Python程序设计

陈远祥

chenyxmail@gmail.com

北京邮电大学 电子工程学院

Python程序设计

上周主要内容

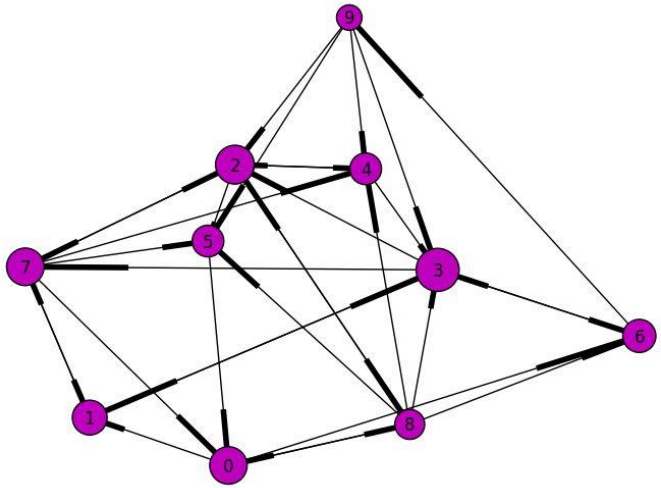
- Python语法元素分析：缩进、关键字、布尔值
- 程序的控制结构（顺序结构，分支结构，循环结构）
- 异常处理机制

Python程序设计

本周主要内容

- 函数和代码的复用

函数的定义



函数的定义

- 函数是一段具有特定功能的、可重用的语句组，用函数名来表示并通过函数名进行功能调用
- 函数也可以看作是一段具有名字的子程序，可以在需要的地方调用执行，不需要在每个执行地方重复编写这些语句
- 每次使用函数可以提供不同的参数作为输入，以实现对不同数据的处理；函数执行后，还可以反馈相应的处理结果

函数的定义

- 函数是一种功能抽象：完成特定功能，与黑盒类似，对函数的使用不需要了解函数内部实现原理，只要了解函数的输入输出方式
- 分类：
  - ✓ 用户定义函数：用户自己编写的
  - ✓ 系统自带函数及第三方函数： Python内嵌的函数（如abs()、eval()）、Python标准库中的函数（如math库中的sqrt()）、图形库中的方法（如myPoint.getX()）

函数的定义

- 使用函数的目的
  - ✓ 降低编程难度
  - ✓ 代码复用

函数的定义

- Python定义一个函数使用def保留字，语法形式如下：

```
def<函数名>(<参数列表>):
    <函数体>
    return<返回值列表>
```

函数的定义

- 函数名<name>：可以是任何有效的Python标识符
- 参数列表<parameters>：是调用函数时传递给它的值（可以由零个，一个或者多个参数组成，当有多个参数时，各个参数用逗号分隔）

函数的定义

- 函数体<body>：函数被调用时执行的代码，由一个或多个语句组成
- 函数调用的一般形式：

  <name>(parameters>)

函数的定义

- 形式参数：定义函数时，函数名后面圆括号中的变量，简称“形参”。形参只在函数内部有效
- 实际参数：调用函数时，函数名后面圆括号中的变量，简称 “实参”

函数的定义

- 定义函数：

```
def add1(x):
    x = x + 1
    return x
```

- 函数功能：将传给它的数值增1，返回增加后的值
- return语句：结束函数调用，并将结果返回给调用者
- return语句是可选的，可出现在函数体任意位置
- 没有return语句，函数在函数体结束位置将控制权返回给调用方

函数的定义

- 调用过程及运行结果

```
===========
>>> add1(3)
4
>>>
```

- 函数接口：参数和返回值
- 函数传递信息的主要途径：
  - ✓ 通过参数传递信息
  - ✓ 通过函数返回值的方式传递信息

函数的定义

- 编写一个程序打印“Happy Birthday”的歌词
- 标准的歌词：

Happy Birthday to you!

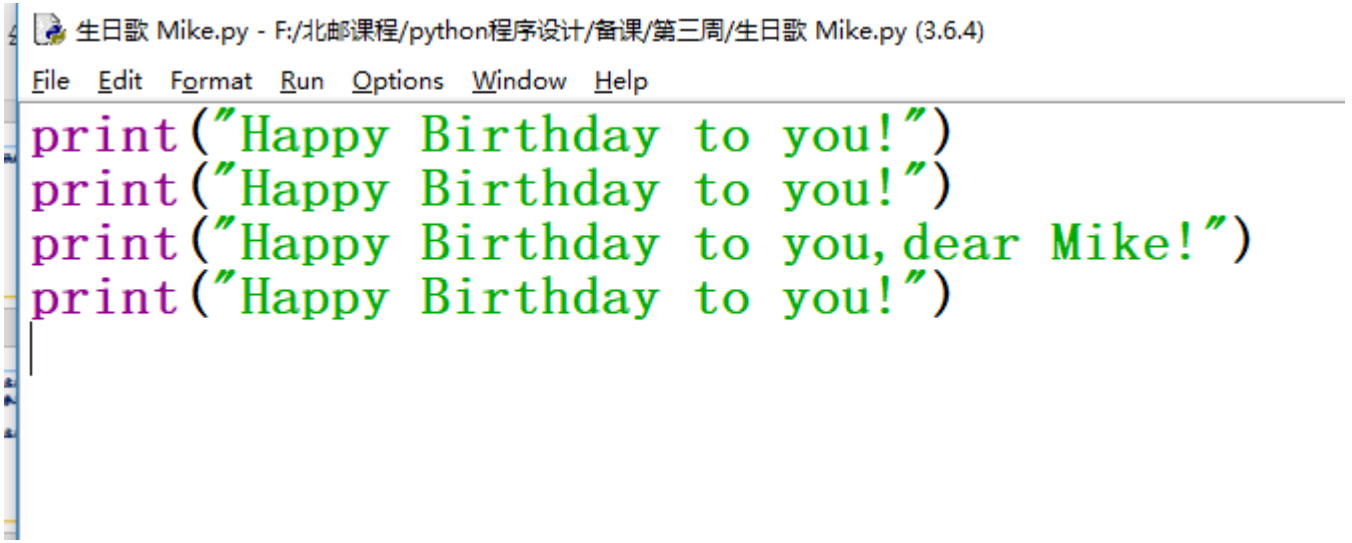
Happy Birthday to you!

Happy Birthday, dear <insert-name>

Happy Birthday to you!

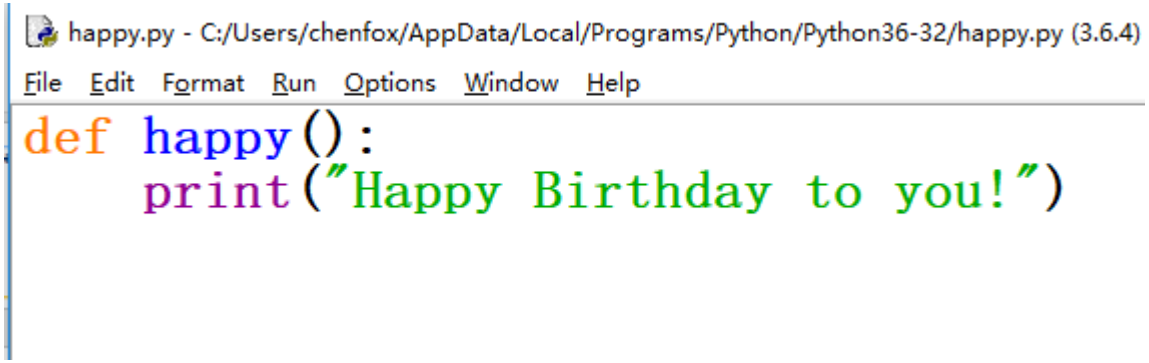
函数的定义

- 方法1：使用四个print语句
  - ✓ 给Mike唱生日快乐歌的程序代码：
  - ✓ #生日歌 Mike



函数的定义

- 方法2：使用函数来打印歌词的第一、二、四行
- 定义函数happy()

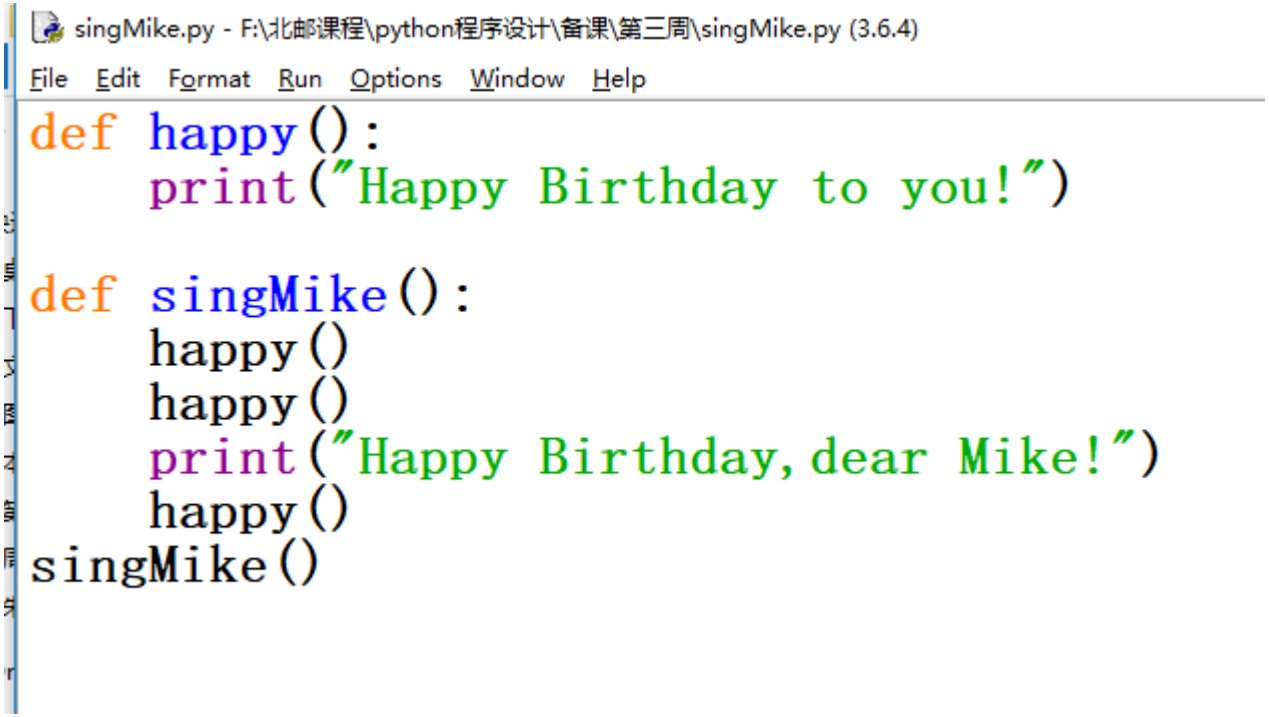


- 调用happy()

```
>>> happy()
Happy Birthday to you!
>>>
```

函数的定义

- 定义函数实现为Mike打印生日歌的歌词：
  - ✓ #singMike



函数的定义

- 写出给Mike和Lily唱生日歌的程序
  - ✓ #singLily

singLily.py - F:\北邮课程\python程序设计\备课\第三周\singLily.py (3.6.4)

File Edit Format Run Options Window Help

```
def happy():
    print("Happy Birthday to you!")

def singMike():
    happy()
    happy()
    print("Happy Birthday,dear Lily!")
    happy()
singLily()
```

函数的定义

- 写出给Mike和Lily唱生日歌的程序
  - ✓ #singMike and singLily

singMike and singLily.py - F:/北邮课程/python程序设计/备课/第三周/singMike and singLily.py (3.6.4)

File Edit Format Run Options Window Help

```
def happy():
    print("Happy Birthday to you!")

def singMike():
    happy()
    happy()
    print("Happy Birthday,dear Mike!")
    happy()

def singLily():
    happy()
    happy()
    print("Happy Birthday,dear Lily!")
    happy()

singMike()
print()
singLily()
```

函数的定义

- 例子3：简化程序，编写通用函数唱生日歌
  - ✓ #singperson

singperson.py - F:/北邮课程/python程序设计/备课/第三周/singperson.py (3.6.4)

File Edit Format Run Options Window Help

```
def happy():
    print("Happy Birthday to you!")

def sing(person):#person参数：此变量在函数被调用时初始化
    happy()
    happy()
    print("Happy Birthday,dear",person+"!")
    happy()

sing("Mike")
```

函数的定义

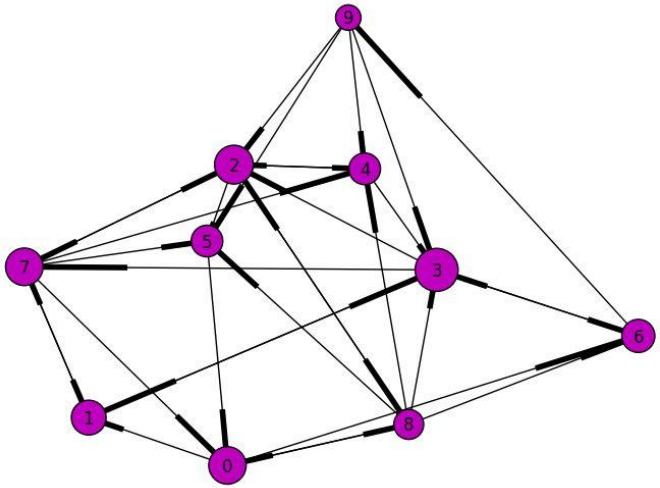
- 综合例子：利用sing()，为Mike、Lily和Lucy三个人唱生日歌
  - ✓ #sing3

sing3.py - F:/北邮课程/python程序设计/备课/第三周/sing3.py (3.6.4)

File Edit Format Run Options Window Help

```
def happy():
    print("Happy Birthday to you!")

def sing(person):#person参数：此变量在函数被调用时初始化
    happy()
    happy()
    print("Happy Birthday,dear",person+"!")
    happy()

def main():
    sing("Mike")
    print()
    sing("Lily")
    print()
    sing("Lucy")
    print()

main()
```

函数的调用和返回



函数的调用

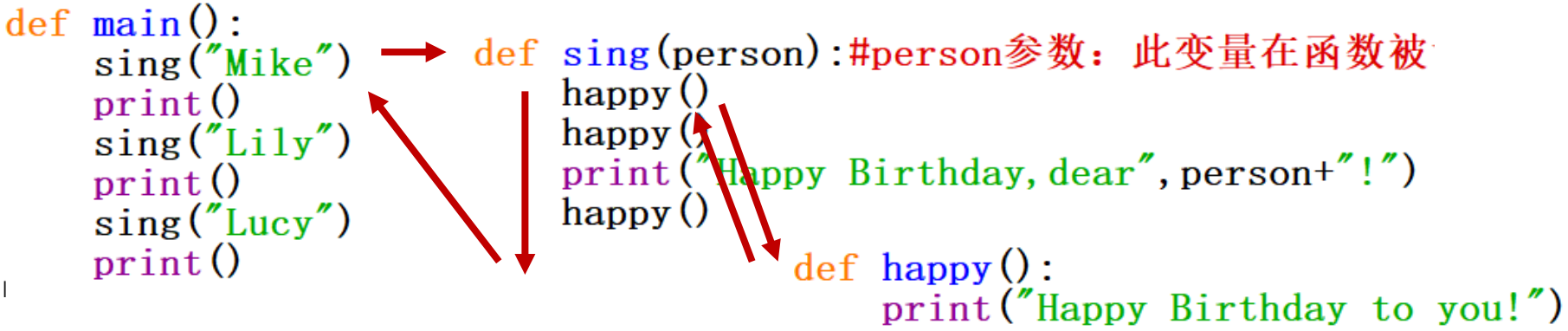
- 函数调用执行的四个步骤：
  - ✓ 调用程序在调用处暂停执行
  - ✓ 函数的形参在调用时被赋值为实参
  - ✓ 执行函数体
  - ✓ 函数被调用结束，给出返回值

函数的调用

- 生日歌程序的main()中部分程序：

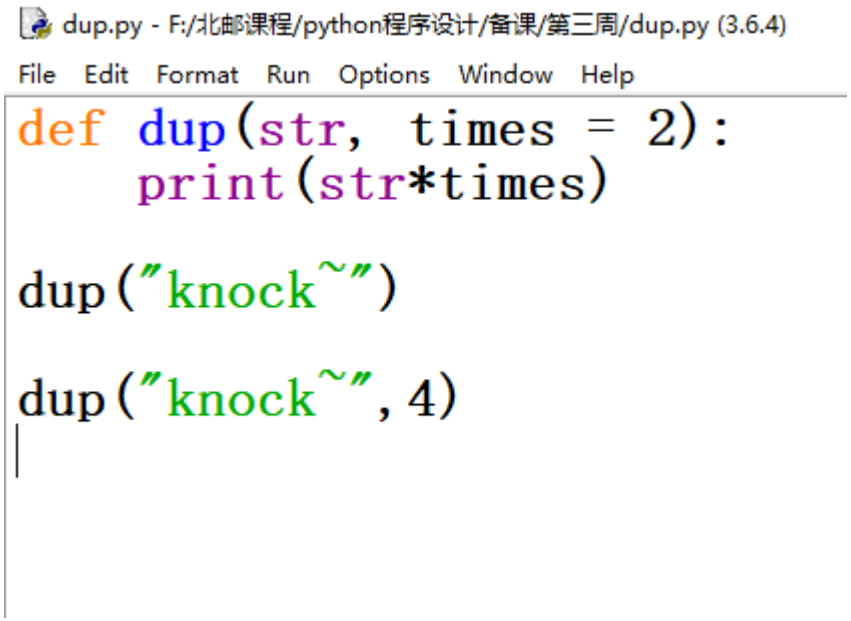
```
sing("Mike")
print()
sing("Lily")
print()
sing("Lucy")
print()
```

- sing()参数person初始化的调用过程图：



函数的参数传递

- 可选参数和可变数量参数
- 在定义函数时，有些参数可以存在默认值
  - ✓ #dup



函数的参数传递

- 在函数定义时，可以设计可变数量参数，通过参数前增加星号（*）实现
  - ✓ #vfun

File Edit Format Run Options Window Help

```
def vfun(a, *b):
    print(type(b))
    for n in b:
        a += n
    return a

vfun(1, 2, 3, 4, 5)
```

函数的参数传递

- Python提供了按照形参名称输入实参的方式，调用如下：

```
def  func(x2, y2, z2, x1, y1, z1)
result=func(4, 5, 6, 1, 2, 3)
result=func(y2=5, x2=4, z2=6, x1=1, y1=2, z1=3)
```

- 由于调用函数时指定了参数名称，所以参数之间的顺序可以任意调整

函数的返回值

- return语句：程序退出该函数，并返回到函数被调用的地方
- return语句返回的值传递给调用程序
- Python函数的返回值有两种形式：
  - ✓ 没有返回值
  - ✓ 返回一个或多个值

函数的返回值

- 无返回值的return语句等价于return None
- None是表示没有任何东西的特殊类型

```
def happy():
    print("Happy Birthday to you!")
```

- 等价于：

```
def happy():
    print("Happy Birthday to you!")
    return None
```

函数的返回值

- 返回值可以是一个变量，也可以是一个表达式
  - ✓ #square

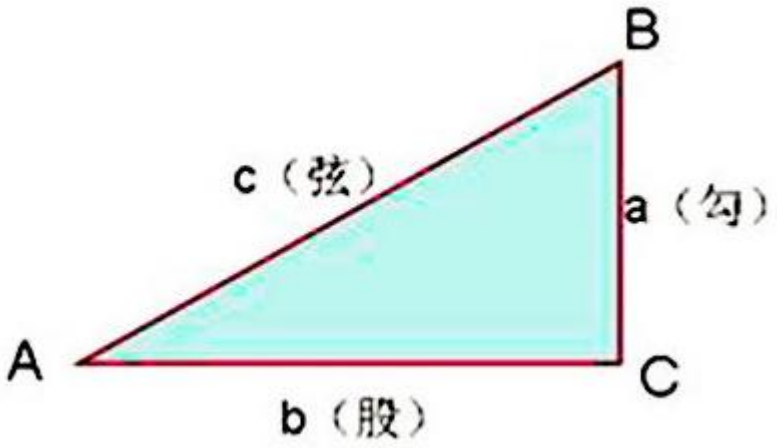
```
def square(x):
    y=x*x
    return y
```

```
def square(x):
    return x*x
```

函数的返回值

- 应用square()函数编写程序计算两点之间的距离
- 数学模型：给定两点坐标（x1, y1）和（x2, y2），由勾股定理，两点间距离公式为：

$$D = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}$$



函数的返回值

- 计算两点距离的函数代码：
  - ✓ #distance

distance.py - F:/北邮课程/python程序设计/备课/第三周/distance.py (3.6.4)

File Edit Format Run Options Window Help

```
#计算两点之间的距离
import math

def square(x):
    y=x*x
    return y

def distance(x1,y1,x2,y2):
    dist=math.sqrt(square(x1-x2)+square(y1-y2))
    return dist

distance(0,0,1,1)
```

函数的返回值

- 应用distance()编写程序计算三角形周长
  - ✓ #triangle
  - ✓ 给定三个坐标点，如何判断是否是三角形？

函数的返回值

triangle.py - F:/北邮课程/python程序设计/备课/第三周/triangle.py (3.6.4)

File Edit Format Run Options Window Help

```
#计算三角形周长
import math

def square(x):
    y=x*x
    return y

def distance(x1,y1,x2,y2):
    dist=math.sqrt(square(x1-x2)+square(y1-y2))
    return dist

def isTriangle(x1,y1,x2,y2,x3,y3):
    flag=((x1-x2)*(y3-y2)-(x3-x2)*(y1-y2))!=0#判断是否在一条直线上
    return flag

def main():
    print("Please enter(x,y) of three points in turn:")
    x1,y1=eval(input("Point1:(x,y)="))
    x2,y2=eval(input("Point1:(x,y)="))
    x3,y3=eval(input("Point1:(x,y)="))

    if (isTriangle(x1,y1,x2,y2,x3,y3)):
        perim=distance(x1,y1,x2,y2)
        +distance(x2,y2,x3,y3)
        +distance(x1,y1,x3,y3)
        print("The perimeter of the triangle is",perim)
    else:
        print("Kindding me? This is not a triangle")

main()
```

函数的返回值

- 程序同一行语句中distance()被调用了三次，用来计算三角形的周长
- 使用函数解决了代码的复用问题

函数的返回值

- 使用return语句返回多个值
  - ✓ 计算两个数的加、减、乘、除
  - ✓ #calculate

函数的返回值

calculate.py - F:/北邮课程/python程序设计/备课/第三周/calculate.py (3.6.4)

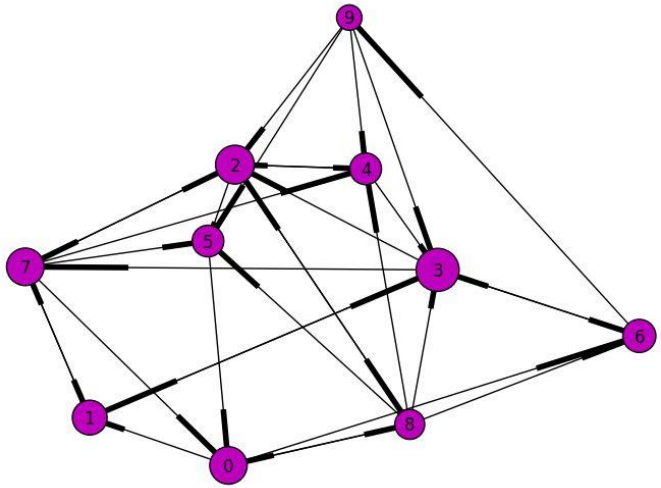
File Edit Format Run Options Window Help

```
#计算两个数加减乘除
def calculate(x,y):
    sum=x+y
    diff=x-y
    pro=x*y
    quo=x/y
    return sum,diff,pro,quo

def main():
    num1,num2=eval(input("Please enter two numbers:"))
    s,d,p,q=calculate(num1,num2)
    print("The sum is",s,"\n"
          "The difference is",d,"\n"
          "The product is",p,"\n"
          "The quotient  is",q)

main()
```

函数的返回值

- 对于多返回值的函数，根据变量的位置来赋值
  - ✓ s将获得return的第一个返回值sum
  - ✓ d将获得第二个返回值diff
  - ✓ …

lambda函数



lambda函数

- Python的有33个保留字，其中一个是lambda，该保留字用于定义一种特殊的函数——匿名函数，又称lambda函数
- 匿名函数并非没有名字，而是将函数名作为函数结果返回，如下：

  <函数名> = lambda <参数列表>:<表达式>

- lambda函数与正常函数一样，等价于下面形式：

```
def <函数名>(<参数列表>):
    return <表达式>
```

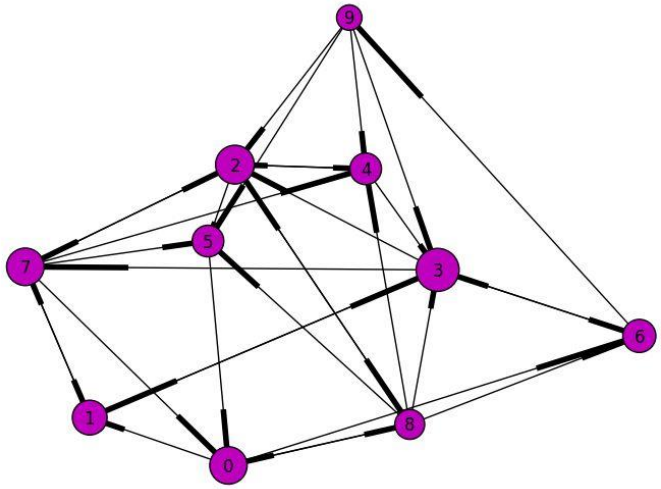
lambda函数

- 简单说，lambda函数用于定义简单的、能够在一行内表示的函数，返回一个函数类型，实例如下：

```
f=lambda x,y:x+y
```

```
Python 3.6.4 (v3.6.4:d48eceb
 on win32
Type "copyright", "credits"
>>> f=lambda x,y:x+y
>>> type(f)
<class 'function'>
>>> f(1,2)
3
>>>
```

函数对变量的作用



函数对变量的作用

- 一个程序中的变量包括两类：全局变量和局部变量
- 全局变量指在函数之外定义的变量，一般没有缩进，在程序执行全过程有效
- 局部变量指在函数内部使用的变量，仅在函数内部有效，当函数退出时变量将不存在

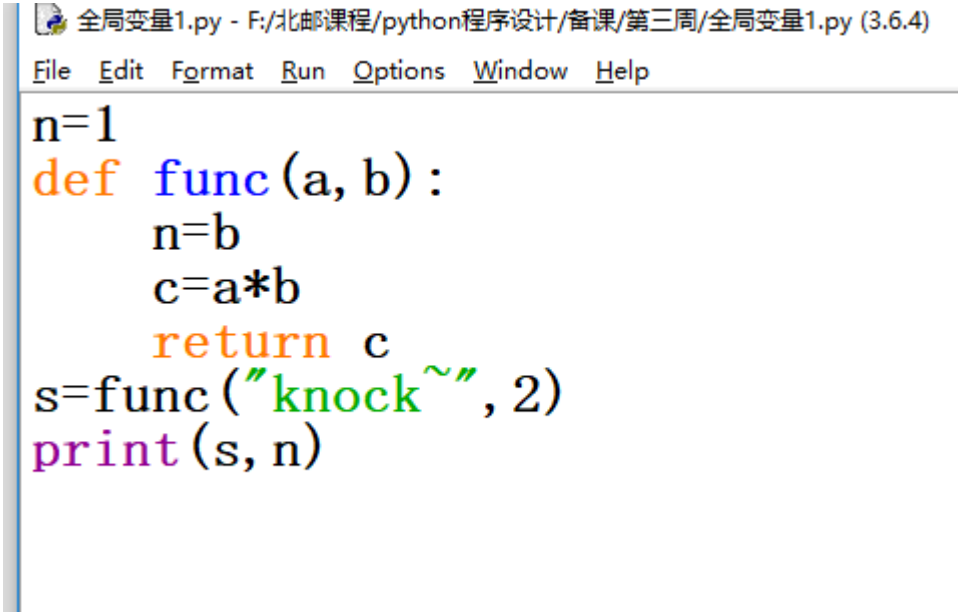
函数对变量的作用

- 当函数执行完退出后，其内部变量将被释放
  - ✓ #全局变量

```
File  Edit  Format  Run  Options  Window  Help
n=1
def func(a, b):
    c=a*b
    return c
func("knock~", 2)
print(n)
print(c)
```

函数对变量的作用

- 如果函数内部使用了全局变量呢？函数func()内部使用了变量n，并且将变量参数b赋值给变量n，为何全局变量n值没有改变？
  - ✓ #全局变量1



函数对变量的作用

- 如果希望让func()函数将n当作全局变量，需要在变量n使用前显式声明该变量为全局变量？
  - ✓ #全局变量2

全局变量2.py - F:/北邮课程/python程序设计/备课/第三周/全局变量2.py (3.6.4)

File Edit Format Run Options Window Help

```
n=1
def func(a,b):
    global n
    n=b
    c=a*b
    return c
s=func("knock~",2)
print(s,n)
```

函数对变量的作用

- 如果此时的全局变量不是整数n，而是列表类型ls，会怎么样呢？
  - ✓ #全局变量3

全局变量3.py - F:/北邮课程/python程序设计/备课/第三周/全局变量3.py (3.6.4)

File Edit Format Run Options Window Help

```
ls=[]    #ls是全局列表变量
def func(a,b):
    ls.append(b)      #将局部变量b增加到全局列表变量ls中
    c=a*b
    return c
s=func("knock~",2)
print(s,ls)
```

函数对变量的作用

File Edit Format Run Options Window Help

```
ls=[]    #ls是全局列表变量
def func(a,b):
    ls.append(b)      #将局部变量b增加到全局列表变量ls中
    c=a*b
    return c
s=func("knock~",2)
print(s,ls)
```

- 与之前的整数变量n不同，全局列表变量在函数func()调用后竟然发生了改变！
- func()函数的ls.append(b)执行语句时需要一个真实创建的列表，此时func()函数专属的内存空间中没有已经创建过且名称为ls的列表，因此func()函数会进一步去寻找全局内存空间，自动关联全局ls列表，并修改其内容
- 对于列表类型，函数可以直接使用全局列表而不需要采用global进行声明

函数对变量的作用

- 如果func()函数内部存在一个真实创建过且名称为ls的列表，则func()将操作该列表而不会修改全局变量
  - ✓ #全局变量4

全局变量4.py - F:/北邮课程/python程序设计/备课/第三周/全局变量4.py (3.6.4)

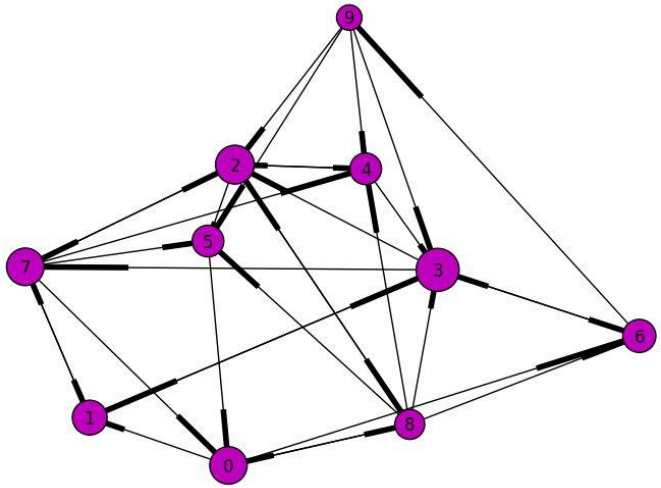
File Edit Format Run Options Window Help

```
ls=[]    #ls是全局列表变量
def func(a,b):
    ls=[]
    ls.append(b)      #将局部变量b增加到全局列表变量ls中
    c=a*b
    return c
s=func("knock~",2)
print(s,ls)
```

函数对变量的作用

- Python函数对变量的作用遵守如下原则：
  - ✓ 简单数据类型变量无论是否与全局变量重名，仅在函数内部创建和使用，函数退出后变量被释放
  - ✓ 简单数据类型变量在用global保留字声明后，作为全局变量
  - ✓ 对于组合数据类型的全局变量，如果在函数内部没有被真实创建的同名变量，则函数内部可直接使用并修改全局变量的值
  - ✓ 如果函数内部真实创建了组合数据类型变量，无论是否有同名全局变量，函数仅对局部变量进行操作

函数和递归



函数和递归

- 函数可以简化程序，函数可以使程序模块化
- 用函数将较长的程序分割成短小的程序段，可以方便理解
- 使用函数的思想编写程序，可以大大增加程序的模块化程度

函数和递归

- 递归：函数定义中使用函数自身的方法
- 递归在数学和计算机应用中非常强大，能够非常简洁的解决重要问题
- 经典例子：阶乘

$$n! = n(n-1)(n-2)\ldots(1)$$

$$5! = 5(4)(3)(2)(1) = 5 * 4!$$

$$n! = n(n-1)!$$

函数和递归

- 阶乘的递归定义：

$$n! = \begin{cases} 1 & n = 0 \\ n(n-1)! & otherwise \end{cases}$$

- 0的阶乘：定义为1
- 其他数字：定义为这个数字乘以比这个数字小1的数的阶乘

函数和递归

- 递归不是循环
- 最后计算基例：0！=1
- 递归定义特征：
  - ✓ 有一个或多个基例是不需要再次递归的
  - ✓ 所有的递归链都要以一个基例结尾

函数和递归

- 通过一个累计器循环计算阶乘
- 阶乘的递归定义函数 ：
  - ✓ #递归

递归.py - F:/北邮课程/python程序设计/备课/第三周/递归.py (3.6.4)

File Edit Format Run Options Window Help

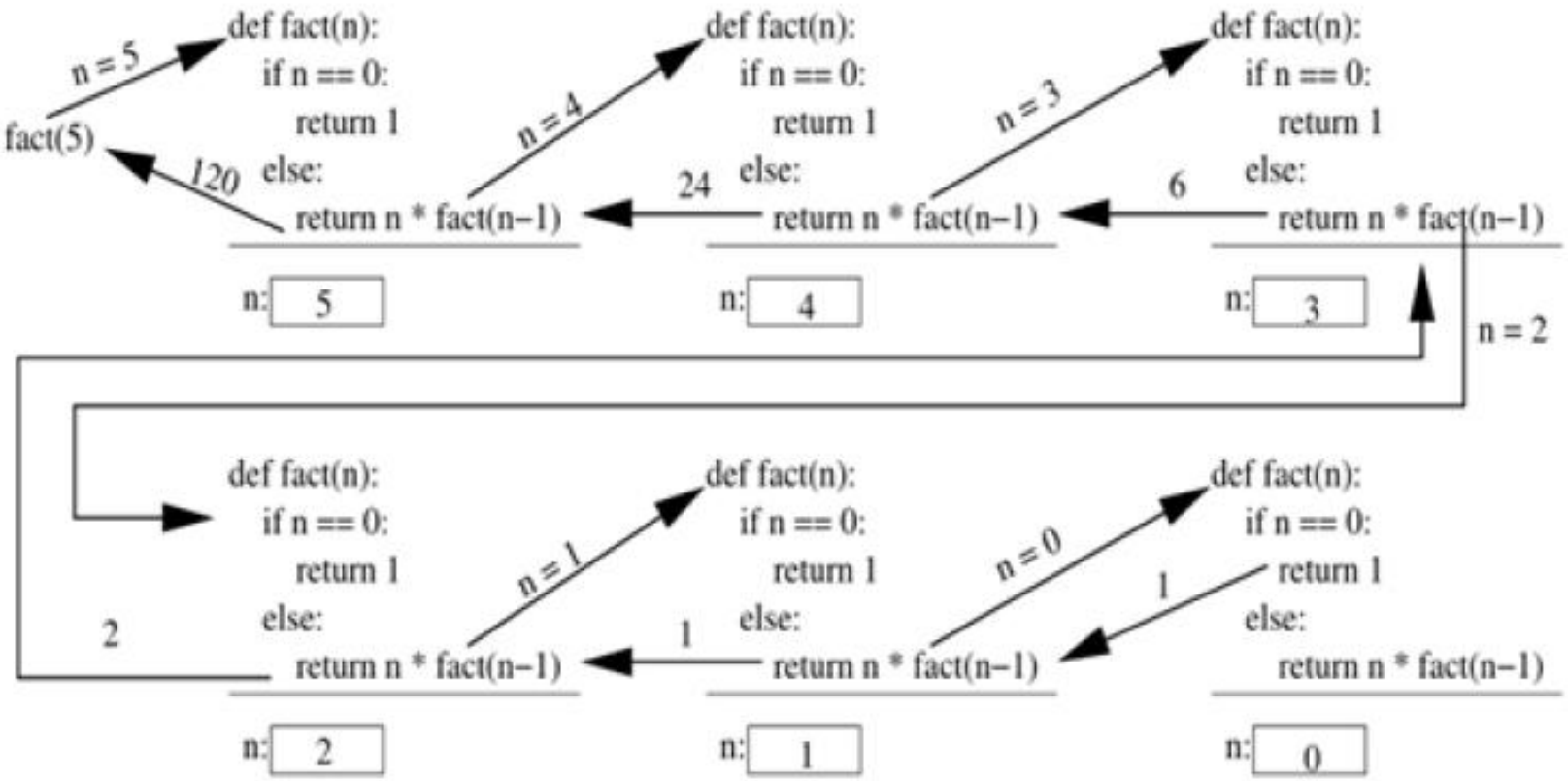
```
def fact(n):
    if n==0:
        return 1
    else:
        return n*fact(n-1)

fact(10)
```

函数和递归

- 递归每次调用都会引起新函数的开始
- 递归有本地值的副本，包括该值的参数
- 阶乘递归函数中：每次函数调用中的相关n值在中途的递归链暂时存储，并在函数返回时使用

函数和递归

- 5！的递归调用过程图



函数和递归

- 字符串反转
  - ✓ 方法1：字符串转换为字符列表，反转列表，列表转换回字符串
  - ✓ 方法2：切片
  - ✓ 方法3：递归

函数和递归

- 字符串反转
  - ✓ 方法1：字符串转换为字符列表，反转列表，列表转换回字符串
  - ✓ #reverse string1

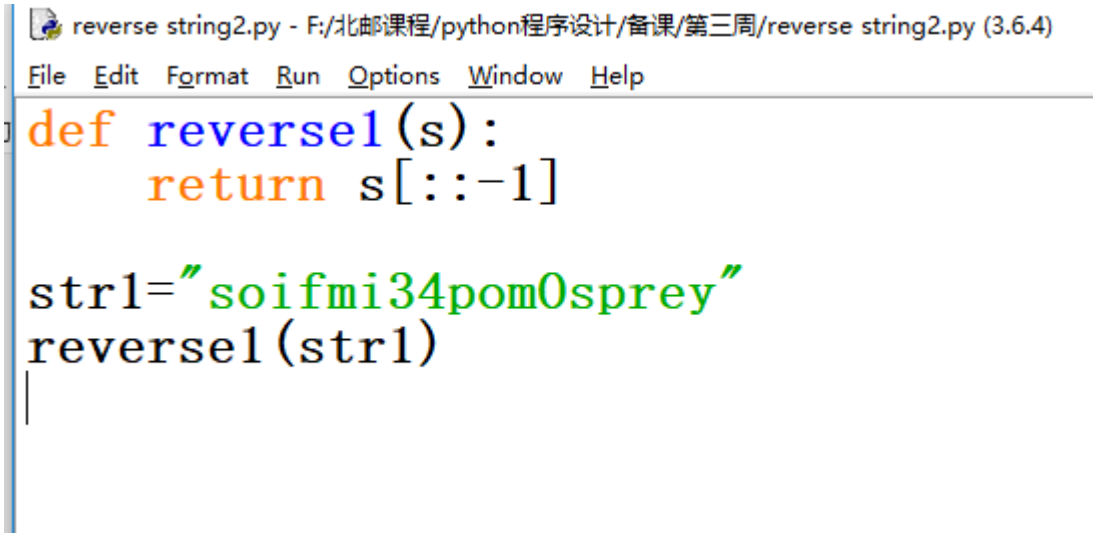
reverse string.py - F:/北邮课程/python程序设计/备课/第三周/reverse str

File Edit Format Run Options Window Help

```
def reverse1(s):
    ls=list(s)
    ls.reverse()
    s="".join(ls)
    print(s)
str1="soifmi34pomOsprey"
reverse1(str1)
```

函数和递归

- 字符串反转
  - ✓ 方法2：切片法
  - ✓ #reverse string2



函数和递归

- 字符串反转
  - ✓ 方法3：递归
  - ✓ 将字符串分割成首字符和剩余子字符串
  - ✓ 反转了剩余部分后把首字符放到末尾，整个字符串反转就完成了

函数和递归

- 字符串反转
  - ✓ 方法3：递归
  - ✓ 字符串反转算法：

reverse string.py - F:\北邮课程\python程序设计\备课\第三周\reverse string.py (3.6.4)

File Edit Format Run Options Window Help

```
def reverse(s):
    return reverse(s[1:])+s[0]
str1="soifmi34pomOsprey"
reverse(str1)
```

函数和递归

- 构造递归函数，需要基例
- 基例不进行递归，否则递归就会无限循环执行
- Python在900余次调用之后，到达默认的“递归深度的最大值”，终止调用
- 此递归调用以字符串形式执行，应设置基例为空串

函数和递归

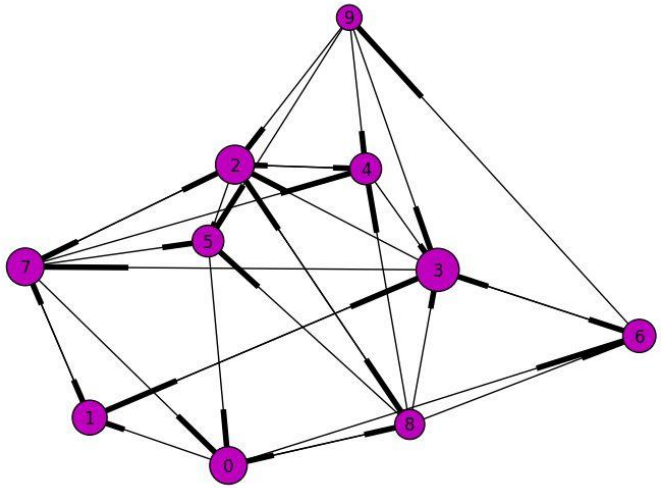
- 正确的字符串反转代码：

reverse string3.py - F:/北邮课程/python程序设计/备课/第三周/reverse string3.py (3.6.4)

File Edit Format Run Options Window Help

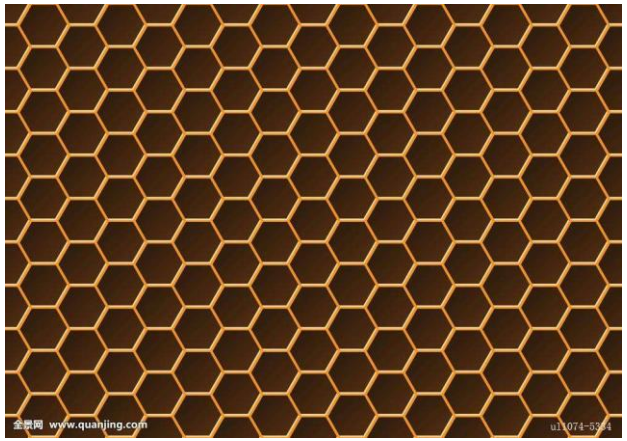
```
def reverse(s):
    if s=="":
        return s
    else:
        return reverse(s[1:])+s[0]
str1="soifmi34pomOsprey"
reverse(str1)
```

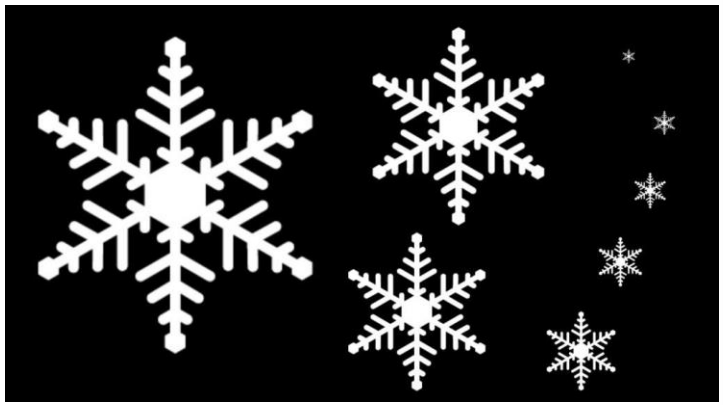
科赫曲线绘制

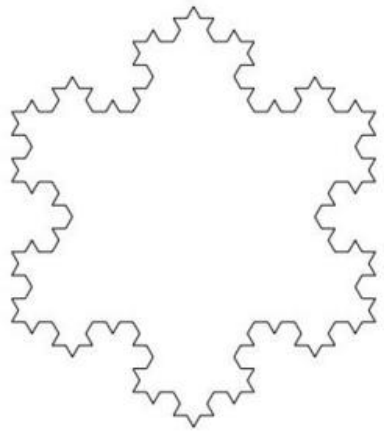


科赫曲线

- 自然界有很多图形很规则，符合一定的数学规律，例如，蜜蜂蜂窝是天然的等边六角形等。科赫(Koch)曲线在众多经典数学曲线中非常著名，由瑞典数学家冯·科赫(H·V·Koch)于1904年提出，由于其形状类似雪花，也被称为雪花曲线

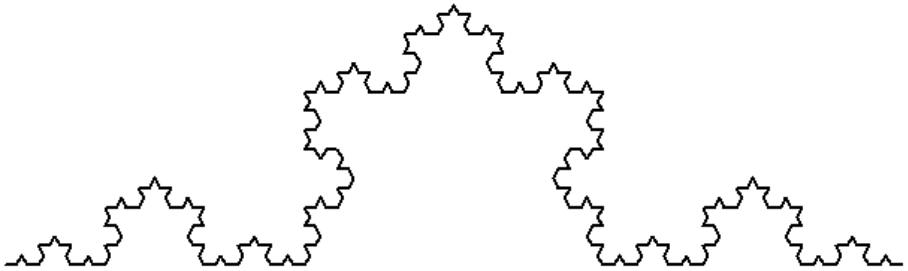






科赫曲线

- 正整数n代表科赫曲线的阶数，表示生成科赫曲线过程的操作次数，科赫曲线初始化阶数为0，表示一个长度为L的直线；对于直线L，将其等分为三段，中间一段用边长为L/3的等边三角形的两个边替代，得到1阶科赫曲线，它包含四条线段；进一步对每条线段重复同样的操作后得到2阶科赫曲线。继续重复同样的操作n次可以得到n阶科赫曲线



科赫曲线

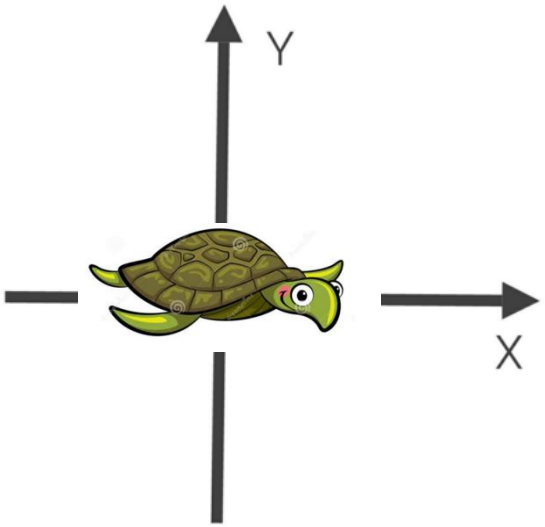
- 科赫曲线属于分形几何分支，它的绘制过程体现了递归思想

科赫曲线

- Turtle 库
  - ✓ Python内置图形化模块，绘制图像的函数库
- Turtle.py文件
  - ✓ 安装目录的Lib文件夹下
- import turtle
  - ✓ import是一个关键字，用来引入一些外部库，这里的含义是引入一个名字叫turtle的函数库
  - ✓ 或者from turtle import *

科赫曲线

- 使用turtle库，需要有这样一个概念：
  - ✓ 想象一个小海龟，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



科赫曲线

- 控制画笔绘制状态方法

方法名称	方法含义
pendown()	放下画笔，移到指定点后继续绘制
penup()	提起画笔，用于另起一个地方绘制时用，与pendown()配对使用
pensize(width)	设置画笔线条的粗细为指定大小

科赫曲线

画笔运动方法

方法名称	方法含义
forward()	沿着当前方向前进指定距离
backward()	沿着当前相反方向后退指定距离
right(angle)	向右旋转angle角度
left(angle)	向左旋转angle角度
goto(x,y)	移动到绝对坐标（x,y）处
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。
circle()	绘制一个指定半径，角度、以及步骤的圆圈
dot(r)	绘制一个指定直径和颜色的圆点
undo()	撤销画笔最后一步动作
speed()	设置画笔的绘制速度，参数为0-10之间

科赫曲线

- 颜色和字体绘制方法

方法名称	方法含义
color()	设置画笔的颜色
begin_fill()	填充图形前，调用该方法
end_fill()	填充图形结束
filling()	返回填充的状态，True为填充，False为未填充
clear()	清空当前窗口，但不改变当前画笔的位置
reset()	清空当前窗口，并重置位置等状态为默认值
screensize()	设置画布的长和宽
hideturtle()	隐藏画笔的turtle形状
showturtle()	显示画笔的turtle形状
isvisible()	如果turtle可见，则返回True
wirte()	输出font字体的字符串

科赫曲线

- 绘制五角星
  - ✓ #五角星

科赫曲线

五角星.py - F:\北邮课程\python程序设计\备课\第三周\五角星.py (3.6.4)

File Edit Format Run Options Window Help

```
import turtle
turtle.setup(800,600)#窗口的大小
turtle.speed(1)#画笔速度
turtle.pensize(3)#画笔粗细
turtle.begin_fill()#填充图形前，调用该方法
turtle.color("red")
turtle.penup()
turtle.goto(-250,50)
turtle.pendown()
for i in range(5):
    turtle.forward(500)#沿着当前方向前进指定距离
    turtle.right(144)#向右旋转144度
turtle.end_fill()
turtle.hideturtle()#隐藏箭头
```

科赫曲线

- 科赫曲线

科赫曲线

- 绘制科赫曲线
  - ✓ #科赫曲线

File Edit Format Run Options Window Help

```
#DrawKoch.py
import turtle
def koch(size, n):
    if n == 0:
        turtle.forward(size)
    else:
        for angle in [0, 60, -120, 60]:
            turtle.left(angle)#向左旋转角度
            koch(size/3, n-1)
def main():
    turtle.setup(800,400)#窗口的大小
    turtle.speed(0)   #控制绘制速度
    turtle.penup()
    turtle.goto(-300, -50)
    turtle.pendown()
    turtle.pensize(2)
    koch(600,0)       # 0阶科赫曲线长度，阶数
    turtle.hideturtle()#隐藏箭头
main()
```

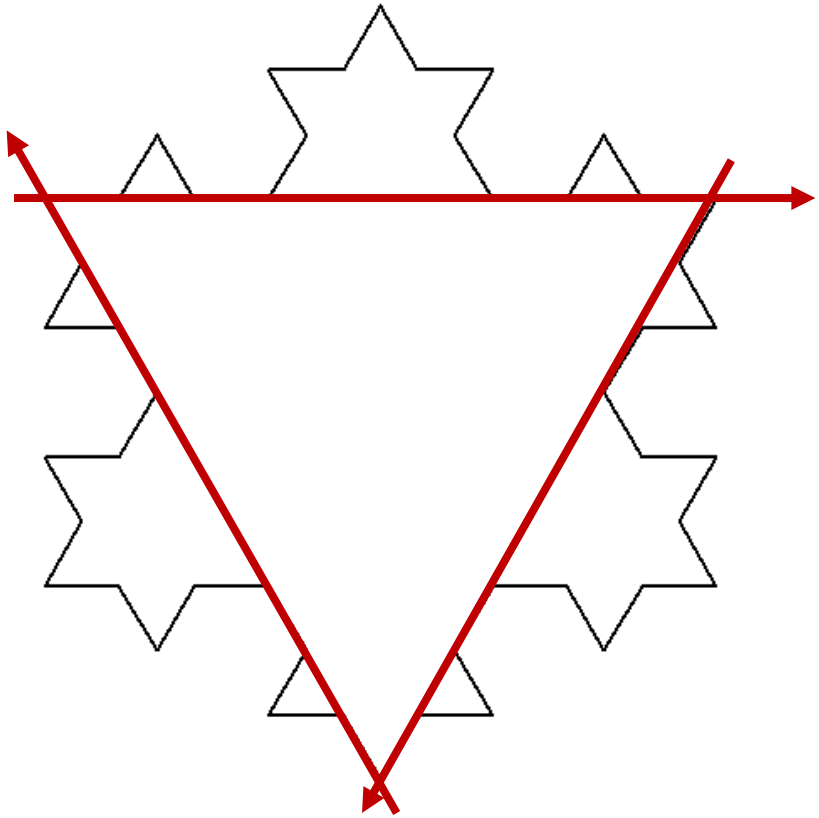
科赫曲线

- 绘制科赫曲线
  - ✓ #科赫雪花

科赫雪花.py - F:/北邮课程/python程序设计/备课/第三周/科赫雪花.py (3.6.4)

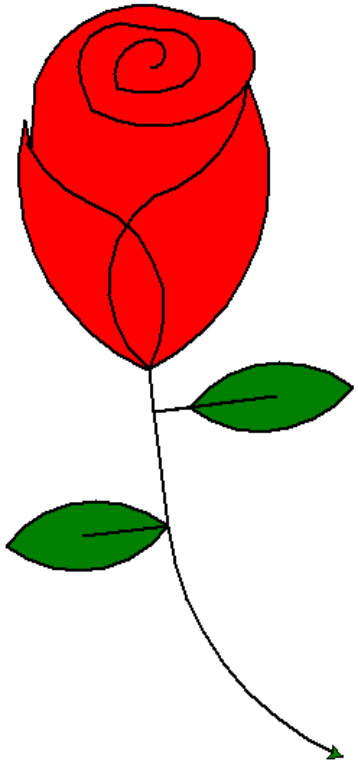
File Edit Format Run Options Window Help

```
#DrawKoch.py
import turtle
def koch(size, n):
    if n == 0:
        turtle.fd(size)
    else:
        for angle in [0, 60, -120, 60]:
            turtle.left(angle)
            koch(size/3, n-1)
def main():
    turtle.setup(600,600)
    turtle.speed(0)
    turtle.penup()
    turtle.goto(-200, 100)
    turtle.pendown()
    turtle.pensize(2)
    level = 2
    koch(400,level)
    turtle.right(120)
    koch(400,level)
    turtle.right(120)
    koch(400,level)
    turtle.hideturtle()
main()
```

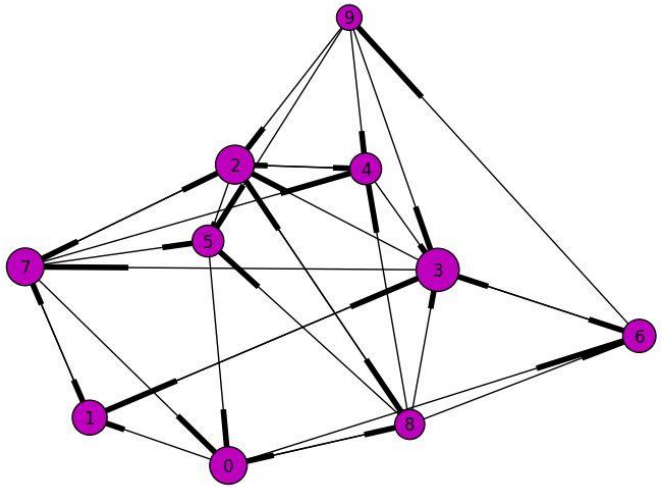


科赫曲线

- 绘制玫瑰



Python内置函数和函数库



Python内置函数和函数库

- Python解释器提供了68个内置函数，这些函数不需要引用库直接使用

abs()	id()	round()	compile()	locals()
all()	input()	set()	dir()	map()
any()	int()	sorted()	exec()	memoryview()
asci()	len()	str()	enumerate()	next()
bin()	list()	tuple()	filter()	object()
bool()	max()	type()	format()	property()
chr()	min()	zip()	frozenset()	repr()
complex()	oct()		getattr()	setattr()
dict()	open()		globals()	slice()
divmod()	ord()	bytes()	hasattr()	staticmethod()
eval()	pow()	delattr()	help()	sum()
float()	print()	bytearray()	isinstance()	super()
hash()	range()	callable()	issubclass()	vars()
hex()	reversed()	classmethod()	iter()	__import()__

Python内置函数和函数库

- Python解释器提供几百个内置函数库，此外，世界各地程序员通过开源社区贡献了十几万个第三方函数库，几乎覆盖了计算机技术的各个领域，编写Python程序可以大量利用已有的内置或第三方代码，具备良好的编程生态

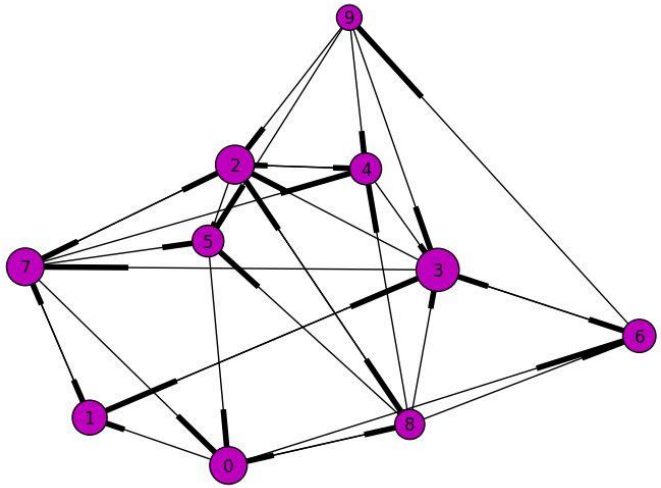
Python内置函数和函数库

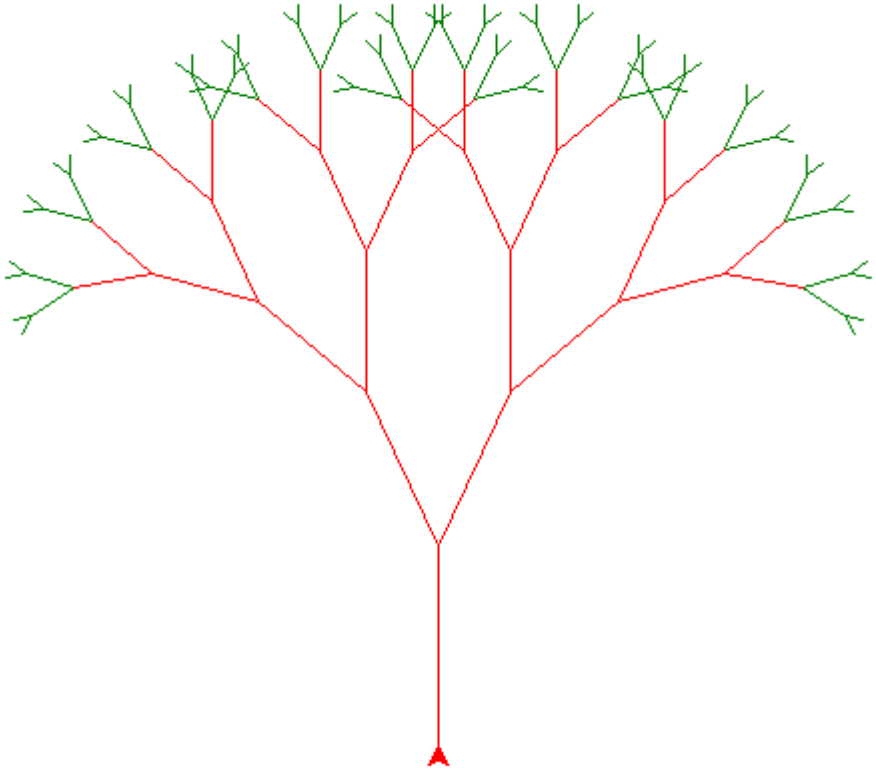
- math库：
  - ✓ math库是Python提供的内置数学类函数库
  - ✓ math库一共提供了4个数学常数和44个函数
- math库的引用
  - ✓ 第一种：import math 对math库中函数采用math.<b>()形式使用
  - ✓ 第二种：from math import <函数名> 对math库中函数可以直接采用<函数名>()形式使用

```
>>> import math
>>> math.fabs(-10)
10.0
```

```
>>> from math import fabs
>>> fabs(-10)
10.0
>>>
```

课后思考题





分形树

谢谢