1. 函数和代码复用

函数的理解和定义

函数是一段具有特定功能的、可重用的语句组。

函数是一种功能的抽象，一般函数表达特定功能。

两个作用：降低编程难度和代码复用。

函数的定义：函数是一段代码的表示。

def <函数名>(<参数(0个或多个)>):

<函数体>

return <返回值>

函数定义时，所指定的参数是一种占位符。

函数定义后，如果不经过调用，不会被执行。

函数定义时，参数是输入，函数体的处理、结果是输出()

函数的使用及调用过程

调用是运行函数代码的方式。

调用时要给出实际参数。

实际参数替换定义中的参数。

函数调用后得到返回值。

函数的参数传递

函数可以有参数，也可以没有，但必须保留括号。

函数定义时可以为某些参数指定默认值，构成可选参数。

def <函数名>(<非可选参数>,<可选参数>):

<函数体>

return<返回值>

函数定义时可以设计可变数量参数，即不确定参数总数量

def <函数名>(<参数>,\*b ):

<函数体>

return <返回值>

参数传递的两种方式：

函数调用时，参数可以按照位置或名称方式传递。

函数的返回值

函数可以返回0个或多个结果。

return保留字用来传递返回值。

函数可以有返回值，也可以没有，可以有return，也可以没有。

return可以传递0个返回值，也可以传递任意多个返回值。

局部变量和全局变量



在函数体内的叫局部变量。

使用规则;

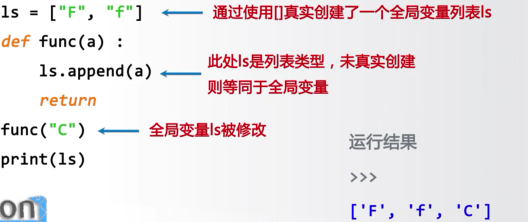
1. 局部变量和全局变量是不同变量。

局部变量是函数内部的占位符，与全局变量可能重名但不同。

函数运算结束后，局部变量被释放。

可以使用global保留字在函数内部使用全局变量。

2、局部变量为组合数据类型且未创建，等同于全局变量。



基本数据类型，无论是否重名，局部变量与全局变量不同。

可以通过global保留字在函数内部声明全局变量。

组合数据类型，如果局部变量未真实创建，则是全局变量。

lambda函数

lambda函数是一种匿名函数，即没有名字的函数。

使用lambda保留字定义，函数名是返回结果。

lambda函数用于定义简单的、能够在一行内表示的函数。

<函数名>=lambda <参数>:<表达式>

等价于

def <函数名>(<参数>):

<函数体>

return <返回值>

谨慎使用lambda函数。

lambda函数主要用作一些特定函数或方法的参数。

Lambda函数有一些固定使用方式。一般情况下，建议使用def定义的普通函数。

实例：七段数码管绘制

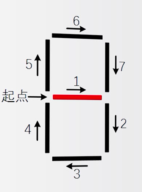
需求：用程序绘制七段数码管。

基本思路：步骤1：绘制单个数字对应的数码管。

七段数码管由7个基本线条组成；

七段数码管可以有固定顺序；

不同数字显示不同线条。



步骤2：获得一串数字，绘制对应的数码管。

import turtle

#绘制单段数码管

def drawLine(draw):

turtle.pendown() if draw else turtle.penup()

turtle.fd(40)

turtle.right(90)

#根据数字绘制七段数码管

def drawDigit(digit):

drawLine(True) if digit in [2,3,4,5,6,8,9] else drawLine(False)

drawLine(True) if digit in [0,1,3,4,5,6,7,8,9] else drawLine(False)

drawLine(True) if digit in [0,2,3,5,6,8,9] else drawLine(False)

drawLine(True) if digit in [0,2,6,8] else drawLine(False)

turtle.left(90)

drawLine(True) if digit in [0,4,5,6,8,9] else drawLine(False)

drawLine(True) if digit in [0,2,3,5,6,7,8,9] else drawLine(False)

drawLine(True) if digit in [0,1,2,3,4,7,8,9] else drawLine(False)

turtle.left(180)

turtle.penup()#为绘制后续数字确定位置。

turtle.fd(20)#为绘制后续数字确定位置

#获得要输出的数字

def drawDate(date):

for i in date:

drawDigit(eval(i))#通过eval()函数将数字变为整数

def main():

turtle.setup(800,350,200,200)

turtle.penup()

turtle.fd(-300)

turtle.pensize(5)

drawDate('20181010')

turtle.hideturtle()

turtle.done()

main()

步骤3：获得当前系统时间，绘制对应的数码管。

import turtle,time

def drawGap():

turtle.penup()

turtle.fd(5)

#绘制单段数码管

def drawLine(draw):

drawGap()

turtle.pendown() if draw else turtle.penup()

turtle.fd(40)

turtle.right(90)

#根据数字绘制七段数码管

def drawDigit(digit):

drawLine(True) if digit in [2,3,4,5,6,8,9] else drawLine(False)

drawLine(True) if digit in [0,1,3,4,5,6,7,8,9] else drawLine(False)

drawLine(True) if digit in [0,2,3,5,6,8,9] else drawLine(False)

drawLine(True) if digit in [0,2,6,8] else drawLine(False)

turtle.left(90)

drawLine(True) if digit in [0,4,5,6,8,9] else drawLine(False)

drawLine(True) if digit in [0,2,3,5,6,7,8,9] else drawLine(False)

drawLine(True) if digit in [0,1,2,3,4,7,8,9] else drawLine(False)

turtle.left(180)

turtle.penup()#为绘制后续数字确定位置。

turtle.fd(20)#为绘制后续数字确定位置

#获得要输出的数字

def drawDate(date):#date为日期，格式为'%Y-%m=%d+'

turtle.pencolor("red")

for i in date:

if i == '-':

turtle.write('年',font=("Arial",18,"normal"))

turtle.pencolor("green")

turtle.fd(40)

elif i == '=':

turtle.write('月',font=("Arial",18,"normal"))

turtle.pencolor("blue")

turtle.fd(40)

elif i == '+':

turtle.write('日',font=("Arial",18,"normal"))

else:

drawDigit(eval(i))#通过eval()函数将数字变为整数

def main():

turtle.setup(800,350,200,200)

turtle.penup()

turtle.fd(-300)

turtle.pensize(5)

drawDate(time.strftime('%Y-%m=%d+',time.gmtime()))

turtle.hideturtle()

turtle.done()

main()

代码复用与模块化设计

把代码当成资源进行抽象。

代码资源化;程序代码是一种用来表达计算的“资源”

代码抽象化：使用函数等方法对代码赋予更高级别的定义。

代码复用：同一份代码在需要时可以被重复使用。

函数和对象是代码复用的两种主要形式。

函数：将代码命名，在代码层面建立了初步抽象。

对象：属性和方法。<a>.<b> <a>.<b>() 在函数之上再次组织进行抽象。

模块化设计

即分而治之。——通过函数或对象封装将程序划分为模块及模块间的表达。

具体包括：主程序、子程序和子程序间关系。

分而治之：一种分而治之、分层抽象、体系化的设计思想。

紧耦合 松耦合

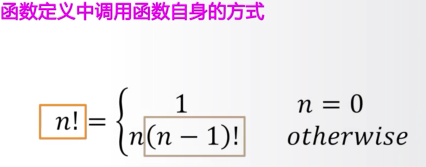
紧耦合：两个部分之间交流很多，无法独立存在。

松耦合：两个部分之间交流很少，可以独立存在。

模块内部紧耦合、模块之间松耦合。

函数递归的理解

函数定义中调用函数自身的方式。



递归的两个关键特征：

链条：计算过程存在递归链条。

基例：存在一个或多个不需要再次递归的基例。

类似数学归纳法

数学归纳法：

证明当n取第一个值n0时命题成立。

假设当nk时命题成立，证明当n=nk+1时命题也成立。

递归是数学归纳法思维的编程体现。

函数递归地调用过程

递归的实现：函数+分支语句。

递归本身是一个函数，需要函数定义方式描述。

函数内部，采用分支语句对输入参数进行判断。

基例和链条，分别编写对应代码。

函数递归实例解析。

步骤1：函数+分支结构

步骤2：递归链条

步骤3：递归基例

1. 将字符串s反转后输出。

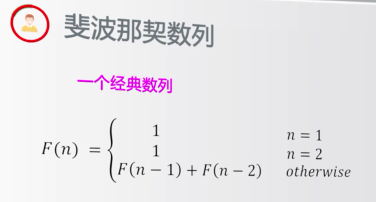
def rvs(s):

if s=="":

return s

else:

return rvs(s[1:])+s[0]



def f(n):

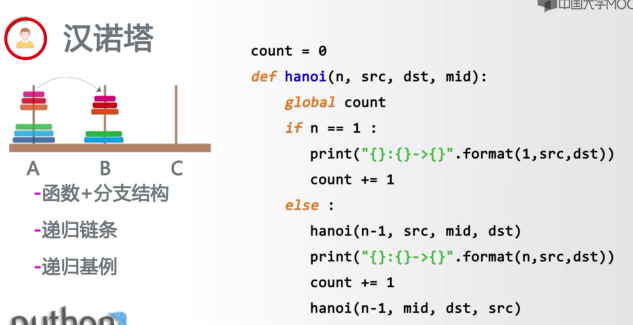
if n==1 or n==2:

return 1

else:

return f(n-1)+f(n+2)

3、



count=0

def hanoi(n,src,dst,mid):

global count

if n ==1:

print("{}:{}->{}".format(1,src,dst))

count +=1

else:

hanoi(n-1,src,mid,dst)

print("{}:{}->{}".format(n,src,dst))

count +=1

hanoi(n-1,mid,dst,src)

hanoi(3,"A","C","B")

print(count)

模块：PyInstaller库的使用

将.py源代码转换成无需源代码的可执行文件



PyInstaller库是第三方库。

官方网站：<http://www.pyinstaller.org>

第三方库：使用前需要额外安装。

安装第三方库需要使用pip工具。

（cmd命令行）pip install pyinstaller

PyInstaller库的使用说明：

简单的使用：

(cmd命令行) pyinstaller-F<文件名.py>

PyInstaller库常用参数：

|  |  |
| --- | --- |
| 参数 | 描述 |
| -h | 查看帮助 |
| -clean | 清理打包过程中的临时文件 |
| -D，--onedir | 默认值，生成dist文件夹 |
| -F，--onefile | 在dist文件夹中只生成独立的打包文件 |
| -i<图标文件名.ico> | 指定打包程序使用的图标（icon）文件 |

实例：科赫雪花小包裹

分形几何：是一种迭代的几何图形，广泛存在于自然界中。

科赫曲线，也叫雪花曲线。



科赫曲线：

#KochDraw1.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(800,400)

turtle.penup()

turtle.goto(-300,-50)

turtle.pendown()

turtle.pensize(2)

koch(600,3) # 3阶科赫曲线，阶数

turtle.hideturtle()

main()

科赫雪花：

#KochDraw2.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.penup()

turtle.goto(-200,100)

turtle.pendown()

turtle.pensize(2)

level=3 # 3阶科赫雪花，阶数

koch(400,level)

turtle.right(120)

koch(400,level)

turtle.right(120)

koch(400,level)

turtle.hideturtle()

main()