

Python艺术编程05

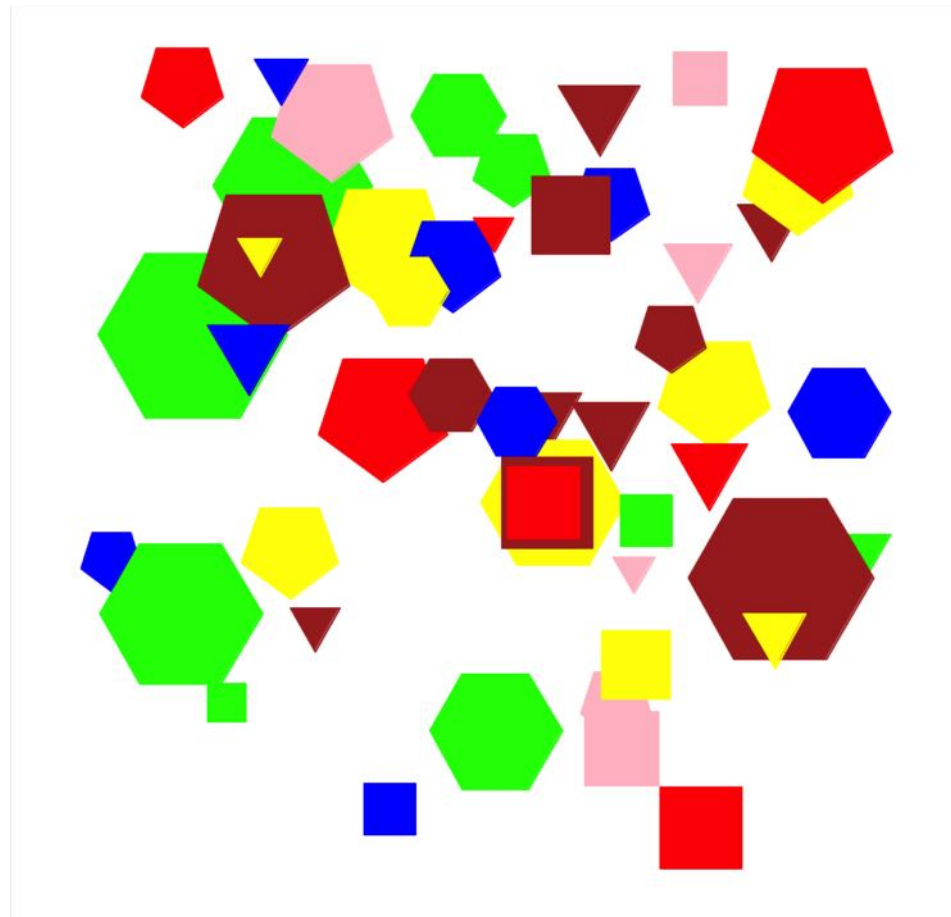
——函数定义和分形树

北京大学 陈斌

2018.10.08

目录

- 函数的基本概念
- 组合图形
- 上机练习
- 递归的概念
- 绘制简单二叉树

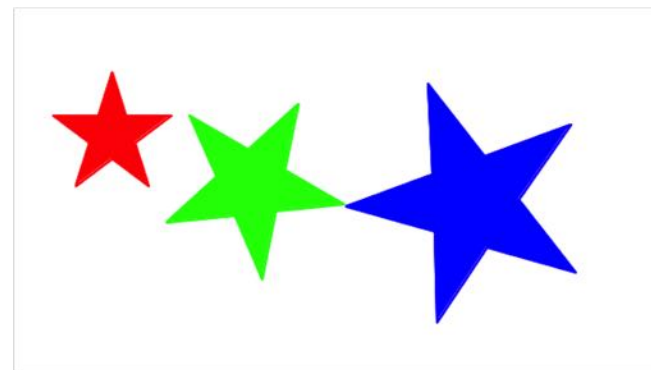


代码复用情境

- 有时候我们需要反复使用某些代码
 - 比如组合图形出现多个五角星
- 如果到处拷贝这些代码，会出现弊端
 - 程序变得冗长，可读性差
 - 一旦需要修改或扩充，要在各处同步改代码，容易出错，可维护性差

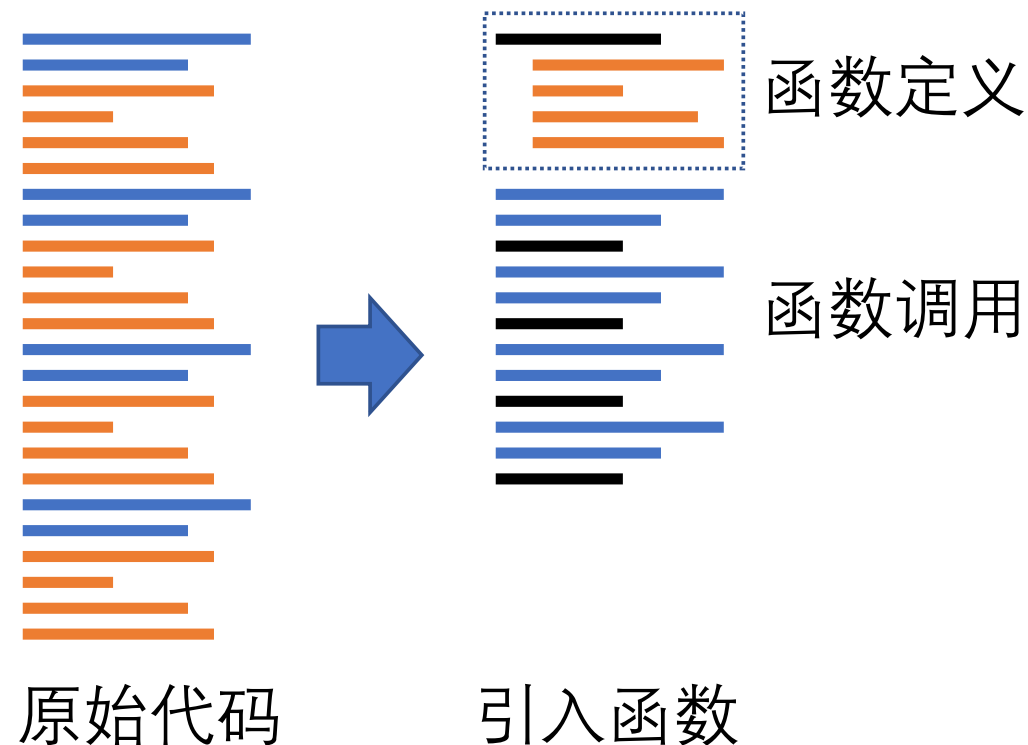
```
1  import turtle
2
3  t = turtle.Turtle()
4
5  t.color('red')
6  t.begin_fill()
7  for i in range(5):
8      t.forward(20)
9      t.left(72)
10     t.forward(20)
11     t.right(144)
12 t.end_fill()
13
14 t.penup()
15 t.forward(60)
16 t.right(30)
17 t.pendown()
18
19 t.color('green')
20 t.begin_fill()
21 for i in range(5):
22     t.forward(30)
23     t.left(72)
24     t.forward(30)
25     t.right(144)
26 t.end_fill()
27
28 t.penup()
29 t.forward(80)
30 t.left(50)
31 t.pendown()
32
33 t.color('blue')
34 t.begin_fill()
35 for i in range(5):
36     t.forward(40)
37     t.left(72)
38     t.forward(40)
39     t.right(144)
40 t.end_fill()
41
42 t.hideturtle()
43 turtle.done()
```

func_star.py



解决方案：函数（functions）

- 我们把这些重复代码单独收集起来，组成一个“函数”对象，并赋予一个名称
- 在需要用到这些代码的时候就通过名称来“呼叫”这些“函数”
- 前者称为函数定义（define）
- 后者称为函数调用（call）



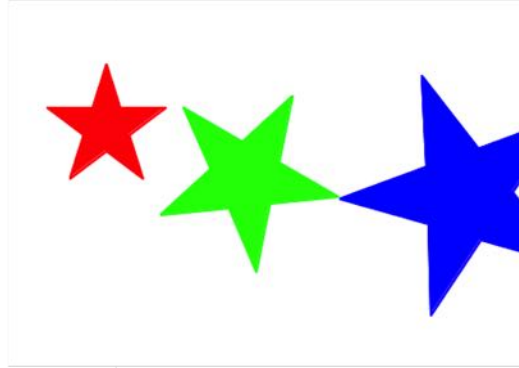
定义函数：def语句

- 函数定义语句def
 - `def` <函数名称>(<参数表>):
 - <语句块>
 - [`return` <返回值>]
- 几个要素
 - `def`关键字
 - 函数名称，后跟一对圆括号
 - (可选的) 参数表
 - 语句块
 - (可选的) 返回值

func_star2.py

```
1 import turtle
2
3
4 def star(size, color):
5     t.color(color)
6     t.begin_fill()
7     for i in range(5):
8         t.forward(size)
9         t.left(72)
10        t.forward(size)
11        t.right(144)
12    t.end_fill()
13
14
15 t = turtle.Turtle()
16
17 star(20, 'red')
18 t.penup()
19 t.forward(60)
20 t.right(30)
21 t.pendown()
22 star(30, 'green')
23 t.penup()
24 t.forward(80)
25 t.left(50)
26 t.pendown()
27 star(40, 'blue')
28
29 t.hideturtle()
30 turtle.done()
```

函数定义



函数的参数

- 如果代码块里没有可供调节的选项，可以定义没有参数的简单函数
- 一般函数会带有可供调节的参数，参数可以有多个
 - 如画五角星的函数，包含两个参数：大小size和颜色color

func_star2.py

```
4  def star(size, color):  
5      t.color(color)  
6      t.begin_fill()  
7      for i in range(5):  
8          t.forward(size)  
9          t.left(72)  
10         t.forward(size)  
11         t.right(144)  
12     t.end_fill()
```

函数的返回值

- 有时候函数会有返回值
 - 如math模块中求平方根的函数`math.sqrt(n)`返回n的平方根
- `return`语句负责结束函数执行，并返回值
- `return`语句可以根据需要，出现在语句块中的任何位置

```
1  import math
2
3  s = math.sqrt(120)
4  print(s)
5
6
7  def my_abs(n):
8      if n >= 0:
9          return n
10     else:
11         return -n
12
13
14 s = my_abs(-10)
15 print(s)
```

函数定义中的代码块

- 由于函数定义def语句仅仅是把代码块“打包封装”
- def语句执行的时候，代码块并不会被执行
- 所以，在执行def语句的时候，除非语句块中包含了明显的语法错误
- Python解释器是不会检查语句块中其它错误的。

func_star2.py

```
1  import turtle
2
3
4  def star(size, color):
5      t.color(color)
6      t.begin_fill()
7      for i in range(5):
8          t.forward(size)
9          t.left(72)
10         t.forward(size)
11         t.right(144)
12     t.end_fill()
13
14
15 t = turtle.Turtle()
16
17 star(20, 'red')
```

并不会出现
“t未定义”
的错误

调用函数： call function

- def定义了函数之后，函数名称仅代表这个“函数对象”
- 如果需要执行语句块代码，需要有如下的要素
 - 函数名称，后加括号
 - 括号内放置参数的具体值
- 没有或者不需要返回值
 - func(a,b,c) #如调用star
- 获取返回值
 - v = func(a,b,c)

func_call.py

```
1 def my_abs(n):
2     if n >= 0:
3         return n
4     else:
5         return -n
6
7
8 # 函数对象
9 s = my_abs
10 print("function object:", s)
11
12 # 加括号和参数, 调用函数
13 s = my_abs(-10)
14 print("call function:", s)
```

随机数模块random

- 产生一定范围内的随机数
 - `random.randint(min,max)`
- 从列表中随机选择
 - `random.choice(list)`

```
>>> import random
>>> colors=['red', 'green', 'blue', 'brown', 'pink']
>>> c=random.choice(colors)
>>> c
'green'
>>> n=random.randint(10,20)
>>> n
19
```

上机练习

- 定义一个多边形函数
 - `def polygon(n,size,color):`
 - 绘制正n边形，边长为size，填充颜色color
- 编写一个程序，绘制现代时尚几何多边形色块抽象装饰画
 - 随机模块random
 - `t.goto(x,y)`
- 可以进一步修改程序
 - 如：将H4中的曲线定义为函数，组合进随机图案中来

