

学习目标:

1. 掌握函数的定义和调用方法
2. 理解函数的参数传递过程以及变量的作用范围
3. 了解lambda函数
4. 掌握时间日期标准库的使用
5. 理解函数递归的定义和使用方法



第五章 函数和代码复用

目

录

CONTENTS

5.1 函数的基本使用

5.2 函数的参数传递

5.3 模块3: datetime库的使用

5.4 实例7: 七段数码管绘制

5.5 代码复用和模块化设计

5.6 函数的递归

5.7 实例8: 科赫曲线绘制

5.8 Python内置函数

函数的定义

- ◆ 函数概念：是一段具有特定功能的，可重用的语句组；
- ◆ 函数分类：自定义函数和安装包自带的内置函数（如abs、eval、input、print等）；
- ◆ 使用函数的目的：降低编程难度、代码重用；
- ◆ Python中自定义函数的定义格式：
def <函数名>(<形式参数列表>):
 <函数体>
 return <返回值列表>
- ◆ 函数的调用和执行的一般格式：
 <函数名>(<实际参数列表>)
 在参数调用时，实际参数列表的参数个数、排列顺序、类型一般要一一对应。

函数的调用过程

函数的调用需要执行4个步骤：

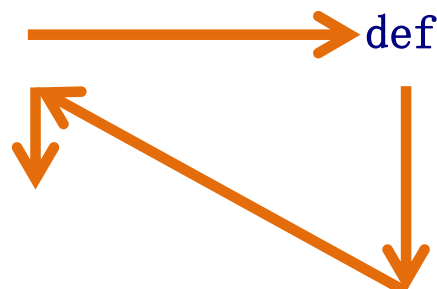
- ◆ 调用程序在调用处暂停执行；
- ◆ 在调用时将实际参数复制给函数的形式参数；
- ◆ 执行函数的语句；
- ◆ 函数调用结束给出返回值，程序回到调用前的暂停处继续执行

```
def happy():  
    print("Happy Birthday to you!")  
def happyB(name):  
    happy()  
    happy()  
    print("Happy Birthday, dear {}".format(name))  
    happy()  
happyB("Mike")  
print()  
happyB("Lily")
```

函数的调用过程

Mike
happyB("Mike")
print()
happyB("Lily")

def happyB(name):
 happy()
 happy()
 print("Happy Birthday, dear {}".format(name))
 happy()



主调程序

被调函数

lambda函数

Lambda函数，又称为匿名函数，其使用方法如下：

<函数名>=lambda <参数列表>:<表达式>

等价于普通函数

def <函数名>(<参数列表>):

return <表达式>

简单的说，lambda函数用于定义简单的，能够在一行内表示的函数，返回一个简单类型，如：

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

```
>>>type(f)
```

```
<class 'function' >
```

```
>>>f(10,12)
```

```
22
```

可选参数

定义函数时，可以为某个参数指定默认值，称为可选参数，可选参数要在非可选参数之后定义，如：

```
>>> def dup(str,times=3):  
        print(str*times)  
>>> dup( "knock!" )  
knock! knock! knock!  
>>> dup( "knock!" ,4)  
knock! knock! knock! knock!
```

可变数量参数

定义函数时，可以为设计可变数量参数，通过在参数前加星号（*）实现，但只能出现在参数列表的最后，调用时，这些参数被当作元组类型传递到函数中，如：

```
>>> def vfunc(a,*b):  
        print(type(b))  
        for n in b:  
            a+=n  
        return a  
>>> vfunc(1,2,3,4,5)    #2 , 3 , 4 , 5做为元组传递给*b  
<class 'tuple' >  
15
```


参数的位置和名称传递

实际参数传递给形式参数时，默认是按照顺序的方式传递给函数的，但是当参数很多时，这种调用方式可读性不好，为解决这个问题，Python提供了按照参数名称输入实参的方式，如：

假设有函数：

func(x1,y1,z1,x2,y2,z2)

可以使用下列调用方式：

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

等价于：

func(1,2,3,4,5,6)

函数的返回值

return语句退出函数并返回函数被调用的位置继续执行，Python中return语句会将0到多个结果返回给被调用处，如：

```
>>> def func(a,b)
```

```
    return a*b
```

```
>>> s=func( "knock!" ,2)
```

```
>>> print(s)
```

```
knock! knock!
```

```
>>> def func(a,b):
```

```
    return b,a
```

```
>>> s=func( "knock" ,2)  #多个值以元组类型保存
```

```
>>> print(s,type(s))
```

```
(2," knock!" ) <class 'tuple' >
```

函数对变量的作用

程序中的变量分为全局变量和局部变量，全局变量是在函数之外定义的变量，在程序的全局有效，局部变量是在函数内部使用的变量，仅在函数内部有效如：

```
n=1
```

```
def func(a,b):
```

```
    c=a*b
```

```
    return c
```

```
s=func("knock!",2)
```

```
print(c)
```

中n为全局变量。而c为局部变量，只能在func函数中有效，所以在print(c)处会出现错误。

```
C:\compare_homework\venv\Scripts\python.exe C:/compare_homework/1.py
Traceback (most recent call last):
  File "C:/compare_homework/1.py", line 6, in <module>
    print(c)
NameError: name 'c' is not defined

Process finished with exit code 1
```

函数对变量的作用

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

上述程序段中，`n=b`语句中的`n`是在函数中重新定义的变量，仍然是局部变量，只是在函数中有效，故`print(s,n)`中，`n`的输出值为1.

如果希望在函数中将`n`视作全局变量，需要在变量`n`前显式声明变量为全局变量：`global n`

函数对变量的作用

```
ls=[]  
def func(a, b):  
    ls.append(b)  
    return a*b
```

```
s=func("knock!", 2)
```



```
C:\compare_homework\venv\Scripts\python.exe C:/compare_homework/1.py
```

```
knock!knock! [2]
```

```
Process finished with exit code 0
```

列表等组合类型的创建和引用有区别，当执行`ls.append(b)`时，系统寻找已经创建的列表变量，函数中没有已经创建的名称为`ls`的列表，然后寻找全局的`ls`列表，然后进行添加。如果函数中有名为`ls`的列表变量，则全局变量`ls`值为空。

实验5.1 将十进制数转换为十六进制，包括如下函数：

- `decimal2hex(decimalvalue)`将十进制`decimalvalue`转换为十六进制
- `hex2char(haxvalue)`将十六进制的数值转换为十六制编码（0~9，'A'~'F'）
- `main()`函数输入十进制数、调用`decimal2hex`函数、输出转换后的十六进制数对应的字符串。

datetime库概述

- ◆ datetime库提供了一系列时间处理方法，并以用户选择的格式输出。
- ◆ 该库包含两个常量：datetime.MINYEAR和datetime.MAXYEAR, 对应能够表示的最小、最大年份。分别为1 和9999；
- ◆ **datetime库提供多种日期和时间表达方式：**

类	描述
datetime.date	日期类，可以表示年、月、日
datetime.time	时间类，可以表示小时、分钟、秒、毫秒
datetime.datetime	日期和时间类，功能覆盖date和time类
datetime.timedelta	与时间间隔有关的类
datetime.tzinfo	与时区有关的信息类

datetime库解析

datetime对象的创建方法:

◆ 使用datetime.now()获得当前日期和时间

```
>>>from datetime import datetime
```

```
>>>today=datetime.now()
```

```
>>>today
```

```
2018-01-31 17:07:15.753678
```

◆ 使用datetime.utcnow()//世界标准时间

```
>>>today=datetime.utcnow()
```

```
>>>today
```

```
2018-01-31 17:09:16.423729
```

◆ 使用datetime.datetime()构造函数

```
datetime.datetime(year,month,day,hour=0,minute=0,second=0,microsecond=0)
```


datetime库解析

假设someday为datetime类的对象，其常用属性有

属性	描述
someday.min	返回最小时间对象，datetime(1, 1, 1, 0, 0, 0, 0)
someday.max	返回最大时间对象，datetime(9999, 12, 31, 23, 59, 59, 999999)
someday.year	返回年份
someday.month	返回月份
someday.day	返回日期
someday.hour	返回小时
someday.minute	返回分钟
someday.second	返回秒数
someday.microsecond	返回微秒数

datetime库解析

常用的时间格式化方法

属性	描述
somday.isoformat()	采用ISO8601标准显示时间
someday.isoweekday()	根据日期计算星期后返回1-7，对应星期一到星期日
someday.strftime()	根据格式化字符串format进行格式化显示

```
from datetime import datetime
```

```
someday=datetime.now()
```

```
print(someday.isoformat())
```

```
print(someday.isoweekday())
```

```
2018-01-31T17:27:58.531752
```

```
3
```

datetime库解析

strftime()是最有效的输出日期时间的方法，其使用方法如下：

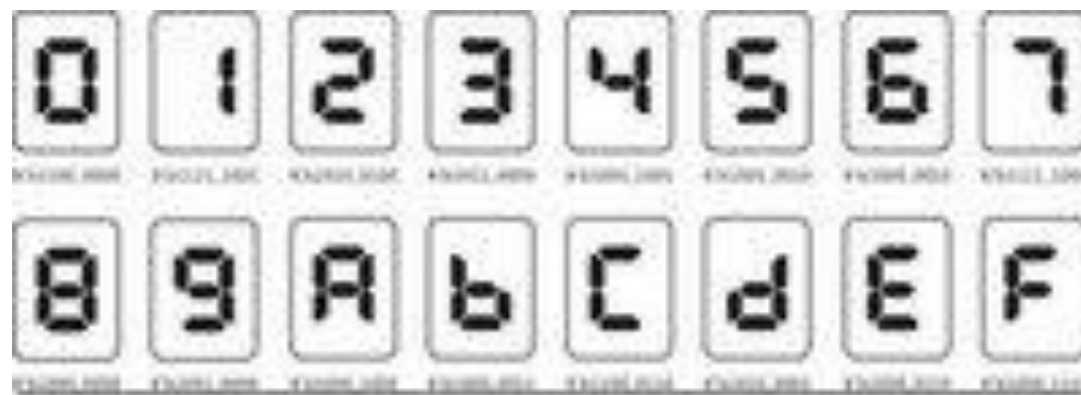
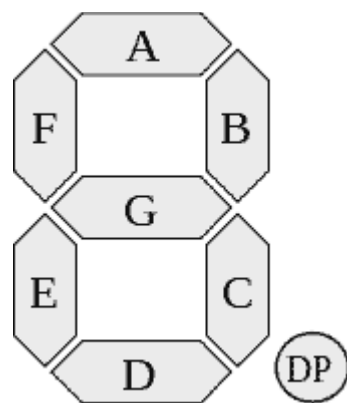
```
>>>someday.strftime("%Y-%n-%d %H:%M:%S")
```

2018-01-31 17:33:51

格式化控制符：

格式控制符	日期/时间	值范围和实例	格式控制符	日期/时间	值范围和实例
%Y	年份	00001-9999	%a	星期缩写	Mon-Sun
%m	月份	01-12,	%H	小时(24h)	00-23
%B	月名	January-December	%I	小时(12h)	01-12
%b	月名缩写	Jan-Dec	%p	上/下午	AM,PM
%d	日期	01-31	%M	分钟	00-59
%A	星期	Monday-Sunday	%S	秒	00-59

七段数码管能够显示易于理解的数字和字母，本文通过turtle库函数绘制七段数码管的日期信息。



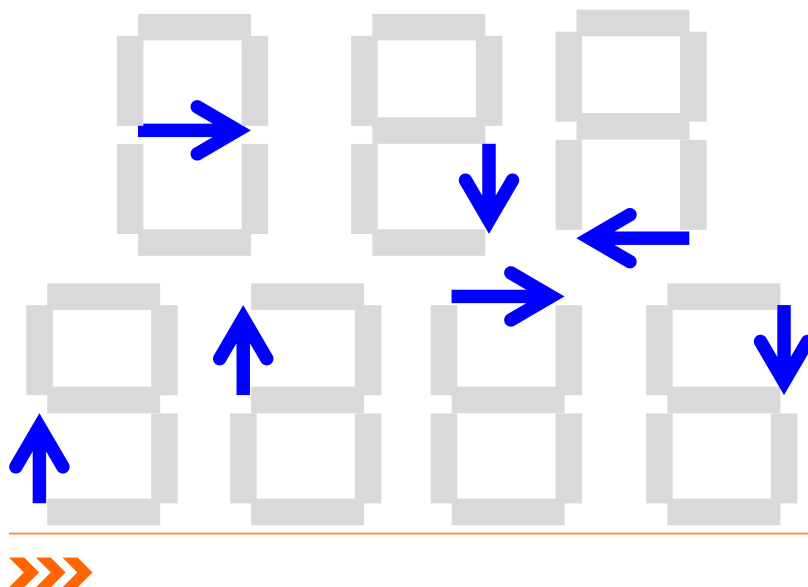
#e7.1drawsevensegdisplay.py

```
import turtle, datetime
def drawline(draw):
    turtle.pendown() if draw else turtle.penup()
    turtle.fd(40)
    turtle.right(90)
def drawdigit(d):
    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)
```

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

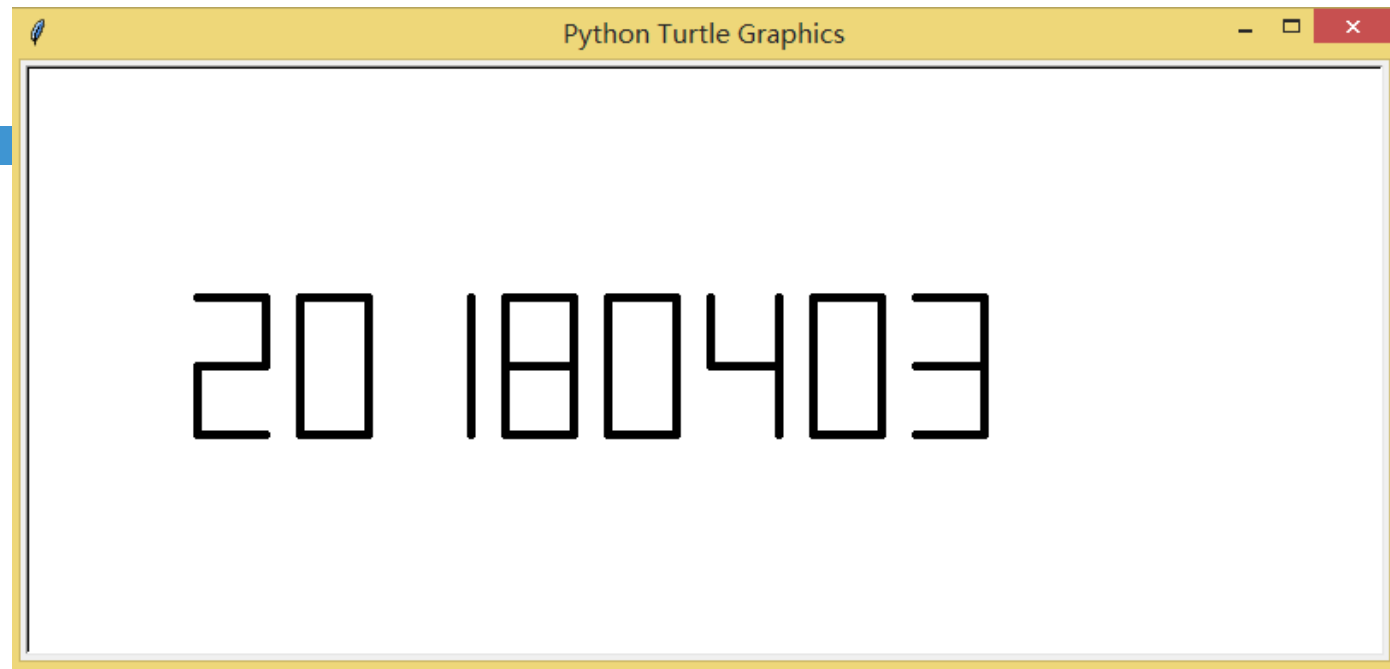
如果draw为true, 执行turtle.pendown, 否则执行turtle.penup()

drawdigit()依次绘制如下数码段:



5.4 实例7：七段数码管绘制

```
def drawdate(date):  
    for i in date:  
        drawdigit(eval(i))  
  
def main():  
    turtle.setup(800, 350, 200, 200)  
    turtle.penup()  
    turtle.fd(-300)  
    turtle.pensize(5)  
    drawdate(datetime.datetime.now().strftime('%Y%m%d'))  
    turtle.hideturtle()  
main()
```



```

优化代码:
#e7.1drawsevensegdisplay.py
import turtle,datetime
def drawgap():
    turtle.penup()
    turtle.fd(5)
def drawline(draw):
    drawgap()
    turtle.pendown() if draw else turtle.penup()
    turtle.fd(40)
    drawgap()
    turtle.right(90)
def drawdigit(d):
    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)

```

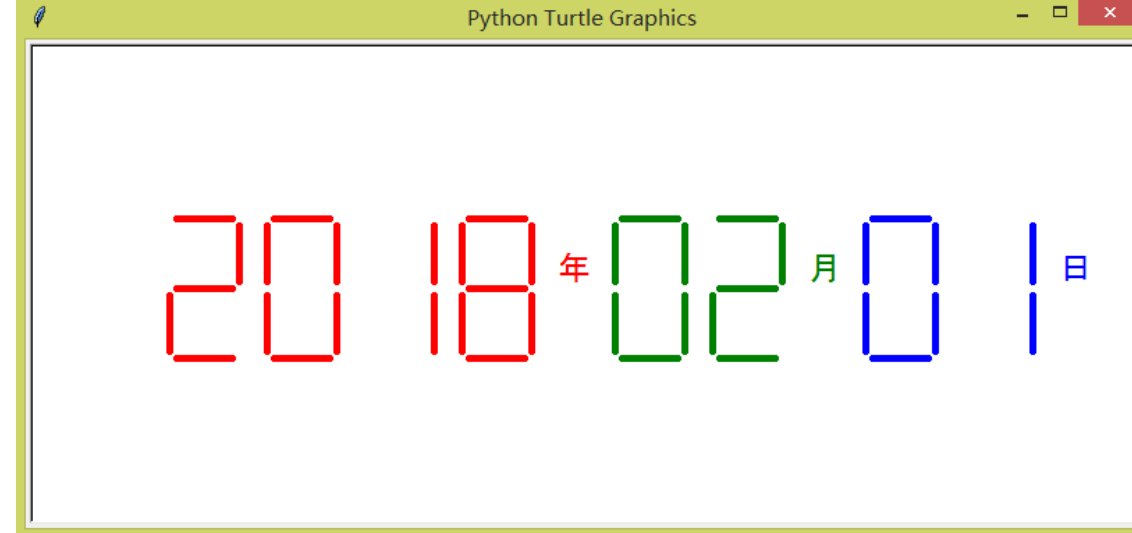
```

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(-300)
    turtle.pensize(5)
    drawdate(datetime.datetime.now().strftime('%Y-%m=%d+'))
    turtle.hideturtle()

main()

```



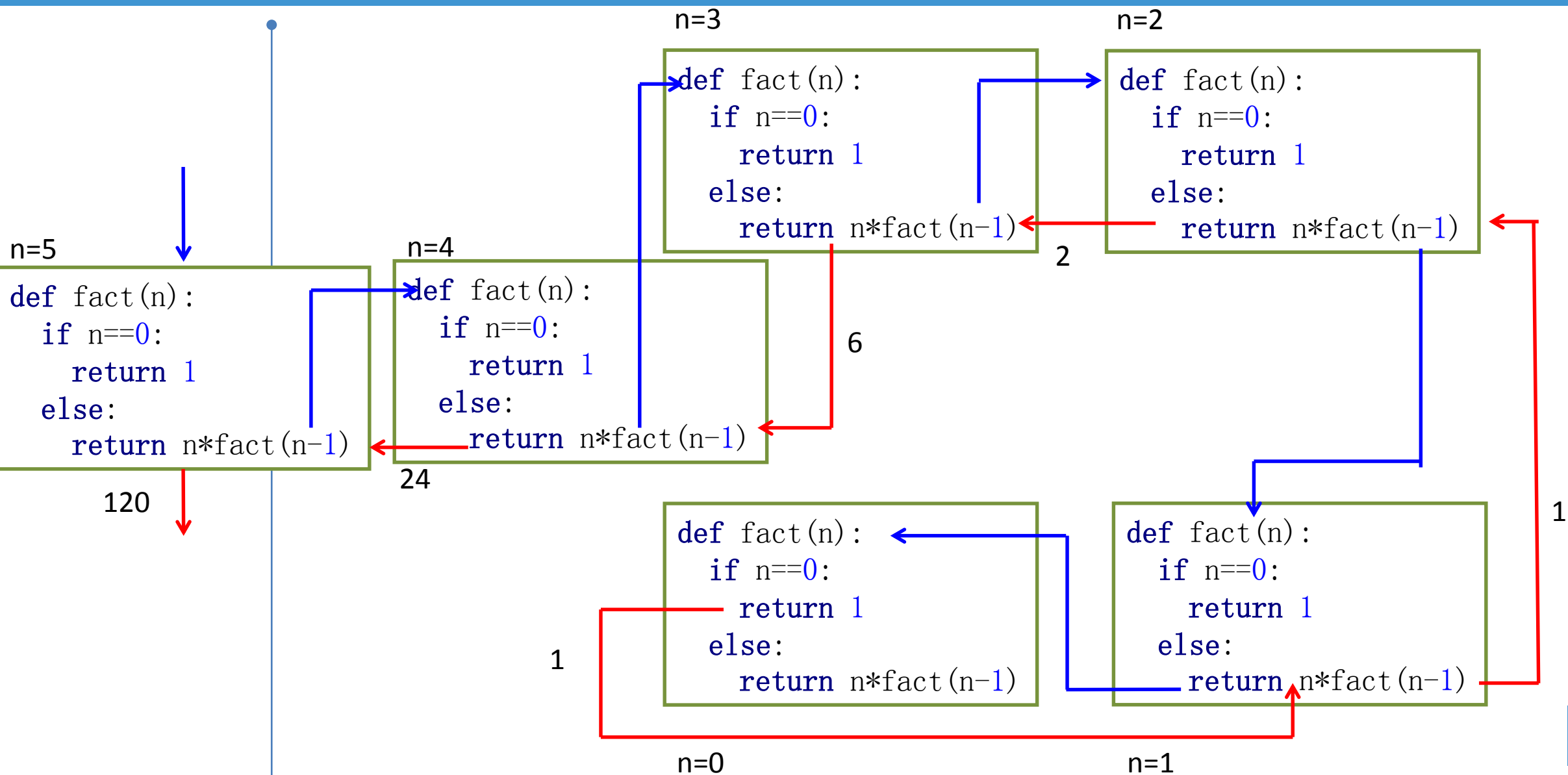
在计算机编程里，**递归**指的是一个过程：函数不断引用自身，直到引用的对象已知。

$$n! = n * (n-1) * (n-2) * \dots * 1$$

递归的两个特征：

- (1) 问题规模不断变小；
- (2) 在某个时候能安全退出

```
def fact(n):  
    if n==0:  
        return 1  
    else:  
        return n*fact(n-1)  
num=eval(input("请输入一个整数: "))  
print(fact(abs(int(num))))
```



0阶曲线表示长度为 L 的线段，将 L 三等分，中间一段用 $L/3$ 的等边三角形代替，得到1阶曲线，它包含4条线段，继续对每条线段进行上述操作，得到2阶科赫曲线，.....

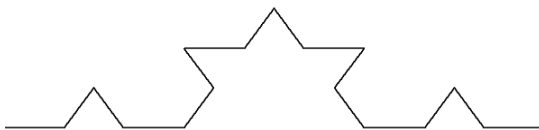
0阶科赫曲线



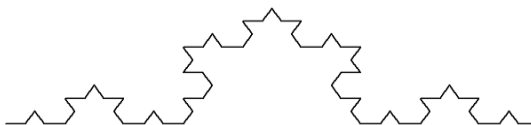
1阶科赫曲线



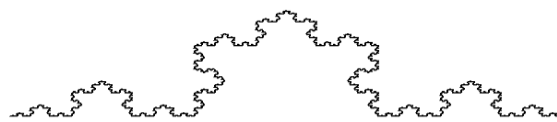
2阶科赫曲线



3阶科赫曲线



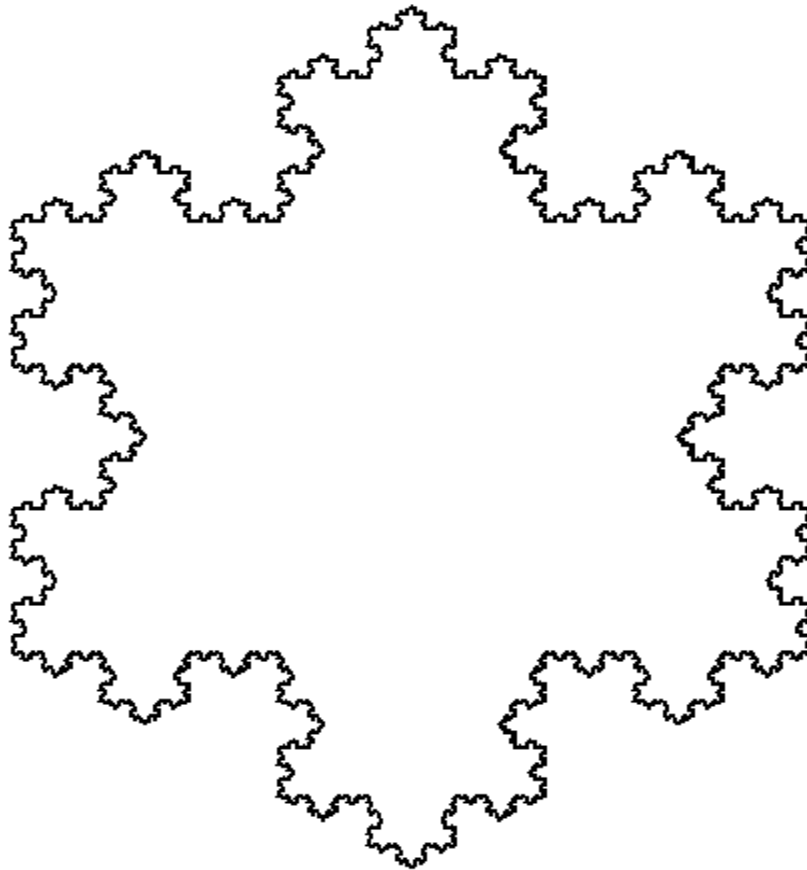
5阶科赫曲线



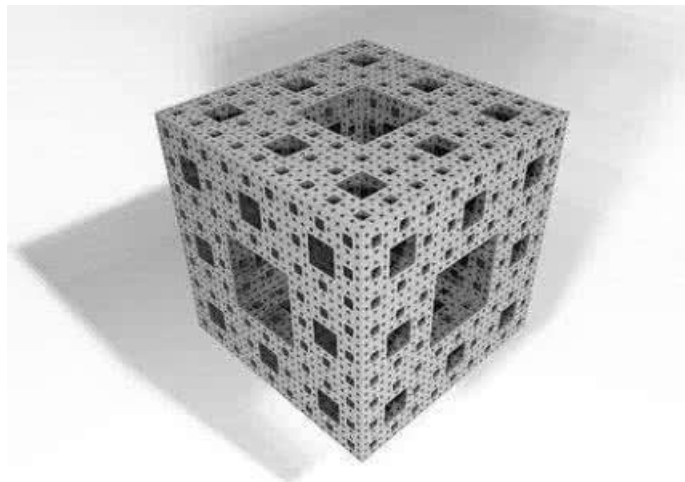
```
#e8.1drawkock.py
import turtle
def kock(size,n):
    if n==0:
        turtle.fd(size)
    else:
        for angle in [0,60,-120,60]:
            turtle.left(angle)
            kock(size/3,n-1)
def main():
    turtle.setup(800,400)
    turtle.speed(0)
    turtle.penup()
    turtle.goto(-300,-50)
    turtle.pendown()
    turtle.pensize(2)
    kock(600,5)
    turtle.hideturtle()
main()
```

如果将main()函数修改为如下代码，则运行结果如右图所示：

```
def main():  
    turtle.setup(600,600)  
    turtle.speed(0)  
    turtle.penup()  
    turtle.goto(-200,100)  
    turtle.pendown()  
    turtle.pensize(2)  
    level=5  
    kock(400,level)  
    turtle.right(120)  
    kock(400,level)  
    turtle.right(120)  
    kock(400,level)  
    turtle.hideturtle()  
main()
```



分形式几何学的分支，以自相似结构为基础，通过无限次递归方式展现复杂表面的内在数学秩序。



Python提供了68个内置函数，不需要引入任何库就可以直接引用，

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

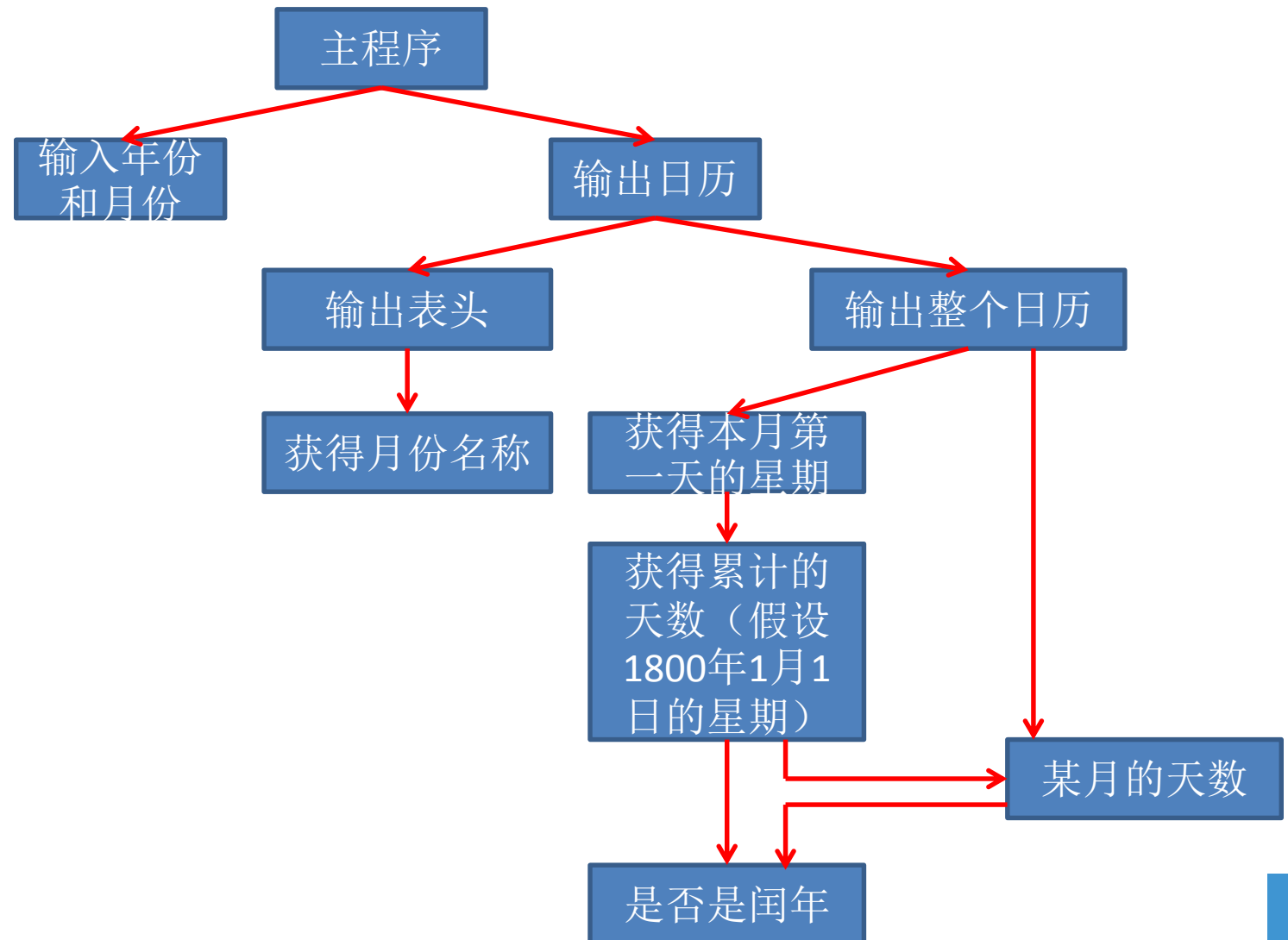
Python提供很多关于使用手册，一些数据类型、函数和标准库的使用可以访问 <https://docs.python.org/3> 中的tutorial、libraryreference、language reference等文档。

实验5.2 一个大作业（3个人一组来完成，自由组合）
编写一个程序，显示给定年月的日历，程序提示输入年月，然后显示该月的
整个日历，显示结果如下：

```
enter full year (e. g., 2018): 2018
enter month as number between 1 and 12: 4
      April      2018
*****
Sun Mon Tue Wed Thu Fri Sat
 1   2   3   4   5   6   7
 8   9  10  11  12  13  14
15  16  17  18  19  20  21
22  23  24  25  26  27  28
29  30
*****
```

```
enter full year (e. g., 2018): 1972
enter month as number between 1 and 12: 3
      March      1972
*****
Sun Mon Tue Wed Thu Fri Sat
                1   2   3   4
 5   6   7   8   9  10  11
12  13  14  15  16  17  18
19  20  21  22  23  24  25
26  27  28  29  30  31
*****
```


程序结构如下：



实验5.1 将十进制数转换为十六进制，包括如下函数：

- decimal2hex(decimalvalue)将十进制decimalvalue转换为十六进制
- hex2char(hexvalue)将十六进制的数值转换为十六制编码（0~9，'A'~'F'）
- main()函数输入十进制数、调用decimal2hex函数、输出转换后的十六进制数对应的字符串。

def hex2char(hexvalue):

```
if 0<=hexvalue<=9:
    return str(hexvalue)
elif hexvalue==10:
    return 'A'
elif hexvalue==11:
    return 'B'
elif hexvalue==12:
    return 'C'
elif hexvalue==13:
    return 'D'
elif hexvalue==14:
    return 'E'
elif hexvalue==15:
    return 'F'
```

def decimal2hex(decimalvalue):

```
str=""
while decimalvalue!=0:
    str=hex2char(decimalvalue%16)+str
    decimalvalue=decimalvalue//16
return str
```

def main():

```
num=eval(input("输入十进制数: "))
resultstr=decimal2hex(num)
print("result=",resultstr)
```

main()