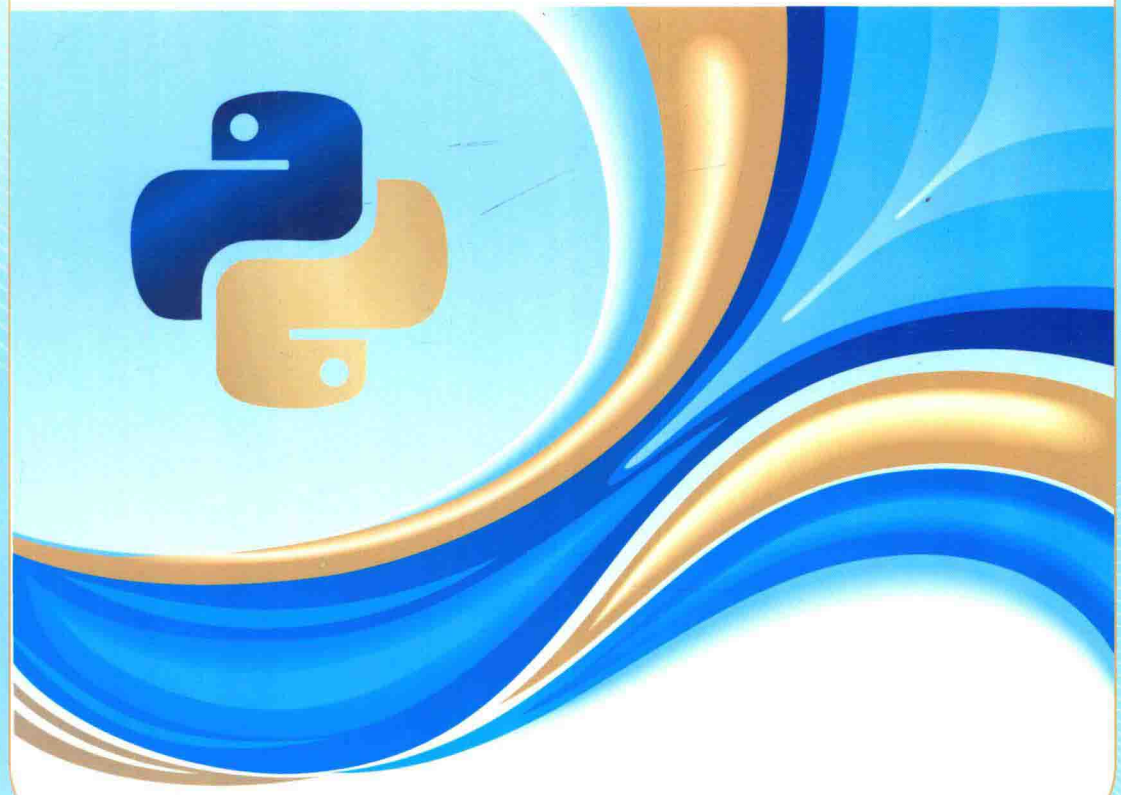
“十二五”普通高等教育规划教材

Python 程序设计教程

刘卫国 主编



北京邮电大学出版社
www.buptpress.com

“十二五”普通高等教育规划教材

Python 程序设计教程

主　编　刘卫国
副主编　陈忠文　夏　耘

北京邮电大学出版社
·北京·

内 容 简 介

Python 作为一门很“年轻”的语言，具有优雅、清晰、简洁的语法特点，能使初学者从语法细节中摆脱出来，而专注于解决问题的方法、分析程序本身的逻辑和算法。本书以 Python 作为实现工具，介绍程序设计的基础知识与基本技术。全书以实际问题的求解过程为向导，突出从问题到算法，再到程序的一种思维过程，强调计算机求解问题的思路引导与程序设计思维方式的训练，重点放在程序设计的思想与方法上。全书主要内容有程序设计概述、程序的数据描述、顺序结构程序设计、选择结构程序设计、循环结构程序设计、字符串处理、列表与元组、字典与集合、函数、文件操作、面向对象程序设计、异常处理、图形处理、图形用户界面设计和综合程序设计等。

本书可作为高等学校计算机程序设计课程的教材，也可供社会各类工程技术与科研人员阅读参考。

图书在版编目(CIP)数据

Python 程序设计教程/刘卫国主编. -- 北京：北京邮电大学出版社，2016.3(2017.6 重印)
ISBN 978-7-5635-4558-2

Ⅰ.①P… Ⅱ.①刘… Ⅲ.①软件工具—程序设计—教材 Ⅳ.①TP311.56

中国版本图书馆 CIP 数据核字(2015)第 254446 号

书　　名	Python 程序设计教程
主　　编	刘卫国
责任编辑	唐咸荣
出版发行	北京邮电大学出版社
社　　址	北京市海淀区西土城路 10 号(100876)
电话传真	010-82333010　62282185(发行部)　010-82333009　62283578(传真)
网　　址	www.buptpress3.com
电子信箱	ctrd@buptpress.com
经　　销	各地新华书店
印　　刷	北京泽宇印刷有限公司
开　　本	787 mm×1 092 mm　1/16
印　　张	22
字　　数	548 千字
版　　次	2016 年 3 月第 1 版　2017 年 6 月第 2 次印刷

ISBN 978-7-5635-4558-2　　定价：45.00 元

前　言

计算机程序设计基础是一门非常重要的计算机课程，其目的是介绍程序设计的基础知识，使学生掌握高级语言程序设计的基本思想和方法，理解利用计算机分析和解决实际问题的基本过程和思维规律，从而更好地培养学生的创新能力，为未来应用计算机进行科学研究与工程应用奠定坚实的基础。

计算思维不仅反映了计算的原理，更重要的是体现了基于计算机的问题求解思路与方法。就课程性质而言，计算机程序设计基础最能够体现问题求解方法，是理解计算机工作过程的有效途径，也是计算思维能力培养的重要载体。因此，计算机程序设计课程的重要性不仅体现在一般意义上的程序设计能力的培养，而且体现在引导学生实现问题求解的思维方式的转换，即学生计算思维能力的培养。当然，要实现计算思维能力的培养不是一件容易的事，这也是程序设计教学改革的重要切入点。本教材正是按照这种改革理念，以实际问题的求解过程为向导，介绍程序设计的基础知识与基本技术，重点放在程序设计的思路与方法上。

Python 作为一门很“年轻”的语言，具有优雅、清晰、简洁的语法特点，能使初学者从语法细节中摆脱出来，而专注于解决问题的方法、分析程序本身的逻辑和算法。当然，任何一种高级语言有各自不同的诞生背景和应用目的，由此决定了其特性和语言功能。现代程序设计语言具有相互吸收各自优点的趋势，功能趋同。程序设计语言的选择跟个人习惯、语言功能、历史积累的程序库等多种因素有关。从学习的角度讲，希望通过一种高级语言的学习，能掌握高级语言共同的语言特征，从而为进一步学习其他高级语言打下基础。

本书的基本定位是，将 Python 作为第一门程序设计语言，介绍 Python 语言的基本语法要素以及结构化程序设计的基本方法，培养读者良好的程序设计风格和运用 Python 语言解决实际问题的程序设计能力。全书以 Python 作为实现工具，介绍程序设计的基本思想和方法。全书主要内容有程序设计概述、程序的数据描述、顺序结构程序设计、选择结构程序设计、循环结构程序设计、字符串处理、列表与元组、字典与集合、函数、文件操作、面向对象程序设计、异常处理、图形处理、图形用户界面设计和综合程序设计等。书中内容不拘泥于语法细节，而以程序设计应用为导向，突出问题求解方法与思维能力训练。

目前 Python 有两个版本，即 2. x 和 3. x 版，这两个版本是不兼容的。虽然 Python 正朝着 3. x 版进化，但是由于历史原因，目前仍有大量的第三方库是用 2. x 版实现的，这些代码要修改后才能在 3. x 上运行。为了保证程序能用到这些优秀的第三方库，使用 2. x 版本是较好的选择。但从教学来说，3. x 版本体现了 Python 的发展，考虑到本书是学习程序设计的基础教材这一基本定位，所以，本书使用的 Python 版本是 3. x(下载安装时的最高版本是 Python 3. 4. 2)。

本书可作为高等学校计算机程序设计课程的教材，也可供社会各类工程技术与科研人员阅读参考。

本书第 1 章、第 10～15 章由刘卫国编写，第 2～9 章由蔡立燕编写，全书由刘卫国统稿，陈忠文、夏耘任副主编。此外，参与部分编写工作的还有蔡旭晖、童键、周欣然、刘胤宏等。

由于作者学识水平有限，书中难免存在疏漏或不妥之处，恳请广大读者批评指正。

作　者

2015 年 9 月

目　录

第1章

程序设计概述

计算机是在程序(program)控制下进行自动工作的,它解决的任何实际问题都得依赖于解决问题的程序,而要编写程序就要熟悉一种程序设计语言,掌握程序设计的基本方法,理解利用计算机分析和解决实际问题的基本过程和思维规律,为应用计算机进行科学研究与实际应用奠定坚实基础。Python是一种功能强大的解释型程序设计语言,在支持面向过程程序设计的同时还支持面向对象程序设计,既有灵活方便的数据结构,又有大量优秀的模块,语法清晰、简洁,而且可以在众多的平台上运行,非常适合于完成各种高层任务。目前,基于这种语言的相关技术的飞速发展,其用户数量的急剧扩大,以及其在软件开发中的广泛应用,本书以Python作为实现工具,介绍程序设计的基本思想和方法。

本章要点:

- 程序设计基础知识。
- Python语言的发展与特点。
- Python程序的运行环境与运行方式。
- Python程序的书写规则。

1.1 程序设计基础知识

在学习Python语言程序设计之前,需要了解一些程序设计的基础知识,包括程序设计的过程、算法的概念、算法的描述方法以及流行的程序设计方法。

1.1.1 程序与程序设计

从一般意义来说,程序是对解决某个实际问题的方法和步骤的描述,而从计算机角度来说,程序是用某种计算机能理解并执行的语言所描述的解决问题的方法和步骤。计算机执行程序所描述的方法和步骤,从而完成指定的功能。所以,程序就是供计算机执行后能完成特定功能的指令序列。

一个计算机程序主要描述两部分内容:一是描述问题的每个数据对象和数据对象之间的关系,二是描述对这些数据对象进行操作的规则。其中关于数据对象及数据对象之间的关系是数据结构(data structure)的内容,而操作规则是求解问题的算法(algorithm)。针对问题所

涉及的数据对象和要完成的操作，设计合理的数据结构可有效地简化算法，数据结构和算法是程序最主要的两个方面。著名的瑞士计算机科学家 N. Wirth 教授曾提出：

算法＋数据结构＝程序

程序设计的任务就是设计解决问题的方法和步骤(即设计算法)，并将解决问题的方法和步骤用程序设计语言来描述。什么叫程序设计？对于初学者来说，往往把程序设计简单地理解为只是编写一个程序，这是不全面的。程序设计反映了利用计算机解决问题的全过程，包含多方面的内容，而编写程序只是其中的一个方面。使用计算机解决实际问题，通常是先要对问题进行分析并建立数学模型，然后考虑数据的组织方式和算法，并用某一种程序设计语言编写程序，最后调试程序，使之运行后能产生预期的结果。这个过程称为程序设计(programming)。具体要经过以下 4 个基本步骤。

(1)分析问题，确定数学模型或方法

要用计算机解决实际问题，首先要对待解决的问题进行详细分析，弄清问题的需求，包括需要输入什么数据，要得到什么结果，最后应输出什么，即弄清要计算机“做什么”。然后把实际问题简化，用数学语言来描述它，这称为建立数学模型。建立数学模型后，需选择计算方法，即选择用计算机求解该数学模型的近似方法。不同的数学模型，往往要进行一定的近似处理。对于非数值计算问题则要考虑数据结构。

(2)设计算法，画出流程图

弄清楚要计算机“做什么”后，就要设计算法，明确要计算机“怎么做”。解决一个问题，可能有多种算法。这时，应该通过分析、比较，挑选一种最优的算法。算法设计后，要用流程图把算法形象地表示出来。

(3)选择编程工具，按算法编写程序

当为解决一个问题确定了算法后，还必须将该算法用程序设计语言编写成程序，这个过程称为编码(coding)。

(4)调试程序，分析输出结果

编写完成的程序，还必须在计算机上运行，排除程序可能的错误，直到得到正确结果为止。这个过程称为程序调试(debugging)。即使是经过调试的程序，在使用一段时间后，仍然会被发现尚有错误或不足之处。这就需要对程序做进一步的修改，使之更加完善。

解决实际问题时，应对问题的性质与要求进行深入分析，从而确定求解问题的数学模型或方法，接下来进行算法设计，并画出流程图。有了算法流程图，再来编写程序就容易多了。有些初学者，在没有把所要解决的问题分析清楚之前就急于编写程序，结果编程思路紊乱，很难得到预想的结果。

1.1.2 算法及其描述

计算机是通过执行人们所编制的程序来完成预定的任务。从广义上说，计算机按照程序所描述的算法对某种结构的数据进行加工处理。程序设计的实质是对实际问题选择一种好的数据结构，并设计一个好的算法，而好的算法在很大程度上取决于描述实际问题的数据结构。

1. 算法的概念

在日常生活中，人们做任何一件事情，都是按照一定规则、一步一步地进行的，这些解决问题的方法和步骤称为算法。例如，工厂生产一部机器，先把零件按一道道工序进行加工，然后，

把各种零件按一定法则组装起来，生产机器的工艺流程就是算法。

计算机解决问题的方法和步骤，就是计算机解题的算法。计算机用于解决数值计算，如科学计算中的数值积分、解线性方程组等的计算方法，就是数值计算的算法；用于解决非数值计算，如用于数据处理的排序、查找等方法，就是非数值计算的算法。要编写解决问题的程序，首先应设计算法，任何一个程序都依赖于特定的算法，有了算法，再来编写程序是容易的事情。

下面举两个简单例子，以说明计算机解题的算法。

例 1-1　求 $u=\dfrac{x-y}{x+y}$，其中 $x=\begin{cases}a^2+b^2 & a<b\\ a^2-b^2 & a\geqslant b\end{cases}$，$y=\begin{cases}\dfrac{a+b}{a-b} & a<b\\ \dfrac{4}{a+b} & a\geqslant b\end{cases}$。

这一题的算法并不难，可写成：

(1)从键盘输入 a，b 的值。

(2)如果 $a<b$，则 $x=a^2+b^2$，$y=\dfrac{a+b}{a-b}$，否则 $x=a^2-b^2$，$y=\dfrac{4}{a+b}$。

(3)计算 u 的值：$\dfrac{x-y}{x+y}$。

(4)输出 u 的值。

例 1-2　输入 10 个数，要求找出其中最大的数。

设 max 单元用于存放最大数，先将输入的第 1 个数放在 max 中，再将输入的第 2 个数与 max 相比较，较大者放在 max 中，然后将第 3 个数与 max 相比，较大者放在 max 中，…，一直到比完 9 次为止。

算法要在计算机上实现，还需要把它描述为更适合程序设计的形式，对算法中的量要抽象化、符号化，对算法的实施过程要条理化。上述算法可写成如下形式：

(1)输入一个数，存放在 max 中。

(2)用 i 来统计比较的次数，其初值置 1。

(3)若 i≤9，执行第(4)步，否则执行第(8)步。

(4)输入一个数，放在 x 中。

(5)比较 max 和 x 中的数，若 x>max，则将 x 的值送给 max，否则，max 值不变。

(6)i 增加 1。

(7)返回到第(3)步。

(8)输出 max 中的数，此时 max 中的数就是 10 个数中最大的数。

从上述算法示例可以看出，算法是解决问题的方法和步骤的精确描述。算法并不给出问题的精确解，只是说明怎样才能得到解。每一个算法都是由一系列基本的操作组成的。这些操作包括加、减、乘、除、判断、置数等。所以研究算法的目的就是要研究怎样把问题的求解过程分解成一些基本的操作。

算法设计好之后，要检查其正确性和完整性，再根据它用某种高级语言编写出相应的程序。程序设计的关键就在于设计出一个好的算法。所以，算法是程序设计的核心。

2. 算法的特性

从上面的例子中，可以概括出算法的 5 个特性：

(1)有穷性。算法中执行的步骤总是有限次数的，不能无止境地执行下去。例如，计算圆周率 π 的值，可用如下公式：

$$\frac{\pi}{4}=1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\cdots$$

这个多项式的项数是无穷的，因此，它是一个计算方法，而不是算法。要计算 π 的值，只能取有限项。例如，计算结果精确到第 5 位，那么，这个计算就是有限次的，因而才能称得上算法。

(2)确定性。算法中的每一步操作必须具有确切的含义，不能有二义性。

(3)有效性。算法中的每一步操作必须是可执行的。

(4)要有数据输入。算法中操作的对象是数据，因此应提供有关数据。但如果算法本身给出了运算对象的初值，也可以没有数据输入。

(5)要有结果输出。算法的目的是用来解决一个给定的问题，因此应提供输出结果，否则算法就没有实际意义。

3. 算法评价标准

在算法设计中，只强调算法特性是不够的。一个算法除了满足 5 个特性之外，还有一个质量问题。一个问题可能有若干个不同的求解算法，一个算法又可能有若干个不同的程序实现。在不同算法中有好算法，也有差算法。设计高质量算法是设计高质量程序的基本前提。如何评价算法的质量呢？不同时期、不同环境其评价标准可能不同，但一些基本评价标准是相同的。目前，评价算法质量有 4 个基本标准：

(1)正确性。一个好算法必须保证运行结果正确。算法正确性，不能主观臆断，必须经过严格验证，一般不能说绝对正确，只能说正确性高低。目前程序正确性很难给出严格的数学证明，程序正确性证明尚处于研究阶段。要多选用现有的、经过时间考验的算法，或采用科学规范的算法设计方法，是保证算法正确性的有效途径。

(2)可读性。一个好算法应有良好的可读性，好的可读性有助于保证正确性。科学、规范的程序设计方法(如结构化方法和面向对象方法)可提高算法的可读性。

(3)通用性。一个好算法要尽可能通用，可适用一类问题的求解。例如，设计求解一元二次方程 $2x^2+3x+1=0$ 的算法，该算法应设计成求解一元二次方程 $ax^2+bx+c=0$ 的算法。

(4)高效率。效率包括时间和空间两个方面。一个好的算法应执行速度快、运行时间短、占用内存少。效率和可读性往往是矛盾的，可读性要优先于效率。目前，在计算机速度比较快，内存容量比较大的情况下，高效率已处于次要地位。

4. 算法效率的度量

算法效率的度量分为时间度量和空间度量。

1)时间度量

算法的执行时间需要依据该算法编制的程序在计算机上运行时所消耗的时间来度量。它大致等于计算机执行一种简单操作(如赋值、比较等)所需的平均时间与算法中进行简单操作的次数的乘积。因为执行一种简单操作所需的平均时间随计算机而异，它是由所使用计算机的软硬件环境决定的，与算法无关，所以只需讨论影响算法执行时间的另一因素，即算法中进行简单操作的次数。通常把算法中进行简单操作的次数的多少称为算法的时间复杂度，它是一个算法执行时间的相对度量。

一般用问题的规模来表示算法所处理数据的多少。若解决问题的规模为 n，那么算法的时间复杂度就是问题规模 n 的一个函数 $f(n)$，假定时间复杂度记作 $T(n)$，则：

$$T(n)=f(n)$$

例如，求 $s=1+2+3+\cdots+n$，这里问题的规模为 n，求 s 的值需要进行 $n-1$ 次加法，所以算法的时间复杂度 $T(n)=n-1$。

这里算法比较简单，时间复杂度容易计算，当算法比较复杂时，时间复杂度的计算就相对困难。实际上，一般也没必要计算出算法的精确复杂度，只要大致计算出相应的数量级即可，即随着问题规模 n 的增大，算法的执行时间的增长率如何，这个时间增长率称为阶。显然求 s 值算法的执行时间与 n 成正比，所以该算法的时间复杂度是 n 阶的，记为 $T(n)=O(n)$。

算法的复杂度采用数量级表示后，将给计算复杂度带来很大的方便，这时只需分析影响一个算法的主要部分即可，不必对每一步进行分析。同时对主要部分的分析也可简化，只需分析循环内简单操作的次数即可。例如，下面 Python 语句的时间复杂度为 $O(n^2)$，即是 n 的平方阶的。

```
for i in range(n)
    for j in range(n)
        s= i+j
```

按数量级递增顺序，常见的几种时间复杂度有 $O(1)$、$O(\log_2 n)$、$O(n)$、$O(n\log_2 n)$、$O(n^2)$、$O(n^3)$、$O(2^n)$等。$T(n)=O(1)$表示算法执行时间与问题规模无关。图 1-1 给出了各种具有代表性的时间复杂度 $T(n)$与问题规模 n 的变化关系。从图中可以看到，当 $T(n)$为对数阶、线性阶、幂阶函数或它们的乘积时，算法的时间复杂度随问题规模的变化是可以接受的，当 $T(n)$为指数阶或它们的乘积时，算法的时间复杂度随问题规模的增大大幅增加，是不可以接受的，这种算法称为无效算法。

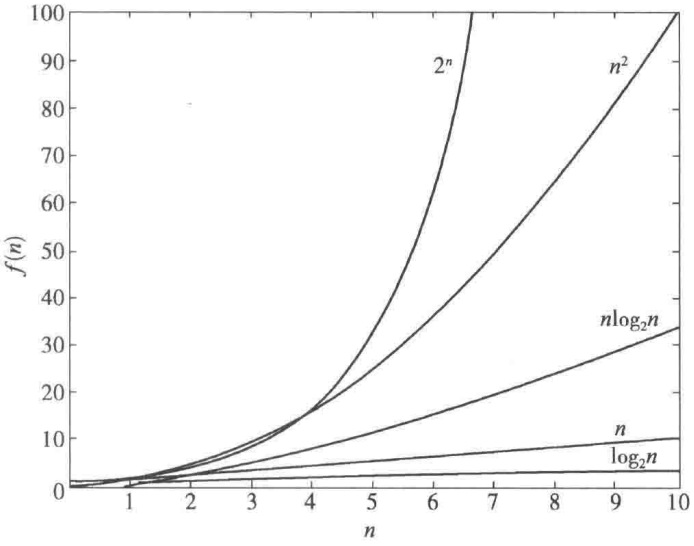


图 1-1　时间复杂度函数曲线比较

一个算法的时间复杂度除了与问题的规模有关外，还与输入的数据的次序有关，输入的次序不同，算法的复杂度也不同，所以当分析算法的复杂度时，还要考虑到最好、最坏和平均复杂度。

2）空间度量

一个算法的实现所占用的存储空间，大致包括 3 个方面：一是存储算法本身所占用的存储空间；二是算法中的输入、输出数据所占用的存储空间；三是算法在运行过程中临时占用的存储空间。存储算法本身所占用的存储空间与算法书写的长度有关，算法越长，占用的存储空间