第1章 走进Python编程

史秋实

目录

Contents


走进Python编程
Python简介
Python集成环境的安装
Windows环境下Python的安装
Windows环境下Pycharm的安装

01

Python简介

1.Python的起源与特点

Python于1989年由荷兰人吉多·范罗苏姆(Guido van Rossum)发明,1991年正式公布。它简洁易上手，开源、免费，是一种解释型高级程序设计语言。它支持命令式编程、面向对象编程、函数式编程，包含了完善且易于理解的标准库，这些功能与其他高级程序设计语言相似。

Python是一种解释型的、面向对象的、交互式的高级程序设计语言。它注重的是如何解决问题而不是编程语言的语法和结构。

Python的作者，Guido von Rossum，荷兰人。

1.Python的起源与特点

Python拥有庞大的社区支持，众多开发者编写、维护实现某种功能的库资源并共享到镜像平台，源源不断地扩充着第三方库，使得其拓展性高于其他程序设计语言。在第三方库的支持下，Python可以将功能拓展到各个领域。

2.Python的版本发展

历经30多年来的发展，Python的版本不断换代革新，目前主要发行有2.x版本和3.x版本，相比而言，Python 3.x版本提供了更好的特性、更清晰的语法、兼容性强、支持更多的第三方库，且Python官方已经在2020年停止了对Python 2.x版本的维护，因此对于新手学习和开发来说，推荐使用Python 3.x版本。

02

Python集成环境

1.Windows环境下Python的安装

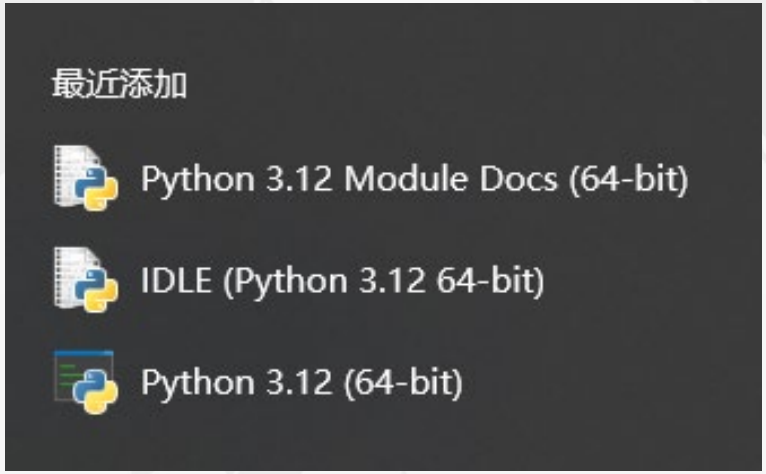
登录Python 官网：https://www.python.org/，单击Downloads，选择“Downloads Python 3.12.3”，如图1-1所示。页面下方有python其他版本，也可根据计算机的配置进行选择。

1.Windows环境下Python的安装

双击刚下载的文件“python-3.12.3-amd64.exe”，启动安装引导进程，勾选最下方的“Add python.exe to PATH”选项，如图1-2所示。然后单击Install Now或Customize installation均可进行安装，其中前者为默认安装方式，后者为自定义安装方式，自定义方式可以更改安装目录，设置软件使用权限等，一般选择“Install Now”的默认安装方式即可。

1.Windows环境下Python的安装

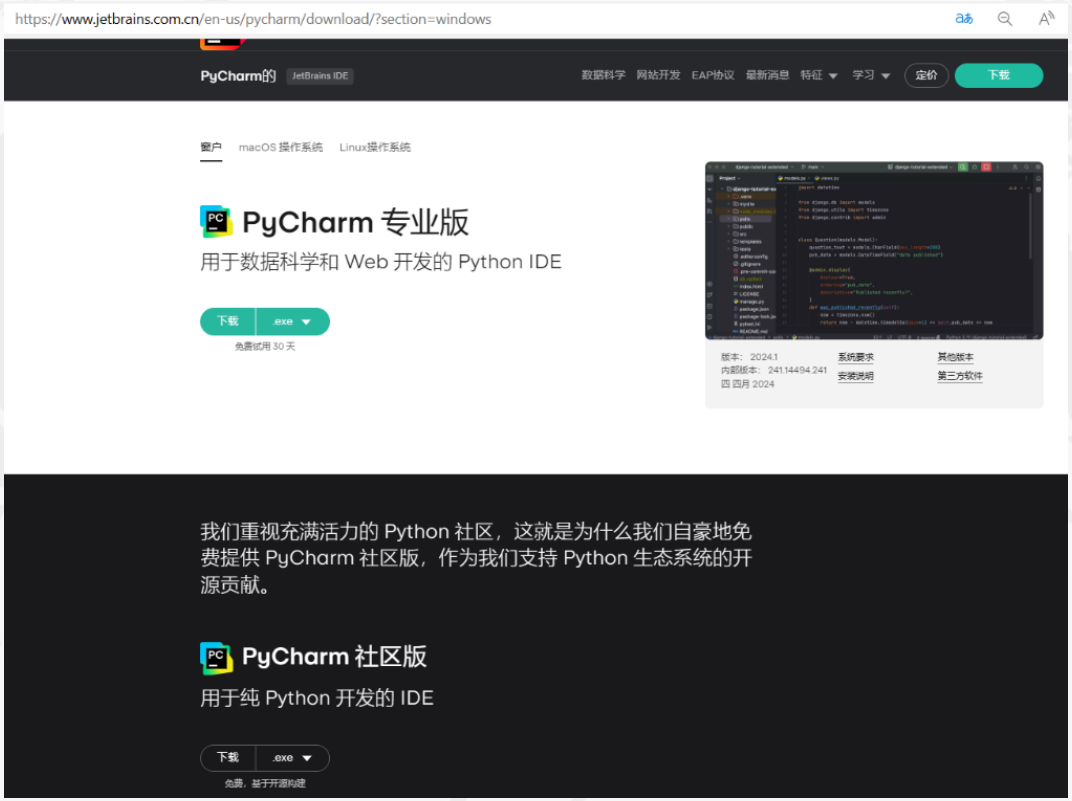
安装完成后，要检查python 3.12是否安装成功，在开始菜单可查看到已安装的Python程序。



2.Windows环境下Pycharm的安装

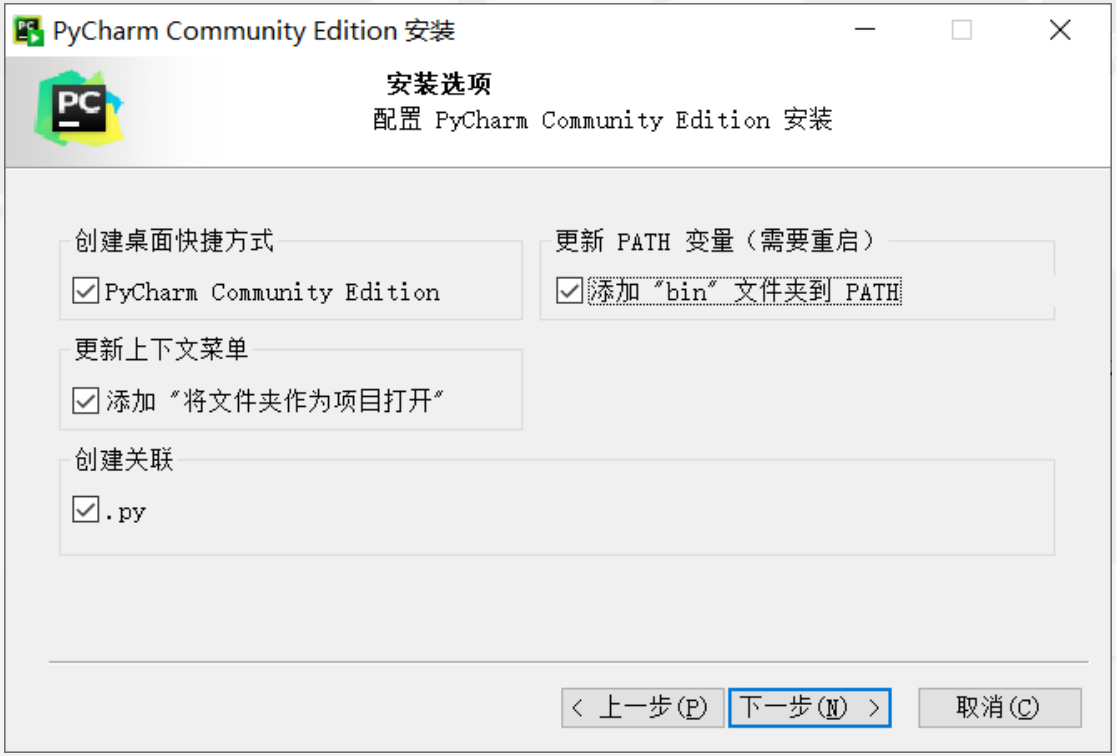
登录https://www.jetbrains.com/zh-cn/pycharm/，在图1-6的下载页面中，可以看到PyCharm提供了两个版本，Professional（专业版）和Community Edition（社区版）。其中Professional（专业版）的功能更为丰富和完备，适用于数据科学和Web开发，支持 HTML、JS和SQL，但需要付费。Community Edition（社区版）开源免费，仅适用于纯 Python 开发。

2.Windows环境下Pycharm的安装



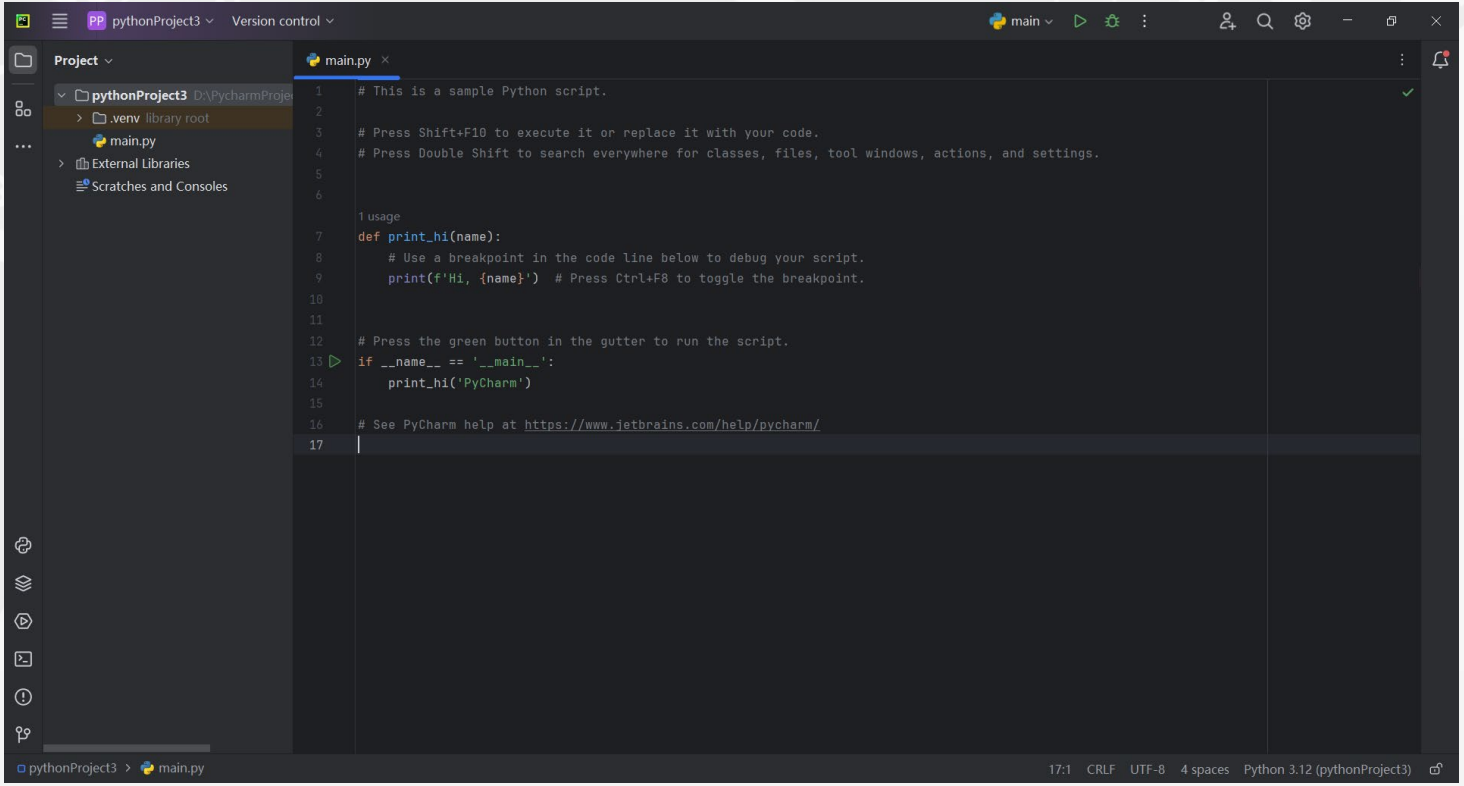
2.Windows环境下Pycharm的安装

下载好安装包“pycharm-community-2024.1.exe”后，双击直接安装即可，要注意的是图1-7中要勾选所有选项，为其配置好环境。



2.Windows环境下Pycharm的安装

Pycharm的工作界面



第2章 Python数据类型

史秋实

目录

Contents


Python数据类型
常量与变量
数值
数值的4种类型
数值运算
格式化输出
字符串
定义字符串
格式化输出字符串
字符串读取与切片
字符串的操作
数据类型转换
列表
创建与删除
索引与切片
列表的操作
元组
元组的创建
元组的操作
元组与列表的转换
集合
集合的创建
集合的操作方法
字典
字典的创建
字典的访问与修改
字典的操作

01

常量与变量

1.常量

在程序执行过程中需要大量的数据来参与运算，常量是不变的数据，用于描述客观事物的属性。人们常用的数字（0，1，2，1.5，-1，-2……）、中文文字（“计算机”、“程序”、“苹果”、“红色”……）、英文文字（“python”、“student” ……）等，均为常量。

常量的数据类型包括数值、字符串、列表、元组、集合、字典，下文将会对每一种数据类型进行详细讲解。

2.变量

程序设计中有时会需要变化的数据来提高程序的灵活性，因此出现了变量。变量是用户自定义的有名字的存储单元，其命名一般遵循以下规则：

（1）变量名可以包含数字或字母和下划线；

（2）变量名不能以数字开头；

（3）变量名区分大小写；

（4）变量名不宜太长，一般最好有一定的含义。

（5）保留字不能用作变量名。

2.变量

```
import keyword
print(keyword.kwlist)
```

Python保留字有35个：'False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield'。

02

数値

1.数值的4种类型

（1）整型（int）

数值数据中最常见的是整型数据，Python整数的取值范围是无限的，不管多大或者多小的数字，Python都能轻松处理。

（2）浮点型（float）

浮点型是带有小数部分的数值，并且可以有正负号。例如12.1、95.8或-22.0。除了常规的小数表示外，Python还支持使用科学计数法来表示浮点数。科学计数法的形式是mEe或mee，其中m是尾数（Mantissa），e是指数（Exponent）。例如，3.14e10表示3.14×10^{10}。

1.数值的4种类型

（3）复数型（complex）

Python中的复数类型用于表示具有实部和虚部的数值，形式为a+bj，其中a是实部，b是虚部，j是虚数单位。如1+2j表示一个实部为1，虚部为2的复数。Python的复数运算与数学的复数完全一致。

（4）布尔型（bool）

布尔类型是一种与逻辑相关的数据类型，只有True和False两个值。在Python中，布尔值是整数类型的一个子类，其中True等同于整数1，False等同于整数0。因此，布尔值可以相加，相加之后类型就会转换为int类型。

2.数值运算

Python中数据的运算，主要包括算术运算（如+、-、*、/），比较运算（如>、<、>=），位运算（如&、|、~、<<），以及逻辑运算（and，or，not），成员运算（in，not in）等，在运算时有一定的优先级，在同等优先级的情况下，按从左往右的顺序依次运算。运算符的详细使用使用说明见表2-1。

2.数值运算

运算符	功能描述	示例	优先级
-x	负号	表示负数-5	1
+	加法运算	3+5，结果为8	4
-	减法运算	3-5，结果为-2	4
*	乘法运算	3*5，结果为15	3
/	除法运算	5/3，结果为1.6666666666666667	3
//	整除运算	9//2，结果为4	3
%	取余运算	5%3，结果为2	3
**	乘方运算	3**5，表示3的5次方，结果为243	2
^	异或运算	5^3的结果为6，以二进制数运算	5
&	与运算	5&3结果为1，以二进制数运算	5
|	或运算	5 | 3的结果为7，以二进制数运算	5
<<	左移运算	5<<2的结果为20，以二进制数运算	5
>>	右移运算	5>>1的结果为2，以二进制数运算	5
~	按位取反	~3的结果为-4	1
==	等于	(a == b) 返回 False	6
!=	不等于	(a != b) 返回 True	6
>	大于	(a > b) 返回 False	6
<	小于	(a < b) 返回 True	6
>=	大于等于	(a >= b) 返回 False	6
<=	小于等于	(a <= b) 返回 True	6
in	如果在指定的序列中找到值返回True，否则返回False。	x in y x在y序列中,如果x在y序列中返回True。	6
not in	如果在指定的序列中没有找到值返回True，否则返回False。	x not in y x不在y序列中,如果x不在y序列中返回True。	6
is	如果两个变量是同一个对象，在同一个内存地址，则返回True	a=[1,2,3]，b=a，b is a 执行结果为True	6
is not	如果两个变量不是同一个对象，不在同一个内存地址，则返回True	a=[1,2,3]，b=[1,2,3], b is not a 执行结果为True	6
and	仅当a、b两者都为True时，结果才为True	0 and 5，结果为0	6
or	只要a、b二者之一为True时，结果为True	0 or 5，结果为5	6
not	如果x为True，返回False。如果x为False，它返回True。	not(0 and 5)，结果为True	6

3.格式化输出

（1）输出语句print()

要输出数据，必然要用到一个内置函数print()，其完整格式为：

print(values,sep,end,file,flush)

values：对象，表示可以一次输出多个对象，输出多个对象时，需要用逗号分隔；

sep：用来间隔多个对象，默认值是一个空格；

end：用来设定以什么结尾。默认值是换行符\n，我们可以换成其他字符串；

file：要写入的文件对象；

flush：输出是否被缓存通常决定于file，但如果flush关键字参数为True，流会被强制刷新。

一般来讲，前3个参数比较常用。

3.格式化输出

（2）输出格式

- **使用%操作符**

```
print("The number is %.2f" % b)
```

这是较旧的字符串格式化方式，借鉴自C语言，不推荐使用。

- **使用.format()方法**

```
print("The number is {:.2f}".format(b))
```

通过大括号{}作为占位符来进行格式化。

- **使用f-string**

在字符串前加上f或F，并在字符串内部使用{}来包含变量或表达式，f-string的方法从.format()方法演化而来，使用更为简便，建议使用。

```
print(f"The number is {b:.2f}")
```

3.格式化输出

（2）输出格式

%输出格式

符号	描述	符号	描述
%c	格式化字符及其ASCII码	%f	格式化浮点数字，可指定小数点后的精度
%s	格式化字符串	%e	用科学计数法格式化浮点数
%d	格式化整数	%E	作用同%e，用科学计数法格式化浮点数
%u	格式化无符号整型	%g	%f和%e的简写
%o	格式化无符号八进制数	%G	%f 和 %E 的简写
%x	格式化无符号十六进制数	%p	用十六进制数格式化变量的地址
%X	格式化无符号十六进制数（大写）		

03

字符串

1.定义字符串

字符串的定义比较灵活，一般使用英文状态下的一对引号表示，可以使用一对单引号、双引号或三引号表示，其中三引号通常用于定义多行字符。

例2-10：以下三种方式均可定义相同的字符串。

```
str1 = 'hello,world'
str2 = "hello,world"
str3 = '''hello,world'''
```

1.定义字符串

在定义字符串时，有时字符串中含有单引号，这时需要对它进行转义，即在前面加一个\。或者使用不同类型的引号也可以解决问题。

有时，也希望在字符串中包含换行符\n和回车符\r等特殊字符，这个时候就需要使用转义字符进行表示，常用的转义字符及其含义如表2-3所示。

转义字符	描述	转义字符	描述
\(在行尾时)	续行符	\n	换行
\\	反斜杠符号	\t	横向制表符
\'	单引号	\r	回车
\"	双引号	\v	纵向制表符
\b	退格(Backspace)	\f	换页

2.格式化输出字符串

字符串的格式化输出方式，与数值的输出方式类似，也包含%、.format()和f-string方法，由于f-string最为简单，在文后主要采用其作为输出方法。

例2-12：分别用%、.format()和f-string方法输出自我介绍。

```
name = "Amy"
age = 20
print("我叫%s，今年%d岁了。" %(name,age))
print("我叫{}，今年{}岁了。".format(name,age))
print(f"我叫{name}，今年{age}岁了。")
```

3.字符串读取与切片

字符串是不变的，但有时进行数据处理时需要提取字符串中的元素，因此，字符串的每个元素可以根据其索引位置进行读取，从左到右依次从0开始编号，也可以从右往左逆向编号，从-1开始，具体索引编号如表2-4所示。

正向索引	0	1	2	3	4	5	6	7	8	9
字符串	这	是	一	个	有	趣	的	字	符	串
反向索引	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

3.字符串读取与切片

（1）单个字符的提取

通过索引号（下标）提取某一个数据元素，操作方式：字符串名[索引号]。

例如，s=“这是一个有趣的字符串”。

要读取其中的“趣”字，可以使用s[5]，或s[-5]表示。

3.字符串读取与切片

（2）字符串切片

切片是Python序列数据的重要操作之一。适用于字符串、列表、元组、range对象等，通过指定索引范围获得一组有序的元素。

操作方式：字符串名[开始索引:结束索引:步长]。注意切片使用规则：前包后不包，步长默认是1，可以是负数。

例如，s=“这是一个有趣的字符串”。

读取“有趣的”，可以使用s[4:7]，或s[-4:-7]表示。

读取偶数位置的字符，使用s[::2]。

读取全部字符串，使用s[::]。

逆序显示字符串，使用s[::-1]。

4.字符串的操作

（1）字符串运算

运算符用于执行程序代码运算，会针对一个以上的操作数项目来进行运算。对于字符串的操作运算符主要有+（加法）、*（乘法）、in（成员包含判断）等，具体使用方法如表2-5所示。

运算符	描述	示例	执行结果
+	字符串的拼接	“py”+”thon”	‘python
*	字符串的重复	“hello”*3	“hellohellohello”
in	成员运算符，元素是否存在	‘t’ in python	True
not in	成员运算符，元素是否不存在	‘T’ not in ‘python’	True

4.字符串的操作

（2）内置函数对字符串的操作

Python提供了对字符串进行处理的函数，可以直接在字符串外部进行调用，常见的有len(s)，用于计算字符串s的长度。字符串的内置函数如表2-6所示。

函数及使用	描述	示例	执行结果
len(s)	返回字符串s的长度	len(“0123456789”)	10
hes(s)	返回整数s的十六进制形式	hex(425)	0x1a9
oct(s)	返回整数s的八进制形式	oct(425)	0o651
chr(s)	s为Unicode编码，返回其对应的字符	chr(65)	A
ord(s)	s为字符，返回其对应的Unicode编码	ord(“A”)	65

4.字符串的操作

（3）字符串的操作方法

字符串在处理文本内容时候使用较频繁，其包含了多种方法可以高效使用。

方法	说明
s.isalpha()	如果 s 只包含字母，则返回 True
s.isnumeric()	如果 s 只包含数字，则返回 True
s.isspace()	如果 s 只包含空格，则返回 True
s.isalnum()	如果 s 只包含字母或数字则返回 True
s.isdigit()	如果 s 只包含数字则返回 True，不包括汉字数字，罗马数字、小数
s.istitle()	如果 s 是标题化的(每个单词的首字母大写)则返回 True
s.islower()	如果 s的所有字符都是小写，则返回 True
s.isupper()	如果 s的所有字符都是大写，则返回 True
s.capitalize()	把s的第一个字符大写
s.title()	把s的每个单词首字母大写
s.lower()	转换 s 中所有大写字符为小写
s.upper()	转换 s 中的小写字母为大写

4.字符串的操作

（3）字符串的操作方法

字符串在处理文本内容时候使用较频繁，其包含了多种方法可以高效使用。

方法	说明
s.swapcase()	翻转 s 中的大小写
s.startswith(str)	检查s是否是以str开头，是则返回 True
s.endswith(str)	检查s是否是以str结束，是则返回 True
s.find(str, start,end)	检测str是否包含在s中，如果start和end指定范围，则检查是否包含在指定范围内，用于返回第一次出现的索引值，如不存在返回-1
s.rfind(str, start=0,end=len(s))	类似于find()，返回最后一次出现的索引值。
s.index(str, start=0,end=len(s))	跟find()方法类似，不过如果str不在s会报错
s.rindex(str, start=0,end=len(s))	类似于index()，不过是从右边开始
s.replace(old_str, new_str,num)	把s中的old_str替换成new_str，如果 num 指定，则替换不超过 num 次

4.字符串的操作

（3）字符串的操作方法

字符串在处理文本内容时候使用较频繁，其包含了多种方法可以高效使用。

方法	说明
s.lstrip()	去除s左边（开始）的空白字符
s.rstrip()	去除s右边（末尾）的空白字符
s.strip()	去除s左右两边的空白字符，当填写长度大于等于2的字符串的时候，则首尾的所有字符串内有的字符都要删除。
s.split(str=“ ”, num)	以 str 为分隔符拆分 s，如果 num 有指定值，则仅分隔 num + 1 个子字符串，str 默认包含 ‘\r’，‘\t’，‘\n’ 和空格
s.join(seq)	以 s 作为分隔符，将 seq 中所有的元素（的字符串表示）合并为一个新的字符串

5.数据类型转换

Python中数据类型较为灵活，在定义变量时，不需要定义其数据类型，可以直接使用，Python会自动识别常用的类型，如a=2，解释器会自动识别到a为int型，如果需要将a转变为字符串，则使用b=str(a)，可以直接将a的int型数据转变为str型，并赋值到变量b。用type(a)可以查看变量的数据类型。

5.数据类型转换

函数	描述
int(x [,base])	将x转换为一个整数
float(x)	将x转换到一个浮点数
complex(real [,imag])	创建一个复数
str(x)	将对象x转换为字符串
repr(x)	将对象x转换为表达式字符串
abs(x)	返回对象x的绝对值
round(x,n)	返回浮点数x四舍五入为n位小数位数的值
eval(str)	计算在字符串中的有效Python表达式,并返回一个对象
tuple(s)	将序列s转换为一个元组
list(s)	将序列s转换为一个列表
set(s)	转换为可变集合
dict(d)	创建一个字典。d必须是一个(key,value)元组序列。
frozenset(s)	转换为不可变集合
chr(x)	将一个整数转换为一个字符
ord(x)	将一个字符转换为它的整数值
hex(x)	将一个整数转换为一个十六进制字符串
oct(x)	将一个整数转换为一个八进制字符串

04

列表

1.列表的创建与删除

创建一个列表，只需要把逗号分隔的不同的数据项用方括号括起来即可。

创建列表常用的方法有以下4种。

（1）创建空列表

```
list1 = []
```

（2）创建有元素的列表

```
list2 = [1，2，"a"，"b"，"c"，"d"，"e"]
```

（3）将字符串转为列表

```
list3=list("python")
```

（4）将字符串分割为列表

```
list5=list("You can do whatever you put your mind to.".split())
```

1.列表的创建与删除

删除列表对象，可以用del list来实现。del可以通用于对任意对象的删除，后文不再赘述。

2.列表的索引与切片

列表的索引与切片方法与字符串相同。

例2-23：已知列表list1=["重庆","北京","上海","天津",2,5,78,3,45,19]，要求：

(1)读取list1中最后一个元素

(2)读取list1中第2个元素

(3)读取list1中的所有元素

(4)读取list1中的前3个元素

(5)读取list1中最后3个元素

(6)读取list1中的奇数位置的元素

(7)逆向输出list1中的所有元素

```
list1=["重庆","北京","上海","天津
",2,5,78,3,45,19]
print(list1[-1])
print(list1[1])
print(list1[::])
print(list1[:3])
print(list1[-3::])
print(list1[1::2])
print(list1[::-1])
```

3.列表的操作

列表的运算与字符串类似，对列表进行运算如表2-9所示。以两个列表为例：list1=[1,2,3]，list2=['a','b','c']。

运算符	描述	示例	执行结果
+	连接两个列表，得到一个新列表	list1+list2	['a', 'b', 'c', 1, 2, 3]
*	用于列表和整数相乘，表示序列重复，返回新列表	list1*3	[1, 2, 3, 1, 2, 3, 1, 2, 3]
in	成员运算符，元素是否存在	2 in list1	True
not in	成员运算符，元素是否不存在	2 not in list2	True

3.列表的操作

Python提供了对列表进行处理的函数，可以直接在列表外部进行调用。常用的内置函数如表2-10所示。

内置函数	说明
max(list)	返回列表中的最大值
min(list)	返回列表中的最小值
sum(list)	返回列表中元素之和
len(list)	返回列表元素个数
sorted(list,reverse=)	对列表元素进行排序，True为降序，False为升序

3.列表的操作

Python提供了很多列表的操作方法。

方法	说明
list.append(x)	在列表list的尾部追加元素x
list.extend(listA)	将列表listA中所有元素追加至列表list的尾部
list.insert(index, x)	在列表list的index位置处插入x，该位置之后的所有元素自动向后移动，索引加1
list.remove(x)	删除列表list中第一个值为x的元素
list.pop([index])	删除并返回列表list中下标为index的元素，index默认为-1
list.index(x)	返回x 在列表list中第一个次出现的索引位置
list.count(x)	返回x在列表list中出现的次数
list.reverse()	对列表list的所有元素进行逆序
list.sort(key=None, reverse=False)	对列表list中的元素进行排序，key用来指定排序规则，reverse为False表示升序，True表示降序
list.clear()	清空列表
list.copy()	复制列表

05

元组

1.元组的创建

元组可以通过圆括号()或逗号来创建。即使只有一个元素的元组，也需要在元素后面添加逗号，否则括号会被当作运算符使用。

例如，定义一个元组：tuple1=(1,2,3)，tuple2=(1,)。

删除元组：del tuple1

2.元组的操作

元组支持的内置函数、计算与列表相同，但由于元组的元素是不可改变的，故不支持涉及元素变化的操作，如append()、extend()、insert()、remove()、pop()、clear() 、sort()等, 仅支持count()和index()操作。

3.元组与列表的转换

元组与列表之间可以互相转换，使用list()函数可以把元组转换成列表，使用tuple()函数也可以把列表转换成元组。

例如，对列表list1=[1,2,3]，将其转换为元组：tuple1=tuple(list1)。

对元组tuple1=(1,2,3），将其转换为元组：list1=list(tuple1)。

06

集合

1.集合的创建

集合是一种无序可变的、不包含重复元素的集合数据类型。

其基本功能包括关系测试和消除重复元素。

集合用花括号{}表示，元素之间用逗号分隔。

创建集合：s1 = {1, 2, 3, 4}

也可以将列表转换为集合。

2.集合的操作方法

集合支持多种数学上的集合运算，如并集、交集、差集，集合还支持添加元素、移除元素等操作，对于集合s1= {1, 2, 3, 4, 5}，s2= {'a','b'}，可以进行操作如表2-12所示。

方法	描述	示例	运行结果
len()	计算集合元素个数	len(s1)	5
union()	返回两个集合的并集	s1.union(s2)	{1, 2, 3, 4, 5, 'a', 'b'}
intersection()	返回集合的交集	s1.intersection(s2)	set()
difference()	返回多个集合的差集	s1.difference(s2)	{1, 2, 3, 4, 5}
add()	为集合添加元素	s1.add("a")	{1, 2, 3, 4, 5, 'a'}
clear()	移除集合中的所有元素	s1.clear()	set()
update()	给集合添加元素	s1.update("c")	{1, 2, 3, 4, 5, 'c'}
remove()	移除指定元素，如果元素不存在，则会发生错误	s1.remove(1)	{2, 3, 4, 5}

07

字典

1.字典的创建

字典属于容器类对象，可存储任意类型对象。其中包含若干元素，每个元素包含“键”和“值”两部分，这两部分之间使用冒号分，表示一种对应关系。不同元素之间用逗号分隔，所有元素放在一对大括号中。

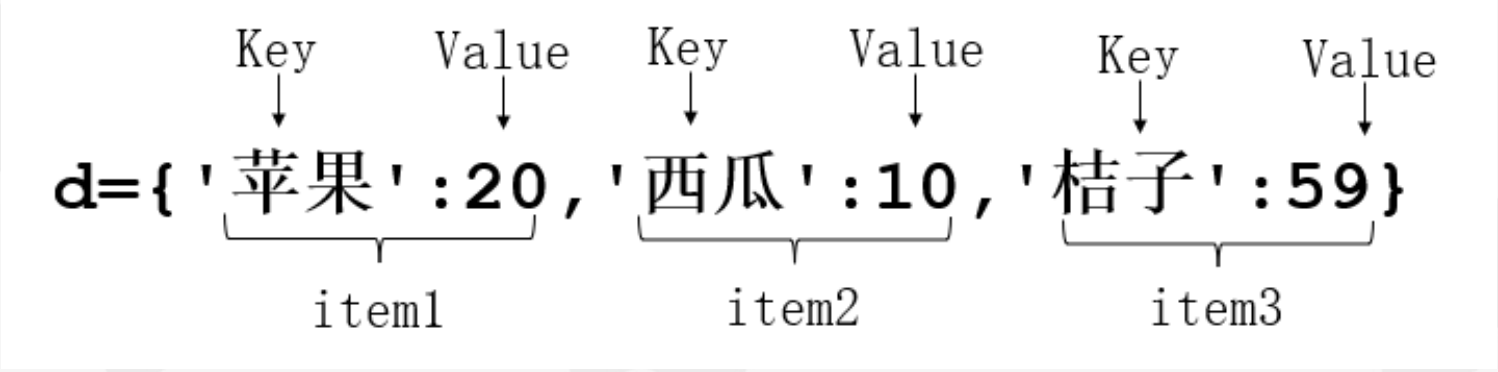
字典是无序的、可变的；字典中的“键”不允许重复，“值”是可以重复的；通过键来读取元素。字典的键可以是一个字符串、数值、逻辑值，字典的值可以是单个的值，也可以是列表、元组等。

1.字典的创建

字典的创建格式如下所示：

d = {key1 : value1, key2 : value2, key3 : value3 }。

创建一个存储水果数量的字典，如图2-1所示。



2.字典的访问与修改

（1）访问字典元素

格式：字典名[键]

例2-35：对水果数量字典d={'苹果':20,'西瓜':10,'桔子':59}，查看苹果的数量。

```
d={'苹果':20,'西瓜':10,'桔子':59}
print(d['苹果'])
```

输出结果为：20

2.字典的访问与修改

（1）访问字典元素

格式：字典名[键]

例2-35：对水果数量字典d={'苹果':20,'西瓜':10,'桔子':59}，查看苹果的数量。

```
d={'苹果':20,'西瓜':10,'桔子':59}
print(d['苹果'])
```

输出结果为：20

2.字典的访问与修改

（2）添加或修改字典元素

格式：字典名[键]=值，已存在键的值会被修改，不存在的则会增加键值对。

例2-36：在上例中，将苹果的数量更改为30，直接使用赋值语句即可。

```
d['苹果']=30
```

增加香蕉的数量为40，也可以直接用赋值语句实现。

```
d['香蕉']=40
```

执行print(d)，得到结果：{'苹果': 30, '西瓜': 10, '桔子': 59, '香蕉': 40}

2.字典的访问与修改

（3）删除字典元素

del命令可以删除元素，也可以删除整个字典。

例如，del d['香蕉']，可以删除水果字典d中香蕉的键值对。

del d，可以删除整个水果字典对象d。

3.字典的操作

Python提供了很多字典的操作方法，如表2-13所示。

方法	功能
dict.keys()	获得键的迭代器
dict.values()	获得值的迭代器
dict.get(k,[default])	获得k（键）对应的值，不存在则返回default
dict.items()	获得由键和值组成元组做为元素的列表
dict.pop(k[,d])	删除k（键）对应的”键：值”对
dict.update(adict)	从另一个字典更新字典元素的值，如不存在，则添加此元素
dict.clear()	清空字典
dict.copy()	复制字典
dict.fromkeys(iter,value)	以列表或元组中给定的键建立字典，默认值为value