Python程序设计

第三讲 程序设计基础
算法和结构化技术

张 华
WHU

算法和结构化技术

- 程序设计的概念
- 算法及其表示
- 结构化技术

程序设计

- 任何计算问题的解决方案包括
  - 基于一组数据，按照特定顺序去执行一系列操作。
- 程序 = 数据结构 + 算法
- 数据结构是程序中数据的类型和组织形式。
- 算法（Algorithm）
  - 为解决某个特定的问题而采用的确定且有限的步骤。
  - 例如，“早晨上学准备算法”：

    起床-穿衣-洗漱-吃早餐-上学

算法是程序的灵魂。

Algorithm is the spirit of a program.

算法的表示

- 算法的表示方法
  - 自然语言
  - 伪代码
  - 流程图
  - 计算机语言
- 举例
  - 问题
    - 计算1+2+3+…+100
  - 数据结构
    - 定义整型变量 i 保存加数
    - 定义整型变量 sum 保存每次累加的和

算法的自然语言表示

- 问题
  - 计算1+2+3+…+100
- 算法的自然语言表示

步骤1：i=1，sum=0。

步骤2：如果i小于或等于100，顺序执行步骤3；否则，执行步骤5。

步骤3： sum加上i，相加后的值仍放在sum中，即：sum = sum+i。

步骤4：使i的值增一得到下一个加数，即i=i+1; 执行步骤2。

步骤5：变量sum中的值就是要得到的结果；输出结果，算法结束。

算法的伪代码表示

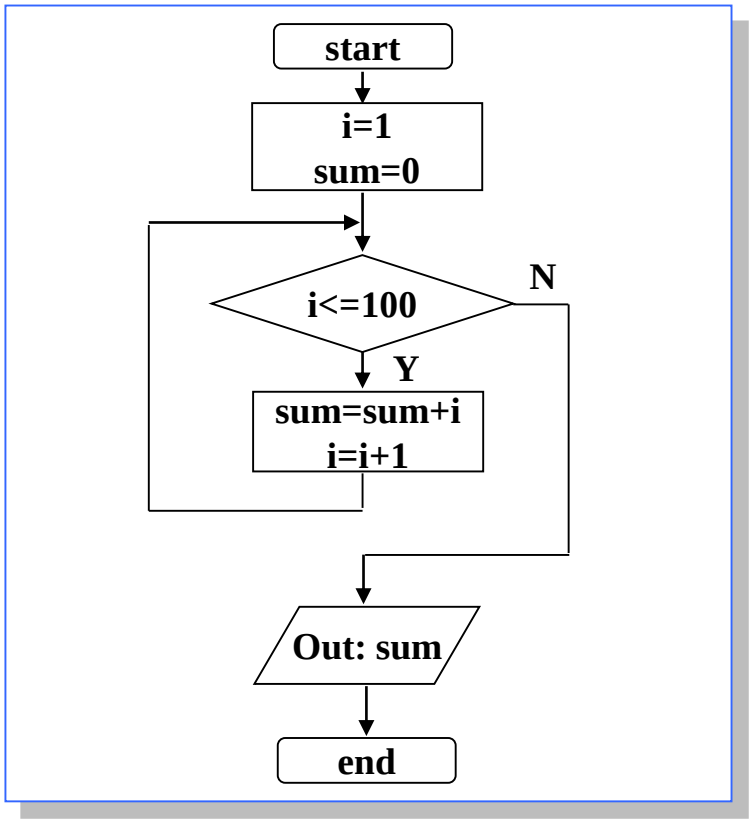
- 问题
  - 计算1+2+3+…+100
- 伪代码表示

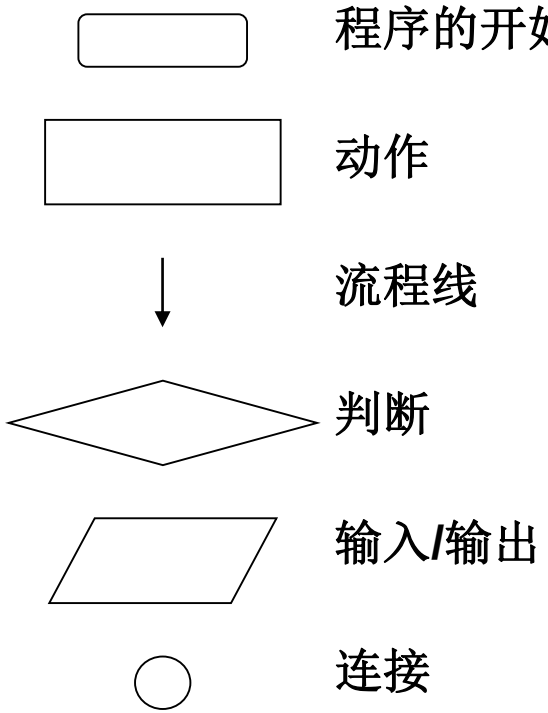
```
i=1
sum=0

while i<=100 do
     sum=sum+i
     i=i+1
end while

print sum
```

- 非正式的语言
- 与日常用语类似
- 帮助程序员在写程序之前“设想出”程序
  - 很容易被转换成正式的程序
  - 一般只包括可执行语句

算法的流程图表示

流程图表示





算法的计算机语言表示

Python语言表示（实现算法）

```
i=1                                          //数据结构
sum=0

while (i<=100):                              //累加100次
   sum=sum+i
   i=i+1

print("1+2+3+...+100 =",sum);                //输出结果
```

程序的控制结构

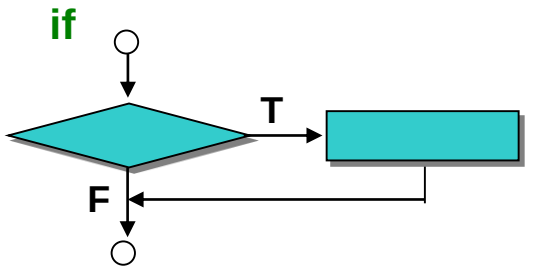
- **程序控制**
  - 计算机程序中语句的执行顺序。
- **顺序执行**
  - 程序中的语句按照它们的书写顺序一句接一句地执行。
  - 是计算机执行程序的自然方式。
- **控制转移**
  - 把待执行的下一条语句指定为不是书写顺序中的下一条语句。
  - goto语句是最直接的方式，但是过多使用会带来很多问题。
    - 程序结构不清晰、可读性差、不利于维护。

流程控制语句

构造选择结构的分支语句

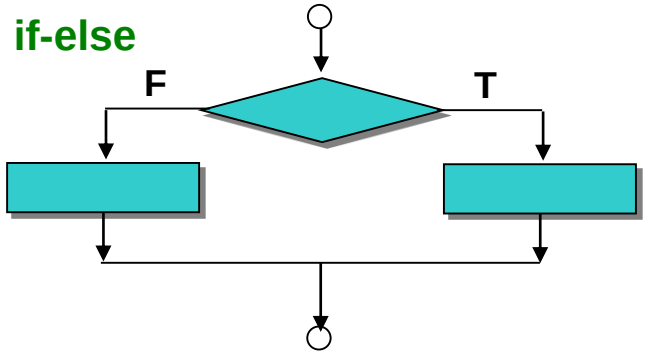
if

```
if <条件表达式>:
    <if子句>
```



if-else

```
if <条件表达式>:
    <if子句>
else:
    <else子句>
```



流程控制语句

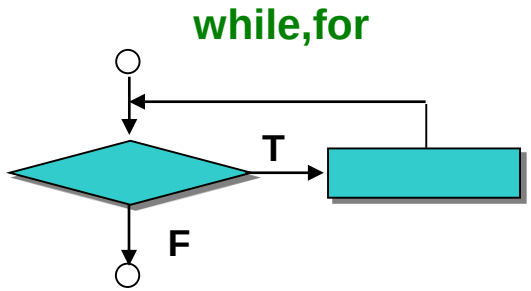
构造循环结构的循环语句

while

```
while <条件表达式>:
    <循环体语句>
```

for

```
for [表达式1] in [表达式2]:
    <循环体语句>
```

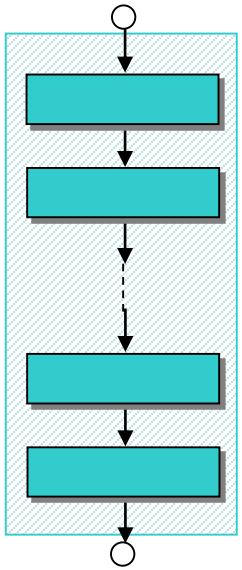


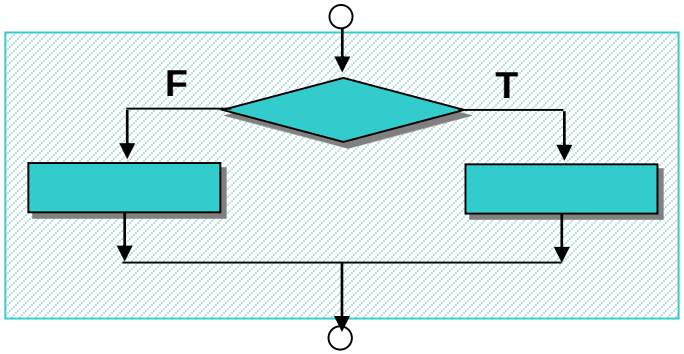
结构化技术

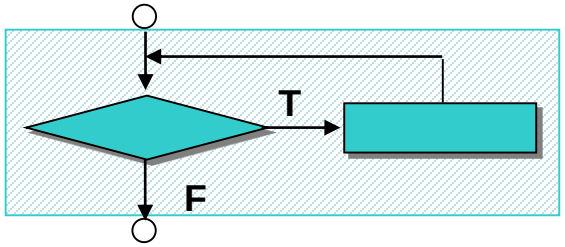
- 程序=数据结构+算法+语言工具+程序设计方法
- 结构化技术（1970s）
  - 程序是由三种基本结构构成的。
- 三种基本结构
  - 顺序结构（Sequence structures）
    - Python语言程序中的语句按书写顺序执行
  - 选择结构（Selection structures）
    - Python语言有两种: if, if-else
  - 循环结构（Repetition structures）
    - Python语言有两种: while, for

结构化程序

- 3种控制结构都是单入/单出的。

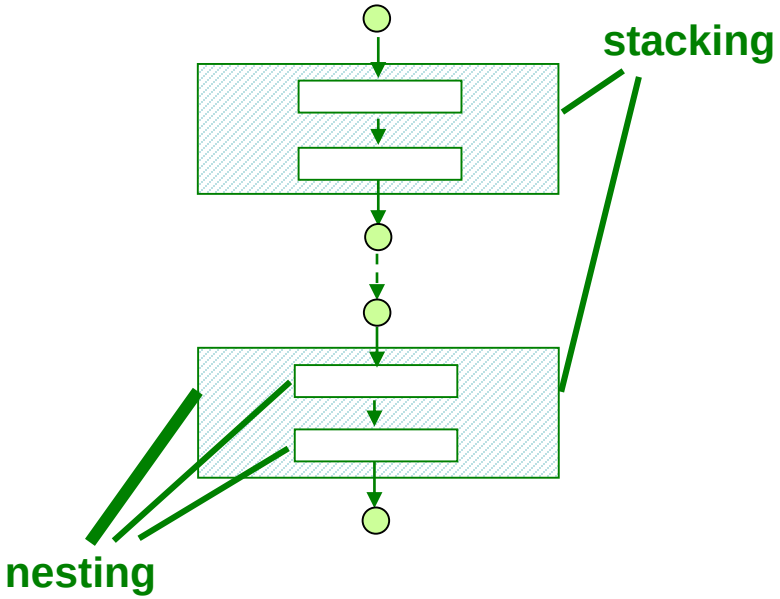






结构化程序

- 结构化程序由3种结构通过以下方式组合而成：
  - 堆叠（stacking）
  - 嵌套（nesting）

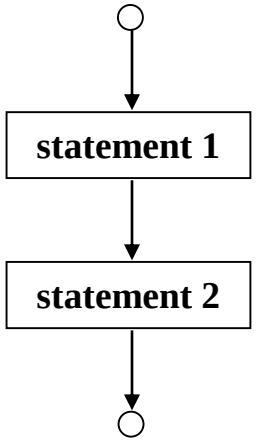


sequence structure

顺序结构

顺序结构

- 语句按书写的顺序执行。

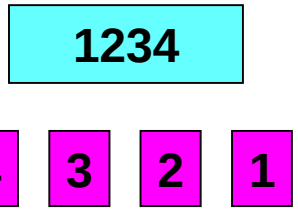


Sequence Structure

程序设计举例

问题

- 输入一个四位的正整数，反序输出该四位数的四个数字字符。



分析与设计

- 用一个无符号整型变量number保存输入的四位正整数。
- 依次分解出个位数字、十位数字、百位数字和千位数字，并分别放到字符变量c1、c2、c3和c4中。
- 怎么分解？
  - 利用除、模等运算。
- 顺序输出变量c1、c2、c3和c4中的字符。

程序设计举例

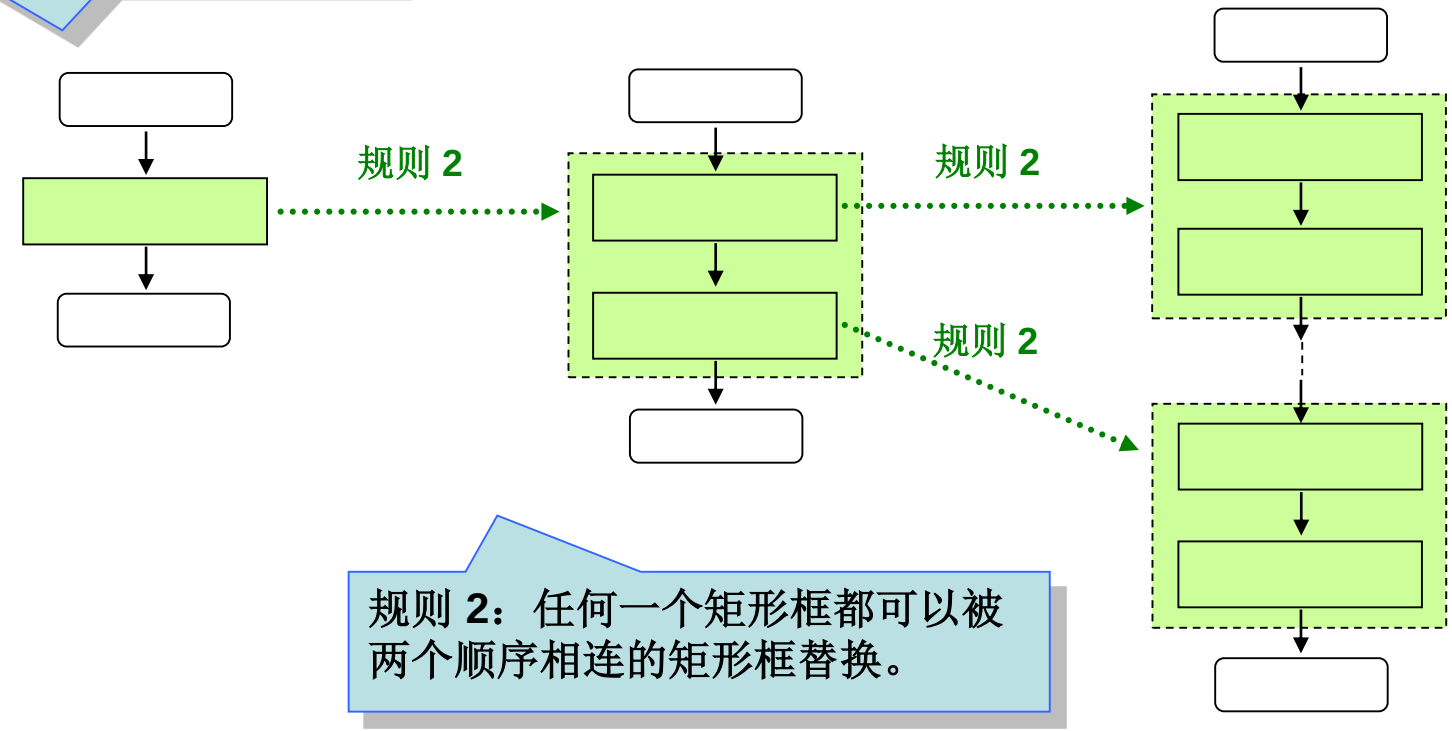
源程序

```
number = int(input('输入一个四位正整数：'))

c1 = str(number%10)
c2 = str(number//10%10)
c3 = str(number//100%10)
c4 = str(number//1000)

print('个十百千位上的数字分别是：',c1,c2,c3,c4);
```

结构化技术

结构化程序设计的规则

规则 1：从一个最简单的流程图开始。



规则 2：任何一个矩形框都可以被两个顺序相连的矩形框替换。

结构化技术

结构化程序设计的规则

规则 3：任何一个矩形框都可以被任何控制结构替换。