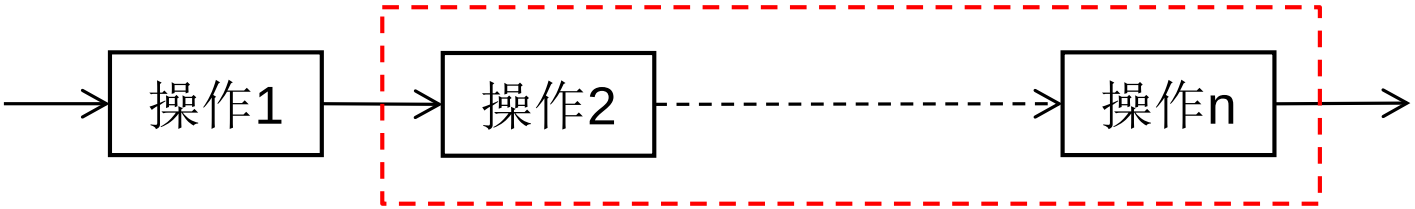
程序初步-4

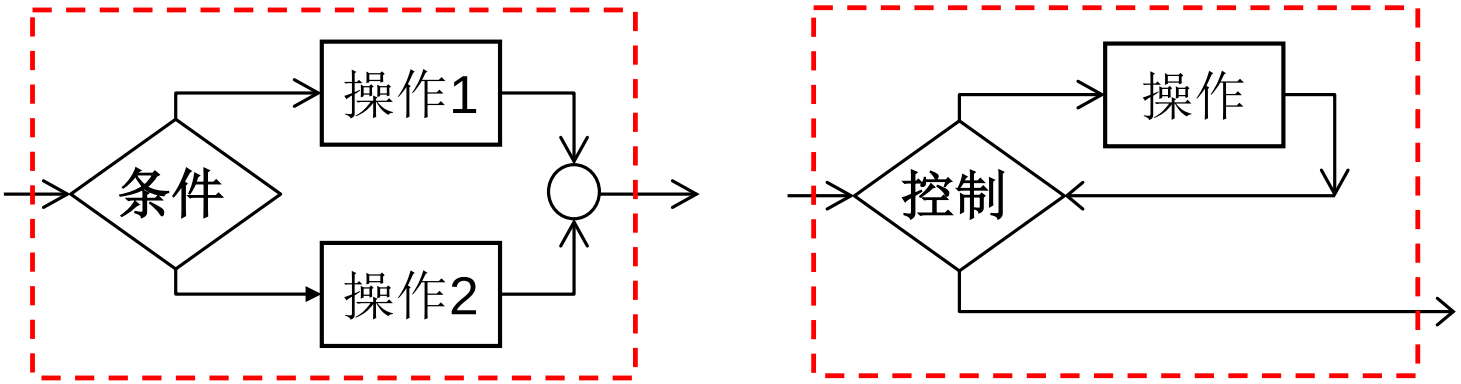
- 程序和抽象
- 函数和计算
- 自定义函数
- 函数定义和函数调用
- 递归函数定义
- 一些 **Python** 机制

操作、控制和抽象

- 顺序执行模式（用语句组描述）：



- 选择和重复执行模式（用 if 和循环结构描述）：



- 注意：一段计算也可以抽象地看作一个组合操作

  这种看法（和做法）很有价值，能帮助分解复杂的程序和计算

值的抽象和计算的抽象

- 变量是对象的名字，是一种抽象
  - 把值（对象）赋给变量，后面可以方便地使用已有计算结果
  - 一次计算，任意多次使用
- 希望重复使用的不仅是算出的值
  - 还可能希望重复使用一段代码（一段计算）
  - 包括对不同的数据做同样计算
  - 能不能做到写一次代码，任意多次重复使用？
- 由此可见，需要代码片段的抽象
  - 将一段代码定义为一个计算体并命名，就是定义函数抽象
  - 定义好的函数可以通过名字，以简单的方式（重复）使用

函数

- 一个函数完成某种特定计算，具体计算可能复杂，但使用方便
  - **math** 库函数完成一些数学函数计算
  - 通过函数调用，提供具体参数就能得到所需结果
  - 在一个程序（一个表达式/一个语句）里，可以任意多次调用同一个数学函数，或调用多个不同的数学函数
- 函数的特点：一个定义，任意使用
  - 函数库开发者做一个函数，所有用 **Python** 的人都能用
- **Python** 提供了标准库，其中有许多有用功能
  - **Python** 还有许多程序库，提供了许多有用功能
  - 有关情况可查库文档，搜索 **Python** 主页或互联网

自定义函数

- 系统提供多少函数，都不可能满足所有人的需要
  - **Python** 允许写程序的人自己定义函数，称为自定义函数
  - 网上很多公开资源就是别人用 **Python** 开发的模块，其中一部分东西就是各种各样的函数
- 自定义函数有什么价值？
  - 考虑前面求三角形面积的程序，执行一次完成一个特定计算
  - 如果有函数 **triangle(a, b, c)** 计算三角形面积，就可以写

```
x = triangle(6, 8, 11)
y = triangle(3, 5, 6) * 3   # 三角柱体积
z = triangle(13, 15, 16) - triangle(6, 8, 11) # 空心三角形
# 等等
```

函数定义

- 一个函数定义包装起一段代码，建立一个函数抽象，并为其命名
- 语法：

  def 函数名 **(**参数列表**) :**

  语句块

  参数列表：逗号分隔的0个或多个名字，称为形式参数（形参）

  语句块称为函数体，前面部分称为函数头部

  函数定义看作一种结构语句，语义（执行）：根据形式参数表和函数体建立一个函数对象，并将其约束到给定函数名

  函数体里的 **return** 有特殊作用，用于描述函数的返回值

- 自定义函数同样通过函数调用方式使用。例：三角形面积的函数
  - 调用这个函数，就是执行其包装的语句，完成一次计算

函数的定义和调用

- 函数里的 return 语句。两种形式：

  return 表达式

  return

  语义：return 的执行导致当前函数立即结束。如果有表达式部分，就以其值作为函数的返回值；否则函数值是 None

- 函数调用时的执行：

  1. 从左到右逐个求值实际参数（表达式），赋给对应形参
  2. 执行函数体里的语句
  3. 遇 return 语句执行结束，求值其表达式作为返回值，没有表达式时返回值为 None
  4. 所有语句都执行完时函数也结束，返回值为 None

定义函数

- 例：求三角形面积函数的改进和变形
- 例：计算并输出三角形面积的函数（无返回值的函数）
- 例：输出摄氏到华氏转换表的函数
- 函数定义，是建立计算抽象（过程抽象）的机制
  - 定义好的函数可以像库函数一样很方便地调用
  - 一次定义，任意多次使用
  - 把任意复杂的一段段程序代码抽象为一个个函数

    把复杂的程序分解为一种层次结构，后面会看到更多这方面的例子
  - 定义包含许多函数的模块，以后使用，有价值的模块还可以提供给别人使用

基本语句和复合语句

- 基本语句（前面讨论过的，描述基本操作）：
  - 赋值语句
  - 表达式语句（只有一个表达式的程序行，如 print 调用）
  - break，continue，return 语句
  - import 语句
- if、for、while 是复合语句
  - 其成分中包含（一个/两个/多个）语句
  - 其作用就是控制其成分语句的执行
- def 语句的执行建立一个函数（在 Python 里也是对象，称为函数对象），将其约束于给定函数名。此后通过这个函数名就可以调用这个函数。函数体里可以写任何语句，包括调用其他函数

递归定义的函数

- 有一种函数定义比较特殊，就是在函数体里调用自身
  - 利用自己完成自己的一部分工作
  - 这种定义称为函数的递归定义，定义的函数称为递归函数

- 考虑求阶乘的函数

  阶乘：
  $$n! = \begin{cases} n \times (n-1)! & n > 0 \\ 1 & n = 0 \end{cases}$$

  这是个分情况定义（数学定义），而且是一个“递归的定义”，n 的阶乘是基于 n-1 的阶乘定义的

  Python 允许在函数定义的函数体里调用被定义的函数（允许以递归的方式定义函数），可以直接把上面的数学定义翻译为 Python 的函数定义

递归程序

- 递归定义的阶乘函数，看起来是一个简单分支型程序
  - 但它在一个分支里调用自己，每次递归调用执行函数里的一条执行路径（总体而言可能多次重复执行函数里的语句）
  - 运行中总的执行路径长度由递归调用这个函数的次数确定（由调用的参数决定，如阶乘函数）
- 理论结果：
  - 由基本语句、顺序组合、条件语句、递归、函数定义构成的语言已经足够强大，足以描述所有可能的计算
  - 基本语句、顺序组合和循环构成的语言也足够强大
- 实际情况：

  在实用的编程语言里，还需要引进另一些机制（控制机制和数据机制），以方便实际的程序开发

求幂函数

- 求 x^n 需要做多少次乘法？

  简单想法是需要做 n 次或 n-1 次

- 但考虑 $x^8 = (x^2)^4 = ((x^2)^2)^2$，只需做 3 次乘法，不需要做 7 次

  似乎有办法减少做乘法的次数

- 一般的认识：
  - 特殊情况：$x^0 = 1$，$x^1 = x$
  - 一般情况：$x^{2n} = (x^2)^n$，$x^{2n+1} = x^{2n} \times x$
- 利用这种看法，可以定义求幂函数（见代码文件）
  - 可以递归定义
  - 也可以写循环定义

assert，pass 语句

- **assert（断言）是一种特殊语句**

  assert 条件

  assert 条件, 表达式

  顺序求值条件，真时继续执行，不真时报错 **AssertionError**

  对第二种形式，表达式作为 **AssertionError** 的参数送出

  - 描述程序执行中必须满足的条件，辅助程序调试。与 **if** 语句不同，适用于描述程序（函数）的必要条件
  - 例：阶乘程序

- **pass** 语句

  pass

  什么也不做，立即结束。用于填补语法结构

Python 的若干问题

- 注释
  - 形式：# 符号开始的字符序列，直至本行结束
  - 语义：没有语义。注释是供人阅读的正文（解释器将其忽略），通常用于提供有关程序代码的说明信息
- 条件表达式也是一种表达式，它计算出一个值
  - 形式：

    表达式1 **if** 条件 **else** 表达式2
  - 语义：

    如果条件为真就以 表达式1 的值作为整个表达式的值，否则就以 表达式2 的值作为整个表达式的值

注意条件表达式与条件语句的不同。语句没有值，执行时产生“效果”，表达式的计算就是求值（举阶乘函数的例）

Python 对程序行的处理

- **Python 系统（解释器）按行读入和处理**
  - 默认一个语句（或表达式）中间不能换行（看例子）
  - 如果一行中的括号没配对，就认为下一行是本行的继续
  - 可以在一行最后用一个 \ 说明下一行是本行的继续

  一个逻辑行可能由一个或几个物理行构成
- **Python 执行环境的基本循环：**
  - 读入一个逻辑程序行，或者一个跨几行的复杂结构，如 **def, for, while** 等
  - 处理读入的结构，输出计算结果（或者报错）

  处理模块中代码的规则也一样，只是不显示求值结果

关键字

- 已经见过的关键字（19/33）

  and as assert break class continue def del

  else except elif False finally for from global

  if in import is lambda None nonlocal not or

  pass return raise True try while with yield

- 本节内容：
  - 函数定义和使用（函数调用），return 语句，意义
  - 函数的递归定义
  - assert 和 pass 语句
  - 条件表达式