第五周：函数和代码复用

1.函数的定义与使用

1.函数的理解与定义

函数是一段具有特定功能的、可重用的语句组，它是一种功能的抽象，一般函数表达特定功能。它有两个作用：**降低编程难度**和**代码复用**。

定义方式：

```
def <name>(parameters)
    <function body>
    return <return value>
```

2.函数的使用及调用过程

不整理

3.函数的参数传递

函数定义时可为某些参数指定默认值，构成可选参数，**可选参数必须在非可选参数之后**。

对于参数不确定的情况下，函数定义时可以设计可变数量参数，即不确定参数总数量。格式如下，其中*b（其他名字也行）代表后面所有不确定的参数：

```
def <name>(<固定参数>,*b):
    <function body>
    return <value>
```

参数传递有两种方式：**按照位置传递**和**按照名称传递**。

4.函数的返回值

不整理

5.局部变量与全局变量

规则 1：局部变量和全局变量是不同的变量。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。若使用全局变量，则使用前面加上 global。

规则 2：局部变量为组合数据类型且未创建，等同于全局变量。例如：

```
ls = ["F","f"]#通过使用[]真是创建了一个全局变量列表 ls
def func(a):
    ls.append(a)#此处 ls 是列表类型，未真实创建，等同于全局变量
    return
func("C")
print(ls)
```

6.lambda 函数

lambda 函数返回函数名作为结果，它是一种匿名函数。使用方式：

```
<function name> = lambda<patameters>:<expresstion function>
```

例如：

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

谨慎使用 lambda 函数，它主要用作一些特定函数或方法的参数，它有一些固定使用方式。

2.实例 7：七段数码管绘制

代码 1：

```
import turtle
def drawLine(draw):     #绘制单段数码管
    turtle.pendown() if draw else turtle.penup()
    turtle.fd(40)
    turtle.right(90)
def drawDigit(digit): #根据数字绘制七段数码管
    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()
    turtle.fd(20)
def drawDate(date):    #获得要输出的数字
    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('20180411')
    turtle.hideturtle()
    turtle.done()
main()
```

结果：

图 1

代码 2：

```
import turtle, time
def drawGap(): #绘制数码管间隔
    turtle.penup()
    turtle.fd(5)
def drawLine(draw):     #绘制单段数码管
    drawGap()
    turtle.pendown() if draw else turtle.penup()
```

```
    turtle.fd(40)
    drawGap()
    turtle.right(90)
def drawDigit(d): #根据数字绘制七段数码管
    drawLine(True) if d in [2,3,4,5,6,8,9] else drawLine(False)
    drawLine(True) if d in [0,1,3,4,5,6,7,8,9] else drawLine(False)
    drawLine(True) if d in [0,2,3,5,6,8,9] else drawLine(False)
    drawLine(True) if d in [0,2,6,8] else drawLine(False)
    turtle.left(90)
    drawLine(True) if d in [0,4,5,6,8,9] else drawLine(False)
    drawLine(True) if d in [0,2,3,5,6,7,8,9] else drawLine(False)
    drawLine(True) if d in [0,1,2,3,4,7,8,9] else drawLine(False)
    turtle.left(180)
    turtle.penup()
    turtle.fd(20)
def drawDate(date):
    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))
def main():
    turtle.setup(800, 350, 200, 200)
    turtle.penup()
    turtle.fd(-350)
    turtle.pensize(5)
#     drawDate('2018-10=10+')
    drawDate(time.strftime('%Y-%m=%d+',time.gmtime()))
    turtle.hideturtle()
    turtle.done()
main()
```

结果：

图 2

3.代码复用与函数递归

1.代码复用与模块化设计

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。

2.函数递归的理解

函数中调用函数自身的方式被称为递归。递归中两个重要的特征是**链条**和**基例**。

链条是指递归链条。

基例存在一个或多个不需要再次递归的实例。

3.函数递归的调用过程

假设我们求一个 5 的阶乘，其调用过程如图 3 所示。

图 3

4.函数递归实例解析

例 1：字符串反转：将字符串反转后输出。

代码：原来可以使用 s[::-1]实现反转。

使用递归代码如下：

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

s = "I LOVE YOU"
rm = rvs(s)
print(rm)
```

输出结果：

```
UOY EVOL I
```

例 2：斐波那契数列

代码：

```
def fib(n):
    if n==1 or n==2:
        return 1
    else:
        return fib(n-1) +fib(n-2)

h=fib(7)
print("{}".format(h))
```

输出结果：

```
13
```

（06:23）

4.模块 4：PyInstaller 库的使用

在所需要打包的程序的目录下，运行 cmd 命令行，输入命令：
pyinstaller -F <文件名.py>
记住，一定是使用命令行!!!!!

表 N 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文件

5.科赫雪花小包裹

代码：

```
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()
```

结果：

图 4

打包过程：命令行调整到当前文件夹，然后输入：pyinstaller -F homework.py

打包完成，命令行效果如图 5 所示。

图 5