附录一 Python 库简介

一、Python3.7 内建函数

函数	返回值类型	函数功能
abs( )	int|float	求绝对值,若是复数则返回复数的模
all( )	bool	若所有元素为真则返回 True(非 0,非空,非 None)
any( )	bool	如果任一元素为真则返回 True
ascii( )	str	同 repr().返回一个可打印的对象字符串方式表示，但非 ascii 字符就会输出\x,\u 或\U 等字符来表示。与 Python2 版本里的 repr()是等效的函数
bin( )	str	把整数转换为二进制字符串
bool( )	bool	返回一个布尔值,内容为空时返回 False
bytearray( )		返回一个新字节数组. 这个数组里的元素是可变的, 并且每个元素的值范围: 0 <= x < 256.
bytes( )	bytes	将字符串转换为字节类型
callable( )	bool	检查对象是否能调用
chr( )	str	把 ascii 码转换为字符
classmethod()		一般通过@classmethod 使用(创建类方法使用)
compile( )		将一个字符串编译为字节代码
complex( )	复数	传入实部和虚部(默认 0)来生成一个复数
delattr( )		删除属性. delattr(x, 'foobar')相当于 del x.foobar
dict( )	dict	新建字典(dict(a=10)返回{‘a’:10})
dir( )	list	无参数,返回当前局部名单列表.有参,试图返回该对象有效的属性列表.
divmod( )	tuple	以元组的形式返回 a//b 以及 a%b
enumerate( )	iterable	枚举,默认从 0 开始(iterable:可迭代对象)
eval( )		函数将字符串 str 当成有效 Python 表达式来求值,并返回计算结果.
exec( )		将字符串 str 当成有效 Python 代码来执行

filter( )	iterable	将每个值传入 fun 函数(None 保留全部)保留返回 True 的那些值
float( )	float	把字符串或者一个数转化成浮点数
format( )	str	格式化
frozenset( )	frozenset([])	返回一个冻结的集合，冻结后集合不能再添加或删除任何元素
getattr( )		获取对象 object 的属性或者方法，如果存在打印出来，如果不存在，打印出默认值，默认值可选. 需要注意的是，如果是返回的对象的方法，返回的是方法的内存地址，如果需要运行这个方法， 可以在后面添加一对括号.
globals( )	dict	以字典类型返回当前位置的全部全局变量
hasattr( )	bool	判断 obj 对象是否有 name 属性或方法
hash( )	int	获取取一个对象（字符串或者数值等）的哈希值
help( )		获取帮助信息
hex( )	str	把整数转换为十六进制字符串
id( )		返回一个对象的“身份”,对象与内存的连接
input( )	str	如果提供 prompt,则会不换行地写入到标准输出设备中.把输入内容读为字符串
int( )	int	截取整数部分或将 base 进制的字符串转换为十进制数
isinstance( )	bool	如果参数 object 是 classinfo 的一个实例则返回 True
issubclass( )	bool	判断参数 cls 是否是类型参数 classinfo(class 或 tuple)的子类
iter( )	iterable	创建一个迭代器
len( )	int	返回长度
list( )	list	将对象转换为列表
locals( )	dict	以字典类型返回当前位置的全部局部变量
map( )	iterable	(func, *iterables) iterables 中每个值代入 func 函数返回值生成新 iterables
max( )		返回最大值
memoryview( )		返回给定参数的内存查看对象(Momory view)
min( )		返回最小值,可输入一个 iterable 或多个值’,’隔开,可设置 key

		参数
next( )		返回迭代器的下一个项目. iterator: 可迭代对象; default ; 可选，用于设置在没有下一个元素时返回该默认值，如果不设置，又没有下一个元素则会触发 StopIteration 异常.
object( )		最基本的类型
oct( )	str	把整数转换为八进制字符串
open( )		打开文件
ord(( )		把字符转换为 ascii 码
pow( )		x 的 y 次方
print( )		sep: 输出字符串之间的分隔符; end: 结束符; file: 字符串要发送到的位置; flush: 是否立即输出
property( )		可以更好的调用类的 get, set, del 方法
range( )	iterable	([start,] stop[, step])开始值,结束值,步长
reload( )		重新加载模块(py2.x)
repr( )	str	将一个对象转成字符串显示,注意只是显示用，有些对象转成字符串没有直接的意思
reversed( )	iterable	返回一个逆置的迭代器
round( )		(number[, ndigits]) 返回浮点数 number 保留 ndigits 位小数四舍五入的值 py3.x 取整时该函数的返回离整数两边一样近优先取偶数.(2 遵四舍五入)
set( )	set	将对象转换为集合(一个无重复项的数组,可用来去重)
setattr( )		给对象的属性赋值, 若属性不存在, 先创建再赋值.
slice( )	切片对象	返回一个切片对象, 用来实现取出第 start 到 stop 间距为 step 的元素
sorted( )	list	对 iterable 进行排序, 可参考列表的 sort 方法
staticmethod( )		一般通过@staticmethod 使用(创建静态方法使用)
str( )	str	把内容转换为字符串
sum( )		元素求和

super( )	class	指向 obj 的父类
tuple( )	tuple	将对象转换为元组
type( )	type	返回独享的类型
vars( )	dict	当函数不接收参数时，其功能和 locals 函数一样，返回当前作用域内的局部变量. 参数可以是模块、类、类实例，或者定义了__dict__属性的对象
zip( )	iterable	从多个 iterables 的相同位置取出元素组成元组,并汇聚成一个新 iterables

二、Python 库

库简介：Python 中只有几十个函数是随 Python 的启动就直接加载到内存中的，可以直接使用，这些函数叫做内建函数（Built_in Funtions，是标准库的一部分）。但还有更多的函数并不能直接使用，需要通过导入后才能使用。Pytthon 中的函数是以库的形式提供给我们使用。库分为标准库和第三方库。标准库是随着 Python 解释器一起安装在电脑中的，它是 Python 的一个组成部分。第三方库是由一些 Python 爱好者编写供大家免费使用的库，需要单独下载安装才能使用。python 编程的挑战很大一部分来自于对库的应用，一旦掌握了核心语言，就需要花大量时间来研究各种内建函数和库。

标准库的组织方式：从文件组织方面看，除了少数几个内建模块外，标准库是由很多.py 文件组成的，这些.py 文件被放在一个文件夹中（默认路径为：..\Python\Python 版本号\Lib）。其中的每个.py 文件称为一个模块。

第三方库的组织方式：第三方库也是由许多,py 文件组成，这些.py 文件被放在一个或多个文件夹中（所有第三方库的默认路径：..\Python\Python 版本号\Lib \site-packages）。其中的每个.py 文件称为一个**模块**，文件夹称为**包**（包含__init__.py 文件的文件夹才能称为包）。

命名空间：这里仅对命名空间作着一个肤浅的比喻。Python 解释器在查找变量、函数、类等对象的时候需要到“命名空间”中去查找，但是，Python 程序运行时，“命名空间”可能有多个，不同的“命名空间”中的对象名称可能不同也可能相同。这里你把“命名空间”想象成班级，变量、函数等对象的名字想象成班级里的每个学生。在同一“命名空间”中对象的名字是不能相同的，比如：

```
01 class A():
02  pass
03 A = 3
```

程序执行完 01 行后，A 是类的名字，执行到 03 后，A 就不再是类的名字了。

导入模块：

有两种方式导入模块，分别介绍如下：

（1）使用 import 模块名 [as 别名] 语句导入

“as 别名”不是必须的，通常是在模块名比较长不容易记住的时候才使用。导入语句中的模块名可以是包或模块，如果是包时，则只能导入__init__.py 中定义的内容，包中的模块不会被导入。

1.使用这种方式导入模块时将产生一个新的命名空间，**相当于新增了一个班级**，命名空间的名字就是模块名或别名，模块中的所有对象的名字就装在这个命名空间中，所以访问这些对象就需要用“模块名.对象名”来访问。

2.一个 import 语句可以导入多个模块，模块之间用逗号（,）隔开即可。

（2）使用 from 模块名 import 对象 [as 别名] 语句导入

1.这种方式导入模块时不会产生一个新的命名空间，是直接将模块中的对象导入到当前命名空间中。所以访问这些模块中的对象就不需要加“模块名.”前缀，直接使用“对象名”或别名即可。

2.导入语句中的模块名可以是“包名.子包名.item”的形式，但 item 只能是包或模块，不能是类、函数、变量等。导入语句中的对象可以是包也可以是模块或其它命名，如函数、类、变量等，可以有多个用逗号隔开对象。

3.如果导入的模块中的对象名字与当前命名空间中的对象同名时，后面的将覆盖前面的，使用时需要注意。

4.使用这种方式导入还有一些其它的需要注意的问题。

下载与安装第三方库：

第三方库可以在 https://pypi.org/中找到。第三方库需要下载并安装才能使用，可以使用 pip 命令在命令行窗口中安装。pip 命令的语法格式如下：

pip command [modulename]

command：要执行的命令。modulename：库名称，当 command 为 instll 或 uninstall 时 modulename 不能省略。

或者先下载安装包（扩展名为 whl），然后使用“pip install 安装包”安装。

三、Numpy 库简介

知识拓展：Python 的绝大多数第三方库都提供了自己的数据结构，并提供了对这些数据结构进行操作的方法，就如同 Python 内建了列表、元组、字典等数据结构并提供了相应的操作方法一样。我们学习这些第三方库主要就是学习怎样使用这些方法，这些方法也称为 API。

NumPy 是 Python 进行科学计算的第三方基础库，需要额外安装才能使用。主要功能是提供同种元素的多维数组及对多维数组进行计算。NumPy 中最重要的对象是称为 ndarray 的 N 维数组类型，是一个快速而灵活的大数据集容器，所有元素具有相同的数据类型，可以使用索引和切片访问。由于这个库涉及到高等数学的知识，所以这里仅选择几个创建 ndarray 的方法和数学计算的方法进行介绍。

矩阵：上文提到的多维数组也称为矩阵。是指由多行和多列组成的一种数据结构。如：

$$A=\begin{bmatrix}3 & -1 & 2\\ 1 & 5 & 7\\ 2 & 4 & 5\end{bmatrix} \qquad B=\begin{bmatrix}7 & 5 & -2\\ 5 & 1 & 9\\ 4 & 2 & 1\end{bmatrix}$$

A、B 即为矩阵。

NumPy 提供的几个方法

方法	描述
array()	创建一个 ndarray 对象，参数为一个序列对象
arange()	用于生成一维等差数组，与 range 类似
linspace()	用于生成指定个数的等差数组
sin()	对矩阵中每个元素取正弦
cos()	对矩阵中每个元素取余弦
tan()	对矩阵中每个元素取正切

array()方法：

```
n=np.array([[10,5,3],[2,9,7],[3,5,1]])          #产生二维数组
```

arange()方法：

```
    n=np.arange(1,10,2)                         #产生一维数组
```

linspace()方法：

```
    n=np.linspace(1,10,num=5)                   #产生一维数组
```

arange()方法与 linspace()方法的差别是 arange()方法的最后一个参数指定步长，linspace()方法的 num 参数指定元素个数。

sin()方法：

对矩阵中每个元素取正弦，例子：

```
>>>n=np.arange(1,10,2)
>>>m=np.sin(n)
>>>print(m)
[ 0.84147098  0.14112001 -0.95892427  0.6569866   0.41211849]
```

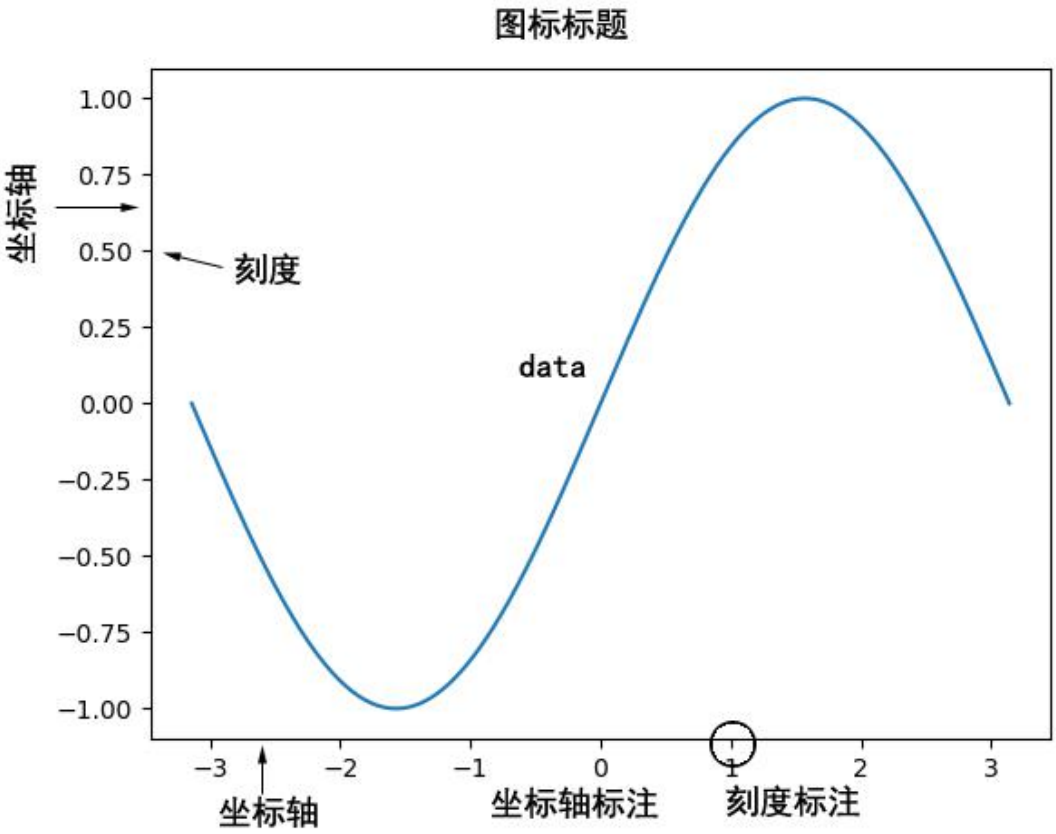
四、Matplotlib 库简介

Matplotlib 的官方说明书有 800 多页，足可见其庞大。这里我们仅仅对一些常用方法进行最简单的介绍，以指引你走上探索之路。Matplotlib 是 Python 用于数据可视化的第三方库，可以很方便地输出二维和三维数据，提供了笛卡尔坐标、极坐标、球坐标、三维坐标等，是最流行的数据可视化工具之一。它的快速绘图函数库主要集中在 Matplotlib.pyplot 模块中，约有 104 个方法。

习惯的导入语法是：import matplotlib.pyplot as plt。**后文都以 plt 为模块别名介绍**。

Matplotlib 绘图的流程是：为图表准备数据、创建图表、设置图表属性、显示图表。

基于笛卡尔坐标系的图表对象的基本结构如下图：



下面介绍 Matplotlib.pyplot 的几个基本方法。

matplotlib.pyplot 提供的常用方法

方法	描述
figure( )	创建画布，可选参数有名称、大小（单位为英寸）、dpi、前景颜色、边框颜色
subplot( )	在画布上创建一个子图，参数 numRows, numCols, plotNum，返回值：图表
subplots( )	在画布上创建多个子图，返回值：画布和图表数组
subplot2grid( )	通过栅格的形式创建一个子图，返回值：图表
scatte( )	绘制散点图
plot( )	绘制线形图
show()	显示图像

subplot( )方法：

matplotlib 一般不直接在 figure 上画图，都是在子图上画图。subplot(numRows, numCols, plotNum)方法用于在默认画布上创建一个子图。参数 numRows, numCols 是指将整个画布按 numRows 行、numCols 列进行重新划分。plotNum 是指按前面的划分，该子图占据编号为 plotNum 的子图。若 numRows, numCols, plotNum 都小于 10，可采用简写形式，如 subplot(3,3,2)可简写为 subplot(332)。创建子图的例子：

```
ax1=plt.subplot(2,2,1)          #将画布按 2×2 重新划分，该子图为第 1 个
ax2=plt.subplot(2,2,4)          #将画布按 2×2 重新划分，该子图为第 4 个
```

格式如**图 1 规则排列**

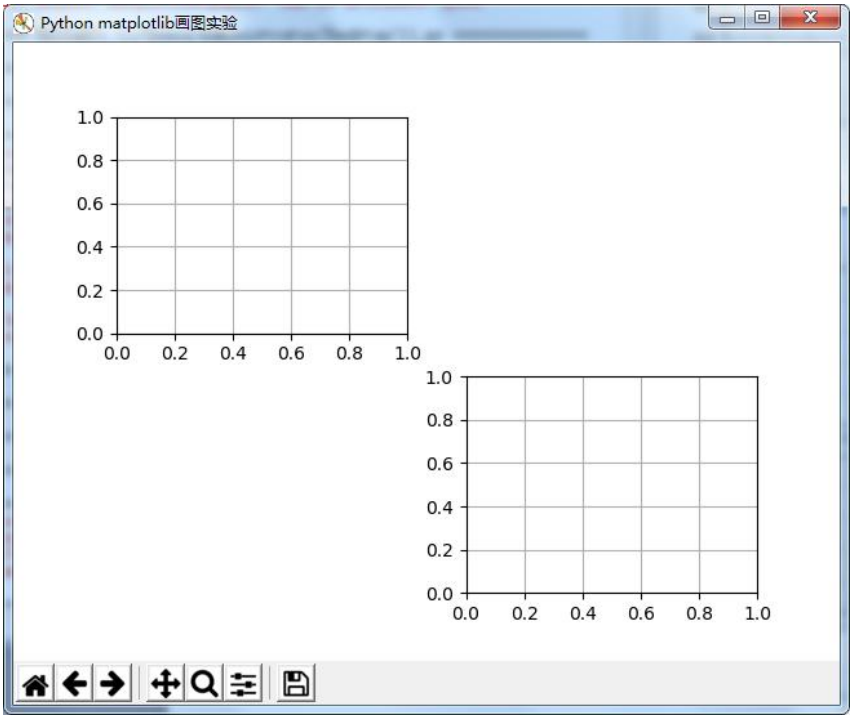


图 1 规则排列

```
ax1=plt.subplot(2,2,1)          #将画布按 2×2 重新划分，该子图为第 1 个
ax2=plt.subplot(2,2,2)          #将画布按 2×2 重新划分，该子图为第 2 个
ax3=plt.subplot(2,1,2)          #将画布按 2×1 重新划分，该子图为第 2 个
```

格式如**图 2 不规则排列**

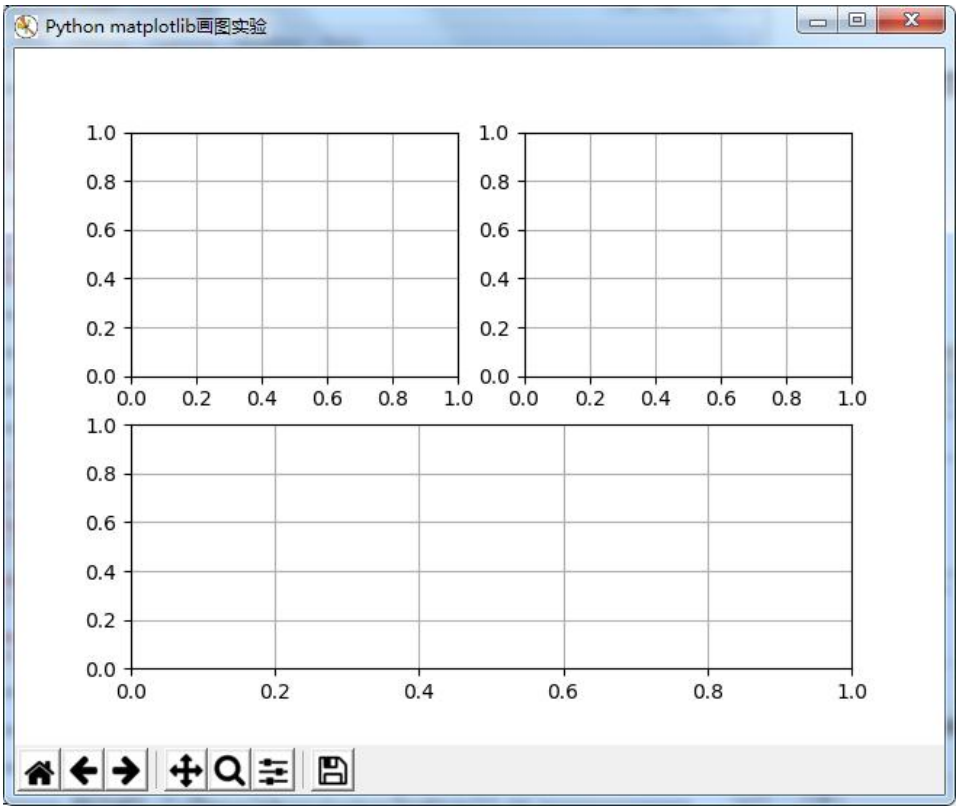


图 2 不规则排列

subplots( )方法：

基本语法格式如下：

fig,ax = subplots(nrows,ncols,**fig_kw)

nrows 和 ncols 表示将画布分割成几行几列，fig_kw 设置画布的关键字参数，fig：表示创建的新画布，ax：表示子图数组，左上角子图用 ax[0,0]引用，右下角子图用 ax[nrows-1, ncols-1]引用，其余按从左到右，从上到下的顺序引用。如：

fig,ax = plt.subplots(2,2,num=’Python matplotlib 画图实验’,subplot_kw=dict(polar=True))

ax[1,0].plot(np.random.randn(10)) #第 2 行第 1 个子图绘图

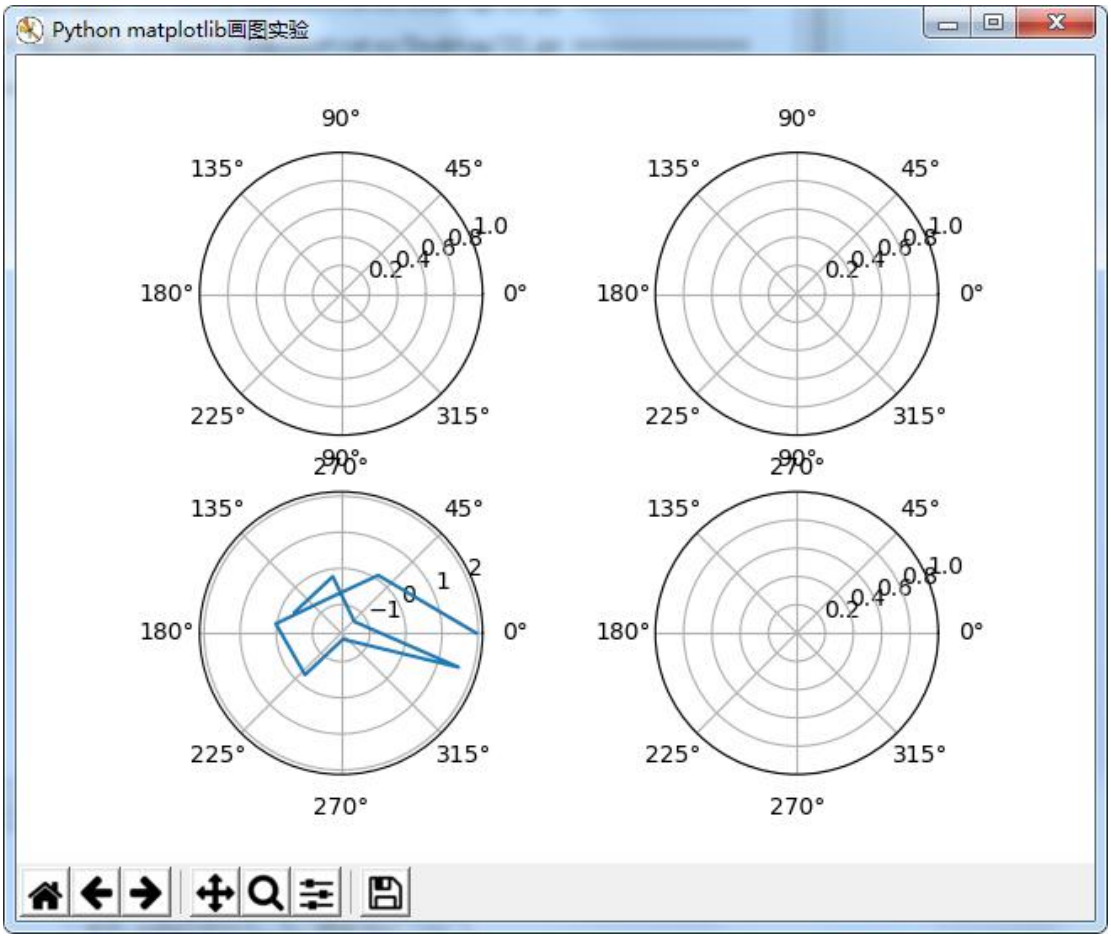


图 3 一次性生成子图规则布局

subplot2grid( )方法：

如果需要创建跨行或跨列的不规则的子图布局，使用 subplot2grid( )方法更容易。基本语法格式如下：

subplot2grid(shape, loc, rowspan=1, colspan=1, fig=None, **kwargs)

shape：含有两个整数的序列，分别表示将画布划分的行数和列数。loc：含有两个整数的序列，分别表示该子图的行索引和列索引。rowspan：跨行数，colspan：跨列数。

如下代码将产生图 4 跨行跨列布局的效果

```
ax1 = plt.subplot2grid((3,3), (0,0), colspan=3)
ax2 = plt.subplot2grid((3,3), (1,0), colspan=2)
ax3 = plt.subplot2grid((3,3), (1, 2), rowspan=2)
ax4 = plt.subplot2grid((3,3), (2, 0))
ax5 = plt.subplot2grid((3,3), (2, 1))
ax1.plot(x,np.sin(x))
ax2.plot(x,np.cos(x))
ax3.plot(x,x**x)
ax4.plot(x,x**3)
ax5.plot(x,x**2+2*x+5)
```

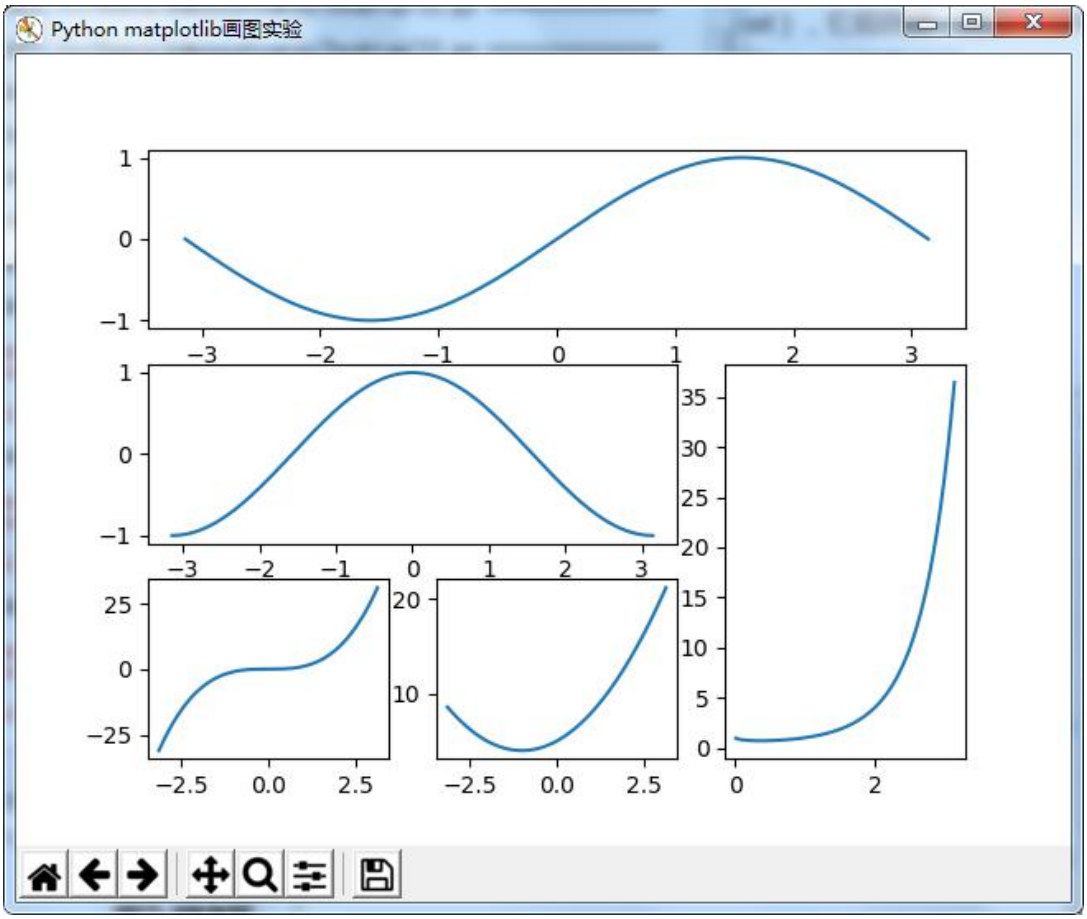


图 4 跨行跨列布局

scatte( )方法：

scatte()方法的功能是绘制散点图，基本语法格式如下：

scatte(x, y, s=None, c=None, marker=None, cmap=None, norm=None, vmin=None, vmax=None, alpha=None, linewidths=None, verts=None, edgecolors=None, *, data=None, **kwargs)

x,y：点的坐标，长度相同的数组

s：标记的大小

c：颜色

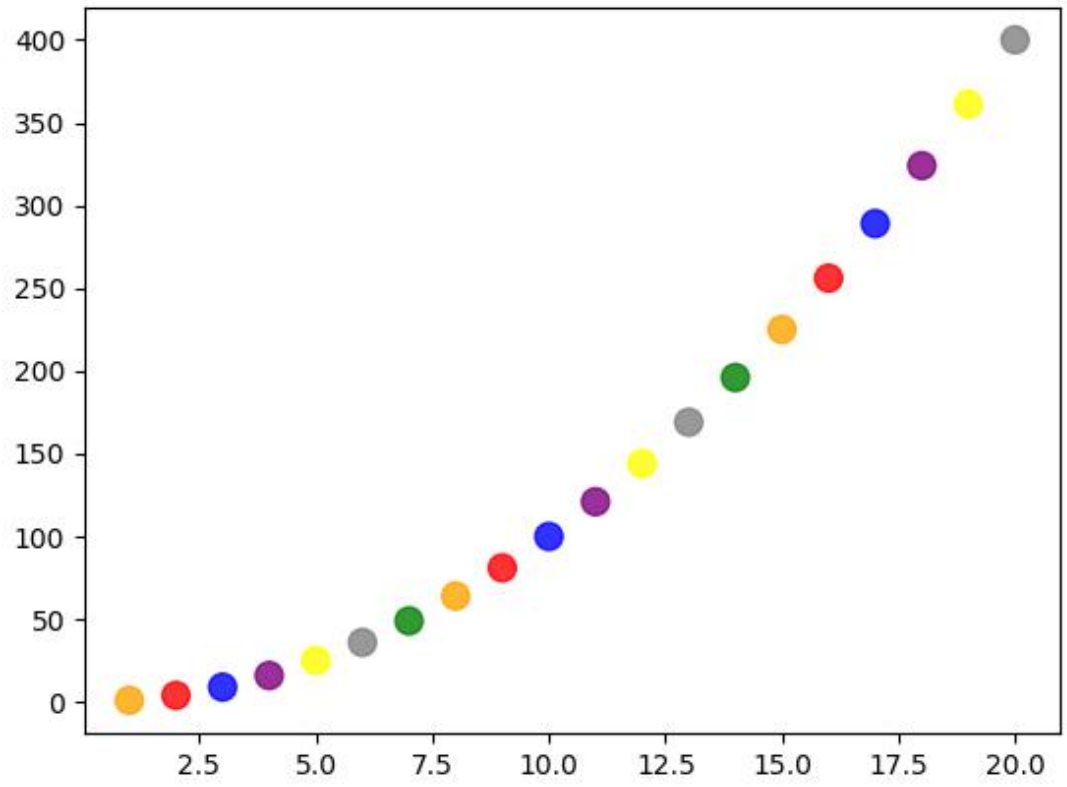
marker：标记样式

cmap：颜色映射。

例子：下面的这几行代码将画出函数 $y=x^2$ 的散点图。

```
01 import random
02 import numpy as np
03 import matplotlib.pyplot as plt
04 x=np.arange(1,21)
05 y=x**2
06 colors=['red','green','gray','purple','yellow','orange','blue']
```

```
07 random_colors=random.sample(colors,7)
08 plt.scatter(x,y,s=100,c=random_colors,alpha=0.8)
09 plt.show()
```



plot( )方法：

plot()方法是绘制线形图，其基本思想是先绘制一定数量的散列点，plt 自动把这些点用线连接起来。语法格式如下：

```
plot([x], y, [fmt], data=None, **kwargs)
plot([x], y, [fmt], [x2], y2, [fmt2], ..., **kwargs)
```

x,y：点的坐标，数组类型。

fmt：格式字符串参数，如果没有**连接线条样式字符**就会画出散点图。

kwargs：关键字参数。可以单独设置颜色、标记符号等属性。

plot()方法接受以下 fmt（格式字符串）字符来控制连接线条样式

标记	效果	标记	效果	标记	效果
'-'	实线	'--'	虚线	'-.'	点与线

plot()方法接受以下 fmt（格式字符串）字符来控制标记

标记	效果	标记	效果	标记	效果
'.'	点标记	','	像素标记	'o'	圆圈标记
'v'	倒三角标记	'^'	正三角标记	'<'	左三角标记
'>'	右三角标记	'1'	向下 Y 标记	'2'	向上 Y 标记

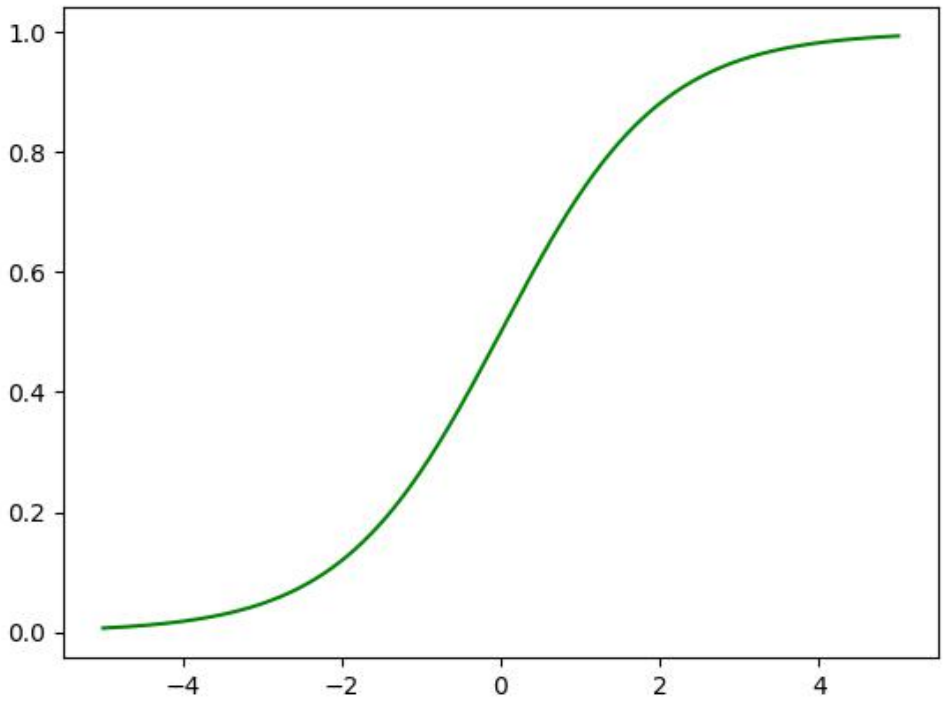
'3'	向左 Y 标记	'4'	向右 Y 标记	's'	正方形标记
'p'	五角星标记	'*'	*标记	'h'	六边形 1 标记
'H'	六边形 2 标记	'+'	+标记	'x'	x 标记
'D'	钻石标记	'd'	薄钻石标记	'|'	垂直线标记
'_'	水平线标记				

plot()方法接受以下 fmt（格式字符串）字符来控制颜色

字符	颜色	字符	颜色	字符	颜色	字符	颜色
'b'	蓝色	'g'	绿色	'r'	红色	'c'	青色
'm'	品红	'y'	黄色	'k'	黑色	'w'	白色

例子：下面的这几行代码将画出函数 $y=\frac{1}{1+e^{-x}}$ 的线形图

```
01 import random
02 import numpy as np
03 import matplotlib.pyplot as plt
04 x=np.linspace(-5,5,1000)
05 y=[1/(1+np.exp(-i)) for i in x]
06 plt.plot(x,y,'g-')
07 plt.show()
```

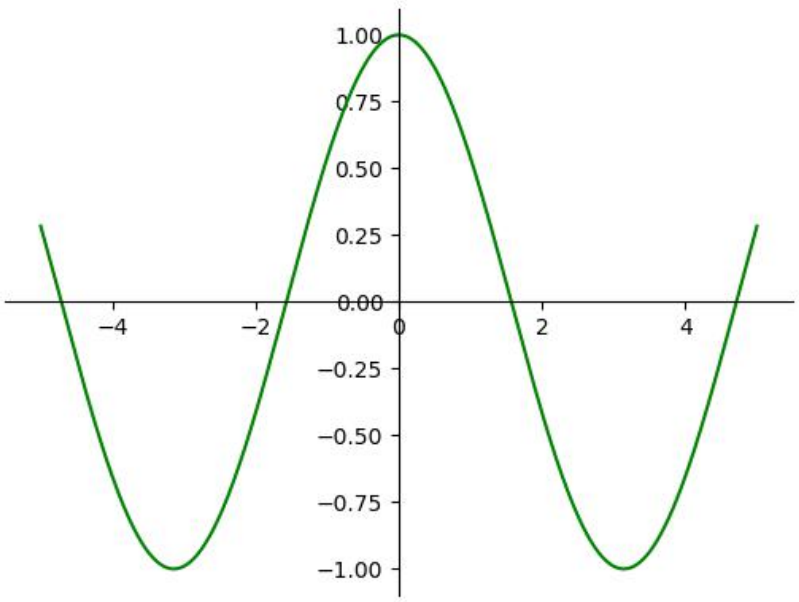


调整坐标轴的位置：

默认情况下，坐标轴是在四周，与我们已习惯的坐标系（零点重合）有所不同。调整方法是：将上边界和右边

界的颜色设置为 none，就隐藏了。然后将下边界和左边界移动到数据空间的 0 处。实现代码如下：

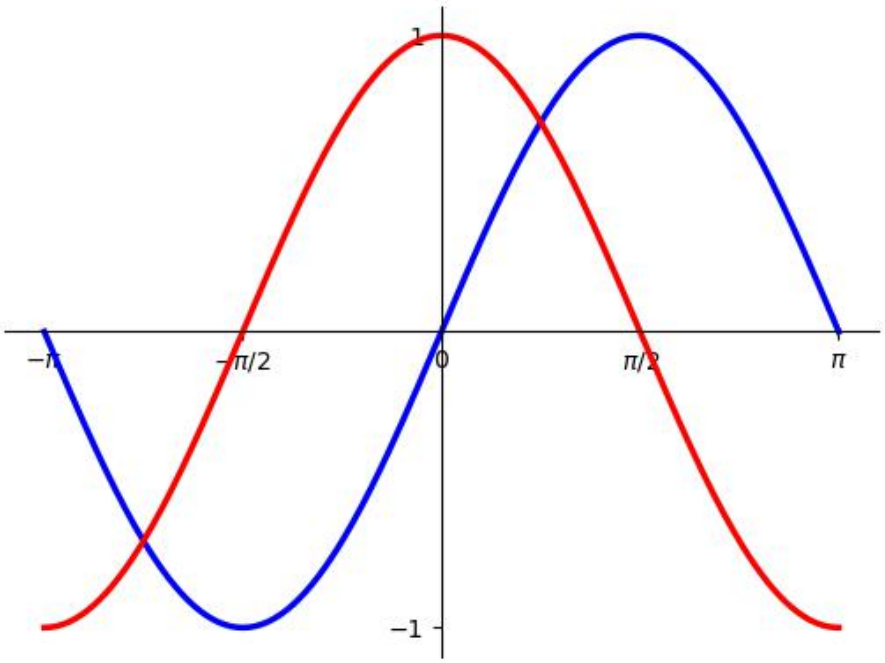
```
ax=plt.gca()                                    #获得当前 axes（轴）对象
ax.spines['right'].set_color('none')            #隐藏右边届
ax.spines['top'].set_color('none')              #隐藏上边届
ax.spines['bottom'].set_position(('data',0))    #移动底边的位置，移动到 y 轴数据区的 0
ax.spines['left'].set_position(('data',0))      #移动左边的位置，移动到 x 轴数据区的 0
```



设置标注：

使用轴对象的 set_xticks()和 set_yticks()方法设置，示例代码如下：

```
x=np.linspace(-np.pi,np.pi,256,endpoint=True)
sin,cos=np.sin(X),np.cos(X)

ax=plt.gca()                                    #获得当前 axes（轴）对象
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')
#ax.xaxis.set_ticks_position('bottom')
ax.spines['bottom'].set_position(('data',0))
#ax.yaxis.set_ticks_position('left')
ax.spines['left'].set_position(('data',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([-1,1])
plt.plot(x,sin,"b-",lw=2.5)
plt.plot(x,cos,"r-",lw=2.5)
```



五、Pygame 库简介

Pygame 是一个基于 SDL 库的基础上开发的用来开发游戏软件的 Python 第三方库，允许在 Python 程序中创建功能丰富的游戏和多媒体程序，适合用来开发小游戏。

pygame 库目前提供了 33 个模块，这里我们对 time 模块、sprite 模块、消息循环及 pygame 的初始化操作进行简单介绍。

导入模块语法：

```
import pygame
from pygame.locals import *
```

在使用 pygame 的任何功能之前须执行：pygame.init()方法进行初始化。

设置窗口大小：pygame.display.set_mode( )

设置窗口标题：pygame.display.set_caption( )

下面创建一个只有关闭功能的游戏窗口：

```
# _*_ coding:utf-8 _*_
import sys, time,  pygame
from pygame.locals import *
pygame.init()                                   #初始化 pygame
pygame.display.set_mode((200,150) )             #设置窗口 大小为 200,150
while True:                                     #死循环处理消息
    for event in pygame.event.get():            #遍历事件队列
        if event.type == pygame.QUIT:           #如果单击关闭按钮
            sys.exit                            #退出
      pygame.display.update()
pygame.quit()                                   #退出 pygame
```

运行程序界面如图所示：

如果你够细心的话，你会发现 CPU 的使用率将一路飙升，这是由于计算机以很快的速度执行死循环并不断刷新界面导致的。实际上并不需要很快的刷新速度，通常 1 秒内执行几十次就够了。解决方法是：首先在循环外创建时钟：clock = pygame.time.Clock()，然后在循环内使用 clock.tick(n)设置每秒循环次数即可降低 CPU 的使用率。

如果一直按住某个键不放， pygame 不会重复响应，只是在键第一次被按下的时候响应一次，因此不能从 pygame.event.get()事件队列中读取。有两种方法解决：一是轮询键盘，从 pygame.key.get_pressed()的返回值中判断键的状态。二是使用 pygame.key.set_repeat( ) 设置按住某个键每隔多少毫秒产生一个“按下”事件，推荐使用第一种方法。

time 模块的常用方法

方法	描述
pygame.time.get_ticks( )	返回自 pygame.init（）调用以来的毫秒数
pygame.time.set_timer( )	定时器，在事件队列上重复创建事件
pygame.time.Clock( )	创建一个时钟，跟踪时间
pygame.time.wait( )	暂停程序一段时间

pygame.sprite 模块：sprite 模块的功能是由精灵序列图产生动画。精灵序列图是由若干个大小相同、内容相似的小图组合而成。每个小图就是一帧。精灵序列图如下图所示：

由精灵序列图产生动画的原理：

每隔一小段时间更换图像，如果这个时间足够短就会有动画效果，这是所有用图像产生动画的基本原理。在创建精灵对象时加载整个精灵图，并根据帧的大小取出第一帧，然后每间隔一个固定时间更换下一帧。更换下一帧的算法代码如下：

```
def update(self,current_time, rate=120):
      if current_time > self.last_time + rate:    # self.last_time 上次更换帧的时间
          self.frame += 1                  # self.frame 当前帧序号
          if self.frame > self.last_frame:      # self.last_frame 最后一帧序号
              self.frame = self.first_frame    # self.first_frame 第一帧
              self.last_time = current_time
```

```
        if self.frame != self.pre_frame:          # self.pre_frame 前一帧序号
    # self.columns 精灵序列图的列数，self.frame_width 帧宽，self.frame_height 帧高，
            frame_x = (self.frame % self.columns) * self.frame_width
            frame_y = (self.frame // self.columns) * self.frame_height
            rect = ( frame_x, frame_y, self.frame_width, self.frame_height )
            self.image = self.master_image.subsurface(rect)
            self.pre_frame = self.frame
```

current_time 是使用时间计数器传来的时间（毫秒），rate 间隔时间（毫秒）。在主程序中不断调用这个方法更新帧即可。

更多内容请参阅官网：https://www.pygame.org/docs/

六、Requests 库简介

requests 库是一个使用 Python 语言编写的用于 http 请求的第三方库，可以方便的对网页进行爬取，是学习 Python 爬虫技术很好的 http 请求模块。

requests 库的几个主要方法

方法	描述
requests.request()	构造一个请求，支持以下各种方法
requests.get()	向 html 网页提交 get 请求的方法
requests.head()	获取 html 头部信息的主要方法
requests.post()	向 html 网页提交 post 请求的方法
requests.put()	向 html 网页提交 put 请求的方法
requests.patch()	向 html 提交局部修改的请求
requests.delete()	向 html 提交删除请求

以上这些方法都将返回 response 对象。

1. requests.get()方法：

该方法向服务器以 get 方式提交一个请求，是使用最多的请求方法，返回一个 response 对象。语法格式如下：

requests.get(url,params,**kwargs)

url 参数：欲爬取的网页地址，params：可选的字典或者字节流格式的参数，kwargs：可变长的关键字参数。

几个使用 get 方法请求的例子：

```
>>>response = requests.get( 'http://i.tryz.net' )
>>> response = requests.get( 'http://i.tryz.net/html/2018/xiaoyuan_0922/3546.html' )
```

url：http://gz.tryz.net/index.php?m=content&c=index&a=show&catid=43&id=15 的请求可写为：

```
data = { 'm': ' content ', 'c':' index',' a' :' show',' catid' :43,' id' :15}
>>> response = requests.get( 'http://gz.tryz.net/index.php', params=data)
```

kwargs 支持的键名为：data、json、headers、cookies、auth、files、timeout、proxies、allow_redirects、stream、verify、cert 共 12 个。具体含义不在此叙述。

```
>>> response = requests.get( 'http://i.tryz.net', timeout=1)  #设置请求超时时间为 1 秒。
```

2.requests.post()方法：

该方法向服务器以 post 方式提交一个请求，基本语法格式如下：

requests.post(url,data, files)

url 参数：欲爬取的网页地址，data：表单形式的数据，files：文件参数。

知识拓展：在 HTML 的表单提交中 get 是从服务器上获取数据，post 是向服务器上传数据。

response 对象的属性：

属性	描述
response.status_code	http 请求的返回状态，若为 200 则表示请求成功。
response. text	http 响应内容的字符串形式，即返回的页面内容
response. encoding	从 http header 中猜测的相应内容编码方式
response. apparent_encoding	从内容中分析出的响应内容编码方式（备选编码方式）
response. content	http 响应内容的二进制形式

requests 库的异常：

在进行资源请求时，可能会发生网络连接错误、http 错误、重定向异常、请求 url 超时等异常，为了使交互更友好，需要捕捉这些异常，通常情况是使用 response.raise_for_status() 方法去引发异常。这个方法会在内部判断 response.status_code 是否等于 200，如果不等于，则引发异常。例子：

```
try:
    response = requests.get(‘http://i.tryz.net’)
    response.raise_for_status()
    #正常情况的语句
except:
    print(‘出现异常’)
```

七、Beautiful Soup 库简介

Beautiful Soup 是一个用于从 HTML 和 XML 文件中提取数据的 Python 第三方库，从网页抓取数据是其最主要的功能。当前版本是 Beautiful Soup 4，可以使用命令：pip install beautifulsoup4 安装。

使用这个库需要对 HTML 文档结构有一个概要的了解，下面对 HTML 文档进行简单介绍：

HTML 文档是由一些可以嵌套的成对的（或自关闭）标签组成。如：

一个简单的 HTML5 文档：

```
<!DOCTYPE html>
  <html lang="en">
  <head>
      <meta charset="UTF-8">
      <title>Title</title>
  </head>
  <body>
    <div class="ch-tab">
        <div class="tab-item js-tab"><a href="javascript:void(0)">新疆工程学院</a></div>
        <div class="tab-item js-tab"><a href="javascript:void(0)">清华大学</a></div>
        <div class="tab-item js-tab"><a href="javascript:void(0)">北京化工大学</a></div>
        <div class="tab-item js-tab"><a href="javascript:void(0)">山东大学</a></div>
    </div>
    </body>
  </html>
```

这个文档被浏览器解析后，显示效果为下面 4 个超链接：

新疆工程学院　清华大学　北京化工大学　山东大学

成对标签的例子：<a　href=’http://i.tryz.net’>铜仁一中

标签嵌套的例子：

　Python

　JAVA

自关闭标签的例子：

成对标签的语法：

<标签名 属性名 1=”属性值 1” 属性名 2=”属性值 2” … >内容< /标签名>

父节点、子节点、兄弟节点：

新疆工程学院的父节点是 ，

反过来 的子节点是新疆工程学院，具有同一父节点的所有节点互称兄弟节点。

Beautiful Soup 的主要工作就是解析 HTML 文档，当把一个 HTML 文档传递给 Beautiful Soup 时，Beautiful Soup 会把这个文档转换成一个复杂的节点树形结构，每个节点都是 Python 对象。下面介绍 Beautiful Soup 的基本使用方法：

在使用 Beautiful Soup 之前，必须要先导入 beautifulsoup 库和创建 beautifulsoup 对象。

（1）导入语法：from bs4 import BeautifulSoup

（2）创建对象语法：soup = BeautifulSoup(html, features="html.parser")，html 为一个符合 HTML 文档格式的字符串。**后文都以 soup 作为已经创建好了的 beautifulsoup 对象进行介绍，下面的一段 HTML 代码将作为例子被多次用到，简称爱丽丝。**

```
html_doc = """
    <html><head><title>The Dormouse's story</title></head>
    <body>
    <p class="title"><b>The Dormouse's story</b></p>
    <p class="story">Once upon a time there were three little sisters; and their names were</p>
    <a href="http://example.com/elsie" class="sister" id="link1">Elsie</a>,
    <a href="http://example.com/lacie" class="sister" id="link2">Lacie</a> and
    <a href="http://example.com/tillie" class="sister" id="link3">Tillie</a>;
    and they lived at the bottom of a well.</p>
    <p class="story">...</p>
    <!--这是一段注释。注释不会在浏览器中显示。-->
    """
```

```
>>>from bs4 import BeautifulSoup
>>>soup = BeautifulSoup(html_doc, 'html.parser')        #创建 soup 对象
```

soup 对象包括**标签、标签中的字符串、文档的全部内容、注释**四种对象。下面是 HTML 标签中的一些名词。

<!-- -->是注释标签，里面的内容是注释内容。

BeautifulSoup 提供的一些属性和方法

属性/方法	描述
.标签名	获取标签，返回 bs4.element.Tag 类型
.next_sibling	获取后一个兄弟节点
.previous_sibling	获取前一个兄弟节点
.find_next_sibling	获取同标签的后一个兄弟节点
.find_ previous _sibling	获取同标签的前一个兄弟节点
.next_elements	后面的所有节点
.previous_elements	前面的所有节点
.contents	返回直接子节点，列表类型
.children	返回直接子节点，生成器类型
.descendants	返回所有子孙节点，生成器类型
.parent	获取父标签
.string	获取标签中的字符串
.strings	获取标签中的所有字符串
.stripped_strings	获取标签中的所有字符串，移除多余空格和空行
.find_all( )	查找符合条件的节点
.select( )	css 选择器

下面我们选择几个属性和方法介绍：

标签的获取：

使用“soup.标签名”获取。

```
>>>soup.title        #获取 title 标签，输出：<title>The Dormouse's story</title>
>>>soup.a          #获取第一个 a 标签，输出：<a href="http://example.com/elsie" class="sister"
```

id="link1">Elsie

>>>soup.p #获取第一个 p 标签,输出： The Dormouse's story

说明：当同一标签名具有多个时，这种方法只能获取第一个标签。如果需要获取所有标签时使用 find_all()方法。例子：

```
>>>for a in soup.find_all( 'a' ):
      print(a)
```

输出：

```
<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>
<a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>
<a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>
```

获取兄弟：

使用 next_sibling 和 previous_sibling 获取兄弟节点。

```
>>>soup.a. previous_sibling          #获取前一兄弟节点
```

输出： Once upon a time there were three little sisters; and their names were

```
>>> soup.a. next_sibling             #获取后一兄弟节点
```

使用 find_next_sibling 和 find_previous_sibling 获取同标签名的兄弟节点，注意与前面两个属性的区别。

获取父亲：

>>>soup.title.parent #获取 title 的父标签，输出：<head><title>The Dormouse's story</title></head>

获取后代：

获取直接子节点使用 contents 和.children 属性。查找子孙节点使用 descendants。

标签具有 name 和 attrs 属性。

获取满足条件的节点：

（1）find_all()方法，对标签进行是否符合过滤器条件的搜索。这里仅举一例。

>>>soup.find_all("p", "title") #搜索标签名为 p，属性值为 title 的标签，输出： The Dormouse's story

（2）select()方法，是使用 CSS 选择器的语法找到 tag，返回 list 类型。

```
>>> soup.select("body a")          #搜索 body 子孙标签中的 a 标签。
>>> soup.select("body > a")        #搜索 body 的儿子 a 标签。
```

这两个方法都具有过滤功能，在实际使用中，需根据具有情况进行选择。

获取标签中的字符串：

使用 string 获取标签中的字符串，但如果一个标签包含了多个子节点，由于 string 无法确定应该调用哪个子节点的内容，所以输出结果是 None，这时可以使用 strings 或 stripped_strings 来循环获取。

```
>>> soup.title.string              #获取 title 标签的字符串内容
```

输出："The Dormouse's story"

```
>>>for s in soup.body. stripped_strings:
    print(s)
```

输出：（略）

八、Python 标准库 turtle 介绍

一、概述

Turtle 库是 Python 语言中一个轻量级绘制图像的函数库，是标准库之一，

只有大约90个方法。具有简单易学、功能强大的特点，适合于学习Python程序设计语言的入门篇。使用时需要使用 import turtle 命令导入。官网地址：https://docs.python.org/3/library/turtle.html，文中的“海龟”、“乌龟”、“画笔”具有相同意义。

二、Turtle的基本概念

1. 画布

在Turtle运行时会打开一个新窗口，这个窗口里面可以绘图的区域称为画布，画布默认大小为(400, 300)。

★注意画布与窗口的区别，窗口包括画布与边界。

2. 坐标系

Turtle采用直角坐标系，坐标原点为绘图窗体中心，默认模式下，向右为x轴正方向，向上为y轴正方向。在坐标原点处有一只面向x轴正方向的“乌龟”。

★对于“乌龟”而言，任何时候都有前进方向、后退方向、左侧方向、右侧方向，随着“乌龟”的爬行，这些方向会不断改变。

3. 画笔对象

Turtle的画图原理是用程序控制“乌龟”在画布上移动，“乌龟”爬过（只能爬线段和圆弧）的痕迹所组成的图案就是我们的作品。这个“乌龟”就称为画笔对象，简称画笔。如果需要创建新“画笔”，使用“别名.clone()或别名.Pen()或别名.Turtle()”语句实现。实现在画布上多个“画笔”同时会话需要使用多线程。

★如果希望在某些地方不留下痕迹，就需要“抬笔”，需要留下痕迹就“落笔”，默认是“落笔”状态。

4. 屏幕对象

屏幕对象可理解为 turtle 开启的窗口，是一个全局对象，包含操作窗口、事件等方法。使用“画笔.getscreen()或画笔.Screen()”语句可获取屏幕对象。

★当“屏幕”类的方法对应函数被调用时会自动创建一个 Screen 对象。当“画笔”类的方法对应函数被调用时会自动创建一个匿名的 Turtle 对象。即：**以“模块名.方法（）”方式可以调用所有方法**。为了程序代码的可读性，建议先获取屏幕对象和画笔对象后，再分别调用。

三、方法详述

屏幕对象方法

方法	描述
全局方法	
screen.screensize(canvwidth=None, canvheight=None, bg=None)	设置**画布**的宽、高、背景颜色
screen.setup(width=0.5, height=0.75, startx=None, starty=None)	设置**窗口**的宽、高，窗口左上角坐标
screen.bgcolor()	设置画布背景颜色
screen.bgpic()	设置或获取屏幕背景图片，只支持 gif 图片
screen.clearscreen()	清除画布
screen.resetscreen()	重置画布
screen.title()	设置窗口标题
screen.addshape()	向 shapelist 中添加画笔形状，别名：register_shape()
screen.getshapes()	返回所有当前可用海龟形状的列表
screen.mode()	设置海龟模式（"standard", "logo" 或 "world"）并执行重置。如未指定模式则返回当前的模式.
screen.colormode()	返回颜色模式或将其设为 1.0 或 255
screen.getcanvas()	返回此 TurtleScreen 的 Canvas 对象
screen.turtles()	返回屏幕上的海龟列表
screen.window_height()	返回海龟窗口的高度
screen.window_width()	返回海龟窗口的宽度
屏幕事件	
screen.listen()	使屏幕获得焦点
screen.onkey()	绑定按键释放事件，别名：onkeyrelease()
screen.onkeypress	绑定指定键的按下事件
screen.onclick()	绑定鼠标单击屏幕事件，别名：onscreenclick()

screen.ontimer()	设置一个计时器
screen.mainloop()	开始事件循环,别名：done()
screen.bye()	关闭海龟绘图窗口
screen.exitonclick()	鼠标单击屏幕关闭海龟绘图窗口
screen.delay()	设置或返回以毫秒数表示的延迟值
screen.tracer()	启用/禁用海龟动画并设置刷新图形的延迟时间
screen.update()	执行一次 TurtleScreen 刷新
输入方法	
screen.textinput()	弹出一个对话框窗口用来输入一个字符串
screen.numinput()	弹出一个对话框窗口用来输入一个数值

（1）screensize()方法 ——设置画布

语法格式：screen.screensize(canvwidth=None, canvheight=None, bg=None)

canvwidth,canvheight,bg 分别表示画布的宽、高、背景颜色。如果希望省略部分参数，需要使用关键字参数格式调用。不带参数时，返回当前画布大小（元组格式）。

（2）setup()方法 ——设置窗口

语法格式：screen.setup(width=0.5, height=0.75, startx=None, starty=None)

width、height、startx、starty 分别表示窗口宽、高、左上角 x 坐标、左上角 y 坐标。width 和 height 为整数时表示像素，为小数时 表示相对屏幕大小的比例。如 0.75 表示 75%。当省略 startx、starty 时，窗口位于屏幕中央。

如果同时使用 screensize()、setup()设置了画布和窗口，当画布大于窗口时，窗口会出现滚动条，“小乌龟”会爬到窗口外绘制图形；当画布小于窗口时，画布会填充窗口。通常情况使用 setup()方法设置即可。

（3）bgcolor()方法 ——设置画布背景颜色

语法格式：screen.bgcolor(*args)

args： 一个颜色字符串或三个取值范围 0..colormode 内的数值或一个取值范围相同的数值 3 元组。例子：

screen.bgcolor（“#FF0000”）

screen.bgcolor(0.1,0.2,0.3),screen.bgcolor(128,128,128)

screen.bgcolor((128,1,121))

当省略 args 时，返回画布当前背景颜色。

（4）bgpic()方法　——设置或获取屏幕背景图片

语法格式：screen.bgpic(picname=None)

picname:如果为文件路径，则设置背景图片为 picname,如果为“nopic”，则删除背景图片，如果为 None 则返回背景图片名。该方法没有拉伸功能。

（5）clearscreen()方法　——清除画布

语法格式：screen.clearscreen()

删除所有海龟的全部绘图。将已清空的屏幕重置为初始状态：一片空白，啥也没有。

（6）resetscreen（）方法　——重置画布

语法格式：screen.resetscreen（）

重置屏幕上的所有海龟为其初始状态,删除所有海龟所绘制的图形，不改变背景颜色、背景图像等设置。

（7）title()方法　——设置绘图窗口标题

语法格式：screen.title(titlestring)

titlestring：一个字符串，海龟绘图窗口的标题栏文本，不能省略。

（8）screen.addshape()方法　——向 shapelist 中添加画笔形状。

语法格式：screen.addshape(name, shape=None)

name:一个 gif 文件名或一个字符串， shape：图形形状或 None。只有这样注册过的形状才能通过执行 turtle.shape(shapename) 命令来使用形状。例子：

screen.addshape（“d:/888.gif”）

screen.addshape(“mm”,((1,1),(45,45),(100,100)))

（9）getshapes()方法 ——返回所有当前可用海龟形状的列表

语法格式：screen.getshapes()

（10）mode()方法 ——设置海龟模式（"standard", "logo" 或 "world"）并执行重置。如未指定模式则返回当前的模式.

语法格式：screen.mode(mode=None)

mode:乌龟模式，可选值：standard、logo、world,不同模式的区别：

模式	初始画笔朝向	正角度
standard	向右（东）	逆时针
logo	向上（北）	顺时针

如果省略该参数，将返回当前模式，在没有省略参数的情况下，将清除所有乌龟已画的图像并重置所有乌龟。

（11）colormode()方法 ——返回颜色模式或将其设为 1.0 或 255。

语法格式：screen.colormode(cmode=None)

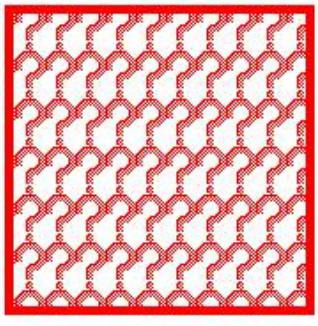
cmode：数值 1.0 或 255 。省略时返回颜色模式

（12）getcanvas()方法 ——返回 Tkinter 的 Canvas 对象

语法格式：screen.getcanvas()

可以使用 Canvas 对象的方法在屏幕上绘图。如：

```
c = screen.getcanvas()
c.create_rectangle(30, 30, 200, 200,
      outline='red',# 边框颜色
      stipple = 'question',# 填充的位图
      fill="red",# 填充颜色
      width=5 # 边框宽度
```



（13）turtles()方法 ——返回屏幕上的乌龟列表

语法格式：screen.turtles()

（14）window_height()方法 ——返回屏幕的高度

语法格式：screen.window_height()

（15）window_width()方法 ——返回屏幕的宽度

语法格式：screen.window_width()

（16）listen()方法 ——使屏幕获得焦点，即使绘图窗口位于最顶层，也便绘图窗口接收键盘和鼠标事件。

语法格式：screen.listen()方法

（17）onkey()方法 ——绑定（注册）按键释放事件

语法格式：screen.onkey(fun, key)

fun：一个无参数的函数名或 None，如果 fun 为 None，则移除事件绑定。key：一个表示按键的字符串，不能省略: 例如 "a"、"space"。

（18）onkeypress（）方法 ——绑定（注册）按键按下事件

语法格式：screen.onkeypress（fun, key）

参数意义同 screen.onkey（）方法中的参数

（19）onclick()方法 ——绑定（注册）鼠标单击屏幕事件

语法格式：screen.onclick(fun, btn=1, add=None)

fun：一个具有两个参数的函数名，调用时将传入两个参数表示在画布上点击的坐标，如果 fun 为 None，则移除现有的绑定。btn：鼠标按钮编号，默认值为 1 (鼠标左键)。add：True 或 False，如为 True 则将添加一个新绑定，否则将取代先前的绑定。

（没有测试出双击和右击的 btn 取值）

（20）ontimer()方法 ——设置定时器

语法格式：screen.ontimer(fun, t=0)

fun：一个无参数的函数，t：一个大于或等于 0 的毫秒数。功能是经过 t 时间后执行 fun。

（21）mainloop()方法 ——开始事件循环

语法格式：screen.mainloop()

开始事件循环，使绘图窗口不关闭。这条语句后面的语句不会被执行，因此必须作为一个海龟绘图程序的结束语句。

（22）bye()方法 ——关闭绘图窗口

语法格式：screen.bye()

（23）exitonclick()方法 ——鼠标单击屏幕，关闭海龟绘图窗口

语法格式：screen.exitonclick()

（24）delay()方法 ——设置或返回以毫秒数为单位的刷新图形的延迟时间

语法格式：screen.delay(delay=None)

delay：延迟的毫秒数。数值越大，动画速度越慢。注意与 turtle.speed()方法的区别。speed()方法是指在一次绘画过程中画笔的移动速度，delay()方法是指完成一次绘画之后延迟多久才开始下一次绘画。

（25）tracer（）方法 ——启用（或禁用）画笔动画，并设置刷新图形的延迟时间。

语法格式：screen.tracer（n=None, delay=None）

n:为 0 时直接显示绘图结果，非 0 整数时启用绘图动画效果，delay：在启用绘图动画效果时，设置绘图延迟时间，等价于 screen.delay（）方法。

（26）update()方法 ——执行一次 TurtleScreen 刷新

语法格式：screen.update()

（27）textinput()方法 ——弹出一个输入文本的输入框

语法格式：screen.textinput(title, prompt)

title：对话框标题文本；prompt：对话框提示信息文本。当按 OK 时，返回输入的文本。

（28）numinput()方法 ——弹出一个输入数字的输入框，可以是实数。

语法格式：screen.numinput(title, prompt, default=None, minval=None, maxval=None)

title：对话框标题文本；prompt：对话框提示信息文本；default：默认值；minval：最小值； maxval：最大值。

画笔对象方法

方法	描述
画笔控制	
turtle.pendown()	落笔，别名 pd()或 down()
turtle.penup()	抬笔，别名 pu()或 up()
turtle.pensize()	画笔粗细，别名 width()
turtle.shapesize()	设置或返回画笔的拉伸因子
turtle.resizemode()	画笔大小调整模式
turtle.color()	设置或返回画笔颜色和填充颜色
turtle.pencolor()	设置或返回画笔颜色
turtle.fillcolor()	设置或返回填充颜色
turtle.shape()	设置或返回画笔形状
turtle.pen()	设置或返回画笔的属性
turtle.speed()	设置或返回画笔移动速度
turtle.clear()	清除该画笔留下的痕迹
turtle.reset()	重置画笔
turtle.degrees()	设置一个圆周为多少度
turtle.radians()	设置角度的度量单位为弧度

（1）pendown()（）方法 ——落笔，使画笔运动后能留下痕迹

语法格式：turtle.pendown()

（2）penup()方法　——抬笔，使画笔运动后不留下痕迹

语法格式：turtle.penup()

（3）pensize()方法　——画笔粗细

语法格式：turtle.pensize(width=None)

width:设置线条的粗细为 width 或返回该值。如果 resizemode 设为 "auto" 并且 turtleshape 为多边形，该多边形也以同样组细的线条绘制。如未指定参数，则返回当前的 pensize。

（4）shapesize()方法　——设置或返回画笔的拉伸因子，且设置画笔大小调整模式为 "user"

语法格式：turtle.shapesize(stretch_wid=None, stretch_len=None, outline=None)

stretch_wid：直于其朝向的宽度拉伸因子，stretch_len：平行于其朝向的长度拉伸因子，outline：形状轮廓线的粗细。

如果有一个参数指定，则设置画笔大小调整模式user。

（5）resizemode()方法 ——画笔大小调整模式

语法格式：turtle.resizemode(rmode=None)

rmode：字符串 "auto"或"user"或"noresize"，如未指定 rmode 则返回当前的画笔大小调整模式。

"auto"：根据画笔粗细自动调整画笔的外观。

"user"：根据拉伸因子和轮廓宽度（outline）值调整画笔的外观，两者是由 shapesize() 设置的。

"noresize"：不调整画笔的外观大小。

（6）color（）方法 ——设置或返回画笔颜色和填充颜色

语法格式：turtle.color(args)

args:0 个、1 个或 2 个表示颜色的参数。

（7）pencolor()方法 ——设置或返回画笔颜色

语法格式：turtle.pencolor(args)

args:0 个或 1 个表示颜色的参数。

（8）fillcolor()方法 ——设置或返回填充颜色

语法格式：turtle.fillcolor(args)

args：0 个或 1 个表示颜色的参数。

（9）shape()方法 ——设置或返回画笔的形状

语法格式：turtle.shape(name=None)

name：一个有效的形状名字符串。

（10）pen()方法 ——设置或返回画笔的属性

语法格式：turtle.pen(pen=None, pendict)

pen：一个包含部分或全部下列键的字典

pendict：一个或多个以下列键为关键字的关键字参数

==========

"shown"：True/False

"pendown"：True/False

"pencolor"：颜色字符串或颜色元组

"fillcolor"：颜色字符串或颜色元组

"pensize"：正数值

"speed"：0..10 范围内的数值

"resizemode": "auto" 或 "user" 或 "noresize"

"stretchfactor": (正数值, 正数值)

"outline": 正数值

"tilt": 数值

(11)speed()方法 ——设置或返回画笔的移动速度

语法格式: turtle.speed(speed=None)

speed: 一个 0 到 10 范围内的整型数或速度字符串，数字越大速度越快，**为 0 表示没有动画效果**。如果输入数值大于 10 或小于 0.5，则速度设为 0。速度字符串如下:

"fastest": 0 最快、"fast": 10 快、"normal": 6 正常、"slow": 3 慢、"slowest": 1 最慢。

（12）clear()方法 ——清除该画笔留下的痕迹

语法格式: turtle.clear()

只清除该画笔留下的痕迹，不改变画笔的位置，颜色等属性。

（13）reset()方法 ——重置画笔

语法格式: turtle.reset()

清除与重置的区别，清除画笔只删除画笔走过的痕迹，不改变画笔的位置、方向及其他属性，重置画笔将清除所有痕迹及属性设置，使画笔为初始状态。

（14）degrees()方法 ——设置一个圆周为多少度，默认值为 360 度。

语法格式: turtle.degrees(fullcircle=360.0)

（15）radians()方法 ——设置角度的度量单位为弧度

语法格式: turtle.radians()

方法	描述

移动和绘制	
turtle.forward()	前进，别名：fd()
turtle.backward()	后退，别名： bk()或 back()
turtle.circle()	画园
turtle.dot()	画点
turtle.begin_fill()	开始填充
turtle.end_fill()	结束填充
turtle.begin_poly()	开始记录多边形
turtle.end_poly()	结束记录多边形
turtle.get_poly()	获取最后记录的多边形
turtle.write()	书写文本
turtle.setheading()	设置朝向，别名：seth()
turtle.right()	右转，别名：rt()
turtle.left()	左转，别名：lt()
turtle.goto()	定位，别名： setpos()或 setposition()
turtle.setx()	设置 x 坐标
turtle.sety()	设置 y 坐标
turtle.home()	返回原点
turtle.undo()	撤销
turtle.stamp()	印章
turtle.clearstamp()	清除印章
turtle.clearstamps()	清除多个印章

（1）forward()方法 ——沿画笔的朝向从当前位置前进

语法格式：turtle.forward(distance)

distance：前进的距离，一个整数或小数

（2）backward()方法 ——沿画笔的朝向从当前位置后退

语法格式：turtle.backward(distance)

distance：后退的距离，一个整数或小数

（3）circle（）方法 ——画园或圆弧或园内接正多边形

语法格式：turtle.circle(radius, extent=None, steps=None)

radius：半径；extent：角度；steps：作半径为 radius 的圆的内接正多边形，多边形边数为 steps。

画圆时：半径的正(负)表示圆心在画笔的左边(右边)，这里的左边和右边是指作一条垂直于画笔的直线，站在画笔的方向看，为正时圆心在左边，为负时圆心在右边。角度为正沿画笔方向前进，角度为负沿画笔方向后退。

（4）dot()方法 ——以当前位置为圆心画一个指定直径和颜色的点。

语法格式：turtle.dot(size=None, color)

size:直径，如果省略，直径为 max{画笔大小加 4,2 倍画笔},color：颜色。画完后画笔回到圆心。

（5）begin_fill()方法 ——记录要填充的多边形的起始点

语法格式：turtle.begin_fill（）

（6）end_fill()方法 ——记录要填充的多边形的结束点并执行填充

语法格式：turtle.end_fill（）

（7）begin_poly()方法 ——记录要获取的多边形的起始点

语法格式：turtle.begin_poly()

（8）end_poly()方法 ——记录要获取的多边形的结束点

语法格式：turtle.end_poly()

（9）get_poly()方法 ——获取最后记录的多边形

语法格式：turtle.get_poly()

返回的结果为元组格式，其元素为多边形各个顶点的坐标，如：

((0.00,0.00),(246.98,-17.10), (285.29,-49.24))

（10）write()方法 ——写文本

语法格式：turtle.write(arg, move=False, align="left", font=("Arial", 8, "normal"))

arg：要写入屏幕上的文本；

move: 绘制完文本后，画笔是否移动;

align: 保持画笔位置不动，绘制完文本后画笔与文本的对齐方式，具体为：left，文本的开头与画笔对齐；center，文本的中央与画笔对齐；right，文本的末尾与画笔对齐;

font: 描述文本格式的元组（字体，字号，字形）。

（11）setheading()方法 ——设置画笔方向

语法格式: turtle.setheading(angle)

angle: 一个表示方向的整数或小数

（12）right（）方法 ——使画笔的朝向右旋（顺时针）

语法格式: turtle.right（angle）

angle: 角度，使画笔的朝向右旋（顺时针）angle

（13）left（）方法 ——使画笔的朝向左旋（逆时针）

语法格式: turtle.left（angle）

angle: 角度，使画笔的朝向左旋（逆时针）angle

（14）goto()方法 ——移动画笔到一个绝对坐标。如果画笔已落下将会画线，不改变画笔的朝向。

语法格式: turtle.goto（x,y=None）

x: 一个数字或一个坐标，如果 y 为 None，x 必须是一个坐标。

（15）turtle.setx()方法 ——设置画笔的横坐标为 x，纵坐标保持不变。

语法格式: turtle.setx(x)

x:一个表示横坐标的整数或小数

（16）turtle.sety()方法 ——设置画笔的纵坐标为 y，横坐标保持不变。

语法格式：turtle.sety(y)

y:一个表示纵坐标的整数或小数

（17）home()方法 ——画笔移至初始坐标原点，朝向设为初始方向，如果画笔为落在状态，则会留下痕迹。

语法格式：turtle.home()

这个方法等价于 turtle.goto(0,0)和 turtle.heth(0)

（18）undo()方法 ——撤销上一次操作

语法格式：turtle.undo()

（19）stamp()方法 ——在当前位置拷贝一份画笔的形状,返回一个 stamp_id 号(画笔形状编号)

语法格式：turtle.stamp()

（20）clearstamp()方法 ——删除指定 stamp_id 的画笔形状

语法格式：turtle.clearstamp(stamp_id)

（21）clearstamps()方法 ——删除拷贝的画笔形状

语法格式：turtle.clearstamps(n=None)

当省略 n 时，删除所有已拷贝的画笔形状，当 n 大于 0 时，删除已拷贝的前 n 个画笔形状，当 n 小于 0 时，删除已拷贝的后 n 个画笔形状。

★可以通过 turtle.shape("turtle")改变画笔形状,默认有以下形状：“arrow”，“turtle”，“circle”，“square”，“triangle”，“classic”。

方法	描述
获取画笔属性	
turtle.position()	位置，别名：pos()
turtle.towards()	目标方向
turtle.xcor()	x 坐标

turtle.ycor()	y 坐标
turtle.heading()	朝向
turtle.distance()	距离
turtle.filling()	是否填充
turtle.isvisible()	是否可见

（1）position()方法 ——返回画笔当前位置坐标（元组格式）

语法格式：turtle.pos()

（2）towards()方法 ——返回画笔当前位置指向(x,y)位置连线的向量的角度。

语法格式：turtle.towards(x, y=None)

x:一个数字或一个坐标，如果 y 为 None，x 必须是一个坐标.

（3）xcor()方法 ——返回画笔的 x 坐标

语法格式：turtle.xcor()

（4）ycor()方法 ——返回画笔的 y 坐标

语法格式：turtle.ycor()

（5）heading()方法 ——返回画笔的朝向

语法格式：turtle.heading()

（6）turtle.distance()方法 ——返回从画笔当前位置到由点 (x,y)的距离

语法格式：turtle.distance(x, y=None)

x:一个数字或一个坐标，如果 y 为 None，x 必须是一个坐标

（7）filling()方法 ——返回填充状态

语法格式：turtle.filling()

当语句处于 begin_fill()与 end_fill()之间时返回 True,否则返回 False。

（8）isvisible()方法 ——返回当前画笔是否可见

语法格式：turtle.isvisible()

方法	描述

事件与其它方法	
turtle.clone()	克隆画笔
turtle.showturtle()	显示画笔，别名：st()
turtle.hideturtle()	隐藏画笔，别名：ht()
turtle.onclick()	绑定鼠标单击事件
turtle.onrelease()	绑定鼠标释放事件
turtle.ondrag()	绑定鼠标拖拽事件

（1）clone()方法 ——克隆画笔

语法格式：turtle.clone()

克隆出来的画笔与原画笔的位置、朝向、形状、颜色等属性相同。但绑定在原画笔上的事件不能克隆。

（2）turtle.showturtle()方法 ——显示画笔

语法格式：turtle.showturtle()

（3）turtle.hideturtle()方法 ——隐藏画笔

语法格式：turtle.hideturtle()

（4）onclick()方法 ——绑定鼠标单击（按下）事件

语法格式：turtle.onclick(fun, btn=1, add=None)

fun：一个带有两个分别表示鼠标点击此画笔时鼠标位置的参数的函数名称。如果 fun 值为 None，则移除现有的绑定

btn:鼠标按钮编号，默认值为 1（鼠标左键)。（**不知怎么绑定右键**）。

add:True 或 False.如为 True 则将添加一个新绑定，否则将取代先前的绑定.

（5）onrelease()方法 ——绑定释放鼠标事件

语法格式：turtle.onrelease(fun, btn=1, add=None)

参数意义同 onclick()方法

（6）ondrag()方法 ——绑定鼠标拖拽事件

语法格式：turtle.ondrag(fun, btn=1, add=None)

参数意义同 onclick()方法，**这个方法并不好用，在一次拖拽中可能会触发多次调用，可以用此事件实现用鼠标绘图功能。**

方法	描述
画笔形状	
turtle.shearfactor()	设置或返回当前的剪切因子
turtle.settiltangle()	旋转画笔形状使其指向某个角度
turtle.tiltangle()	旋转画笔形状使其指向某个角度
turtle.tilt()	旋转画笔
turtle.shapetransform（）	设置或返回画笔形状的当前变形矩阵

（1）shearfactor()方法 ——设置或返回当前的剪切因子

语法格式：shearfactor(shear=None)

shear：指定的剪切因子即剪切角度的切线来剪切画笔形状，不改变画笔的朝向。**（没有明白实际剪切思想）**

（2）settiltangle()方法 ——旋转画笔形状使其指向某个角度

语法格式：turtle.settiltangle(angle)

angle：旋转角度，旋转后画笔朝向不变，3.1 版后已移除.

（3）tiltangle()方法 ——旋转画笔形状使其指向某个角度

语法格式：turtle.tiltangle(angle)

angle：旋转角度，旋转后画笔朝向不变。

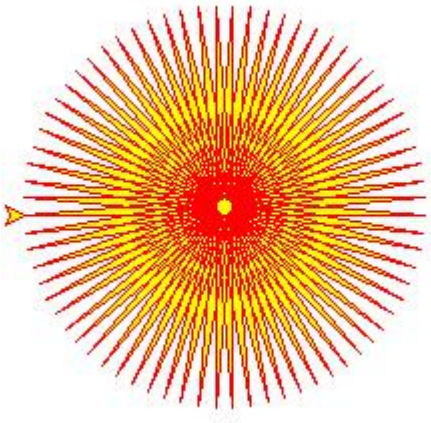
（4）tilt()方法 ——画笔形状自其当前的倾角转动 angle 指定的角度，不改变画笔的朝向

语法格式：turtle.tilt(angle)，angle：旋转角度

（5）shapetransform（）方法 ——设置或返回画笔形状的当前变形矩阵

四、实例

1. 绘制“齿轮状”图形

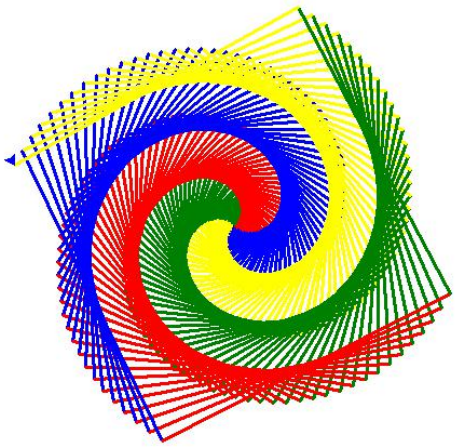


```
#_*_ coding:utf-8 _*_
#repeat   重复次数
#length   边长
#angle    偏转角度
import turtle as t
pen = t.Turtle()
pen.color("red", "yellow")
pen.speed(10)
def
drawpoly(repeat=360,length=200,angle=170):
    pen.begin_fill()
    for _ in range(repeat):
        pen.forward(length)
        pen.left(angle)
    pen.end_fill()
if   __name__ =="__main__":
   drawpoly(50,200,170)
   t.done()
```

绘图原理：乌龟爬行定长距离然后左旋一个固定角度。

偏转角度越接近 0 度，图形越接近于圆；偏转角度越接近 180 度，图形中间的“空心”越小；偏转角度大于 90 度时就能绘制出“齿轮状”图形。此程序可以绘制圆内接正 n 边形，五角星等常见图形。

2.绘制多彩“螺旋状”图形



```
#_*_ coding:utf-8 _*_
#repeat   重复次数
#angle    偏转角度
import turtle as t
pen = t.Turtle()
colors = ['red','green','yellow','blue']
pen.speed(10)
pen.pensize(3)
def drawpoly(repeat=360,angle=91):
      for i in range(repeat):
        pen.forward(i+2)
        pen.left(angle)
        pen.color(colors[i % len(colors)])
```

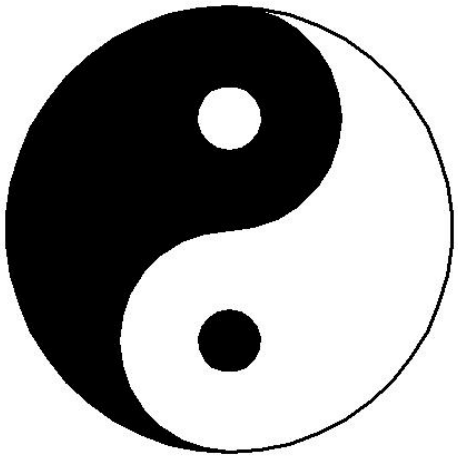
```
if   __name__ =="__main__":
        drawpoly(300,91)
        t.done()
```

绘图原理：乌龟爬行变长距离然后左旋一个固定角度，周期性地改变颜色。
由于每次绘制时，边长逐渐增加，导致图形会越来越大；只要旋转角度不等于某个正 n 边形的内角，就会形成“螺旋状”图形。当旋转角度等于某个正 n 边形的内角时，就会形成由多个相似的从小到大的多边形嵌套组成的图形。

3.绘制太极图

黑色：阴；白色：阳；太阴：多的黑色部分；少阳：少的白色的圆

```
import turtle as t
#radius 直径，pen_color 画笔颜色，filling_color 填充颜色
pen = t.Turtle()
t.hideturtle()
def   draw(radius,pen_color,filling_color):
        pen.pendown()
        pen.color('black',pen_color)
        pen.begin_fill()
        pen.circle(radius/2,180)
        pen.circle(radius,180)
        pen.circle(radius/2,-180)
        pen.end_fill()

        pen.right(90)
        pen.penup()
        pen.forward(radius*0.5)
        pen.dot(radius*0.3,filling_color)
```



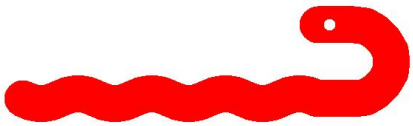
```
if __name__ =="__main__":
          draw(200,'black','white')
          pen.penup()
          pen.home()
          pen.seth(180)
          draw(200,'white','black')
          pen.home()
```

绘图原理：太极图的太阴和太阳部分可以分解为两个半径相等的小半圆和一个 2 倍于小圆的半径的大半圆组成。

4.绘制蟒蛇

```
import turtle as t
t.hideturtle()
```

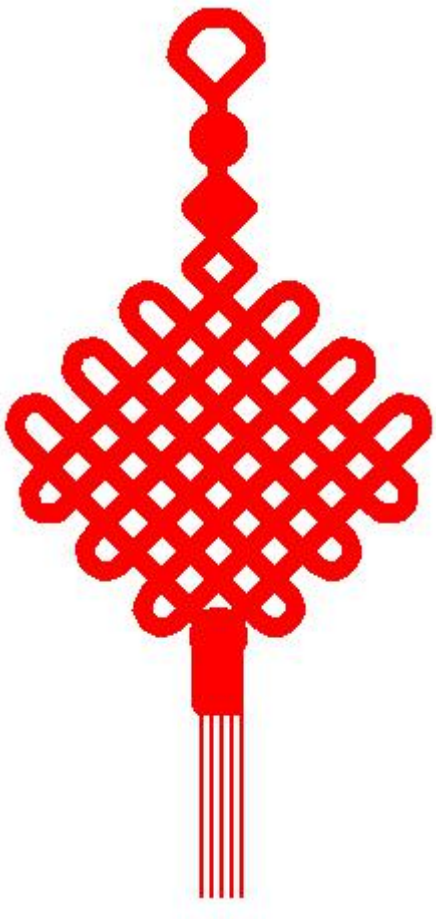
```
def draw(pensize=30,color='green',angle=30):
    t.penup()
    t.pensize(pensize)
    t.color(color)
    t.backward(pensize*5)
    t.pendown()
    t.setheading(-angle)
    for _ in range(4):
        t.circle(pensize,angle*2)
        t.circle(-pensize,angle*2)
    t.setheading(0)
    t.forward(50)
    t.circle(pensize,180)
    t.forward(pensize)
    t.backward(pensize/3)
    t.dot(pensize/3,'white')

if __name__=="__main__":
    draw(50,'red',30)
```



绘图原理：核心部分是绘制蛇身，画笔的起始方向和圆弧的度数。

5.绘制中国结

```
import turtle as t
t.hideturtle()
def draw(pensize=10,color='red'):
        t.pensize(pensize)
        t.color(color,color)
        t.seth(90)
        t.penup()
        t.forward(200)
        t.pendown()
        t.seth(45)
        t.forward(30)
        t.seth(90)
        t.circle(20,180)
        t.seth(-45)
        t.forward(30)
        t.seth(-90)
        t.forward(20)
        t.dot(30)
        t.forward(20)
        t.seth(-45)
```



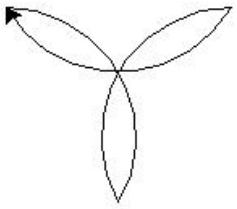
```
t.begin_fill()
for _    in range(4):
            t.forward(20)
            t.right(90)
t.end_fill()

t.seth(-90)
t.forward(30)
cur_pos = t.pos()
t.seth(225)
t.forward(20)
t.left(90)
for _ in range(3):
          t.forward(150)
          t.circle(-10,180)
          t.forward(150)
          t.circle(10,180)
t.forward(140)
t.penup()
t.goto(cur_pos[0],cur_pos[1])
t.pendown()
t.seth(-45)
t.forward(20)
t.right(90)
for _ in range(3):
          t.forward(150)
          t.circle(10,180)
          t.forward(150)
          t.circle(-10,180)
t.forward(140)

t.seth(-90)
t.dot(30)
t.forward(5)
t.seth(0)
t.begin_fill()
t.forward(8)
t.right(90)
t.forward(30)
t.right(90)
t.forward(16)
t.right(90)
```

```
        t.forward(35)
        t.end_fill()

        t.pensize(2)
        t.seth(-90)
        t.forward(30)
        cur_pos = t.pos()

        for i in    range(5):
            t.pendown()
            t.goto(cur_pos[0]+5*i,cur_pos[1]-100)
            t.penup()
            t.goto(cur_pos[0]+5*(i+1),cur_pos[1])


if __name__ =="__main__":
        draw(10,'red')
```

绘图原理：分解形状，具有重复性质的使用循环语句绘制，没有的逐一绘制。

6.绘制多段圆弧组成封闭图形

```
import turtle as t
#radius 半径，当半径或 angle 小于 0 时，图形呈凹状
#number   基本图形个数
#angle 每段弧对应的圆心角的度数
t.speed(10)
def cirarc(radius,number,angle):
      for i in range(number):
            t.setheading(i * 360/number)
            t.circle(radius,angle)

if __name__ =="__main__":
      cirarc(-80,3,120)
```

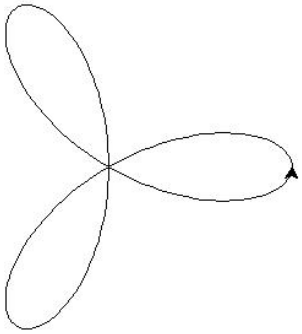


绘图原理：核心代码是 t.setheading((i * 360/number))，保证均匀分布在一个圆内，终点与起点重合。

★不同的参数可得到不同的图像，number 和 angle 是决定图形样式的主要参数。当 number,angle 相同时，不同 radius，得到的是相似形，radius 越大，图像越大；

7.根据极坐标方程绘图

```
import turtle as t
import math
```



```
def   drawpoint(r,angle,isfirst):
      radian = math.radians(angle)
      x = r*math.cos(radian)
      y = r*math.sin(radian)
      direction = t.towards(x,y)
      t.seth(direction)
      t.penup() if isfirst else    t.pendown()
      t.goto(x,y)
def   drawgraphical(number,theta):
      for angle in range(theta+1):
            radian = math.radians(angle)
            #根据玫瑰曲线极坐标方程计算极径
            r = 150*math.cos(number*radian)
            isfirst = True    if not angle else False
            drawpoint(r,angle,isfirst)
if __name__ =="__main__":
      drawgraphical(3,360)
```

绘图原理：描点法。可用此代码画出所有极角从 0 开始，具有显式表达的极坐标方程的曲线图形。