

Python程序语言入门与应用



python








Life is short, use Python
人生苦短，我用Python



课程回顾



第4章 Python程序控制结构

-  4.1 程序的基本结构
-  4.2 程序的分支结构
-  4.3 实例：身体质量指数BMI
-  4.4 程序的循环结构
-  4.5 random库的使用
-  4.6 π 值的计算
-  4.7 程序异常处理



上周练习题



4.1 猜数游戏

在程序中预设一个1-100的随机整数，让用户通过键盘输入所猜数，若大于预设数，显示“遗憾，太大了”；若小于预设数，显示“遗憾，太小了”，如此循环，直至猜中该数，显示“预测N次，恭喜你猜中了！”。要求当用户输入非整数时，给出“输入内容必须为整数！”的提示，并让用户重新输入。

```
11 from random import randint
12 num = randint(1, 100)
13 print("#"*40)
14 print("{0:^34}".format("猜数游戏"))
15 print("#"*40)
16 guess = input("请输入一个1~100的整数[输入Q结束]: ")
17 t_num = 0
18 while guess[0] not in ["Q", "q"]:
19     t_num += 1
20     try:
21         if int(guess) == num:
22             print("预测{}次，恭喜你猜中了!".format(t_num))
23             break
24         elif int(guess) > num:
25             print("遗憾，太大了!")
26         else:
27             print("遗憾，太小了!")
28     except ValueError:
29         print("输入错误，请输入一个整数!")
30     guess = input("请再次输入一个1~100的整数[输入Q结束]: ")
```

```
#####
猜数游戏
#####
请输入一个1~100的整数[输入Q结束]:
```



上周练习题



4.1 猜数游戏

在程序中预设一个1-100的随机整数，让用户通过键盘输入所猜数，若大于预设数，显示“遗憾，太大了”；若小于预设数，显示“遗憾，太小了”，如此循环，直至猜中该数，显示“预测N次，恭喜你猜中了！”。要求当用户输入非整数时，给出“输入内容必须为整数！”的提示，并让用户重新输入。

```
from random import randint
num = randint(1, 100)
print("#"*40)
print("{0:^34}".format("猜数游戏"))
print("#"*40)
guess = input("请输入一个1~100的整数[输入Q结束]: ")
t_num = 0
while guess[0] not in ["Q", "q"]:
    try:
        if int(guess) == num:
            t_num += 1
            print("预测{}次，恭喜你猜中了!".format(t_num))
            break
        elif int(guess) > num:
            t_num += 1
            print("遗憾，太大了!")
        else:
            t_num += 1
            print("遗憾，太小了!")
    except:
        print("输入错误，请输入一个整数!")
    guess = input("请再次输入一个1~100的整数[输入Q结束]: ")
```



上周练习题



4.2 统计不同字符个数

用户从键盘输入一行字符，编写一个程序，统计并输出英文字符、数字、空格和其他字符的个数。

```
chr_en_num = 0
chr_digit_num = 0
chr_space_num = 0
chr_other_num = 0
✓ for c in chr_seq:
✓     if "A" <= c <= "Z" or "a" <= c <= "z":
        chr_en_num += 1
✓     elif "0" <= c <= "9":
        chr_digit_num += 1
✓     elif c == " " or c == "\t":
        chr_space_num += 1
✓     else:
        chr_other_num += 1
print("\n方法2: ")
✓ print("英文字符数为{}\n数字数为{}\n空格数为{}\n其他字符个数为{}".\
        format(chr_en_num, chr_digit_num, chr_space_num, chr_other_num))
```



程序练习题



4.3 最大公约数和最小公倍数计算

从键盘接受2个整数，编写程序求出这两个整数的最大公约数和最小公倍数（提示：最小公倍数等于两数之积除以其最大公约数）。

```
E: > 课程 > python > 第5次课 > 上次作业 > 第三题.py > ...
1  a = int(input('请输入第一个数字: '))
2  b = int(input('请输入第二个数字: '))
3  smin = min(a,b)
4  divisor, multiple= 0,0
5  for i in range(1,smin+1):
6      if a % i == 0 and b % i == 0:
7          divisor = i
8  multiple = a*b/divisor
9  print('最小公约数是: {},最小公倍数是: {}'.format(divisor,multiple))
10
```



新乡医学院

Python程序语言入门与应用

第一章 Python函数与模块

王海蛟

新乡医学院



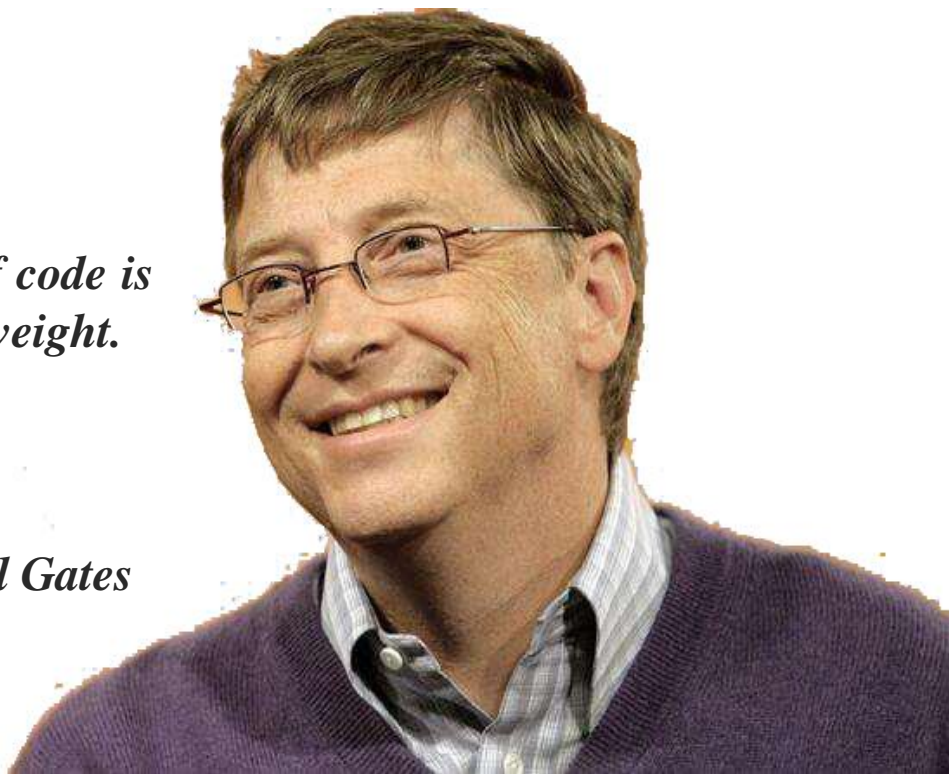


学习目标



Measuring programming progress by lines of code is like measuring aircraft building progress by weight.

Bill Gates



以代码数量来衡量程序设计的进度,就好比以重量来衡量飞机的制造速度.



学习目标



基本要求

☞ 掌握

☞ 函数的定义和调用方法

☞ 理解

☞ 时间日期标准库的使用

☞ 了解

☞ 函数递归的定义和使用方法



本课概要



第5章 函数和模块

- 5.1 函数的基本使用
- 5.2 函数的参数传递
- 5.3 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
- 5.8Python内置函数



5.1 函数的基本使用



5.1.1 函数的定义

Python的程序由包、模块和函数组成。

函数是一段可重用的有名称的代码。函数也可以看成是具有一定功能的程序。

函数是一种功能的抽象。

函数运行后会返回相应的处理结果。

模块是处理某一类问题的集合，模块由函数和类组成。

模块和常规Python程序之间的唯一区别是用途不同：模块用于编写其他程序。因此，模块通常没有main函数。

比如我们在写Python时，第一行代码通常是import **，这里的**就是模块，也说是库。



5.1 函数的基本使用



5.1.1 函数的定义

包是一个完成特定任务的工具箱。

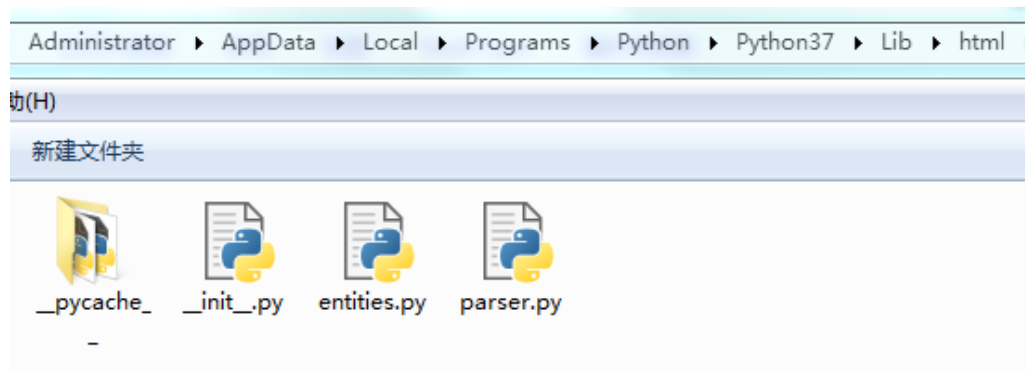
- Python提供了许多有用的工具包，如字符串处理、图形用户接口、Web应用、图像处理等。使用自带的工具包，可以提高程序开发效率、减少编程复杂度，达到代码重用的效果。
- Python中，包其实就是一个文件夹或者目标，里面放着很多功能归属统一的库。
- Python中，包目录中会放置一个`__init__.py`的文件，用于表示这个目录是一个包。
- Python自带的工具包和模块安装在其安装目录的Lib子目录中。



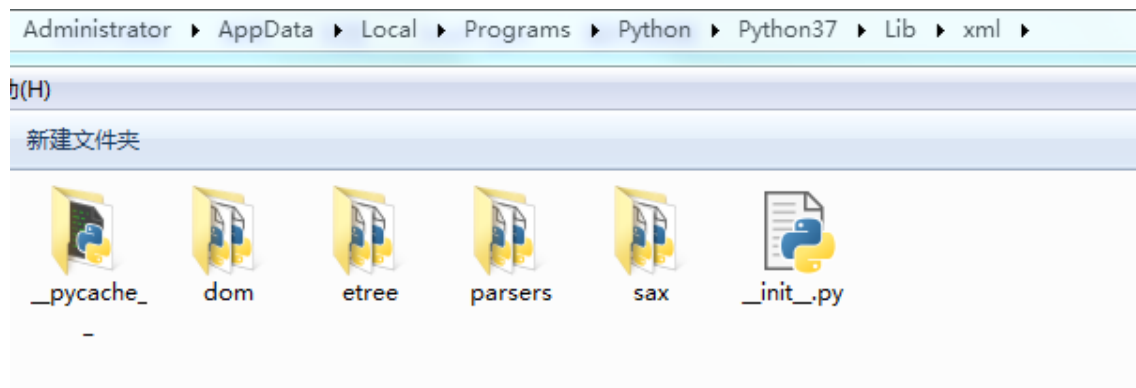
5.1 函数的基本使用



5.1.1 函数的定义



这个html就是python程序的一个包，里面有一个__ini __.py的文件



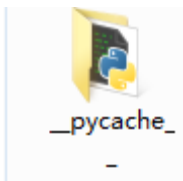
这个xml也是python程序的一个包，里面有一个__ini __.py的文件，同时，它里面还有其它包。



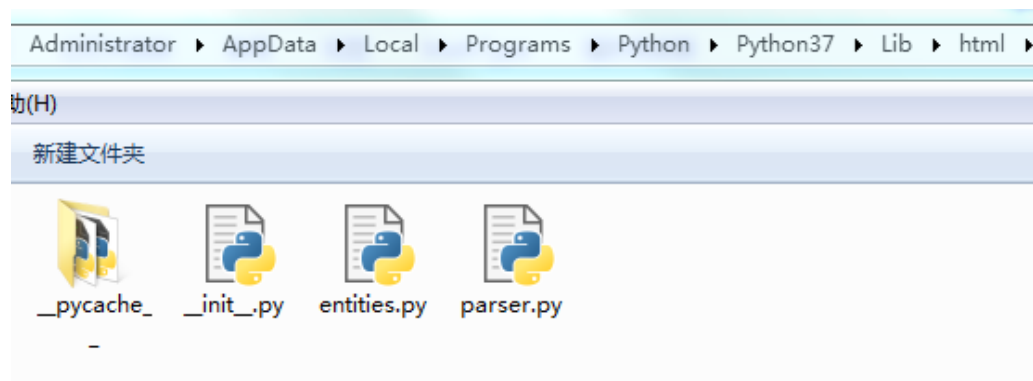
5.1 函数的基本使用



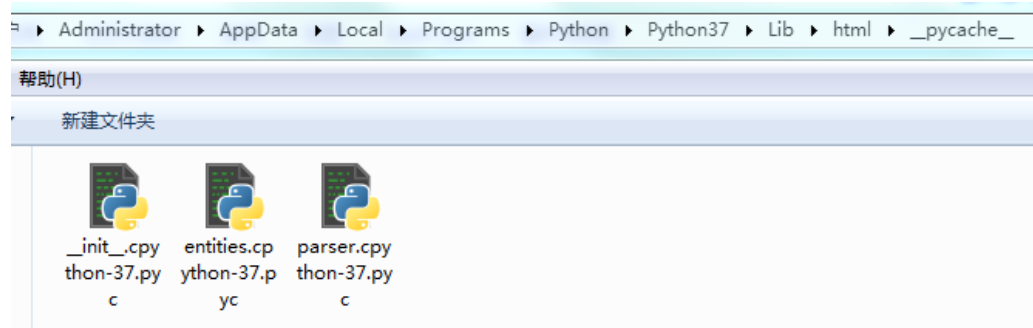
5.1.1 函数的定义



每个包文件里面还都有一个 **`__pycache__`** 文件，这个文件里面存放的是包里库文件编译后的字节码文件。



Python在导入模块的时候，首先会查找字节码文件。我们在发布程序时，可以发布编译后的程序，这样可以保护源代码。

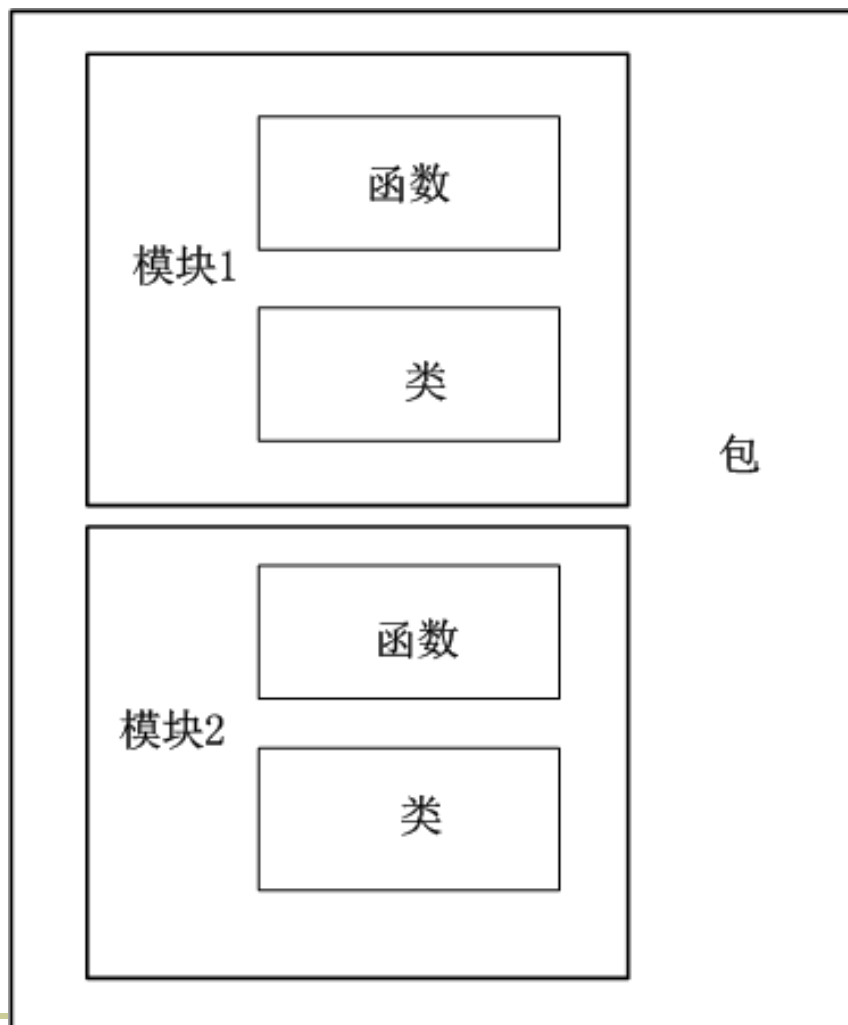




5.1 函数的基本使用



5.1.1 函数的定义





5.1 函数的基本使用



5.1.1 函数的定义

Python定义一个函数，使用**def**关键字，其语法形式如下：

```
def <函数名>(<参数列表>):
```

```
    <函数体>
```

```
    return <返回列表>
```

函数的调用(实参列表)

函数名

函数必须先定义，后使用；

函数名与变量名的命名规则相同，只能包含字母、数字和下划线_，且不能以数字打头。



5.1 函数的基本使用



5.1.1 函数的定义

微实例5.1 生日歌。

过生日时要为朋友唱生日歌，歌词如下：

Happy birthday to you!

Happy birthday to you!

Happy birthday, dear <名字>

Happy birthday to you!

编写Python程序为Mike和Lily输出生日歌。最简单的就是重复使用print()函数。



5.1 函数的基本使用



5.1.1 函数的定义

```
E: > 课程 > python > 第5次课 > 实例 > 生日歌5.1.py
1
2
3
```

OUTPUT DEBUG CONSOLE TERMINAL 1: cmd

```
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

C:\Users\Administrator>
```



5.1 函数的基本使用



5.1.1 函数的定义

代码如下：

```
1  print('Happy birthday to you !')
2  print('Happy birthday to you !')
3  print('Happy birthday, dear Mike')
4  print('Happy birthday to you !')
```

代码中1, 2, 4行都一样，如果要给其它人过生日，还得重新再写一遍代码。



5.1 函数的基本使用



5.1.1 函数的定义

```
1  def happy ():
2      print('Happy birthday to you !')
3  def happyB(name):
4      happy()
5      happy()
6      print('Happy birthday, dear {} !'.format(name))
7      happy()
8  happyB('Mike')
9  print()
10 happyB('Lily')
11
```

函数输出结果如下：

```
Happy birthday to you !
Happy birthday to you !
Happy birthday, dear Mike !
Happy birthday to you !
```

```
Happy birthday to you !
Happy birthday to you !
Happy birthday, dear Lily !
Happy birthday to you !
```



5.1 函数的基本使用



5.1.2 函数调用的过程

程序调用一个函数需要执行以下四个步骤：

- (1) 调用程序在调用处暂停执行；
- (2) 在调用时将实参复制给函数的形参；
- (3) 执行函数体语句；
- (4) 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。



5.1 函数的基本使用



5.1.2 函数调用的过程

函数HappyB的调用过程：

```
name="Mike"
```

```
happyB("Mike")  —→  def happyB(name):  
                        print()  
                        happy()  
happyB("Lily")      happy()  
                        print("Happy birthday, dear!".format(name))  
                        happy()
```




5.1函数的基本使用



5.1.2 函数调用的过程

整个程序的执行调用过程：

```
name="Mike"

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

```
name="Mike"

happyB("Mike") → def happyB(name):
print()           happy()
happyB("Lily")    happy()
                 print("Happy birthday, dear!".format(name))
                 happy()
```



5.1 函数的基本使用



5.1.3 lambda 函数

- Lambda函数用于创建一个匿名函数，函数名未和标识符进行绑定。
- Lambda是python中33个保留字里面的一个。
- 匿名函数并非没有名字，而是将函数作为函数结果返回。
- Lambda用于定义简单，能够在一行内表示的函数。

<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>
```



5.1 函数的基本使用



5.1.3 lambda 函数

```
1  f = lambda x,y : x+y
2  print(type(f))
3  print(f(10,12))
4
```

运行结果:

```
<class 'function'>
22
```



5.1函数的基本使用



 Lambda可以直接作为函数使用

(lambda <参数列表>:<表达式>)(实参)

(lambda x,y : x+y)(10,12)

```
--  
>>> (lambda x,y : x+y)(10,12)  
22  
>>>
```

(lambda x : -x)(-5)

```
--  
>>> (lambda x : -x)(-5)  
5  
>>>
```



5.2 函数参数的传递



5.2 函数参数的传递

5.2.1 可选参数和可变参数

- ✓ 可选参数：调用时可以选择输入也可以不输入的参数。
- ✓ 可变参数：参数的数量可以变化。

在定义函数时，有些参数可以设置默认值：

```
>>>def dup(str, times = 2):  
    print(str*times)  
>>>dup("knock~")  
knock~knock~  
>>>dup("knock~", 4)  
knock~knock~knock~knock~
```



5.2函数参数的传递



5.1.1 函数的定义

 在定义函数时，可以设置可变数量的参数：

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


通过参数前增加星号实现



5.2函数参数的传递



5.2.2 参数的位置和名称

 实参按照顺序从左→右依次传递给形参

例如 `func(age, background, job, salary)`

`func(45, 'PhD' , 'engineer' , 10000)`

`func(age = 45, background = 'PhD' , job = 'engineer' , salary = 10000)`

调用时可以直接把参数名字写上，增加程序的可读性！



5.2函数参数的传递



5.2.3变量的返回值

 return用来返回函数值

```
>>>def func(a, b):  
    return a*b  
>>>s = func("knock~", 2)  
>>>print(s)  
knock~knock~
```

 return可以返回多个值-返回值以元组形式保存

```
>>>def func(a, b):  
    return b,a  
>>>s = func("knock~", 2)  
>>>print(s, type(s))  
(2, 'knock~') <class 'tuple'>
```



5.2函数参数的传递



5.2.3变量的返回值

无return函数

```
def happy():  
    print("Happy birthday to you.")
```

等价于:

```
def happy():  
    print("Happy birthday to you.")  
    return None
```



5.2 函数参数的传递



5.2.4 函数对变量的作用

局部变量

- 全局变量是能够被不同的函数、类或文件共享的变量。
- 在函数之外定义的变量都可以称为全局变量

```
>>> n = 5
>>> def col(a,b):
>>>     c=a*b
>>>     return c

>>> print(col(3,2))
6
>>> print(c)
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    print(c)
NameError: name 'c' is not defined
>>> print(n)
5
>>>
```

尝试输出c时，程序报错：



5.2函数参数的传递



5.2.4 函数对变量的作用

全局变量

- 全局变量是能够被不同的函数、类或文件共享的变量。
- 在函数之外定义的变量都可以称为全局变量。

```
>>> n = 5
>>> def col(a,b):
>>>     c=a*b
>>>     return c

>>> print(col(3,2))
6
>>> print(c)
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    print(c)
NameError: name 'c' is not defined
>>> print(n)
5
>>>
```

尝试输出n时，程序正常运行：



5.2函数参数的传递



5.2.4 函数对变量的作用

全局变量

➤全局变量的值在函数内部使用且改变时，会怎么样呢？

```
>>> n = 5
>>> def col(a,b):
    c = a*b
    n = b
    return c+n

>>> col(3,2)
8
>>> print(n)
5
>>>
```

在函数col中，有自己的变量存储空间，内部的n被函数当成是局部变量处理。

```
>>> n=5
>>> def col(a,b):
    c=a*b
    global n
    n =b
    return c+n

>>> col(3,2)
8
>>> n
2
>>>
```

使用global关键字，可以显式声明其为全局变量。



5.2函数参数的传递



5.2.4 函数对变量的作用

全局变量

➤如果需要修改的全局变量是列表：

```
>>>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
    return a*b
>>>s = func("knock~", 2)
>>>print(s, ls)    #测试一下ls值是否改变
knock~knock~ [2]
```

函数在给列表添加值的时候，会自动搜寻自己内存空间内是否存在这个名字的列表，如果不存在，就调用全局变量的列表。

```
>>>ls = []    #ls是全局列表变量
>>>def func(a, b):
    ls = []    #创建了名称为ls的局部列表变量
    ls.append(b)    #将局部变量b增加到全局列表变量ls中
    return a*b
>>>s = func("knock~", 3)
>>>print(s, ls)    #测试一下ls值是否改变
knock~knock~ []
```





5.2函数参数的传递



5.2.4 函数对变量的作用

 Python函数对变量的作用遵循以下规则：

 简单变量，例如整数，函数退出即销毁

 Global关键字显式声明

 组合类型的变量，函数内部如无同名变量，则直接使用全局变量




5.3datetime库的使用



5.3.1 datetime库的概述

 Datetime库是用来处理时间的标准函数库

 Datetime.date 日期表示类, 年、月、日

 Datetime.time 时间表示类, 小时、分、秒、毫秒

 **Datetime.datetime** 日期和时间表示类

 Datetime.timedelta 与时间间隔有关的类

 Datetime.tzinfo 与时区有关的信息表示类

 计算机计算时间的标准是格林威治时间1970年1月1日



5.3 datetime库的使用



5.3.1 datetime库解析

datetime类

datetime.now() 获取当前时间，精确到微秒

```
1 from datetime import datetime
2 today = datetime.now()
3 print(today)
4
```

运行结果：

```
2019-09-29 17:35:40.171000
```




5.3datetime库的使用



5.3.1 datetime库解析

datetime类

-  `datetime.utcnow()` 获取当前时间的世界标准时间表示，精确到微妙

```
from datetime import datetime
today = datetime.utcnow()
print(today)
```

运行结果：

```
2019-09-29 09:41:14.343000
```



5.3 datetime库的使用



5.3.1 datetime库解析

- Datetime.now()和utcnow()都是直接返回一个时间对象
- 可以直接使用datetime实例化一个时间对象，比如

```
from datetime import datetime  
someday = datetime(2020, 5, 28, 23, 59, 59)  
print(someday)
```

运行结果：

```
2020-05-28 23:59:59.000059
```



5.3datetime库的使用



5.3.1 datetime库解析

Datetime类的属性

属性	描述
someday.min	固定返回datetime的最小时间对象， datetime(1,1,1,0,0)
someday.max	固定返回datetime的最大时间对象， datetime(9999, 12, 31, 23, 59, 59, 999999)
someday.year	返回someday包含的年份
someday.month	返回someday包含的月份
someday.day	返回someday包含的日期
someday.hour	返回someday包含的小时
someday.minute	返回someday包含的分钟
someday.second	返回someday包含的秒钟
someday.microsecond	返回someday包含的微秒值

可以使用dir (datetime) 来查看这个类的属性和方法



5.3datetime库的使用



5.3.1 datetime库解析

Datetime类的属性

```
>>> ty=datetime(2015, 12, 11, 21, 56, 7, 0)
>>> print(ty)
2015-12-11 21:56:07
>>> ty.min
datetime.datetime(1, 1, 1, 0, 0)
>>> ty.max
datetime.datetime(9999, 12, 31, 23, 59, 59, 999999)
>>> ty.year
2015
>>> ty.month
12
>>> ty.day
11
>>> ty.hour
21
>>> ty.minute
56
>>> ty.second
7
>>> ty.microsecond
0
... ..
```

可以使用dir (datetime) 来查看这个类的属性和方法



5.3datetime库的使用



5.3.1 datetime库解析

Datetime类的格式化时间方法

属性	描述
<code>someday.isoformat()</code>	采用ISO 8601标准显示时间
<code>someday.isoweekday()</code>	根据日期计算星期后返回1-7,对应星期一到星期日
<code>someday.strftime(format)</code>	根据格式化字符串format进行格式显示的方法

```
>>> ty.isoformat()
'2015-12-11T21:56:07'
>>> ty.isoweekday()
5
>>> ty.strftime('%Y - %m - %d')
'2015 - 12 - 11'
>>>
```



5.3datetime库的使用



5.3.1 datetime库解析

Datetime类的格式化时间控制符

格式化字符串	日期/时间	值范围和实例
%Y	年份	0001~9999, 例如: 1900
%m	月份	01~12, 例如: 10
%B	月名	January~December, 例如: April
%b	月名缩写	Jan~Dec, 例如: Apr
%d	日期	01 ~ 31, 例如: 25
%A	星期	Monday~Sunday, 例如: Wednesday
%a	星期缩写	Mon~Sun, 例如: Wed
%H	小时 (24h制)	00 ~ 23, 例如: 12
%I	小时 (12h制)	01 ~ 12, 例如: 7
%p	上/下午	AM, PM, 例如: PM
%M	分钟	00 ~ 59, 例如: 26
%S	秒	00 ~ 59, 例如: 26



5.3 datetime库的使用



5.3.1 datetime库解析

Strftime和print联合使用：

```
>>> ty.strftime('%Y - %m - %d')  
'2015 - 12 - 11'  
>>> print('今天是{0:%Y}年{0:%m}月{0:%d}日'.format(ty))  
今天是2015年12月11日
```

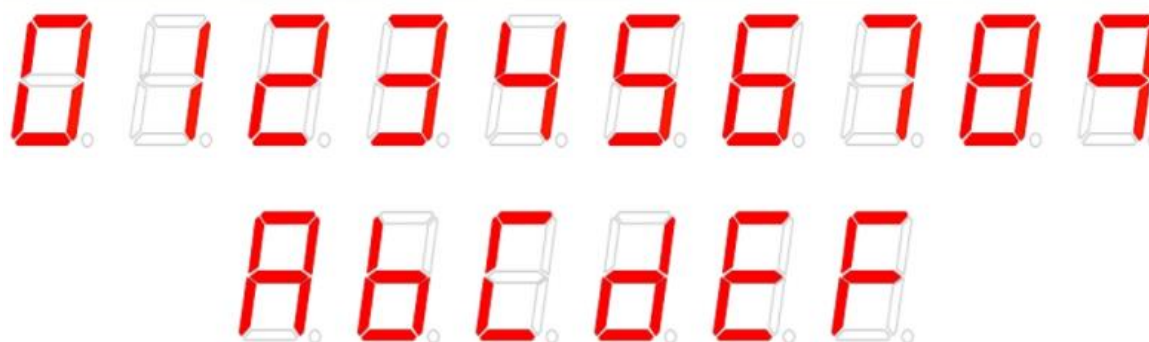
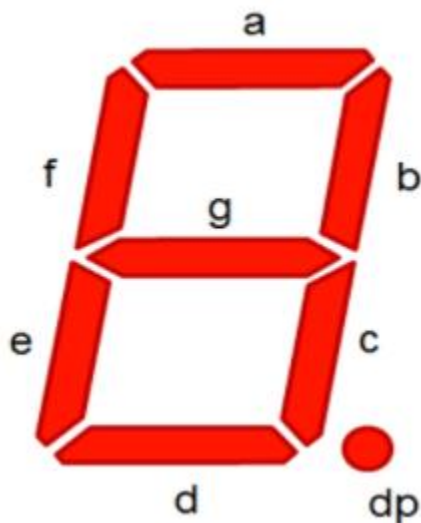


5.4 实例7：七段数码管绘制



5.4 七段数码管

- 由7个棒状的数码管拼接而成，可以显示数字0-9和英语字符。一些电子器件上面使用较多。本实例绘制一个显示当前系统日期的数码管。



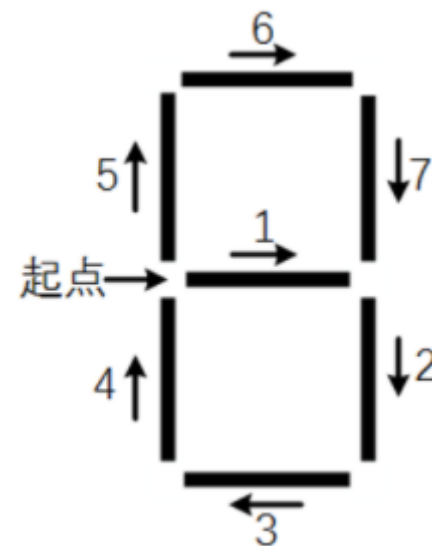


5.4 实例7：七段数码管绘制



5.4 七段数码管

```
1 import datetime
2 import turtle
3 def drawLine(draw): #绘制单端数码管
4     turtle.pendown() if draw else turtle.penup()
5     turtle.fd(40)
6     turtle.right(90)
7 def drawDigit(d): #根据数字绘制七段数码管
8     drawLine(True) if d in [2,3,4,5,6,8,9] else drawLine(False) #第1个数码管段
9     drawLine(True) if d in [0,1,3,4,5,6,7,8,9] else drawLine(False) #第2个数码管段
10    drawLine(True) if d in [0,2,3,5,6,8,9] else drawLine(False) #第3个数码管段
11    drawLine(True) if d in [0,2,6,8] else drawLine(False) #第4个数码管段
12    turtle.left(90) #画笔方向朝上
13    drawLine(True) if d in [0,4,5,6,8,9] else drawLine(False) #第5个数码管段
14    drawLine(True) if d in [0,2,3,5,6,7,8,9] else drawLine(False) #第6个数码管段
15    drawLine(True) if d in [0,1,2,3,4,7,8,9] else drawLine(False) #第7个数码管段
16    turtle.left(180) # 画笔方向定位右边
17    turtle.penup()
18    turtle.fd(20)
19 def drawDate(date): #获得需要输出的数字
20     for i in date:
21         drawDigit(eval(i)) #注意：通过eval()函数将数字变为整数
22 def main():
23     turtle.setup(800,350,200,200)
24     turtle.penup()
25     turtle.fd(-300)
26     turtle.pensize(5)
27     drawDate(datetime.datetime.now().strftime('%Y%m%d'))
28     turtle.done()
29 main()
30
```





5.4 实例7：七段数码管绘制



5.4 七段数码管

```
E:\> 课程 > python > 第5次课 > 实例 > 生日歌5.4七段管绘制.py > ...
1  import turtle, datetime
2  def drawLine(draw): #绘制单端数码管
3      turtle.pendown() if draw else turtle.penup()
4      turtle.fd(40)
5      turtle.right(90)
6  def drawDigit(d): #根据数字绘制七段数码管
7      drawLine(True) if d in [2,3,4,5,6,8,9] else drawLine(False) #第1个数码管段
8      drawLine(True) if d in [0,1,3,4,5,6,7,8,9] else drawLine(False) #第2个数码管段
9      drawLine(True) if d in [0,2,3,5,6,8,9] else drawLine(False) #第3个数码管段
10     drawLine(True) if d in [0,2,6,8] else drawLine(False) #第4个数码管段
11     turtle.left(90) #画笔方向朝上
12     drawLine(True) if d in [0,4,5,6,8,9] else drawLine(False) #第5个数码管段
13     drawLine(True) if d in [0,2,3,5,6,7,8,9] else drawLine(False) #第6个数码管段
14     drawLine(True) if d in [0,1,2,3,4,7,8,9] else drawLine(False) #第7个数码管段
15     turtle.left(180) # 画笔方向定位右边
16     turtle.penup()
17     turtle.fd(20)
18  def drawDate(date): #获得需要输出的数字
19     for i in date:
20         drawDigit(eval(i)) #注意：通过eval()函数将数字变为整数
21  def main():
22     turtle.setup(800,350,200,200)
23     turtle.penup()
24     turtle.fd(-300)
25     turtle.pensize(5)
26     drawDate(datetime.datetime.now().strftime('%Y%m%d'))
27     turtle.done()
28  main()
29
```



5.4 实例7：七段数码管绘制



5.4 七段数码管

```
8 | drawLine(True) if d in [2,3,4,5,6,8,9] else drawLine(False) #第1个数码管段
```

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



5.4 实例7：七段数码管绘制



5.4 七段数码管

```
1. import datetime
2. import turtle
3. def drawLine(draw): #绘制单端数码管
4.     turtle.pendown() if draw else turtle.penup()
5.     turtle.fd(40)
6.     turtle.right(90)
7. def drawDigit(d): #根据数字绘制七段数码管
8.     drawLine(True) if d in [2,3,4,5,6,8,9] else drawLine(False) #第1个数码管段
9.     drawLine(True) if d in [0,1,3,4,5,6,7,8,9] else drawLine(False) #第2个数码管段
10.    drawLine(True) if d in [0,2,3,5,6,8,9] else drawLine(False) #第3个数码管段
11.    drawLine(True) if d in [0,2,6,8] else drawLine(False) #第4个数码管段
12.    turtle.left(90) #画笔方向朝上
13.    drawLine(True) if d in [0,4,5,6,8,9] else drawLine(False) #第5个数码管段
14.    drawLine(True) if d in [0,2,3,5,6,7,8,9] else drawLine(False) #第6个数码管段
15.    drawLine(True) if d in [0,1,2,3,4,7,8,9] else drawLine(False) #第7个数码管段
16.    turtle.left(180) # 画笔方向定位右边
17.    turtle.penup()
18.    turtle.fd(20)
19. def drawDate(date): #获得需要输出的数字
20.     for i in date:
21.         drawDigit(eval(i)) #注意：通过eval()函数将数字变为整数
22. def main():
23.     turtle.setup(800,350,200,200)
24.     turtle.penup()
25.     turtle.fd(-300)
26.     turtle.pensize(5)
27.     drawDate(datetime.datetime.now().strftime('%Y%m%d'))
28.     turtle.done()
29. main()
```





5.4 实例7：七段数码管绘制



5.4 七段数码管

进阶：绘制更加复杂的数码管

20 16 年 09 月 25 日



5.4 实例7：七段数码管绘制



```
1 import datetime
2 import turtle
3 def drawGap(): #绘制数码管间隔
4     turtle.penup()
5     turtle.fd(5)
6 def drawLine(draw): #绘制单端数码管
7     drawGap()
8     turtle.pendown() if draw else turtle.penup()
9     turtle.fd(40)
10    drawGap()
11    turtle.right(90)
12 def drawDigit(d): #根据数字绘制七段数码管
13    drawLine(True) if d in [2,3,4,5,6,8,9] else drawLine(False) #第1个数码管段
14    drawLine(True) if d in [0,1,3,4,5,6,7,8,9] else drawLine(False) #第2个数码管段
15    drawLine(True) if d in [0,2,3,5,6,8,9] else drawLine(False) #第3个数码管段
16    drawLine(True) if d in [0,2,6,8] else drawLine(False) #第4个数码管段
17    turtle.left(90) #画笔方向朝上
18    drawLine(True) if d in [0,4,5,6,8,9] else drawLine(False) #第5个数码管段
19    drawLine(True) if d in [0,2,3,5,6,7,8,9] else drawLine(False) #第6个数码管段
20    drawLine(True) if d in [0,1,2,3,4,7,8,9] else drawLine(False) #第7个数码管段
21    turtle.left(180) # 画笔方向定位右边
22    turtle.penup()
23    turtle.fd(20)
```




5.4 实例7：七段数码管绘制



```
24 def drawDate(date): #获得需要输出的数字
25     turtle.pencolor('red')
26     for i in date:
27         if i == '-':
28             turtle.write('年',font=('Arial',18,'normal'))
29             turtle.pencolor('green')
30             turtle.fd(40)
31         elif i == '=':
32             turtle.write('月',font=('Arial',18,'normal'))
33             turtle.pencolor('blue')
34             turtle.fd(40)
35         elif i == '+':
36             turtle.write('日',font=('Arial',18,'normal'))
37         else:
38             drawDigit(eval(i)) #注意：通过eval()函数将数字变为整数
39 def main():
40     turtle.setup(800,350,200,200)
41     turtle.penup()
42     turtle.fd(-300)
43     turtle.pensize(5)
44     drawDate(datetime.datetime.now().strftime('%Y-%m=%d+'))
45     turtle.done()
46 main()
```



5.5 代码复用和模块化设计



5.5 代码复用和模块化设计

函数是程序的一种基本抽象方式，它将一系列代码组织起来通过命名供其他程序使用。函数封装的直接好处是代码复用，任何其他代码只要输入参数即可调用函数，从而避免相同功能代码在被调用处重复编写。代码复用产生了另一个好处，当更新函数功能时，所有被调用处的功能都被更新。



5.5 代码复用和模块化设计



5.5 代码复用和模块化设计

当程序的长度在百行以上，如果不划分模块就算是最好的程序员也很难理解程序含义程序的可读性就已经很糟糕了。解决这一问题的最好方法是将一个程序分割成短小的程序段，每一段程序完成一个小的功能。无论面向过程和面向对象编程，对程序合理划分功能模块并基于模块设计程序是一种常用方法，被称为“模块化设计”。



5.5 代码复用和模块化设计



5.5 代码复用和模块化设计

模块化设计一般有两个基本要求：

- 紧耦合：尽可能合理划分功能块，功能块内部耦合紧密；
- 松耦合：模块间关系尽可能简单，功能块之间耦合度低。

使用函数只是模块化设计的必要非充分条件，根据计算需求合理划分函数十分重要。一般来说，完成特定功能或被经常复用的一组语句应该采用函数来封装，并尽可能减少函数间参数和返回值的数量。



5.6函数的递归



5.6.1 递归的定义

- 函数内部对自己本身的调用，即递归。
- 递归的例子：

数学上有个经典的递归例子叫阶乘，阶乘通常定义为：

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

这个关系给出了另一种方式表达阶乘的方式：

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



5.6函数的递归



5.6.1递归的定义

递归函数的特征：

- 1.存在基例。
- 2.递归链最终要以基例结束。

这个关系给出了另一种方式表达阶乘的方式：

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



5.6函数的递归



5.6.1递归的使用方法

以阶乘为例：

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

```
1 def factorial(num):
2     if num == 0:
3         return 1
4     else:
5         return num * factorial(num-1)
6 n = eval(input("请输入一个整数: "))
7 print(factorial(abs(int(n))))
8
```

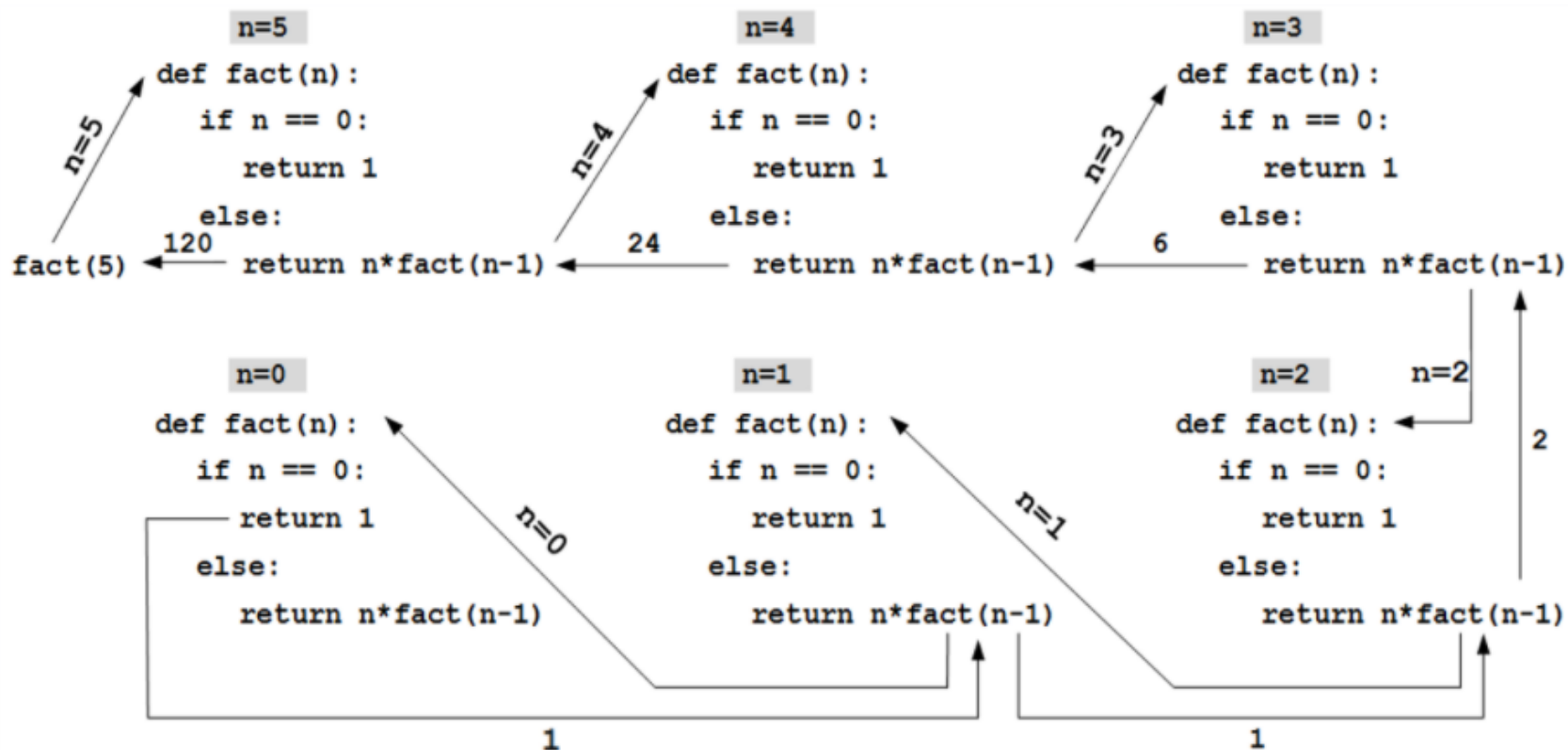


5.6函数的递归



5.6.1 递归的使用方法

阶乘的递归流程图：





5.6函数的递归



5.6.1递归的使用方法

字符串反转：

```
1  def reverse(s):  
2      if s == "":  
3          return s  
4      else:  
5          return reverse(s[1:]) + s[0]  
6  str = input("请输入一个字符串: ")  
7  print(reverse(str))  
8  
9
```



5.7科赫曲线绘制



5.7.1科赫曲线绘制

科赫曲线：1904年数学家科赫提出，科赫曲线类似于雪花状。

科赫曲线绘制过程：

一个长度为 L 的曲线，平均分成3段，其中中间的一段用一边长是 $L/3$ 的等边三角形的两个边代替，即得1阶赫科曲线。再对每个线段重复，即的2阶，依次类推，直至 n 阶。



5.7科赫曲线绘制



5.7.1科赫曲线绘制

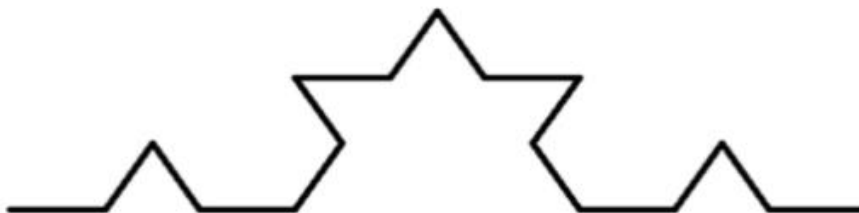
0阶科赫曲线



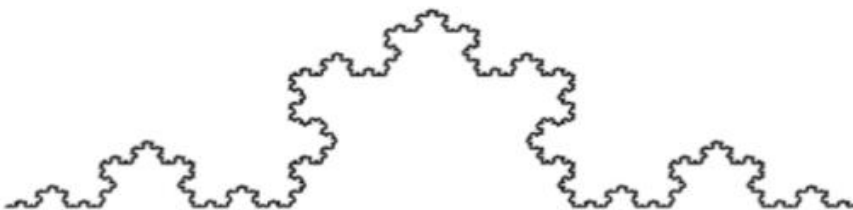
1阶科赫曲线



2阶科赫曲线



5阶科赫曲线





5.7科赫曲线绘制



5.7.1科赫曲线绘制

E: > 课程 > python > 第5次课 > 实例 >  赫科曲线绘制.py > ...

```
1  #e8.1DrawKoch.py
2  import turtle
3  def koch(size, n):
4      if n == 0:
5          turtle.fd(size)
6      else:
7          for angle in [0, 60, -120, 60]:
8              turtle.left(angle)
9              koch(size/3, n-1)
10 def main():
11     turtle.setup(800,400)
12     turtle.speed(0) #控制绘制速度
13     turtle.penup()
14     turtle.goto(-300, -50)
15     turtle.pendown()
16     turtle.pensize(2)
17     koch(600,6) # 0阶科赫曲线长度, 阶数
18     turtle.hideturtle()
19     turtle.done()
20 main()
```



5.7科赫曲线绘制



5.7.1科赫曲线绘制

```
1 #e8.2DrawKoch.py
2 import turtle
3 def koch(size, n):
4     if n == 0:
5         turtle.fd(size)
6     else:
7         for angle in [0, 60, -120, 60]:
8             turtle.left(angle)
9             koch(size/3, n-1)
10 def main():
11     turtle.setup(600,600)
12     turtle.speed(0)
13     turtle.penup()
14     turtle.goto(-200, 100)
15     turtle.pendown()
16     turtle.pensize(2)
17     level = 5
18     koch(400,level)
19     turtle.right(120)
20     koch(400,level)
21     turtle.right(120)
22     koch(400,level)
23     turtle.hideturtle()
24 main()
```



5.8pyhton内置函数



5.8python内置函数

Python共内置了68个函数，常见的需要掌握的大概有36个。

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



下周课程&课后作业



第六章 Python组合数据类型

课后练习

- 实现isOdd () 函数，参数为整数。参数如果是奇数，返回真 (True)。否则假 (false)。

代码文件命名：5-1isOdd

- 实现multi () 函数，参数个数不限，返回所有参数的乘积。

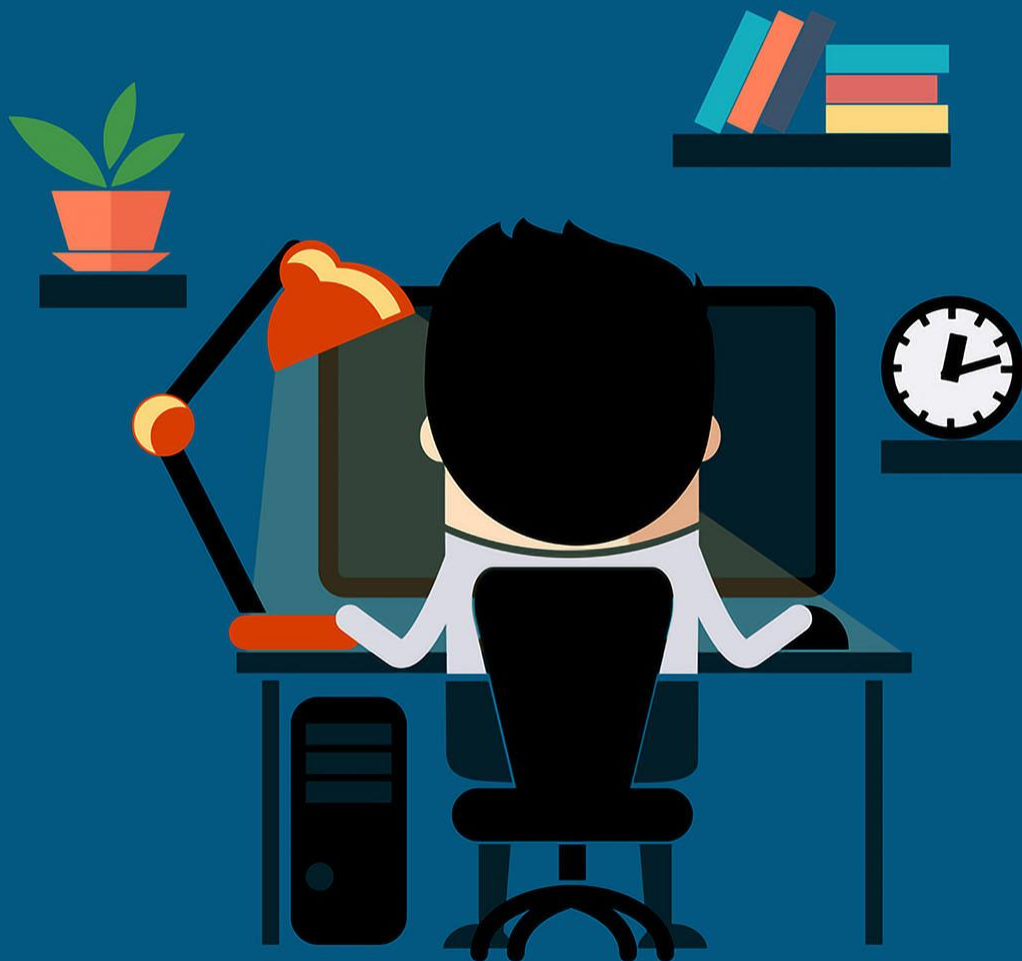
代码文件命名：5-2multi

- 使用datetime库，获取系统时间并按照自己喜欢个一种格式输出。

代码文件命名：5-3mysysdatetime

.py代码文件打包(4.学号+姓名)发送到
python_xxmu@163.com

编程辣么好，还等什么？开始学习吧！



Programing is an Art