第3章 Python数据类型

知识导图

本章知识导图如图 3-0 所示。

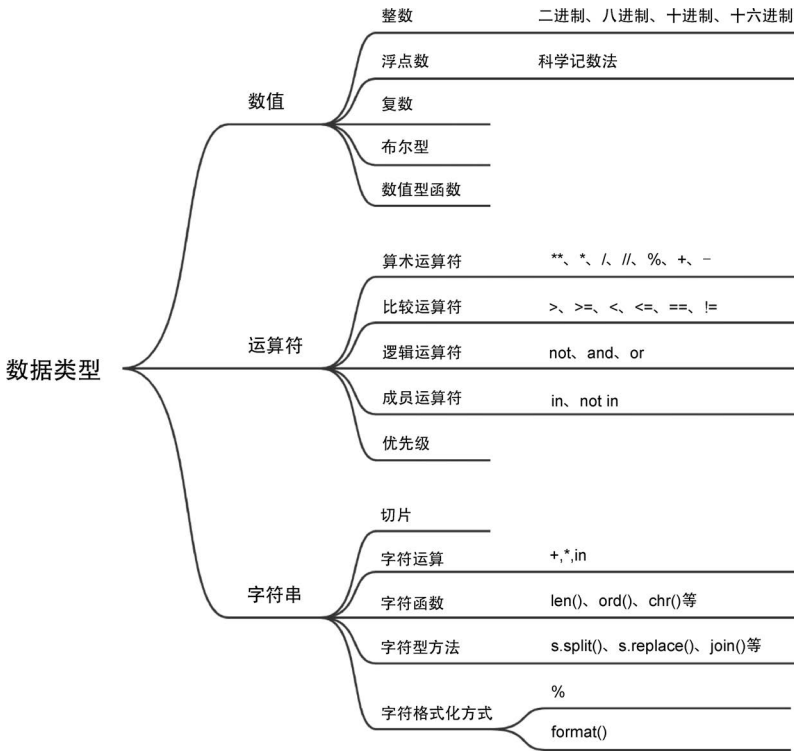

图 3-0　第 3 章知识导图

问题导向

- Python 语言中数据的表现形式有哪几种？
- 每种数据类型有什么特征？
- 数据的运算符号有哪些？如何构成表达式？
- 复杂的数据计算用什么来实现？
- 各种数据类型如何实现相互转换？
- 字符型数据有哪些函数和方法？

重点与难点

- 数据类型的使用。
- 运算符的灵活应用。
- 字符串切片。

“等闲识得东风面，万紫千红总是春”，春风成就了春天的绝色，那程序世界中的无限风光又是谁造就的呢？当然非数据莫属了。Python世界有两大原住居民，分别是数值型和字符型。下面通过“三天打鱼，两天晒网”的故事来认识下Python的数据成员。

【案例 3-1】 三天打鱼，两天晒网 。

“三天打鱼，两天晒网”是曹雪芹《红楼梦》中第九回：“因此也假说来上学，不过三日打鱼，两日晒网，白送些束修礼物与贾代儒。”现在常用来比喻一个人对学习或工作没有恒心，经常中断，不能长久坚持。下面使用Python来模拟实现。假设最初的能力值为1.0，学习一天，能力会比前一天提高1%；懒散一天，能力会比前一天下降1%。由此可以得知，在“三天打鱼，两天晒网”情况中的能力计算公式为

$$(1+0.01)^3\times(1-0.01)^2$$

用Python来实现：

```
fish = (1 + 0.01) ** 3
net = (1 - 0.01) ** 2
result = fish * net
print(result)
print(result > (1 + 0.01))
```

运行结果：

```
1.0097980101000001
False
```

从“三天打鱼，两天晒网”的结果1.0097980101000001<1+0.01可以看出，如果一个人不能持之以恒地学习，最终将一无所获。

希望大家每天都不要停止学习的脚步，日积月累，坚持不懈，终有一天会得到回报。

3.1 认识数据类型


观看视频

从案例3-1中可以看到有许多数据，在Python中，将表示数字或数值的类型称为数据类型。而案例中的数据有整数，也有小数，在Python中分别对应数据类型中的整型(int)和浮点型(float)。

在数学中除了整数、小数之外，还有复数，故Python中还有一个数据类型为复数类型(complex)。

除此之外，还有一个比较特殊的整型：布尔型(bool)。值为1时表示真(True)，值为0时表示假(False)。

3.1.1 整数

案例中出现的数字 1、2、3 这样的数据称为整型，与数学中整数的概念一致。Python 3 中整型数据的长度不受机器字长的影响，它的取值范围只与计算机的内存有关。也就是说，只要计算机的内存足够大，无论整型的长度为多少，都不用担心溢出问题。

在计算机中，经常用 4 种进制来表示整型：二进制、八进制、十进制、十六进制。默认的是十进制，如果想要用其他进制表示，需要加上引导符号。

二进制：以 0B 或 0b 开头。

八进制：以 0O 或 0o 开头。

十六进制：以 0X 或 0x 开头。

例如：

```
a = 0b101
print('a 的十进制表示：%d' % a)
b = 0o101
print('b 的十进制表示：%d' % b)
c = 0x101
print('c 的十进制表示：%d' % c)
d = 101
print('d 的十进制表示：%d' % d)
```

运行结果：

```
a 的十进制表示:5
b 的十进制表示:65
c 的十进制表示:257
d 的十进制表示:101
```

不同的进制之间还可以相互转换。例如：

dec(x)：将数值 x 转换为十进制。

bin(x)：将数值 x 转换为二进制。

oct(x)：将数值 x 转换为八进制。

hex(x)：将数值 x 转换为十六进制。

int(x)：将字符串 x 转换为整数。

```
a = 10
print('a 的二进制为:',  bin(a))
print('a 的八进制为:',  oct(a))
print('a 的十六进制为:',  hex(a))
print('二进制 101 的整数值为:',  int('101',2))
```

运行结果：

```
a 的二进制为: 0b1010
a 的八进制为: 0o12
a 的十六进制为: 0xa
二进制 101 的整数值为: 5
```

3.1.2 浮点数

像案例 3-1 中出现的 0.01 这样的带小数的数据称为浮点数。Python 的浮点数一般以十进制表示，由整数和小数两部分组成，如 0.0、2.34、0.00000051、3.14159 都是浮点数。

对于非常大或者非常小的浮点数可以用科学记数法表示。例如，0.00000051 可以表示成 5.1e-7，314000 可以表示成 3.14E5。

Python 中的浮点数是双精度的，每个浮点数占 8B(64b)，包括 52b 存储尾数，11b 存储阶码，1b 存储符号，故浮点数的取值范围为－1.8e308～1.8e308。超出这个范围将视为无穷大(inf)或者无穷小(－inf)。

Python 中最长可以输出浮点数的 17 位数字，但是计算机只能保证 15 位数字的精度，对于超出 17 位的浮点数会产生截断。一旦产生截断，结果就会产生误差。如平时人们都会认为：0.1＋0.2 的结果是 0.3，但是实际执行的结果却是 0.30000000000000004。

```
>>> 0.1 + 0.2
0.30000000000000004
>>> 3 * 0.1
0.30000000000000004
```

3.1.3 复数

格式为 2＋3j、5.6＋7.8j 这样的数据称为复数。一个复数由“实部”和“虚部”两部分组成，实部是一个实数，虚部是一个实数后加 j 或 J 组成，虚部不能单独存在。

获取一个复数实部的方法是调用属性 real，获取虚部的方法是调用属性 imag。

将一个数据转换为复数的函数是 complex()。例如：

```
a = complex(2,4)
b = 6
print(a)
print('a 的实部是:', a.real,  ', a 的虚部是:', a.imag)
print(complex(b))
```

运行结果：

```
(2+4j)
a 的实部是: 2.0,  a 的虚部是:4.0
(6+0j)
```

3.1.4 布尔型

布尔型只有 True 和 False 两个值，本质上来说，布尔型其实是一种特殊的整数，True 对应非 0，False 对应 0。

任何对象都具有布尔属性，在 Python 中，以下数据的值均为 False。

(1) None。

(2) False。

（3）任何为 0 的数字类型：0、0.0、0j。

（4）任何空字符、空列表、空字典：""、( )、[ ]、{}。

（5）用户定义的类实例，如果类中定义了__bool__() 或者__len__() 方法，并且方法返回 0 或者布尔值 False。

以下结果均为 False。

```
bool()
bool('')
bool(0)
bool([])
```


观看视频

3.2 运算符

在案例 3-1 中有几种运算符号：+、-、*、**、>。通过这些运算符号可以将两个不同的数据组合起来得到一个运算结果。由此可见，运算符是告诉编译程序执行指定运算操作的符号，是针对操作数进行运算。例如，表达式 1+0.01 中，1 和 0.01 均为操作数，+是运算符。Python 中运算符非常丰富，功能也很强大。

3.2.1 数值运算符

数值运算符是一类对数值型操作数有效的运算符。按照不同的功能，又可以分成算术运算符、赋值运算符、比较运算符、逻辑运算符等。

1. 算术运算符

以 $x=2$，$y=9$ 为例，对算术运算符进行说明，如表 3-1 和表 3-2 所示。

表 3-1 算术运算符

操作符	描　　述	示　　例
+	加，$x+y$ 为 x 与 y 之和	$x+y$ 结果为 11
-	减，$x-y$ 为 x 与 y 之差	$x-y$ 结果为 77
*	乘，$x*y$ 为 x 与 y 之积	$x*y$ 结果为 18
/	除，x/y 为 x 除以 y 之商，结果为浮点数	x/y 结果为 0.2222222222222222
//	整数除，$x//y$ 为 x 除以 y 之商的整数部分	$x//y$ 结果为 0
%	取余运算，$x\%y$ 为 x 除以 y 的余数部分	$x\%y$ 结果为 2
**	幂，$x**y$ 为 x 的 y 次方	$x**y$ 结果为 512
	开方运算，当 y 是小数时，如 10 ** 0.5 结果是 $\sqrt{10}$	

表 3-2 二元操作符

操作符	描　　述	操作符	描　　述
$x+=y$	相当于 $x=x+y$，x 的结果为 11	$x*=y$	相当于 $x=x*y$，x 的结果为 18
$x-=y$	相当于 $x=x-y$，x 的结果为-7	$x/=y$	相当于 $x=x/y$，x 的结果为 0.22
$x//=y$	相当于 $x=x//y$，x 的结果为 0	$x\%=y$	相当于 $x=x\%y$，x 的结果为 2
$x**=y$	相当于 $x=x**y$，x 的结果为 512		

Python 中的算术运算符既支持对相同类型的数值进行运算，也支持对不同类型的数值

进行混合运算。在混合运算时，Python 会强制将数值进行临时类型转换。遵循原则是将简单的数据类型转换为相对复杂的那一种数据类型。

布尔类型进行算术运算时，会将值视为 0 或 1。

整型与浮点型进行混合运算时，会将整型转换为浮点型。

其他类型与复数运算时，会将其他类型转换为复数。

```
1 + True                    #结果为 2
1 * 2.0                     #结果为 2.0
1 + (2 + 3j)                #结果为(3 + 3j)
```

【案例 3-2】 时间转换。

给定一个以 s 为单位的时间 t，要求用“$<H>:<M>:<S>$”的格式来表示这个时间。$<H>$表示时间，$<M>$表示分钟，而$<S>$表示秒，它们都是整数且没有前导的“0”。例如，若 $t=0$，则输出“0:0:0”；若 $t=3661$，则输出“1:1:1”。

案例分析：输入的数字为一个总秒数，需要将总秒数拆成几小时几分钟几秒钟，可通过 1h=60min，1min=60s 的规则进行拆分。

输入：

输入只有一行，是一个整数 t（$0 \leqslant t \leqslant 86\,399$），如 5436。

输出：

输出只有一行，是以“$<H>:<M>:<S>$”的格式所表示的时间，不包括引号，如 1:30:36。

解题思路：

给定的数字 t 是一个总秒数，所以有

$$t = S + M \times 60 + H \times 60 \times 60$$

反推过来则是：小时 $H = t//3600$，余数是剩下的 $S + M \times 60$，用余数继续可整除 60 得到分钟 M=余数//60，余数则为秒 S。

可以按照这个顺序：整除→取余→整除→取余。

代码实现：

```
t = eval(input())
H = t // 3600
t = t % 3600
M = t // 60
S = t % 60
print("{}:{}:{}".format(H,M,S))
```

2. 赋值运算符

“=”为赋值运算符。案例 3-1 中出现的 fish=(1+0.01) ** 3 就是一个赋值语句，其作用就是将一个表达式或对象赋给等号左边的变量。所有的运算符都可以与“=”组合起来形成一个赋值语句，包括二元操作中，如“+=”。

3. 比较运算符

比较运算符是比较符号左右两边的操作数，运算结果是一个布尔值。以 $x=2$，$y=9$

为例，对比较运算符进行说明，如表 3-3 所示。

表 3-3 比较运算符

操 作 符	描 述	示 例
$x==y$	判断 x 与 y 是否相等	$x==y$ 的值为 False
$x!=y$	判断 x 是否不等于 y	$x!=y$ 的值为 True
$x>y$	判断 x 是否大于 y	$x>y$ 的值为 False
$x<y$	判断 x 是否小于 y	$x<y$ 的结果为 True
$x>=y$	判断 x 是否大于或等于 y	$x>=y$ 的结果为 False
$x<=y$	判断 x 是否小于或等于 y	$x<=y$ 的结果为 True

【案例 3-3】 猜数游戏。

程序中给出一个固定的数字，用户通过键盘输入一个数，如果这个数与程序给出的数字相同，则输出“恭喜你猜对了!”；如果比程序给出的数字大，则输出“太大了!”；如果比程序给出的数字小，则输出“太小了!”。

案例分析：

输入：用户猜的数字。

处理：将用户猜的数字与程序给出的数字进行大小比较，可使用多分支结构来进行判断。

输出：根据比较结果输出不同的结果。

```
y = 50
x = eval(input('请输入你猜的数字:'))
if x == y:
  print('恭喜你猜对了!')
elif x > y:
  print('太大了!')
else:
  print('太小了!')
```

运行结果：

```
请输入你猜的数字:40
太小了!
```

4. 逻辑运算符

逻辑运算符可以把多个条件表达式连接起来，形成更为复杂的条件，如表 3-4 所示。

表 3-4 逻辑运算符

操 作 符	描 述
and	与，左操作数为 False 时返回左操作数，否则返回右操作数或计算结果
or	或，左操作数为 True 时返回左操作数，否则返回右操作数结果
not	非，若操作数为 False 返回 True，否则返回 False

例如：

```
>>> 2 + 3  and 3 + 5                    # 左为 True,返回右操作数 3 + 5 的值
8
```

```
>>> 2 - 2 and 3 + 5                    # 左为 False,返回左操作数 2 - 2 的值
0
>>> 2 + 3 or 3 + 5                     # 左为 True,返回左操作数 2 + 3 的值
5
>>> 2 - 2 or 3 + 5                     # 左为 False,返回右操作数 3 + 5 的值
8
>>> not (3 + 5)
False
>>> not False
True
```

【案例 3-4】 “剪刀石头布”游戏。

小明和小红想玩“剪刀石头布”游戏。在这个游戏中,两个人同时说“剪刀”“石头”或“布”,压过另一方的为胜者。规则是:“布”胜过“石头”,“石头”胜过“剪刀”,“剪刀”胜过“布”。

案例分析:通过输入两人的选择,程序自己判断输赢,并输出相应的结果。

输入:两个数,分别代表小明和小红的选择,范围为{0,1,2},用逗号隔开。0 表示石头,1 表示布,2 表示剪刀。这两个数分别表示两个人所选的物品。例如 0,2,则表小明出石头,小红出剪刀。

输出:如果前者赢,输出“小明胜”;如果后者赢,输出“小红胜”;如果是平局,输出“平局”。

代码实现:

```
a, b = eval(input())
if (a == 0 and b == 1)or (a == 1 and b == 2) or (a == 2 and b == 0) :
  print('小红胜')
elif (a == 1 and b == 0)or (a == 2 and b == 1) or (a == 0 and b == 2) :
  print('小明胜')
elif a == b:
  print('平局')
```

【案例 3-5】 闰年的判断。

从生活常识我们了解到,如果一个年份数字能被 4 整除但是不能被 100 整除,或者这个年份的数字能被 400 整除,那么这一年是闰年,否则是平年。编写程序实现输入一个年份,判断它是否是闰年。

案例分析:

输入:需要判断的年份。

处理:满足两种情况的年份是闰年,第一种是能被 4 整除但是不能被 100 整除,这两个条件需要同时成立,可以用逻辑运算符 and 将两个条件表达式连接。第二种情况是能被 400 整除。这两种情况是只要满足其中的一个,指定年份就是闰年,两种情况可以用逻辑运算符 or 来连接。

输出:输出当前的判断结果。

代码实现:

```
year = int(input('请输入年份:'))
if(year % 4 == 0 and year % 100 != 0) or year % 400 == 0:
  print('{}年是闰年'.format(year))
else:
  print('{}年不是闰年'.format(year))
```

运行结果：

```
请输入年份:2020
2020 年是闰年
```

再次运行，输入数字 2200，运行结果：

```
请输入年份:2200
2200 年不是闰年
```

5. 成员运算符

成员运算符用于判断左操作数是否存在于右侧的序列中。序列可以是字符串、列表、字典等。成员运算符具体描述如表 3-5 所示。

表 3-5 成员运算符

操 作 符	描 述
in	左操作数是否在右操作数中存在，例如，'ab' in 'abcd' 返回值为 True
not in	左操作数是否不在右操作数中，例如，'ac' not in 'abcd' 返回值为 True

【案例 3-6】 “三天打鱼，两天晒网”周末版。

一年 365 天，一周 5 个工作日，每天都努力打鱼，进步 1%。

一年 365 天，一周 2 个休息日，每天都休息晒网，退步 1%。

初始能力为 1，这种生活方式，一年后的能力如何呢？

案例分析：如何确定一周中的 5 个工作日，2 个休息日？

一周 7 天可以用天数对 7 取余，结果为 0～6，设其中 0 和 6 相当于周六、周日，即休息日，转换为表达式则是，取余运算结果在[0,6]这些值中的天数就是休息日。

```
power = 1                          #初始能力
for i in range(365):
  if i % 7 in [0,6]:               #取余结果为 0 或 6
    power = power * (1 - 0.01)
  else:
    power = power * (1 + 0.01)
print('一年后的能力为:{:.2f}'.format(power))
```

运行结果：

```
一年后的能力为:4.63
```

3.2.2 运算符优先级

对于表达式(2+3)×4−5×2，运算的顺序是什么样的呢？小学数学告诉我们应该先算

括号内的，再算括号外的，先算乘除，再算加减。这种优先顺序就是运算符的优先级，乘、除的优先级高于加、减，括号的优先级高于乘除。

常见运算符优先级由高到低的顺序如下：乘方 **，按位取反～，正负号＋x、－x，乘、除、求余 *、/、%，加、减＋、－，比较＜、＜＝、＞、＞＝、!＝、＝＝，成员判断 not in、in，逻辑运算 not、and、or。

一般来说，同一优先级的运算符在运算时是按照从左往右的顺序结合，如 2＋3－4 的运算顺序是：解释器先运算 2＋3，得到结果 5 再与运算符结合执行 5－4。

而赋值运算符的结合则相反，按照从右往左的顺序，如 $x=y=z$，会先将 z 的值赋给 y，再将 y 的值赋给 x。

3.2.3 常用数值函数

常用的数值函数如表 3-6 所示。

表 3-6 常用数值函数

函 数 名	描 述
abs(x)	绝对值函数，x 的绝对值
divmod(x，y)	商余，(x//y，x%y)，同时输出商和余数
pow(x，y[，z])	幂余，(x ** y)%z，[..]表示参数 z 可省略
round(x[，d])	四舍五入，d 是保留小数位数，默认值为 0
max(x_1，x_2，…，x_n)	最大值，返回 x_1，x_2，…，x_n 中的最大值，n 不限
min(x_1，x_2，…，x_n)	最小值，返回 x_1，x_2，…，x_n 中的最小值，n 不限
int(x)	将 x 变成整数，舍弃小数部分
float(x)	将 x 变成浮点数，增加小数部分
complex(x)	将 x 变成复数，增加虚数部分

例如：

```
>>> abs( - 10.01)
10.01
>>> divmod(10, 3)
(3, 1)
>>> pow(3, pow(3, 99), 10000)
4587
>>> round( - 10.123, 2)
 - 10.12
>>> max(1, 9, 5, 4 3)
9
>>> min(1, 9, 5, 4 3)
1
>>> int(123 - 45)
123
>>> int("123")
123
>>> float(12)
12.0
>>> float("1.23")
1.23
>>> complex(4)
(4 + 0j)
```

3.2.4 math 库

对数字类型的数据不仅可以做简单的基本运算，还可以进行更加复杂的数学运算，如求对数、三角函数、高斯误差等。在 Python 中要实现这些运算需要用到数学模块 math 库。math 库是一个比较成熟的库，不属于 Python，但是可以通过 import 方法将 math 库导入进来。

导入第三方库的方法如下。

方法一：import math

这是直接导入法，使用库的函数时需要加上前缀，如 math.sqrt(x)表示对 x 求平方根。

方法二：import math as m

命别名导入法，导入进来的 math 库赋予新的名字 m，使用库中 sqrt()函数时可用 m.sqrt(x)。

方法三：from math import *

从库中导入指定方法，使用库中函数时可直接调用，不需要加别名，如 sqrt(x)。

使用 math 库会使计算效率更高效，math 库中提供了许多数学函数，包含 4 个数学常数(圆周率 pi、自然对数 e、正无穷大 inf、非浮点数标记 nan)和 44 个函数。44 个函数又分为 4 类，其中，16 个数值表示函数、8 个幂对数函数、16 个三角对数函数、4 个高等特殊函数。常见的 math 库函数如表 3-7 所示。

表 3-7 math 库函数

函 数	数学表示	描 述
sqrt(x)	$\sqrt{x}$	x 的平方根
sin(x)	$\sin x$	x 的正弦
cos(x)	$\cos x$	x 的余弦
tan(x)	$\tan x$	x 的正切
asin(x)	$\arcsin x$	x 的反正弦
acos(x)	$\arccos x$	x 的反余弦
atan(x)	$\arctan x$	x 的反正切
log(x)	$\ln x$	x 的自然对数，以 e 为底数
log$_{10}$(x)	$\log_{10} x$	x 的常用对数，以 10 为底
exp(x)	e^x	e 的 x 次方
fabs(x)	$\lvert x \rvert$	返回 x 的绝对值
floor(x)	$\lfloor x \rfloor$	向下取整
ceil(x)	$\lceil x \rceil$	向上取整
pow(x,y)	x^y	x 的 y 的幂
gcd(x,y)		x 和 y 的最大公约数

【案例 3-7】 求两点之间的距离。

已知平行线上两点 A 和 B 的坐标，$A(x=22,y=33)$，$B(x=62,y=105)$，编写程序求出 A 和 B 两点之间距离。

案例分析：应用数学的知识可以知道，两点间的距离公式为$\sqrt[2]{(x_2-x_1)^2+(y_2-y_1)^2}$，编写程序可以先分别给 x_1、x_2、y_1、y_2 赋值，再用数学函数 pow()求平方，再用函数 sqrt()求

平方根，则可得到两点间的距离。由于用到了数学函数，所以在程序的开头部分需要导入math库。

实现代码：

```
import math
x1,y1 = 22,33
x2,y2 = 62,105
distx = math.pow((x2 - x1),2)
disty = math.pow((y2 - y1),2)
dist = math.sqrt(distx + disty)
print('A 和 B 的距离是：%.2f' % dist)
```

运行结果：

```
A 和 B 的距离是:82.37
```

3.3 字符串


观看视频

Python中的字符串是由一对单引号(' ')、一对双引号(" ")或一对三引号(''' ''')括起来的字符序列。

单引号：'单引号括起来的单行，可以使用 "双引号" 作为字符串的一部分'

双号号："双引号括起来的单行，可以用'单引号'作为字符串的一部分"

三引号：'''三引号括起来的多行，可以用'单引号'
"双引号" 作为字符串的一部分，
也可以换行'''

如果非要在单引号(或双引号)括起来的字符串中包含单引号(双引号)，可使用转义字符反斜杠(\)对字符串中的单引号(双引号)进行转义处理，使得转义字符与特殊字符组成新的含义。

```
#合法的字符串
'hello "python"'
"Life is short ,I 'm learning Python"
"Hello \"Python\""

#非法的字符串
"'Hello "Python""
'Life is short ,I 'm learning Python'
```

提示：在字符串定义时需要遵守以下几条规则。

(1) 字符串可以使用单引号或双引号来定义，但是最好在一个文件中统一使用同一种，避免混合使用的情况。

(2) 如果在字符串中包含某种引号时，那么优先使用另一种形式的引号来定义字符串，尽量不使用转义字符。

3.3.1 字符串的基本操作

1. 基本操作符

Python 提供了众多字符串的基本操作符，见表 3-8。

表 3-8 字符串操作符

操 作 符	描 述
$x+y$	将两个字符串 x 和 y 拼接成一个字符串
$x*n$ 或 $n*x$	将字符串 x 复制 n 次
x in y	如果 x 是 y 的子串，则返回 True，否则为 False
>，>=，<，<=，!=，==	两个字符串按 ASCII 码值比较大小

例如：

```
>>>'abc' + '123'
'abc123'
>>>'a' * 5
'aaaaa'
>>>'a' in '123abc'
True
>>>'a' > 'b'
False
>>>'bc'>'bac'
True
```

2. 字符索引

字符串中的多个字符，都会按顺序给出一个编号，这个编号就是索引。例如，s＝'Python'，字母'P'的编号为 0，具体索引如表 3-9 所示。

表 3-9 字符串索引号

字符	P	y	t	h	o	n
编号	0	1	2	3	4	5
反向编号	−6	−5	−4	−3	−2	−1

Python 中的字符串，可以通过索引取出其中的一个字符或一段字符子串，但是不支持动态修改。例如，s[1] 可取出字符'y'。但是如果想通过 s[1]＝'a'将'y'修改成'a'，则是非法操作。

在字符串中通过索引取出一个或一段字符子串的操作非常灵活，取其中一段的操作称为切片。切片的操作非常多样，具体如下。

$s[n:m]$：表示从字符串 s 中取索引号从 n 到 $m-1$ 的字符子串，其中不包含索引号 m 的字符。

$s[n:]$：省略结束索引号，表示切取索引号从 n 到最后一个字符的子串。

$s[:m]$：省略开始索引号，表示切取索引号从 0 到索引号为 $m-1$ 的子串。

$s[-n:-m]$：表示从字符串 s 中切取索引号从倒数 n 到倒数 $m-1$ 的字符子串。

$s[::]$：省略开始和结束索引号，表示切取整个字符串。

$s[::-1]$：获得整个字符串的逆序。

例如：

```
>>> s = 'HelloWorld'
>>> s[0]
'H'
>>> s[-1]
'd'
>>> s[2:6]
'lloW'
>>> s[:5]
'Hello'
>>> s[5:]
'World'
>>> s[-5:-2]
'Wor'
>>> s[-5:]
'World'
>>> s[::]
'HelloWorld'
>>> s[::-1]
'dlroWolleH'
```

3.3.2 字符串的处理函数

Python 提供了许多内置函数，其中有 6 个与字符串处理相关的函数，如表 3-10 所示。

表 3-10 字符串函数

函 数 名	描 述
len(x)	返回字符串 x 的长度，整数
str(x)	返回任意数据类型 x 所对应的字符串形式
chr(x)	返回 Unicode 编码 x 对应的单字符
ord(x)	返回单字符 x 对应的 Unicode 编码
hex(x)	返回整数 x 对应的十六进制数的小写形式的字符串
oct(x)	返回整数 x 对应的八进制数的小写形式的字符串

例如：

```
>>> s = 'HelloWorld'
>>> len(s)
10
>>> str(3-5)
'3-5'
>>> chr(65)
'A'
>>> chr(9801)
'♉'
>>> ord('A')
65
>>> hex(18)
'0x12'
>>> oct(18)
'0o22'
```

【案例 3-8】 凯撒密码。

凯撒密码是古罗马凯撒大帝用来对军事情报进行加密的算法，它采用替换方法将每一个英文字符循环替换为字母表序中该字符后面第三个字符，对应关系如下。

原文：a b c d e f g h i j k l m n o p q r s t u v w x y z

密文：d e f g h i j k l m n o p q r s t u v w x y z a b c

假设原文字符为 P，对应密文为 S，则两者的关系为

$$S=(P+3)\%26$$

$$P=(S-3)\%26$$

加密的程序设计过程如下。

(1) 输入一串原文字符串(假设信息全是小写字母 a～z)。

(2) 将字符串中的每一个字符拿出来进行转换，规则为 $S=(P+3)\%\ 26$。由于字符没有办法直接与数字进行加法运算，所以需要先将字符转换为相应的 Unicode 编码，再进行加法运算，算完之后得到的 Unicode 编码又需要再转为相应的字符。模 26 是因为字母表总共为 26 个字母，起始字母为'a'，所以是当前字母与字母'a'的差去取模运算，结果再加上'a'的编码值得到的才是加密后的字母编码值。

(3) 将加密码后得到的新字符串进行输出。

代码实现：

```
F = input('请输入需要加密的信息:')
for P in F :
  if 'a'<= P <= 'z':
    S = chr((ord(P) - ord('a') + 3) % 26 + ord('a'))
  else:
    S = P
  print(S, end = '')
```

运行结果：

```
请输入需要加密的信息:Life is short, I use Python!
Llih lv vkruw, I xvh Pbwkrq!
```

3.3.3 字符串的处理方法

在 Python 的解释器内部，所有数据类型都采用面向对象方式实现，封装成一个类。字符串就是其中的一个类。每一个类里面都有许多自己的方法和属性，想要使用类的方法和属性，需要用 a. b()的方式来进行调用，其中，a 指明相应类创建的对象。例如，创建的一个字符串变量 s ='Python'，变量 s 就是一个字符串类的实例化对象。b()指的是这个类中相应的方法。类里面的方法其实就是根据特定功能创建的函数，对外调用时一般称之为“方法”。字符串方法如表 3-11 所示。

表 3-11 字符串方法

函 数 名	描 述
str. lower()	返回字符串 str 的副本，全部小写。原字符串不变
str. upper()	返回字符串 str 的副本，全部大写。原字符串不变

续表

函 数 名	描 述
str.islower()	若 str 全是小写,则返回 True,否则返回 False
str.isprintable()	若 str 所有字符都是可打印的,则返回 True,否则返回 False
str.isnumeric()	若 str 所有字符都是数字,则返回 True,否则返回 False
str.isspace()	若 str 所有字符都是空格,则返回 True,否则返回 False
str.endswith(suf,start[,end]])	str[start:end]是以 suf 结尾返回 True,否则返回 False
str.startswith(suf,start[,end]])	str[start:end]是以 suf 开始返回 True,否则返回 False
str.split(sep,maxsplit=-1)	返回一个列表,由 str 根据 sep 进行分隔得到的元素组成
str.count(sub[,start[,end]])	返回 str[start:end]中子串 sub 出现的次数
str.replace(old,new[,count])	返回 str 的副本,所有 old 子串被 new 替换,前 count 个被替换
str.center(width[,fillchar])	字符串 str 居中,有 fillchar 则在 str 左右填充
str.strip([chars])	返回 str 的副本,去掉左右两侧的 chars 列出的字符
str.zfill(width)	返回 str 的副本,长度为 width,不足部分填 0
str.format()	返回字符串的格式化排版,常用于 print()格式化输出中
str.join(iterable)	返回新串,将 str 加到 iterable 的每个字符中间

示例代码:

```
>>> s = 'Python'
>>> s.lower()
'python'
>>> s.upper()
'PYTHON'
>>> a = '1234567'
>>> a.isnumeric()
True
>>> b = 'a,b,a,c'
>>> b.split(',')
['a', 'b', 'a', 'c']
>>>> b.count('a')
2
>>> b.replace('a','abc')
'abc,b,abc,d'
>>> 'Python'.center(20, '=')
'=======Python======='
>>>'  P y t h o n  '.strip()
'P y t h o n'
>>>'*'.join('Python'.)
'P*y*t*h*o*n'
```

【案例 3-9】 利用 center()函数输出如图 3-1 所示的菱形图形,宽度为 9,行数为 9。

分析:

(1) 先输出正三角形,为 5 行,可用循环实现。假定行数为 i,初始值为 0,则每一行的'* '的个数为 $2i+1$,让'* '居中打印。

(2) 再输出倒三角形,为 4 行。假定行数为 j,初始值为 4,则每一行的'* '个数为 $2j-1$,同样用 center()函数实现居中打印。

代码实现:

```
    *
   ***
  *****
 *******
*********
 *******
  *****
   ***
    *
```

图 3-1 菱形输出

```
for i in range(5):
  print(('*' * (2 * i + 1)).center(9))
for j in range(4,0,-1):
  print(('*' * (2 * j - 1)).center(9))
```

3.3.4 字符串的格式化方法

字符串可以通过 format()函数进行格式化处理。

例如，小明同学想用一个程序输出他每一天的运动量，如 2019-12-12：跑步 1 小时，行程 9.85 千米。下画线中的内容每天都会发生改变，可以用特定的函数运算得到结果，填充到指定的位置，最终形成格式化的字符串。

Python 提供了两种字符串格式化方法，一种是类 C 语言中 printf()函数的格式化方法，另一种是采用专门的 format()格式化方法。

1. 使用%符号进行格式化

使用%符号对字符串格式化的形式：

```
'%[对齐][正号][0][宽度][.精度]指定类型'%变量
```

其基本思想是：第一个%号表示格式开始标志，单引号外面的%后面跟待格式化的变量。

[对齐]：－：表示左对齐，＋：表示右对齐。

[正号]：＋，对正数加上正号，仅对数值有效。

[0]：当指定宽度超出数值的宽度时，用 0 对多余位置进行填充，对数值有效，对字符无效。

[宽度]：指出当前字符串输出的宽度，如果对应字符串长度超过宽度值，则使用字符串数的实际长度。

[精度]：浮点数小数部分的精度或字符串的最大输出长度。即保留几位小数，字符串输出长度。

指定类型：如表 3-12 所示。

表 3-12 字符串格式化指定类型

类型	说　明	类型	说　明
%s	字符串(采用 str()的显示)	%x	十六进制整数
%r	字符串(采用 repr()的显示)	%e	指数(基底为 e)
%c	单个字符(Unicode 编码对应字符)	%E	指数(基底为 E)
%b	二进制整数	%f、%F	浮点数
%d	十进制整数	%g	指数(e)或浮点数(由长度决定)
%i	十进制整数	%G	指数(E)或浮点数(由长度决定)
%o	八进制整数	%%	百分数

采用这种方式进行字符串格式化时，要求被格式化的内容和格式字符之间必须一一对应。

```
a = 'Python'
b = 3-1415926
```

```
print('% +20s 右对齐' % a)              # 指定宽度 20,右对齐输出字符串
print('% -20.4s 左对齐' % a)            # 指定宽度 20,精度为 4,左对齐输出字符串
print('% +020.2f' % b)                  # 加上正号,宽度 20,0 填充,输出精度为 2 的浮点数
print('65 的 Unicode 编码:%c' % 65)
```

运行结果：

```
      Python 右对齐
Pyth                左对齐
+0000000000000003-14
65 的 Unicode 编码:A
```

2. 使用 format()函数进行格式化

format()函数的基本使用：

```
<模板字符串>.format(逗号分隔的参数)
```

模板字符串由一系列槽组成，用来控制修改字符串中嵌入值出现的位置，其基本思想是将 format()函数中逗号分隔的参数按照逗号关系替换到模板字符串的槽中。槽用花括号{}表示，如果花括号中没有序号，则按照出现的顺序替换。

参数的序号是从 0 开始编号，调用 format()函数后会得到一个新的字符串。上面的输出可以由以下语句实现：

```
'{}:跑步{}小时,行程{}千米'.format('2019-12-12', '1', '9.85')
'{0}:跑步{1}小时,行程{2}千米'.format('2019-12-12', '1', '9.85')
```

用变量来存储具体数值，上述代码可变成：

```
day = '2019-12-12'
hours = 1
dis = 9.85
print('{}:跑步{}小时,行程{}千米'.format(day, hours, dis))
print('{0}:跑步{1}小时,行程{2}千米'.format(day, hours, dis))
```

format()函数中的槽除了包括参数序号，还可以包含格式控制的信息，具体格式信息如下。

{<序号>:<格式控制标记>}

格式控制标记如表 3-13 所示。

表 3-13 格式控制标记

:	<填充>	<对齐>	<宽度>	<,>	.<精度>	<类型>
序号	用于填充的单个字符个数	<左对齐 >右对齐 ^居中对齐	槽的设定输出宽度	数字的千分位分隔符，适用于整数和浮点数	浮点数小数部分的精度或字符串的最大输出长度	整数类型：b,c,d,o,x,X 浮点型：e,E,f,%

<填充>、<对齐>、<宽度>是三个相关字段。

<宽度>指出当前槽输出字符的宽度，如果槽中对应参数长度超过<宽度>值，则使用参数的实际长度。

如果参数长度小于<宽度>值，则用<填充>中的字符进行填充，<填充>参数省略，则默认用空格字符填充。

<对齐>是指参数在指定<宽度>中输出时的对齐方式，分别用＜、＞、^对应左对齐、右对齐、居中对齐。

例如：

```
>>> print('=号填充左对齐输出{:=<30}'.format('Python'))
=号填充左对齐输出 Python========================

>>> print('空格填充右对齐输出:{:>30}'.format('Python'))
空格填充右对齐输出:                         Python

>>> print('等号填充居中对齐输出:{:=^30}'.format('Python'))
等号填充居中对齐输出: ============ Python ============

>>> print('长度不够,等号填充右对齐输出:{:>5}'.format('Python'))
长度不够,等号填充右对齐输出:Python
```

千分位分隔符“,”用于显示数字类型的千分位分隔符。例如：

```
>>> print('{:-^20,}'.format(123456789))
----123,456,789-----
```

整数类型的后缀包括 b、c、d、o、x、X，这些后缀的含义分别如下。

b：输出整数的二进制值。

c：输出整数的 Unicode 编码。

d：输出整数的十进制值。

o：输出整数的八进制值。

x：输出整数的小写十六进制值。

X：输出整数的大写十六进制值。

例如：

```
>>> print('十进制{0:d},\n 二进制是:{0:b},\n Unicode 编码是{0:c}\n 八进制是{0:o},\n 小写十六进制是{0:x},\n 大写十六进制是{0:X}'.format(65))
```

运行结果：

```
十进制 65,
二进制是:1000001,
Unicode 编码是 A
八进制是 101,
小写十六进制是 41,
大写十六进制是 41
```

浮点型的后缀包括e、E、f、%，这些后缀的含义分别如下。

e：输出浮点数对应的小写字母e的指数形式。

E：输出浮点数对应的大写字母E的指数形式。

f：输出浮点数的标准浮点形式。

%：输出浮点数的百分形式。

浮点数输出时尽量使用<精度>表示小数部分的宽度，即保留几位小数。这样有助于更好地控制输出格式。

例如：

```
>>> print('e指数形式:{0:.2e},\nE指数形式:{0:.2E},\n浮点数:{0:.2f},\n百分制:{0:.2%}'.
format(0.034)
```

运行结果：

```
e指数形式:3.40e-02,
E指数形式:3.40E-02,
浮点数:0.03,
百分制:3.40%
```

【案例 3-10】 小明的记账单。

小明学会了用Python做各种计算编程，决定对自己的购物做个记账单。记账单功能如下。

(1) 记录每天的消费内容，包括日期、物品名、数量、单价。

(2) 统计总的数量、总金额，以及每天的平均消费金额。

(3) 记账格式整齐、美观。

(4) 打印记账单。

```
n1,n2,n3 = 7,9,10
price1,price2,price3 = 12,7.28,11.5
good1,good2,good3 = '奶茶','苹果','零食'
date = '2020年5月'
total_num = n1 + n2 + n3
total_amount = n1 * price1 + n2 * price2 + n3 * price3
print('--'*9+'小明的周账单'+'--'*9)
print('购物日期\t名称\t数量\t单价\t总价')
print(date+'1日\t'+good1+'\t'+str(n1)+'\t'+str(price1)+'\t'+str(n1 * price1))
print(date+'2日\t'+good2+'\t'+str(n2)+'\t'+str(price2)+'\t'+str(n2 * price2))
print(date+'3日\t'+good3+'\t'+str(n3)+'\t'+str(price3)+'\t'+str(n3 * price3))
print('----'*12)
print('总数量\t%d'%total_num)
print('总金额\t¥ %.2f元'%total_amount)
print('日均消费\t¥ %.2f元'%(total_amount/30))
```

运行结果：

```
-------------------------- 小明的周账单 --------------------------
购物日期          名称    数量    单价    总价
2020 年 5 月 1 日  奶茶    7       12      84
2020 年 5 月 2 日  苹果    9       7.28    65.52
2020 年 5 月 3 日  零食    10      11.5    115.0
------------------------------------------------------------------------
                                    总金额 ¥ 264.52 元
总数量   26
日均消费   ¥ 8.82 元
```

在前面的代码中出现了新的符号\n、\t，使用之后发现\n 的输出结果换行了。这是一种特殊的格式化控制字符，用来控制输出效果，以反斜杠(\)开头。常用转义字符如表 3-14 所示。

表 3-14　转义字符

转 义 字 符	转义字符含义	ASCII 码
\n	换行，光标移动到下一行首位	10
\a	蜂鸣，响铃	7
\b	回退，向后退一格	8
\f	换页	12
\r	回车，光标移到本行首字符位	13
\t	水平制表符	9
\v	垂直制表符	11
\0	NULL，什么都不做	0
\\	反斜杠字符(\)	92
\'	单引号字符(')	39
\"	单引号字符(")	34

【案例 3-11】 文本进度条。

进度条是计算机处理任务或执行软件中常用的增强用户体验的重要手段，能实时显示任务或软件的执行进度。print()函数结合字符串的格式化可以实现非刷新文本进度条和单行刷新文本进度条。

先按任务执行百分比将整个任务分成 100 个单位，每执行 n%就输出一次进度条，每一次输出包含进度百分比，完成的部分用(**)表示，未完成的部分用(--)表示。中间用一个小箭头(->)分隔。例如：

```
10 %[ ***** ->..... ..... ..... ..... ..... ..... ..... .....]
```

由于程序执行速度非常快，远超人眼的视觉感知，直接输出，看不出来效果，因而每一次输出时让计算机等待 ts，增强显示效果。而等待需要使用时间库 time 中的 sleep()方法。

非刷新文本进度程序代码如下。

```
import time
scale = 10
for i in range(scale + 1):
  a = '**' * i
  b = '..' * (scale - i)
  c = (i / scale) * 100
  print('{:<-3.0f} %[{}->{}]'.format(c,a,b))
  time.sleep(0.1)
```

输出结果：逐行实现 0%～100%的变化、如图 3-2 所示。

单行刷新文本进度：想要实现在单行中动态刷新，需要将所有的输出都固定在同一行，不断用后面新生成的字符串覆盖之前的输出，形成动态效果。可以利用特殊的格式化控制字符(\r)来实现，它的功能是光标移到本行首字符位。再将输出控制不换行，即在 print()函数中设置 end 属性，这样可以将所有的输出固定在一行。

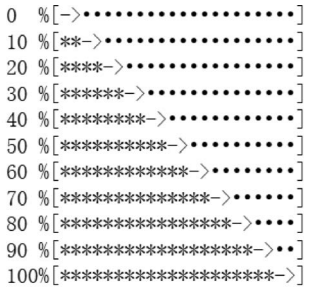

图 3-2 逐行输出

```
import time
scale = 10
for i in range(scale+1):
  a = '**' * i
  b = '..' * (scale - i)
  c = (i / scale) * 100
  print('\r{:<3.0f}%[{}->{}]'.format(c,a,b),end='')
  time.sleep(0.1)
```

输出效果：单行动态实现 0%～100%的变化。

```
100%[********************->]
```

3.4 实践应用

【案例 3-12】 成绩单管理。

小明学会了 Python 的各种数据类型，现在老师分配给他一项任务，需要使用 Python 对班级的成绩进行统计管理，成绩如下。

学号	姓名	高等数学	大学英语	程序设计
2020001	张华	85	96	92
2020002	赵云	92	90	96
2020003	李全	86	79	90

要求完成以下操作。

(1) 记录每一门课的平均分(保留两位小数)。

(2) 记录每个人的总分、平均分(保留两位小数)。

(3) 记录格式整齐、美观。

实现代码如下。

```
name1,name2,name3 = '张华','赵云','李全'
math1,math2,math3 = 85,92,86
english1,english2,english3 = 96,90,79
program1,program2,program3 = 92,96,90
avg_math = (math1 + math2 + math3)/3
```

```
avg_eng = (english1 + english2 + english3)/3
avg_pro = (program1 + program2 + program3)/3
total1 = math1 + english1 + program1
total2 = math2 + english2 + program2
total3 = math3 + english3 + program3
total = total1 + total2 + total3
avg1 = total1/3
avg2 = total2/3
avg3 = total3/3
avg = total / 3
print('--'*15+'成绩单管理'+'--'*15)
print('姓名\t高等数学\t大学英语\t程序设计\t总分\t平均分')
print('{:<4}\t{:^8}\t{:^8}\t{:^8}\t{:^4}\t{:^6.2f}'.format(name1,math1,english1,
program1,total1,avg1))
print('{:<4}\t{:^8}\t{:^8}\t{:^8}\t{:^4}\t{:^6.2f}'.format(name2,math2,english2,
program2,total2,avg2))
print('{:<4}\t{:^8}\t{:^8}\t{:^8}\t{:^4}\t{:^6.2f}'.format(name3,math3,english3,
program3,total3,avg3))
print('--'*35)
print('{:<4}\t{:^8.2f}\t{:^8.2f}\t{:^8.2f}\t{:^4.2f}\t{:^6.2f}'.format('平均分',avg_
math,avg_eng,avg_pro,total/3,avg/3))
```

运行结果：

```
------------------------------成绩单管理------------------------------
姓名      高等数学      大学英语      程序设计      总分      平均分
张华      85            96            92            273       91.00
赵云      92            90            96            278       92.67
李全      86            79            90            255       85.00
----------------------------------------------------------------------
平均分    87.67         88.33         92.67         268.67    89.56
```

习题

1. 给定一个以 s 为单位的时间 t，要求用“$<H>:<M>:<S>$”的格式来表示这个时间。$<H>$ 表示时间，$<M>$ 表示分钟， 而 $<S>$ 表示秒，它们都是整数且没有前导的“0”。例如，若 $t=0$，则应输出“0:0:0”；若 $t=3661$，则应输出“1:1:1”。请用常用数值函数，如 mod()、divmod()等编写代码实现。

2. 学校评优的基本条件是语文、数学、英语三门课的成绩均要超过 94 分，或者三门课程的总分超过 285。请编写程序，根据输入的三门课程的成绩，判断该学生是否有资格参与评选。

3. 对天天向上案例进行改写。尽管每天坚持，但人的能力发展不是无限的，而是符合特定模型。假设能力增长符合如下模型：以 7 天为一周期，连续学习 3 天能力值不变，从第 4 天开始到第 7 天每天能力增长为前一天的 1%。如果 7 天中有 1 天间断学习，则周期从头计算。请编写程序求出，如果初始能力为 1，连续学习 365 天后的能力值是多少。

4. 设计程序对凯撒密码进行解密。凯撒密码是古罗马凯撒大帝用来对军事情报进行

加密的算法，它采用替换方法将每一个英文字符循环替换为字母表序中该字符后面第三个字符，对应关系如下。

原文：a b c d e f g h i j k l m n o p q r s t u v w x y z

密文：d e f g h i j k l m n o p q r s t u v w x y z a b c

假设原文字符为 P，对应密文为 S，则两者的关系为

$$S=(P+3)\%\ 26$$

$$P=(S-3)\%\ 26$$

5. 仿照案例 3-10，编写程序模拟超市、酒店等 POS 机打印出来的小票据，完成一次超市购物清单的记录。购物清单如表 3-15 所示。

表 3-15 购物清单

购物日期	名　称	数　量	单　价	总　价
2020 年 6 月 1 日	花生	5	16.8	84
2020 年 6 月 1 日	橙子	7	6.28	43.96
2020 年 6 月 1 日	面包	6	15	90