Python绘图Turtle库详解

知行流浪 于 2017-08-02 17:36:45 发布

Turtle库是Python语言中一个很流行的绘制图像的函数库，想象一个小乌龟，在一个横轴为x、纵轴为y的坐标系原点，(0,0)位置开始，它根据一组函数指令的控制，在这个平面坐标系中移动，从而在它爬行的路径上绘制了图形。

turtle绘图的基础知识：

1. 画布(canvas)

画布就是turtle为我们展开用于绘图区域，我们可以设置它的大小和初始位置。

设置画布大小

turtle.screensize(canvwidth=None, canvheight=None, bg=None)，参数分别为画布的宽(单位像素), 高, 背景颜色。

如：turtle.screensize(800,600, "green")

turtle.screensize() #返回默认大小(400, 300)

turtle.setup(width=0.5, height=0.75, startx=None, starty=None)，参数：width, height: 输入宽和高为整数时, 表示像素; 为小数时, 表示占据电脑屏幕的比例，(startx, starty)这一坐标表示矩形窗口左上角顶点的位置, 如果为空,则窗口位于屏幕中心。

如：turtle.setup(width=0.6,height=0.6)

turtle.setup(width=800,height=800, startx=100, starty=100)

2. 画笔

2.1 画笔的状态

在画布上，默认有一个坐标原点为画布中心的坐标轴，坐标原点上有一只面朝x轴正方向小乌龟。这里我们描述小乌龟时使用了两个词语：坐标原点(位置)，面朝x轴正方向(方向)， turtle绘图中，就是使用位置方向描述小乌龟(画笔)的状态。

2.2 画笔的属性

画笔(画笔的属性，颜色、画线的宽度等)

1) turtle.pensize()：设置画笔的宽度；

2) turtle.pencolor()：没有参数传入，返回当前画笔颜色，传入参数设置画笔颜色，可以是字符串如"green", "red",也可以是RGB 3元组。

3) turtle.speed(speed)：设置画笔移动速度，画笔绘制的速度范围[0,10]整数，数字越大越快。

2.3 绘图命令

操纵海龟绘图有着许多的命令，这些命令可以划分为3种：一种为运动命令，一种为画笔控制命令，还有一种是全局控制命令。

(1) 画笔运动命令

命令	说明
turtle.forward(distance)	向当前画笔方向移动distance像素长度
turtle.backward(distance)	向当前画笔相反方向移动distance像素长度
turtle.right(degree)	顺时针移动degree°
turtle.left(degree)	逆时针移动degree°
turtle.pendown()	移动时绘制图形，缺省时也为绘制
turtle.goto(x,y)	将画笔移动到坐标为x,y的位置
turtle.penup()	提起笔移动，不绘制图形，用于另起一个地方绘制
turtle.circle()	画圆，半径为正(负)，表示圆心在画笔的左边(右边)画圆
setx( )	将当前x轴移动到指定位置

sety( )	将当前y轴移动到指定位置
setheading(angle)	设置当前朝向为angle角度
home()	设置当前画笔位置为原点，朝向东。
dot(r)	绘制一个指定直径和颜色的圆点

(2) 画笔控制命令

命令	说明
turtle.fillcolor(colorstring)	绘制图形的填充颜色
turtle.color(color1, color2)	同时设置pencolor=color1, fillcolor=color2
turtle.filling()	返回当前是否在填充状态
turtle.begin_fill()	准备开始填充图形
turtle.end_fill()	填充完成
turtle.hideturtle()	隐藏画笔的turtle形状
turtle.showturtle()	显示画笔的turtle形状

(3) 全局控制命令

命令	说明
turtle.clear()	清空turtle窗口，但是turtle的位置和状态不会改变
turtle.reset()	清空窗口，重置turtle状态为起始状态
turtle.undo()	撤销上一个turtle动作
turtle.isvisible()	返回当前turtle是否可见
stamp()	复制当前图形
turtle.write(s [,font=("font-name",font_size,"font_type")])	写文本，s为文本内容，font是字体的参数，分别为字体名称，大小和类型；font为可选项，font参数也是可选项

(4) 其他命令

<table>
<tr><td>命令</td><td>说明</td></tr>
<tr><td>turtle.mainloop()或turtle.done()</td><td>启动事件循环 -调用Tkinter的mainloop函数。
必须是乌龟图形程序中的最后一个语句。</td></tr>
<tr><td>turtle.mode(mode=None)</td><td>设置乌龟模式（“standard”，“logo”或“world”）并执行重置。如果没有给出模式，则返回当前模式。
<table>
<tr><td>模式</td><td>初始龟标题</td><td>正角度</td></tr>
<tr><td>standard</td><td>向右（东）</td><td>逆时针</td></tr>
<tr><td>logo</td><td>向上（北）</td><td>顺时针</td></tr>
</table></td></tr>
<tr><td>turtle.delay(delay=None)</td><td>设置或返回以毫秒为单位的绘图延迟。</td></tr>
<tr><td>turtle.begin_poly()</td><td>开始记录多边形的顶点。当前的乌龟位置是多边形的第一个顶点。</td></tr>
<tr><td></td><td></td></tr>
</table>

turtle.end_poly()	停止记录多边形的顶点。当前的乌龟位置是多边形的最后一个顶点。将与第一个顶点相连。
turtle.get_poly()	返回最后记录的多边形。

3. 命令详解

3.1 turtle.circle(radius, extent=None, steps=None)

描述：以给定半径画圆

参数：

radius(半径)：半径为正(负)，表示圆心在画笔的左边(右边)画圆；

extent(弧度) (optional)；

steps (optional) (做半径为radius的圆的内切正多边形，多边形边数为steps)。

举例:

circle(50) # 整圆;

circle(50,steps=3) # 三角形;

circle(120, 180) # 半圆

实例：

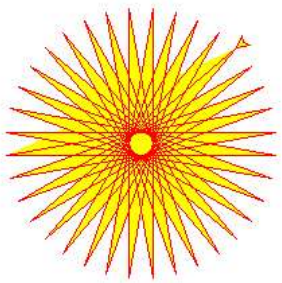
1、太阳花

```
# coding=utf-8
import turtle
import time

# 同时设置pencolor=color1, fillcolor=color2
turtle.color("red", "yellow")

turtle.begin_fill()
for _ in range(50):
turtle.forward(200)
turtle.left(170)
turtle.end_fill()

turtle.mainloop()
```



2、五角星

```
# coding=utf-8
import turtle
import time
```

```
  5 turtle.pensize(5)
turtle.pencolor("yellow")
turtle.fillcolor("red")

turtle.begin_fill()
for _ in range(5):
  turtle.forward(200)
  turtle.right(144)
turtle.end_fill()
time.sleep(2)

turtle.penup()
turtle.goto(-150,-120)
turtle.color("violet")
turtle.write("Done", font=('Arial', 40, 'normal'))

turtle.mainloop()
```

3、时钟程序

```
# coding=utf-8

import turtle
from datetime import *

# 抬起画笔，向前运动一段距离放下
def Skip(step):
    turtle.penup()
    turtle.forward(step)
    turtle.pendown()

def mkHand(name, length):
    # 注册Turtle形状，建立表针Turtle
    turtle.reset()
    Skip(-length * 0.1)
    # 开始记录多边形的顶点。当前的乌龟位置是多边形的第一个顶点。
    turtle.begin_poly()
    turtle.forward(length * 1.1)
    # 停止记录多边形的顶点。当前的乌龟位置是多边形的最后一个顶点。将与第一个顶点相连。
    turtle.end_poly()
    # 返回最后记录的多边形。
    handForm = turtle.get_poly()
    turtle.register_shape(name, handForm)

def Init():
    global secHand, minHand, hurHand, printer
    # 重置Turtle指向北
    turtle.mode("logo")
    # 建立三个表针Turtle并初始化
    mkHand("secHand", 135)
    mkHand("minHand", 125)
    mkHand("hurHand", 90)
    secHand = turtle.Turtle()
    secHand.shape("secHand")
    minHand = turtle.Turtle()
```

```
    minHand.shape("minHand")
    hurHand = turtle.Turtle()
    hurHand.shape("hurHand")

    for hand in secHand, minHand, hurHand:
        hand.shapesize(1, 1, 3)
        hand.speed(0)

    # 建立输出文字Turtle
    printer = turtle.Turtle()
    # 隐藏画笔的turtle形状
    printer.hideturtle()
    printer.penup()

def SetupClock(radius):
    # 建立表的外框
    turtle.reset()
    turtle.pensize(7)
    for i in range(60):
        Skip(radius)
        if i % 5 == 0:
            turtle.forward(20)
            Skip(-radius - 20)

            Skip(radius + 20)
            if i == 0:
                turtle.write(int(12), align="center", font=("Courier", 14, "bold"))
            elif i == 30:
                Skip(25)
                turtle.write(int(i/5), align="center", font=("Courier", 14, "bold"))
                Skip(-25)
            elif (i == 25 or i == 35):
                Skip(20)
                turtle.write(int(i/5), align="center", font=("Courier", 14, "bold"))
                Skip(-20)
            else:
                turtle.write(int(i/5), align="center", font=("Courier", 14, "bold"))
            Skip(-radius - 20)
        else:
            turtle.dot(5)
            Skip(-radius)
        turtle.right(6)

def Week(t):
    week = ["星期一", "星期二", "星期三",
            "星期四", "星期五", "星期六", "星期日"]
    return week[t.weekday()]

def Date(t):
    y = t.year
    m = t.month
    d = t.day
    return "%s %d%d" % (y, m, d)

def Tick():
    # 绘制表针的动态显示
    t = datetime.today()
    second = t.second + t.microsecond * 0.000001
    minute = t.minute + second / 60.0
    hour = t.hour + minute / 60.0
    secHand.setheading(6 * second)
    minHand.setheading(6 * minute)
    hurHand.setheading(30 * hour)

    turtle.tracer(False)
    printer.forward(65)
    printer.write(Week(t), align="center",
                  font=("Courier", 14, "bold"))
    printer.back(130)
    printer.write(Date(t), align="center",
                  font=("Courier", 14, "bold"))
```

```
    printer.home()
    turtle.tracer(True)

    # 100ms后继续调用tick
    turtle.ontimer(Tick, 100)

def main():
    # 打开/关闭龟动画，并为更新图纸设置延迟。
    turtle.tracer(False)
    Init()
    SetupClock(160)
    turtle.tracer(True)
    Tick()
    turtle.mainloop()

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