

# Python 程序设计

陈春晖 翁 恺 季江民 编著

## 图书在版编目(CIP)数据

Python 程序设计 / 陈春晖, 翁恺, 季江民编著. —  
杭州: 浙江大学出版社, 2019. 1

ISBN 978-7-308-18977-4

I. ①P… II. ①陈 …②翁… ③季… III. ①软件工  
具—程序设计—高等学校—教材 IV. ①TP311.561

中国版本图书馆 CIP 数据核字 (2019) 第 028536 号

## Python 程序设计

陈春晖 翁 恺 季江民 编著

---

策 划 黄娟琴

责任编辑 王元新 黄娟琴

责任校对 刘 郡 汪志强

封面设计 程 晨 黄小意

出版发行 浙江大学出版社

(杭州市天目山路 148 号 邮政编码 310007)

(网址: <http://www.zjupress.com>)

排 版 杭州中大图文设计有限公司

印 刷 杭州杭新印务有限公司

开 本 787mm×1092mm 1/16

印 张 16.25

字 数 375 千

版 印 次 2019 年 1 月第 1 版 2019 年 1 月第 1 次印刷

书 号 ISBN 978-7-308-18977-4

定 价 45.00 元

---

版权所有 翻印必究 印装差错 负责调换

浙江大学出版社市场运营中心联系方式: 0571-88925591; <http://zjdxcbbs.tmall.com>

# 前言

为了适应信息技术的发展,切实满足社会各个领域对计算机应用人才不断增长的需求,本教材设计了“Python 程序设计”通识课程教学方案,力求融入计算思维的思想,并将多年教学实践所形成的解决实际问题的思维模式和方法渗透到整个教学过程中。与传统的程序设计类教材不同,本教材在介绍程序设计的基本技能外,还着重介绍分析问题和解决问题的方法与思路,通过构建典型案例,为学生在未来利用 Python 程序设计语言解决各自专业中遇到的实际问题打下良好的基础。

本书具有以下特点:

## 1. 注重与中学内容的衔接

本教材的内容编排凝聚了作者多年的教学经验与体会,如第 2 章就引入列表解析并用于求和编程。新生很熟悉的求和形式是:

$$1+2+3+\cdots+100=\sum_{i=1}^{100}i$$

$\sum$  对应于 sum 函数, $\sum i$  就是  $\text{sum}(i)$ , $\sum_{i=1}^{100}$  对应  $\text{for } i \text{ in range}(1,101)$ ,合起来就是:

```
sum(i for i in range(1,101))
```

程序完成了? 是的! 这样编程简单明了,非常容易学习,还符合 Python 的编程习惯。这种情况在本教材中有不少,读者可自己研究。

## 2. 完善的配套练习

本教材配套的教学网站是 PTA(Programming Teaching Assistant),配套的练习都可在该网站上进行。PTA 官网网址:<https://pintia.cn/>,教师账号申请可与陈越老师联系,邮箱:chenyue@zju.edu.cn。

## 3. 精心选择第三方模块教学

Python 有十几万个官方认可的第三方模块,同一功能可能有好几个模块都

能完成,因此选择合适的模块进行教学就很重要。本教材尽量选择成熟、使用方便、功能强大的模块进行教学。网络爬虫教学通常会使用“requests”和“beautifulsoup”两个模块完成,但本教材选择了“requests-html”模块(与“requests”模块是同一作者),既方便教学,又能让学生掌握网络爬虫程序的更多技能。

#### 4. 教学内容符合社会实际需求

具有开发 Web 应用程序的能力是社会的普遍需求。本教材重点介绍了 Python 的 Web 应用程序开发,即用一个实例展示开发的整个过程。学生被要求结合本专业的实际情况,完成相应的 Web 应用程序,并部署到云端。从实际的教学效果看,各专业的同学都能完成,学生成就感满满!

本教材的教学内容是为 64 学时设计的,每周 4 节课,2 节理论教学,2 节上机实践。

如果教学安排是 32 学时,可选择完成第 1 至第 6 章和 7.1 节的教学内容。

本教材第 1、2、6、7、9 章和附录由陈春晖老师执笔,第 3、4、5 章由翁恺老师执笔,第 8 章由季江民老师执笔。全书由陈春晖老师统稿。

本教材是在许端清教授的大力支持下完成的,在教材的编写过程中,方宁老师提出了不少宝贵意见,陈越老师为教学工作提供了优质的教学实践平台,在此一并表示感谢!

由于时间仓促和作者水平有限,书中难免有不妥之处,恳请广大读者批评指正。

本书的配套课件、源代码等请联系责任编辑(邮箱:13588451163@163.com)。关于本教材及课件使用中的问题,请发邮件至 cchui@zju.edu.cn。

作 者  
2019 年 1 月

# 目 录

第 1 章 Python 语言概述 .....	1
1.1 计算机基础 .....	1
1.1.1 计算机特点 .....	1
1.1.2 计算机常用的数制及编码 .....	2
1.1.3 计算机系统组成 .....	5
1.1.4 操作系统 .....	7
1.1.5 程序设计语言 .....	8
1.2 Python 语言简介 .....	9
1.3 Python IDLE 开发环境 .....	11
1.3.1 Python IDLE 开发环境安装 .....	11
1.3.2 运行 Python 程序 .....	13
1.4 标识符和变量 .....	15
1.4.1 标识符和关键字 .....	15
1.4.2 常量和变量 .....	16
1.5 输入及输出函数 .....	17
1.5.1 输入函数 .....	17
1.5.2 输出函数 .....	18
本章小结 .....	20
习 题 .....	21
第 2 章 用 Python 语言编写程序 .....	23
2.1 数字类型 .....	23
2.1.1 整数 .....	23
2.1.2 浮点数 .....	26
2.1.3 复数 .....	27

2.1.4	数学库(math 库)的使用 .....	28
2.2	字符串 .....	29
2.3	布尔值、空值和列表 .....	32
2.3.1	布尔值 .....	32
2.3.2	列表 .....	35
2.4	内置转换函数 .....	36
2.5	语句 .....	39
2.5.1	赋值语句 .....	39
2.5.2	if 语句 .....	41
2.5.3	for 语句 .....	42
2.5.4	列表推导式 .....	44
2.6	格式化输出 .....	46
	本章小结 .....	48
	习 题 .....	48
<b>第 3 章</b>	<b>使用字符串、列表和元组 .....</b>	<b>51</b>
3.1	序列的访问及运算符 .....	51
3.1.1	什么是序列 .....	51
3.1.2	通用序列操作 .....	51
3.1.3	序列的运算符 .....	55
3.1.4	计算序列的长度和最值 .....	56
3.2	字符串的使用 .....	57
3.2.1	什么是字符串 .....	57
3.2.2	字符串常用方法或函数 .....	59
3.2.3	将数字转换成字符串 .....	62
3.3	列表和元组使用 .....	66
3.3.1	列表 .....	66
3.3.2	基本的列表操作 .....	67
3.3.3	列表的函数或方法 .....	69
3.3.4	字符串和列表的互相操作 .....	71
3.3.5	元组 .....	73
3.4	随机函数库(random 库) .....	74
	本章小结 .....	77
	习 题 .....	77

<b>第 4 章 条件、循环和其他语句</b>	79
4.1 条件语句	79
4.1.1 基本的条件语句	79
4.1.2 有分支的条件语句	81
4.1.3 嵌套的条件语句	83
4.1.4 连缀的 if-elif-else	84
4.1.5 条件表达式	85
4.2 while 循环	86
4.2.1 while 循环	86
4.2.2 循环内的控制	90
4.3 for 循环	93
4.3.1 for...in 循环	93
4.3.2 range()函数	94
4.4 搜索和排序	96
4.4.1 线性搜索	96
4.4.2 搜索最值	97
4.4.3 二分搜索	97
4.4.4 选择排序	99
4.4.5 冒泡排序	101
4.5 异常处理	102
本章小结	106
习 题	107
<b>第 5 章 集合和字典</b>	110
5.1 集合	111
5.1.1 创建集合	111
5.1.2 操作和访问集合的元素	111
5.1.3 元素、子集、超集和相等判断	112
5.1.4 集合运算	113
5.2 字典	114
5.2.1 创建字典	114
5.2.2 字典的基本运算	115
5.2.3 字典方法或函数	116
5.3 集合和字典的应用	117
5.3.1 数据结构	118

5.3.2 过程 .....	119
本章小结 .....	121
习 题 .....	121
<b>第 6 章 函 数</b> .....	123
6.1 函数的定义和调用 .....	123
6.2 函数参数 .....	126
6.2.1 位置参数 .....	126
6.2.2 关键字参数 .....	126
6.2.3 默认值参数 .....	127
6.2.4 不定长数目参数 .....	130
6.3 函数的返回值 .....	135
6.4 命名空间和作用域 .....	136
6.5 递归 .....	137
6.6 内置函数 .....	139
6.6.1 sorted() 函数 .....	139
6.6.2 map() 函数 .....	140
6.6.3 zip() 函数 .....	140
6.6.4 eval() 和 exec() 函数 .....	141
6.6.5 all() 和 any() 函数 .....	141
6.7 程序结构 .....	142
6.7.1 模块和包 .....	142
6.7.2 sys 模块 .....	143
本章小结 .....	145
习 题 .....	146
<b>第 7 章 文 件</b> .....	150
7.1 文件读写 .....	150
7.2 用 Pandas 模块读写常见格式文件 .....	156
7.2.1 Python 第三方模块的安装 .....	156
7.2.2 Pandas 和 Plotly 模块 .....	158
7.2.3 用 Pandas 读写各种类型的文件 .....	163
7.2.4 JSON 文件读写 .....	167
7.3 数据可视化 .....	169
本章小结 .....	172
习 题 .....	173



<b>第 8 章 类和对象</b>	175
8.1 类和对象的概念	175
8.2 类和对象的创建	176
8.2.1 定义类	176
8.2.2 创建对象	177
8.2.3 访问对象成员	177
8.2.4 属性值	178
8.3 使用对象编写程序	180
8.4 封装	187
8.4.1 类成员	187
8.4.2 私有成员与公有成员	190
8.5 继承和多态	191
8.5.1 继承	191
8.5.2 多态	192
本章小结	192
习 题	193
<b>第 9 章 Web 应用程序开发及网络爬虫</b>	199
9.1 Web 应用程序开发概述	199
9.1.1 Web 应用程序运行原理	199
9.1.2 超文本标记语言简介	201
9.1.3 层叠样式表	203
9.2 Web 应用框架 Flask	207
9.3 云上部署 Web 应用程序	219
9.4 网络爬虫	232
9.4.1 获取网页	232
9.4.2 获取元素	234
本章小结	238
习 题	238
<b>参考文献</b>	239
<b>附录 A Python 语言简明参考手册</b>	240
A.1 导引	240
A.2 词法分析	241

A.3	数据模型 .....	243
A.4	表达式 .....	244
A.5	语句 .....	245
A.6	顶层组件 .....	247
<b>附录 B</b>	<b>PTA 平台常见问题解答 .....</b>	<b>249</b>
B.1	评分 .....	249
B.2	常见问题 .....	250

# 第 1 章 Python 语言概述

## 1.1 计算机基础

### 1.1.1 计算机特点

自从 1946 年第一台电子计算机问世以来,计算机科学与技术已成为近代发展速度最快的一门学科,尤其是微型计算机的出现和计算机网络的发展,使计算机的应用渗透到社会的各个领域,有力地推动了信息社会的发展。计算机作为一种通用的信息处理工具,它具有极快的处理速度、很强的存储能力、精确的计算和逻辑判断能力。其主要特点如下。

#### 1. 运算速度快

当今计算机系统的运算速度已达到每秒几十亿亿次,微机也可达每秒亿次以上,使大量复杂的科学计算问题得以解决。例如,导弹轨道的计算、大型水坝的计算、长期天气预报的计算等,过去人工计算需要几年、几十年的时间,而现在使用计算机进行计算只需几天甚至几分钟的时间。

#### 2. 计算精确度高

科学技术的发展特别是尖端科学技术的发展,需要高度精确的计算。计算机控制的导弹之所以能准确地击中预定的目标,是与计算机的精确计算分不开的。一般计算机可以有几十位甚至几百位(二进制)有效数字,计算精度可由千分之几到亿分之几,这是任何其他计算工具所望尘莫及的。

#### 3. 具有存储和逻辑判断能力

随着计算机存储容量的不断增大,可存储记忆的信息越来越多。计算机不仅能进行计算,而且能把参加运算的数据、程序以及中间结果和最后结果保存起来,以供用户随时调用;还可以对各种信息(如语言、文字、图形、图像、音乐等)通过编码技术进行算术运算和逻辑运算,甚至进行推理和证明。

#### 4. 具有自动控制能力

计算机内部操作是根据人们事先编好的程序自动控制进行的。用户根据解题需要,事先设计好运行步骤与程序,计算机十分严格地按程序规定的步骤操作,整个过程无须人工干预。

### 1.1.2 计算机常用的数制及编码

数制也称计数制,是指用一组固定的符号和统一的规则来表示数值的方法。编码是采用少量的基本符号,选用一定的组合原则,以表示大量复杂多样的信息的技术。计算机是信息处理的工具,任何信息必须转换成二进制数据后才能由计算机进行处理、存储和传输。

#### 1. 二进制数

我们习惯使用的十进制数由 0、1、2、3、4、5、6、7、8、9 十个不同的符号组成,每一个符号处于十进制数中不同的位置时,它所代表的实际数值是不一样的。例如 1999 可表示成:

$$1 \times 1000 + 9 \times 100 + 9 \times 10 + 9 \times 1 = 1 \times 10^3 + 9 \times 10^2 + 9 \times 10^1 + 9 \times 10^0$$

式中:每个数字符号的位置不同,它所代表的数值也不同,这就是经常所说的个位、十位、百位、千位的意思。二进制数和十进制数一样,也是一种进位计数制,但它的基数是 2。二进制数中 0 和 1 的位置不同,它所代表的数值也不同。例如二进制数 1101 表示十进制数 13:

$$(1101)_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 8 + 4 + 0 + 1 = 13$$

一个二进制数具有下列两个基本特点:

- (1) 有两个不同的数字符号,即 0 和 1。
- (2) 逢二进一。

一般我们用<sub>角标</sub>表示不同进制的数。例如,十进制数用<sub>10</sub>表示,二进制数用<sub>2</sub>表示。

#### 2. 二进制数制与其他数制

数位是指数码在一个数中所处的位置。基数是指在某种进位计数制中,每个数位上所能使用的数码的个数。例如,二进制数基数是 2,每个数位上所能使用的数码为 0 和 1 两个数码。在数制中有一个规则,如果是 N 进制数,必须是逢 N 进 1。下面主要介绍与计算机有关的常用的几种计数制。

##### (1) 十进制(十进位计数制)

十进制数具有十个不同的数码符号 0、1、2、3、4、5、6、7、8、9,其基数为 10,特点是逢十进一,例如:

$$(1011)_{10} = 1 \times 10^3 + 0 \times 10^2 + 1 \times 10^1 + 1 \times 10^0$$

##### (2) 八进制(八进位计数制)

八进制数具有八个不同的数码符号 0、1、2、3、4、5、6、7,其基数为 8,特点是逢八进

一,例如:

$$(1011)_8 = 1 \times 8^3 + 0 \times 8^2 + 1 \times 8^1 + 1 \times 8^0 = (521)_{10}$$

(3)十六进制(十六进位计数制)

十六进制数具有十六个不同的数码符号 0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F(或小写 a、b、c、d、e、f),其基数为 16,特点是逢十六进一,例如:

$$(1011)_{16} = 1 \times 16^3 + 0 \times 16^2 + 1 \times 16^1 + 1 \times 16^0 = (4113)_{10}$$

如表 1-1 所示为 4 位二进制数的不同进制对照表。

表 1-1 4 位二进制数的不同进制对照表

二进制	十进制	八进制	十六进制	二进制	十进制	八进制	十六进制
0000	0	0	0	1000	8	10	8
0001	1	1	1	1001	9	11	9
0010	2	2	2	1010	10	12	A
0011	3	3	3	1011	11	13	B
0100	4	4	4	1100	12	14	C
0101	5	5	5	1101	13	15	D
0110	6	6	6	1110	14	16	E
0111	7	7	7	1111	15	17	F

3. ASCII 码

计算机中,对非数值的文字和其他符号进行处理时,要对文字和符号进行数字化处理,即用二进制编码来表示文字和符号。字符编码(Character Code)是用二进制编码来表示字母、数字以及专门符号。在计算机系统中,目前普遍采用的是 ASCII(American Standard Code for Information Interchange)码,即美国信息交换标准代码(见表 1-2)。ASCII 码有 7 位版本和 8 位版本两种,国际上通用的是 7 位版本,7 位版本的 ASCII 码有 128 个元素,只需用 7 个二进制位( $2^7=128$ )表示,其中控制字符 34 个,阿拉伯数字 10 个,大小写英文字母 52 个,各种标点符号和运算符号 32 个。在计算机中实际用 8 位表示一个字符,最高位为“0”。

表 1-2 ASCII 码

ASCII 值	字符	ASCII 值	字符	ASCII 值	字符	ASCII 值	字符
0	NUT	32	(space)	64	@	96	、
1	SOH	33	!	65	A	97	a
2	STX	34	"	66	B	98	b
3	ETX	35	#	67	C	99	c
4	EOT	36	\$	68	D	100	d

续表

ASCII 值	字符	ASCII 值	字符	ASCII 值	字符	ASCII 值	字符
5	ENQ	37	%	69	E	101	e
6	ACK	38	&	70	F	102	f
7	BEL	39	,	71	G	103	g
8	BS	40	(	72	H	104	h
9	HT	41	)	73	I	105	i
10	LF	42	*	74	J	106	j
11	VT	43	+	75	K	107	k
12	FF	44	,	76	L	108	l
13	CR	45	—	77	M	109	m
14	SO	46	.	78	N	110	n
15	SI	47	/	79	O	111	o
16	DLE	48	0	80	P	112	p
17	DC1	49	1	81	Q	113	q
18	DC2	50	2	82	R	114	r
19	DC3	51	3	83	S	115	s
20	DC4	52	4	84	T	116	t
21	NAK	53	5	85	U	117	u
22	SYN	54	6	86	V	118	v
23	TB	55	7	87	W	119	w
24	CAN	56	8	88	X	120	x
25	EM	57	9	89	Y	121	y
26	SUB	58	:	90	Z	122	z
27	ESC	59	;	91	[	123	{
28	FS	60	<	92	/	124	
29	GS	61	=	93	]	125	}
30	RS	62	>	94	^	126	~
31	US	63	?	95	_	127	DEL

#### 4. Unicode 编码和 UTF-8 编码

Unicode 编码(统一码、万国码、单一码)是计算机科学领域里的一项业界标准,包括字符集、编码方案等。Unicode 编码是为了解决传统的字符编码方案的局限而产生的,它为每种语言中的每个字符设定了统一且唯一的二进制编码,以满足跨语言、跨平台进

行文本转换与处理的要求。Unicode 编码通常用两个字节表示一个字符,原有的英文编码从单字节变成双字节,只需要把高字节全部填为 0 就可以了。

在 Unicode 编码中,“汉字”这两个字对应的 Unicode 编码是\u6c49\u5b57,“u”表示 Unicode 编码。

在 Unicode 编码中,我们有很多方式将数字 23389(孝)表示成程序中的数据,包括 UTF-8、UTF-16、UTF-32。UTF 是“Unicode Transformation Format”的缩写,可以翻译成 Unicode 字符集转换格式,即怎样将 Unicode 定义的数字转换成程序数据。UTF-8 格式用得最多,其以字节为单位对 Unicode 字符进行编码。

UTF-8 的特点是对不同范围的字符使用不同长度的编码。对于 0x00~0x7F 的字符,UTF-8 编码与 ASCII 编码完全相同。UTF-8 编码的最大长度是 6 个字节。Python 3 中的字符串是 Unicode 字符串而不是字节数组,这是与 Python 2 的主要差别之一。

1.1.3 计算机系统组成

完整的计算机系统包括硬件和软件两大部分。所谓硬件,是指构成计算机的实体设备,即由物理器件构成的具有输入/输出、存储、计算、控制的实体部件。广义地说,软件是指系统中的程序以及所有相关文档的集合。我们平时讲到的“计算机”一词,都是指含有硬件和软件的计算机系统,其结构如图 1-1 所示。

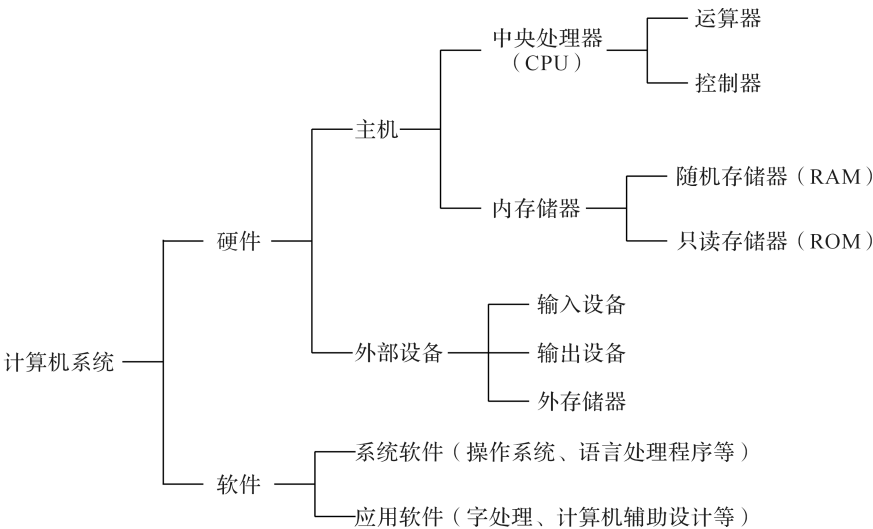


图 1-1 计算机系统组成

计算机硬件系统由运算器、控制器、存储器、输入设备/输出设备五个基本部分组成,也称计算机的五大部件。

运算器又称算术逻辑单元(Arithmetic Logic Unit, ALU),是计算机对数据进行运算处理的实体,它的主要功能是对二进制编码进行算术运算及基本逻辑运算,实现逻辑判断。运算器在控制器的控制下实现其功能,运算结果由控制器指挥送到内存储器中。

控制器主要由指令寄存器、译码器、程序计数器和操作控制器等组成。控制器用来控制计算机各部件协调工作,并使整个处理过程有条不紊地进行。它的基本功能就是从内存储器中取指令和执行指令,即控制器按程序计数器指出的指令地址从内存储器中取出该指令进行译码,然后翻译该指令,向有关部件发出控制命令,执行该指令。另外,控制器在工作过程中,还要接收各部件反馈回来的信息。

存储器具有记忆功能,用来保存信息,如数据、指令和运算结果等。存储器可分为两种:内存储器与外存储器。内存储器直接与 CPU 相连接,存储容量较小,但速度快,用来存放当前运行程序的指令和数据,并直接与 CPU 交换信息。内存储器由许多存储单元组成,每个单元能存放一个二进制数。存储器的存储容量以字节为基本单位,每个字节都有自己的编号,称为“地址”,如果要访问存储器中的某个信息,就必须知道它的地址,然后再按地址存入或取出信息。如果地址线是 8 位,则可以标识 256 单元,如表 1-3 所示。

表 1-3 存储器地址和单元内容

二进制内存储器单元地址	单元内容
00000000	01010001
00000001	11001110
00000010	00110011
...	...
...	...
...	...
11111110	10010000
11111111	01100010

为了度量信息存储容量,将 8 位二进制码(8bits)称为一个字节(Byte,简称 B)。字节是计算机中数据处理和存储容量的基本单位。1024 字节称为 1K 字节(1KB),1024K 个字节称 1 兆字节(1MB),1024M 个字节称为 1G 字节(1GB),1024G 个字节称为 1T 字节(1TB)。现在微型计算机主存容量大多数在几个 G 字节以上。

计算机处理数据时,一次可以运算的数据长度称为一个“字”(Word)。字的长度称为字长。一个字可以是一个字节,也可以是多个字节。常用的字长有 8 位、16 位、32 位、64 位等。如某一类计算机的字由 4 个字节组成,则字的长度为 32 位,相应的计算机称为 32 位机。

外存储器又称辅助存储器(简称辅存),它是内存储器的扩充。外存储器容量大,价格低,但存储速度较慢,一般用来存放大量暂时不用的程序、数据和中间结果,需要时,可成批地和内存储器进行信息交换。外存储器只能与内存储器交换信息,不能被计算机系统的其他部件直接访问。常用的外存储器有硬盘、磁带、光盘等。

输入/输出设备简称 I/O(Input/Output)设备。用户通过输入设备将程序和数据输入计算机,输出设备将计算机处理的结果显示或打印出来。



常用的输入设备有:键盘、鼠标、扫描仪、数字化仪等。

常用的输出设备有:显示器、打印机、绘图仪等。

人们通常把内存储器、运算器和控制器合称为计算机主机。而把运算器、控制器放在一个大规模集成电路块上,称为中央处理器,又称 CPU(Central Processing Unit)。也可以说,主机是由 CPU 与内存储器组成的,而主机以外的装置称为外部设备,外部设备包括输入/输出设备、外存储器等。

#### 1.1.4 操作系统

操作系统(Operating System, OS)是管理和控制计算机各种资源的计算机程序,是直接运行在“裸机”上的最基本的系统软件。操作系统是用户和计算机的接口,同时也是计算机硬件和其他软件的接口。操作系统的功能包括进程管理、存储管理、设备管理和文件管理,为其他应用软件提供支持,让计算机系统所有资源最大限度地发挥作用,提供各种形式的用户界面,使用户有一个好的工作环境,为其他软件的开发提供必要的服务和相应的接口。

操作系统的种类相当多,各种设备安装的操作系统从简单到复杂,常用的是个人计算机操作系统、多处理器操作系统、网络操作系统和大型机操作系统。

桌面操作系统主要用于个人计算机。从软件上主要分为 UNIX 和类 UNIX 操作系统、Windows 操作系统两大类。

(1)UNIX 和类 UNIX 操作系统:Mac OS, Linux 发行版(如 Debian, Ubuntu, Linux Mint, openSUSE, Fedora, Mandrake, Red Hat, CentOS 等)。

(2) Windows 操作系统: Windows 98, Windows 2000, Windows XP, Windows Vista, Windows 7, Windows 8, Windows 8.1, Windows 10。

服务器操作系统一般指安装在服务器上的操作系统,比如 Web 服务器、应用服务器和数据库服务器等。服务器操作系统主要集中于以下两大类。

(1)UNIX/Linux 系列: SUN Solaris, IBM-AIX, HP-UX, FreeBSD, Red Hat Linux, CentOS, Debian, Ubuntu Server 等。

(2) Windows 系列: Windows NT Server, Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, Windows Server 2016。

2014 年 10 月 1 日,微软在旧金山召开新品发布会,对外展示了新一代 Windows 操作系统,将它命名为“Windows 10”,新系统的名称跳过了数字“9”。

计算机系统对系统中的软件资源,无论是程序或数据、系统软件或应用软件都以文件方式来管理。文件是存储在某种介质上的(如磁盘、磁带等)并具有文件名的一组有序信息的集合。文件名是由字符和数字组成的。例如,Windows 文件名由以下三部分组成,格式如下:

```
[ <盘符>]: <文件名>[. 扩展名]
```

格式 [ ] 中是可以省略的,盘符为存放文件的磁盘驱动器号,文件名由不超过 255 个字符组成。扩展名通常表示文件的性质,如 docx 表示 Word 文件。Windows 用文件

夹表示目录。D 盘 python 目录下的 hello.py 文件可表示为：

```
d:\python\hello.py
```

绝对路径是从盘符开始的路径，如：

```
c:\windows\system32\cmd.exe
```

相对路径是从当前路径开始的路径，假如当前路径为 c:\windows，那么要描述上述路径，只需输入 system32\cmd.exe。实际上，严格的相对路径写法应为：

```
.\system32\cmd.exe
```

其中，“.”表示当前路径，在通常情况下可以省略，只有在特殊情况下才不能省略。

### 1.1.5 程序设计语言

程序设计语言是用于书写计算机程序的语言。语言的基础是一组符号和一组规则。根据规则由符号构成的符号串的总体就是语言。在程序设计语言中，这些符号串就是程序。程序设计语言有两个方面的因素，即语法、语义。语法表示程序的结构或形式，即表示构成语言的各个符号之间的组合规律，但不涉及这些符号的特定含义，也不涉及使用者。语义表示程序的含义，即表示按照各种方法所表示的各个符号的特定含义，但不涉及使用者。虽然大多数的语言既可被编译又可被翻译，但大多数只在一种情况下能够良好运行。Python 语言采取翻译这种方式。

自计算机出现以来，世界上公布的程序设计语言已有上千种之多，但是只有很少一部分得到了广泛的应用。

从发展历程来看，程序设计语言可以分为 4 代。

#### 第一代：机器语言

机器语言是由二进制 0、1 代码指令构成，不同的 CPU 具有不同的指令系统。机器语言程序难学难写，需要用户直接对存储空间进行分配，编程效率极低。这种语言已经被逐渐淘汰了。

#### 第二代：汇编语言

汇编语言指令是机器指令的符号化，与机器指令基本上存在着一一对应关系，所以汇编语言同样存在着难学难用、容易出错、维护困难等缺点。但是，汇编语言也有自己的优点：可直接访问系统接口，汇编程序翻译成机器语言程序的效率高。从软件工程角度来看，只有在高级语言不能满足设计要求，或不具备支持某种特定功能的技术性能（如特殊的输入输出）时，汇编语言才被使用。

#### 第三代：高级语言

高级语言是面向用户的、基本上独立于计算机硬件的语言。其最大的优点是：形式上接近于算术语言和自然语言，概念上接近于人们通常使用的概念。高级语言的一个命令可以代替几条、几十条甚至几百条机器语言的指令。因此，高级语言易学易用，通用性强，应用广泛。高级语言种类繁多，可以从应用特点和对客观系统的描述两个方面

对其进行进一步分类。C、Java 和 Python 都是高级语言。

#### 第四代:非过程化语言

第四代语言是非过程化语言,编码时只需说明“做什么”,不需描述算法细节。数据库查询和应用程序生成器是第四代语言的两个典型应用。用户可以用数据库查询语言(Structured Query Language,SQL)对数据库中的信息进行复杂的操作。用户只需将要查找的内容在什么地方、根据什么条件进行查找等信息告诉 SQL,SQL 将自动完成查找过程。应用程序生成器则是根据用户的需求“自动生成”满足需求的高级语言程序。

## 1.2 Python 语言简介

Python 语言是一种面向对象的解释型计算机程序设计语言,由荷兰人 Guido van Rossum 于 1989 年发明,第一个公开发行人版发行于 1991 年。之所以选中 Python(“大蟒蛇”的意思)作为该编程语言的名字,是因为 Guido 是 Monty Python 喜剧团体的爱好者。Python 从 ABC 语言继承了很多特性。ABC 语言是由 Guido 参加设计的一种教学语言。Guido 本人觉得,ABC 这种语言非常优美和强大,是专门为非专业程序员设计的。但是,ABC 语言并没有成功,究其原因,Guido 认为是其非开放造成的。Guido 决心在 Python 语言中避免这一错误。就这样,Python 语言在 Guido 手中诞生了。可以说,Python 语言是从 ABC 语言发展起来,并且结合了 UNIX Shell 和 C 语言的习惯。

Python 语言已经成为最受欢迎的程序设计语言之一。2019 年 1 月,它被 TIOBE 编程语言排行榜评为 2018 年度语言。最近,Python 语言的使用率呈线性增长。由于 Python 语言的简洁性、易读性以及可扩展性,在国外用 Python 语言做科学计算的研究机构日益增多,一些知名大学已经采用 Python 语言来教授程序设计课程。例如,麻省理工学院的计算机科学及编程导论就使用 Python 语言讲授。众多开源的科学计算软件包都提供了 Python 语言的调用接口,如著名的计算机视觉库 OpenCV、三维可视化库 VTK。而 Python 语言专用的科学计算扩展库就更多了,如 3 个十分经典的科学计算扩展库:NumPy、Pandas 和 Matplotlib,它们分别为 Python 语言提供了快速数组处理、数值运算以及绘图功能。因此,Python 语言及其众多的扩展库所构成的开发环境十分适合工程技术、科研人员处理实验数据和制作图表,甚至开发科学计算应用程序。

Python 语言的设计哲学是“明确”“简单”“优雅”。Perl 语言中“总是有多种方法来做同一件事”的理念在 Python 开发者中通常是难以忍受的。Python 开发者的哲学是“用一种方法,最好是只有一种方法来做一件事”。在设计 Python 语言时,如果面临多种选择,Python 开发者一般会拒绝花哨的语法,而选择没有或者很少有歧义的语法。这些准则被称为 Python 格言。在 Python 解释器内运行“import this”可以获得完整的列表。

Python 开发人员尽量避开不成熟或者不重要的优化。一些针对非重要部位的加快运行速度的方法通常用 Python 的模块实现。所以很多人认为 Python 运行速度很慢。不过,根据二八定律,大多数程序对速度要求不高。对于某些对运行速度要求很高的情况,Python 设计师倾向于使用 JIT 技术,或者使用 C/C++ 语言改写这部分程序。

可用的 JIT 技术是 PyPy。

Python 是完全面向对象的语言。函数、模块、数字、字符串都是对象,并且完全支持继承、重载、派生,有益于增强源代码的复用性。Python 支持重载运算符和动态类型。相对于 Haskell 这种传统的函数式编程语言,Python 对函数式设计只提供了有限的支持。其有两个标准库(functools、itertools)提供了久经考验的函数式程序设计工具。

Python 在执行时,首先会将“.py”文件中的源代码翻译成 Python 的 byte code(字节码),然后再由 Python Virtual Machine(Python 虚拟机)来执行这些翻译好的 byte code。这种机制的基本思想跟 Java 是一致的。然而,Python Virtual Machine 与 Java 的 Virtual Machine 不同的是,Python 的 Virtual Machine 是一种更高级的 Virtual Machine。这里的高级并不是通常意义上的高级,不是说 Python 的 Virtual Machine 比 Java 的功能更强大,而是说和 Java 相比,Python 的 Virtual Machine 距离真实机器更远。或者可以这么说,Python 的 Virtual Machine 是一种抽象层次更高的 Virtual Machine。基于 C 的 Python 翻译出的字节码文件,通常是“.pyc”格式。

Python 采用动态类型系统。在翻译时,Python 不会检查对象是否拥有被调用的方法或者属性,而是直至运行时,才做出检查,所以操作对象时可能会抛出异常。不过,虽然 Python 采用动态类型系统,但它同时也是强类型的。Python 禁止没有明确定义的操作,比如列表加字符串。

Python 语言有以下特点。

(1)简单易学:Python 是一种代表简单主义思想的语言。Python 极其容易上手,阅读一个良好的 Python 程序就感觉像是在读英语一样。

(2)免费、开源:Python 是自由软件之一。使用者可以自由地发布这个软件的拷贝,阅读它的源代码,对它做改动,把它的一部分用于新的自由软件中。

(3)高级语言:用 Python 语言编写程序时无须考虑如何管理你的程序使用的内存等这类底层细节。

(4)可移植性:由于它的开源本质,Python 已经被移植到许多平台上。这些平台包括 Linux、Windows、Mac OS。用 Python 语言写的程序不需要编译成二进制代码。你可以直接从源代码运行程序。在计算机内部,Python 解释器把源代码转换成称为字节码的中间形式,然后再把它翻译成计算机使用的机器语言并运行。这使得 Python 语言更加简单,也使得 Python 程序更加易于移植。

(5)可扩展性:如果需要一段关键代码运行得更快或者希望某些算法不公开,可以部分程序用 C 或 C++编写,然后在 Python 程序中使用它们。反过来也可以把 Python 嵌入 C/C++程序,从而向程序用户提供脚本功能。

(6)丰富的库:Python 标准库很庞大。它可以帮助处理各种工作,包括正则表达式、文档生成、单元测试、线程、数据库、网页浏览器、CGI、FTP、电子邮件、XML、XML-RPC、HTML、WAV 文件、密码系统、GUI(图形用户界面)、Tk 和其他与系统有关的操作。除了标准库以外,还有许多其他高质量的库,如 wxPython、Flask 和 Pillow 图像库等。

Python 是一个跨平台的脚本语言。如果它规定了一个 Python 语法规则,实现了

Python 语法的解释程序,就成为 Python 的解释器。Python 解释器有许多种,常用的有:

- (1)CPython:ClassicPython,也就是原始的 Python 实现,需要区别于其他实现时才以 CPython 称呼;或称作 C 语言实现的 Python。这是最常用的 Python 版本。
- (2)Jython: Java 语言实现的 Python,Jython 可以直接调用 Java 的各种函数库。
- (3)PyPy:使用 Python 语言写的 Python 解释器。
- (4)IronPython:运行在 .net 平台的 Python 解释器。

Python 有许多集成开发环境,常用的有:

- (1)IDLE:Python 内置 IDE (随 Python 安装包提供)。
- (2)PyCharm:由著名的 JetBrains 公司开发,带有一整套可以帮助用户在使用 Python 语言开发时提高其效率的工具,比如调试、语法高亮、Project 管理、代码跳转、智能提示、自动完成、单元测试、版本控制。
- (3)Spyder:安装 Anaconda 自带的高级 IDE,与 Matlab 开发环境类似。
- (4)Jupyter:安装 Anaconda 自带的高级 IDE,为数据科学家首选的开发环境。
- (5)Python Tutor:在线开发环境,网址是 <http://www.pythontutor.com/>,可以可视化 Python 程序运行过程。本书有不少例子用此开发环境进行说明。

### 1.3 Python IDLE 开发环境

#### 1.3.1 Python IDLE 开发环境安装

##### 1. Windows 操作系统安装

打开浏览器前往 Python 下载网站 (<https://www.python.org/downloads/>),如图 1-2 所示。

Looking for a specific release?

Python releases by version number:













Release version	Release date	Click for more	
Python 3.7.0	2018-06-27	 Download	Release Notes
Python 3.6.6	2018-06-27	 Download	Release Notes
Python 2.7.15	2018-05-01	 Download	Release Notes
Python 3.6.5	2018-03-28	 Download	Release Notes
Python 3.4.8	2018-02-05	 Download	Release Notes
Python 3.5.5	2018-02-05	 Download	Release Notes
Python 3.6.4	2017-12-19	 Download	Release Notes
Python 3.6.3	2017-10-03	 Download	Release Notes
Python 3.3.7	2017-09-19	 Download	Release Notes
Python 2.7.14	2017-09-16	 Download	Release Notes
Python 3.4.7	2017-08-09	 Download	Release Notes
Python 3.5.4	2017-08-08	 Download	Release Notes

图 1-2 Python 各种版本

点击准备安装的 Python 语言版本,建议安装 Python 3.6 及以后的版本,如图 1-3 所示。

Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		9de6494314ea199e3633211696735f65	22710891	SIG
XZ compressed source tarball	Source release		1325134dd525b4a2c3272a1a0214dd54	16992824	SIG
Mac OS X 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	9fba50521dffa9238ce85ad640abaa92	27778156	SIG
Windows help file	Windows		17cc49512c3a2b876f2ed8022e0afe92	8041937	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64, not Itanium processors	d2fb546fd4b189146dbefeba85e7266b	7162335	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64, not Itanium processors	bee5746dc6ece6ab49573a9f54b5d0a1	31684744	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64, not Itanium processors	21525b3d132ce15cae6ba96d74961b5a	1320128	SIG
Windows x86 embeddable zip file	Windows		15802be75a6246070d85b87b3f43f83f	6400788	SIG
Windows x86 executable installer	Windows		67e1a9bb336a5eca0efcd481c9f262a4	30653888	SIG
Windows x86 web-based installer	Windows		6c8ff748c554559a385c986453df28ef	1294088	SIG

图 1-3 Python 不同操作系统的安装文件

如果你的操作系统是 64 位 Windows,可下载 Windows x86-64 executable installer。  
如果你的操作系统是 32 位 Windows,可下载 Windows x86 executable installer。  
如下载 64 位的“Python 3.6.3”,就会出现“python-3.6.3-amd64.exe”文件。执行“python-3.6.3-amd64.exe”文件,出现图 1-4 所示界面。

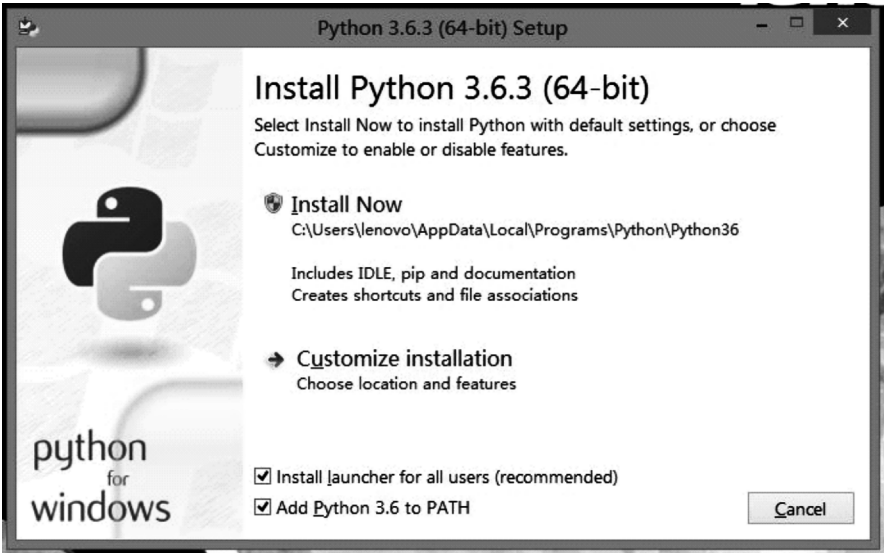


图 1-4 Python 3.6.3 的安装界面

选择“Add Python 3.6 to PATH”选项可以确保 PATH 路径中已包含 python.exe 的路径。再选择“Install Now”就可以安装了。注意要用管理员身份安装！

如希望安装别的 Python 开发环境,打开网页:<https://www.python.org/download/>

alternatives/,如图 1-5 所示。

## Alternative Python Implementations

This site hosts the "traditional" implementation of Python (nicknamed CPython). A number of alternative implementations are available as well, namely

- IronPython (Python running on .NET)
- Jython (Python running on the Java Virtual Machine)
- PyPy (A fast python implementation with a JIT compiler)
- Stackless Python (Branch of CPython supporting microthreads)
- MicroPython (Python running on micro controllers)

Other parties have re-packaged CPython. These re-packagings often include more libraries or are specialized for a particular application:

- ActiveState ActivePython (commercial and community versions, including scientific computing modules)
- pythonxy (Scientific-oriented Python Distribution based on Qt and Spyder)
- winpython (WinPython is a portable scientific Python distribution for Windows)
- Conceptive Python SDK (targets business, desktop and database applications)
- Enthought Canopy (a commercial distribution for scientific computing)
- PyIMSL Studio (a commercial distribution for numerical analysis – free for non-commercial use)
- Anaconda Python (a full Python distribution for data management, analysis and visualization of large data sets)

图 1-5 其他 Python 开发环境

建议下载“Anaconda Python”软件,安装使用。

### 2. Mac OS 和 Linux / UNIX 操作系统安装

根据操作系统从网站上下载相应的软件:

Python Releases for Mac OS X(<https://www.python.org/downloads/mac-osx/>)。

Python Source Releases(Linux and UNIX,<https://www.python.org/downloads/source/>)。

如下载的是 Mac OS X 64-bit/32-bit installer,则链接下载 Mac 中使用的.dmg 文件。下载完成后双击它,桌面上会弹出一个包含 4 个图标的窗口。右键单击 Python.mpkg,在弹出的对话框中点击“打开”,连续点击“继续”按钮,期间会显示一些法律声明,在最后的对话框出现后点击“安装”。

Linux 和 UNIX 用户可以选择下载 Python 源代码的压缩文件:

XZ compressed source tarball;

Gzipped source tarball。

下载上面两个链接中的一个,使用 tar xj 或 tar xz 命令解压,然后运行解压得到的 shell 脚本即可。

### 1.3.2 运行 Python 程序

安装好 Python 3.6.3 之后,可以用它来运行本书中的 Python 程序和你自己的 Python 代码。注意本书中的程序在“Python 3.6 及以后”的各个版本都可运行。

运行 IDLE 程序,启动 Python IDLE 开发环境,界面如图 1-6 所示。

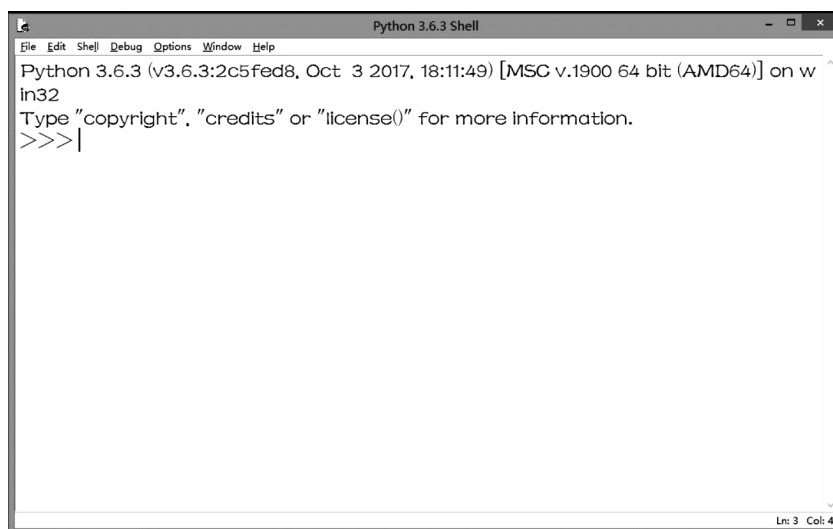


图 1-6 IDLE 运行界面

如何运行 Python 程序呢？通常有以下三种方法：

第一种是用 Python 自带的交互式解释器执行 Python 表达式或程序。你可以一行一行输入命令，然后立刻查看运行结果。这种方式可以很好地结合输入和输出结果，从而快速进行一些实验。交互式解释器的提示符是“>>>”。看到提示符是“>>>”时，说明解释器已处于等待输入的状态，输入表达式后按回车键，就能看到结果。

```
>>> 3 + 5 * 6
33
>>> print("hello world")
hello world
```

第二种是选择“File”→“New”，出现编程界面。输入程序（见图 1-7）：

```
print("hello world")
```

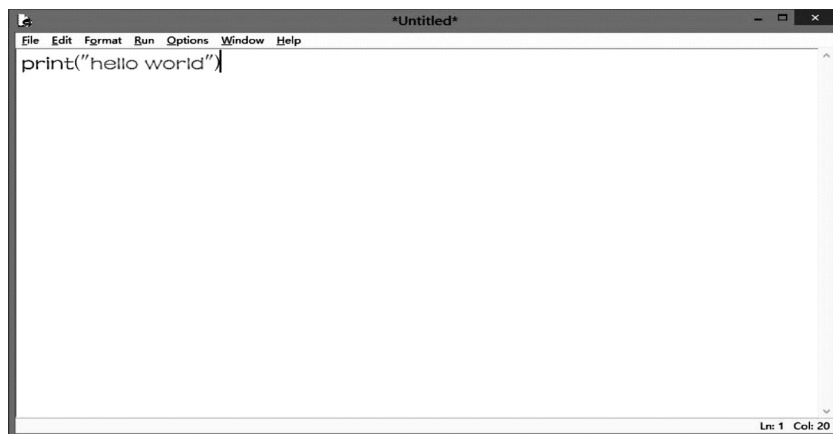


图 1-7 编程界面



选择 Save 菜单,保存程序到文件“hello.py”。

选择 Run 菜单运行,可看见结果“hello world”。

第三种是在命令行环境运行 Python 程序。如文件“hello.py”在 D 盘的根目录下,执行命令:

```
d:\>python hello.py
```

可看见结果“hello world”。

## 1.4 标识符和变量

### 1.4.1 标识符和关键字

标识符是指用来标识某个实体的符号。它在不同的应用环境下有不同的含义。在日常生活中,标识符是用来指定某个东西、人,要用到它、他或她的名字;在数学中解方程时,我们也常常用到这样或那样的变量名或函数名;在编程语言中,标识符是用户编程时使用的名字,对于变量、常量、函数、语句块而言也可以有名字;我们把这些统称为标识符。标识符由字母、下划线和数字组成,且不能以数字开头。

下面是正确的标识符:

```
my_Boolean  
Obj2  
myInt  
Mike2jack  
_test
```

下面是不正确的标识符:

```
my - Boolean  
2ndObj  
test!32  
haha(da)tt  
if  
jack&rose  
G. U. I
```

Python 中的标识符是区分大小写的,如 Andy 与 andy 就是不同的标识符。

if 完全符合标识符的定义,但为什么会错呢?

Python 中一些具有特殊功能的标识符,是所谓的关键字。关键字是 Python 语言已经使用了的,所以不允许开发者自己定义和关键字相同名字的标识符。if 是关键字,所以不能当标识符。

交互式解释器输入如下命令,就可显示 Python 关键字。

```
>>> help()

help > keywords

Here is a list of the Python keywords. Enter any keyword to get more help.

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

### 1.4.2 常量和变量

常量就是不变的量,比如常用的数学常数 3.14159 就是一个常量。

编程语言允许定义变量。变量名就是程序为了方便地引用内存中的值而为其取的名称。Python 变量名是区分大小写的,如 Julie 和 julie 就是不同的变量名。在 Python 中,我们用“=”来给一个变量赋值。

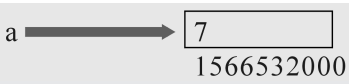
```
>>> a = 7
>>> a
7
```

7 是一个对象,可以通过变量 a 引用这个对象。

注意,Python 中的变量有一个非常重要的性质:变量是将名字和对象关联起来的。赋值操作并不会实际复制值,它只是为数据对象取个相关的名字。名字是对象的引用而不是对象本身。

id 是 Python 的内置函数,显示对象的地址。

```
>>> id(a)
1566532000
>>> id(7)
1566532000
```



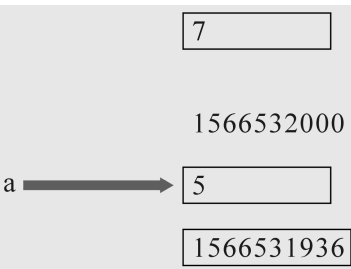
这表示“7”这个对象的地址是 1566532000。

再执行以下命令:

```

>>> a = 5
>>> a
5
>>> id(a)
1566531936
>>> id(5)
1566531936
>>> id(7)
1566532000

```



The diagram illustrates the memory state after the code execution. It shows three boxes representing objects in memory: a box containing '7' at address 1566532000, a box containing '5' at address 1566531936, and another box containing '5' at address 1566531936. An arrow labeled 'a' points from the text 'a' to the '5' object at address 1566531936, indicating that the variable 'a' now references this object.

a 这个变量现在引用对象“5”，地址是 1566531936；对象“7”地址还是 1566532000，没有变。

## 1.5 输入及输出函数

Python 输入函数是 `input()`，输出函数是 `print()`。

### 1.5.1 输入函数

`input()` 接受从键盘输入一个字符串。'9' 表示是一个字符串。

```

>>> a = input()
9
>>> a
'9'

```

如果需要输入数字，则需要用 `int()` 函数。

```

>>> b = int(input("请输入一个数字:"))
请输入一个数字: 9
>>> b
9

```

`input()` 函数的参数“请输入一个数字:”是输入的提示符。

可用 `split()` 函数在一行中输入多个值，用空格分开。

```

>>> m,n = input("请输入多个值:").split()
请输入多个值: 3 5
>>> m
'3'
>>> n
'5'

```

### 1.5.2 输出函数

`print()`是输出函数,参数是输出值。

在程序的编写过程中,我们需要对一些程序进行注释,除了方便自己阅读外,更是为了别人能更好地理解我们的程序。“#”常被用作单行注释符号。在代码中使用“#”时,它右边的任何内容都会被忽略,当作是注释。

```
>>> print(3)    # 输出 1 个数字
3
>>> print(3,7)  # 输出 2 个数字
3 7
>>> a = 6
>>> print(a)    # 输出 1 个变量
6
>>> b,c = 3,4
>>> print(b, c, 5)  # 输出 1 个数字,2 个变量
3 4 5
```

`print()`函数缺省是执行一次换一行,如何不换行呢?

**【例 1-1】** 用 3 个 `print()` 函数,在同一行输出 3 个数“3 4 5”。

参数 `end=' '` 表示下一个 `print()` 函数接着上一个 `print()` 函数在同一行打印。

程序代码:

```
# 每行输出 1 个值
print(3)
print(4)
print(5)
# 一行输出 3 个值
print(3,end=' ')
print(4,end=' ')
print(5,end=' ')

```

程序输出:

```
3
4
5
3 4 5
```

**【例 1-2】** 输入三角形的三条边的长度 3,4,5,求这个三角形的面积。

程序代码:

```
import math    #引入数学库
a = int(input())
b = int(input())
c = int(input())

s = (a + b + c)/2

area = math.sqrt(s*(s-a)*(s-b)*(s-c)) # '*'表示乘,math.sqrt 表示开根号
print("三角形的边长:",a,b,c,end = ' ')
print("三角形的面积:",area)
```

程序输入：

```
3
4
5
```

程序输出：

```
三角形的边长: 3 4 5 三角形的面积: 6.0
```

“#”表示注释,它右边的部分是对程序的说明,如“#引入数学库”。

要特别注意,语句块要对齐,不要随便加空格。下面程序“b=int(input())”语句前多了一个空格就错了。

```
import math    #引入数学库

a = int(input())
    b = int(input())

c = int(input())

s = (a + b + c)/2

area = math.sqrt(s*(s-a)*(s-b)*(s-c)) # '*'表示乘,math.sqrt 表示开根号
print("三角形的边长:",a,b,c,end = ' ')
print("三角形的面积:",area)

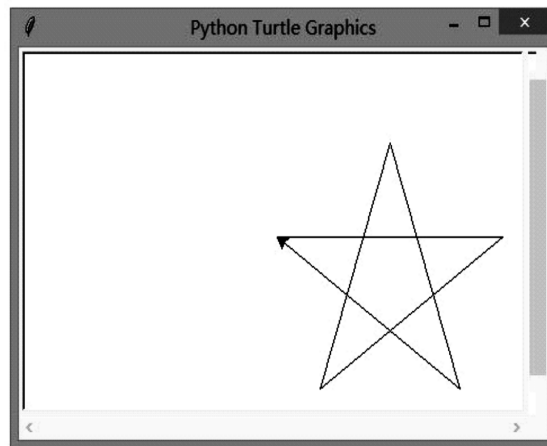
File "<python - input - 5 - 49a965827e2e>", line 3
b = int(input())
^
IndentationError: unexpected indent
```

**【例 1-3】** 画五角形。

Python 有很多库,turtle 是一个绘图库,用下面程序可画五角形。

```
import turtle
turtle.forward(200)
turtle.right(144)
turtle.forward(200)
turtle.right(144)
turtle.forward(200)
turtle.right(144)
turtle.forward(200)
turtle.right(144)
turtle.forward(200)
turtle.done()
```

程序输出：



## 本章小结

---

本章主要介绍：

1. 计算机系统组成。
  2. 二进制、八进制、十进制和十六进制。
  3. ASCII 码、Unicode 码和 UTF 码。
  4. Python 开发环境 IDLE 的安装和 Python 运行。
  5. Python 的输入函数 `input()` 和输出函数 `print()` 的使用。
-

## 习 题

### 一、选择题

- 计算机存储器的单位使用字节(Byte,B),1B 等于\_\_\_\_\_。  
A. 一位二进制      B. 四位二进制      C. 八位二进制      D. 十六位二进制
- Python 程序的扩展名是\_\_\_\_\_。  
A. py      B. exe      C. docx      D. jpg
- Python 的输出函数是\_\_\_\_\_。  
A. input()      B. print()      C. math()      D. turtle()
- 10 的二进制值是\_\_\_\_\_。  
A. 1100      B. 1010      C. 0011      D. 1110
- 八进制 35 的十进制值是\_\_\_\_\_。  
A. 30      B. 29      C. 19      D. 25
- 计算机系统由硬件和\_\_\_\_\_组成。  
A. 软件      B. 语言      C. 控制器      D. 内存储器
- \_\_\_\_\_是不等长编码。  
A. ASCII 码      B. UTF-8 码  
C. Unicode 码      D. 前三种编码都不是
- \_\_\_\_\_表示后面部分是注释。  
A. #      B. \*      C. %      D. &
- 正确的标识符是\_\_\_\_\_。  
A. 2you      B. my-name      C. \_item      D. abc \* 234
- Python 语言的官方网站是\_\_\_\_\_。  
A. www.python.com      B. www.python.org  
C. www.python.edu      D. pythonic.org
- 不是面向对象的程序设计语言是\_\_\_\_\_。  
A. Java      B. Python      C. C++      D. C

### 二、判断题

- 高级语言程序要被机器执行,只能用解释器来解释执行。 ( )
- Python 是一种跨平台、开源、免费的动态编程语言。 ( )
- 不可以同一台计算机上安装多个不同的 Python 版本。 ( )
- Python 3.X 完全兼容 Python 2.X。 ( )
- math 库是 Python 语言的数学库。 ( )
- 在 Python 3.X 中,input()函数把用户的键盘输入作为字符串返回。 ( )
- 在 Python 中,可以用 else 作为变量名。 ( )
- 已知  $x = 3$ , 则  $x = '3'$  是错误的。 ( )

10. 已知  $x = 3$ ，则执行“ $x = 7$ ”后， $\text{id}(x)$ 的返回值与原来没有变化。 ( )

### 三、编程题

1. 从键盘输入两个数，求它们的和并输出。
2. 从键盘输入三个数到  $a, b, c$  中，求  $b*b-4*a*c$  的值并输出。
3. 在屏幕上输出“Python 语言简单易学”。



## 第 2 章 用 Python 语言编写程序

计算机顾名思义就是可以做数学计算的机器，因此，计算机程序理所当然地可以处理各种数据。但是，计算机能处理的远不止数字，还可以处理文本、图形、音频、视频、网页等各种各样的数据。不同的数据需要定义不同的数据类型。在 Python 中，能够直接处理的数据类型如表 2-1 所示。

表 2-1 Python 中的基本数据类型

数据类型	说 明
整数	可以处理非常大的整数，整数运算永远是精确的，如 57
浮点数	浮点数运算则可能会有四舍五入的误差，如 3.14159
复数	由实部(real)和虚部(imaginary)两部分组成的数
字符串	字符串是以'或"括起来的任意文本，比如'abc'，"xyz"
布尔值	布尔值只有 True、False 两种值，要么是 True，要么是 False
空值	空值是 Python 中的一个特殊值，用 None 表示

### 2.1 数字类型

Python 数字类型包括整数、浮点数和复数。

#### 2.1.1 整数

任何仅含数字的序列在 Python 中都被认为是整数。

```
>>> 66
66

>>> 0
0
```

你可以单独使用数字 0,但不能把它作为前缀放在其他数字前面。

```
>>> 07
SyntaxError: invalid token
>>>
```

一个数字序列定义了一个正整数。你也可以显式地在前面加上正号“+”,这不会使数字发生任何变化。

```
>>> 5
5

>>> + 5
5
```

在数字前添加负号“-”可以定义一个负数。

```
>>> - 123
- 123
```

在 Python 中,整数默认使用十进制数,除非你在数字前添加前缀,显式地指定使用其他进制。也许你永远都不会在自己的代码中用到其他进制,但你很有可能在其他人编写的 Python 代码里见到它们。

进制指的是在必须进位前可以使用的数字的最大数量。以二进制为例,可以使用的数字只有 0 和 1,这里的 0 和十进制的 0 代表的意义相同,1 和十进制的 1 所代表的意义也相同。然而二进制时,1 与 1 相加得到的将是 10。

在 Python 中,除十进制外你还可以使用其他三种进制的数字:

- (1)0b 或 0B 代表二进制;
- (2)0o 或 0O 代表八进制;
- (3)0x 或 0X 代表十六进制。

Python 解释器会打印出它们对应的十进制整数。我们来试试这些不同进制的数。

```
>>> 0b10
2

>>> 0o10
8

>>> 0x10
16
```

在 Python 3 中,整数类型可以存储很大的整数,甚至超过 64 位。在许多其他编程语言中,进行类似下面的计算会造成整数溢出,这是因为计算中的数字或结果需要的存储空间超过了计算机所提供的(如 32 位或 64 位)。在程序编写中,溢出会产生许多负面影响。而 Python 在处理超大数计算方面不会产生任何错误,这也是它的一个加分点。

[illegible]

你可以对这些数字进行表 2-2 中的计算。

表 2-2 运算符

运算符	说明	示例	运算结果
+	加法	5+10	15
-	减法	100-5	95
*	乘法	8 * 9	72
/	浮点数除法	100/5	20.0
//	整数除法	100//5	20
%	模(求余)	9%4	1
**	幂	2**3	8

可以连续运算任意个数:

```
>>> 5+6*2
17
```

数字和运算符之间的空格不是强制的,你也可以写成下面这种格式:

```
>>> 5 + 6*2
17
```

在 Python 里,幂的优先级高于乘除,乘除的优先级高于加减。

在 Python 里有两种除法：

/用来执行浮点除法(十进制小数);

//用来执行整数除法(整除)。

与其他语言不同,在 Python 中即使运算对象是两个整数,使用/仍会得到浮点型的结果。

```
>>> 9 / 5
1.8
```

使用整除运算得到的是一个整数,余数会被截去。

```
>>> 9 // 5
1
```

如果除数为 0,任何一种除法运算都会使 Python 产生错误。

```
>>> 5 / 0
Traceback (most recent call last):
File "<pyshell #4>", line 1, in <module>
5/0 ZeroDivisionError: division by zero
```

### 2.1.2 浮点数

浮点数也就是小数,之所以称为浮点数,是因为按照科学计数法表示时,一个浮点数的小数点位置是可变的,如  $1.23 \times 10^9$  和  $12.3 \times 10^8$  是相等的。

浮点数可以用数学写法,如 1.23,3.14,-9.01,等等。

但是对于很大或很小的浮点数,就必须用科学计数法表示,把“10”用“e”替代, $1.23 \times 10^9$  就是 1.23e9,或者 12.3e8,0.000012 可以写成  $1.2 \times 10^{-5}$ 。

```
>>> -9.5
-9.5
>>> 1.2e3
1200.0

>>> e3
Traceback (most recent call last):
File "<pyshell#2>", line 1, in <module>
e3 NameError: name 'e3' is not defined

>>> 3.5e4.0
SyntaxError: invalid syntax
>>> 3.5e
SyntaxError: invalid syntax
```

“e”的前后都不能空,“e”的后面一定要是整数。

整数和浮点数在计算机内部存储的方式是不同的,整数运算是精确的,而浮点数运算则可能会有四舍五入的误差。

浮点数与整数一样,你可以使用运算符(+、-、\*、/、//、%)进行计算。

```
>>> 6.0/3.0
2.0

>>> 6.0//3.0
2.0

>>> 6.0%4.0
2.0
```

注意:浮点数的整除结果还是浮点数。

divmod()函数同时计算商和余数,4 是商,1 是余数。

```
>>> divmod(9,2)
(4, 1)

>>> divmod(9.0,2)
(4.0, 1.0)
```

使用 float()函数可以将整数转换为浮点数,用 int()函数可以将浮点数转换为整数。

```
>>> float(9)
9.0

>>> int(3.7)
3
```

Python 支持对整数和浮点数直接进行四则混合运算,运算规则和数学上的四则混合运算规则完全一致。整数和浮点数混合运算的结果就变成浮点数了。

```
>>> 2+0.5*6
5.0
```

### 2.1.3 复数

Python 语言支持复数运算。所谓复数,就是由实部(real)和虚部(imaginary)两部分组成的数,虚部用 j 表示。

```
>>> 3 + 2j
(3 + 2j)

>>> 8j           # 只有虚部
8j

>>> (7 + 1j) * 1j
-1 + 7j
```

real 方法取实部,imag 方法取虚部,complex()函数用于创建一个值为 real+imag \* j 的复数。

```
>>> a = 5 - 4j
>>> a.real
5.0
>>> a.imag
-4.0

>>> complex(4, -6)
(4 - 6j)
```

### 2.1.4 数学库(math 库)的使用

math 库是一个数学库,包含了很多常用函数和的数学常数,如表 2-3 所示。

表 2-3 math 库的常用函数和数学常数

函数名或常数	含 义	示 例
math. e	自然常数 e	math. e
math. pi	圆周率 pi	math. pi
math. log(x[, base])	返回 x 的以 base 为底的对数,base 默认为 e	math. log(math. e) math. log(2, 10)
math. log10(x)	返回 x 的以 10 为底的对数	math. log10(2)
math. pow(x, y)	返回 x 的 y 次方	math. pow(5, 3)
math. sqrt(x)	返回 x 的平方根	math. sqrt(3)
math. ceil(x)	返回不小于 x 的最小整数	math. ceil(5. 2)
math. floor(x)	返回不大于 x 的最大整数	math. floor(5. 8)
math. trunc(x)	返回 x 的整数部分	math. trunc(5. 8)
math. fabs(x)	返回 x 的绝对值	math. fabs(- 5)
math. sin(x)	返回 x(弧度)的三角正弦值	math. sin(3)
math. asin(x)	返回 x 的反三角正弦值	math. asin(0. 5)
math. cos(x)	返回 x(弧度)的三角余弦值	math. cos(1. 8)
math. acos(x)	返回 x 的反三角余弦值	math. acos(math. sqrt(2)/2)
math. tan(x)	返回 x(弧度)的三角正切值	math. tan(4)
math. atan(x)	返回 x 的反三角正切值	math. atan(1. 77)
math. atan2(x, y)	返回 x/y 的反三角正切值	math. atan2(2, 1)

要使用 math 库,先用“import math”语句引入 math 库。

```
>>> import math

>>> math.pi
3.141592653589793

>>> math.e
2.7182818284590

>>> math.pow(3,3)
27.0
```

```
>>> math.sqrt(9)
3.0

>>> math.sin(3)
0.1411200080598672

>>> math.cos(5)
0.28366218546322625

>>> math.tan(4)
1.1578212823495777

>>> math.ceil(5.8)
6

>>> math.floor(5.8)
5

>>> math.log(math.e)
1.0
```

用“.”记法调用的函数也称方法,两者都是一种操作,只是调用方式不同而已,本书不严格区分这两种说法。

## 2.2 字符串

字符串是以'或"括起来的任意文本,比如'abc'、"xyz"等。请注意,'或"本身只是一种表示方式,不是字符串的一部分。因此,字符串'abc'只有a、b、c这3个字符。

```
>>> 'hello world'
'hello world'

>>> "hello world"
'hello world'
```

交互式解释器输出的字符串永远是用单引号包裹的,但无论使用哪种引号,Python对字符串的处理方式都是一样的,没有任何区别。

既然如此,为什么要使用两种引号?这么做的好处是可以创建本身就包含引号的字符串。可以在双引号包裹的字符串中使用单引号,或者在单引号包裹的字符串中使用双引号。

你还可以使用连续三个单引号'''，或者三个双引号""" 创建字符串，三元引号在创建短字符串时没有什么特殊用处，它多用于创建多行字符串。

```
>>> '''hello python
    人生苦短
    我用 python'''
'hello python\n    人生苦短\n    我用 python'

>>> " "    # 空字符串
''
```

Python 允许空字符串的存在，它不包含任何字符且完全合法。

Python 允许对某些字符进行转义操作，以此来实现一些难以单纯用字符描述的效果。在字符的前面添加反斜线符号“\”，会使该字符的意义发生改变。最常见的转义符是“\n”，它代表换行符，便于在一行内创建多行字符串。

```
>>> print('hello python\n人生苦短\n我用 python')
hello python
人生苦短
我用 python
```

转义符“\t”(tab 制表符，等于 8 个空格)常用于对齐文本，之后会经常见到。有时还可能还会用到“\'”和“\””来表示单、双引号，尤其当该字符串由相同类型的引号包裹时，如表 2-4 所示。

```
>>> print('\thello python')
    hello python

>>> testimony = "\"I did nothing! \" he said. \"Not that either! Or the other thing.\""
>>> testimony
'I did nothing!' he said. "Not that either! Or the other thing."
```

如果你需要输出一个反斜线字符，连续输入两个反斜线(\\)即可。

表 2-4 转义符

转义字符	描 述
\\	反斜杠符号
\'	单引号
\"	双引号
\a	响铃
\b	退格(Backspace)
\n	换行
\v	纵向制表符



续表

转义字符	描 述
\t	横向制表符
\r	回车
\f	换页
\ooo	最多三位八进制数,例如:\12 代表换行
\xyy	十六进制数,yy 代表数字,例如:\x0a 代表换行

```
>>> '\12'      # 八进制
'\n'
>>> '\x0a'     # 十六进制
'\n'
>>> '\141'     # 八进制
'a'
```

在 Python 中,你可以使用“+”将多个字符串或字符串变量拼接起来,产生新字符串。

```
>>> "人生苦短" + " 我用 Python"
'人生苦短 我用 Python'
```

也可以直接将一个字面字符串(非字符串变量)放到另一个的后面直接实现拼接。

```
>>> "人生苦短" " 我用 Python"
'人生苦短 我用 Python'
```

使用 \* 可以进行字符串的复制,产生新字符串。

```
>>> '2' * 3
'222'
>>> i = 5
>>> '2' * i
'22222'
```

Python 3 的字符串缺省编码方式是 Unicode 码,这与 Python 2 不同。

type() 函数是一个内置的函数,调用它就能知道想要查询对象的类型信息,该函数的返回值为查询对象的类型。下面举例说明 type() 函数的使用。

```
>>> type(1)      # 整型
<class 'int'>
>>> type("python") # 字符串
<class 'str'>
>>> type(3 + 2j)   # 复数
<class 'complex'>
```

## 2.3 布尔值、空值和列表

### 2.3.1 布尔值

布尔值和布尔代数的表示完全一致,一个布尔值只有 True、False 两种值,要么是 True,要么是 False。在 Python 中,可以直接用 True、False 表示布尔值(请注意大小写),也可以通过逻辑运算符和关系运算符计算出来。关系运算符是 <、<=、>、>=、== 和 !=,逻辑运算符是 and、or 和 not。

#### 1. 关系运算符

Python 中关系运算符可以连用,其含义与人们日常的理解完全一致。使用关系运算符的最重要的前提是操作数之间必须可以比较大小。例如,把一个字符串和一个数字进行大小比较是毫无意义的,所以 Python 也不支持这样的运算,关系运算符如表 2-5 所示。

表 2-5 关系运算符

运算符	表达式	含义	实例	结果
==	x == y	x 等于 y	"ABCD" == "ABCDEF"	False
!=	x != y	x 不等于 y	"ABCD" != "abcd"	True
>	x > y	x 大于 y	"ABC" > "ABD"	False
>=	x >= y	x 大于等于 y	"123" >= "23"	True
<	x < y	x 小于 y	"ABC" < "DEF"	True
<=	x <= y	x 小于等于 y	"123" <= "23"	True

Python 中关系运算符的基本比较法则:

(1) 关系运算符的优先级相同。

(2) 对于两个预定义的数值类型,关系运算符按照操作数的数值大小进行比较。

(3) 对于字符串类型,关系运算符比较字符串的值,即按字符的 ASCII 码值从左到右一一比较:首先比较两个字符串的第一个字符,其 ASCII 码值大的字符串大,若第一个字符相等,则继续比较第二个字符,以此类推,直至出现不同的字符或穷尽字符串为止。

以下是关系运算符的实例:

```
>>> 1 < 5          # 等价于 1 < 5 and 5 < 5
True
>>> 3 < 5 > 2
True
>>> 1 > 6 < 8
```

```
False
>>> import math          # sqrt()是 math 模块下的函数,使用前必须先导入 math 模块
>>> 1>6 <math.sqrt(9)
False
>>> 'Hello'>'world'      # 比较字符串的大小  ascii('H') = 72 <ascii('w') = 119
False
>>> 'Hello'>3            # 字符串和数字不能进行比较
TypeError: unorderable types: str()>int()
```

2. 逻辑运算符

逻辑运算符 and、or、not 常用来连接条件表达式构成更加复杂的条件表达式,并且 and 和 or 具有惰性求值或者逻辑短路的特点,即当连接多个表达式时只计算必须要计算的 值。在编写复杂条件表达式时可以充分利用这个特点,合理安排不同条件的先后顺序,在 一定程度上可以提高代码的运行速度。另外要注意的是,运算符 and 和 or 并不一定会返 回 True 或 False,而是得到最后一个被计算的表达式的值,但是运算符 not 一定会返回 True 或 False。

and 运算是与运算,只有所有都为 True ,and 运算结果才是 True ,如表 2-6 所示。

表 2-6 and 运算

逻辑量 1	逻辑量 2	结果
False	False	False
True	False	False
False	True	False
True	True	True

or 运算是或运算,只要其中有一个为 True ,or 运算结果就是 True ,如表 2-7 所示。

表 2-7 or 运算

逻辑量 1	逻辑量 2	结果
False	False	False
True	False	True
False	True	True
True	True	True

not 运算是非运算,它是一个单目运算符,把 True 变成 False,False 变成 True ,如表 2-8 所示。

表 2-8 not 运算

逻辑量	结果
False	True
True	False

以下是逻辑运算符的实例：

```
>>> 3>5 and a>3          # 注意,此时并没有定义变量 a
False
>>> 3>5 or a>3            # 3>5 的值为 False,所以需要计算后面的表达式
NameError: name 'a' is not defined
>>> 3<5 or a>3            # 3<5 的值为 True,所以不需要计算后面的表达式
True
>>> 3 and 5                # 最后一个计算的表达式的值作为整个表达式的值
5
>>> 3 and 5>2
True
>>> not 3
False
>>> not 0
True
```

3. 空值

空值是 Python 里一个特殊的值,用 None 表示。None 不能理解为 0,因为 0 是有意义的,而 None 是一个特殊的空值。

```
>>> bool(None)
False
>>> None == 0
False
```

4. 运算符的优先级和结合性

数值数据常用运算符的优先级由高到低,如表 2-9 所示。

表 2-9 运算符的优先级和结合性

优先级 (1 最高,8 最低)	运算符	描 述	结合性
1	+ x, - x	正,负	
2	x ** y	幂	从右向左

续表

优先级 (1 最高,8 最低)	运算符	描 述	结合性
3	$x * y, x / y, x \% y$	乘、除、取模	从左向右
4	$x + y, x - y$	加、减	从左向右
5	$x < y, x <= y, x == y, x != y, x >= y, x > y$	比较	从左向右
6	<code>not x</code>	逻辑否	从左向右
7	<code>x and y</code>	逻辑与	从左向右
8	<code>x or y</code>	逻辑或	从左向右

下面举例说明运算符的优先级和结合性。

```
>>> 3 + 5 * 4  # 先乘后加
23
>>> 5 * 3 / 2  # 从左向右
7.5
>>> 2 * 3 * 2  # 从右向左
12
>>> 3 < 5 or a > 3  # 从左向右
True
```

2.3.2 列表

大多数编程语言都有特定的数据结构来存储由一系列元素组成的序列,这些元素以它们所处的位置为索引:从第一个到最后一个依次编号。前一节已经讲过 Python 字符串了,它本质上是字符组成的序列。列表非常适合利用顺序和位置定位某一元素,尤其是当元素的顺序或内容经常发生改变时。

列表可以由零个或多个元素组成,元素之间用逗号分开,整个列表被方括号所包裹:

```
>>> empty_list = []  # 空列表

>>> weekdays = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday']

在列表名后面添加[],并在括号里指定偏移量可以提取该位置的元素。第一个元素(最左侧)的偏移量为 0,下一个是 1,以此类推。

>>> weekdays[2]
'Wednesday'
```

第三个位置的元素是 'Wednesday',偏移量是 2。下面显示了如何把列表第五个元素改为 5。在同一列表中可以同时包含字符串和数字。列表也可以比较大小,规则与字

字符串比较类似。

```
>>> weekdays[4] = 5
>>> weekdays
['Monday', 'Tuesday', 'Wednesday', 'Thursday', 5]
>>> [1,2,3] < [1,2,4]  # 比较列表大小
True
```

列表也有加法和乘法运算。

```
>>> [1,2,3] + ['c','java','python']
[1, 2, 3, 'c', 'java', 'python']

>>> [1]*10          # 可以用作列表初始化
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
>>>
```

## 2.4 内置转换函数

布尔、整数、浮点数、复数、字符串和列表可以通过内置函数进行转换,如表 2-10 所示。

表 2-10 内置转换函数

函数名	含 义
bool()	根据传入的参数创建一个新的布尔值
int()	根据传入的参数创建一个新的整数
float()	根据传入的参数创建一个新的浮点数
complex()	根据传入的参数创建一个新的复数
str()	创建一个字符串
ord()	返回 Unicode 字符对应的整数
chr()	返回整数所对应的 Unicode 字符
bin()	将整数转换成二进制字符串
oct()	将整数转换成八进制字符串
hex()	将整数转换成十六进制字符串
list()	根据传入的参数创建一个新的列表

## 1. bool()函数

```
>>> bool()
False
>>> bool(1)
True
>>> bool(0)          # 数值0、空序列等值为False
False
>>> bool('str')
True
```

## 2. int()函数

```
>>> int() # 不传入参数时,得到结果0。
0
>>> int(3)
3
>>> int(3.6)
3
```

int()函数可以把字符串转换成数字。

```
>>> int('02')
2
>>> int('      35      ')
35
```

当字符串无法转换成数字,则报错。

```
>>> int('      3 5      ')
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    int('      3 5      ')
ValueError: invalid literal for int() with base 10: '      3 5      '
```

int()函数还可以带另一个参数,表示进制,如“8”表示字符串代表的数字是八进制。

```
>>> int('      35      ',8)
29
```

### 3. float()函数

```
>>> float() # 不提供参数时,返回 0.0
0.0
>>> float(3)
3.0
>>> float('3')
3.0
```

### 4. complex()函数

```
>>> complex() # 当两个参数都不提供时,返回复数 0j。
0j
>>> complex('1 + 2j') # 传入字符串创建复数
(1 + 2j)
>>> complex(1,2) # 传入数值创建复数 (1 + 2j)
(1 + 2j)
```

### 5. str()函数

```
>>> str() # 创建空字符串
''
>>> str(None)
'None'
>>> str('abc')
'abc'
>>> str(123)
'123'
```

### 6. ord()函数

```
>>> ord('a')
97
>>> ord('中')
20013 # 汉字'中'的 Unicode 码
```

### 7. chr()函数

```
>>> chr(97) # 参数类型为整数
'a'
```

### 8. bin()函数

```
>>> bin(3) # 0b 为默认
'0b11'
```



## 9. oct()函数

```
>>> oct(10)
'0o12'
```

## 10. hex()函数

```
>>> hex(15)
'0xf'
```

注意:bin()、oct()和 hex()函数返回的是字符串。

## 11. list()函数

```
>>> list() # 不传入参数,创建空列表
[]
>>> list('abcd') # 传入字符串,使用其元素创建新的列表
['a', 'b', 'c', 'd']
```

表达式是可以计算的代码片段,由常量、变量和运算符或函数按规则构成,返回运算结果。

```
>>> import math
# 计算表达式 cos(a(x+1)+b)/2, a 等于 2, x 等于 5, b 等于 3
>>> math.cos(2*(5+1)+3)/2
- 0.37984395642941066
>>> n=int(input())
5
>>> 1 if n%2==1 else 0 # 计算表达式:当 n 是奇数时为 1,偶数时为 0
1
```

## 2.5 语句

Python 语言常用的有赋值语句、if 语句和 for 语句。语句通常是一行一条语句。如一行中有多条语句,则用分号(;)分开;如语句太长要跨行时,可以用续行符(\)跨行表示一个语句。

### 2.5.1 赋值语句

基本形式的赋值语句是“变量=值”的形式。

**【例 2-1】** 基本赋值语句。

程序代码:

```
x=1
y=2
k=x+y
print(k)
```

程序输出：

```
3
```

Python 支持序列赋值,可以把赋值运算符“=”右侧的一系列的值,依次赋给左侧的变量。

```
>>> x,y=4,8
>>> print(x,y)
4 8

>>> a,b="34"
>>> a
'3'
>>> b
'4'
```

**【例 2-2】** 输入两个数字存入变量 a,b 中,交换 a,b 值。

注意:赋值语句“a,b=b,a”交换 a 和 b 的值,程序代码如下。

```
a=int(input())
b=int(input())
print(a,b)
a,b=b,a
print(a,b)
```

程序输入：

```
7
9
```

程序输出：

```
7 9
9 7
```

在之前的序列赋值中,赋值运算符左侧的变量个数和运算符右侧的值的个数总是相等的,如果不相等,则 Python 报错。Python 中使用带星号变量名,如 \*j 等,实现扩展序列赋值。

```
>>> i,*j=[1,2,3]
>>> i
1
>>> j
[2, 3]
```

可以把变量值一次赋给多个变量,这叫多变量赋值。

```
>>> a=b=c=5
>>> print(a,b,c)
5 5 5
>>> b=b+6
>>> print(a,b,c)
5 11 5
```



a=b=c 表示变量 a,b,c 都引用同一对象"5"

注意:当变量的值是数字时,修改其中一个变量,不会影响其他变量的值。  
赋值运算符可与“+”“-”“\*”和“\”组合在一起。如“x=x+y”可以写成“x+=y”。

```
>>> i=2
>>> i*=3
>>> i
6
```

2.5.2 if 语句

在 Python 中,if 语句可以是以下形式:

```
if 逻辑表达式:
    语句块 1
else:
    语句块 2
```

逻辑表达式的取值为 True 或 False。当逻辑表达式的取值为 True,执行语句块 1,逻辑表达式的取值为 False,执行语句块 2。缩进在 Python 中是具有语法意义的。例如,语句块 1 缩进的,表示它是属于 if 代码块。

【例 2-3】 输入一个数,判断它的奇偶性。

程序代码:

```
x=int(input())
if x%2==0:
    print("偶数")
else:
    print("奇数")
```

程序输入:

```
8
```

程序输出:

```
偶数
```

程序输入:

```
9
```

程序输出：

奇数

当  $x$  除以 2 的余数为 0 时,表达式  $x\%2==0$  取值为 True,否则为 False。请记住,“==”是用来做比较运算的,而“=”用于赋值。

**【例 2-4】** 为鼓励居民节约用水,自来水公司采取按用水量阶梯式计价的办法,即居民应交水费  $y$ (元)与月用水量  $x$ (吨)相关:当  $x$  不超过 15 吨时, $y=4x/3$ ;超过后, $y=2.5x-17.5$ ,小数部分保留 2 位。请编写程序实现水费的计算。

程序代码：

```
x=float(input())
if x <=15:
    y=4*x/3
else:
    y=2.5*x-17.5
print("{:.2f}".format(y))    # 格式化输出,具体见 2.6 节
```

程序输入：

9.5

程序输出：

12.67

程序输入：

15

程序输出：

20.00

程序输入：

21.3

程序输出：

35.75

### 2.5.3 for 语句

for 语句的一种形式如下：

```
for variable in 列表:
    语句块
```

for 后面的变量先被赋给列表的第一个值,并执行下面的语句块。然后变量被赋给列表中的第二个值,再次执行代码块。该过程一直持续,直到穷尽这个列表。语句块缩进表示它是属于 for 语句块。

**【例 2-5】** 遍历列表,打印列表[1,2,3,4]所有元素。

程序代码:

```
for i in [1,2,3,4]:
    print(i)
```

程序输出:

```
1
2
3
4
```

range()函数返回在特定区间的自然数序列。range()函数的用法:

```
range(start,stop,step)
```

start:计数从 start 开始,默认是从 0 开始。例如,range(5)等价于 range(0,5)。

stop:计数到 stop 结束,但不包括 stop。例如,list(range(0,5))是[0, 1, 2, 3, 4],没有 5。

step:步长,默认为 1。例如,range(0,5)等价于 range(0,5,1)。

```
>>> list(range(10))          # 从 0 开始到 10,不包括 10
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
>>> list(range(1,11))        # 从 1 开始到 11
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
>>> list(range(0, 30, 4))     # 步长为 4
[0, 4, 8, 12, 16, 20, 24, 28]
```

```
>>> list(range(0, -10, -1))   # 步长为负数
[0, -1, -2, -3, -4, -5, -6, -7, -8, -9]
```

sum()是Python的另一个内置函数,它可以求列表的和。如求和:1+2+3+...+10,可以用下面的程序实现。

```
>>> sum([1,2,3,4,5,6,7,8,9,10])
55
```

更简单的程序可以写成:

```
>>> sum(list(range(1,11)))
55
```

**【例 2-6】** 输入  $n(n \geq 10)$ , 求  $1+2+\cdots+n$  之和。

程序代码:

```
n=int(input())
s=sum(list(range(n+1)))
print(s)
```

程序输入:

10

程序输出:

55

程序输入:

100

程序输出:

5050

**【例 2-7】** 输入  $n(n \geq 5)$ , 求  $n!$

程序代码:

```
n=int(input())
factor=list(range(1,n+1))

f=1
for i in factor:
    f=f*i

print(f)
```

程序输入:

20

程序输出:

2432902008176640000

注意:这是一个很大的整数。

#### 2.5.4 列表推导式

列表推导式是从一个或者多个列表快速简洁地创建列表的一种方法,又被称为列表解析。它可以将循环和条件判断结合,从而避免语法冗长的代码,同时提高程序性能。会使用推导式,说明你已经超越 Python 初学者的水平了。也就是说,使用列表推导式更符合 Python 的编程风格。

列表推导式最简单的形式如下:

```
[ expression for item in iterable ]
```

下面的例子将通过列表推导创建一个整数列表：

```
>>> number_list = [number for number in [1,2,3,4,5]]
>>> number_list
[1, 2, 3, 4, 5]
```

在第一行中，第一个 number 变量为列表生成值，也就是说，把循环的结果放在列表 number\_list 中。第二个 number 为循环变量。其中第一个 number 位置处可以为表达式，请试试下面改编的例子：

```
>>> number_list=[2*number for number in [1,2,3,4,5]]
>>> number_list
[2, 4, 6, 8, 10]
```

列表推导把循环放在方括号内。这种例子和之前的不大一样，却是更为常见的方式。同样，列表推导也可以像下面的例子加上 if 条件：

```
[expression for item in iterable if condition]
```

下面产生元素是奇数的列表：

```
>>> number_list=[number for number in range(1,8) if number % 2 == 1]
>>> number_list
[1, 3, 5, 7]
```

**【例 2-8】** 求  $1+1/2+\dots+1/20$  之和。

如果能方便地产生列表： $[1, 1/2, 1/3, \dots, 1/20]$ ，则求和就很容易了。

程序代码：

```
print(sum([1/i for i in range(1,21)]))
```

程序输出：

```
3.597739657143682
```

**【例 2-9】** 求  $1-1/2+1/3-1/4+\dots$  之前 n 项和 ( $n \geq 10$ )。

如何产生这样的列表？

```
[1, -1/2, 1/3, -1/4, ...]
```

下面列表推导式产生列表： $[1, -1/2, 1/3, -1/4, 1/5]$

```
>>> [1/i if i%2==1 else -1/i for i in range(1,6)]
[1.0, -0.5, 0.3333333333333333, -0.25, 0.2]
```

“ $1/i$  if  $i \% 2 == 1$  else  $-1/i$ ”是条件表达式，表示奇数项为正，偶数项为负。

程序代码：

```
n=int(input())
print(sum([1/i if i%2==1 else -1/i for i in range(1,n+1)]))
```

程序输入：

```
100
```

程序输出：

```
0.688172179310195
```

**【例 2-10】** 求  $1 - 1/3 + 1/5 - 1/7 + \dots - 1/47 + 1/49$ 。

列表推导式的 if 条件和条件表达式可以同时使用吗？可以！

注意下列列表推导式：

```
>>> [i if i%4==1 else -i for i in range(1,50) if i%2==1]
[1, -3, 5, -7, 9, -11, 13, -15, 17, -19, 21, -23, 25, -27, 29, -31, 33, -35, 37, -39, 41, -43, 45, -47, 49]
```

程序代码：

```
print(sum([1/i if i%4==1 else -1/i for i in range(1,50) if i%2==1]))
```

程序输出：

```
0.7953941713587581
```

**【例 2-11】** 6 是一个幸运的数字,求  $6 + 66 + 666 + \dots + 666 \dots 666$  ( $n$  个 6,  $5 \leq n \leq 10$ ) 的和。

如何生成列表？

```
[6,66,666,...,666...666]
```

注意下面的表达式产生数字:66666

```
>>> int('6'*5)
66666
```

程序代码：

```
n=int(input())
print(sum([int('6'*i) for i in range(1,n+1)]))
```

程序输入：

```
7
```

程序输出：

```
7407402
```

## 2.6 格式化输出

**【例 2-12】** 输入 2 个正整数 lower 和 upper ( $lower \leq upper \leq 100$ ), 请输出一张取值范围为  $[lower, upper)$ , 且每次增加 2 华氏度的华氏—摄氏温度转换表, 小数部分保留一位。温度转换的计算公式:  $C = 5 \times (F - 32) / 9$ , 其中, C 表示摄氏温度, F 表示华氏温度。

程序代码：



```

lower,upper=input().split()      # 一行输入两个数,是字符串类型
lower,upper=int(lower),int(upper) # 字符串变成整数
for i in range(lower,upper,2):
    print(i,"{:.1f}".format(5*(i-32)/9))

```

程序输入:

```
30 40
```

程序输出:

```

30 - 1.1
32 0.0
34 1.1
36 2.2
38 3.3

```

format()函数是 Python 的内置函数,用来设置输出格式,详细介绍见后面章节。

```
>>> s="字符串格式化"
```

```

>>> "{0}".format(s)      # 默认格式化,字符串左对齐
'字符串格式化'

```

```

>>> "{0:>30}".format(s)   # 最小宽度 30,">"表示右对齐
'      字符串格式化'

```

```

>>> "{0:^30}".format(s)   # 最小宽度 30,"^"表示居中
'      字符串格式化      '

```

```
>>> x=3.14159
```

```
>>> y=2*x*3
```

```
>>> x
```

```
3.14159
```

```
>>> y
```

```
18.849539999999998
```

下面的语句中,0 和 1 表示 format()函数中的第 1 和第 2 个参数,.2f 表示小数部分保留两位,四舍五入。

```

>>> print("{0:.2f} {1:.2f}".format(x,y))
3.14 18.85

```

## 本章小结

1. 在 Python 中,不需要事先声明变量名和类型,直接赋值就可以创建任意类型的变量。
2. 整数、浮点数、复数、布尔和字符串是基本数据类型。
3. math 是一个数学库,包含大量的数学函数。
4. None 是一个特殊值,表示空值。
5. 列表是 Python 常用的数据类型,非常有用。
6. 语句主要有赋值、条件和循环三种。
7. 列表推导式是 Python 编程的有力工具,书写简洁,相对性能高。
8. format() 是一个内置函数,用于字符串格式化。

## 习 题

### 一、单选题

1. 下列数据类型中,Python 不支持的是\_\_\_\_\_。  
A. char                      B. int                      C. float                      D. list
2. Python 语句 `print(type(1j))` 的输出结果是\_\_\_\_\_。  
A. `<class 'complex'>`                      B. `<class 'int'>`  
C. `<class 'float'>`                      D. `<class 'dict'>`
3. Python 语句 `print(type(1/2))` 的输出结果是\_\_\_\_\_。  
A. `<class 'int'>`                      B. `<class 'number'>`  
C. `<class 'float'>`                      D. `<class 'double'>`
4. Python 语句 `print(type(1//2))` 的输出结果是\_\_\_\_\_。  
A. `<class 'int'>`                      B. `<class 'number'>`  
C. `<class 'float'>`                      D. `<class 'double'>`
5. Python 语句 `a=121+1.21; print(type(a))` 的输出结果是\_\_\_\_\_。  
A. `<class 'int'>`                      B. `<class 'float'>`  
C. `<class 'double'>`                      D. `<class 'long'>`
6. Python 语句 `print(0xA+0xB)` 的输出结果是\_\_\_\_\_。  
A. `0xA+0xB`                      B. `A+B`                      C. `0xA0xB`                      D. 21
7. Python 语句 `x='car'; y=2; print(x+y)` 的输出结果是\_\_\_\_\_。  
A. 语法错                      B. 2                      C. 'car2'                      D. 'carcar'
8. Python 表达式 `sqrt(4)*sqrt(9)` 的值为\_\_\_\_\_。  
A. 36.0                      B. 1296.0                      C. 13.0                      D. 6.0
9. 关于 Python 中的复数,下列说法错误的是\_\_\_\_\_。

- A. 表示复数的语法是 `real+image j`
- B. 实部和虚部都是浮点数
- C. 虚部必须加后缀 `j`，且必须是小写
- D. 方法 `real` 返回复数的实部

10. Python 语句 `print(chr(65))` 的运行结果是\_\_\_\_\_。

- A. 65
- B. 6
- C. 5
- D. A

11. 关于 Python 字符串, 下列说法错误的是\_\_\_\_\_。

- A. 字符即长度为 1 的字符串
- B. 字符串以 `\0` 标志字符串结束
- C. 既可以用单引号, 又可以用双引号创建字符串
- D. 在三引号字符串中可以包含换行、回车等特殊字符

## 二、填空题

1. Python 表达式 `10 + 5//3 - True+False` 的值为\_\_\_\_\_。

2. Python 表达式 `3**2**3` 的值为\_\_\_\_\_。

3. Python 表达式 `17.0/3**2` 的值为\_\_\_\_\_。

4. Python 表达式 `0 and 1 or not 2 < True` 的值为\_\_\_\_\_。

5. Python 语句 `print(pow(-3, 2), round(18.67, 1), round(18.67, -1))` 的输出结果是\_\_\_\_\_。

6. Python 语句 `print(round(123.84, 0), round(123.84, -2), floor(15.5))` 的输出结果是\_\_\_\_\_。

7. Python 语句 `print(int("20", 16), int("101", 2))` 的输出结果是\_\_\_\_\_。

8. Python 语句 `print(hex(16), bin(10))` 的输出结果是\_\_\_\_\_。

9. Python 语句 `print(abs(-3.2), abs(1-2j))` 的输出结果是\_\_\_\_\_。

10. 数学表达式:

$$\sin(35^\circ) + \frac{e^x - 15x}{\sqrt{x^4 + 1}} - \ln(7x)$$

的 Python 表达式为\_\_\_\_\_。

11. Python 语句 `x=True; y=False; z=False; print(x or y and z)` 的程序运行结果是\_\_\_\_\_。

12. Python 语句 `x=0; y=True; print(x>=y and 'A'<'B')` 的程序运行结果是\_\_\_\_\_。

13. 已知 `a=3; b=5; c=6; d=True`, 则表达式 `not d or a>=0 and a+c>b+3` 的值是\_\_\_\_\_。

14. Python 表达式 `16 - 2*5 > 7*8/2 or "XYZ"!="xyz" and not (10-6 > 18/2)` 的值是\_\_\_\_\_。

15. Python 语句 `print("hello" 'world')` 的结果是\_\_\_\_\_。

### 三、编程题

1. 输入一个正整数  $n$ , 求  $1+2+3+\cdots+n$  的累加和。

2. 计算分段函数。

本题目要求计算下列分段函数  $f(x)$  的值:

$$y=f(x)=\begin{cases} \frac{1}{x}, & x \neq 0 \\ 0, & x = 0 \end{cases}$$

在一行中输入实数  $x$ 。

在一行中按“ $f(x)=\text{result}$ ”的格式输出, 其中  $x$  与  $\text{result}$  都保留一位小数。

3. 阶梯电价。

为了提倡居民节约用电, 某省电力公司执行“阶梯电价”, 安装一户一表的居民用户电价分为两个“阶梯”: 月用电量 50 千瓦时(含 50 千瓦时)以内的, 电价为 0.53 元/千瓦时; 超过 50 千瓦时, 超出部分的用电量, 电价上调 0.05 元/千瓦时。请编写程序计算电费。

在一行中输入某用户的月用电量(单位: 千瓦时)。

在一行中输出该用户应支付的电费(元), 结果保留两位小数, 格式如: “cost= 应付电费值”; 若用电量小于 0, 则输出“Invalid Value!”。

4. 特殊  $a$  串数列求和。

给定两个均不超过 9 的正整数  $a$  和  $n$ , 要求编写程序求  $a+aa+aaa+\cdots+aa\cdots a$  ( $n$  个  $a$ ) 之和。

在一行中输入给出不超过 9 的正整数  $a$  和  $n$ 。在一行中按照“ $s = \text{对应的和}$ ”的格式输出。

5. 求奇数和。

本题要求计算给定的一系列正整数中奇数的和。

输入是在一行中给出一系列正整数, 其间以空格分隔。当读到零或负整数时(一定是序列最后一个值), 表示输入结束, 该数字不要处理。输入样例: 8 7 4 3 70 5 6 101 -1。在一行中输出正整数序列中奇数的和, 如样例输出 116。

6. 求交错序列前  $N$  项和。

本题要求编写程序, 计算交错序列  $1 - 2/3 + 3/5 - 4/7 + 5/9 - 6/11 + \cdots$  的前  $N$  项之和。

在一行中输入一个正整数  $N$ , 在一行中输出部分和的值, 结果保留三位小数。