程序设计入门 – Python

车万翔

哈尔滨工业大学

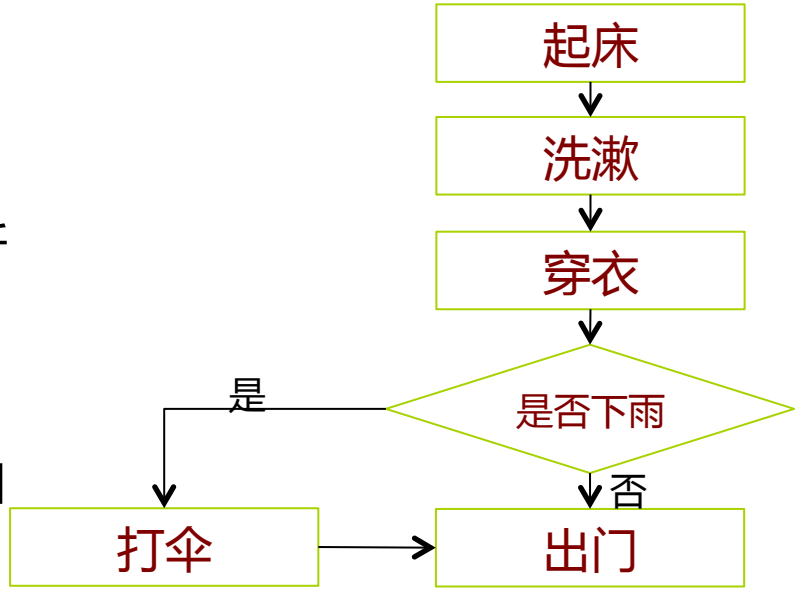
基础知识

车万翔

哈尔滨工业大学

什么是程序？

- 程序（Program）
  - 流程、议程、行程、…
  - 为了完成某项任务，解决某个问题所需要执行的一系列步骤
- 计算机程序
  - 为了完成某项任务，解决某个问题由计算机执行的一系列指令（步骤）



什么是计算机？

- 计算机（Computer）
  - 能够按照程序自动运行的机器
- 组成
  - 硬件（Hardware）
    - 计算机的躯壳
  - 软件（Software）
    - 计算机程序
    - 计算机的灵魂

计算机的发展历史

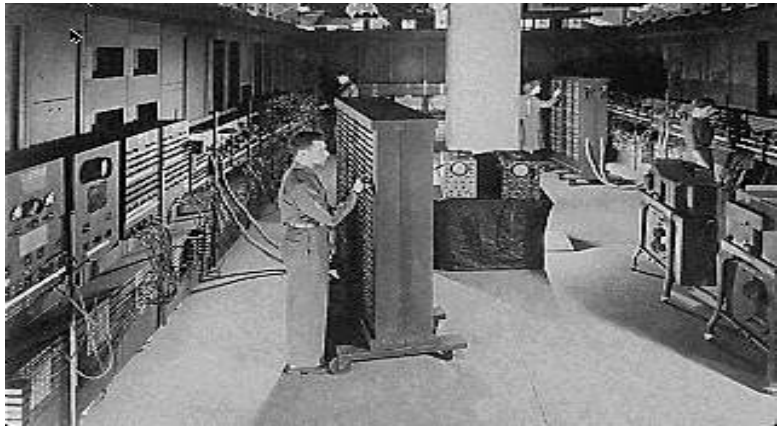
机电计算机

机械计算机

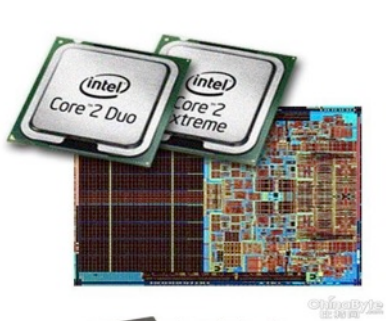
算筹\算盘

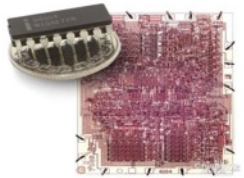
计算机的发展历史

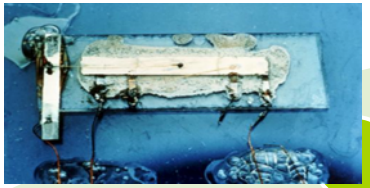
- 电子计算机
  - 1946年，美国宾西法尼亚大学
  - 第一台电子计算机（ENIAC）
    - 数以万计的电子器件
    - 占地170平方米左右
    - 总重量达到30吨
    - 达到每秒钟5000次加法



电子计算机的发展历程







超大规模
集成电路
1972

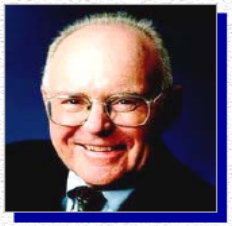
集成电路
1964

晶体管
1959

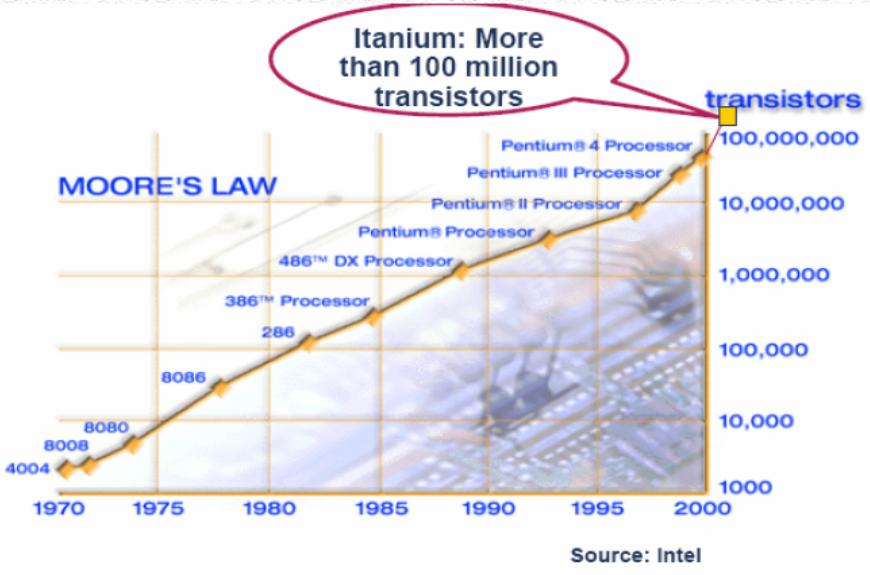
电子管
1946

电子计算机的发展历程

Moore's Law



Gordon E. Moore

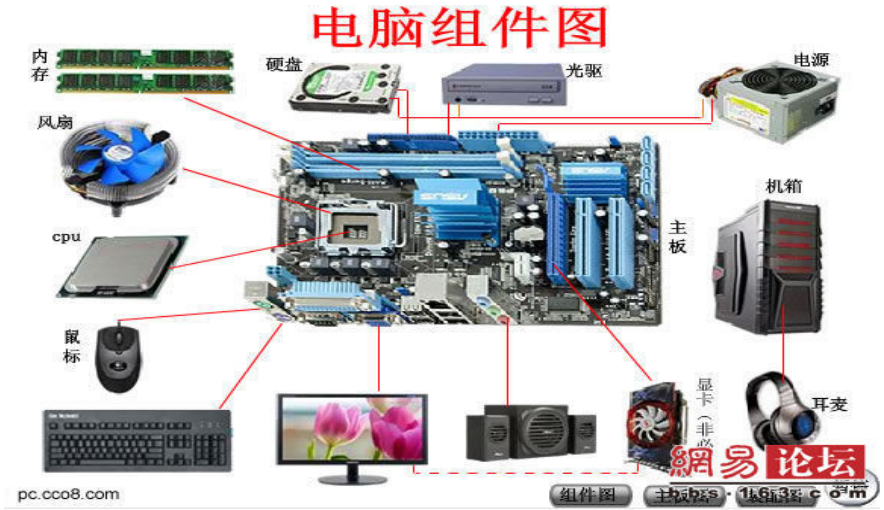


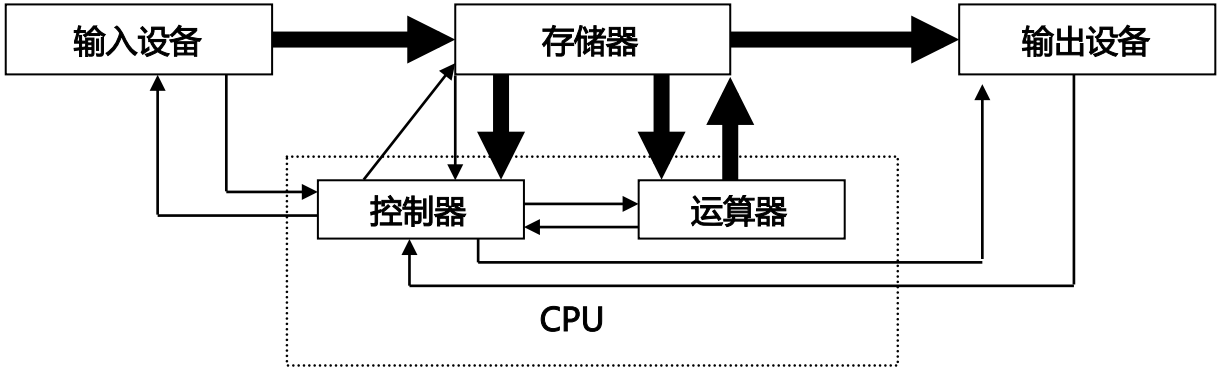
Number of components doubles every 18 months

每18个月芯片能力增长一倍。

计算机硬件

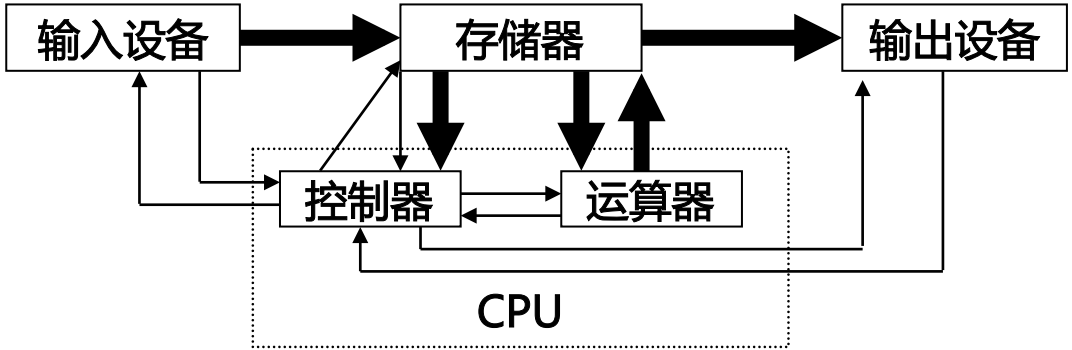
计算机硬件组成


电脑组件图
内存
硬盘
光驱
电源
风扇
机箱
主板
cpu
鼠标
耳麦
显卡（非必
pc.cco8.com
组件图


输入设备
存储器
输出设备
控制器
运算器
CPU

计算机的基本组成

冯•诺依曼体系结构



计算机之父：冯•诺依曼

- 计算机应该按照程序顺序执行
- 采用二进制作为计算机的数制基础

机器执行的程序

10111000
00000001
00000000
00000101
00000001
00000000

- Compute 1+1

人与机器的沟通

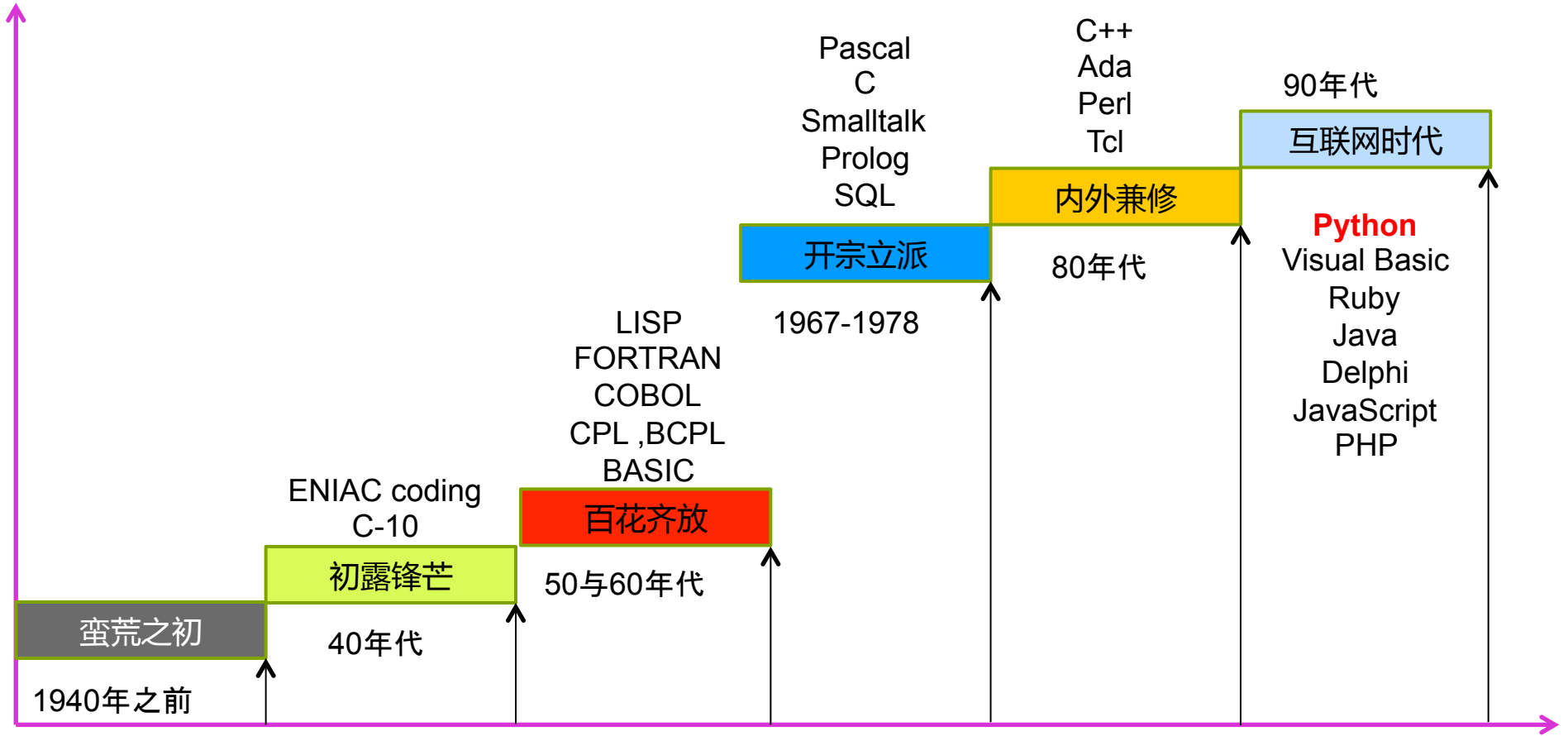
- 不同母语人之间的沟通
  - 一人学习另一人的语言
  - 共同学习第三种语言
- 人与机器的沟通
  - 机器学习人类的语言
    - 自然语言处理
  - 人学习机器的语言
  - 共同学习第三种语言
    - 程序设计语言

MOV AX, 1	10111000 00000001 00000000
ADD AX, 1	00000101 00000001 00000000

- 如：Python 语言

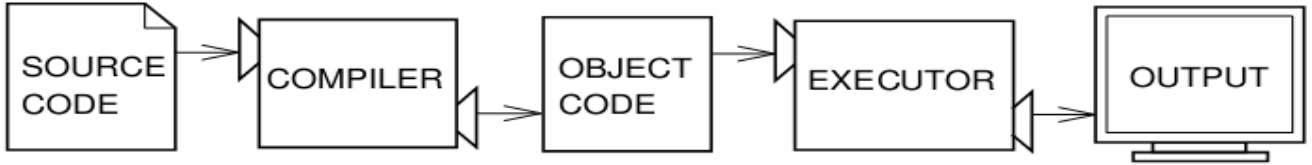
```
print 1+1
```

计算机程序设计语言的历史



高级语言的分类

- 编译型语言（C/C++等）



- 解释型语言（BASIC、Python等）

Python 语言介绍

- 诞生于1989年， 英国发音：/'paɪθən/，美国发音：/'paɪθɑːn/
- 创始人为吉多·范罗苏姆（Guido van Rossum）

Python 语言的特点

- 解释型语言
- 设计哲学是“优雅”、“明确”、“简单”
  - 易学、易用
  - 可读性高
- 开发哲学是“用一种方法，最好是只用一种方法来做一件事”
- 现代编程语言
  - 面向对象
  - 支持泛型设计
  - 支持函数式编程
- 丰富的数据结构和第三方函数库
  - 功能强大

Hello, World!

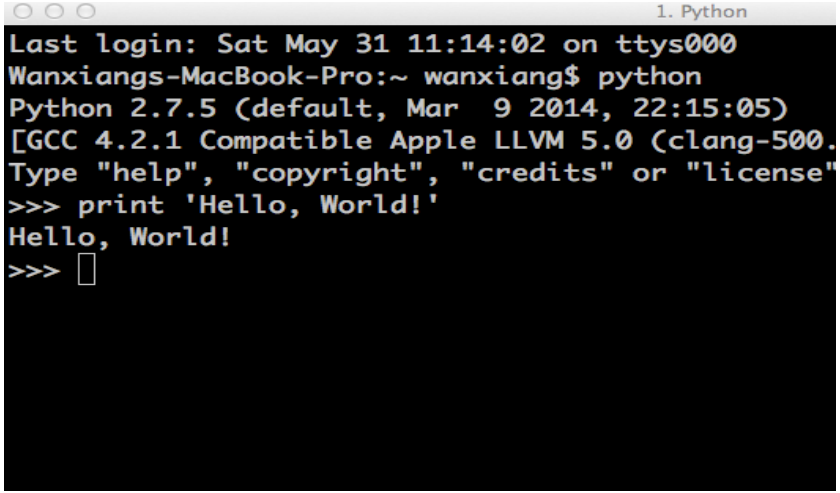
- 它是编程之神的传统咒语，可以帮助你开始这段感情……

```
# 打印Hello, World!到屏幕
print 'Hello, World!'
```

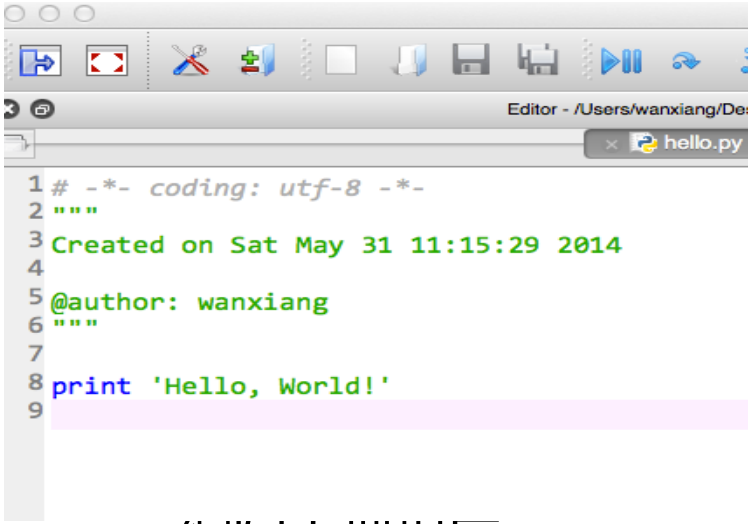
注释

执行语句

控制台和脚本的比较



- 语句功能测试



- 编写大型程序

对象和类型

车万翔

哈尔滨工业大学

学生的属性

- 姓名：小明
- 性别：男
- 年龄：15
- 身高：1.48
- 体重：43.2
- 籍贯：江苏

对象

五种基本对象类型

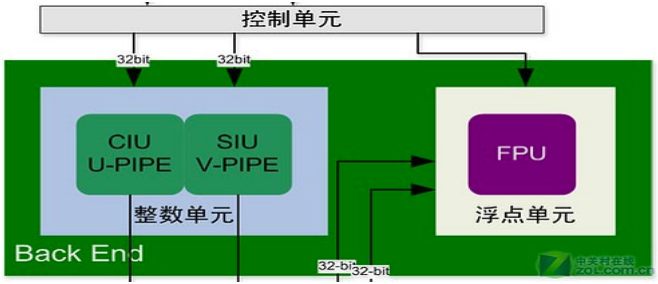
- **字符串（string），简记为 str**
  - 使用 ' ' 或 " " 括起来的一系列字符
- **整数（integer），简记为 int**
  - 十进制：21，八进制：025，十六进制：0x15
- **浮点数（float）**
  - 1.48，21.0，21.，.21，2.1E2
- **布尔数（boolean），简记为 bool**
  - True，False
- **复数（complex）**
  - 1+1j

对象类型

小明	type('小明')	➔<type 'str'>
男	type('男')	➔<type 'str'>
15	type(15)	➔<type 'int'>
1.48	type(1.48)	➔<type 'float'>
43.2	type(43.2)	➔<type 'float'>
江苏	type('江苏')	➔<type 'str'>

为什么区分对象类型？

- **不同类型对象运算规则不同**
  - 如：整数的加法和字符串的加法含义不同
- **不同类型对象在计算机内表示方式不同**
  - 5 ➔ 101，'5' ➔ 1001101
- **为何区分整数与浮点数？**
  - 浮点数表示能力更强
  - 浮点数有精度损失
  - CPU有专门的浮点数运算部件



强制类型转换

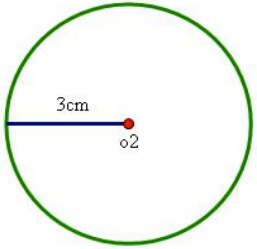
- int('123') ➔ 123
- str(123) ➔ '123'
- float('123') ➔ 123.0
- float(123) ➔ 123.0
- bool(123) ➔ True
- bool(0) ➔ False

运算符

车万翔

哈尔滨工业大学

算术运算



- 若圆的半径为3cm，求圆的面积
  - $\pi \times 3^2$
  - 3.14 * 3 * 3

- 若三名学生的身高分别为：1.65, 1.78, 1.82，求他们的平均身高
  - $\frac{1.65+1.78+1.82}{3}$
  - (1.65 + 1.78 + 1.82) / 3

算术运算符（Arithmetic Operators）

算数运算符	含义	举例
+	加法（Addition）	10 + 20 = 30
-	减法（Subtraction）	10 - 20 = -10
*	乘法（Multiplication）	10 * 20 = 200
/	除法（Division）	10 / 2 = 5
%	求余（Modulus）	10 % 3 = 1
**	指数（Exponent）	2 ** 3 = 8

算术运算示例

- **将华氏度（*F*）转化为摄氏度（*C*）**
  - 转化公式 $C = \frac{5}{9}(F - 32)$
  - 假设 *F* = 75，则相应的Python代码为：
    - 5 / 9 * (75 – 32) ✗
    - 5.0 / 9 * (75 – 32) ✓
- **为什么？**
  - Python 2 中，“/”表示向下取整除（floor division）
  - 两个整数相除，结果也是整数，舍去小数部分
  - 如果有一个数为浮点数，则结果为浮点数

自动类型转换

- **若参与运算的两个对象的类型同，则结果类型不变**
  - 如：1 / 2 = 0
- **若参与运算的两个对象的类型不同，则按照以下规则进行自动类型转换**
  - bool → int → float → complex
  - 如：
    - 1.0 + 3 = 4.0
    - True + 3.0 = 4.0

求余运算符

- **求余运算符（%）**
  - 如：10 % 3 = 1
- **应用**
  - 若今天是星期六，则10天后是星期几？
    - (6 + 10) % 7 = 2
  - 判断一个数 x 是否为偶数
    - x % 2 是否等于 0

math 模块

- **模块（module）**
  - 实现一定的功能的 Python 脚本集合
- **引入模块**
  - import module_name
- **math模块**
  - import math
  - 查看模块内容
    - dir(math)
  - 查看帮助
    - help(math.sin)

```
>>> dir(math)
['__doc__', '__file__', '__name__', '__package__', 'acos',
'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil',
'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp',
'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum',
'gamma', 'hypot', 'isinf', 'isnan', 'ldexp', 'lgamma',
'log', 'log10', 'log1p', 'modf', 'pi', 'pow', 'radians', 'sin',
'sinh', 'sqrt', 'tan', 'tanh', 'trunc']
```

关系运算符（Relational Operators）

- **判断一个数 x 是否为偶数**
  - x % 2 是否等于 0
  - x % 2 == 0
  - 若为True，则 x 为偶数
  - 若为False，则 x 为奇数
- **用于判断两个值的关系**
  - 大小、相等或不相等
- **运算的结果只有两种（布尔型）**
  - 若结果为True，表示条件成立
  - 若结果为False，表示条件不成立

关系运算符

关系运算符	含义	举例
==	等于（equal）	10 == 20 is false
!=，<>	不等于（not equal）	10 != 20 is true
>	大于（greater）	10 > 20 is false
<	小于（less）	10 < 20 is true
>=	大于等于（greater or equal）	10 >= 20 is false
<=	小于等于（less or equal）	10 <= 20 is true

逻辑运算符（Logical Operators）

- **现实世界中处处体现逻辑**
  - 你们班有没有身高一米九以上的男生？ 身高 > 1.9 and 性别 == 男
  - 地铁里禁止喝水吃东西 禁止：喝水 or 吃东西
  - ……
- **逻辑运算符**

关系运算符	含义	举例
and	与（全真才真）	True and False == False
or	或（全假才假）	True or False == True
not	非（真变假、假变真）	not True == False

逻辑运算示例

该图形是否为红色三角形？

颜色 == 红色　　and　　形状 == 三角形

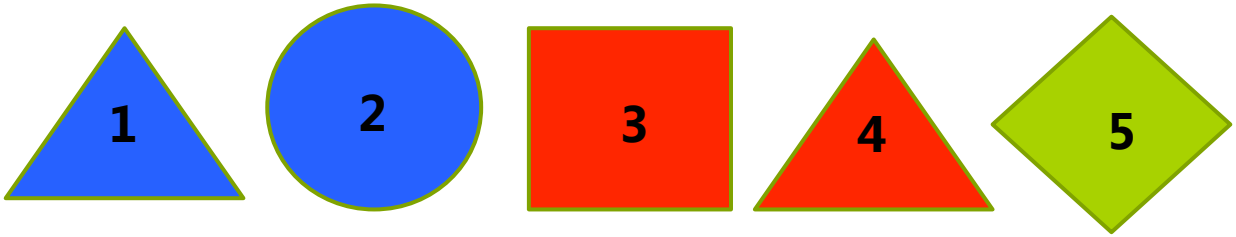
	A	B	
1	F	T	F
2	F	F	F
3	T	F	F
4	T	T	T
5	F	F	F

逻辑运算真值表

- and

A	*B*	*A* and *B*
F	F	F
F	T	F
T	F	F
T	T	T

逻辑运算示例



该图形是否为红色或三角形？

颜色 == 红色　or　形状 == 三角形

	A	B	
1	F	T	T
2	F	F	F
3	T	F	T
4	T	T	T
5	F	F	F

逻辑运算真值表

or

A	*B*	*A* or *B*
F	F	F
F	T	T
T	F	T
T	T	T

逻辑运算示例

该图形是否非红色？

not 颜色 == 红色

	A	
1	F	T
2	F	T
3	T	F
4	T	F
5	F	T

逻辑运算真值表

not

A	*not A*
T	F
F	T

下面哪些是港台女星？

性别 == 女 and (籍贯 == 香港 or 籍贯 == 台湾)

判断闰年

- **如果年份 y 能被 4 整除但是不能被 100 整除，或者能被 400 整除，则是闰年**
  - 2014、1900 年不是闰年
  - 2012、2000 年是闰年


```
(y % 4 == 0 and y % 100 != 0) or (y % 400 == 0)
```

运算符优先级

- **看看下面两个表达式**
  - 2 * 1 + 3　　先乘后加
  - 2 * (1+3)　　先加后乘
- **括号 ( )**
  - 改变了语言内在的默认优先级
  - 具有最高优先级
- **嵌套括号按照由内而外结合**
  - (2 * (1 + 2))**2 == 36
  - 2 * (1 + 2)**2 == 18

运算符优先级

- 括号：( )
- 一元运算：+ ，-
- 幂次：**
- 算术运算：* ，/ ，% ，//
- 算术运算：+ ，-
- 比较运算：== ， != ， <> <= >=
- 逻辑非：not
- 逻辑与：and
- 逻辑或：or
- 赋值运算：=, *=, /=,+=,-=,%=,//=

规则1：

自上而下

括号最高

逻辑最低

规则2：

自左向右

依次结合

变量

车万翔

哈尔滨工业大学

判断闰年

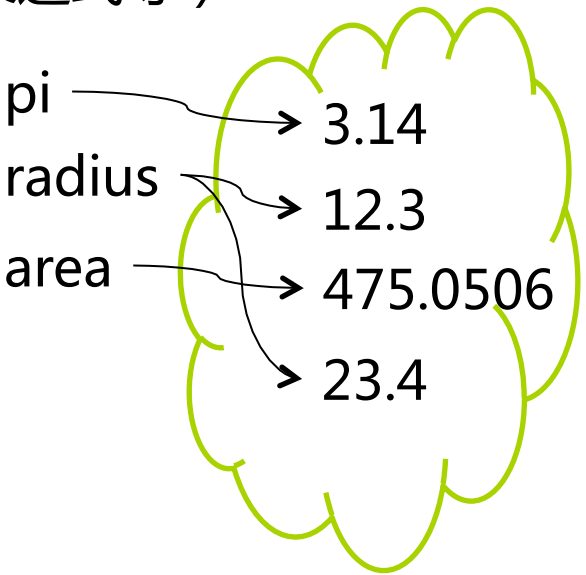
- **如果年份能被 4 整除但是不能被 100 整除，或者能被 400 整除，则是闰年**
  - 2014、1900 年不是闰年
  - 2012、2000 年是闰年


```
(2014 % 4 == 0 and 2014 % 100 != 0) or (2014 % 400 == 0)

(1900 % 4 == 0 and 1900 % 100 != 0) or (1900 % 400 == 0)

(2000 % 4 == 0 and 2000 % 100 != 0) or (2000 % 400 == 0)

(y % 4 == 0 and y % 100 != 0) or (y % 400 == 0)
```

变量（Variable）

- 用于引用（绑定）对象的标识符
- 语法
  - 变量名 = 对象（数值、表达式等）
- 如计算圆面积

```
pi = 3.14
radius = 12.3
area = pi * radius**2
radius = 23.4
print area
```



增量赋值运算符

- 累加
  - count = count + 1
  - 简写为
    - count += 1
- 更多增量赋值运算符

增量赋值	等价表示
x += 2	x = x + 2
x -= 2	x = x - 2
x *= 2	x = x * 2
x /= 2	x = x / 2
x %= 2	x = x % 2
x **= 2	x = x ** 2

标识符（Identifier）

- **标识符**
  - 变量，函数，模块等的名字
- **命名规则**
  - 可以任意长
  - 包含数字和字母、下划线
  - 但首个必须是字母或下划线
  - 大小写敏感
  - 标识符不能是关键字

```
# This is an example of variable identifier.

x = 1
y = 2
my_name = 'x-man'
76trombones = 'big parade' ✗
more@ = 1000000 ✗
class = 'Advanced ' ✗
```

Python的关键字

- Python 2.x 31个关键字（保留字）

and	del	from	not	while
as	elif	global	or	with
assert	else	if	pass	yield
break	except	import	print	class
exec	in	raise	continue	finally
is	return	def	for	lambda
try				

标准（键盘）输入

- **raw_input 函数**
  - 功能：读取键盘输入，将所有输入作为字符串看待
  - 语法：raw_input([prompt])
    - [prompt]是提示符，最好提供
  - 举例：radius = float(raw_input('Radius: '))

- **根据用户输入的圆半径，求解并输出圆的面积**

```
pi = 3.14
radius = float(raw_input('Radius: '))
area = pi * radius**2
print area
```

标准（控制台）输出

- **print 关键字**
  - 功能：将对象的值输出到控制台上
  - 语法：print object or variable
  - 举例：print area
- **如何将多个对象输出到一行？**
  - print 'The area for the circle of radius', radius, 'is', area
- **如何将一个字符串输出到多行？**
  - print 'Hello\nWorld!'

常用转义符	含义
\n	回车（Newline）
\t	制表符（Tab）
\\	一个 \
\a	响铃（Bell）
\'	单引号（Single quote）
\"	双引号（Double quote）

程序控制结构-选择结构

车万翔

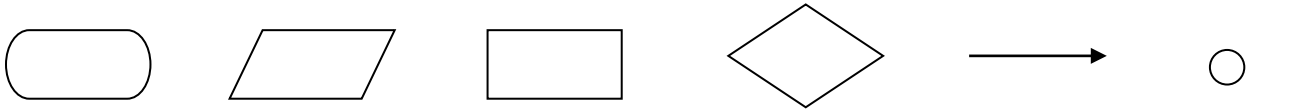
哈尔滨工业大学

程序流程图

程序流程图

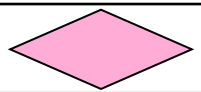
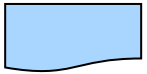
- 以简单的图形符号来表示问题的解决步骤，亦称为框图
- [重点] 流程图是问题求解的最基本、最重要的分析技术

常用流程图图形符号



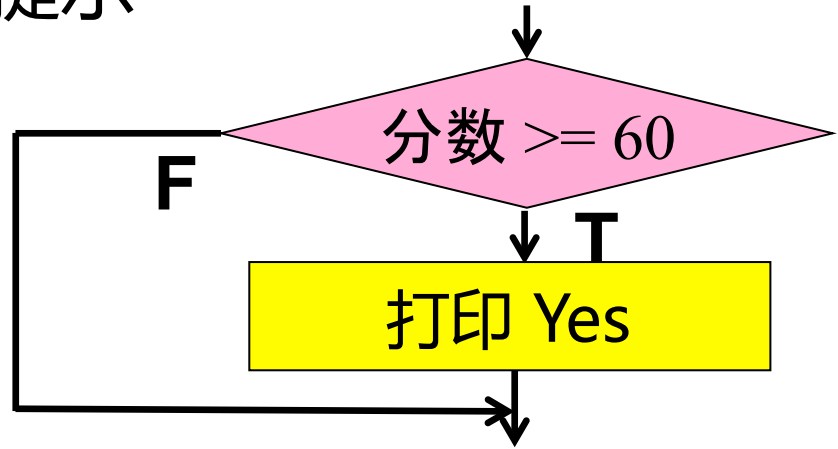
起止框　输入/输出框　处理框　判断框　流程线　连接符

程序流程图

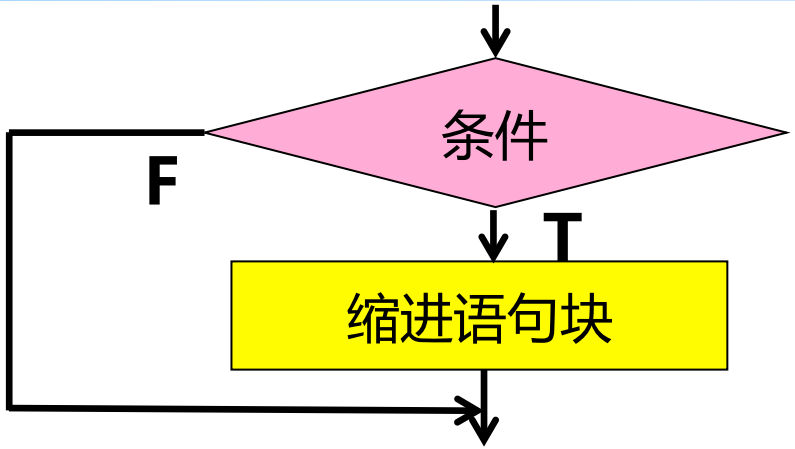
符 号	名称	含 义
	起止框	标准流程的开始与结束
	处理框	算法/程序要执行的处理操作
	判断框	判断条件是否成立
	文档框	以文件的方式输入/输出
	流程线	表示算法/程序执行的方向与顺序
	输入输出框	表示数据的输入/输出
	关联	同一流程图中从一个进程到另一个进程的交叉引用

程序流程图示例

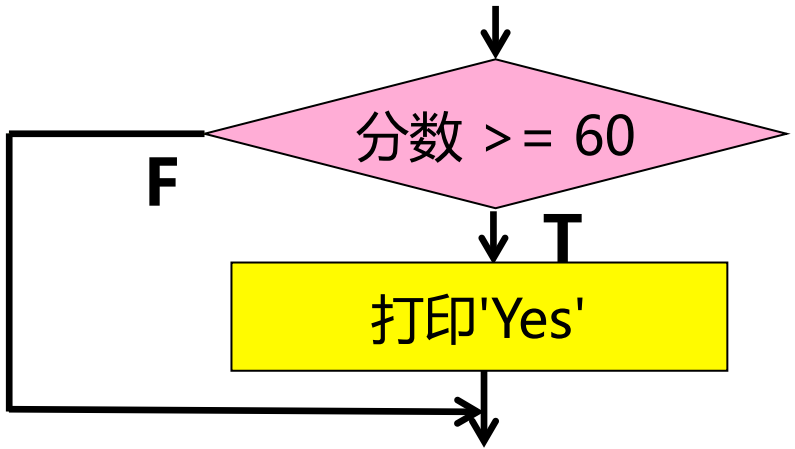
- 如果成绩合格，则打印相应提示
- 条件：合格 ⇔ 分数 >= 60
- 动作：打印 Yes



if 语句

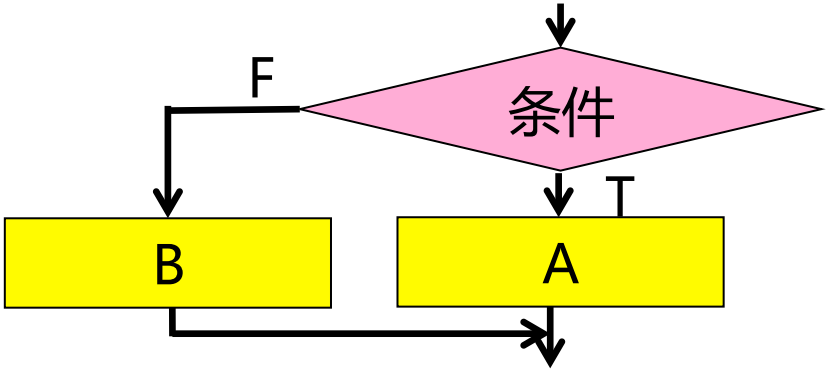



```
if 条件 :
    缩进语句块
其余语句
```

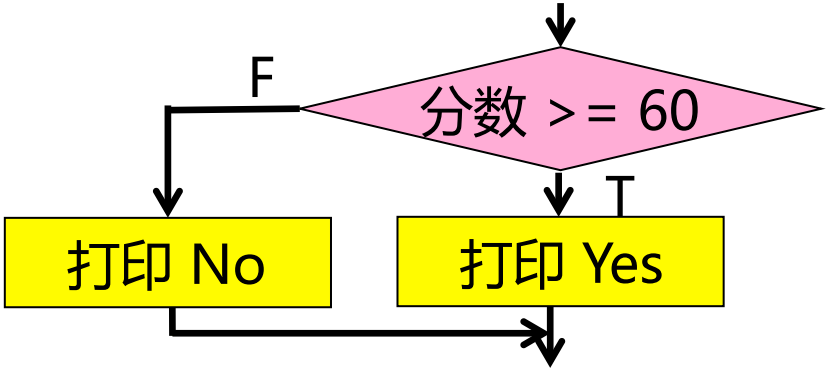



```
if score >=60 :
    print 'Yes'
```

if-else 语句



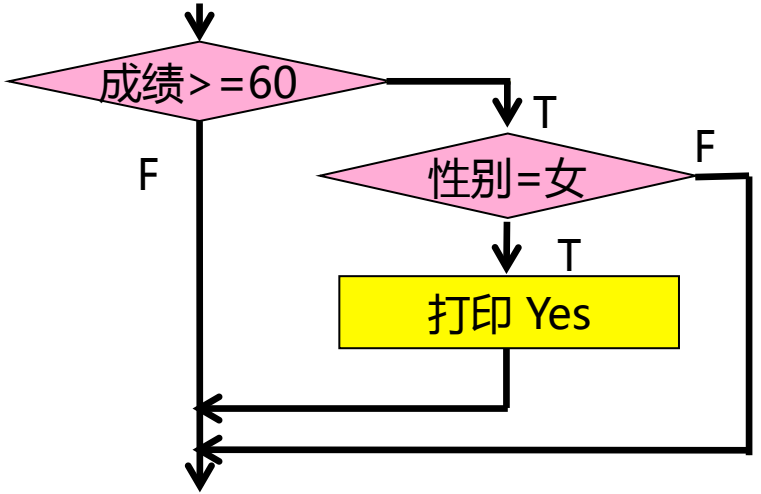

```
if 条件 :
    条件真缩进语句块
else:
    条件假缩进语句块
```




```
if score >=60 :
    print 'Yes'
else:
    print 'No'
```

if 语句-嵌套结构（Nested）

- 如果是成绩合格的女生，则打印提示
- 条件：成绩>=60 且 性别=女
- 动作：打印 Yes

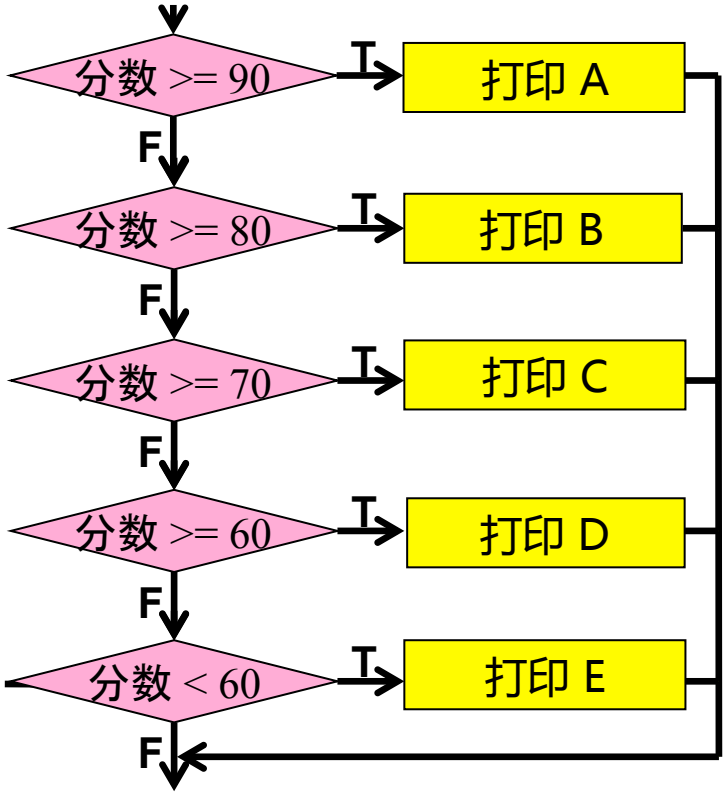



```
if score >=60 :
    if gender == '女' :
        print 'Yes'
```

```
if score >=60 and gender == '女' :
    print 'Yes'
```

多分支结构（Chained）

- 将考试分数转换为等级




```
7  score = 78
8
9  if score >= 90:
10     print 'A'
11 else:
12     if score >= 80:
13         print 'B'
14     else:
15         if score >= 70:
16             print 'C'
17         else:
18             if score >= 60:
19                 print 'D'
20             else:
21                 print 'E'
```

多分支结构（Chained）

```
score = 78

if score >= 90:
    print 'A'
else:
    if score >= 80:
        print 'B'
    else:
        if score >= 70:
            print 'C'
        else:
            if score >= 60:
                print 'D'
            else:
                print 'E'
```

elif

```
if score >= 90:
    print 'A'
elif score >= 80:
    print 'B'
elif score >= 70:
    print 'C'
elif score >= 60:
    print 'D'
else:
    print 'E'
```

if-elif-else语句：

- elif 相当于 else：if，但和第一个if条件并列
- if-elif-else 语句中有 else 条件时，else 条件放最后，否则SyntaxError

示例：求一元二次方程的解

- 一元二次方程 $ax^2 + bx + c = 0$
- 解为：

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

```
 8 import math
 9
10 a = float(raw_input('Input a: '))
11 b = float(raw_input('Input b: '))
12 c = float(raw_input('Input c: '))
13
14 root = math.sqrt(b ** 2 - 4 * a * c)
15 s1 = (-b + root) / (2 * a)
16 s2 = (-b - root) / (2 * a)
17
18 print 'The solutions are: ', s1, s2
19
```

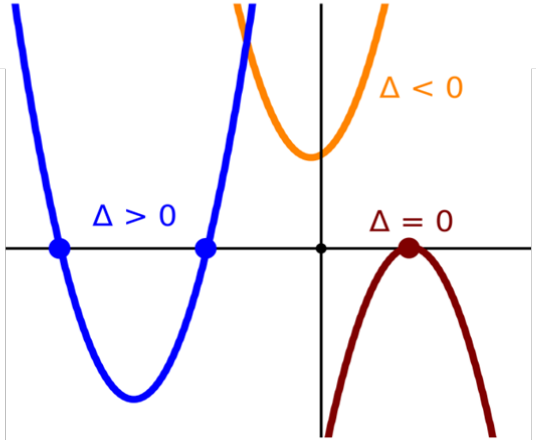
示例：求一元二次方程的解

一元二次方程$ax^2+bx+c=0$（$a\neq0$）的根与$\triangle=b^2-4ac$有如下关系：

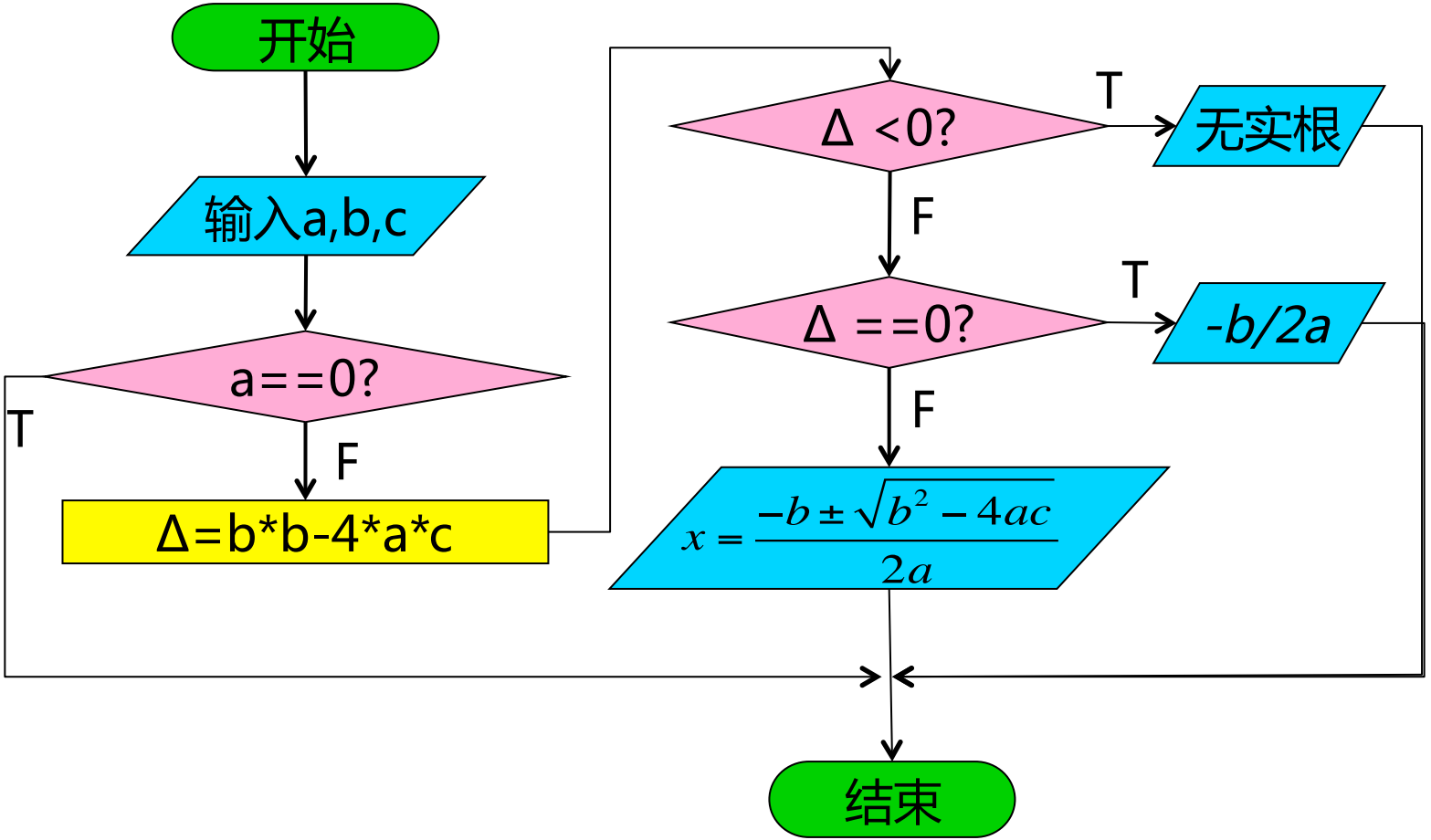
①当$\triangle>0$时，方程有两个不相等的两个实数根；

②当$\triangle=0$时，方程有两个相等的两个实数根；

③当$\triangle<0$时，方程无实数根.



示例：流程图


开始
输入a,b,c
a==0?
T
F
Δ=b*b-4*a*c
Δ <0?
T
无实根
F
Δ ==0?
T
-b/2a
F
$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$
结束

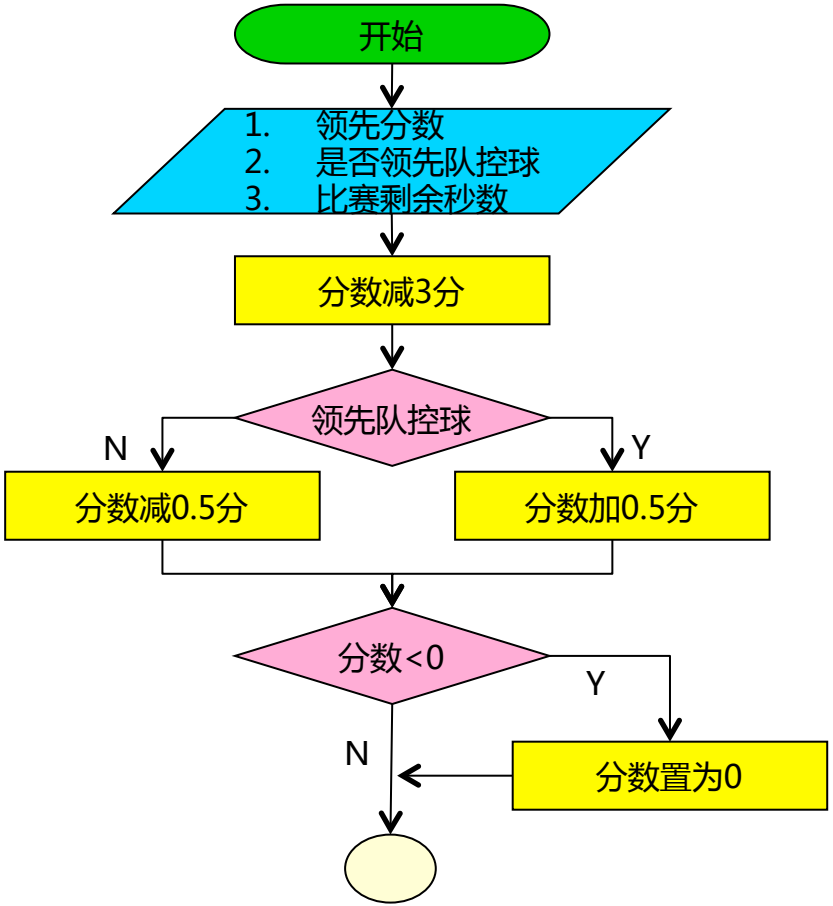
示例：源代码

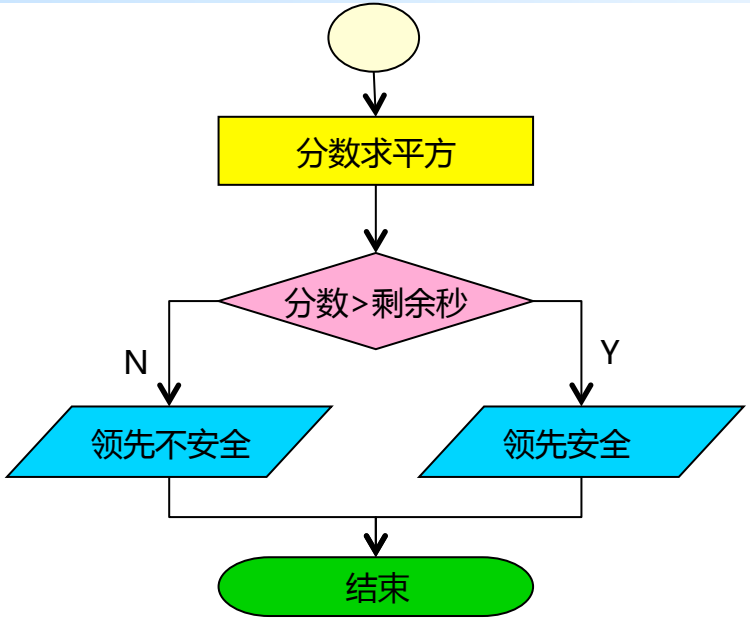

```
 8 import math
 9
10 a = float(raw_input('Input a: '))
11 b = float(raw_input('Input b: '))
12 c = float(raw_input('Input c: '))
13
14 if a == 0:
15     print 'The equation is linear, not quadratic'
16 else:
17     delta = b ** 2 - 4 * a * c
18     if delta < 0:
19         print 'No real roots!'
20     elif delta == 0:
21         print 'Only one root is ', -b / (2 * a)
22     else:
23         root = math.sqrt(delta)
24         s1 = (-b + root) / (2 * a)
25         s2 = (-b - root) / (2 * a)
26
27         print 'Two distinct solutions are: ', s1, s2
28
```

示例：篮球比赛领先多少才安全？

- 篮球比赛是高的分的比赛，领先优势可能很快被反超。作为观众，希望能在球赛即将结束时，就提早知道领先是否不可超越。体育作家Bill James发明了一种算法，用于判断领先是否“安全”
- 算法描述
  - 获取领先的分数
  - 减去三分
  - 如果目前是领先队控球，则加0.5；否则减0.5（数字小于0则变成0）
  - 计算平方后的结果
  - 如果得到的结果比当前比赛剩余时间的秒数大，则领先是“安全”的

示例：流程图


开始
1. 领先分数
2. 是否领先队控球
3. 比赛剩余秒数
分数减3分
领先队控球
N
Y
分数减0.5分
分数加0.5分
分数<0
Y
N
分数置为0


分数求平方
分数>剩余秒
N
Y
领先不安全
领先安全
结束

示例：代码


```
8  points = int(raw_input('Input the lead in points: '))
9  has_ball = raw_input('Does the lead team have the ball (Yes or No): ')
10 seconds = int(raw_input('Input the number of secondes remaining: '))
11
12 points -= 3
13
14 if has_ball == 'Yes':
15     points += 0.5
16 else:
17     points -= 0.5
18
19 if points < 0:
20     points = 0
21
22 points **= 2
23
24 if points > seconds:
25     print 'Lead is safe'
26 else:
27     print 'Lead is not safe'
28
29
```

程序控制结构-循环结构

车万翔

哈尔滨工业大学

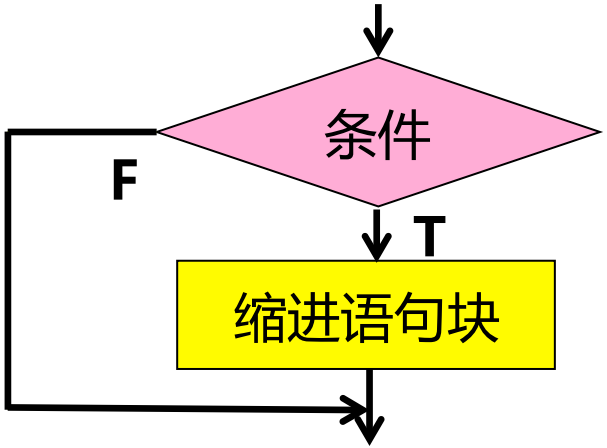
多次求解一元二次方程

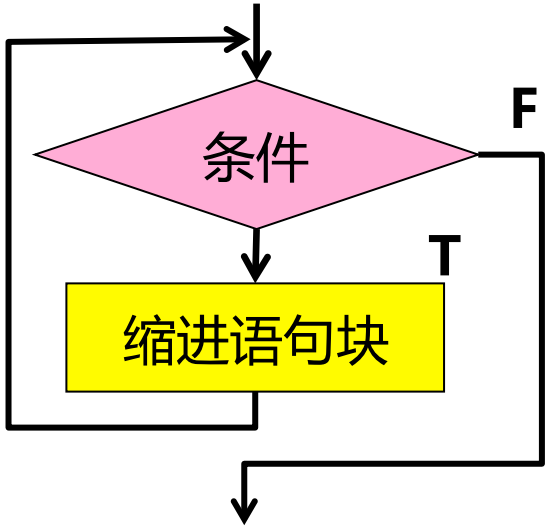
- a=0, b=1, c=1
- a=1, b=2, c=1
- a=1, b=3, c=1
- a=1, b=1, c=1
- a=-1, b=-1, c=-1

- 程序可以多次计算（输入字符 'q' 退出程序，输入其它字符则继续执行）。

循环结构

while 循环结构





while 循环继续条件：

　　缩进语句块（循环体）

其余语句

while 循环结构的分析策略

- 循环体外设定循环可执行的初始条件
- 书写需重复执行的代码（循环体）
- 设定循环条件并在循环体内设定条件改变语句

- 打印字符串 5 次

```
count = 0

while count < 5:
    print 'Programming is fun!'
    count += 1
```

循环执行过程

生成 count 变量，值为 0

```
 8 count = 0
 9
10 while count < 5:
11     print 'Programming is fun!'
12     count += 1
13
```

循环执行过程


```
 8 count = 0
 9
10 while count < 5:
11     print 'Programming is fun!'
12     count += 1
13
```

count 值为 0，小于 5

循环执行过程


```
8  count = 0
9
10 while count < 5:
11     print 'Programming is fun!'
12     count += 1
13
```

打印字符串

循环执行过程


```
8  count = 0
9
10 while count < 5:
11     print 'Programming is fun!'
12     count += 1
13
```

count 值加 1，结果为 1

循环执行过程

```
 8 count = 0
 9
10 while count < 5:
11     print 'Programming is fun!'
12     count += 1
13
```

count 值为 1，小于 5

循环执行过程


```
8  count = 0
9
10 while count < 5:
11     print 'Programming is fun!'
12     count += 1
13
```

打印字符串

循环执行过程


```
 8 count = 0
 9
10 while count < 5:
11     print 'Programming is fun!'
12     count += 1
13
```

count 值加 1，结果为 2

循环执行过程


```
 8 count = 0
 9
10 while count < 5:
11     print 'Programming is fun!'
12     count += 1
13
```

count 值为 2，小于 5

循环执行过程


```
8  count = 0
9
10 while count < 5:
11     print 'Programming is fun!'
12     count += 1
13
```

打印字符串

循环执行过程


```
 8 count = 0
 9
10 while count < 5:
11     print 'Programming is fun!'
12     count += 1
13
```

count 值加 1，结果为 3

循环执行过程


```
 8 count = 0
 9
10 while count < 5:
11     print 'Programming is fun!'
12     count += 1
13
```

count 值为 3，小于 5

循环执行过程


```
8  count = 0
9
10 while count < 5:
11     print 'Programming is fun!'
12     count += 1
13
```

打印字符串

循环执行过程


```
 8 count = 0
 9
10 while count < 5:
11     print 'Programming is fun!'
12     count += 1
13
```

count 值加 1，结果为 4

循环执行过程


```
 8 count = 0
 9
10 while count < 5:
11     print 'Programming is fun!'
12     count += 1
13
```

count 值为 4，小于 5

循环执行过程


```
8  count = 0
9
10 while count < 5:
11     print 'Programming is fun!'
12     count += 1
13
```

打印字符串

循环执行过程


```
8  count = 0
9
10 while count < 5:
11     print 'Programming is fun!'
12     count += 1
13
```

count 值加 1，结果为 5

循环执行过程


```
 8 count = 0
 9
10 while count < 5:
11     print 'Programming is fun!'
12     count += 1
13
```

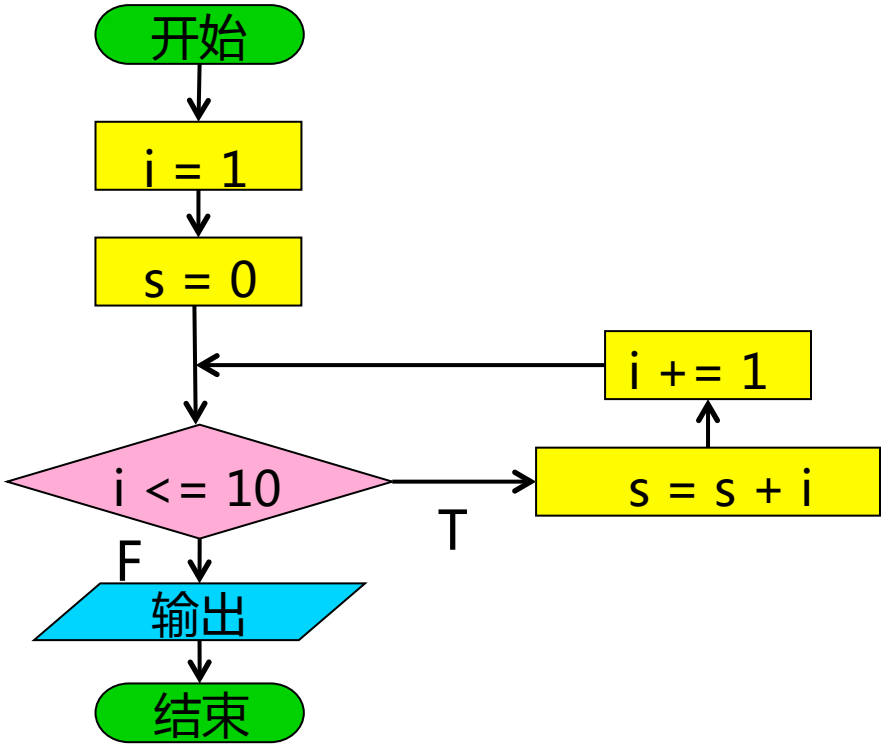
count 值为 5，不小于 5

循环执行过程


```
8  count = 0
9
10 while count < 5:
11     print 'Programming i
12     count += 1
13
```

继续执行 while 后面的语句

循环示例：1 + 2 + … + 10




```
8  s = 0
9  i = 1
10
11 while i < 10:
12     s += i
13     i += 1
14
15 print 'sum is ', s
16
```

错误代码示例


```
8  count = 0
9
10 while count < 10:
11     print count
12
```

```
8  count = 0
9
10 while count < 10:
11     print count
12     count -= 1
13
```

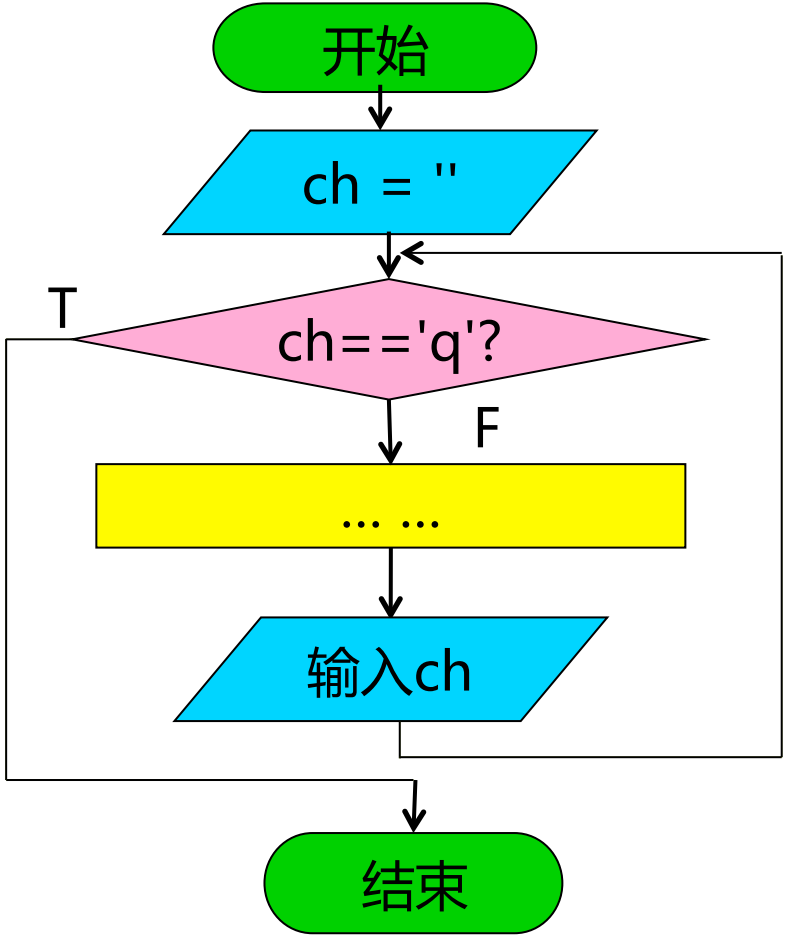
```
8  count = 0
9
10 while count < 10:
11     print count
12 count += 1
13
```

```
8  count = 0
9
10 while count < 10:
11     if count % 2 == 0:
12         print count
13
```

```
8  count = 0
9
10 while count < 10:
11     if count % 2 == 0:
12         print count
13         count += 1
```

```
8  count = 0
9
10 while count < 10:
11     if count % 2 == 0:
12         print count
13     count += 1
14
```

多次求解一元二次方程



代码


```
7  import math
8  ch = 'a'
9  while ch != 'q':
10     a = float(raw_input('Enter coefficient a: '))
11     b = float(raw_input('Enter coefficient b: '))
12     c = float(raw_input('Enter coefficient c: '))
13     if a==0:
14         print 'The equation is linear, not quadratic'
15     else:
16         delta = b ** 2 - 4 * a * c
17         if delta < 0:
18             print 'Without real roots'
19         elif delta == 0: # elif delta <1e-17:
20             print 'Only one root is ',(-b/2.0/a)
21         else:
22             root = math.sqrt(delta)
23             s1 = (-b + root) / (2 * a)
24             s2 = (-b - root) / (2 * a)
25             print 'Two distinct solutions are:', s1, s2
26     ch = raw_input('Please input \'q\' to end or any keys to continue\n')
27
```

break 和 continue 语句


```
 8 count = 0
 9
10 while count < 5:
11     if count > 2:
12         break
13     print 'Programming is fun!'
14     count += 1
15
```

break 结束当前循环体

```
 8 count = 0
 9
10 while count < 5:
11     count += 1
12     if count > 2:
13         continue
14     print 'Programming is fun!'
15
```

continue 结束当次循环

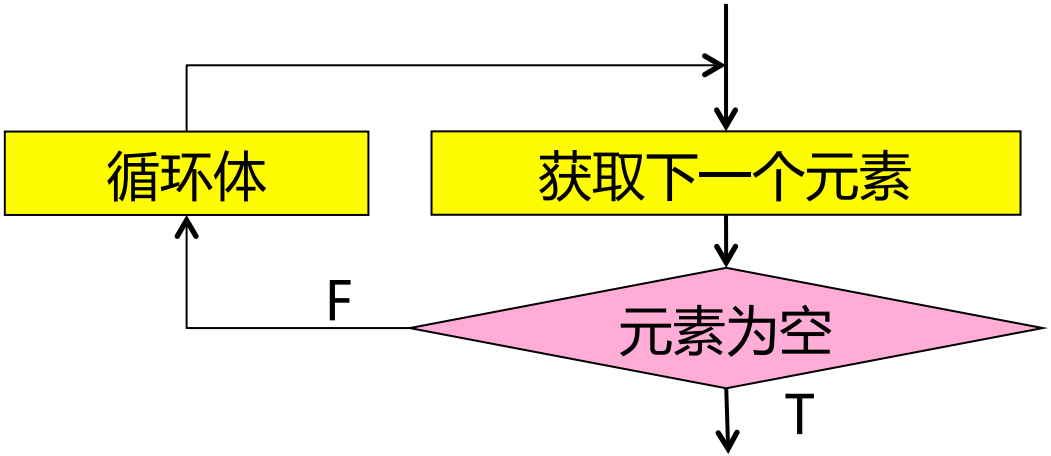
示例


```
7  import math
8  while True:
9      a = float(raw_input('Enter coefficient a: '))
10     b = float(raw_input('Enter coefficient b: '))
11     c = float(raw_input('Enter coefficient c: '))
12     if a==0:
13         print 'The equation is linear, not quadratic'
14     else:
15         delta = b ** 2 - 4 * a * c
16         if delta < 0:
17             print 'Without real roots'
18         elif delta == 0: # elif delta <1e-17:
19             print 'Only one root is ',(-b/2.0/a)
20         else:
21             root = math.sqrt(delta)
22             s1 = (-b + root) / (2 * a)
23             s2 = (-b - root) / (2 * a)
24             print 'Two distinct solutions are:', s1, s2
25     ch = raw_input('Please input \'q\' to end or any keys to continue\n')
26     if ch == 'q':
27         break
28
```

for 循环语句

```
for anElement in object:
    # 缩进语句块（循环体）
```

- 依次遍历对象（object）中的每个元素，并赋值给anElement，然后执行循环体内语句



for 循环语句示例

- 计算1+2+3+...+10的值

range 函数生成 0, 1, ..., 10 序列

```
 8 s = 0
 9
10 for i in range(11):
11     s += i
12
13 print 'sum is:', s
14
```

```
 8 s = 0
 9 i = 1
10
11 while i <= 10:
12     s += i
13     i += 1
14
15 print 'sum is:', s
16
```

range 函数

range

Definition : range(stop)

Type : Function of __builtin__ module

range(stop) -> list of integers range(start, stop[, step]) -> list of integers

Return a list containing an arithmetic progression of integers. range(i, j) returns [i, i+1, i+2, ..., j-1]; start (!) defaults to 0. When step is given, it specifies the increment (or decrement). For example, range(4) returns [0, 1, 2, 3]. The end point is omitted! These are exactly the valid indices for a list of 4 elements.

- range(2, 10) → [2, 3, 4, 5, 6, 7, 8, 9]
- range(2, 10, 3) → [2, 5, 8]
- range(10, 2, -1) → [10, 9, 8, 7, 6, 5, 4, 3]

求常数 e

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \ldots + \frac{1}{i!}$$

```
8  import math
9
10 e = 1
11
12 for i in range(1, 100):
13     e += 1.0 / math.factorial(i)
14
15 print 'e is', e
16
```

```
8  e = 1
9  factorial = 1
10
11 for i in range(1, 100):
12     factorial *= i
13     e += 1.0 / factorial
14
15 print 'e is', e
16
```

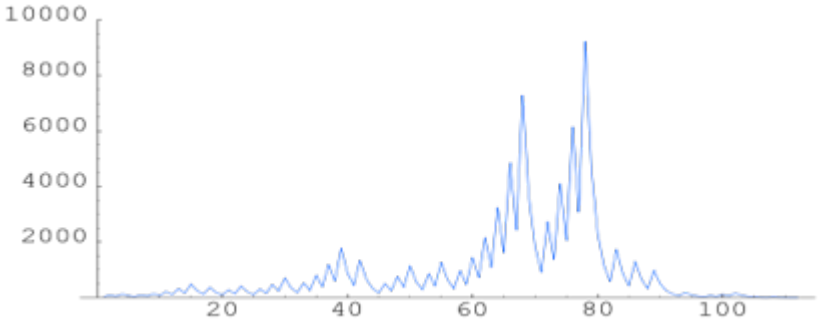
求常数 π

$$\pi = 4\left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \ldots + \frac{(-1)^{i+1}}{2i-1}\right)$$

```
 8 pi = 0
 9 sign = 1
10 divisor = 1
11
12 for i in range(1, 1000000):
13     pi += 4.0 * sign / divisor
14     sign *= -1
15     divisor += 2
16
17 print 'pi is', pi
18
```

冰雹猜想（序列）

- 考拉兹猜想（英语：Collatz conjecture），又称奇偶归一猜想，3n + 1 猜想、冰雹猜想、角谷猜想、哈塞猜想、乌拉姆猜想或叙拉古猜想
  - 对于每一个正整数，如果它是奇数，则对它乘 3 再加 1，如果它是偶数，则对它除以 2，如此循环，最终都能够得到 1
  - 如 n = 6，得出序列6, 3, 10, 5, 16, 8, 4, 2, 1



n = 27时的序列分布

嵌套循环

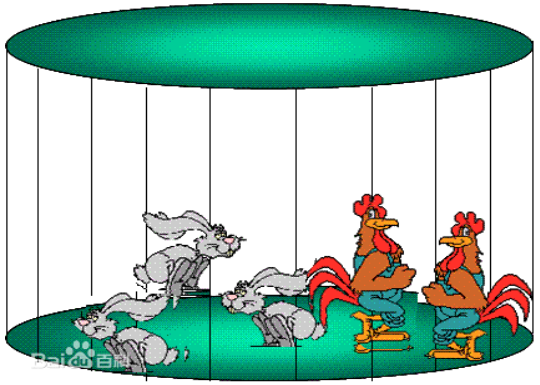
打印乘法表

```
1  2  3  4  5  6  7  8  9
-  -  -  -  -  -  -  -  -
1  2  3  4  5  6  7  8  9
2  4  6  8  10 12 14 16 18
3  6  9  12 15 18 21 24 27
4  8  12 16 20 24 28 32 36
5  10 15 20 25 30 35 40 45
6  12 18 24 30 36 42 48 54
7  14 21 28 35 42 49 56 63
8  16 24 32 40 48 56 64 72
9  18 27 36 45 54 63 72 81
```

```
8 for i in range(1, 10):
9     for j in range(1, 10):
10        print format(i * j, '3'),
11    print
12
```

鸡兔同笼问题

- 鸡兔同笼是中国古代的数学名题之一。大约在1500年前，《孙子算经》中就记载了这个有趣的问题。书中是这样叙述的：“今有雉兔同笼，上有三十五头，下有九十四足，问雉兔各几何？”



- 穷举法

```
for chickens in range(35 + 1):
    for rabbits in range(35 + 1):
        if 2 * chickens + 4 * rabbits == 94 and chickens + rabbits == 35:
            print 'The number of chickens is:', chickens
            print 'The number of rabbits is:', rabbits
            print
```

while vs. for 循环

- **while 循环更通用**
  - 任何 for 循环写的程序都能用 while 循环实现
- **适用场景**
  - for 循环
    - 已知循环的范围（range），即起止值和步长
  - while 循环
    - 其它情况，如：不确定循环何时终止

程序控制结构-练习

车万翔

哈尔滨工业大学

二分法求平方根

- 基本思想
  - 猜测一个平方根（x/2）
  - 如果猜小了，则正确的平方根在猜测数字和原数字之间
  - 如果猜大了，则正确的平方根在0和猜测数字之间
- 算法描述
  - Input: x
  - Output: √x

1. low = 0, high = x
2. guess = (low + high) / 2
3. 如果 $guess^2$ == x，则输出 guess，程序结束
4. 如果 $guess^2$ < x，则 low = guess; 继续执行步骤2
5. 如果 $guess^2$ > x，则 high = guess; 继续执行步骤2

二分法求平方根


```
 8 x = float(raw_input('Enter the number'))
 9 low = 0
10 high = x
11 guess = (low + high)/2
12
13 while abs(guess ** 2 - x) > 1e-5:
14     if guess ** 2 < x:
15         low = guess
16     else:
17         high = guess
18     guess = (low + high)/2
19
20 print 'The root of x is:', guess
21
```

- **问题**
  - 如果输入的 x < 0？
  - 如果输入的 x < 1？

素数（Prime Number）

- 素数（质数）
  - 一个大于1的自然数，除了1和它本身外，不能被其他自然数整除；否则称为合数

```
8  num = int(raw_input('En                ']))
9
10 for i in range(2, num):
11     if num % i == 0:
12         print 'The number is not a prime'
13         break
14 else:
15     print 'The number is a prime'
```

能更快么？

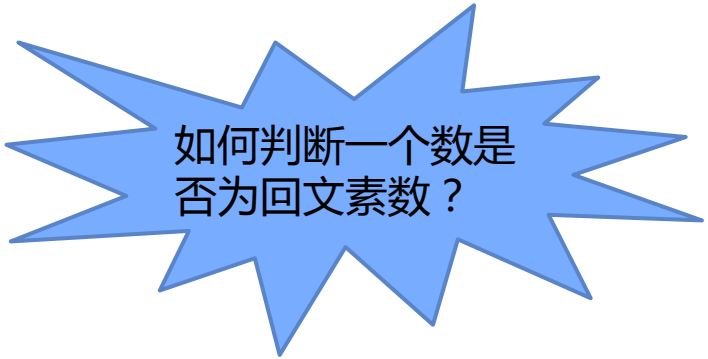
前 50 个素数

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	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

```
7  import math
8
9  count = 0
10 num = 2
11
12 while count < 50:
13     for i in range(2, int(math.sqrt(num)) + 1):
14         if num % i == 0:
15             break
16     else:
17         print num,
18         count += 1
19     num += 1
```

回文数

- 一个正数如果顺着和反过来都是一样的（如13431，反过来也是13431），就称为回文数
- 判断一个数 num 是否为回文数
- 算法
  - 求 num 的逆序 num'
  - 如果 num == num'，则 num 为回文数
  - 否则 num 非回文数




```
num = int(raw_input('Enter the number: '))
num_temp = num
num_prime = 0

while num_temp != 0:
    num_prime = num_prime * 10 + num_temp % 10
    num_temp /= 10

if num == num_prime:
    print 'The number', num, 'is a palindrome.'
else:
    print 'The number', num, 'is not a palindrome.'
```

函数

车万翔

哈尔滨工业大学

如何判断回文素数？

- 如何判断一个回文素数，即它既是回文数，又是素数
  - 如：2, 3, 5, 7, 11, 101, 131, 151, …
- 这两个子问题我们都曾学过
  - ① 判断回文数

```
num = 151
num_p = 0
num_t = num

while num_t != 0:
    num_p = num_p * 10 + num_t % 10
    num_t = num_t / 10

if num == num_p:
    print 'Is a palin'
else:
    print 'NO'
```

② 判断素数

```
num = 151

for i in range(2, num):
    if num % i == 0:
        break
else:
    print 'Is a prime!'
```

- 如何将这两个程序合并在一起？

解决办法--函数


```
num = 151
if is_palin(num) and is_prime(num):
    print 'Yes'
else:
    print 'No'
```

函数是什么？

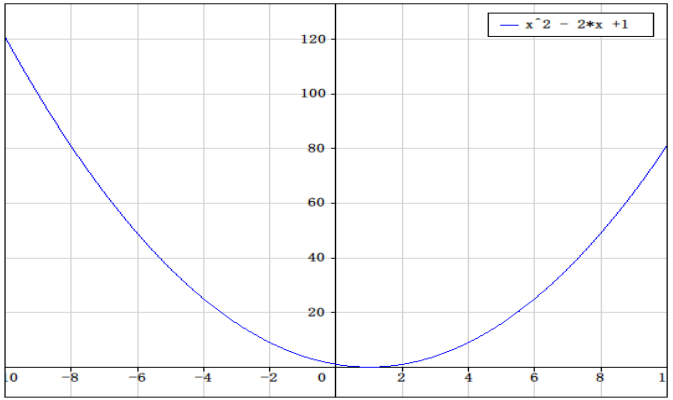
函数

- 完成特定功能的一个语句组，这组语句可以作为一个单位使用，并且给它取一个名字
- 通过函数名执行

函数名　　参数

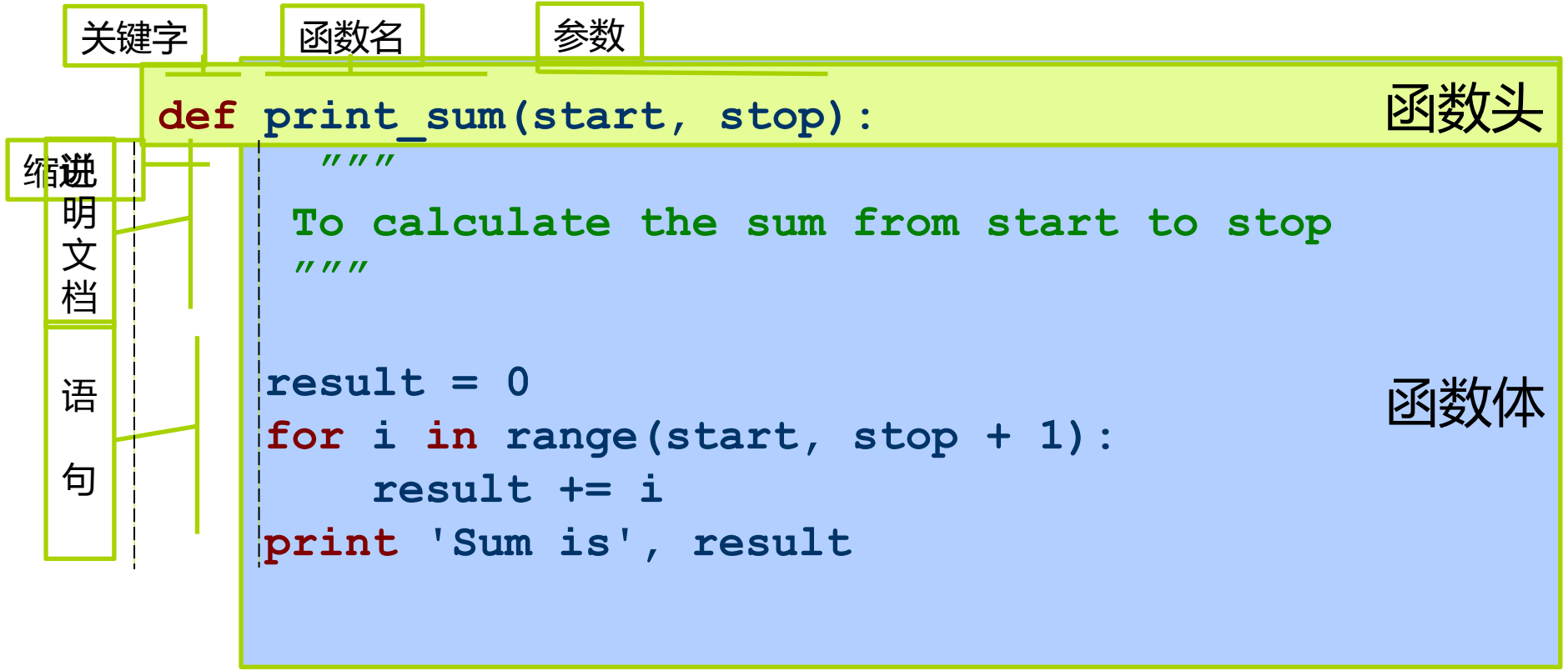
function(x) =

$$x^2 - 2x + 1$$



如：abs(x) #求x的绝对值

定义函数


关键字
函数名
参数
def print_sum(start, stop):
函数头
缩进
说明文档
"""
To calculate the sum from start to stop
"""
语句
result = 0
for i in range(start, stop + 1):
result += i
print 'Sum is', result
函数体

定义和调用函数

- 定义函数

```
def print_sum(start, stop):
    result = 0
    for i in range(start, stop + 1):
        result += i
    print 'Sum is', result
```

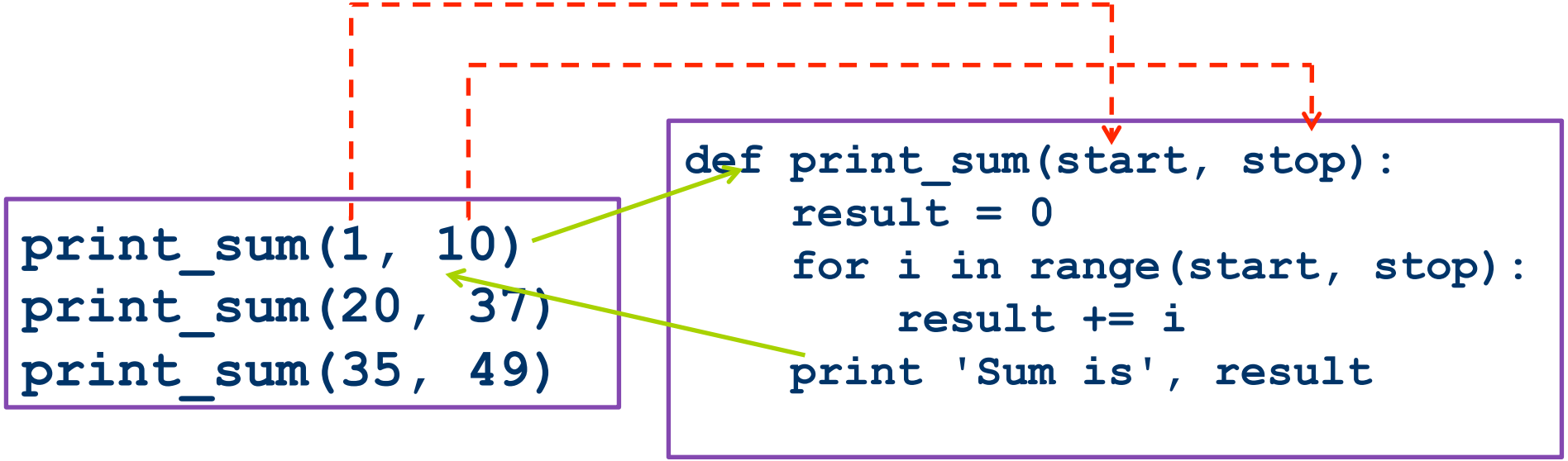
形式参数（形参，parameter）

- 调用函数

```
print_sum(1, 10)
```

实际参数（实参，argument）

函数调用的过程


print_sum(1, 10)
print_sum(20, 37)
print_sum(35, 49)
def print_sum(start, stop):
result = 0
for i in range(start, stop):
result += i
print 'Sum is', result

函数参数 – 缺省参数


```
def defaultParameters(arg1, arg2=2, arg3=3):
    print 'arg1=', arg1
    print 'arg2=', arg2
    print 'arg3=', arg3
```

```
defaultParameters(10)
```

```
defaultParameters(10, 10)
```

```
defaultParameters(10, 10, 10)
```

有返回值的函数


```
def sum(start, stop):
    result = 0
    for i in range(start, stop + 1):
        result += i

    return result
```

- 函数调用完成后，返回数据
- return语句终止当前函数的执行
- return后的语句将被忽略

函数 – 变量作用域


```
globalVar = 1

def f1():
    localVar = 2
    print globalVar
    print localVar

f1()
print globalVar
print localVar
```

变量作用域

- **局部变量**
  - 只能在程序的特定部分使用的变量
  - 函数内部
- **全局变量**
  - 为整个程序所使用的变量
  - 所有函数均可以使用

变量作用域


```
x = 1

def f1():
    x = 2
    print x

f1()
print x
```

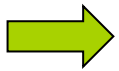
```
x = 1

def increase():
    global x
    x = x + 1
    print x

increase()
print x
```

函数实现回文素数


```
num = 151
if is_palin(num) and is_prime(num):
    print 'Yes'
else:
    print 'No'
```

函数实现回文素数

```python
num = 152

is_palin = False
is_prime = False

num_p = 0
num_t = num

while num != 0:
    num_p = num_p * 10 + num % 10
    num = num / 10

if num_t == num_p:
    is_palin = True


for i in range(2, num):
    if num % i == 0:
        break
else:
    is_prime = True

if is_palin and is_prime:
    print 'Yes'
else:
    print 'No'
```



```python
num = 151

def is_palin(num):
    num_p = 0
    num_t = num

    while num != 0:
        num_p = num_p * 10 + num % 10
        num = num / 10

    if num_t == num_p:
        return True
    else:
        return False


def is_prime(num):
    for i in range(2, num):
        if num % i == 0:
            return False
    return True

if is_palin(num) and is_prime(num):
    print 'Yes'
else:
    print 'No'
```

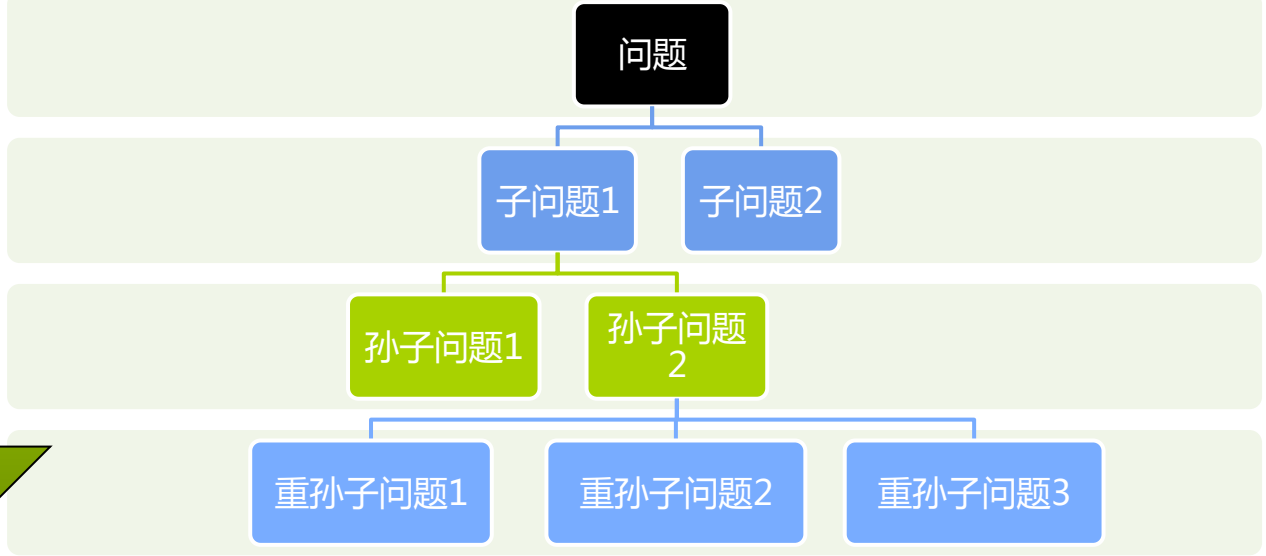
函数的优点

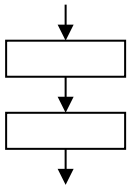
- **代码可重用**
  - 提高开发效率
  - 减少重复编码
- **代码更简洁**
  - 函数功能相对独立，功能单一
  - 结构清晰，可读性好
- **编程更容易把握**
  - 复杂程序分解成较小部件
- **封装与信息隐藏**

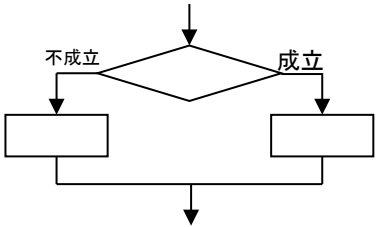
函数 - 结构化程序设计方法

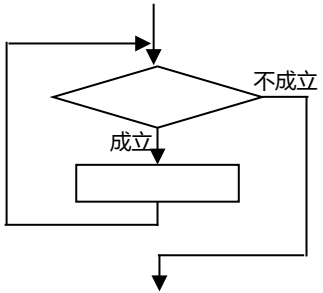
自顶向下

自底向上







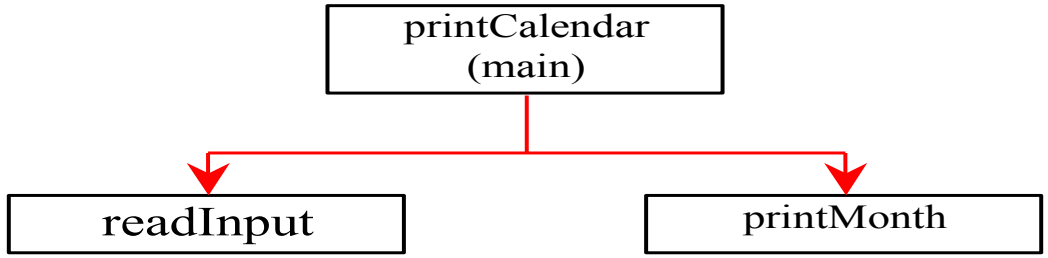


打印给定年、月的日历


```
                December   2033
----------------------------------------
  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
```

Design Diagram

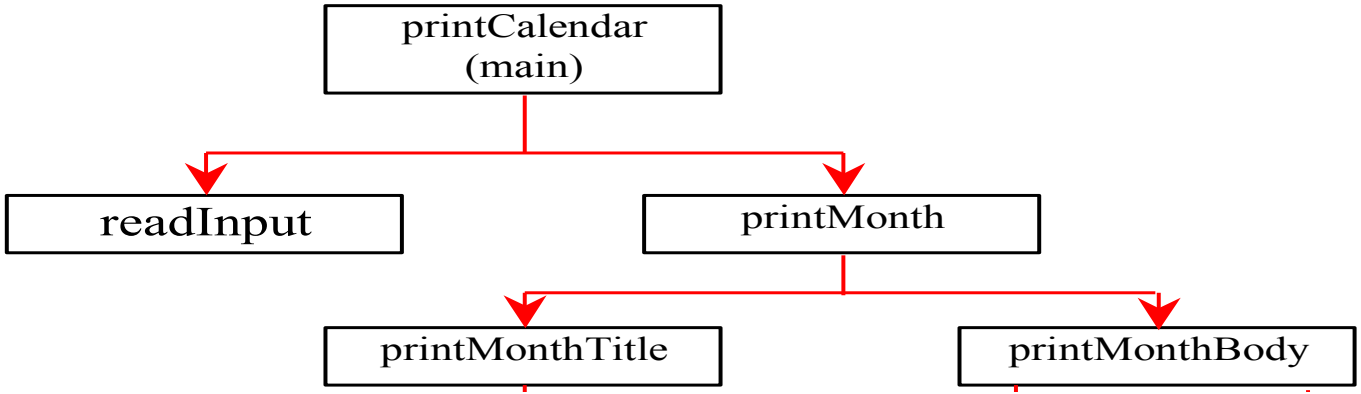
Design Diagram



Year: 2033
Month: 12

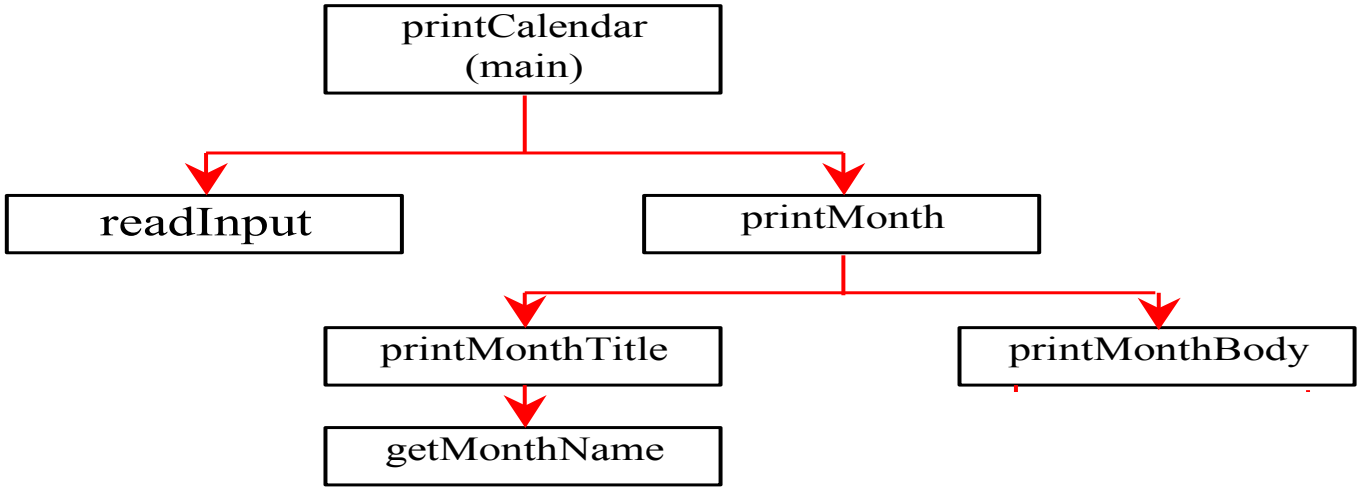
```
                December   2033
-----------------------------------
 Sun   Mon   Tue   Wed   Thu   Fri   Sat
                           1     2     3
   4     5     6     7     8     9    10
  11    12    13    14    15    16    17
  18    19    20    21    22    23    24
  25    26    27    28    29    30    31
```

Design Diagram



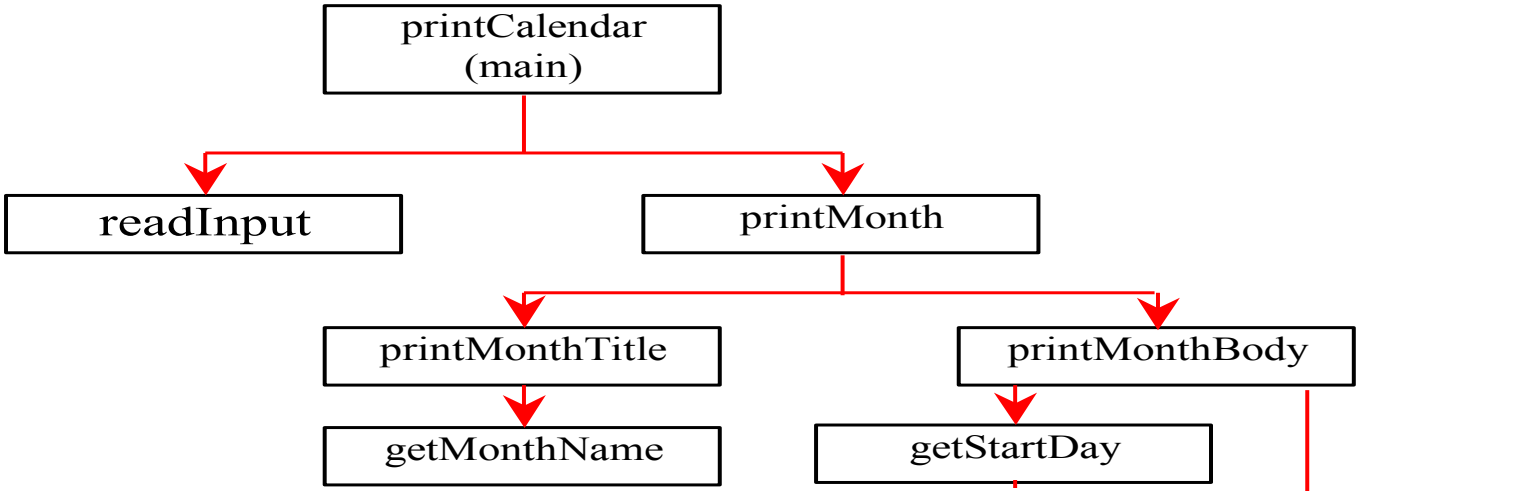

```
                December   2033
------------------------------------
  Sun   Mon   Tue   Wed   Thu   Fri   Sat
                            1     2     3
    4     5     6     7     8     9    10
   11    12    13    14    15    16    17
   18    19    20    21    22    23    24
   25    26    27    28    29    30    31
```

Design Diagram



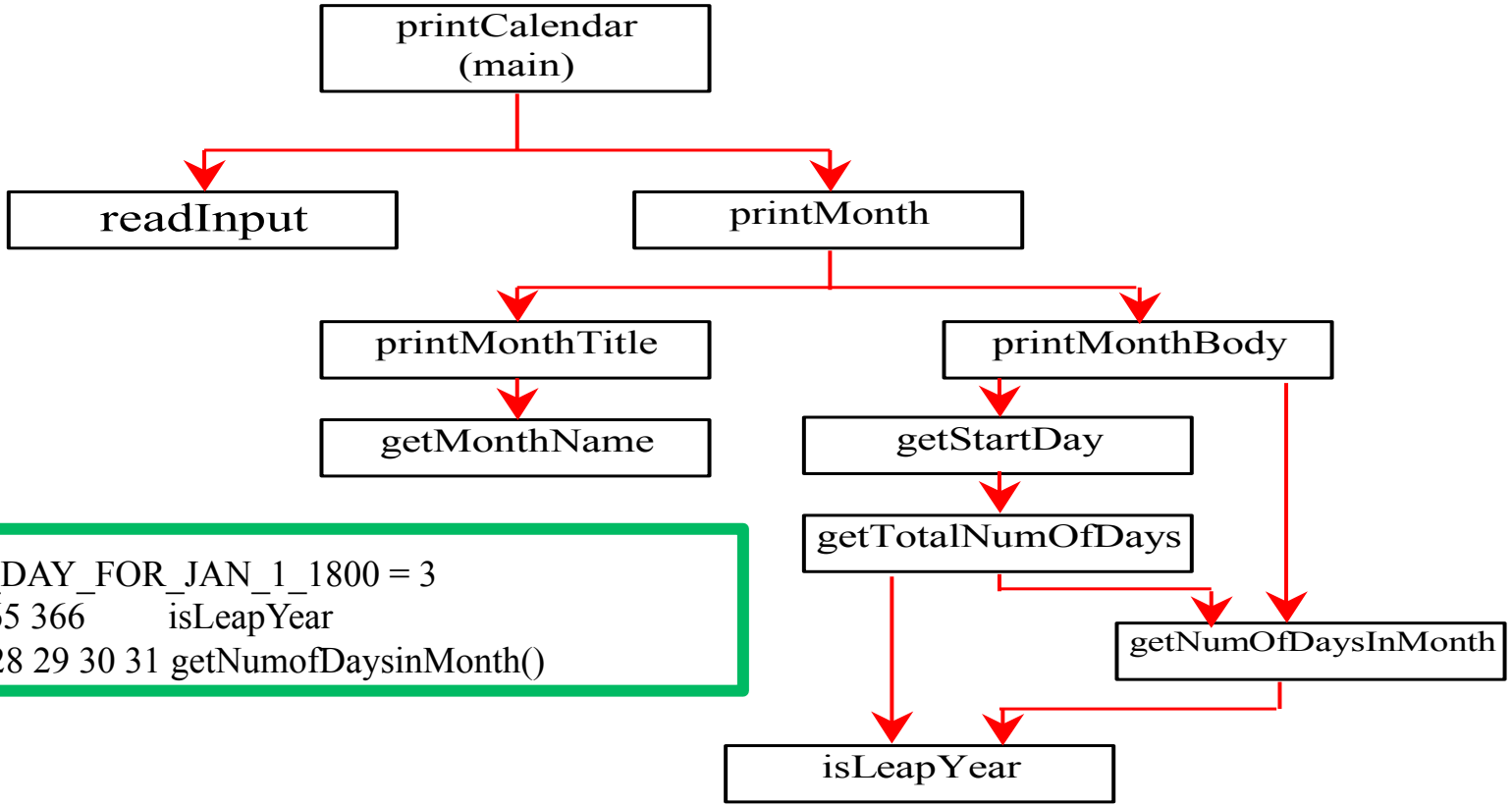

```
            December   2033
------------------------------------
  Sun   Mon   Tue   Wed   Thu   Fri   Sat
                            1     2     3
    4     5     6     7     8     9    10
   11    12    13    14    15    16    17
   18    19    20    21    22    23    24
   25    26    27    28    29    30    31
```

Design Diagram



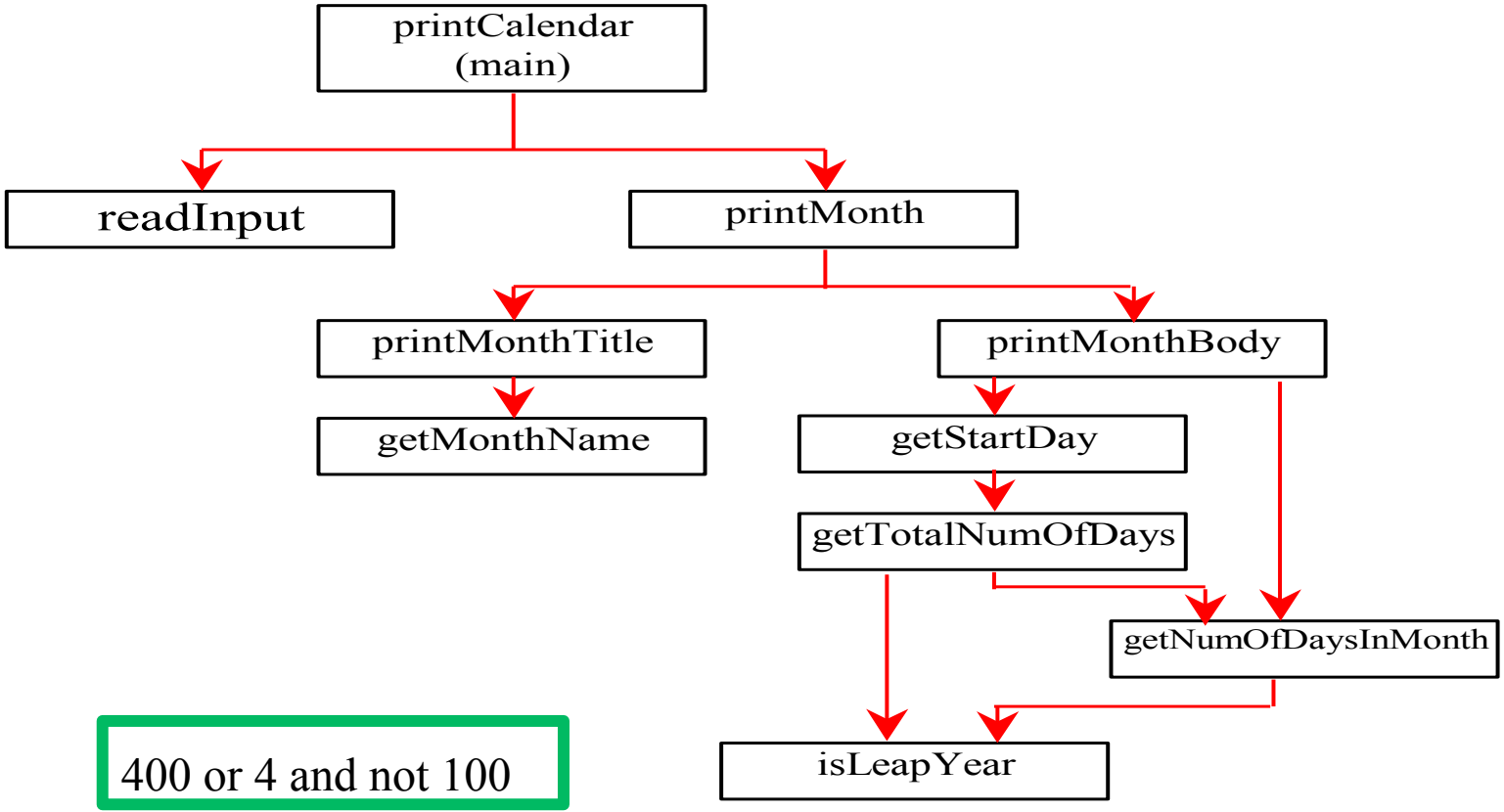

```
           December    2033
-----------------------------------
  Sun   Mon   Tue   Wed   Thu   Fri   Sat
                            1     2     3
    4     5     6     7     8     9    10
   11    12    13    14    15    16    17
   18    19    20    21    22    23    24
   25    26    27    28    29    30    31
```

Design Diagram



START_DAY_FOR_JAN_1_1800 = 3
Year *365 366 isLeapYear
Month*28 29 30 31 getNumofDaysinMonth()

Design Diagram



递归函数

车万翔

哈尔滨工业大学

两个和尚的故事

“从前有座山,山里有座庙,庙里有个老和尚给小和尚讲故事,讲什么呢?” “从前有座山,山里有座庙,庙里有个老和尚给小和尚讲故事,讲什么呢?”

“从前有座山,山里有座庙,庙里有个老和尚给小和尚讲故事,讲什么呢?” ……

递归的定义

递归：程序调用自身

形式：在函数定义有直接或间接调用自身

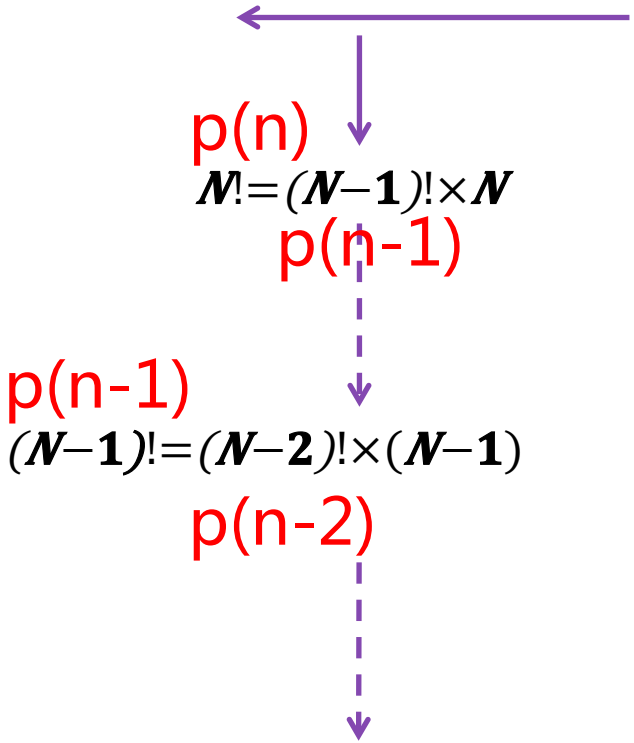
阶乘

- 阶乘：

```
def p(n):
    x = 1
    i = 1
    while i <= n:
        x = x * i
        i= i + 1
    return x

n = int(raw_input("请输入一个整数:"))
print n, "!的值为", p(n)
```

阶乘

- 阶乘：

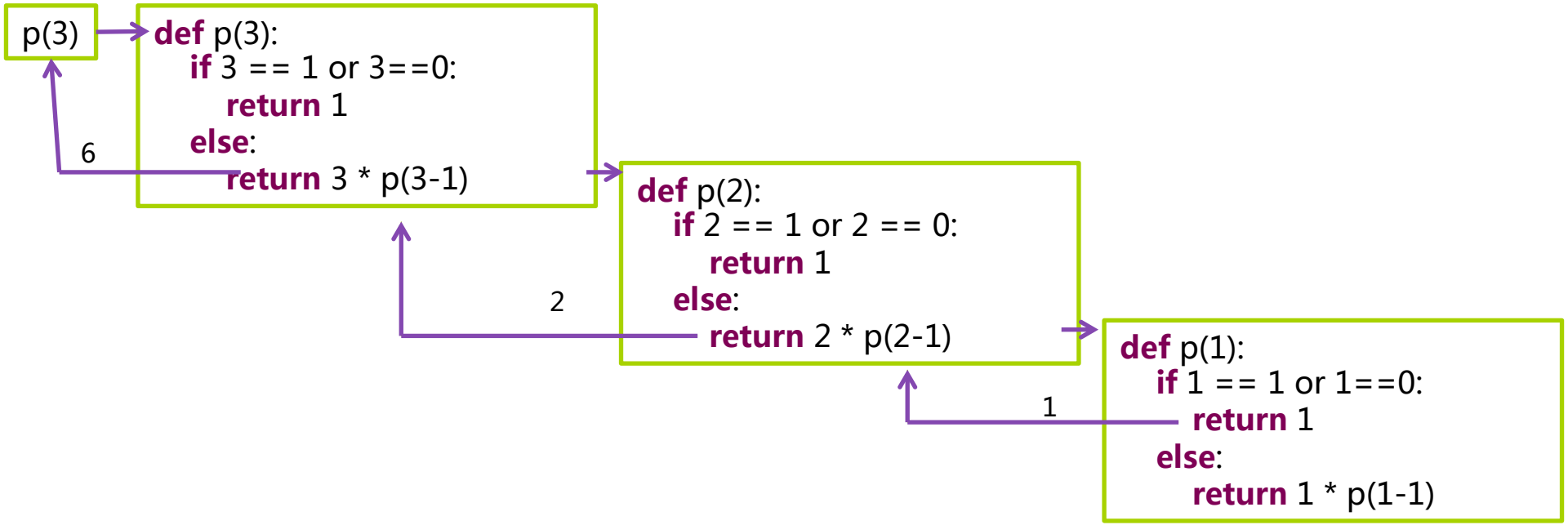


阶乘

- 阶乘：

```
def p(n):
    if n == 1 or n == 0:
        return 1
    else:
        return n * p(n-1)

n = int(raw_input("请输入一个整数:"))
print n, " !的值为：", p(n)
```

阶乘


p(3)
def p(3):
if 3 == 1 or 3==0:
return 1
else:
return 3 * p(3-1)
6
def p(2):
if 2 == 1 or 2 == 0:
return 1
else:
return 2 * p(2-1)
2
def p(1):
if 1 == 1 or 1==0:
return 1
else:
return 1 * p(1-1)
1

阶乘

```
def p(n):
    if n == 1 or n == 0:
        return 1
    else:
        return n * p(n-1)
```

初始条件

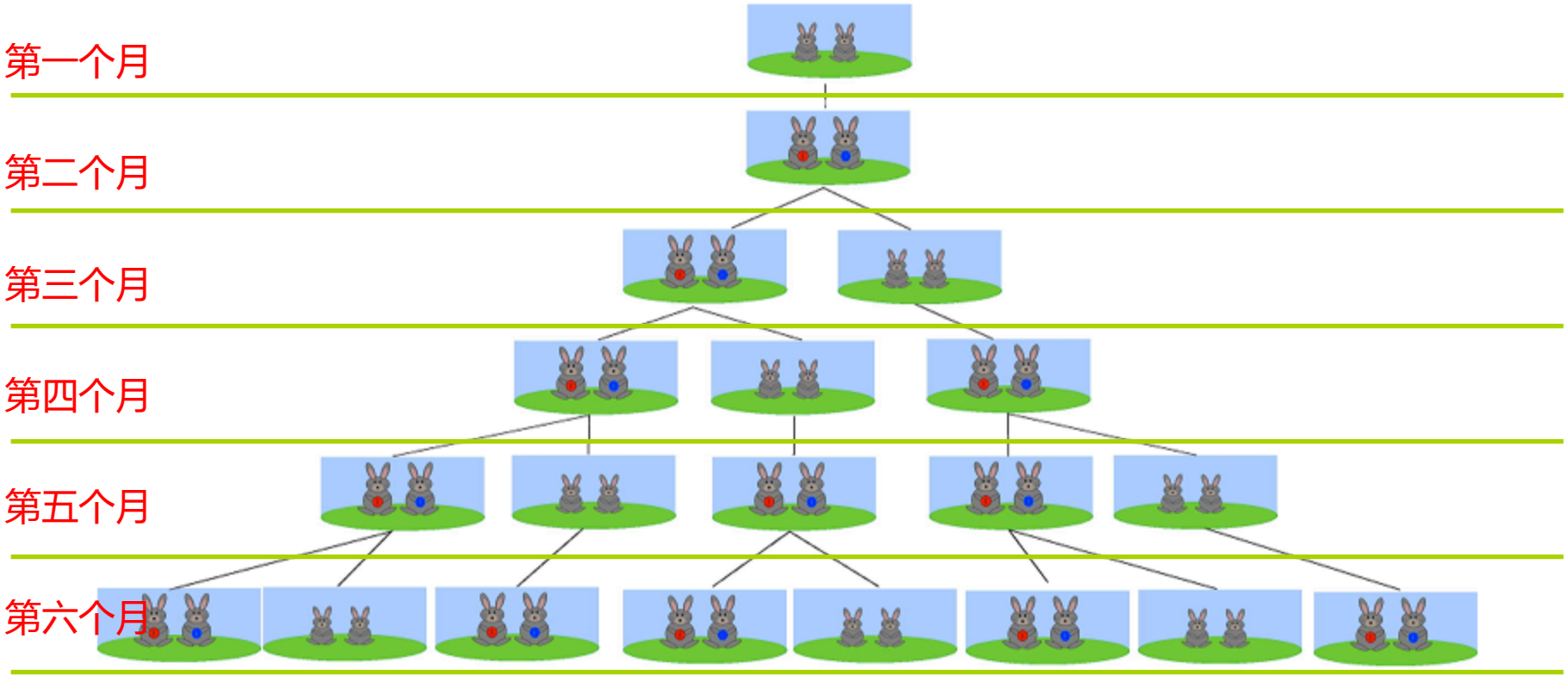
递归

掐头去尾留中间

递归解决问题的思想

- if 问题足够简单：
  - 直接解决问题
  - 返回解
- else:
  - 将问题分解为与原问题同构的一个或多个更小的问题
  - 逐个解决这些更小的问题
  - 将结果组合为，获得最终的解
  - 返回解

兔子数列

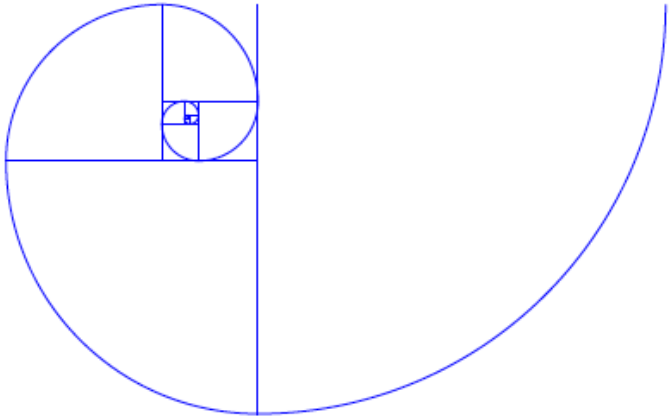


兔子数列

斐波那契数列

- 是这样一个数列：1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89……

$$f(n) = \begin{cases} 1 & if\ n = 1 \\ 1 & if\ n = 2 \\ f(n-1) + f(n-2) & if\ n > 2 \end{cases}$$



斐波那契数列

- **斐波那契数列：1, 1, 2, 3, 5, 8, 13, 21……**

$$f(n) = \begin{cases} 1 & if\ n = 1 \\ 1 & if\ n = 2 \\ f(n-1) + f(n-2) & if\ n > 2 \end{cases}$$

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

初始条件

递归

斐波那契数列

fib(4)

```
def fib(4):
    if 4 == 1 or 4 == 2:
        return 1
    else:
        return fib(4-1) + fib(4-2)
```

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

```
def fib(3):
    if 3 == 1 or  3 == 2:
        return 1
    else:
        return fib(3-1) + fib(3-2)
```

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

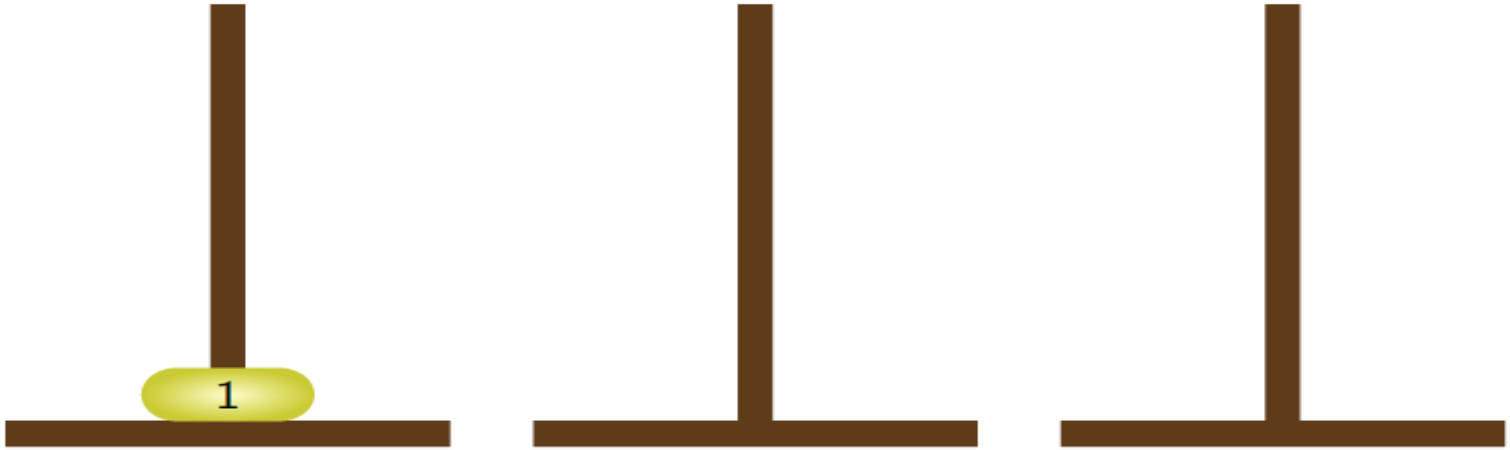
递归 - 汉诺塔

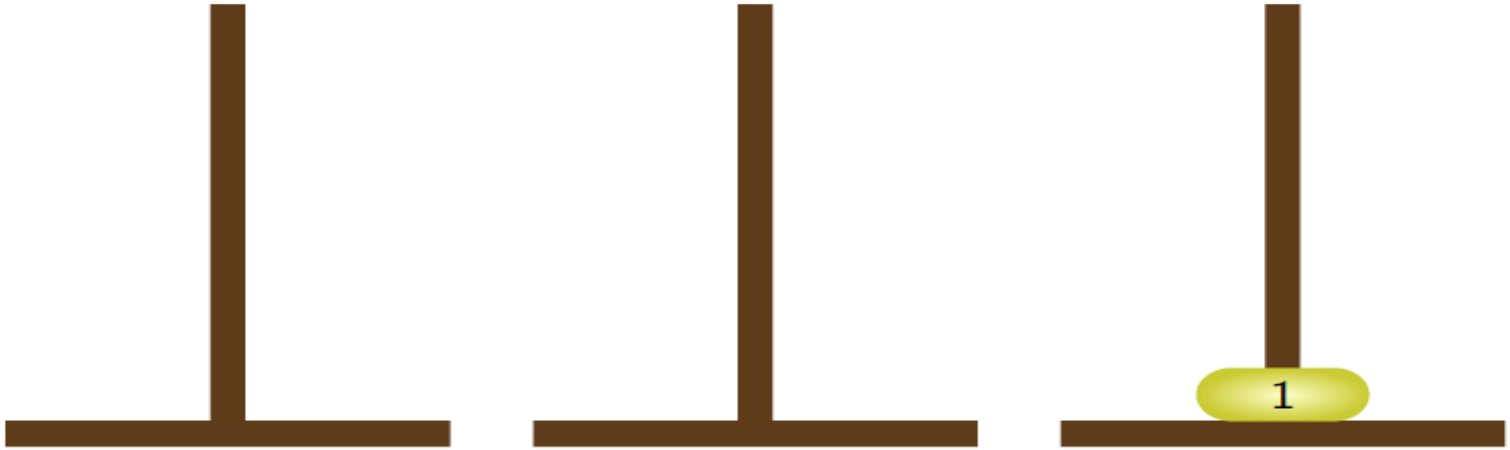
- **在印度，有这么一个古老的传说:**

开天辟地的神勃拉玛（和中国的盘古差不多的神）在一个庙里留下了三根金刚石的棒，第一根上面套着64个圆的金片，最大的一个在底下，其余一个比一个小，依次叠上去，庙里的众僧不倦地把它们一个个地从这根棒搬到另一根棒上，规定可利用中间的一根棒作为帮助，但每次只能搬一个，而且大的不能放在小的上面

移动圆片的次数:

18446744073709551615

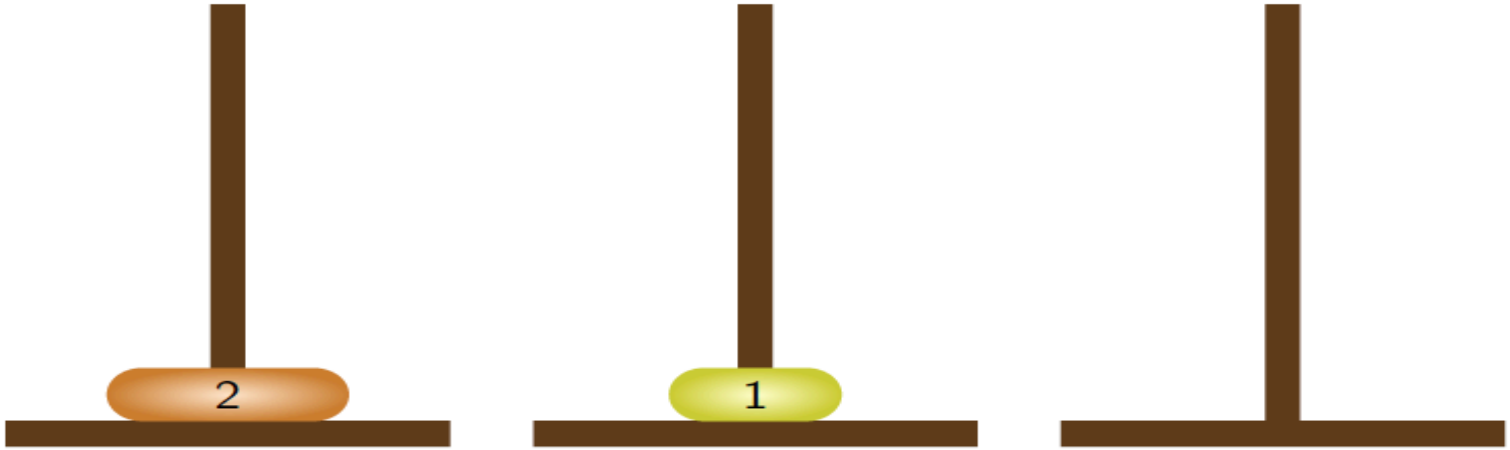

1



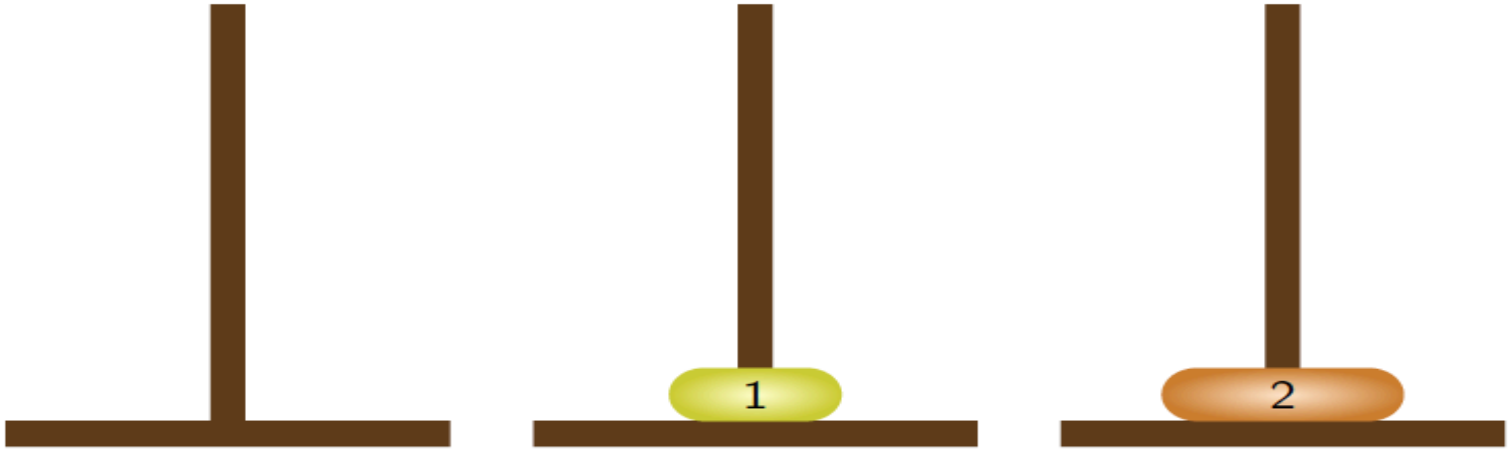
Moved disc from pole 1 to pole 3.


ok
1

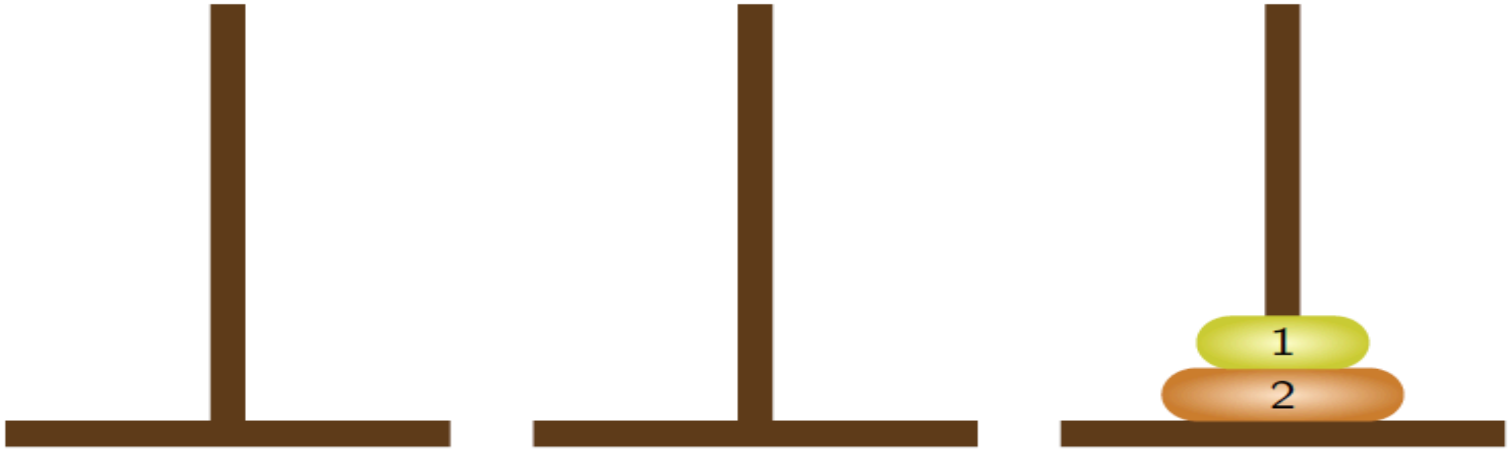

1
2



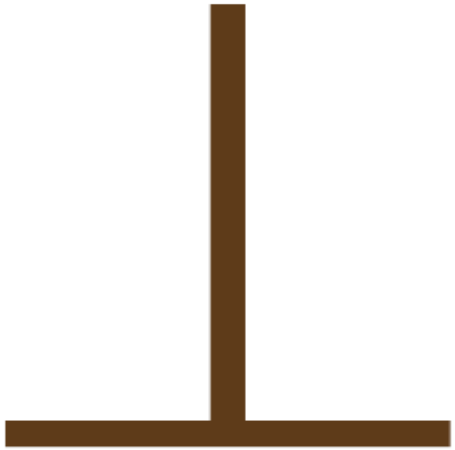
Moved disc from pole 1 to pole 2.



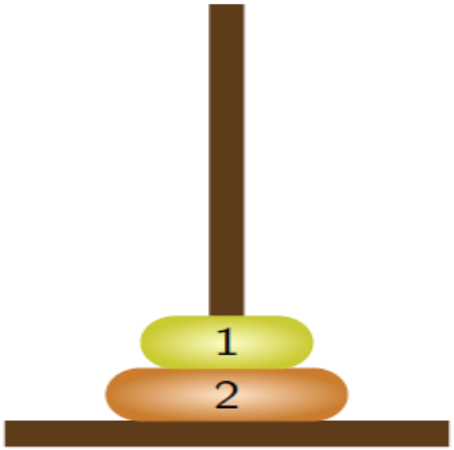
Moved disc from pole 1 to pole 3.

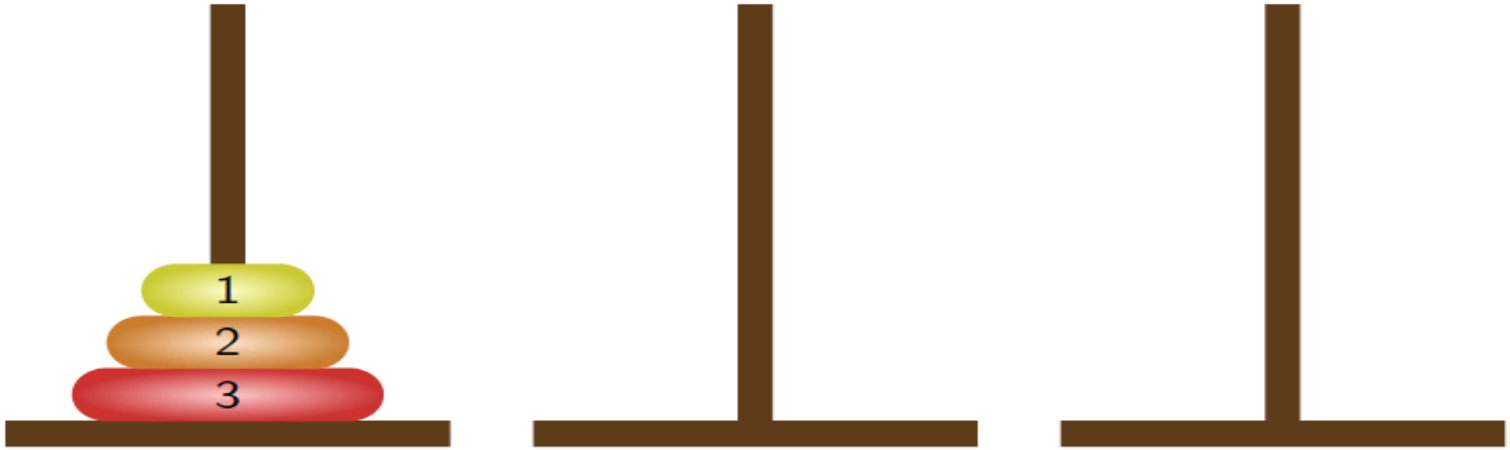


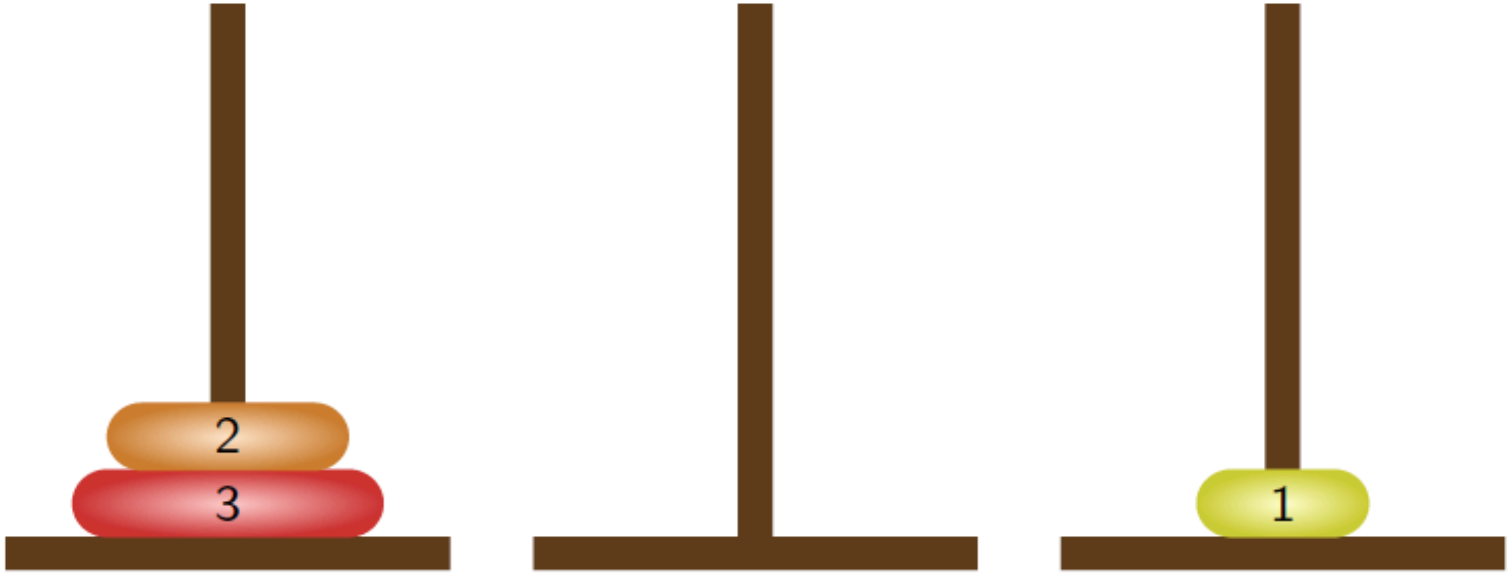
Moved disc from pole 2 to pole 3.



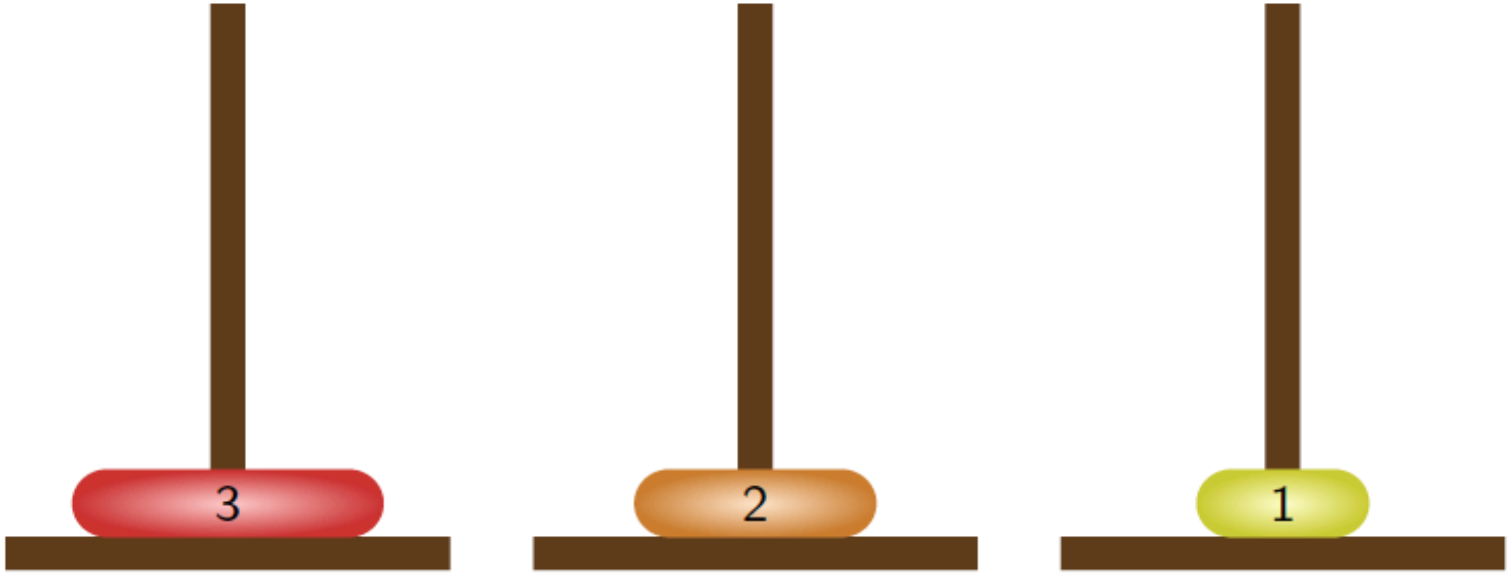

ok


1
2

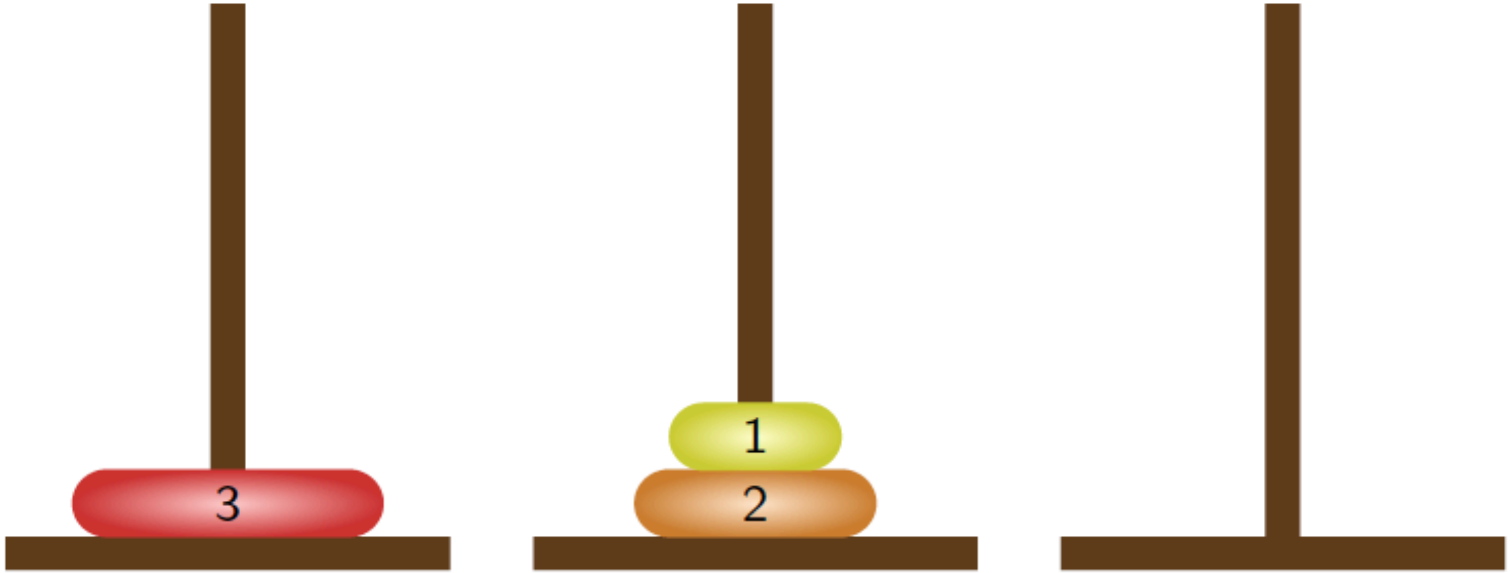

1
2
3



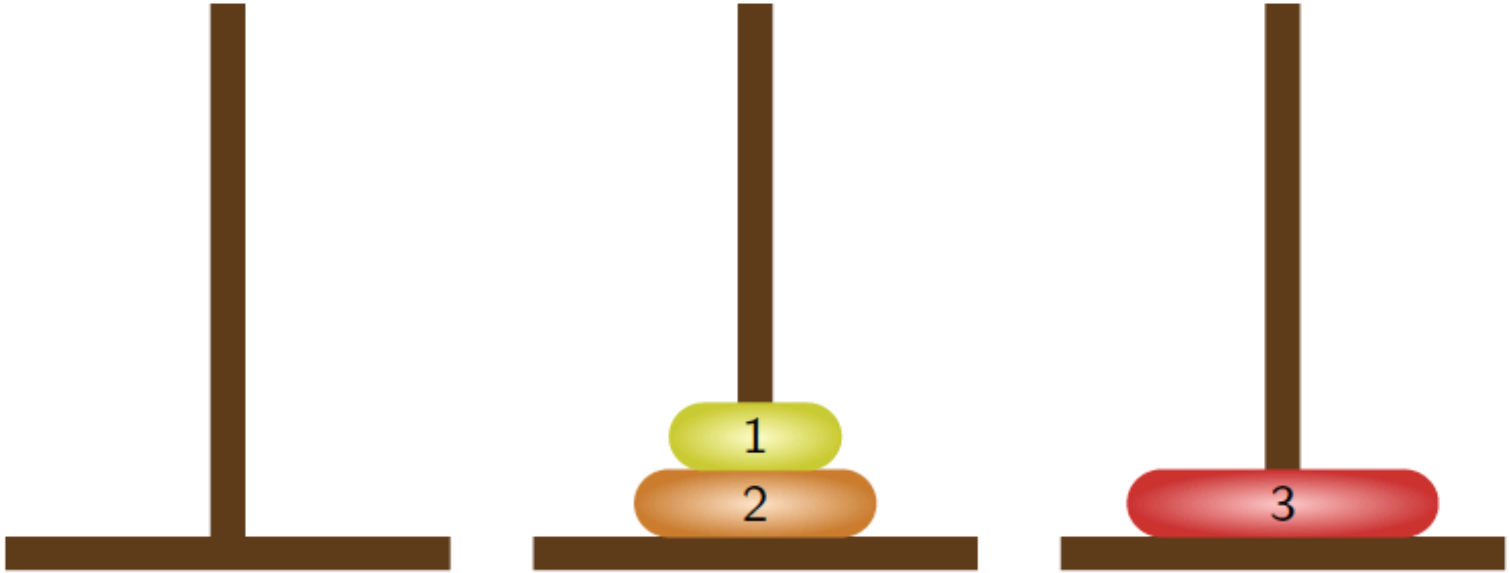
Moved disc from pole 1 to pole 3.



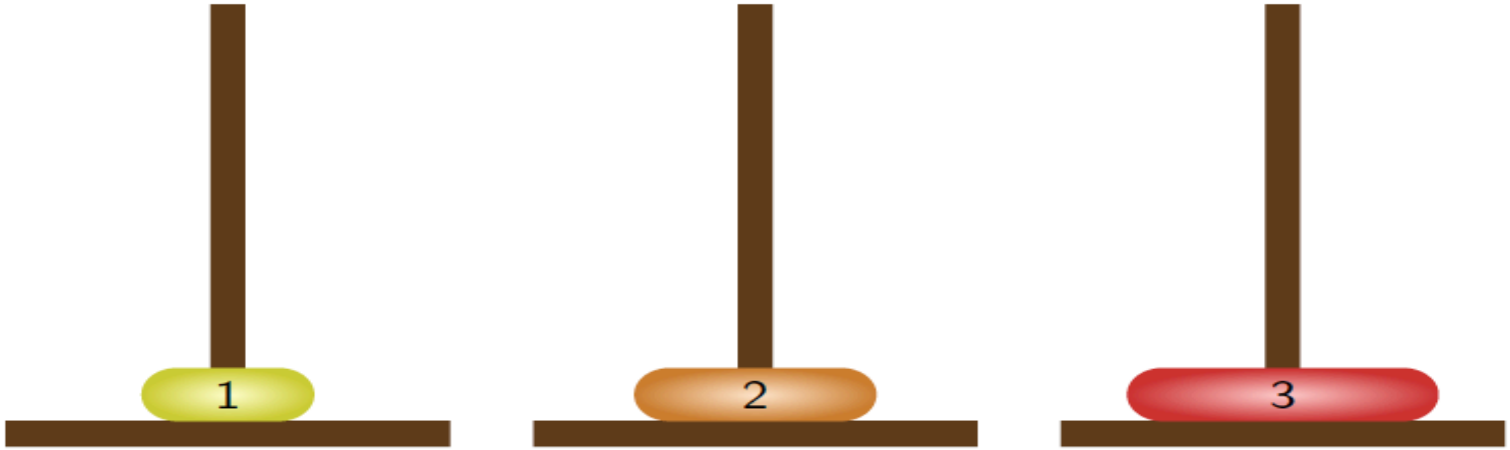
Moved disc from pole 1 to pole 2.



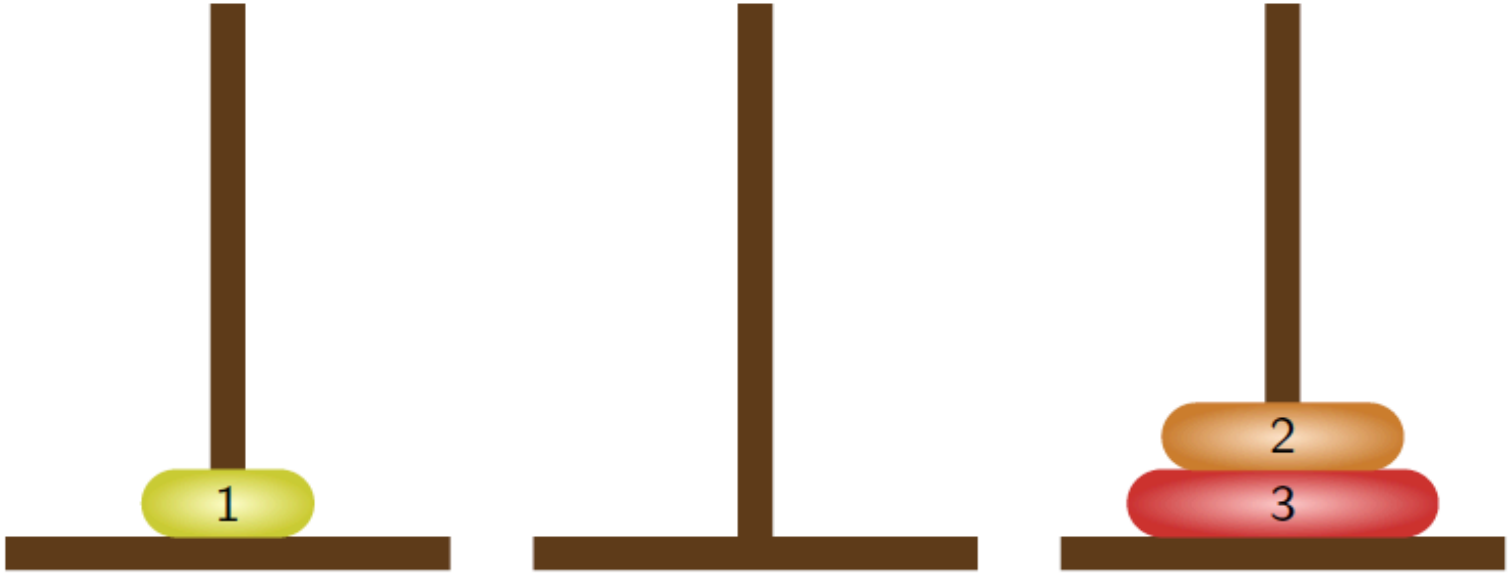
Moved disc from pole 3 to pole 2.



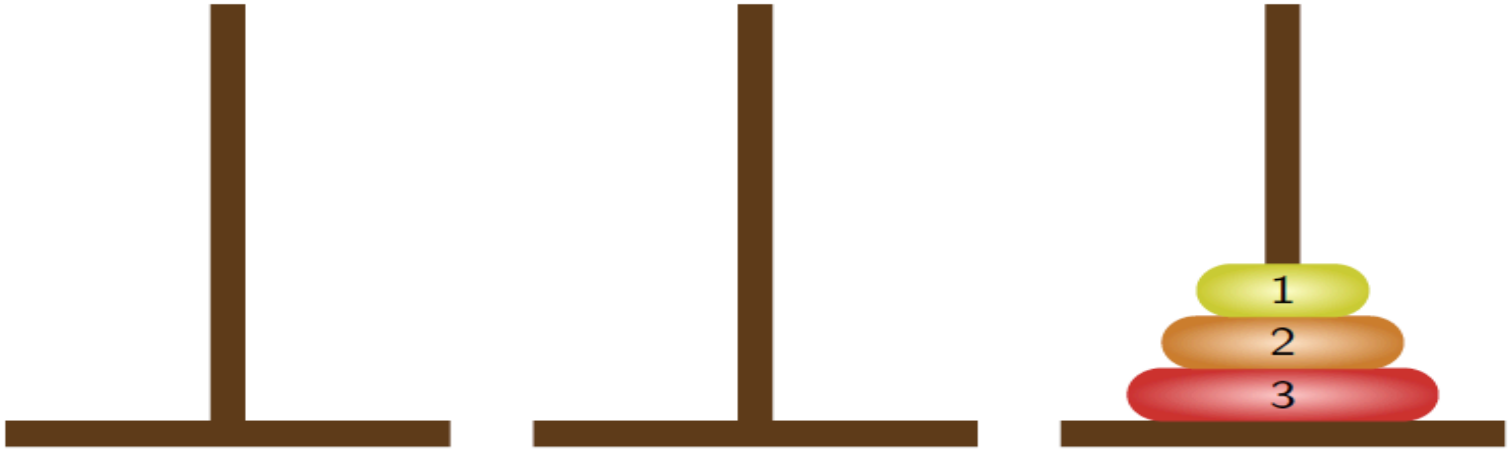
Moved disc from pole 1 to pole 3.



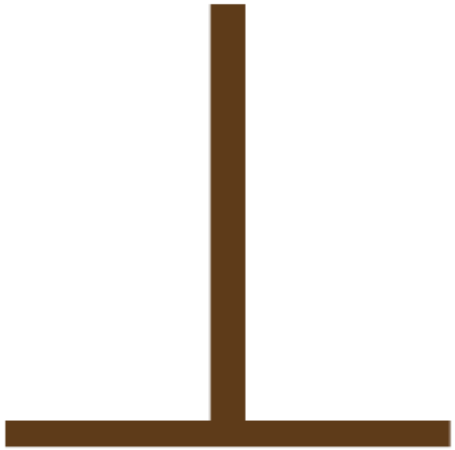
Moved disc from pole 2 to pole 1.



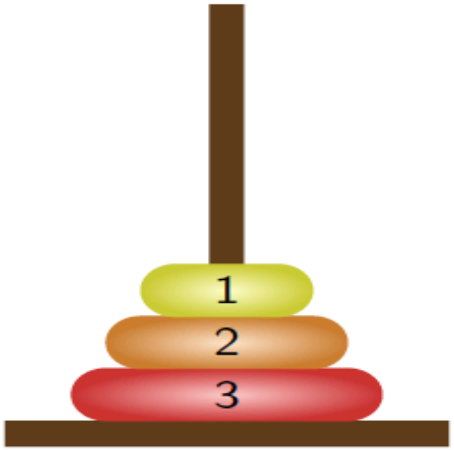
Moved disc from pole 2 to pole 3.

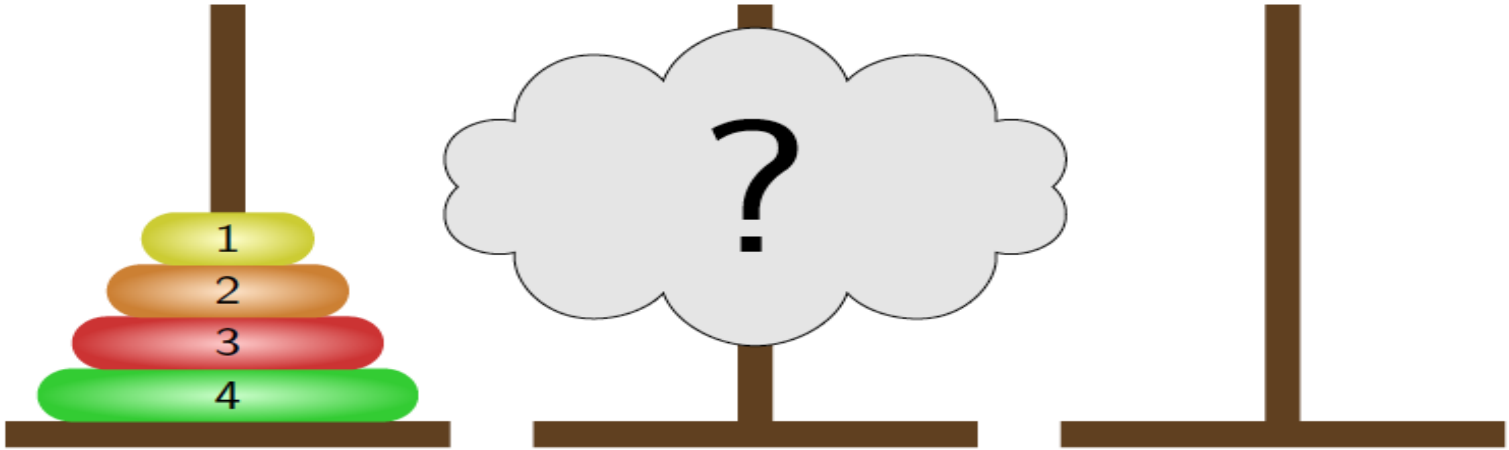


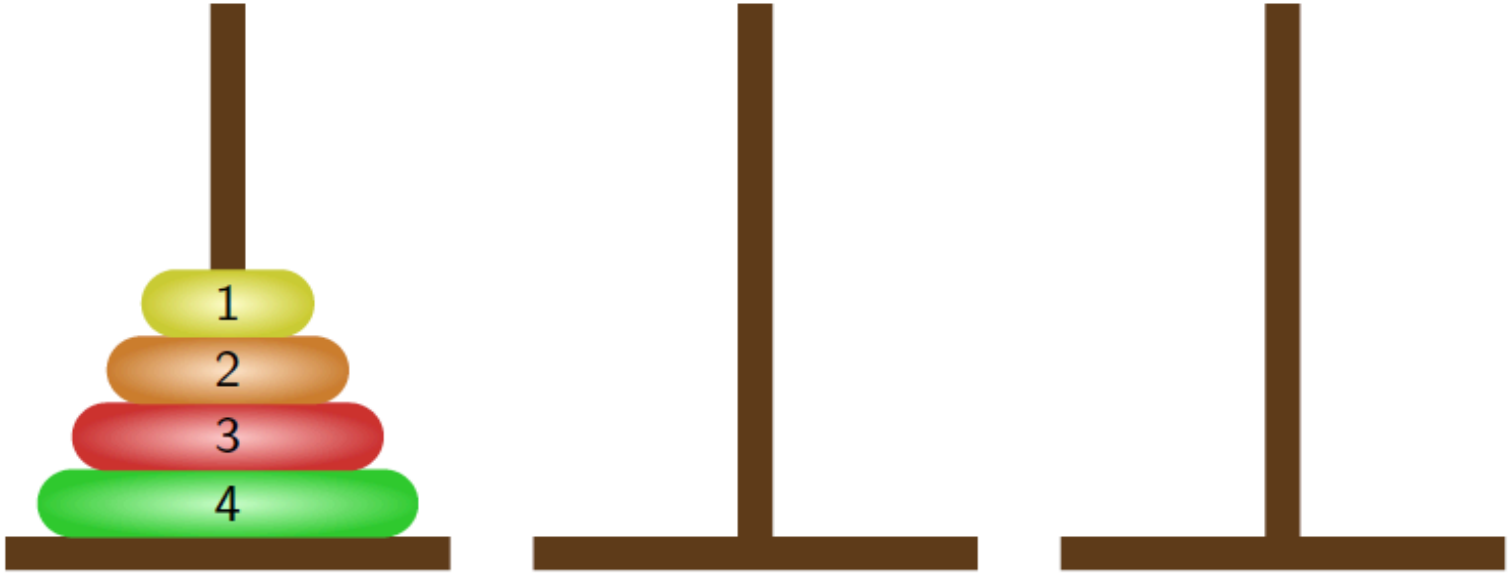
Moved disc from pole 1 to pole 3.

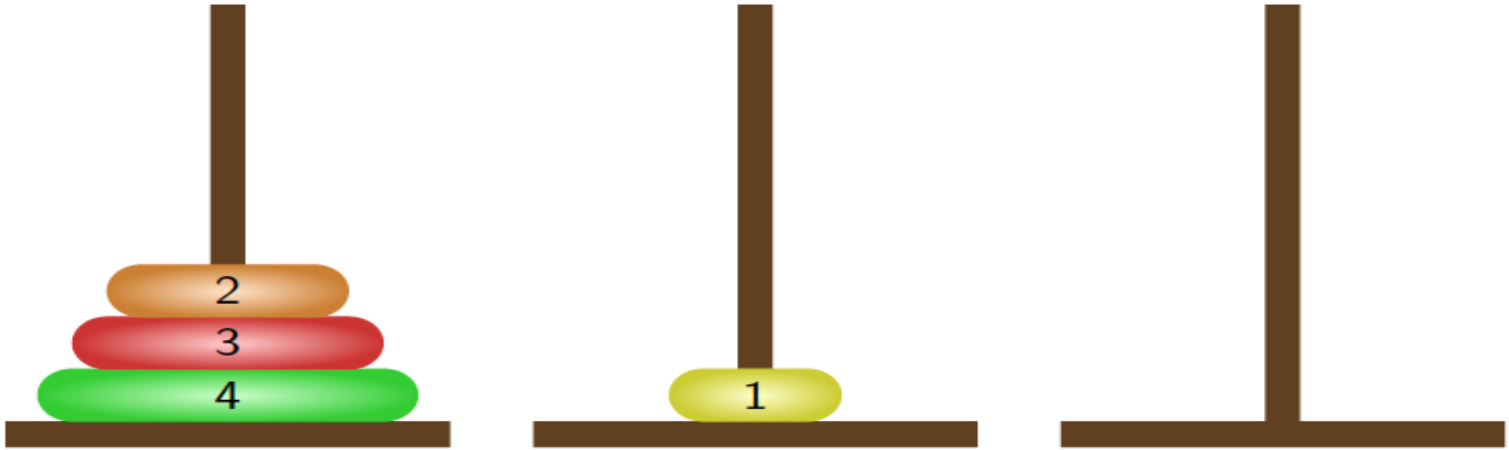



ok

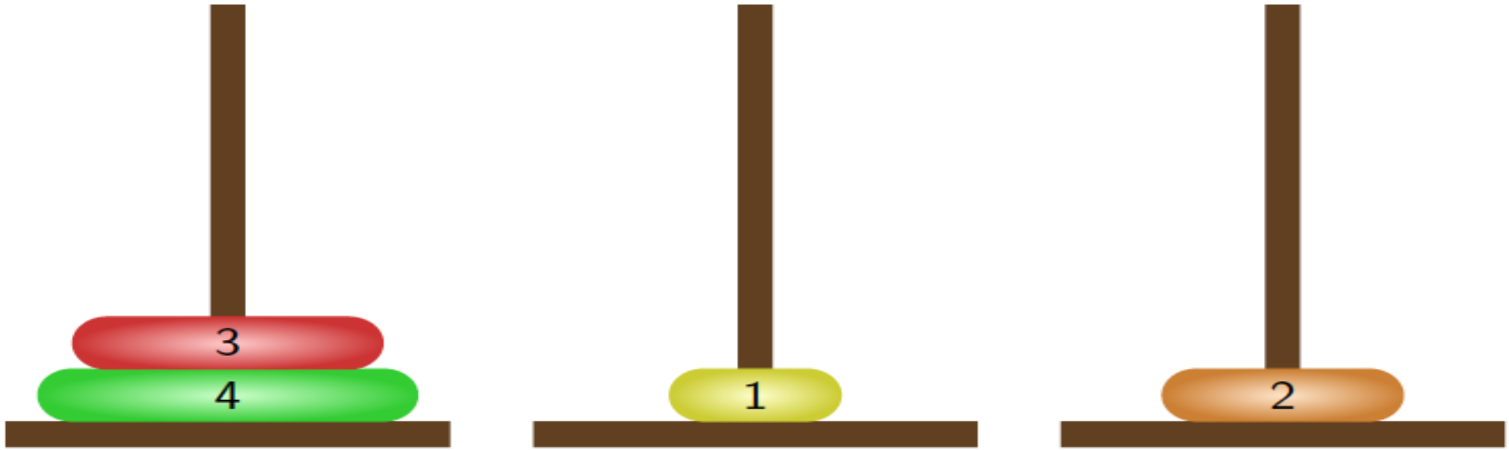

1
2
3


1
2
3
4
?


1
2
3
4

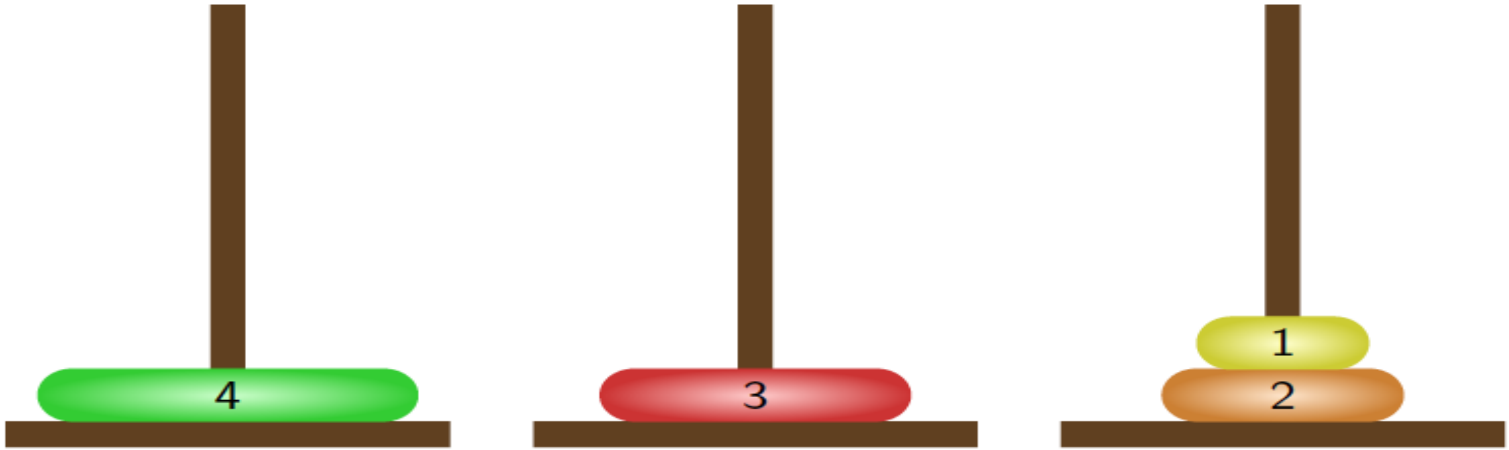


Moved disc from pole 1 to pole 2.

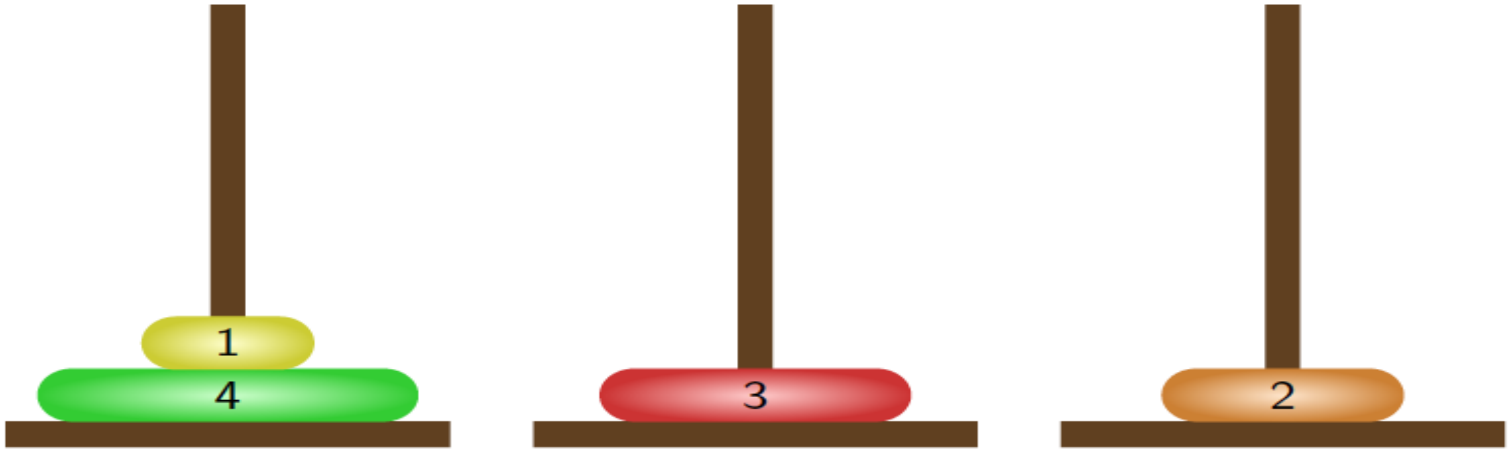


Moved disc from pole 1 to pole 3.

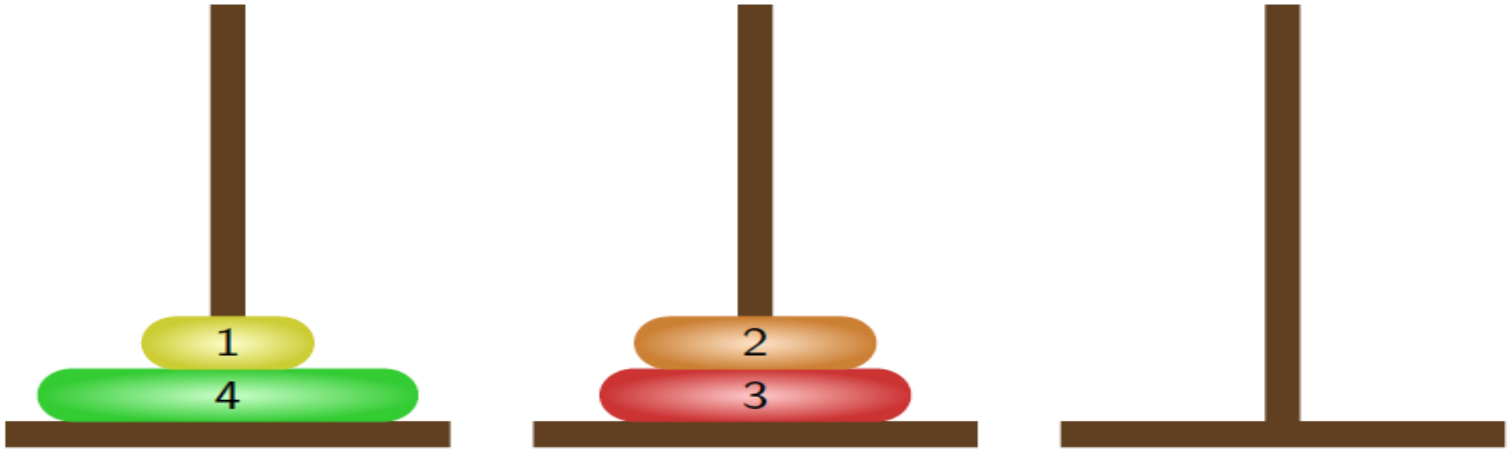
Moved disc from pole 2 to pole 3.



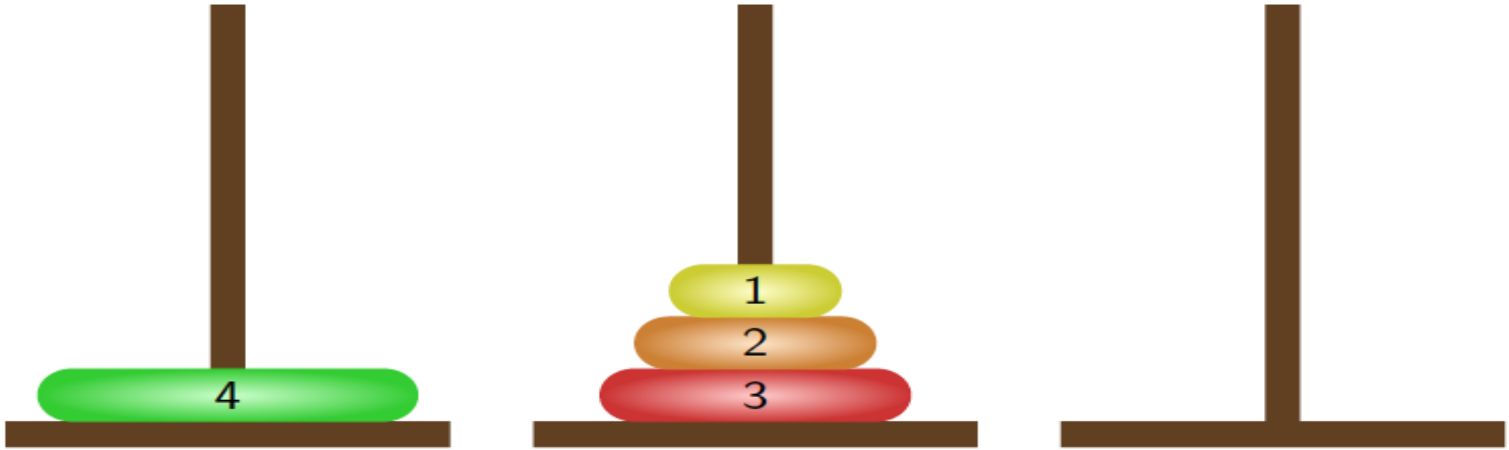
Moved disc from pole 1 to pole 2.



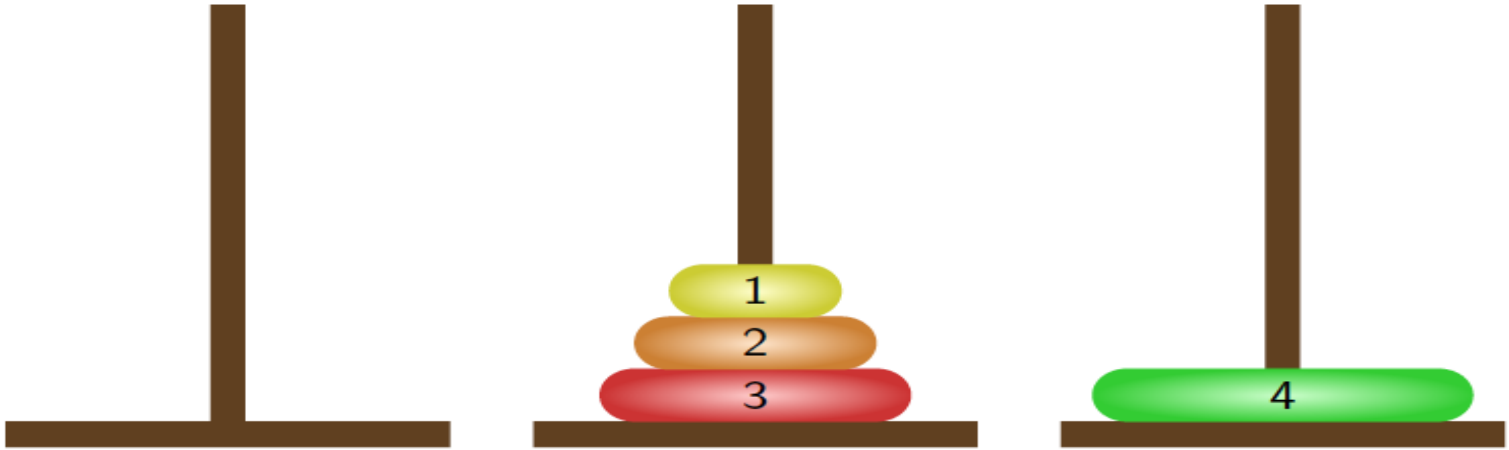
Moved disc from pole 3 to pole 1.



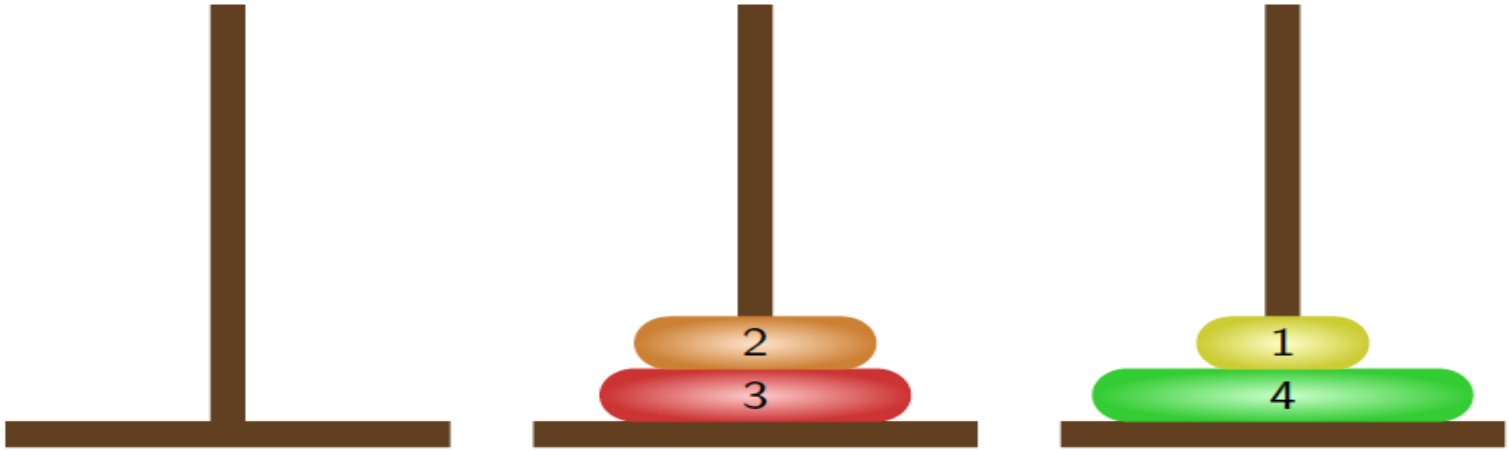
Moved disc from pole 3 to pole 2.



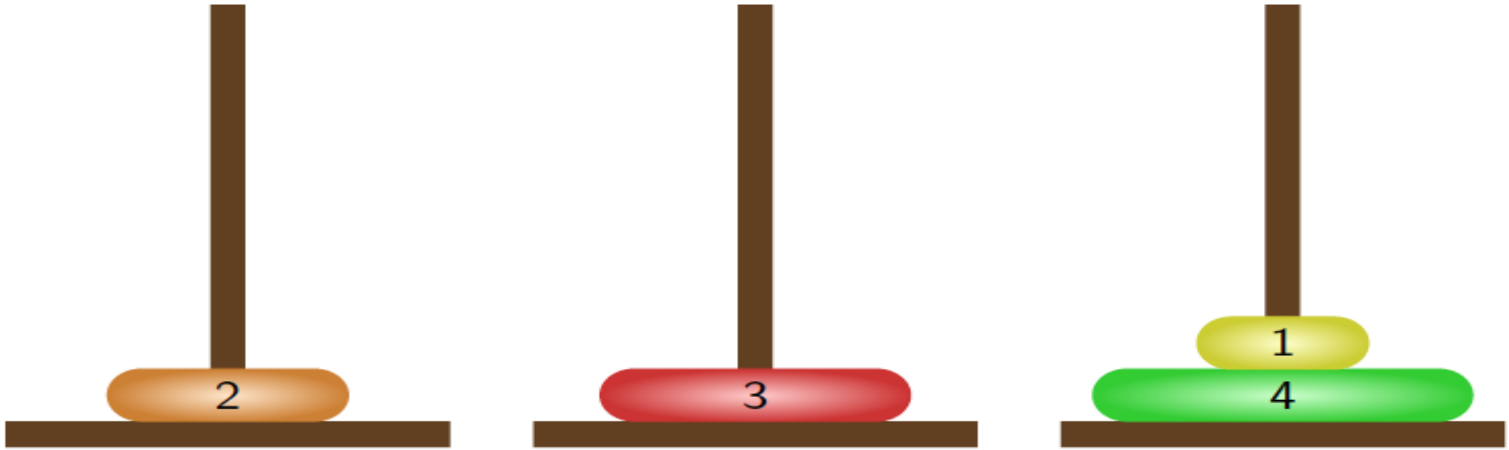
Moved disc from pole 1 to pole 2.



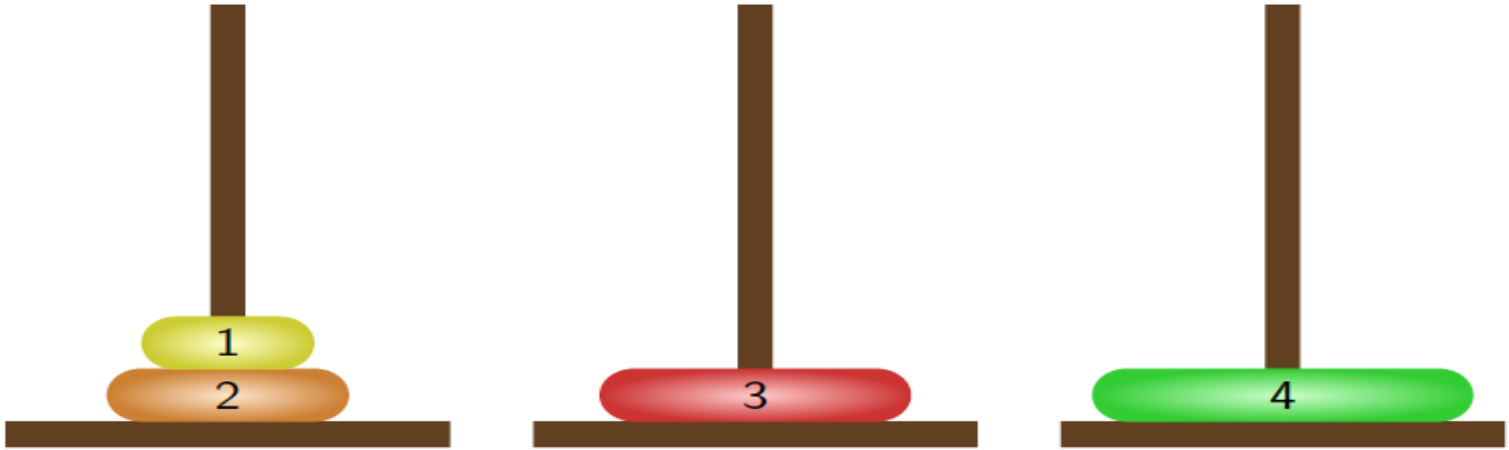
Moved disc from pole 1 to pole 3.



Moved disc from pole 2 to pole 3.

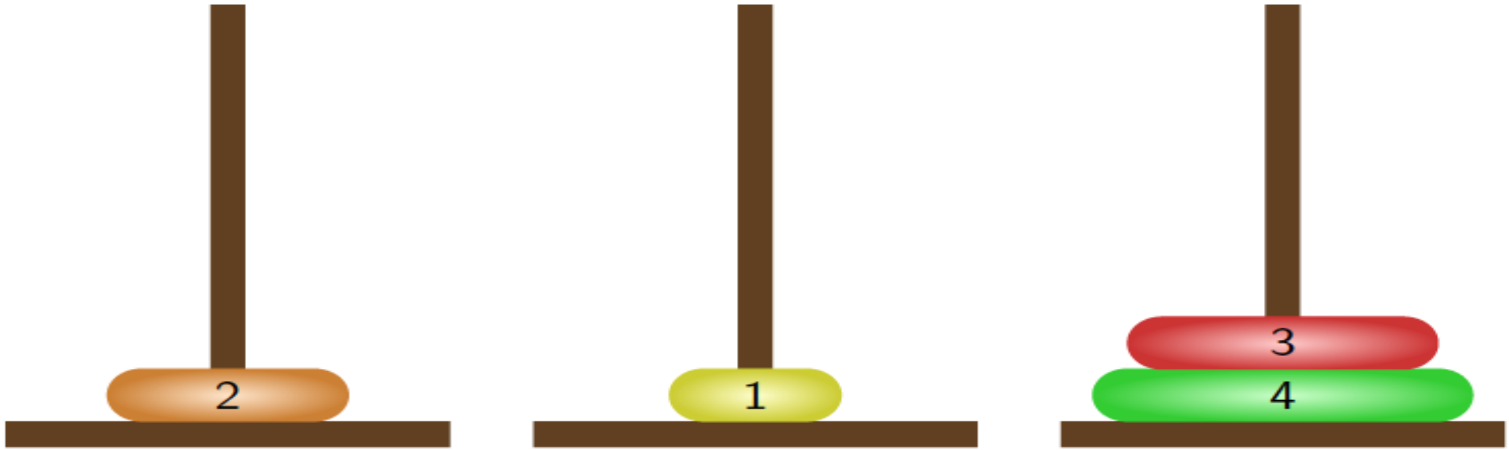


Moved disc from pole 2 to pole 1.

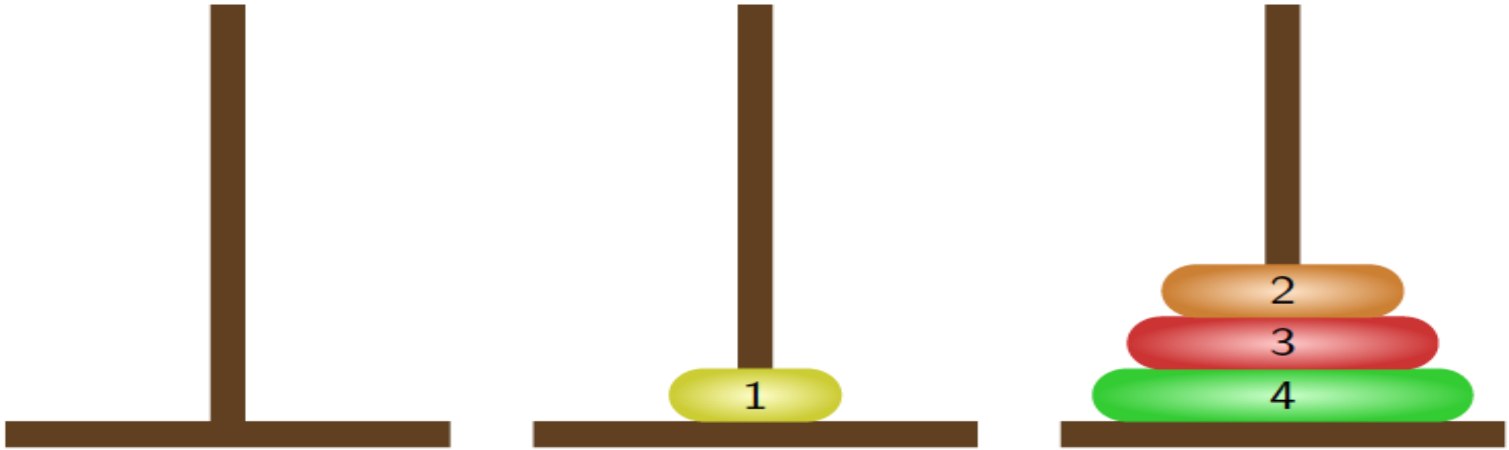


Moved disc from pole 3 to pole 1.

Moved disc from pole 2 to pole 3.



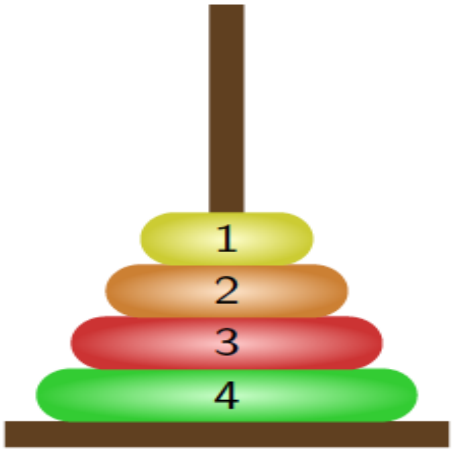
Moved disc from pole 1 to pole 2.

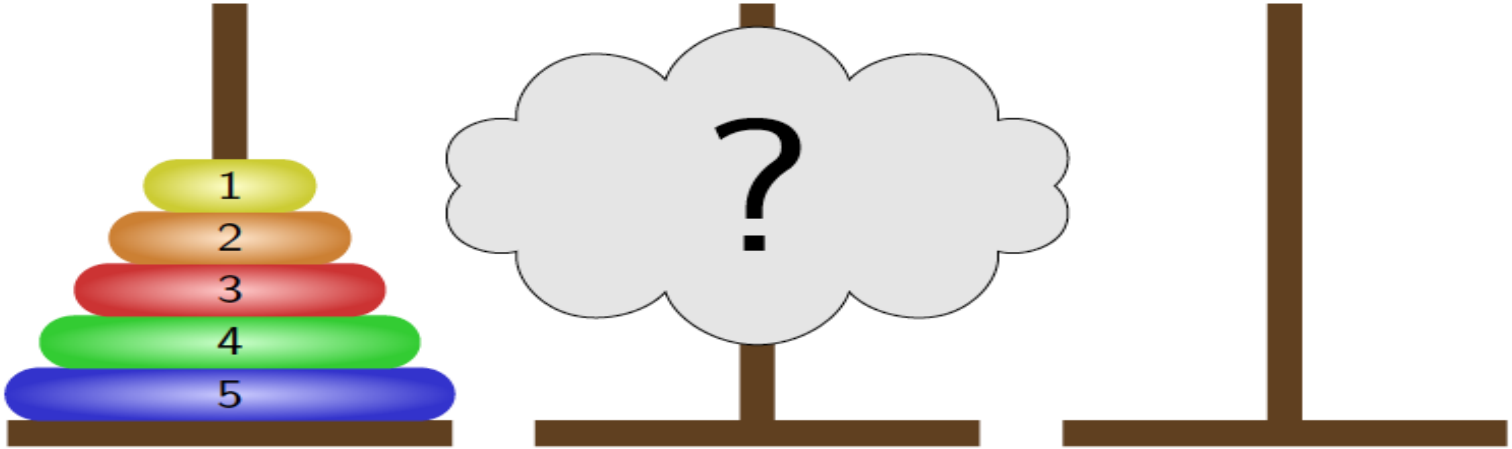


Moved disc from pole 1 to pole 3.

Moved disc from pole 2 to pole 3.


ok


1
2
3
4

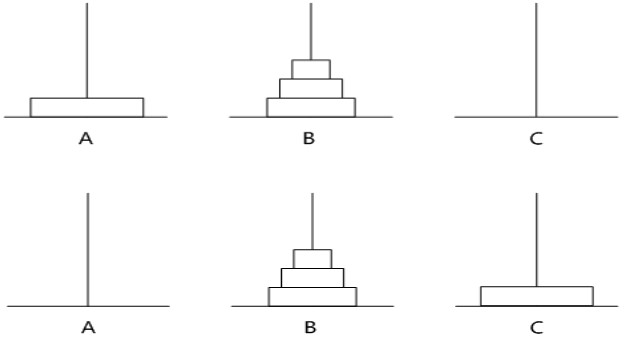

1
2
3
4
5
?

递归解决方案

- 将前 n-1 个盘子，通过 C，从 A 移动到 B
- 从 A 到 C 移动第 n 个盘子
- 将前 n-1 个盘子，通过 A，从 B 移动到 C

递归 - 汉诺塔

- 定义函数hanoi(n, A, B, C)表示把A上的n个盘子移动到C上，其中可以用到B

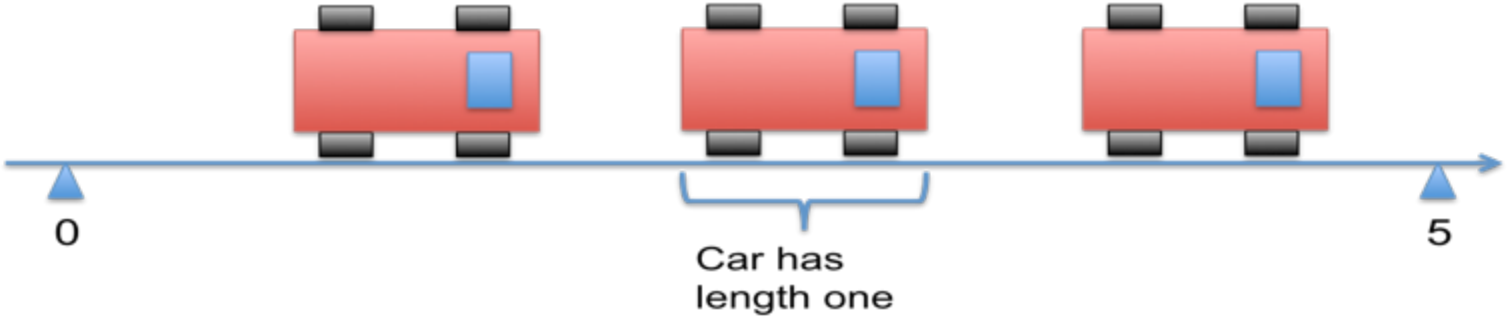



```
def hanoi(n, A, B, C):
    if n == 1:
        print "Move disk ", n, " from ", A, " to ", C
    else:
        hanoi (n-1, A, C, B)
        print "Move disk ", n, " from ", A, " to ", C
        hanoi (n-1, B, A, C)

n = int(raw_input("请输入一个整数: "))
hanoi(n, '左', '中','右')
```

路边停车

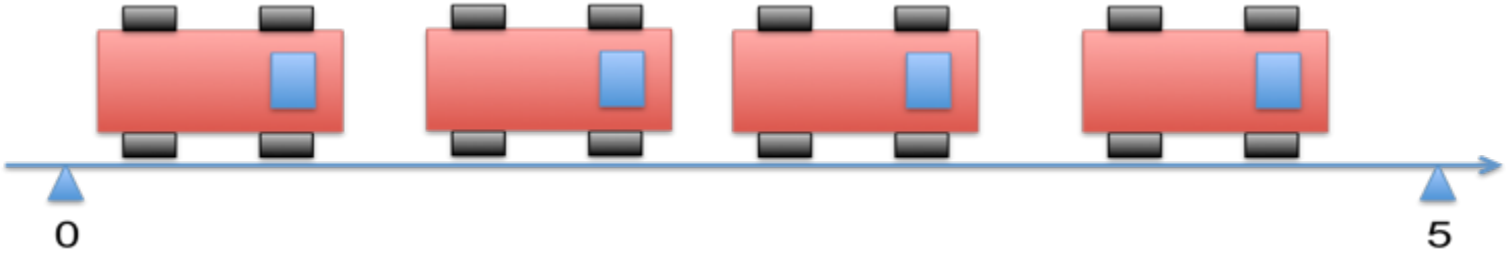
随机停车

- 长度为5的马路，平均能停多少量长度为1的汽车？

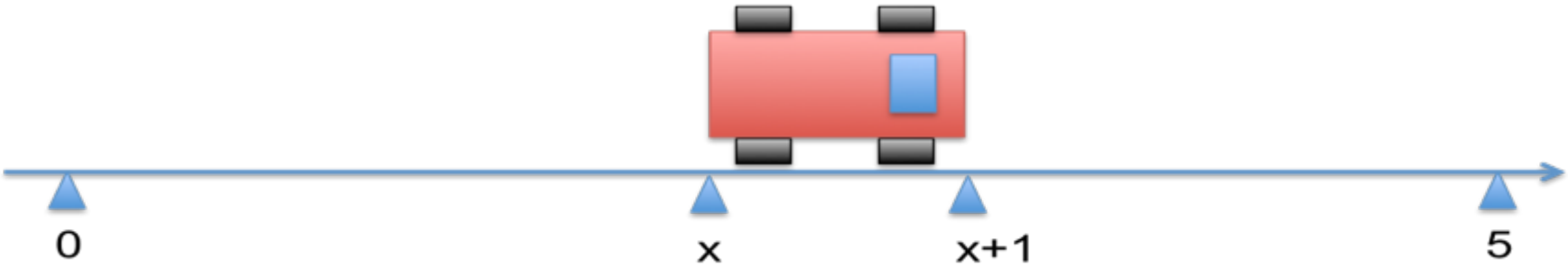


随机停车

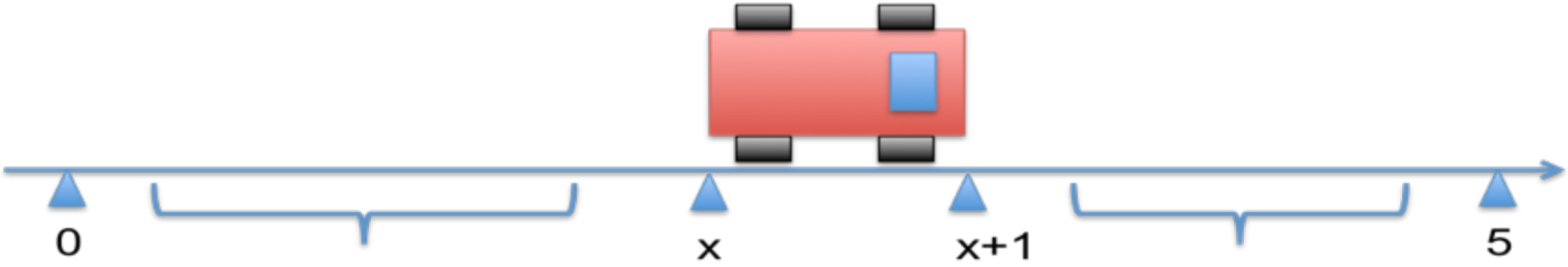
- 长度为5的马路，平均能停多少量长度为1的汽车？



随机停车



随机停车



Place cars randomly in these ranges

随机停车


```
def park_randomly(low, high):
    if high - low < 1:
        return 0;
    else:
        x = random.uniform(low, high - 1)
        return 1 + park_randomly(low, x) \
                 + park_randomly(x + 1, high)
```

随机停车

- 当宽度w足够大时，平均停车约 0.7475972w 辆
- 常数 0.7475972 被称作 Renyi 停车常数
- 该算法巧妙的运用了递归思想，将大问题分解为更小的、独立的相似问题，然后分别加以解决
- 许多问题能够以此方式解决

递归的时间开销


```
def fib_loop(n):
    if n == 1 or  n == 2:
        return 1
    else:
        i = 2
        f1 = 1
        f2 = 1
        while(i < n):
            f3 = f1 + f2
            f1 = f2
            f2 = f3
            i = i + 1
        return f3
```

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

fib_loop(100)	不到0.01秒
fib_recursive(100)	超过1小时

时间都去哪了？

递归的时间开销

fib(4)

```
def fib(4):
    if 4 == 1 or 4 == 2:
        return 1
    else:
        return fib(4-1) + fib(4-2)
```

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

```
def fib(3):
    if 3 == 1 or  3 == 2:
        return 1
    else:
        return fib(3-1) + fib(3-2)
```

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

递归的优劣分析

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- **优势（strength）**
  - 它能使一个蕴含递归关系且结构复杂的程序简洁精炼，增加可读性
  - 特别是在难于找到从边界到解的全过程的情况下，如果把问题推进一步,其结果仍维持原问题的关系
- **劣势（weakness）**
  - 嵌套层次深，函数调用开销大
  - 重复计算

字符串

车万翔

哈尔滨工业大学

学生的属性

- 姓名： 小明
- 性别： 男
- 年龄： 15
- 身高： 1.48
- 体重： 43.2
- 籍贯： 江苏

字符串的应用


```
 9 import pygame, sys
10 from pygame.locals import *
11
12 pygame.init()
13
14 FPS = 30 # frames per second setting
15 fpsClock = pygame.time.Clock()
16
17 # set up the window
18 DISPLAYSURF = pygame.display.set_mode((400, 300), 0, 32)
19 pygame.display.set_caption('Animation')
20
21 WHITE = (255, 255, 255)
22 catImg = pygame.image.load('cat.png')
23 catx = 10
24 caty = 10
25 direction = 'right'
26
27 while True: # the main game loop
28     DISPLAYSURF.fill(WHITE)
29
30     if direction == 'right':
31         catx += 5
32         if catx == 280:
33             direction = 'down'
34     elif direction == 'down':
35         caty += 5
36         if caty == 220:
37             direction = 'left'
38     elif direction == 'left':
39         catx -= 5
40         if catx == 10:
41             direction = 'up'
42     elif direction == 'up':
43         caty -= 5
44         if caty == 10:
45             direction = 'right'
46
```

cat

Replace with: dog

Replace/find Replace all

字符串查找和替换

字符串的应用


Google
哈尔滨工业大学
网页 图片 地图 新闻 更多 搜索工具
找到约 8,800,000 条结果 （用时 0.32 秒）
哈尔滨工业大学
www.hit.edu.cn/
包括学校总体介绍，招生就业，院系及教学科研简介，学生管理等。
评分： 26 / 30 · 26 条 Google 评论 · 撰写评论
黑龙江省哈尔滨市南岗区西大直街92号 邮政编码: 150001
0451 8641 2114
哈工大电子邮件系统 - 院系部门 - 学校简介
哈尔滨工业大学_百度百科
baike.baidu.cn/view/13897.htm
哈尔滨工业大学，隶属于国家工业和信息化部，是由工信部、教育部、黑龙江省共建的全国重点大学，其前身是创建于1920年的哈尔滨中俄工业学校，是中国近代培养 ...
学校简介 - 师资力量 - 院系设置 - 教学情况
哈尔滨工业大学,分数线,专业设置_新浪院校库_新浪教育_新浪网
kaoshi.edu.sina.com.cn › 所有院校
哈尔滨工业大学隶属于工业和信息化部，是由工信部、教育部、黑龙江省共建的国家重点大学，是首批进入国家"211工程"和"985工程"建设的若干所大学之一。
2013哈尔滨工业大学录取分数线_哈尔滨工业大学各省 ... - 高考院校库
college.gaokao.com › 全部高校
50+ 项 – 高考院校库为广大高中生提供哈尔滨工业大学录取分数线、哈尔滨工业 ...
专业名称 专业小类 平均分 最高分
城市规划 土建类 573 630
软件工程 电气信息类 574 609
哈尔滨工业大学的新闻搜索结果
哈工大研究生院副院长马广富分享研究生培养问题
中国教育在线 - 22 小时前
哈工大研究生院副院长马广富分享研究生培养问题，6月27日，由中国教育在线举办的.
中国教育在线
利海综合商场
博学院
哈尔滨工业大学
Map data ©2013 AutoNavi, Google
哈尔滨工业大学
路线
哈尔滨工业大学，简称哈工大，创建于1920年，是一所以理工为主，理、工、文、管相结合，多学科、开放式、研究型国家重点大学，为C9联盟成员之一。1996年首批进入中国"211工程"重点建设的院校之一。 维基百科
地址: 黑龙江省哈尔滨市南岗区西大直街92号 邮政编码: 150001
电话号码: 0451 8641 2114
省份: 黑龙江省
成立时间: 1920 年
用户还搜索了
哈尔滨工程大学
哈尔滨市
清华大学
北京市
哈尔滨理工大学
哈尔滨市
北京大学
北京市
大连理工大学
大连市
反馈/详情

搜索引擎

字符串定义

- 字符串（String）是一个字符的序列
- 使用成对的单引号或双引号括起来

```
two_quote_str ="hello world"
one_quote_str =’hello world’
```

- 或者三引号（"""或'''）
  - 保留字符串中的全部格式信息

```
’’’ this is
a test
today’’’
```

基本的字符串运算

- 长度（len()函数）

```
>>> first_name = 'Michael'
>>> len(first_name)
7
```

- 拼接（+）

```
>>> name = first_name + ' Jordan '
>>> print name
Michael Jordan
```

- 重复（*）

```
>>> name * 3
'Michael Jordan Michael Jordan Michael Jordan '
```

成员运算符（in）

- 判断一个字符串是否是另一个字符串的子串
  - 返回值：True 或者 False

```
>>> name = 'Michael Jordan'
>>> 'a' in name
True
>>> 'Jordan' in name
True
>>> 'jordan' in name
False
>>> 'x' in name
False
```

for 语句

- 枚举字符串的每个字符

```
>>> my_str = 'hello world'
>>> for char in my_str:
...     print char
...
h
e
l
l
o

w
o
r
l
d
```

示例

- 编写 vowels_count 函数，计算一个字符串中元音字母（aeiou或AEIOU）的数目

```
def vowels_count(string):
    ret = 0
    for c in string:
        if c in 'aeiouAEIOU':
            ret += 1

    return ret
```

字符串索引（index）

- 字符串中每个字符都有一个索引值（下标）
- 索引从0（前向）或-1（后向）开始

forward index	0	1	2	3	4	5	6	7	8	9	10
	h	e	l	l	o		w	o	r	l	d
backward index	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

- 索引运算符 [ ]

```
>>> my_str = 'hello world'
>>> my_str[2]
'l'
>>> my_str[-1]
'd'
>>> my_str[11]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: string index out of range
```

切片（Slicing）

- **选择字符串的子序列**
- **语法 [start : finish]**
  - start：子序列开始位置的索引值
  - finish：子序列结束位置的下一个字符的索引值

h	e	l	l	o		w	o	r	l	d
0	1	2	3	4	5	6	7	8	9	10

- **如果不提供 start 或者 finish，默认 start 为第一个字符开始，finish 为最后一个字符**

```
>>> my_str = 'hello world'
>>> my_str[6:10]
'worl'
>>> my_str[6:]
'world'
>>> my_str[:6]
'hello '
```

计数参数（Count by Argument）

- **接收三个参数**
  - [start : finish : countBy]
  - 默认 countBy 为 1

```
>>> my_str = 'hello world'
>>> my_str[0:11:2]
'hlowrd'
```

- **获得逆字符串**

```
>>> my_str = 'spam'
>>> reverse_str = my_str[::-1]
>>> print reverse_str
maps
```

字符串是不可变的（Immutable）

- 一旦生成，则内容不能改变

```
>>> my_str = 'hello world'
>>> my_str[1] = 'a'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item
                                    assignment
```

- 通过切片等操作，生成一个新的字符串

```
>>> my_str = my_str[:1] + 'a' + my_str[2:]
>>> print my_str
hallo world
```

字符串方法（Methods）

- **方法**
  - 对象提供的函数
  - 如：my_str.replace(old, new) 生成一个新的字符串，其中使用 new 替换 old 子串

```
>>> my_str = 'hello world'
>>> my_str = my_str.replace('e', 'a')
>>> print my_str
hallo world
```

- **注意**
  - replace 方法返回一个新的字符串，原字符串内容不变
  - 新字符串重新赋值给原来的变量

更多字符串方法

- **find**

```
>>> my_str = 'hello world'
>>> my_str.find('l') # find index of 'l' in my_str
2
```

- **split**

```
>>> my_str = 'hello world'
>>> my_str.split()
['hello', 'world']
```

- **其它方法**
  - dir(str)
  - http://docs.python.org/lib/string-methods.html

示例：人名游戏

- **读取人名列表文件 names.txt，将每个人名转换为首字母大写，其它字母小写的格式**

names.txt

```
AA
AAB
AABERG
AABY
AADLAND
AAFEDT
AAGAARD
AAGARD
AAGESEN
AAKER
AAKHUS
AAKRE
AALAND
AALBERS
AALDERINK
AALFS
AALGAARD
AALTO
AAMODT
AAMOLD
AAMOT
AAMOTH
AANDERUD
AANENSON
AANERUD
AANESTAD
AANONSEN
AARDAL
AARDEMA
AARDSMA
```

- 文件操作
- 打开文件
  f = open(filename, mode)
  - mode 有 r（读，默认），w（写）等
- 按行读取文件内容

```
for line in f:
    pass
```

- 关闭文件
  f.close()
- 写文件
  f.write(str)
- 更多说明
  - https://docs.python.org/2/tutorial/inputoutput.html#reading-and-writing-files

示例：人名游戏

- 读取人名列表文件 names.txt，将每个人名转换为首字母大写，其它字母小写的格式

names.txt

```
AA
AAB
AABERG
AABY
AADLAND
AAFEDT
AAGAARD
AAGARD
AAGESEN
AAKER
AAKHUS
AAKRE
AALAND
AALBERS
AALDERINK
AALFS
AALGAARD
AALTO
AAMODT
AAMOLD
AAMOT
AAMOTH
AANDERUD
AANENSON
AANERUD
AANESTAD
AANONSEN
AARDAL
AARDEMA
AARDSMA
```

```
f = open('names.txt')

for line in f:
    name = line.strip()
    print name.title()

f.close()
```

示例：人名游戏

- 编写一个名为 is_palindrome 的函数，判断一个人名是否为回文，如"BOB" 是回文

```
def is_palindrome(name):
    low = 0
    up = len(name) - 1
    while low < up:
        if name[low] != name[up]:
            return False
        low += 1
        up -= 1
    return True
```

- 递归实现

```
def is_palindrome(name):
    if len(name) < 2:
        return True
    if name[0] == name[-1]:
        return is_palindrome(name[1:-1])
    else:
        return False
```

字符串比较

- 任何一个字符都对应一个数字
  - ASCII (American Standard Code for Information Interchange)
- 直接比较对应数字的大小

ASCII CONTROL CODE CHART

b7 b6 b5 / BITS b4 b3 b2 b1	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
	CONTROL		SYMBOLS NUMBERS		UPPER CASE		LOWER CASE	
0 0 0 0	0 NUL 0	16 DLE 20	32 SP 40	48 0 60	64 @ 100	80 P 120	96 ‘ 140	112 p 160
0 0 0 1	1 SOH 1	17 DC1 21	33 ! 41	49 1 61	65 A 101	81 Q 121	97 a 141	113 q 161
0 0 1 0	2 STX 2	18 DC2 22	34 ” 42	50 2 62	66 B 102	82 R 122	98 b 142	114 r 162
0 0 1 1	3 ETX 3	19 DC3 23	35 # 43	51 3 63	67 C 103	83 S 123	99 c 143	115 s 163
0 1 0 0	4 EOT 4	20 DC4 24	36 $ 44	52 4 64	68 D 104	84 T 124	100 d 144	116 t 164
0 1 0 1	5 ENQ 5	21 NAK 25	37 % 45	53 5 65	69 E 105	85 U 125	101 e 145	117 u 165
0 1 1 0	6 ACK 6	22 SYN 26	38 & 46	54 6 66	70 F 106	86 V 126	102 f 146	118 v 166

字符串比较

- 字典序（Dictionary order）
  - 首先比较两个字符串的第一个字符
  - 如果相同，则比较下一个字符
  - 如果不同，则字符串的大小关系由这两个字符的关系决定
  - 如果其中一个字符为空（较短），则其更小
- 举例

```
>>> 'a' < 'b'
True
>>> 'aaab' < 'aaac'
True
>>> 'aa' < 'aab'
True
>>> 'abc' > 'bcd'
False
```

示例：人名游戏

- 编写函数 is_ascending，判断一个人名的字母是否为升序排列（允许重复字母）

```
def is_ascending(name):
    p = name[0]
    for c in name:
        if p > c:
            return False
        else:
            p = c
    return True
```

字符串格式化（Formatting）

- 输出更规格的结果
- format 方法，如：

```
>>> print "Hello {} good {}.".format(5, 'DAY')
Hello 5 good DAY.
```

- 括号的格式
  - {field name:align width.precision type}

```
>>> print math.pi
3.14159265359
>>> print 'PI is {:.4f}'.format(math.pi)
PI is 3.1416
>>> print 'PI is {:9.4f}'.format(math.pi)
PI is     3.1416
>>> print 'PI is {:e}'.format(math.pi)
PI is 3.141593e+00
```

- 参考
  - http://docs.python.org/2/library/string.html#format-string-syntax

正则表达式（Regular Expressions）

- **判断一个人名（name）是否满足下列模式**
  - Michael：name == 'Michael'
  - 以 Mi 开始：name[:2] == 'Mi'
  - 包含 cha 子串：'cha' in name
  - 包含 c?a 子串：？
  - 包含 c*e 子串：？
- **正则表达式用来描述字符串的模式**
  - . 表示任意字符
  - \d+ 表示一系列数字
  - [a-z]表示一个小写字母
  - ……
- **参考**
  - https://docs.python.org/2/library/re.html

示例：人名游戏

- 判断一个人名是否含有 C.A 模式

```
import re

f = open('names.txt')

pattern = '(C.A)'

for line in f:
    name = line.strip()
    result = re.search(pattern, name)
    if result:
        print 'Find in {}'.format(name)

f.close()
```

列表与元组

车万翔

哈尔滨工业大学

一个例子

- 读取三个数字，并计算平均数

```
num1 = float(raw_input())
num2 = float(raw_input())
num3 = float(raw_input())
avg = (num1 + num2 + num3) / 3
```

- 如果是30个数呢？

```
num1 = float(raw_input())
num2 = float(raw_input())
num3 = float(raw_input())
...

avg = (num1 + num2 + num3 + ...) / 30
```

列表（List）

- 内建（built-in）数据结构（data structure），用来存储一系列元素（items）
- 如：lst = [5.4, 'hello', 2]

lst	5.4	‘hello’	2
index	0	1	2
index	-3	-2	-1

  - lst[0] is 5.4
  - lst[3] ERROR!
  - lst[1:3] is ['hello', 2]

列表与字符串

- **相同点**
  - 索引（[ ] 运算符）
  - 切片（[:]）
  - 拼接（ + ）和重复（ * ）
  - 成员（in 运算符）
  - 长度（len( ) 函数）
  - 循环（for ）
- **不同点**
  - 使用 [ ] 生成，元素之间用逗号分隔
  - 可以包含多种类型的对象；字符串只能是字符
  - 内容是可变的；字符串是不可变的

列表的方法

- **列表内容是可变的**
  - my_list[0] = 'a'
  - my_list[0 : 2] = [1.2, 3, 5.6]
  - my_list.append(), my_list.extend() #追加元素
  - my_list.insert() #任意位置插入元素
  - my_list.pop(), my_list.remove() #删除元素
  - my_list.sort() #排序
  - my_list.reverse() #逆序
- **更多文档**
  - http://docs.python.org/2/tutorial/datastructures.html#more-on-lists

回到第一个例子

- 读取30个数字，并计算平均数

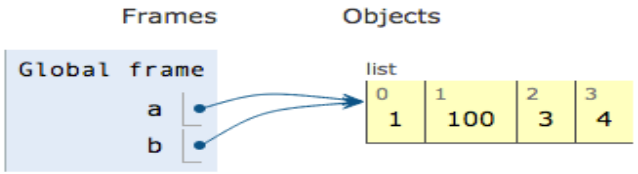
```
nums = []
for i in range(30):
    nums.append(float(raw_input()))
s = 0
for num in nums:
    s += num
avg = s / 30
print avg
```

- 内建 sum 函数
  - avg = sum(nums) / len(nums)
- 更多内建函数，如 max，min
  - http://docs.python.org/2/library/functions.html

列表赋值

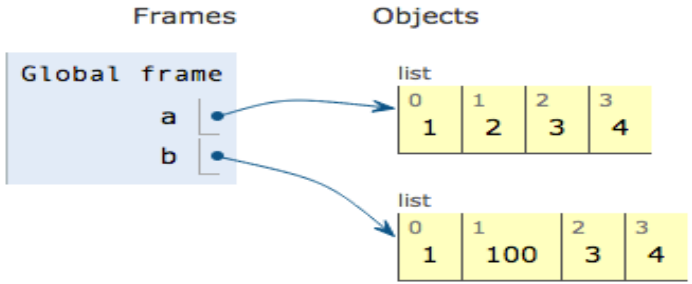
- **下列代码的执行结果是？**

```
a = [1, 2, 3, 4]
b = a
b[1] = 100
print a[1]
```



- **下面的呢？**

```
a = [1, 2, 3, 4]
b = a[:]
b[1] = 100
print a[1]
```



- **动态演示**
  - http://www.pythontutor.com/

列表作函数参数

- 如交换列表中两个元素的函数

```
def swap(a, b):
    tmp = a
    a = b
    b = tmp

x = 10
y = 20

swap(x, y)
print x, y
```

10, 20

VS.

```
def swap(lst, a, b):
    tmp = lst[a]
    lst[a] = lst[b]
    lst[b] = tmp

x = [10, 20, 30]

swap(x, 0, 1)
print x
```

示例：查找

- **在列表中查找一个值，并返回该值第一次出现的位置；如果该值不存在，则返回 -1**

```
def search(lst, x):
    for i in range(len(lst)):
        if lst[i] == x:
            return i
    return -1
```

- **list.index( ) 方法**
  - [1, 2, 2].index(2)
- **线性查找**
  - 最坏运行时间： $k_0n + k_1$

时间复杂度

- **量化一个算法的运行时间为输入长度的函数**
- **不需要显式的计算这些常数**
  - 如： 4n +10 和 100n +137 都与输入规长度正比
- **大*O*表示，只保留高阶项**
  - $4n + 4 = O(n)$
  - $137n + 271 = O(n)$
  - $n^2 + 3n + 4 = O(n^2)$
  - $2^n + n^3 = O(2^n)$
- **线性查找的时间复杂度为：*O*(n)**

时间复杂度

- **大O能告诉我们什么？**
  - 如果算法A的复杂度为O(n)，算法B的复杂度为$O(n^2)$，对于较大的输入，A总是比B快
  - 如果算法A的复杂度为O(n)，当输入规模翻倍时，运行时间也翻倍
- **大O不能告诉我们什么？**
  - 实际运行时间
    - $10^{100}n = O(n)$
    - $10^{-100}n = O(n)$
  - 对于小规模输入的行为
    - $n^3 = O(n^3)$
    - $10^6 = O(1)$

函数的增长率

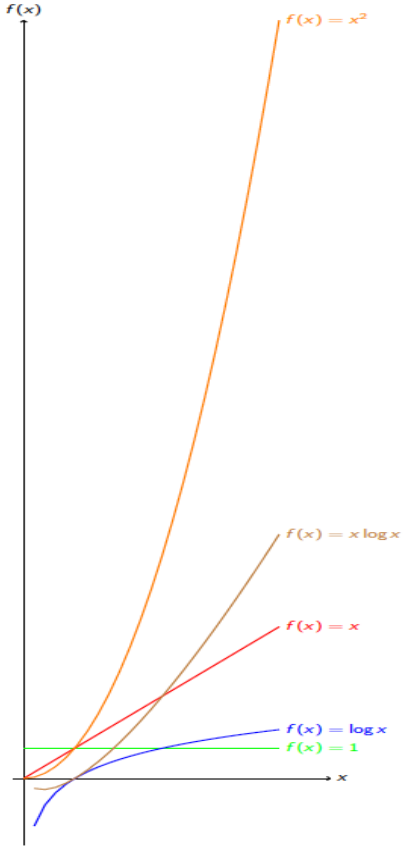


Table : 1 operation = 1 μs

Size	1	log n	n	$n \log n$	n^2	n^3
100	1μs	7μs	100μs	0.7ms	10ms	<1min
200	1μs	8μs	200μs	1.5ms	40ms	<1min
300	1μs	8μs	300μs	2.5ms	90ms	1min
400	1μs	9μs	400μs	3.5ms	160ms	2min
500	1μs	9μs	500μs	4.5ms	250ms	4min
600	1μs	9μs	600μs	5.5ms	360ms	6min
700	1μs	9μs	700μs	6.6ms	490ms	9min
800	1μs	10μs	800μs	7.7ms	640ms	12min
900	1μs	10μs	900μs	8.8ms	810ms	17min
1000	1μs	10μs	1000μs	10ms	1000ms	22min
1100	1μs	10μs	1100μs	11ms	1200ms	29min
1200	1μs	10μs	1200μs	12ms	1400ms	37min
1300	1μs	10μs	1300μs	13ms	1700ms	45min
1400	1μs	10μs	1400μs	15ms	2000ms	56min

二分查找

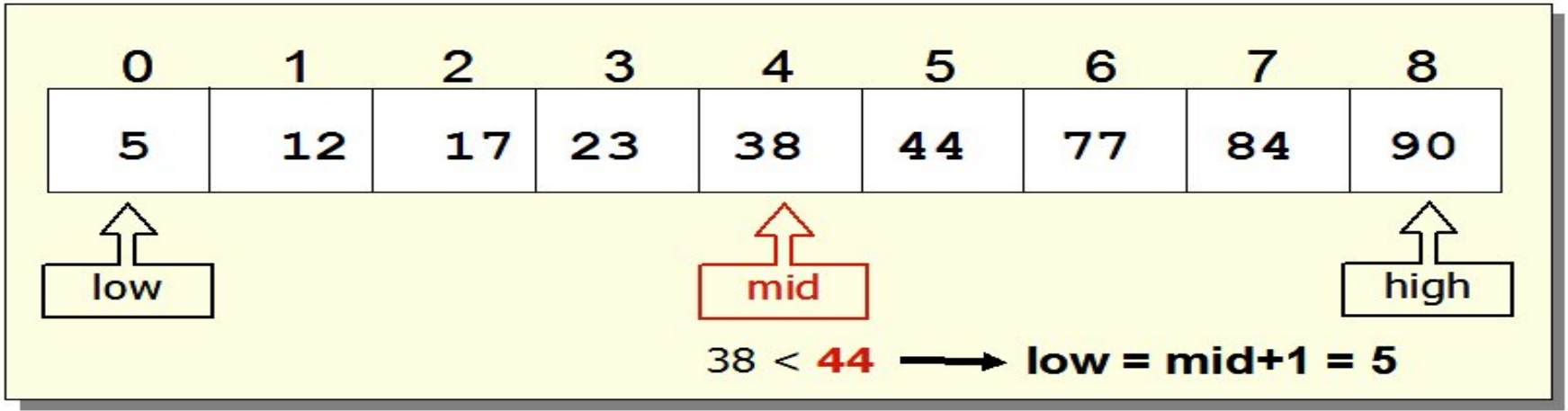
- **编写函数 bi_search，输入一个有序（由小到大）列表和一个值，如果该值在列表中，则返回相应的位置，否则返回 -1**
- **考虑以下三种情况**
  - 如果输入值**小于**列表中间的元素，则只需要在列表的**前**半部分继续查找
  - 如果输入值**大于**列表中间的元素，则只需要在列表的**后**半部分继续查找
  - 如果输入的值**等于**列表中间的元素，则返回该位置

二分查找示例

	low	high	mid
#1	0	8	4

search(44)

$$mid = \left\lfloor \frac{low + high}{2} \right\rfloor$$

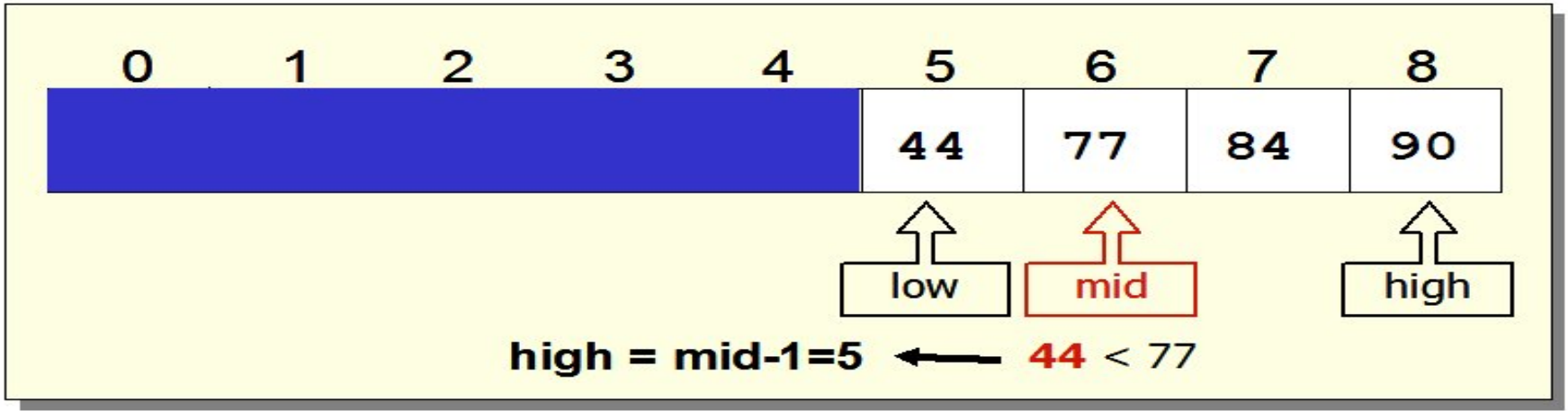


二分查找示例

	low	high	mid
#1	0	8	4
#2	5	8	6

search(44)

$$mid = \left\lfloor \frac{low + high}{2} \right\rfloor$$

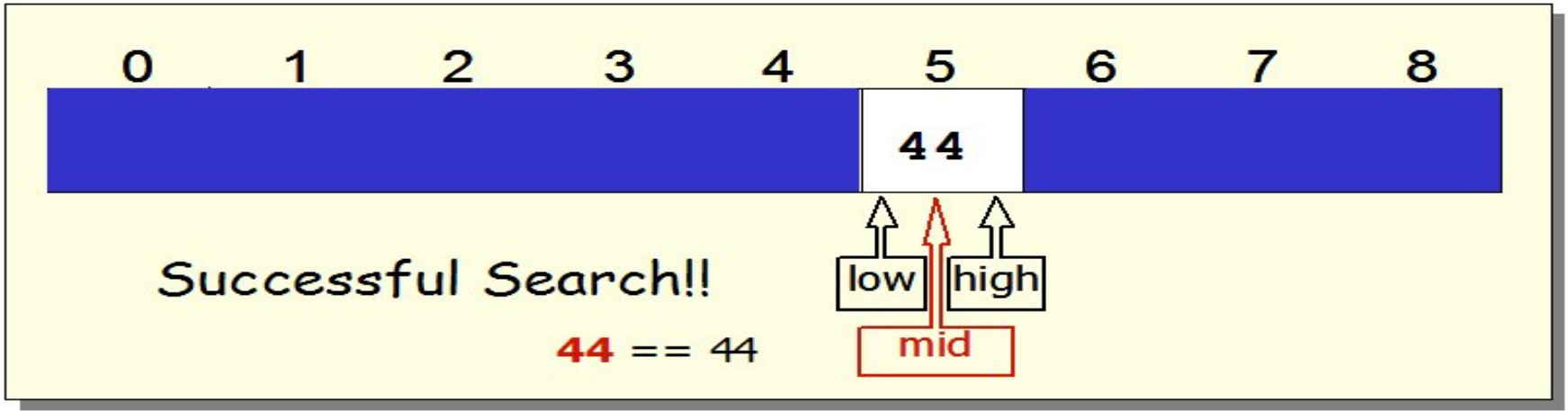


二分查找示例

	low	high	mid
#1	0	8	4
#2	5	8	6
#3	5	5	5

search(44)

$$mid = \left\lfloor \frac{low + high}{2} \right\rfloor$$



二分查找实现

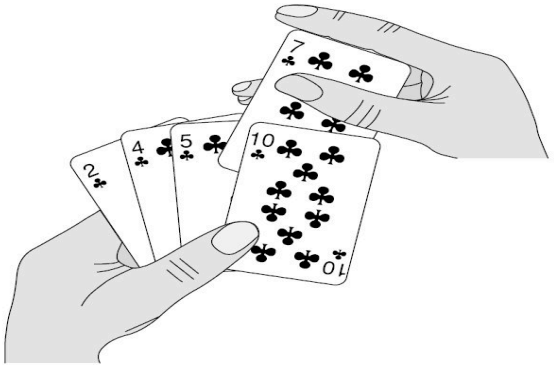

```
def bi_search(lst, v):
    low = 0
    up = len(lst) - 1

    while low <= up:
        mid = (low + up) / 2
        if lst[mid] < v:
            low = mid + 1
        elif lst[mid] == v:
            return mid
        else:
            up = mid - 1

    return -1
```

- 时间复杂度：$O(\log_2 n)$

排序（Sort）

- **将一个无序列表，按照某一顺序（由小到大或由大到小）排列**
- **是计算机科学中常见而且重要的任务**
- **有许多不同的算法，我们只介绍两种最简单直观的**
  - 选择排序（selection sort）
  - 冒泡排序（bubble sort）

选择排序（版本1）



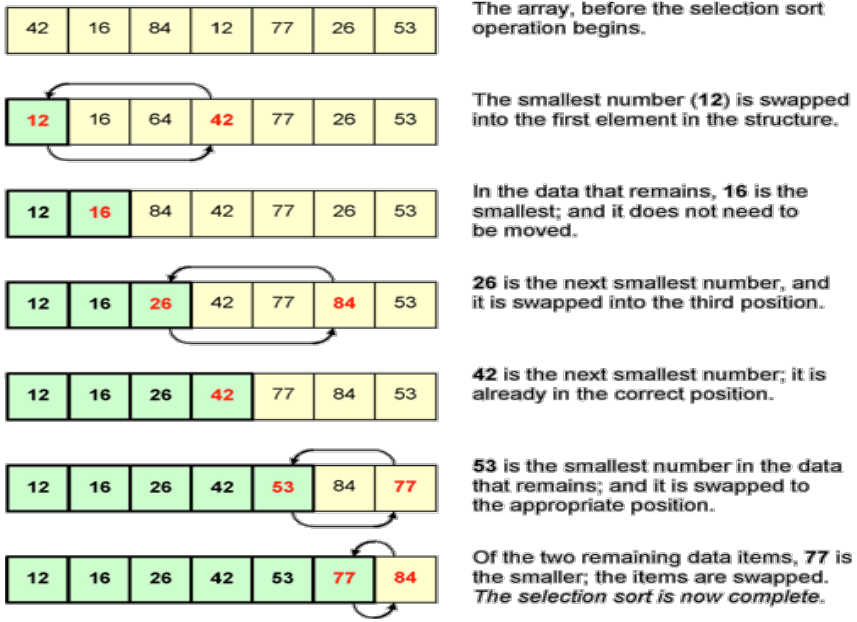
1. 找到最小的元素
2. 删除它，然后将其插入相应的位置
3. 对于剩余的元素，重复步骤 1 和 2

```
def selection_sort(lst):
    for i in range(len(lst) - 1):
        min_idx = i
        for j in range(i + 1, len(lst)):
            if lst[j] < lst[min_idx]:
                min_idx = j
        lst.insert(i, lst.pop(min_idx))

lst = [42, 16, 84, 12, 77, 26, 53]
print lst
selection_sort(lst)
print lst
```

选择排序（版本2）

1. 找到最小的元素
2. 和第一个元素**交换**
3. 对于剩余的元素，重复步骤 1 和 2

```
def selection_sort(lst):
    for i in range(len(lst) - 1):
        min_idx = i
        for j in range(i + 1, len(lst)):
            if lst[j] < lst[min_idx]:
                min_idx = j
        swap(lst, min_idx, i)

lst = [42, 16, 84, 12, 77, 26, 53]
print lst
selection_sort(lst)
print lst
```

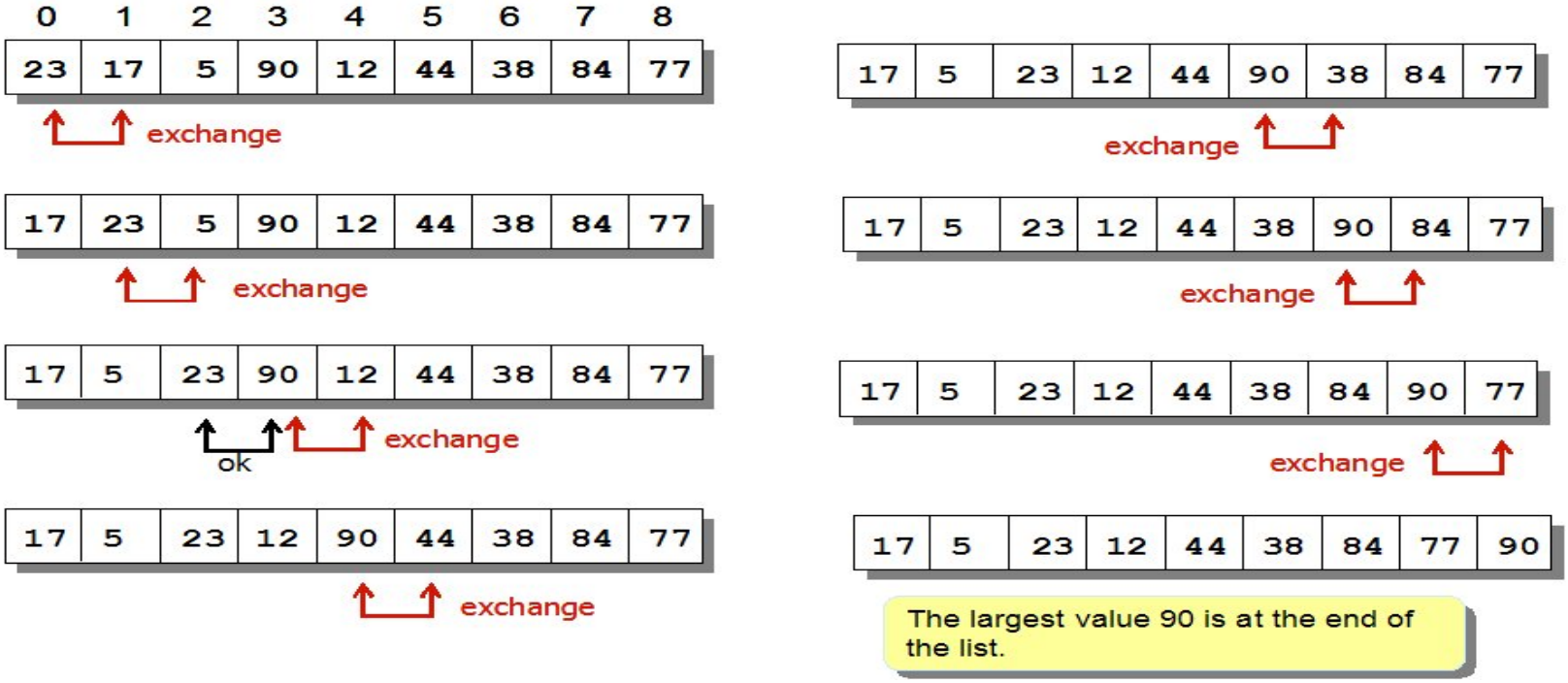


选择排序的时间复杂度

- **所需步骤**
  - 找到最小的元素需要 n 步
  - 找到剩余的最小元素需要 n-1 步
  - ……
- **总运行时间**
  - $n + (n - 1) + \ldots + 2 + 1$
- **时间复杂度**
  - $O(n^2)$

冒泡排序

- 与选择排序类似，但是每次遍历不止交换一次
- 每次遍历，将最大的值排在最后



冒泡排序

- **提示：一旦列表排好序，算法可以停止**

```
def bubble_sort(lst):
    exchanged = True
    top = len(lst) - 1
    while exchanged:
        exchanged = False
        for i in range(top):
            if lst[i] > lst[i+1]:
                swap(lst, i, i+1)
                exchanged = True
        top -= 1
```

- **时间复杂度**
  - $O(n^2)$，与选择排序相同，但是通常速度更快，为什么？

内建排序函数

- **sorted() 函数**

```
>>> a = [5, 2, 3, 1, 4]
>>> sorted(a)
[1, 2, 3, 4, 5]
```

- **list.sort() 方法**

```
>>> a = [5, 2, 3, 1, 4]
>>> a.sort()
>>> a
[1, 2, 3, 4, 5]
```

- **算法：quicksort**
  - 时间复杂度：O($n$log$n$)

嵌套列表

- 如何存储如下的数据表？

	0	1	2	3
0	5	4	7	3
1	4	8	9	7
2	5	1	2	3

- 列表的列表
  - x = [[5,4,7,3], [4,8,9,7], [5,1,2,3]]
  - 访问第三行、第二列的元素： x[2][1]
  - 请问：len(x) 的结果是？
  - 如何获取列数？

嵌套列表示例

- 计算所有学生的平均分

```
students = [['Zhang', 84],
            ['Wang',  77],
            ['Li',    100],
            ['Zhao',  53]]
s = 0

for student in students:
    s += student[1]

print float(s) / len(students)
```

列表解析或推导（List Comprehension）

- 一种由原列表创建新列表的简洁方法
  - [表达式 for 变量 in 列表 if 条件]
- 如生成值为 $\{x^2 : x \in \{1 \ldots 9\}\}$ 的列表

```
lst = []

for x in range(1, 10):
    lst.append(x**2)

print lst
```

- 列表解析

```
lst = [x**2 for x in range(1, 10)]
```

列表解析示例

- **列表推导实现求平均分**
  - sum([x[1] for x in students]) / len(students)
- **使用列表解析对所输入数字 x 的因数求和**
  - 如：如果输入 6，应该显示12，即 1 + 2 + 3 + 6 = 12
  - sum([i for i in range(1, x + 1) if x % i == 0])

嵌套列表示例

- 按照成绩由高到低排序

```
students = [['Zhang', 84],
            ['Wang',  77],
            ['Li',    100],
            ['Zhao',  53]]

def f(a):
    return a[1]

students.sort(key = f, reverse = True)

print students
```

lambda 函数

- 定义匿名函数

```
>>> def f (x): return x**2
...
>>> print f(8)
64
>>>
>>> g = lambda x: x**2
>>>
>>> print g(8)
64
```

- lambda 函数实现按成绩排序

```
students.sort(key = lambda x: x[1], reverse=True)
```

元组

什么是元组（Tuple）？

- **元组即不可变（immutable）列表**
  - 除了可改变列表内容的方法外，其它方法均适用于元组
  - 因此，索引、切片、len( )、print等均可用
  - 但是，append、extend、del等不可用
- **使用，（可以加（））创建元组**

```
my_tuple = 1, 'a', 3.14, True
my_tuple = (1, 'a', 3.14, True)
```

- **为什么需要元组？**
  - 保证列表内容不被修改

元组赋值

- 交换两个值

```
temp = a
a = b
b = temp
```

- 或者

```
a, b = b, a
```

- 如切分一个邮件地址

```
name, domain = 'car@hit.edu.cn'.split('@')
```

函数和元组

- **函数只能有一个返回值**
  - 但是该值可以是一组值，如返回一个元组
- **如同时返回列表中的最大和最小值**

```
def max_min(lst):
    max = min = lst[0]
    for i in lst:
        if i > max:
            max = i
        if i < min:
            min = i
    return max, min
```

DSU模式

- **Decorate, Sort and Undecorate (DSU) 模式**
  - 装饰、排序和反装饰
- **如根据单词的长度对一个单词列表进行排序**

```
def sort_by_length(words):
    # decorate
    t = []
    for word in words:
        t.append((len(word), word))

    # sort
    t.sort(reverse = True)

    # undecorate
    res = []
    for length, word in t:
        res.append(word)

    return res
```

```
words.sort(key = lambda x: len(x), reverse = True)
```

字典与集合

车万翔

哈尔滨工业大学

什么是字典（Dictionary）？

- 一系列“键-值（key-value）”对
- 通过“键”查找对应的“值”
- 类似纸质字典，通过单词索引表找到其相应的定义
  - C++: map、Java: HashTable or HashMap
- 例如：电话本

姓名（键）	电话号码（值）
John	86411234
Bob	86419453
Mike	86412387
……	……

字典的使用

- **创建字典**
  - 使用 { } 创建字典
  - 使用 : 指明 键:值 对
    - my_dict = {'John': 86411234, 'Bob': 86419453,'Mike': 86412387}
  - 键必须是不可变的且不重复，值可以是任意类型
- **访问字典**
  - 使用 [ ] 运算符，键作为索引
    - print my_dict['Bob']
    - print my_dict['Tom'] #WRONG!
    - 增加一个新的对
      - my_dict['Tom'] = 86417639

字典运算符和方法

- **len(my_dict)**
  - 字典中键-值对的数量
- **key in my_dict**
  - 快速判断 key 是否为字典中的键：O(1)
  - 等价于 my_dict.has_key(key)
- **for key in my_dict:**
  - 枚举字典中的键，注：键是无序的
- **更多的方法**
  - my_dict.items() – 全部的键-值对
  - my_dict.keys() – 全部的键
  - my_dict.values() – 全部的值
  - my_dict.clear() – 清空字典

示例：字母计数

- **读取一个字符串，计算每个字母出现的个数**
- **方案一**
  - 生成 26 个变量，代表每个字母出现的个数
- **方案二**
  - 生成具有 26 个元素的列表，将每个字母转化为相应的索引值，如 a ➔ 0，b ➔ 1，…

```
count = [0] * 26
for i in 'abcdad':
    count[ord(i) - 97] += 1
```

- **方案三**
  - 生成一个字典，字母做键，对应出现的次数做值

```
count = {}
for i in 'abcdad':
    if i in count:
        count[i] += 1
    else:
        count[i] = 0
```

示例：单词计数

- **读取小说"emma.txt"，打印前 10 个最常见单词**
  - 是否还能直观的将每个单词转化为相应的数字？

```
f = open('emma.txt')
word_freq = {}
for line in f:
    words = line.split() # split a line by blanks
    for word in words:
        if word in word_freq:
            word_freq[word] += 1
        else:
            word_freq[word] = 1

word_freq_lst = []
for word, freq in word_freq.items():
    word_freq_lst.append((freq, word))

word_freq_lst.sort(reverse = True)

for freq, word in word_freq_lst[:10]:
    print word, freq
```

示例：翻转字典

- **生成一个新字典，其键为原字典的值，值为原字典的键**
  - 同一个值，可能对应多个键，需要用列表存储

```
def invert_dict(d):
    inverse = {}
    for key in d:
        val = d[key]
        if val in inverse:
            inverse[val].append(key)
        else:
            inverse[val] = [key]
    return inverse
```

集合（Set）

- **集合**
  - 无序不重复元素（键）集
  - 和字典类似，但是无“值”
- **创建**
  - x = set()
  - x = {key1, key2, ...}
- **添加和删除**
  - x.add('body')
  - x.remove('body')
- **集合的运算符**

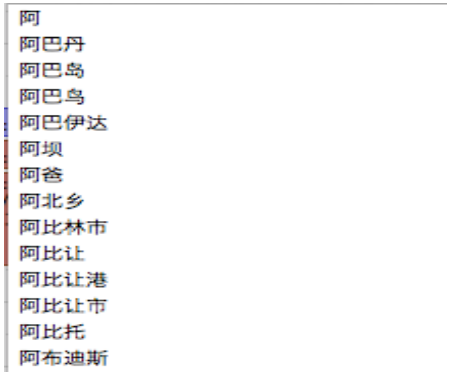
运算符	含义
-	差集
&	交集
|	并集
!=	不等于
==	等于
in	成员
for key in set	枚举

示例：中文分词

- **我爱北京天安门。➔ 我/爱/北京/天安门/。**
- **算法：正向最大匹配**
  - 从左到右取尽可能长的词
  - 如：研究生命的起源 ➔研究生/命/的/起源
    - “研究生”是词，且比“研究”更长

示例：中文分词

- 加载词典：lexicon.txt




```
def load_dic(filename):
    f = open(filename)
    word_dic = set()
    max_length = 1
    for line in f:
        word = unicode(line.strip(), 'utf-8')
        word_dic.add(word)
        if len(word) > max_length:
            max_length = len(word)
    return max_length, word_dic
```

示例：中文分词

正向最大匹配分词

```
def fmm_word_seg(sentence, word_dic, max_length):
    begin = 0
    words = []
    sentence = unicode(sentence, 'utf-8')

    while begin < len(sentence):
        for end in range(min(beg + max_len, len(sent))
                                        , beg, -1):
            word = sentence[begin:end]
            if word in word_dic or end == begin + 1:
                words.append(word)
                break
        begin = end
    return words
max_len, word_dic = load_dic('lexicon.dic')
words = fmm_word_seg(raw_input(), word_dic, max_len)
for word in words:
    print word
```

数据结构对比

	string	list	tuple	set	dict
Mutable					
Sequential					
Sortable					
Slicable					
Index/key type					
Item/value type					
Search complexity					

数据结构对比

	string	list	tuple	set	dict
Mutable	No	Yes	No	Yes	Yes
Sequential	Yes	Yes	Yes	No	No
Sortable	No	Yes	No	No	No
Slicable	Yes	Yes	Yes	No	No
Index/key type	Int	Int	Int	Immut	Immut
Item/value type	Char	Any	Any	No	Any
Search complexity	O(n)	O(n)	O(n)	O(1)	O(1)