                                  #将图形补丁添加到数轴中
ax.spines['right'].set_color('none')                       #将图像右边的轴设为透明
ax.spines['top'].set_color('none')                        #将图像上面的轴设为透明
ax.xaxis.set_ticks_position('bottom')                  #将x轴刻度设在下面的坐标轴上
ax.yaxis.set_ticks_position('left')                   #将y轴刻度设在左边的坐标轴上
ax.spines['bottom'].set_position(('data', 0))    #将两个坐标轴的位置设在原点
ax.spines['left'].set_position(('data', 0))
ax.set_xlim(-2,6);                             #设定x轴范围
ax.set_ylim(-2,6);                             #设定y轴范围
plt.show();
```