C程序设计基础
——海龟画图

林大 经管学院 瞿华

海龟画图

一. 海龟画图简介

二. 基本命令

三. 循环与基本图形

一、简介

一、简介

- 在海龟作图中，我们用程序控制小海龟在屏幕上“爬行”，海龟留下的痕迹就构成了各种几何图形。

二、基本命令

- python的基本库中自带turtle模块。但是该模块看起来不够美观。
- 本课程使用easygraphics.turtle模块来进行海龟绘图。
  - 需要预先用pip安装easygraphics模块
- 在程序或交互式环境中，首先导入easygraphics.turtle模块：

```
from easygraphics.turtle import *
```

2.1 程序初始化和控制命令

- create_world(width,height)
  - 打开大小为width*height的绘图窗口
- close_world()
  - 关闭和清理海龟作图窗口
- set_speed(x)
  - 设置海龟的爬行速度，1为最慢
- set_immediate(x)
  - 设置是否显示动画。0为显示，1为不显示。缺省为显示动画
- pause()
  - 程序等待用户按任意键

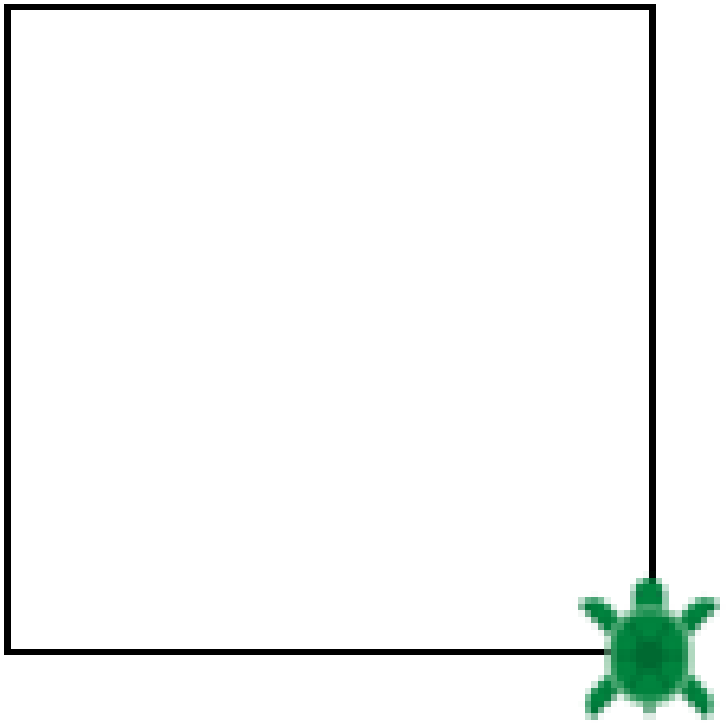
2.2 基本运动命令

- fd(x)
- forward(x)
  - 向前爬行x个像素（x可以为小数）
- bk(x)
- backward(x)
  - 向后爬行x个像素
- lt(degree)
- left_turn(degree)
  - 海龟向左转degree度（degree可以为小数）
- rt(degree)
- right_turn(degree)海龟向右转degree度
- home()
  - 海龟返回起始点（屏幕正中央）

2.2.1 简单示例

绘制正方形

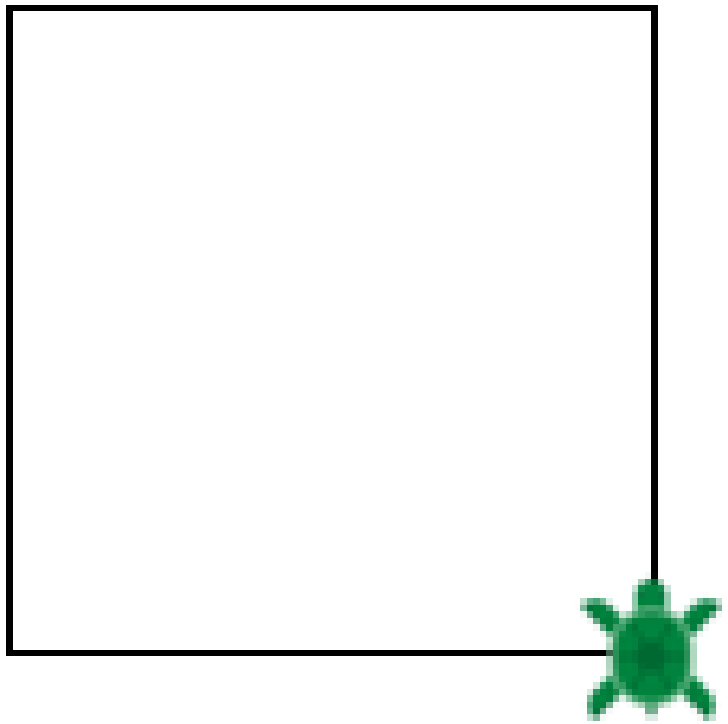
```
fd(100)
lt(90)
fd(100)
lt(90)
fd(100)
lt(90)
fd(100)
lt(90)
```



2.2.1 简单示例

绘制正方形（使用循环）

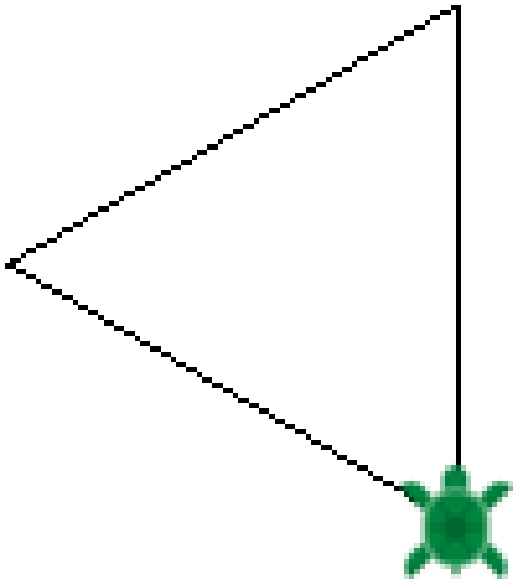
```
for i in range(4):
    fd(100)
    lt(90)
```



2.2.2 三角形绘制

- 绘制三角形，思考：
  - 每次绘制完一条边后，海龟应该转多少度？
  - 应循环多少次？

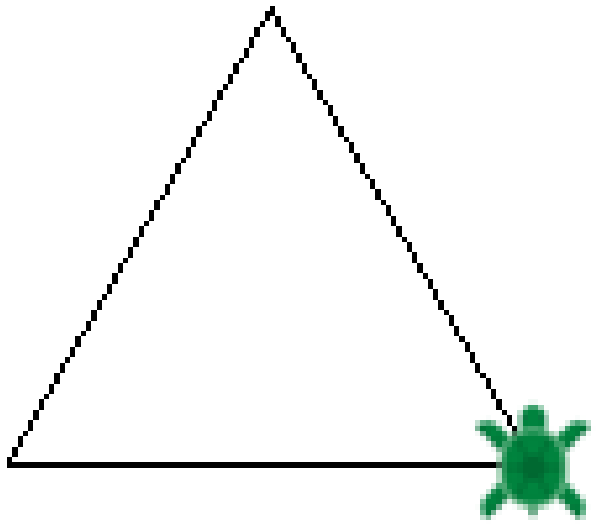
```
for i in range(3):
    fd(100)
    lt(120)
```



2.2.2 三角形绘制

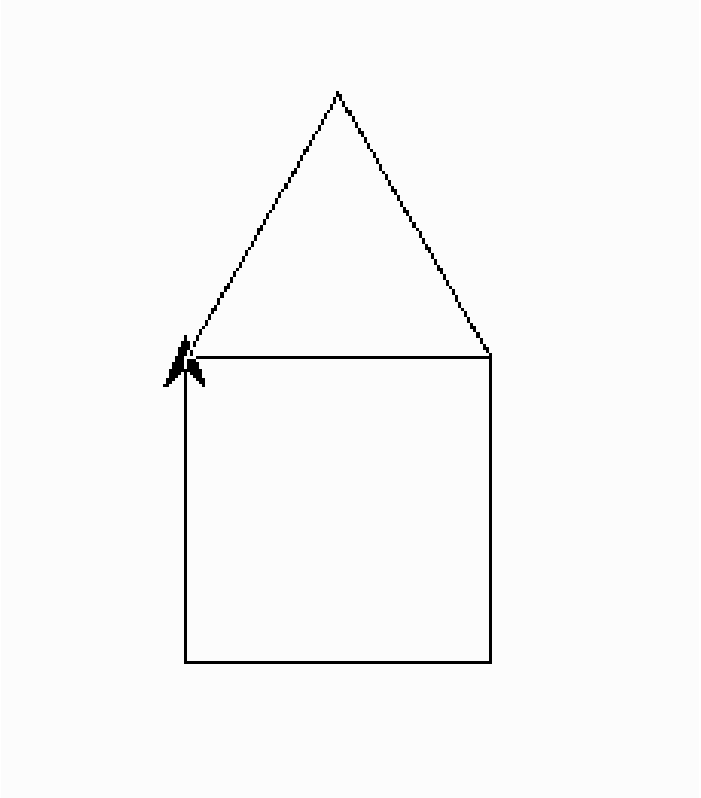
- 思考（见下图）：
  - 如果想绘制一个底边和窗口下沿平行的三角形，该怎么办？
  - 希望画完图后，海龟的头部还是指向正上方，该怎么办？

```
lt(30)
for i in range(3):
    fd(100)
    lt(120)
rt(30)
```



2.2.3 复合图形

- 思考：想绘制如下的图形，该怎么办？

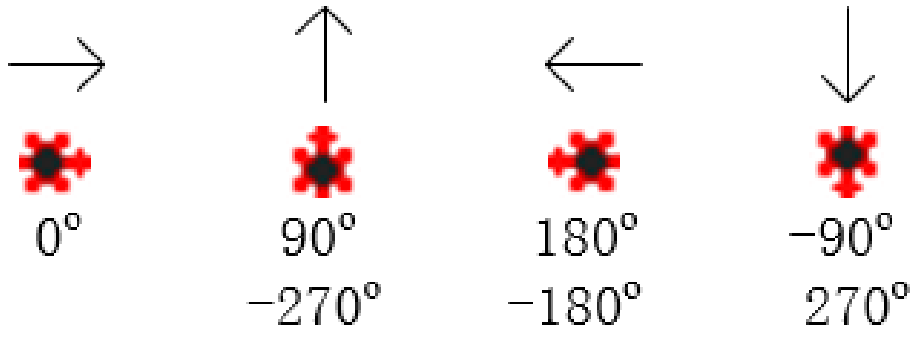


2.3 画笔相关命令

- 画笔相关：
  - cs()
  - clear_screen()
    - 海龟原地不动，清除屏幕上的所有痕迹
  - pen_up()
  - pu()
    - 抬笔，此时海龟爬行不留痕迹
  - pd();
  - pen_down();
    - 落笔，此时海龟爬行留痕迹
  - hide();
    - 隐藏海龟
  - show();
    - 显示海龟

2.4 方向与坐标系设定

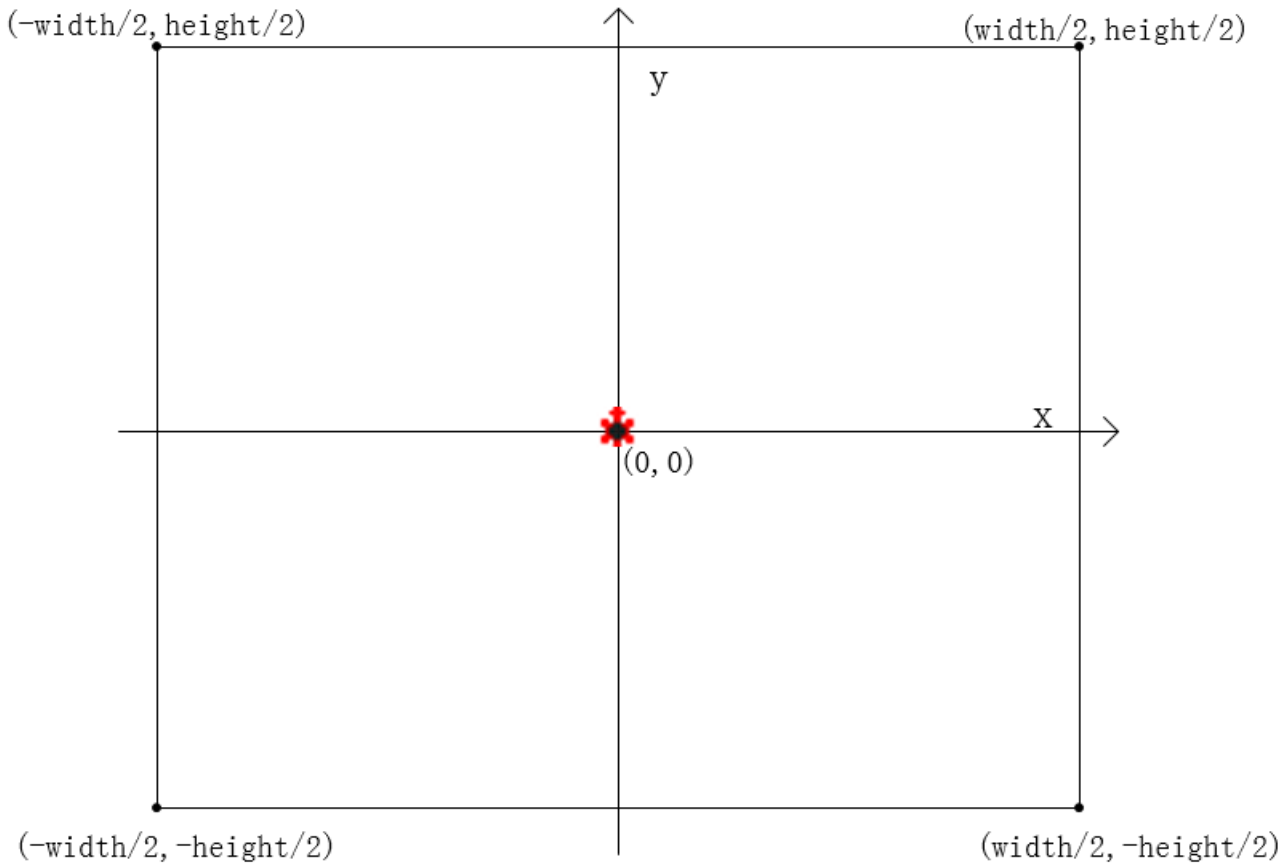
海龟朝向



规定：　海龟朝向屏幕右侧为0度
朝向屏幕正上方为90度或-270度
朝向屏幕正下方为-90度或270度
朝向屏幕左侧为180度

2.4 方向与坐标系设定

坐标系定义



采用常用的平面直角坐标系，x轴正向指向屏幕右侧，y轴正向指向屏幕正上方
窗口正中为原点（0,0）

2.5 颜色与填充

- 以下函数在使用前需要先导入easygraphics包
- 可以使用颜色名或rgb格式颜色值指定颜色
- set_color()
  - 设置画笔颜色
- set_fill_color()
  - 设置填充颜色
- set_background_color()
  - 设置背景色

2.5.1 常见颜色

	black	黑色		lightgray	浅灰色
	blue	蓝色		gray	深灰色
	green	绿色		lightblue	浅蓝色
	cyan	青色		lightgreen	浅绿色
	red	红色		lightcyan	浅青色
	magenta	紫色		lightpink	浅红色
	brown	棕色		lightmagenta	浅紫色
	yellow	黄色		whie	白色

2.6 方向与坐标命令

- setxy(x,y)
  - 海龟瞬移到点(x,y), 方向不变，不留痕迹
- set_heading(angle)
  - 海龟朝向设为angle度
- gotoxy(x,y)
  - 海龟移动到点(x,y)
- get_x();
- get_y();
  - 获取海龟当前位置的x坐标和y坐标
- get_heading(angle)
  - 获取海龟当前朝向
- turn_to(angle)
  - 旋转海龟指向angle度
- facing(x,y)
  - 旋转海龟指向点(x,y)

2.7 其他

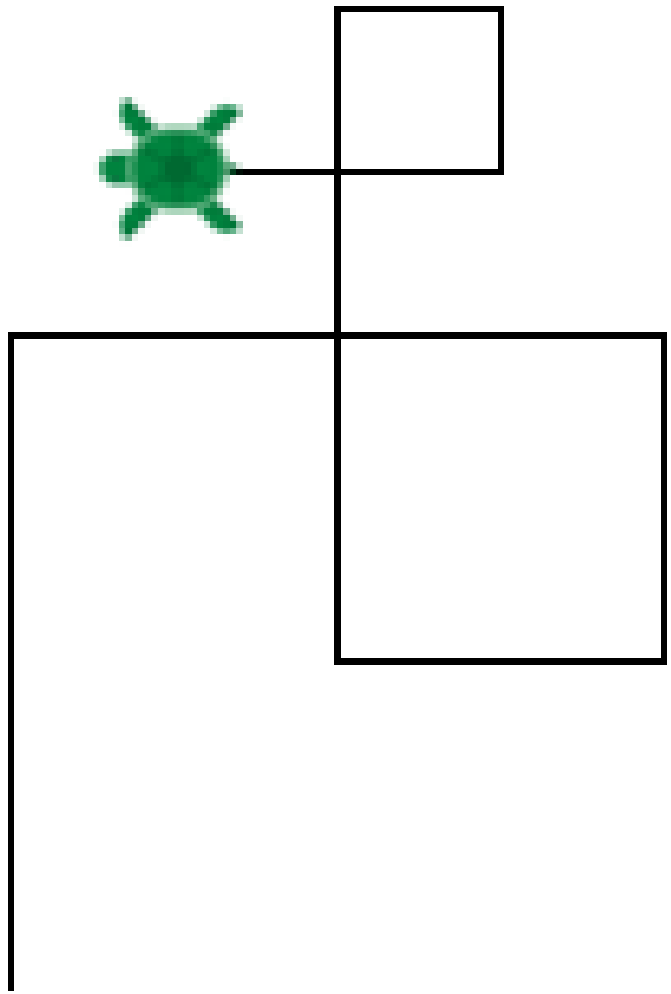
- is_run()
  - 判断程序是否应继续进行
    （窗口是否已被用户关闭）

三、循环绘图

- 使用循环和多重循环，可以绘制出更多美丽的图案

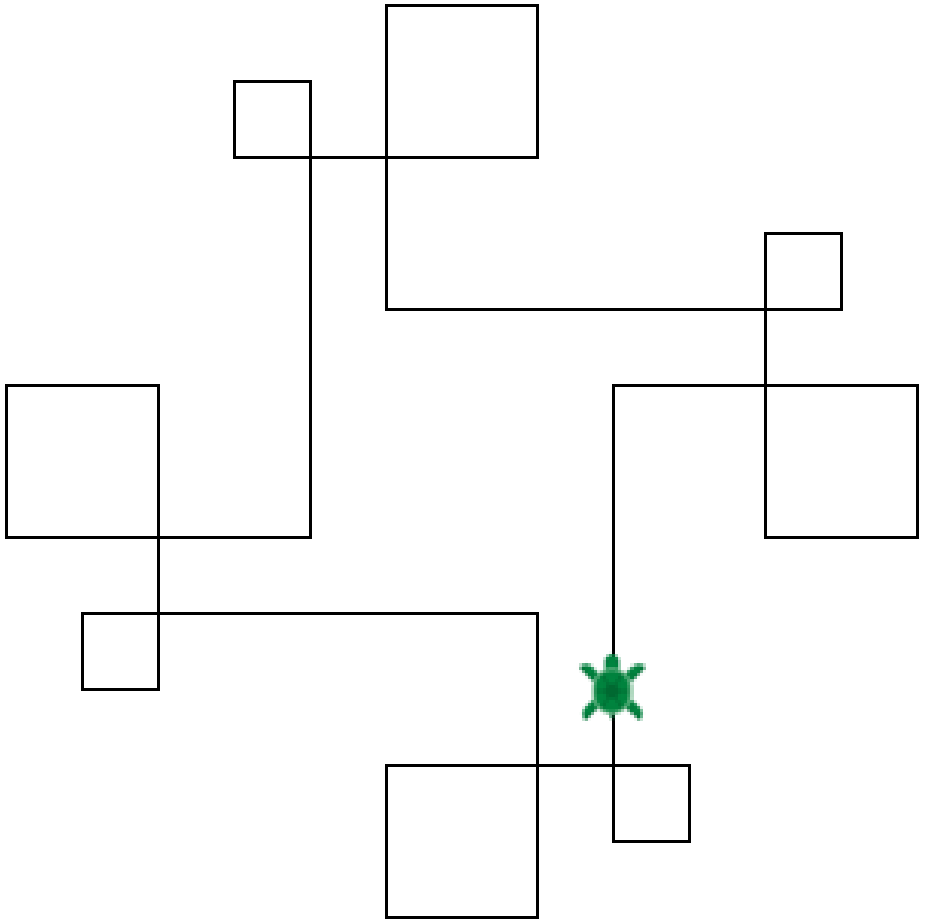
3.1 美丽花样

```
fd(100)
rt(90)
fd(100)
rt(90)
fd(50)
rt(90)
fd(50)
rt(90)
fd(100)
rt(90)
fd(25)
rt(90)
fd(25)
rt(90)
fd(50)
```



3.1 美丽花样(2)

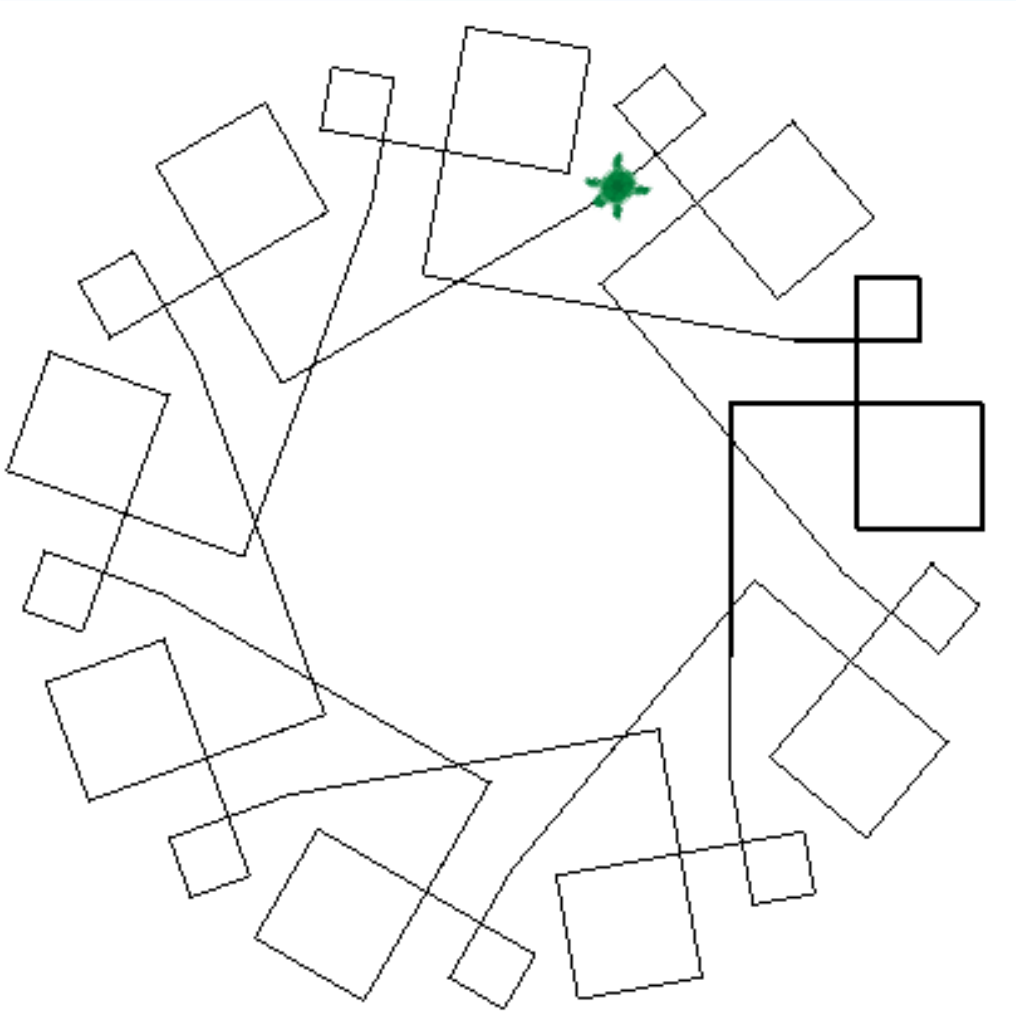
```
for i in range(4):
    fd(100)
    rt(90)
    fd(100)
    rt(90)
    fd(50)
    rt(90)
    fd(50)
    rt(90)
    fd(100)
    rt(90)
    fd(25)
    rt(90)
    fd(25)
    rt(90)
    fd(50)
```



循环4次

3.1 美丽花样(3)

```
while is_run():
    fd(100)
    rt(90)
    fd(100)
    rt(90)
    fd(50)
    rt(90)
    fd(50)
    rt(90)
    fd(100)
    rt(90)
    fd(25)
    rt(90)
    fd(25)
    rt(90)
    fd(50)

    rt(10)
    forward(50)
```



错开一点距离和角度，不断重复

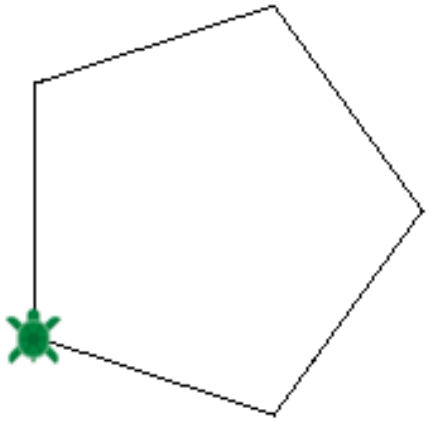
3.2 多边形

- 我们知道,任意多边形都可以由一系列连续下面两条语句的组合来绘出:
  - forward(side)
  - right_turn(angle)
- 因此,我们可以准备下面这段程序,只要改变参数变量n,side和angle的值，就能绘制出多种不同多变形来

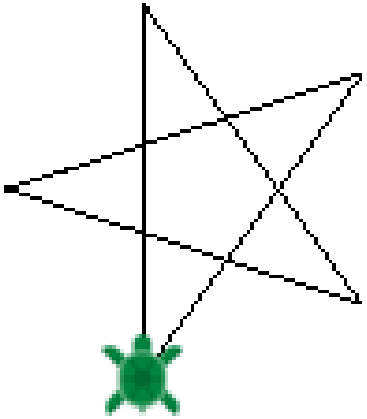
```
for i in range(n):
    fd(side)
    rt(angle)
```

```
for i in range(n):
    fd(side)
    rt(angle)
```

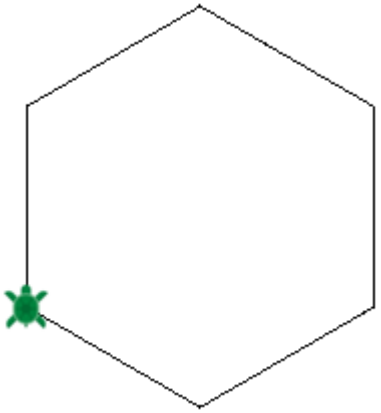
```
side=100
angle=72
n=5
```



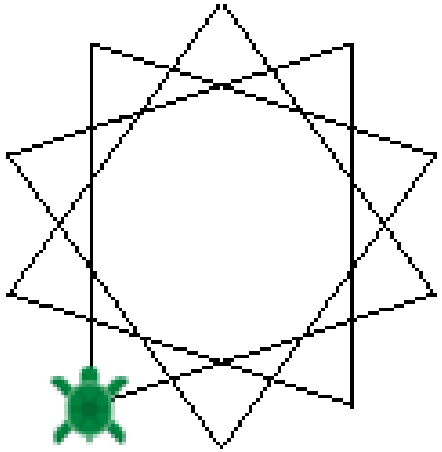
```
side=100
angle=144
n=5
```



```
side=100
angle=60
n=6
```



```
side=100
angle=108
n=10
```

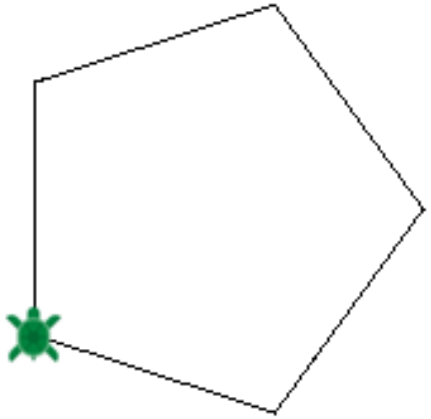


3.3 圆与正多边形

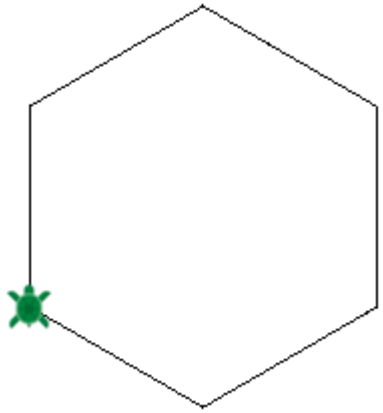
- 我们知道，任意凸多边形的内角和等于（边数-2)*180
- 所以可以这样来画任意的n正多边形（边长为x）：
  - 计算每个角的度数angle=(n-2)*180/n
  - 循环n次，每次：
    - ✓ 海龟前进x
    - ✓ 海龟右转180-angle度

```
angle = (n - 2) * 180.0 / n
for i in range(n):
    fd(side)
    rt(180 - angle)
```

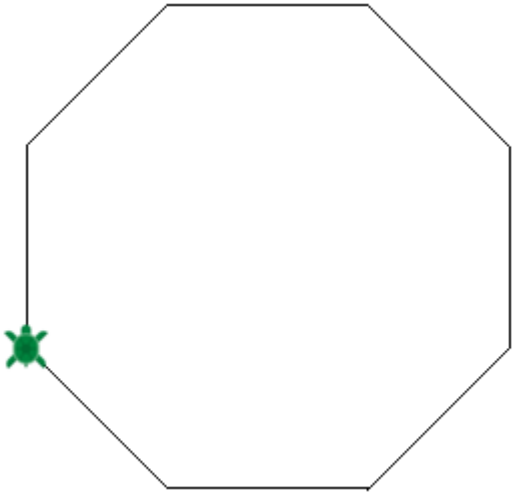
3.3 圆与多边形



正五边形



正六边形

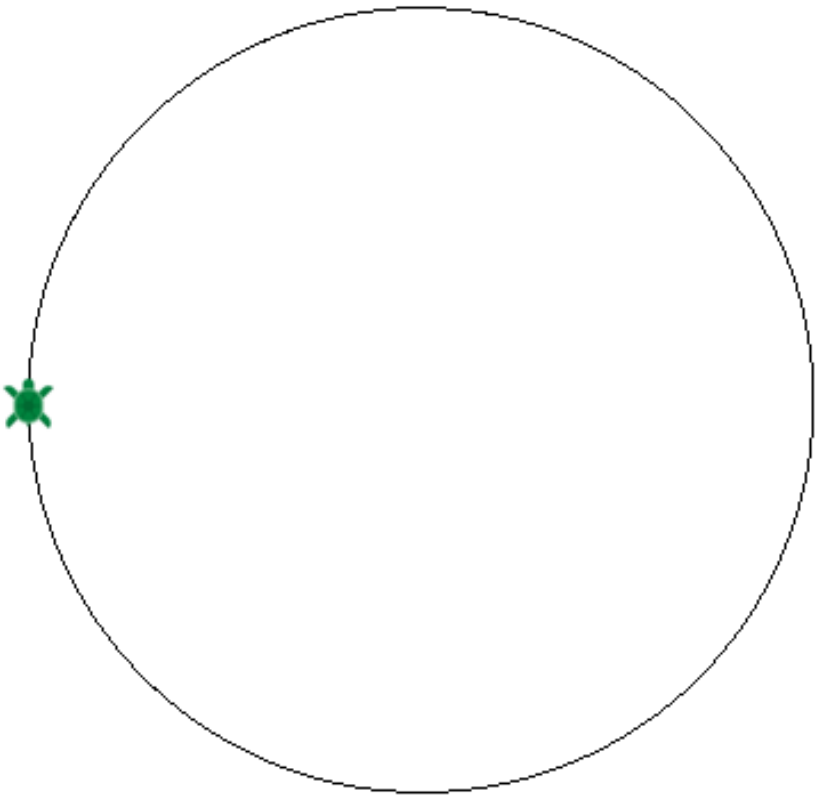


正八边形，边长为100

3.3 圆与多边形

- 当n越来越大时，我们会发现多边形越来越接近圆形
- 我们可以用正360边型来模拟圆形，也可以用同样的方法来绘制圆弧

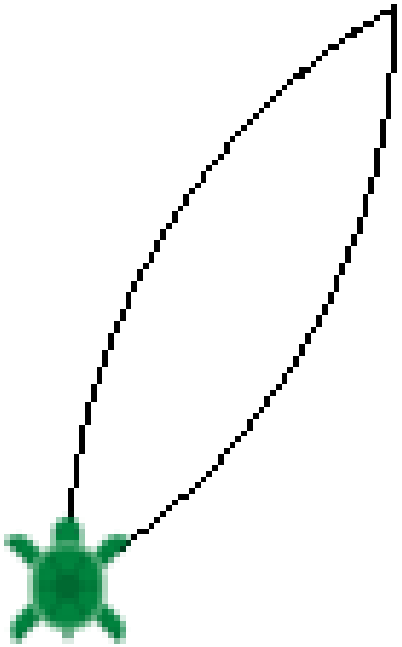
```
size = 3
for i in range(360):
    fd(size)
    rt(1)
```



3.3 圆与多边形

- 用两个60度的圆弧组成一片叶子：

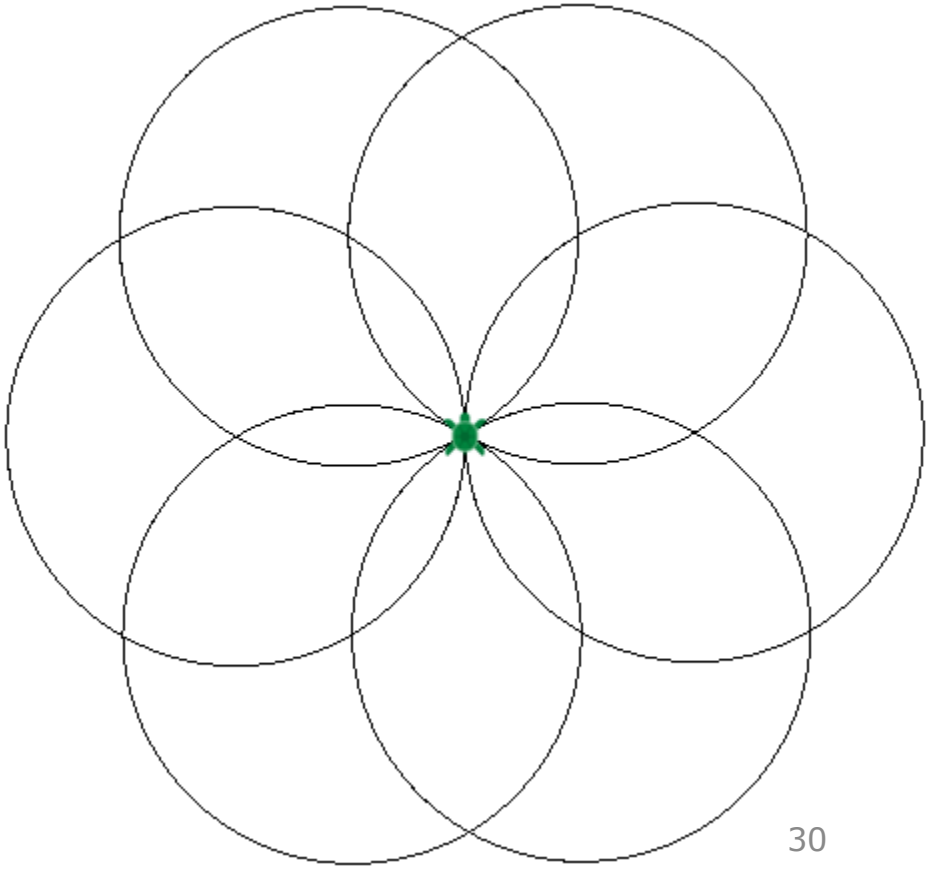
```
for i in range(60):
    fd(2)
    rt(1)
rt(120)

for i in range(60):
    fd(2)
    rt(1)
rt(120)
```



3.4 多重循环绘图

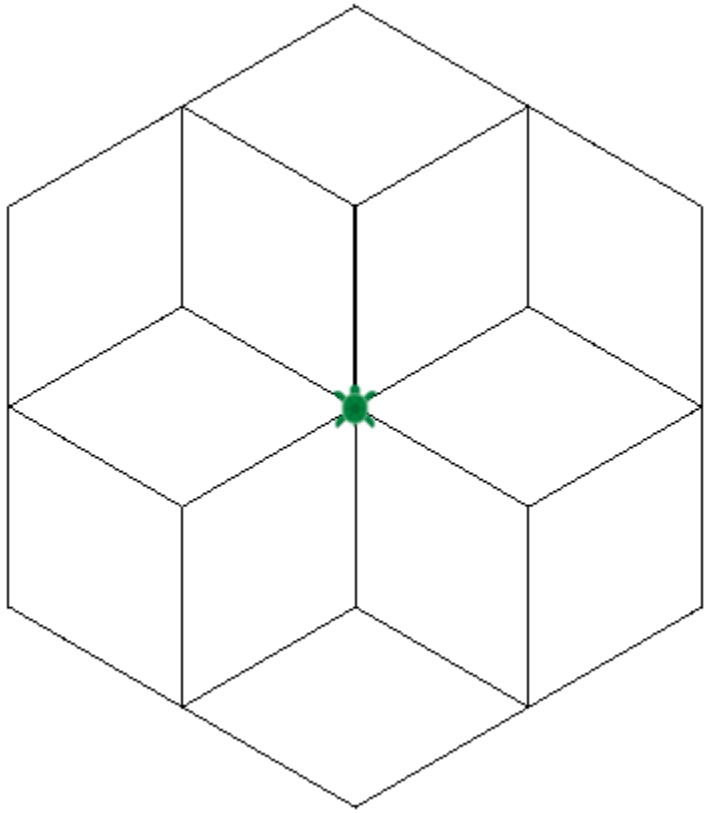
- 利用多重循环，我们可以在圆和多边形的基础上绘制出更复杂的图案

```
for i in range(6):
    for j in range(360):
        fd(2)
        rt(1)
    rt(60)
```



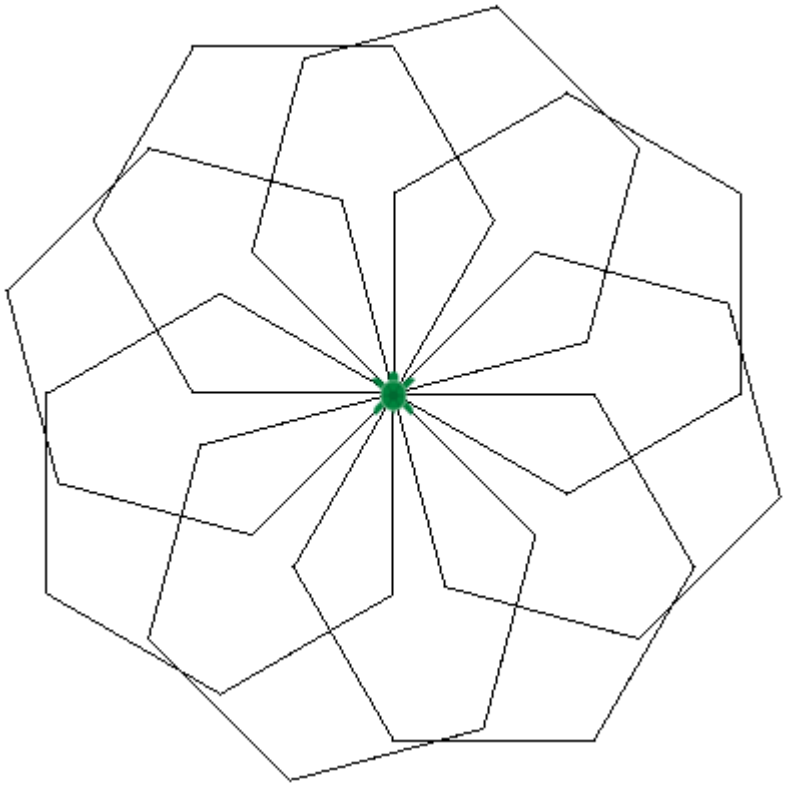
3.4 多重循环绘图

```
for i in range(6):
    for j in range(6):
        fd(100)
        rt(60)
    rt(60)
```



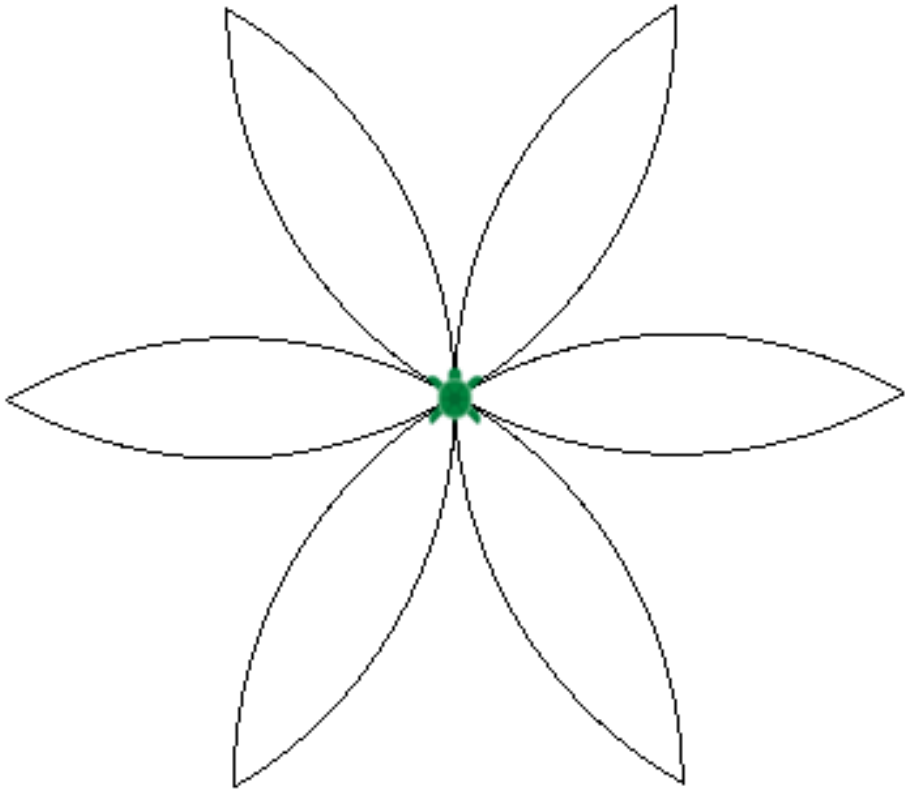
3.4 多重循环绘图

```
for i in range(8):
    for j in range(6):
        fd(100)
        rt(60)
    rt(45)
```



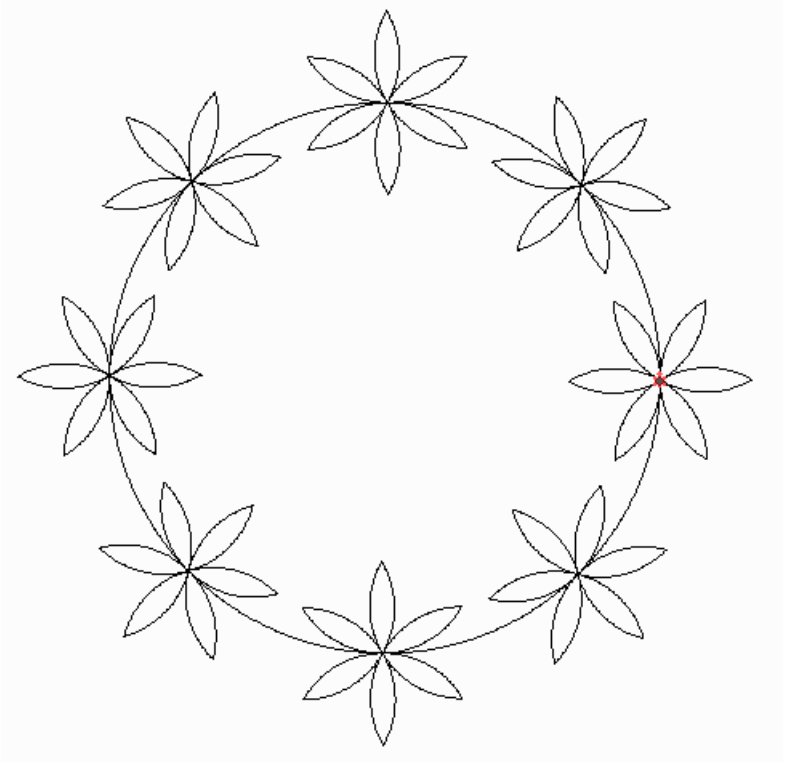
3.4 多重循环绘图

```
for i in range(6):
    for j in range(60):
        fd(3)
        rt(1)
    rt(120)
    for j in range(60):
        fd(3)
        rt(1)
    rt(120)
    rt(60)
```



3.4 多重循环绘图

想想，该如何绘制出右侧的花环？



四、函数

- 在上一章绘制的过程中，我们可以发现，复杂图形的绘制可以分解为若干个的简单图形的绘制。
  - 很多简单图形是类似的
- 如果能够将简单图形的绘制步骤用一条语句代替，我们的绘制过程将更简洁
- 在C语言里，我们可以用函数来实现这一点

4.1 自定义函数

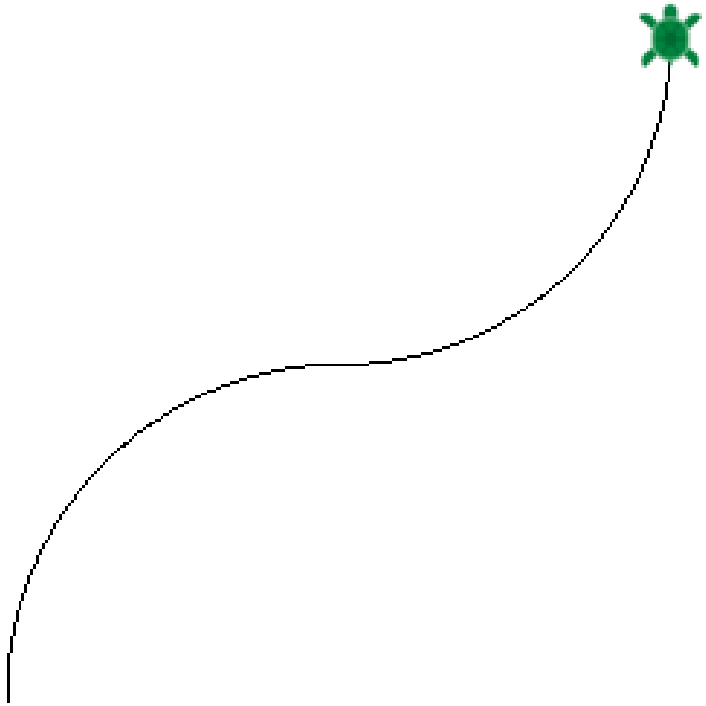
```
def arcl(side, degree):
    for i in range(degree):
        fd(side)
        lt(1)


def arcr(side, degree):
    for i in range(degree):
        fd(side)
        rt(1)
```

我们在主程序前面定义两个绘制弧线的函数arcr和arcl

4.1 自定义函数

有了这两个自定义函数，我们就可以在主函数中调用它们：

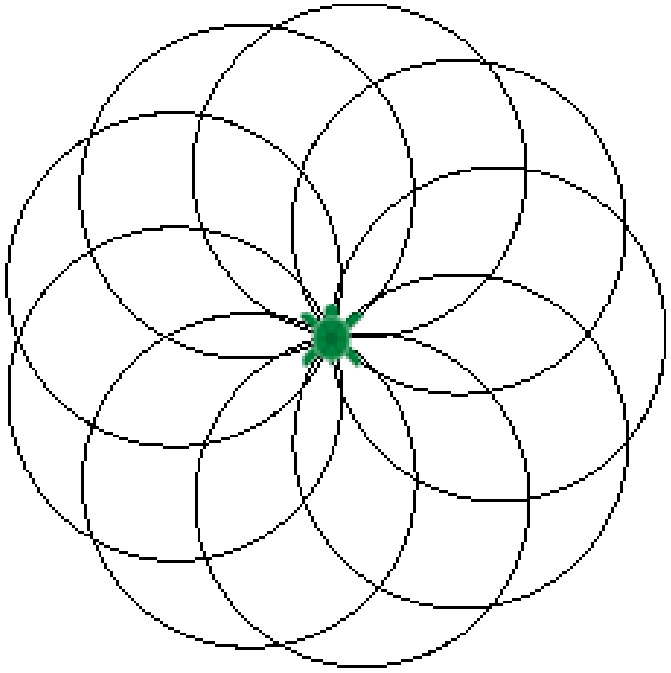
```
arcr(2, 90)
arcl(2, 90)
```



4.1 自定义函数

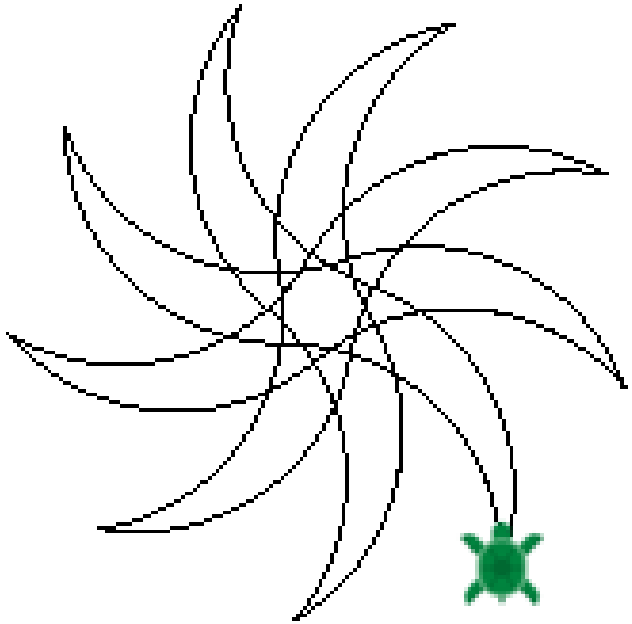
- 利用这些函数，我们可以绘制出更复杂的图形：

```
for i in range(9):
    arcr(1, 360)
    rt(40)
```



4.1 自定义函数

```
for i in range(9):
    for j in range(2):
        arcl(1, 90)
        arcr(1, 90)
        rt(160)
```



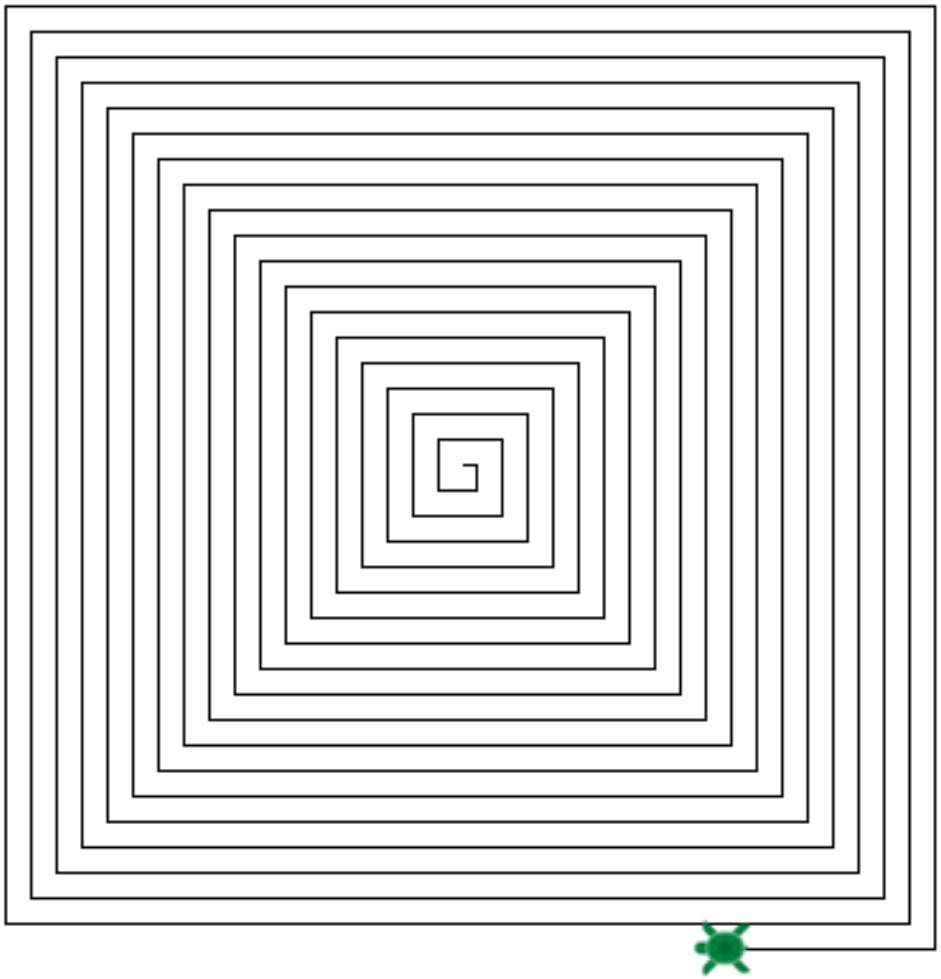
4.2 重复曲线

使用函数，我们可以使主函数更简洁，也便于调整程序的参数：

```
def polyspi(side, angle, inc):
    while is_run():
        fd(side)
        rt(angle)
        side += inc
```

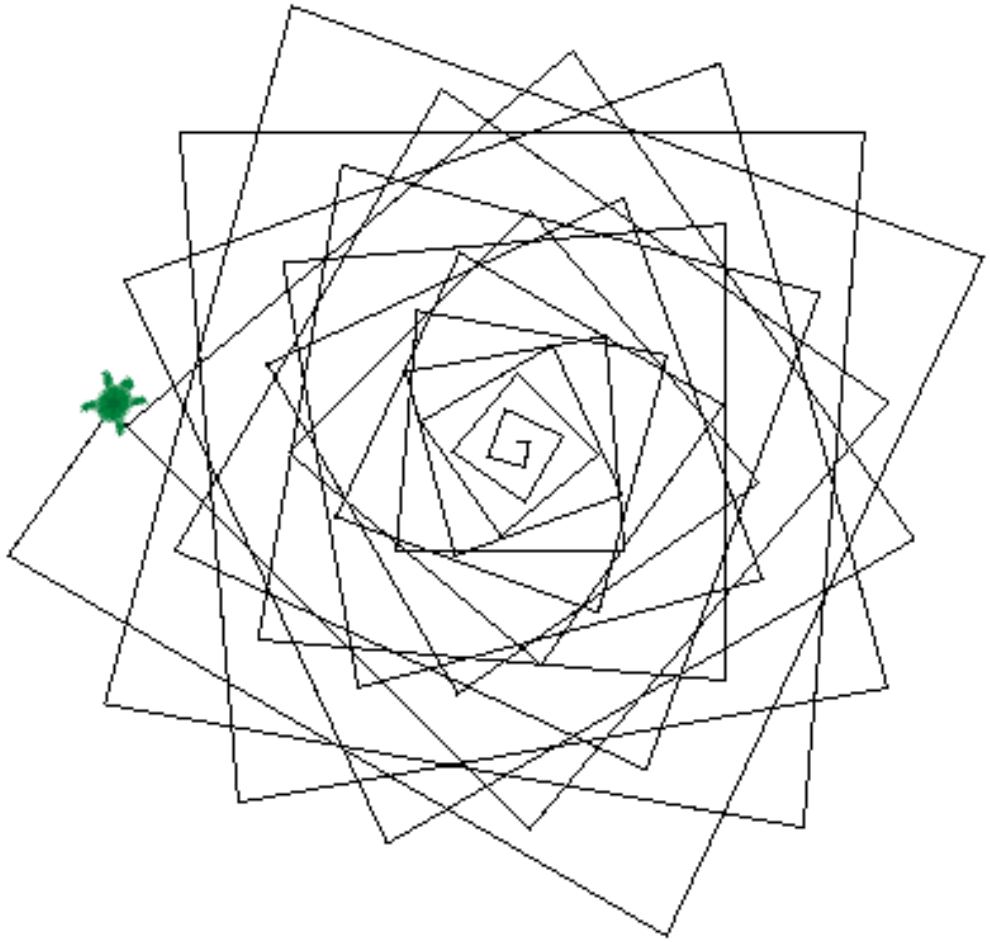
4.2 重复曲线

```
polyspi(0, 90, 5)
```



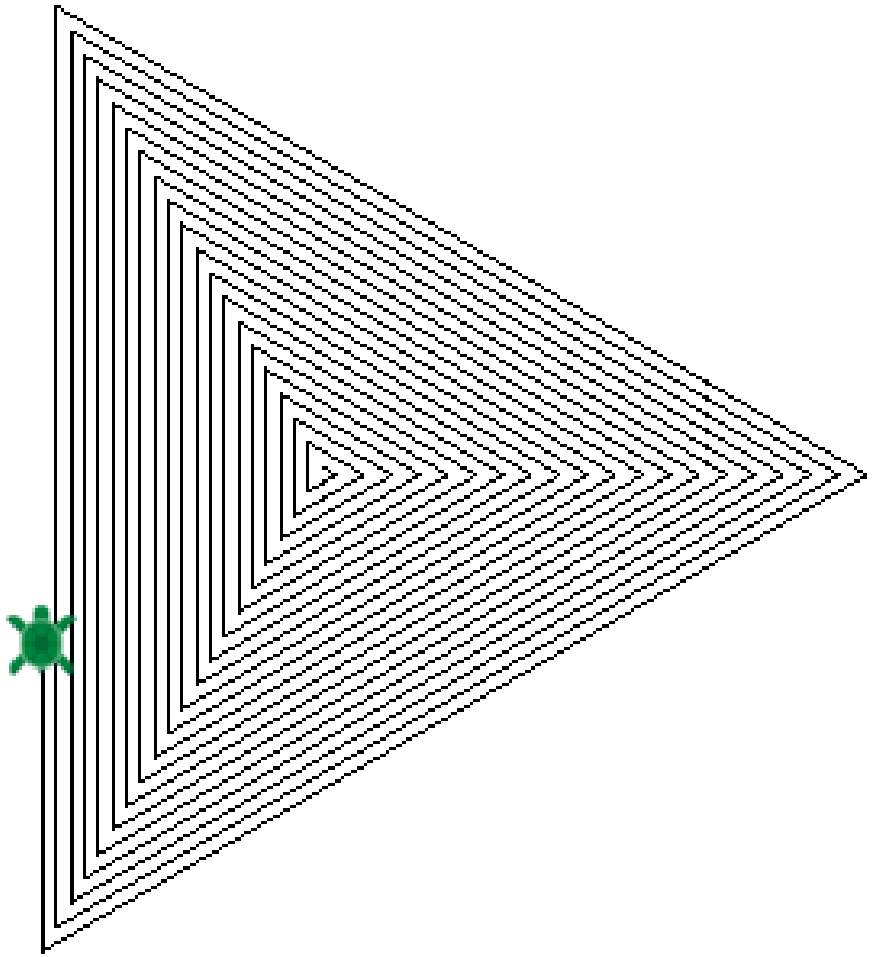
4.2 重复曲线

```
polyspi(0, 95, 5)
```



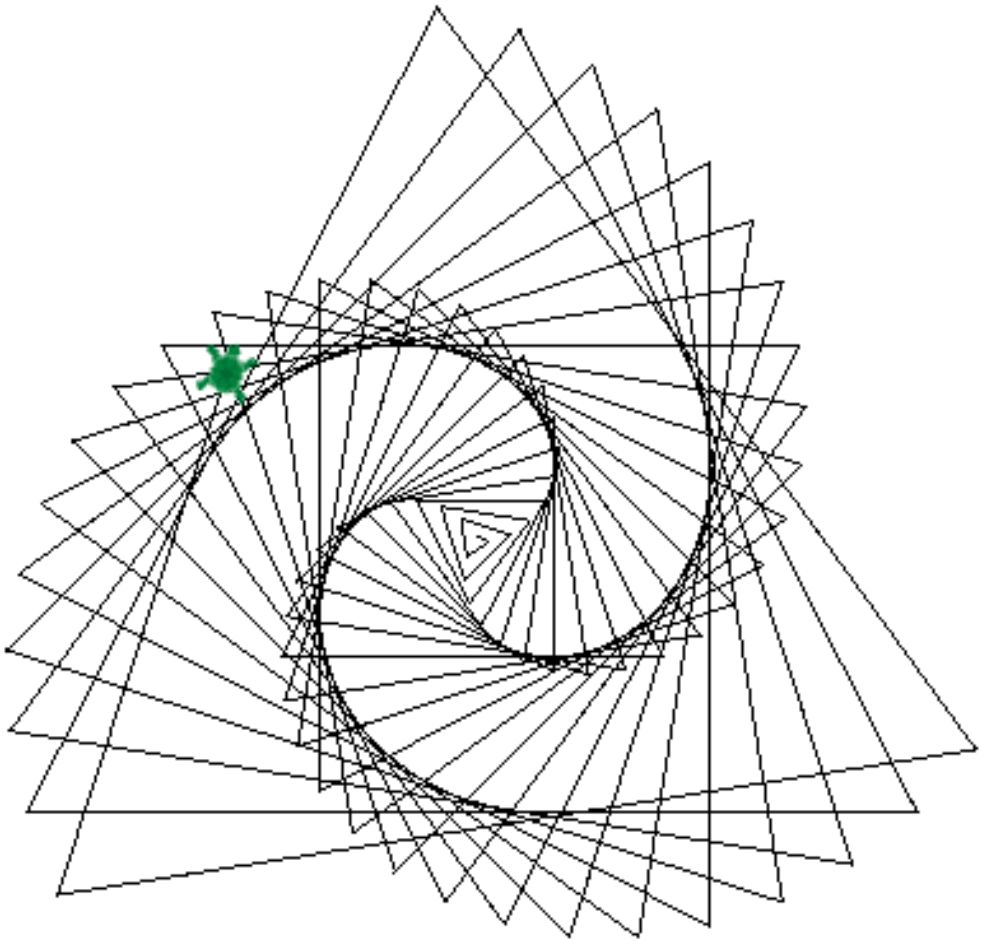
4.2 重复曲线

```
polyspi(0, 120, 5)
```



4.2 重复曲线

```
polyspi(0, 117, 5)
```



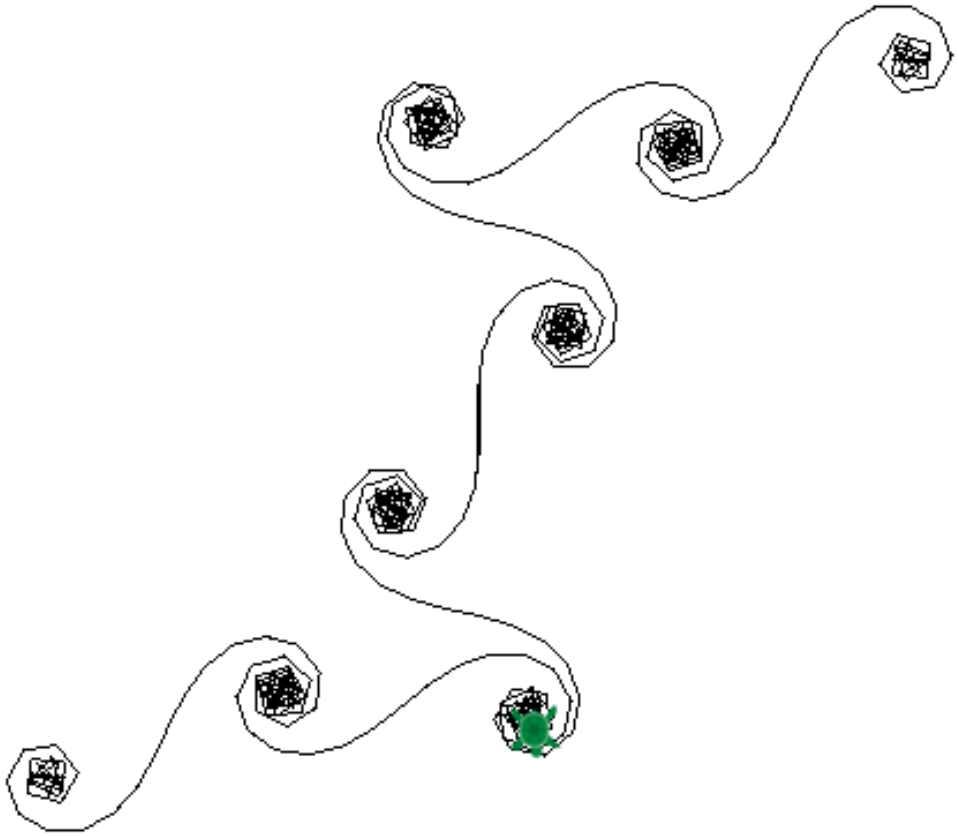
4.3 重复曲线2

我们来看看另外一种重复曲线，它和前面曲线的区别在什么地方？
：

```
def inspi(side, angle, inc):
    while is_run():
        fd(side)
        rt(angle)
        angle += inc;
```

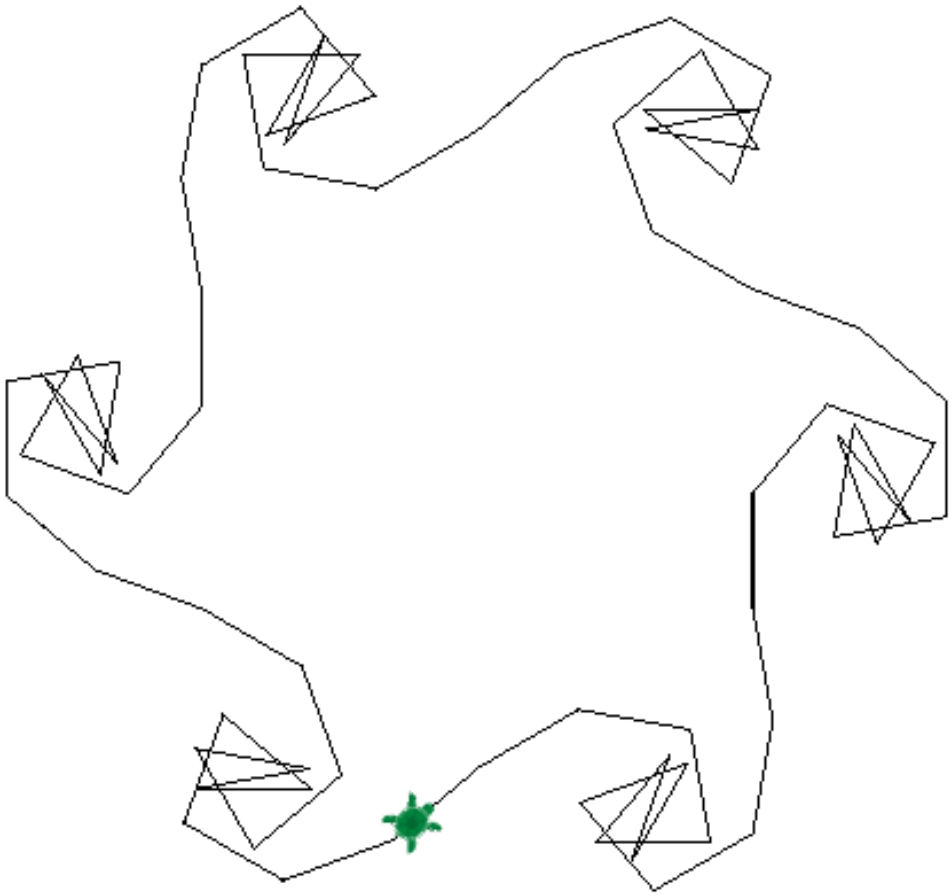
4.3 重复曲线2

```
inspi(15,0,7)
```



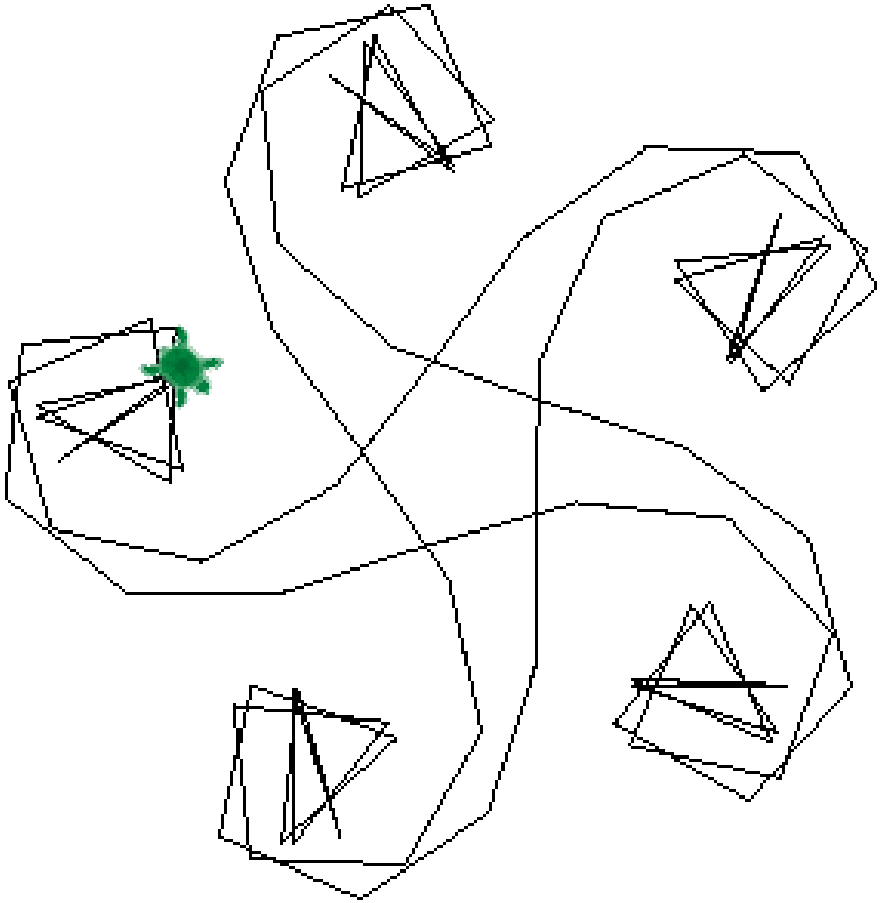
4.3 重复曲线2

```
inspi(50,40,30)
```



4.3 重复曲线2

```
inspi(50,2,20)
```



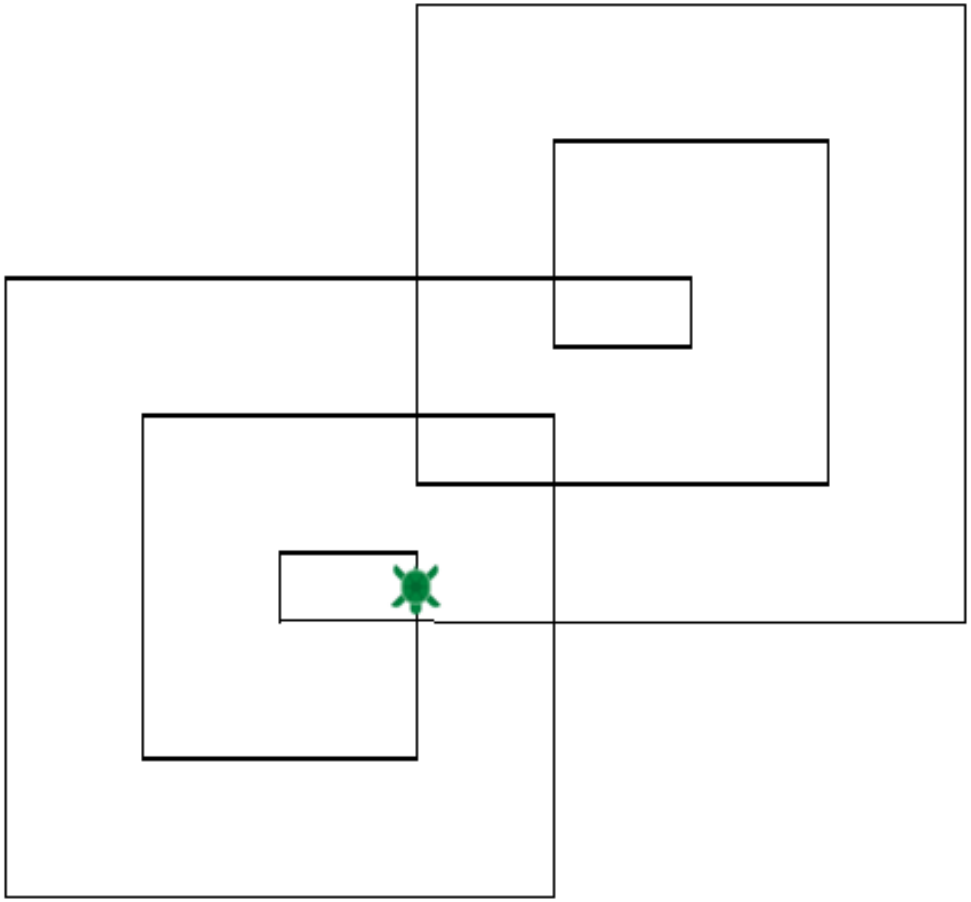
4.4 函数嵌套

在自定义函数中也可以调用已经定义的其他函数，例：

```
def subspiro(side, angle, n):
    for i in range(1,n+1):
        fd(side * i)
        rt(angle)

def spiro(side, angle, n):
    while is_run():
        subspiro(side, angle, n)
```

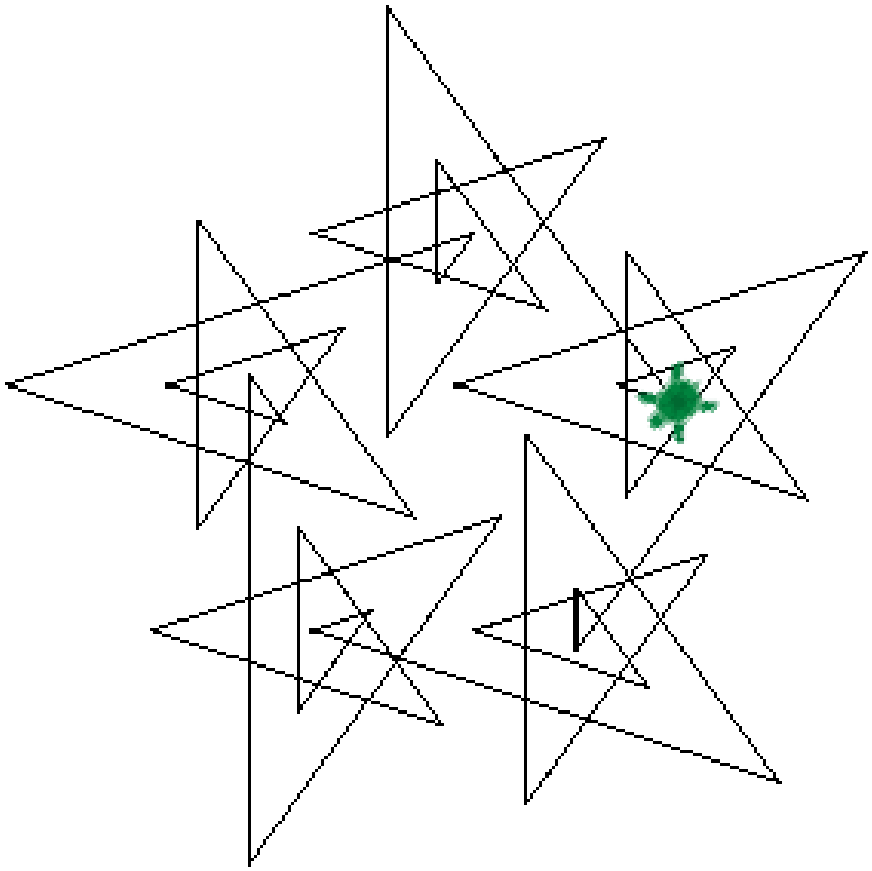
4.4 函数嵌套

```
spiro(30,90,10)
```



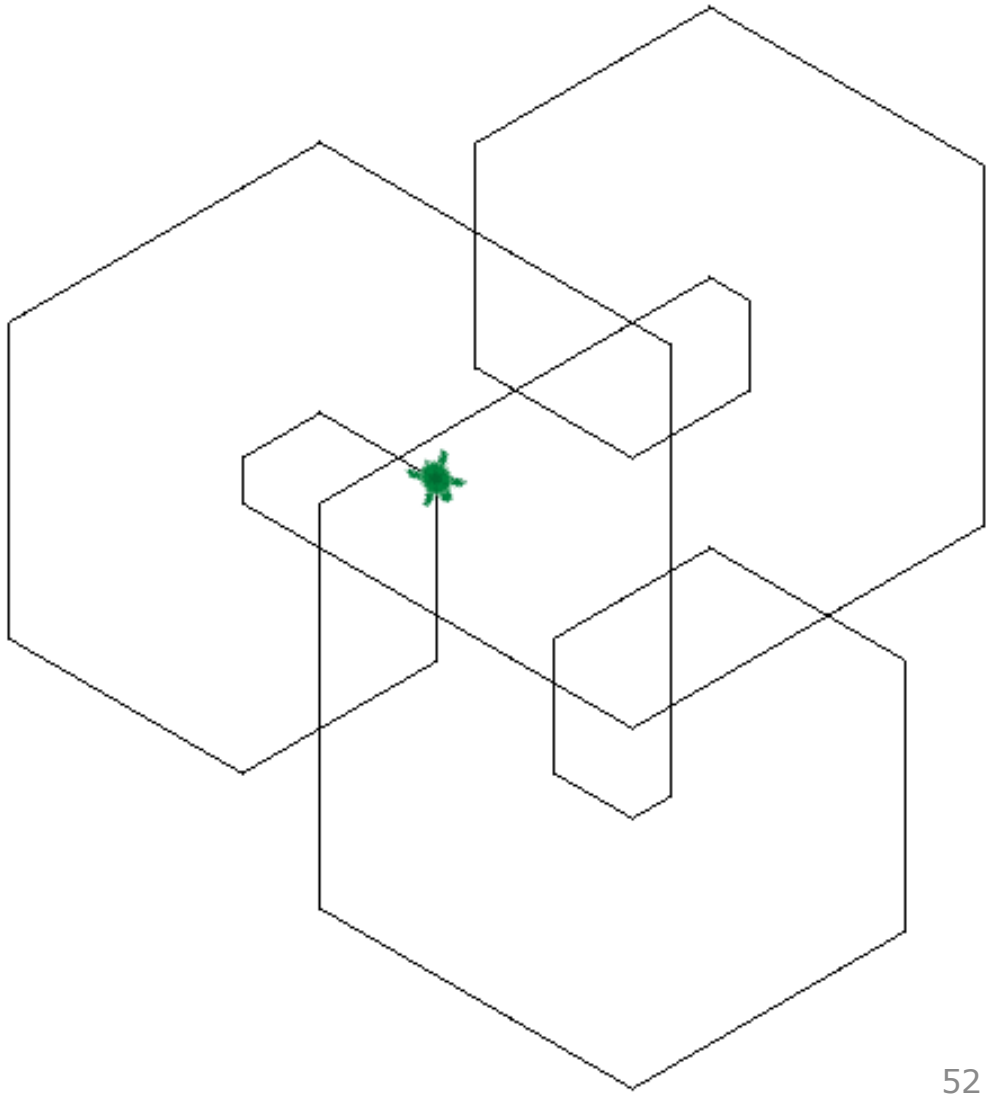
4.4 函数嵌套

```
spiro(20,144,8)
```



4.4 函数嵌套

```
spiro(20,60,10)
```



参考书目

- easygraphics网站: http://easygraphics.royqh.net/
- 《Turtle Geometry》. Harold Abelson, Andrea A. Disessa. The MIT Press. 1980