第1讲 python的基本语法

孟权令

计算机科学与技术学院

18463103158, quanling.meng@hit.edu.cn

第1讲 python的基本语法

2

一、基本书写规则

二、变量、表达式与数字类型

三、基本控制结构

四、模块化程序设计—函数

基本书写规则

3

python中的词、句、段

```
#求1+2+3+5+...+n的值
'''
@author zxd
@date 2025-09-01
'''
s = 0
n = int(input("input a number:"))
if n>0:
    for i in range(n+1):
        s += i
    print("1+2+...+n=%d"%s)
else:
    print("n<=0")
```

```
# 行注释
'''
段注释—注释不是程序组成部分，运行时被忽略
'''
```

语句：换行或使用分号‘;’如a=1;b=2
输入语句：依据功能区分语句，如赋值语句等
分段：+ 换行
缩进

```
如，if n>0: #if结构段落
    for i in range(n+1): #if子句，循环段落
        s += i #循环子句
```

子句必须对齐

第1讲 python的基本语法

一、基本书写规则

二、变量、表达式与数字类型

三、基本控制结构

四、模块化程序设计—函数

变量、表达式与数字类型

5

命名、变量、赋值运算符、保留字、表达式

变量1,变量2,… = 值1,值2,…

```
sName,nAge = 'Xiaoming',19

fPi = 3.1415926 #浮点型变量
def fSum(n):      #函数名称
    s = 0
    for i in range(n+1):
        s += i
    return s

cName = input("input name:")
```

'=': 赋值符号，声明变量及其数据类型

变量：用来保存和表示具体的数据值,唯一性。
命名：用一个标识符表示变量的过程。
标识符：用来命名变量、函数、类、模块等程序元素的名称。

对标识符的要求：
(1)首字符不能是数字
(2)中间不能出现空格
(3)长度没有限制
(4)小写敏感，python和Python不同

单选题 1分

6

测练题：表达式cN,nA = ‘Xiaoming‘,19中，cN和nA的数据类型是____。

A 未知数据类型

B cN是字符类型, nA整数类型

C cN是字符串类型, nA整数类型

D cN是字符串类型, nA数字类型

变量、表达式与数字类型

命名、变量、运算符、保留字、表达式

保留字(Keyword, 关键字, 33）

【概念】指被编程语言内部定义并保留使用的标识符

【作用】用来构成程序整体框架、表达关键值和具有结构性的复杂语义等

【注意】编写程序时不能定义与保留字相同的标识符

if	else	elif	for	while	break	continue
try	except	finally	raise	assert	pass	def
and	not	or	in	None	True	False
class	return	with	as	global	from	nonlocal
lambda	import	del	yield	is		

变量、表达式与数字类型

数字类型—通过变量赋值确定类型；运算符

【分类】整数、浮点数(科学计数法)、高精度数、复数、分数

【整数与浮点数】通过变量赋值确定类型

【高精度数与分数】

```
#4.高精度数
>>> import decimal
>>> a = decimal.Decimal('3.141592653')

#5.分数
>>> from fractions import Fraction
>>> Fraction(12,20)   #Fraction(3, 5)
```

```
>>> a = 5; b = 2.3  #1. 整型与浮点型
>>> type(a)
<class 'int'>
>>> type(b)
<class 'float'>
>>> c = 2.0e-3  #2.科学计数
>>> d = 2 – 3j   #3.复数
```

变量、表达式与数字类型

9

数字类型与运算符

【分类】一元运算符(+,-,...)、二元运算(+, -, *, /,%, //, **,...)、多元运算符(...if...else...)

优先级	运算符	描述
16 (最高)	`()`、`[]`、`{}`	括号、索引、字典
15	`**`	幂运算
14	`+` (一元正)、`-` (一元负)、`~`	按位取反、一元运算符
13	`*`、`/`、`%`、`//`	乘法、除法、取模、整除
12	`+`、`-`	加法、减法
11	`<<`、`>>`	左移、右移
10	`&`	按位与
9	`^`	按位异或
8	`|`	按位或
7	`==`、`!=`、`>`、`<`、`>=`、`<=`	比较运算符
6	`is`、`is not`	身份比较
5	`in`、`not in`	成员比较
4	`not`	逻辑非
3	`and`	逻辑与
2	`or`	逻辑或
1	`=`、`+=`、`-=` 等	赋值运算符
0 (最低)	条件表达式	条件表达式

运算符

单选题 1分

设置

10

测练题：请写出下列程序的运行结果____。

```
>>> import decimal
>>> a = decimal.Decimal('3.14')
>>> b = decimal.Decimal('1.20')
>>> c = a*b
>>> print(c)
```

```
>>> from fractions import Fraction
>>> a=Fraction(1,2)
>>> b=Fraction(8,3)
>>> print(a + b)
```

提交

变量、表达式与数字类型

实例分析与设计

【例1-1】判断一个4位整数是否为回文数

➢ 分析：$a*10^3 + b*10^2 + c*10 + d == d*10^3 + c*10^2 + b*10 + a$

```
# 1221为一个回文数
n=input("请输入一个四位数：")        # 输入，字符串
n=int(n)                            # 转换为整数
a=n//1000                           # 取千位,赋给变量a
b=n//100%10                         # 取百位,如n=1221//100=12,12%10=2
c=n//10%10                          # 取十位,赋给变量c
d=n%10                              # 取个位,赋给变量d
m=d*1000+c*100+b*10+a               # 按分析，构造新数m
result=(n==m)                       # 判断是否为回文数，返回boolean类型
print("回文数的判断结果是：", result) #输出结果
```

m = n//1000 + n//100%10*10 + n//10%10*100 + n%10*103**

实例

变量、表达式与数字类型

实例分析与设计

【例1-2】 24点游戏。程序随机生成4张牌的面值，游戏者用其组成一个表达式输入。程序计算该表达式的值。

```
print("4张牌的面值分别为：2，3，1，6：")
strExpr=input("请输入你组合出的表达式：")
print(strExpr," = ",eval(strExpr))
```

```
4张牌的面值分别为：2，3，1，6：
请输入你组合出的表达式：6*((3-1)*2)
6*((3-1)*2) = 24
```

实例

第1讲 python的基本语法

13

一、基本书写规则

二、变量、表达式与数字类型

三、基本控制结构

四、模块化程序设计—函数

基本控制结构

程序的顺序执行与条件分支

程序按照语句的书写次序自上而下顺序执行

【例1-3】输入圆的半径，计算圆的周长与面积

```
pi=3.1415926
r=float(input("输入圆半径："))
c=2*pi*r
s=pi*r**2
print("圆周长为：%.2f"%c,";圆的面积为：%.2f"%s)
```

输入圆半径：1
圆周长为：6.28 ;圆的面积为：3.14

基本控制结构

15

程序的顺序执行与条件分支

【分类】一路分支if，二路分支if...else，多路分支if...elif...else

【一路分支if】

```
if<条件表达式>:
    <语句块>
```

【例1-4】从键盘输入两个数，输出它们的最大值

```
a=int(input("输入第一个数："))
b=int(input("输入第二个数："))
Max=a
if Max<b:
    Max=b
print("max=",Max)
```

```
输入第一个数：25
输入第二个数：36
max= 36
```

基本控制结构

程序的顺序执行与条件分支

【二路分支if...else】

```
if<条件表达式>:
        <语句块1>
else:
        <语句块2>
```

【例1-4】编写程序，解一元二次方程a*x2+bx+c=0。用户输 入系数a, b, c，如果有实根计算实根并显示，如果没有， 显示“没有实根”

```
a, b, c = eval(input('输入a, b, c:'))
beta = b*b-4*a*c
if beta >= 0:
    x1=(-b+beta**0.5)/(2*a); x2=(-b-beta**0.5)/2/a
    print("x1=",x1,";x2=%.2f"%x2)
else:
    print("没有实根")
```

```
输入a,b,c:2,3.0,1
x1= -0.5 ;x2=-1.00
```

变量、表达式与数字类型

实例分析与设计

【例1-5】 24点游戏。程序随机生成4张牌的面值，游戏者用其组成一个表达式输入。程序计算该表达式的值、输出并判断正误。

```
print("4张牌的面值分别为：2，3，1，6：")
strExpr=input("请输入你组合出的表达式：")
tw = eval(strExpr)
print(strExpr," = ",tw)
if tw==24:
    print("结果正确")
else:
    print("结果错误")
```

```
4张牌的面值分别为：2，3，1，6：
请输入你组合出的表达式：2+3+1+6
2+3+1+6 = 12
结果错误
```

实例

基本控制结构

程序的顺序执行与条件分支

【多路分支if...elif...else 】

```
if<条件1>:
    <语句块1>
elif<条件2>:
    <语句块2>
 ......
elif<条件n>:
    <语句块n>
else:
    <语句块n+1>
```

可以省略

【例1-6】输入一个成绩，判断其所处的分段等级（90分及以上A，80分及以上B，70分及以上C，60分及以上D，60以下为E）。

```
a = int(input("input a score:"))
level = 'e'
if a>=90:
   level = 'A'
elif a>=80:
   level = 'B'
elif a>=70:
   level = 'C'
elif a>=60:
   level = 'D'
else:
   level='E'
print(level)
```

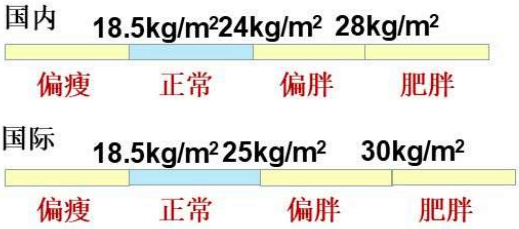
基本控制结构

程序的顺序执行与条件分支

【例1-6】身体质量指数，是BMI(Body Mass Index)指数，简称体质指数，是国际上常用的衡量人体胖瘦程度以及是否健康的一个标准。编写一个程序计算自己的BMI及分类。

```
h, w = eval(input("请输入身高(米)和体重(kg)[用,分隔]："))
bmi = w/(h**2)
print("BMI数值为:%0.2f"%bmi)
if bmi < 18.5:
    who, dom = "偏瘦","偏瘦"
elif 18.5 <= bmi < 24:
    who, dom = "正常","正常"
elif 24 <= bmi < 25:
    who, dom = "正常","偏胖"
elif 25 <= bmi < 28:
    who, dom = "偏胖","偏胖"
elif 28 <= bmi < 30:
    who, dom = "偏胖","肥胖"
else:
    who, dom = "肥胖","肥胖"
print("BMI指标为：国际'{0}',国内'{1}'".format(who, dom))
```

$$BMI=体重(kg)/身高m^2$$



条件分支

基本控制结构

如何应对如 $(x+a)^n=\sum_{k=0}^{n}\binom{n}{k}x^k a^{n-k}$ 形式的大量迭代？循环

【分类】遍历循环**for**与无限循环**while**。

【格式】

```
for<variable> in <iterator>:
        statements1
 else:
        <statements2>
```

典型应用

循环N次：0—N-1
```
for i in range(N):
    <语句块>
```

遍历文件每一行：
```
for line in fi:
    <语句块>
```

遍历字符串/列表：
```
for c in s:
    <语句块>
```

【例1-7】求1~n之间正整数的平方和。n由用户输入。

- 问题分析

  $sum=1^2+2^2+3^2+\ldots+n^2$

- 计算模型

  $sum+=i*i \qquad i\in[1,n]$

```
n=int(input('input n:'))
sum=0
for i in range(1,n+1,1):
   sum+=i*i
else:
   print('sum=',sum)
```

基本控制结构

如何应对如 $(x+a)^n=\sum_{k=0}^{n}\binom{n}{k}x^k a^{n-k}$ 形式的大量迭代？循环

Jacques Bernoulli

【例1-8】雅各布.伯努利的利率计算（复利计算）

$$e=\lim_{n\to\infty}\frac{1}{i!}$$

- 复利计算即为计算e的近似值。循环终止条件可设为最后一项的值小于10^{-6}
- 数学模型：e≈1+1/1!+1/2!+...+1/n!
- 计算模型

```
u=1
ev=1
u=u/i      i∈[1,n]
ev+=u      u>10E-6
```

```
u=1;ev=1;i=1
while(u>1e-6):
    u=u/i;  ev+=u; i+=1
print("ev=",ev)
```

基本控制结构

如何应对如 $(x+a)^n=\sum_{k=0}^{n}\binom{n}{k}x^k a^{n-k}$ 形式的大量迭代？循环

循环保留字：break、continue

```
for s in "Hitter":
    if s=="t":
        continue
    print(s,end="")
```

Hier

```
for s in "Hitter":
    if s=="t":
        break
    print(s,end="")
```

Hi

循环

基本控制结构

综合应用

【例题1-9】在天天向上的同学，在偶尔放假后，工作日怎样努力，才能赶上不放假一直努力的同学?

- 问题分析

  每天努力所取得的成就T = dayUp *(1 + dayFactor)^{365}

  只有工作日努力的成就S = T – 休假的下滑

  所以，T、S、dayUp之间不存在简单的线性关系

循环

- 解决方案

  (1) 计算天天向上的同学的最终成就T

  (2) 计算只在工作日努力同学的成就S，若S<T，则dayFactor+=0.001(梯度计算)，再次计算S值…直到S>=T为止。

  (3) S的计算方法会(2)测试中被反复使用，能否做独立功能进行封装?

第1讲 python的基本语法

24

一、基本书写规则

二、变量、表达式与数字类型

三、基本控制结构

四、模块化程序设计—函数

模块化程序设计—函数

当程序越来越宏大，如何进行有效管理？模块化程序设计—函数

【分类】内置函数（68）、库函数与自定义函数等。

函数名	描述
eval(<字符串>)	将<字符串>解析并执行为Python表达式
input(prompt)	从用户那里获取输入. prompt为提示字符串
int(x)	将x转换为整数。x可以是浮点数或字符串
float(x)	将x转换为浮点数。x可以是整数或字符串
ord(x)	将x转换为其ASCII值。x必须为单个字符
abs(x)	x的绝对值
divmod(x, y)	(x/y,x%y),输出为二元组形式(也称为元组类型)
pow(x, y[,z])	(x**y%z, [.]表示该参数可以省略，即pow(x,y),它与x**Yy相同
round(x[,ndigits])	对x四舍五入，保留ndigits 位小数。round(x)返回四舍五入的整数值
chr(x)	将x转换为字符，x通常为字母的ASCII值
…	……

模块化程序设计—函数

当程序越来越宏大，如何进行有效管理？模块化程序设计—函数

【分类】内置函数、库函数与自定义函数等。

- 关于内置函数print

【格式】print(*objects, sep=' ', end='\n', file=None, flush=False)

【作用】将 objects 打印输出至 file 指定的文本流，以 sep 分隔并在末尾加上 end。

【参数】sep 、end 、file 和 flush 必须以关键字参数的形式给出。

```
print("hit","edu","cn",sep=".",end=": ")
print("Welcome","hitters",sep=', ',end="\n")
```

---

hit.edu.cn: Welcome, hitters

模块化程序设计—函数

软件工程管理基础：函数、模块、包与库

【软件工程管理】模块、包、库。

【模块】具有相对完整功能代码集合的python文件,解决代码复用问题。

```
# math_utils.py  #模块定义
def sqrtz(x): #计算 x 的开方
    return x **0.5
```

引用模块方法

```
# main.py
import math_utils      # 引用模块
print(math_utils.sqrtz(5))  # 输出 2.236
```

【包】管理 Python 模块命名空间的方式。 “带有 __init__.py 文件的文件夹”

```
data_processing/
├── __init__.py
└── text_processing/
    ├── __init__.py
    └── text_utils.py
```

```
from data_processing.text_processing import text_utils
```

模块化程序设计—函数

模块化程序设计—库函数

【库】成套的工具集合。

【引入】 from 库 import 模块/类　或　import 库

函数名(参数)　　|　　库.函数名(参数)

函数名	数学表示	描述
math.pow(x,y)	x^y	返回x的y次幂
math.exp(x)	e^x	返回e的x次寒，e是自然对数
math.sqrt(x)	$\sqrt{x}$	返回x的平方根
math.log(x[,base])	$\log_{base} x$	返回x的对数值，只输入x时，返回自然对数，即ln(x)
math.log10(x)	$\log_{10} x$	返回x的10对数值
math.pi	π	x的绝对值
math.e	e	(x/y,x%y),输出为二元组形式(也称为元组类型)
…		……

模块化程序设计—函数

模块化程序设计—库函数

```
from math import pi     #【例1-3】输入圆的半径，计算圆的周长与面积
r=float(input("输入圆半径："))
c=2*pi*r
s=pi*r**2
print("圆周长为：%.2f"%c,";圆的面积为：%.2f"%s)
```

```
import math    #【例1-1】判断一个4位整数是否为回文数
n = int(input("请输入一个4位整数:"))
m = n//1000 + n//100%10*10 + n//10%10*100 + n%10*math.pow(10,3)
print("数{}判断是否为回文数的结果为：{}".format(n, n == m))
```

实例

模块化程序设计—函数

模块化程序设计—库函数

random 模块

方法	描述
seed()	初始化随机数生成器
randrange()	从 range(start, stop, step) 返回一个随机选择的元素。
randint(a, b)	返回随机整数 N 满足 a <= N <= b。
choice(seq)	从非空序列 seq 返回一个随机元素。 如果 seq 为空，则引发 IndexError。
shuffle(x[, random])	将序列 x 随机打乱位置。
sample(population, k, *, counts=None)	返回从总体序列或集合中选择的唯一元素的 k 长度列表。 用于无重复的随机抽样。
random()	返回 [0.0, 1.0) 范围内的下一个随机浮点数。
uniform()	返回一个随机浮点数 N， 当 a <= b 时 a <= N <= b， 当 b < a 时 b <= N <= a 。
triangular(low, high, mode)	返回一个随机浮点数 N， 使得 low <= N <= high 并在这些边界之间使用指定的 mode 。 low 和 high 边界默认为零和一。 mode 参数默认为边界之间的中点，给出对称分布。
betavariate(alpha, beta)	Beta 分布。 参数的条件是 alpha > 0 和 beta > 0。 返回值的范围介于 0 和 1 之间。
expovariate(lambd)	指数分布。 lambd 是 1.0 除以所需的平均值，它应该是非零的。
gammavariate()	Gamma 分布（ 不是伽马函数） 参数的条件是 alpha > 0 和 beta > 0。
gauss(mu, sigma)	正态分布，也称高斯分布。 mu 为平均值，而 sigma 为标准差。
normalvariate(mu, sigma)	正态分布。 mu 是平均值，sigma 是标准差。

基本控制结构

综合应用

【例题1-9】24点游戏。程序随机生成4张牌的面值（模拟发牌），游戏者用其组成一个表达式输入。程序计算该表达式的值并判断对错。该游戏可循环起进行，直到玩家输入n为止。

分析：J、Q、K、A均为1，故取值[1,10]

```
开始游戏？（Y/N）y
牌的面值分别为：10,2,9,7
请输入你组合出的表达式：10-2+9+7
10-2+9+7 = 24
结果正确
继续游戏？（Y/N）n
游戏结束！
```

```
from random import randint
c=input("开始游戏？（Y/N）")
while c in ['Y','y']:
    a = randint(1,10)
    b = randint(1,10)
    c = randint(1,10)
    d = randint(1,10)
    print("牌的面值分别为：%d,%d,%d,%d"%(a,b,c,d))
    strExpr=input("请输入你组合出的表达式：")
    tw = eval(strExpr)
    print(strExpr," = ",tw)
    if tw==24:
        print("结果正确")
    else:
        print("结果错误")
    c = input("继续游戏？（Y/N）")
print("游戏结束！")
```

模块化程序设计—函数

模块化程序设计—库函数

【turtle库】 内置的绘图标准库，通过一组函数控制画笔的行进动作，完成绘制。

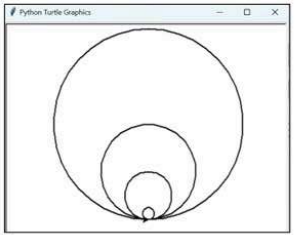
函数名	描述
turtle.setup(width, height, startx, starty)	设置主窗体的大小和位置
turtle.pensize(width)	设置笔尖粗细，width指像素
turtle.penup()	抬起画笔
turtle.pendown()	落下画笔
turtle.color(colorstr)	给画笔设置颜色，colorstr为颜色字符串，如“purple”
turtle.seth(to_angle)	改变画笔方向，to_angle为绝对方向角度值
turtle.fd(distance)	控制画笔沿当前行进方向前进distance距离
turtle.circle(radius, extent=None)	根据radius绘制extent角度的弧形
…	……

模块化程序设计—函数

模块化程序设计—turtle库

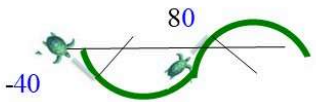
```
import turtle
turtle.pensize(2)  #画笔粗2个像素
turtle.circle(10)  #半径10个像素点
turtle.circle(40)
turtle.circle(80)
turtle.circle(160)
```

创建4个同切圆



turtle库绘制图形有一个基本框架:一个小海龟在坐标系中爬行，有前进、后退、旋转、变换方向等爬行行为，其爬行轨迹绘制成图形。

```
seth(-40) #角度
for i in range(4):
    circle(40,80)# 半径，角度
    circle(-40, 80)
```

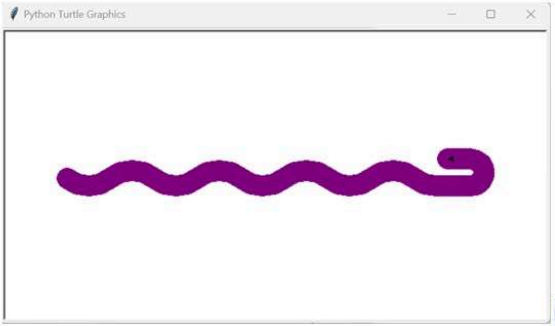


模块化程序设计—函数

模块化程序设计—turtle库

【例题1-10】利用turtle库画一条大python。

```
from turtle import *
setup(650,350,200,200) #主窗体大小及位置
penup() #抬笔
fd(-250) #倒退
pendown() #落笔
pensize(10) #笔线粗25
pencolor("purple") #颜色紫色
seth(-40) #角度
for i in range(4):
    circle(40,80)# 半径，角度
    circle(-40, 80)
circle(40,80/2)
fd(40)
circle(16,180)
fd(40*2/3)
```



模块化程序设计—函数

模块化程序设计—自定义函数

【函数的定义】

```
def 函数名(<形参1,形参2…>):
    <函数体>
    [return <对象>]
```

【调用格式】

```
函数名(<实际参数>)
```

【例1-11】编写函数求出区间[i, j]内所有偶数的和

```
def mySum(i=1, j=5):
    s=0
    for k in range(i,  j+1):
        if k%2:
            continue
        s=s + k
    return s
```

```
>>> mySum()
6
>>> mySum(2,16)
72
```

```
>>> mySum(3)
4
>>> mySum(j=8)
20
```

技巧

```
def mySum(i=1, j):
    …
```

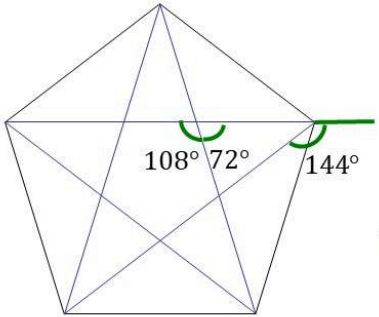
SyntaxError: non-default argument follows default argument

模块化程序设计—函数

模块化程序设计—自定义函数

【例1-12】绘制一个红色的五角星图形

正多边形内角和为：(n-2)*180度→五边形内角和为540度，每个内角108度



```
from turtle import *
def drawStar(color):
    fillcolor(color)
    begin_fill()
    while True:
        forward(200)
        right(144)
        if abs(pos()) < 1:
            break
    end_fill()
```

自定义函数

模块化程序设计—函数

模块化程序设计—递归


John McCarthy

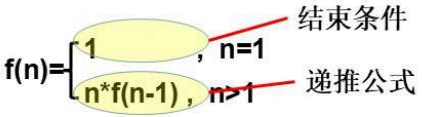
【设计思想】把一个复杂的大问题逐步转换为与原问题相似的小问题，在求解小问题时，又用到原问题的求解方式，直到分解为可以简单或直接求解的小问题，求得小问题的解后，再回归，直到把大问题解决。

【递归算法的设计要点】

（1）递推公式

（2）递归结束条件

【例1-13】编程求n!




```
def f(n):
    if n==1:
        return 1
    return n*f(n-1)
```

模块化程序设计—函数

模块化程序设计—递归

【例1-14】汉诺(Hanoi)塔问题。

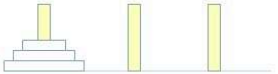


图 汉诺塔

借助B将n个盘子从A移到C =
- 将一个盘子从A移到C　　n=1　结束条件
- 借助C将n-1个盘子从A移到B
  将一个盘子从A移到C　　n>1　递推公式
  借助A将n-1个盘子从B移到C

```
def Hanoi(n,ch1,ch2,ch3):
    if n==1:
        print(ch1,'->',ch3)
    else:
        Hanoi(n-1,ch1,ch3,ch2)
        print(ch1,'->',ch3)
        Hanoi(n-1,ch2,ch1,ch3)
```

自定义函数

模块化程序设计—函数

模块化程序设计—递归

Koch

【例1-15】用递归方式绘制科赫曲线。

➢ 问题分析

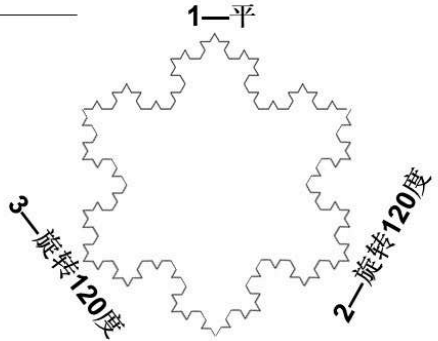
正整数n代表科赫曲线的阶数，表示生成科赫曲线过程的操作次数。

0阶：一个长度为L的直线。 ————————

1阶：

2阶：

3阶：



自定义函数

模块化程序设计—函数

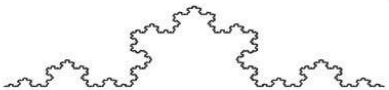
模块化程序设计—递归

Koch

【例1-15】用递归方式绘制科赫曲线。

➢ 程序实现

```
from turtle import *
def koch(size, n):
    if n == 0:
        fd(size)
    else:
        for angle in [0,60, -120, 60]:
            left(angle)
            koch(size/3, n-1)
```



```
def drawKoch():
    setup(800,400)
    speed(0)
    penup()
    goto(-300, -50)
    pendown()
    pensize(2)
    koch(600,5)
    hideturtle()
drawKoch()
```