海龟画图

绘制圆形：

首先我们想要使用东东来画画的话，python自带的内值函数没有这个功能的，那怎么办呢，还好我们有一个第三方库叫做turtle，他的作用就是召唤小海龟出来散散步

那我们想让这个龟出来走走该怎么做呢？

很简单我们想导入这个龟龟

```
import turtle as t
```

tips: 有些时候我们要调用这个库的时候懒得打完全称，就使用 as 在后面添加你自己想偷懒的输入

现在我想想让这个龟跑一圈我该怎么做捏？

```
t.circle(10)
```

哦哦哦！就这么简单一个圆圈就画好了

tips:

如果参数为负数，表示圆心在画笔的右边，同理，正数在右边

那现在我想让这龟多跑两圈我该怎么做了？

控制方向：

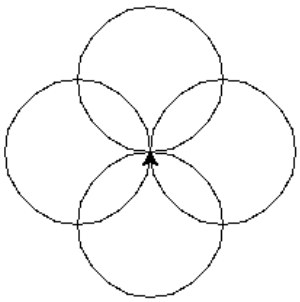
我们要知道，这只龟龟默认是向右走的，那我们不能每次都让他从右边开始走吧，但我们想给他换个方向的时候该怎么办呢？

我现在想要让龟 ⬅ ⬆ ⬇ 该怎么实现呢？

很简单，你就告诉他向左/右转多少度

```
t.left(90) and t.right(90) and t.left(180) or t.right(180)
```

那我现在想画一个如图的图片我们该怎么做呢？



那当然我们可以改变参数，让他画出很多很多非常奇特的东西出来

tips：但我们画图的速度很慢的时候，我们可以使用speed(0)给他加速

```
t.speed(10)
```

1----10

1最慢，10最快，5默认

接下来就开动你的脑瓜子来创作吧

```
import turtle as t
import random
t.speed(0)

for i in range(10000000):
    t.right(random.randint(100, 1000))
    t.circle(random.randint(1, 100))

t.done()
```

tips: t.done()表示程序已经结束了，如果不加，程序会直接退出

直走：

我们会让这个龟转圈，换方向，那我们怎么让他走直线呢？

```
t.forward(steps)
```

我们来画一个图试试：

```
import turtle as t

t.forward(100)
t.done()
```

很好一条直线出来了，那我们试试画一个正方形：

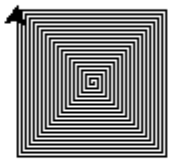
```
import turtle as t

for i in range(4):
    t.forward(100)
    t.left(90)
t.done()
```

正方形会画了，正三角形呢？

```
import turtle as t
for i in range(3):
    t.forward(100)
    t.left(120)
t.done()
```

题目：画回形状



```
import turtle as t

for i in range(100):
    t.fd(i)
    t.right(90)
```

换颜色：

当然如果我们不还颜色的话，看起来还是单调了一些，所以我们需要换个颜色看看，搞个彩虹🌈也不是不行

```
t.color("red")
t.color("blue")
t.color("green")
```

color参数（画笔颜色，画笔的填充颜色）

已被定义的颜色表（若需要不同的颜色，需要去查询RGB值）

- 红色 : red
- 橙色 : orange
- 绿色 : grenn
- 蓝色: blue
- 紫色 : purple
- 白色 : white
- 黑色 : black
- 灰色 : grey
- 粉色 : pink
- 金色 : gold
- 银色 : silver

那么笔的颜色会换了，那么我们该如何设置画布颜色呢?

```
t.screensize(10, 10, "red")
```

那么我们该怎么设置画笔的起始坐标呢?

tips：计算机的（0，0）坐标在屏幕的左上角

```
t.setup(800, 600, 0, 0)
# 可以设置画布的宽高，还可以设置x，y坐标
```

如果我们想要把画好的图形填上颜色我们该怎么做呢?

```
t.begin_fill()
t.end_fill()
```

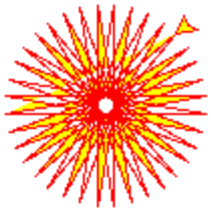
当然这个填充不是边画边填充的，而是在图形画完以后才开始填充的

填充的颜色为设置的画笔颜色的填充值

并且我们需要将画图的指令夹在他两之中

```
import turtle as t
t.color("red", "blue")
t.begin_fill()
for i in range(4):
    t.fd(100)
    t.rt(90)
t.end_fill()
t.done()
```

题目：花一个太阳花



```
import turtle as t

t.color("red", "yellow")

t.begin_fill()
for i in range(50):
    t.fd(100)
    t.left(170)

t.end_fill()

t.done()
```

进阶：

`goto()`

```
turtle.goto(x, y=None)
turtle.setpos(x, y=None)
turtle.setposition(x, y=None)
```

参数:
x： 一个数或一个坐标
y： 一个数或者为空

在turtle 绘制的区域，是一个2D平面，这个平面由坐标定位，在turtle中使用pos表示坐标，初始位置为(0,0)，使用goto传入坐标将会跳转至该坐标点并且绘制

```
from turtle import *

goto(30,30)
done()
```

`setx()/sety()`

```
`turtle.setx(x)`
参数: x: 一个数字 (整数或者浮点)
设置x坐标的值，y坐标不变
`turtle.sety(y)`
参数: y ： 一个数字(整数或者浮点)
设置y坐标的值，x坐标不变
```

代码实例:

```
from turtle import *

setx(10)
done()
```

`setheading() | seth()`

```
turtle.setheading(to_angle)
turtle.seth(to_angle)
```

参数:

- to_angle：一个整数（整数或者浮点）

使用 setheading 将会把turtle初始状态设置为0°，类似于一个量角器，初始状态下turtle是个垂直于一条平面的线，并没有角度，所以为0，如果使用 setheading 传入参数 45 后，将会方向指向相对于初始状态的45°

```
from turtle import *

setheading(45)
done()
```

`home`

```
turtle.home()
```

将turtle移动到原点坐标(0,0)-并将其航向设置为起始方向

```
from turtle import *

forward(80)
home()
input()
```

`home()`：将turtle至于初始坐标

`circle()`

参数:

- radius： 数字 半径
- extent： 数字 范围 可以为空
- steps： 整数 可以为空

使用circle可以画出一个圆，或者弧度，第一个参数为半径，第二个参数可以控制绘制的范围多少，如果输入90，那么只绘制到90°位置；steps为阶梯，圆的边其实是锯齿状的，相当于像素点的感觉，一下代码将会作出对比，方便理解

画出一个圆代码示例:

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

加上一个参数extent进行绘制:

```
from turtle import *

circle(120,180)
# 180度
input()
```

第三个参数不作介绍

`dot()`

```
turtle.dot(size=None, *color)
```

参数:

size： 大于0的整数
color： 颜色标识 用一个指定颜色绘制一个尺寸为size的原点。
代码示例:

```
from turtle import *

dot(20,"blue")
input()
```

`stamp()`

`turtle.stamp()`

作用：复制一个turtle

```
from turtle import *

stamp()
color("blue")
goto(100,0)
input()
```

代码释义：
stamp()：复制一个自己；
color("blue")：设置当前的自己为蓝色；
goto(100,0)：移动到右边x为100，y为0处。

为了区别颜色一个为蓝一个就是初始颜色了

`clearstamp()`

```
turtle.clearstamp(stampid)
```

参数:

- stampid ： 使用 stamp 复制一个自己后将会返回一个 stamp的id值，传入id值即可删除复制的对象

代码示例:

```
from turtle import *

turtle2=stamp()
color("blue")
goto(100,0)
clearstamp(turtle2)
input()
```

代码释义:
`turtle2=stamp()`：接收返回的stampid
`clearstamp(turtle2)`：删除复制的turtle

笔控制:

`penup() | pu() | up()`

```
turtle.penup()
turtle.pu()
turtle.up()
```

作用：移动时不绘制

代码示例：

```
from turtle import *
import time

penup()
speed('normal')#也可以使用0代替
circle(100)
input()
```

`pendown() | pd() | down()`

```
turtle.pendown()
turtle.pd()
turtle.down()
```

作用：移动时绘制

```
from turtle import *
import time

penup()
circle(100)
pendown()
circle(100)
input()
```

`pensize() | width()`

```
turtle.pensize(width=None)
turtle.width(width=None)
```

参数：

width：线条宽度，可以为空
设置线条宽度，或将其返回。
代码示例：

```
from turtle import *
import time

pensize(10)
circle(100)
input()
```

代码释义：
pensize(10)：设置代码的宽度，使用width方法结果相同

`pen()`

```
turtle.pen(pen=None, **pendict)
```

参数:

pen – 有特定标识值
pendict： 有多个关键字参数
设置pen的相关方法。
以下为官方给出的设置值:

- `“shown”`: True/False
- `“pendown”`: True/False
- `“pencolor”`: color-string or color-tuple
- `“fillcolor”`: color-string or color-tuple
- `“pensize”`: positive number
- `“speed”`: number in range 0...10
- `“resizemode”`: “auto” or “user” or “noresize”
- `“stretchfactor”`: (positive number, positive number)
- `“outline”`: positive number
- `“tilt”`: number

可在一条语句中设置多个属性。
代码示例:

```
from turtle import *
import time

pen(speed=10, pencolor="red", pensize=10)
circle(100)
input()
# pen(speed=10, pencolor="red", pensize=10)：设置pen（笔）的绘制速度为10，颜色为红色，线条大小为10.
```

`isdown()`

```
turtle.isdown()
```

判断比是否抬起或放下，换句话解释就是判断是否移动时绘制，如果绘制返回 True 否则返回 False。
代码示例:

```
from turtle import *
import time

pen(speed=10, pencolor="red", pensize=10)
circle(100)
print(isdown())
penup()
pen(speed=10, pencolor="red", pensize=10)
circle(100)
print(isdown())
input()
```

代码释义：
print(isdown())：在默认情况下是绘制的，输出笔是否放下，输出Ture
penup()：使用penup()抬起笔，移动时不会只，此时输出False

颜色控制

`pencolor()`

```
turtle.pencolor(*args)
```

返回当前线条设置的颜色或设置线条的颜色。
pencolor可传参与不传参：

- `pencolor()`:不传参返回当前颜色的设置
- `pencolor(colorstring)`：可以传入字符串设置颜色值
- `pencolor((r, g, b))`：传入一个元组值进行设置
- `pencolor(r, g, b)`：直接赋予r、g、b值

代码示例：

```
from turtle import *
import time

pencolor("brown")
circle(100)
input()
```

代码释义：
`pencolor("brown")`：设置颜色后画圆
也可以使用`RGB`传入颜色：

```
from turtle import *
import time

tup = (0.2, 0.8, 0.55)
pencolor(tup)
circle(100)
input()
```

`fillcolor()`

```
turtle.fillcolor(*args)
```

返回设置的颜色或设置turtle的颜色。

fillcolor可传参与不传参：

- `fillcolor()`:不传参返回当前颜色的设置
- `fillcolor(colorstring)`:可以传入字符串设置颜色值
- `fillcolor((r, g, b))`：传入一个元组值进行设置
- `fillcolor(r, g, b)`：直接赋予r、g、b值

`color()`

```
turtle.color(*args)
```

返回设置笔的颜色或设置填充颜色。
color可传参与不传参：

- `color()`:不传参返回当前颜色的设置
- `color(colorstring)`：可以传入字符串设置颜色值
- `color(colorstring), color((r,g,b))`：传入2个rgb值一个设置turtle颜色一个设置绘制线条颜色
- `color(colorstring1, colorstring2)`:传入2个字符串值一个设置turtle颜色一个设置绘制线条颜色
  代码示例：

```
from turtle import *
import time

color("red", "blue")
circle(100)
input()
```

代码释义：
color("red", "blue")：设置线条为红色，turtle为蓝色
运行结果：

`begin_fill()/end_fill()`

```
turtle.begin_fill()
```

开始填充颜色。

```
turtle.end_fill()
```

结束填充颜色。
代码示例：

```
from turtle import *
import time

color("black", "red")
begin_fill()
circle(100)
end_fill()
input()
```

代码释义：
begin_fill()：开始填充，这个需要放在图形绘制之前。
end_fill()：结束绘制，一定要加不然不会进行填充。

更多绘制控制

`reset()`

```
turtle.reset()
```

从屏幕上删除turtle，并将海龟重新居中，重置所有值。
代码示例：

```
from turtle import *
import time

fd(100)
left(30)
fd(100)
time.sleep(1)
reset()
input()
```

代码释义：
time.sleep(1)：停止一秒对比效果
reset()：重置

`clear()`

```
turtle.clear()
```

清除屏幕绘制线条。并不影响turtle位置。
代码示例：

```
from turtle import *
import time

fd(100)
left(30)
fd(100)
time.sleep(1)
clear()
input()
```

代码释义：
clear()：清除以上绘制效果
运行结果：

`write()`

```
turtle.write(arg, move=False, align="left", font=("Arial", 8, "normal"))
```

参数:

- arg – 你想输出到屏幕的值
- move – True/False 输出值时是否移动
- align – 对齐方式
- font – 字体样式类别

输出值到屏幕上，可以设置字体以及对齐方式。
代码示例：

```
from turtle import *
import time

fd(100)
left(30)
fd(100)
time.sleep(1)
write("你好呀", False, align="center")
input()
```

代码释义：
write("你好呀", False, align="center")：显示你好呀，但是在显示值的时候不会移动，对其方式为居中对齐。

把write("你好呀", False, align="center")改为write("你好呀", True, align="center")，使输出值时turtle移动。

Turtle 状态

`hideturtle()|showturtle()|isvisible()`

```
turtle.hideturtle()
turtle.ht()
```

设置turtle是否可见。
代码示例：

```
from turtle import *
import time

hideturtle()
input()
```

代码释义：
hideturtle()：使turtle不可见

`showturtle()`

```
turtle.showturtle()
turtle.st()
```

使turtle可见。
代码示例：

```
from turtle import *
import time

hideturtle()
time.sleep(1)
showturtle()
input()
```

代码释义：
showturtle()：设置turtle可见。

`turtle.isvisible()`
判断turtle是否显示
代码示例：

```
from turtle import *
import time

hideturtle()
print(isvisible())
time.sleep(1)
showturtle()
print(isvisible())
input()
```

代码释义：
print(isvisible())：读取状态，显示为True否则为False