第3章

基本数据类型

数据类型是每种编程语言必备的属性，只有给数据赋予明确的数据类型，计算机才能对数据进行运算处理。所以，合理运用数据类型就可以大幅度提升代码的执行效率。本章将通过具体案例详细介绍数字类型、数字运算的应用，并介绍字符串类型以及数据类型的判断和操作。本章将通过6个综合课业任务对数据类型的基本操作进行实例演示，以便为后续开发智能语音识别与翻译平台提升执行效率。

【教学目标】

- 了解数字类型特点
- 掌握数字运算符的使用
- 掌握Python中常见的内置数学函数
- 了解字符串的常量和操作符
- 掌握Python中常见的字符串处理函数和处理方法
- 掌握数据类型之间的转换和判断

【课业任务】

王小明想使用Django+百度翻译API开发一个智能语音识别与翻译平台，Django是一个开放源代码的Web应用框架，由Python写成。在学习完Python基本语法后，需要熟悉Python的基本数据类型的使用，现通过6个课业任务来完成。

课业任务3.1 字符串索引的应用

课业任务3.2 整除运算

课业任务3.3 求矩形面积

课业任务3.4 课程等级评定

课业任务3.5 判断识别字符串

课业任务3.6 华氏温度转换为摄氏温度

3.1 数字类型

3.1.1 整数类型

整数用来表示整数数值，即没有小数部分的数值，包括正整数、负整数和0，并且它的位数是任意的。如果要指定一个非常大的整数，只需要写出其所有位数即可（当指定的整数超过计算机自身的计算功能时，会自动转用高精度计算）。整数类型可细分为整型（int）和布尔型（bool）。

整数类型如图 3.1 所示。

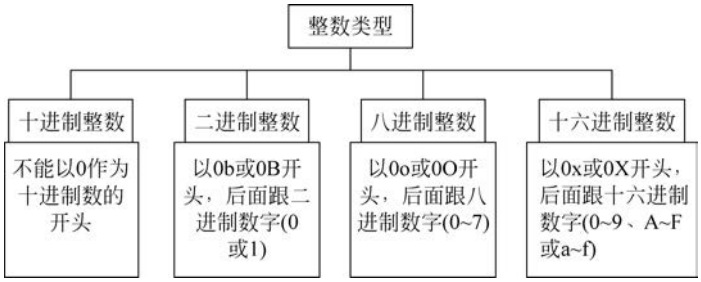

图 3.1　整数类型

Python 3 不再区别整数和长整数，只要计算机内存空间足够，理论上整数可以是无穷大。

案例 3.1　计算 4 的 100 次方和 6 的 100 次方，示例代码如下。

```
x = 4 ** 100                    # 计算 4 的 100 次方
y = 6 ** 100                    # 计算 6 的 100 次方
print(x)                        # 输出 x 的值
print(y)                        # 输出 y 的值
```

执行上述代码，结果如图 3.2 所示。

```
1606938044258990275541962092341162602522202993782792835301376
65331862350007090609669026715805782053714371047295487154307196636949714147737 6

进程已结束,退出代码0
```

图 3.2　案例 3.1 执行结果

如图 3.3 所示，不同进制只是整数的书写形式不同，程序运行时会将整数处理为十进制数。布尔型常量也称为逻辑常量，只有 True 和 False 两个值。

图 3.3　整数和字符串的转换

3.1.2　浮点数类型

浮点数由整数部分和小数部分组成，主要用于处理包括小数的数，浮点数类型的名称为 float。

说明：浮点数取值范围为 -10^{308}～10^{308}；与整数不同的是，如果超过浮点数取值范围，则会产生溢出错误(OverflowError)；14.5、1.2345、−1.32 等都是合法的浮点数常量，可以使用科学记数法表示，如 2.7e1、−3.14e2 等。

因为计算机硬件特点，浮点数不能执行精确运算。

案例 3.2　浮点数不能执行精确运算的情况，示例代码如下。

```
x = 0.1 + 0.3                   # 结果得到想要的 0.4
y = 0.1 + 0.2                   # 结果得到 0.30000000000000004
```

```
print(x)                    # 输出 x 的结果
print(y)                    # 输出 y 的结果
```

执行上述代码,结果如图 3.4 所示。

```
0.4
0.30000000000000004

进程已结束,退出代码0
```

图 3.4　案例 3.2 执行结果

说明:对于这种情况,所有语言都存在这个问题,暂时忽略多余的小数位数即可。

3.1.3　复数类型

Python 中的复数(Complex)与数学中的复数的形式完全一致,由实部和虚部组成,并且使用 j 或 J 表示虚部,如 2+5j、2-5J、5j。

可以用 complex()函数创建复数,基本语法格式如下。

```
complex(实部,虚部)
```

案例 3.3　使用 complex()函数创建复数,示例代码如下。

```
x = complex(3,7)                # 创建复数
print(x)                        # 输出已创建好的复数
```

执行上述代码,运行结果如图 3.5 所示。

```
(3+7j)

进程已结束,退出代码0
```

图 3.5　案例 3.3 执行结果

3.1.4　分数类型

分数是 Python 2.6 和 3.0 版本引入的新类型,分数对象明确地拥有一个分子和分母,分子和分母保持最简,使用分数可以避免浮点数的不精确性。

使用 fractions 模块中的 Fraction()函数创建分数,基本语法格式如下。

```
x = Fraction(a,b)
```

案例 3.4　使用 Fraction()函数创建分数,示例代码如下。

```
from fractions import Fraction              # 从模块导入函数
x = Fraction(1,8)                           # 创建分数
print(x)
Fraction(2,6)                               # 创建新的分数
print(x + 2)                                # 计算 1/3 + 2
y = Fraction.from_float(1.5)                # 将浮点数转换为分数
print(y)
```

执行上述代码，运行结果如图 3.6 所示。

```
1/8
17/8
3/2

进程已结束，退出代码0
```

图 3.6　案例 3.4 执行结果

3.2　数字运算

3.2.1　数字运算操作符

Python 的运算符主要包括算术运算符、赋值运算符、比较(关系)运算符、逻辑运算符和位运算符。

算术运算符的相关解释如表 3.1 所示。

表 3.1　算术运算符

运　算　符	说　　明	举　　例
**	幂运算	1 ** 3
～	按位取反	～5
－	负号	－5
*、%、/、//	乘法、求余数、真除法、Floor 除法	1 * 3、3%2、7/2、7//2
＋、－	加法、减法	1＋2、3－2

赋值运算主要用来为变量等赋值，可以直接把赋值运算符“＝”右边的值赋给左边的变量，也可以进行运算后再赋值给左边的变量。

赋值运算符的相关解释如表 3.2 所示。

表 3.2　赋值运算符

运　算　符	说　　明	举　　例	展 开 形 式
＝	简单的赋值运算	x＝y	x＝y
＋＝、－＝	加法、减法赋值	x＋＝y、x－＝y	x＝x＋y、x＝x－y
*＝、/＝	乘法、除法赋值	x *＝y、x/＝y	x＝x * y、x＝x/y
%＝	取模赋值	x%＝y	x＝x%y
**＝	幂赋值	x **＝y	x＝x ** y
//＝	取整数赋值	x//＝y	x＝x//y

比较运算符也称为关系运算符，用于对变量或表达式的结果进行大小、真假等比较。

比较运算符的相关解释如表 3.3 所示。

表 3.3　比较运算符

运　算　符	说　　明	举　　例	结　　果
＞	大于	'a'＞'b'	False
＜	小于	140 ＜ 460	True
＝＝	等于	'z'＝＝'z'	True
！＝	不等于	'y'！＝'x'	True

续表

运　算　符	说　　明	举　　例	结　　果
>=	大于或等于	456 >= 412	True
<=	小于或等于	40.31 <= 20.35	False

逻辑运算(也称为布尔运算)指对逻辑值(True 或 False)执行 not、and 或 or 操作。

逻辑运算符的相关解释如表 3.4 所示。

表 3.4　逻辑运算符

运算符	说明
not	逻辑非
and	逻辑与
or	逻辑或

在判断 True 或 False 之外的数据是否为逻辑值时,Python 将属于下列情况的值都视为 False,其他则视为 True。

- None;
- 各种数字类型的 0,如 0、0.0、(0+0j)等;
- 空序列,如''、()、[]等;
- 空映射,如{};
- 当类的实例对象的__bool__()方法返回 False 或__len__()方法返回 0 时,实例对象视为 False。

位运算按操作数的二进制位进行操作。

位运算符的相关解释如表 3.5 所示。

表 3.5　位运算符

运　算　符	说　　明	举　　例
<<	向左移位	5 << 3
>>	向右移位	11 >> 4
&	按位与	5 & 3
^	按位异或	5^3
|	按位或	5|3

案例 3.5　位运算符的用法,示例代码如下。

```
# 按位取反
# 5 的 8 位二进制形式为 00000101,按位取反为 11111010
a = ~5
# 按位与
# 4 的二进制形式为 00000100,5 为 00000101,所以结果为 00000100
b = 4 & 5
# 按位异或
# 结果相同为 0,否则为 1
c = 5 ^ 5
# 按位或
# 按位或在相同位上的数有一个为 1 时结果为 1,否则为 0
d = 4 | 5
# 向左移位
# 向左移动 1 位等同于乘以 2
```

```
e = 1 << 2
# 向右移位
# 向右移动 1 位等同于除以 2
f = 10 >> 2
print(a,b,c,d,e,f)          # 输出 a,b,c,d,e,f 的值
```

执行上述代码，结果如图 3.7 所示。

```
-6 4 0 5 4 2

进程已结束，退出代码0
```

图 3.7　案例 3.5 执行结果

Python 运算符的运算规则是优先级高的运算先执行，优先级低的运算后执行，同一优先级的操作按照从左到右的顺序进行，如表 3.6 所示。

表 3.6　运算符优先顺序

类　　型	说　　明	优先级
**	幂	高
～、+、-	取反、正号和负号	↓
～、/、%、//	算术运算符	↓
+、-	算术运算符	↓
<<、>>	左移和右移	↓
&	按位与	↓
^	按位异或	↓
|	按位或	↓
<、<=、>、>=、!=、==	比较运算符	低

3.2.2　数字处理函数

数字处理函数中的常用数学函数如表 3.7 所示。

表 3.7　常用数学函数

类　　型	说　　明	举　　例
abs()	绝对值	abs(-2)
max()、min()	最大值、最小值	max([1,3,5])、min([1,3,5])
len()	序列长度	len('Python')、len([1,3,5])
divmod()	取模	divmod(3,2)
pow()	乘方	pow(2,2)
round()	浮点数	round(2)
sum()	求和	sum(1,2)

案例 3.6　掌握常用的数学函数，示例代码如下。

```
li = [10,2,45,15,6]
print('-10 的绝对值为:', abs(-10))                # 取绝对值
print('取商和余数为:', divmod(3,2))               # 同时取商和余数
print('和为:', sum(li))                           # 求和计算
print('3.1 四舍五入为:', round(3.1))              # 四舍五入
print('3.5 四舍五入为:', round(3.5))              # 四舍五入
print('2 的 5 次方为:', pow(2,5))                 # 计算任意 N 次方值
```

```
print('最小值为:', min(li))          # 取最小值
print('最大值为:', max(li))          # 取最大值
print('长度为:', len(li))            # 返回一个对象或一个项目的长度
```

执行上述代码,结果如图 3.8 所示。

```
-10的绝对值为:  10
取商和余数为:  (1, 1)
和为:  78
3.1四舍五入为:  3
3.5四舍五入为:  4
2的5次方为:  32
最小值为:  2
最大值为:  45
长度为:  5

进程已结束,退出代码0
```

图 3.8　案例 3.6 执行结果

数字处理函数中的常用数学常量如表 3.8 所示。

表 3.8　常用数学常量

常量	说　　明
pi	数学常量 π=3.1415926…,可用精度
e	数学常量 e=2.718281…,可用精度
tau	数学常量 tau=6.283185…,可用精度。tau 是一个等于 2pi 的常量,即圆的周长与其半径之比
inf	浮点正无限(对于负无穷大,请使用 math.inf),等同于 float('inf')的输出
nan	浮点"不是数字"(NaN)值,等同于 float('nan')的输出

案例 3.7　认识 math 模块中的几个数学常量,示例代码如下。

```
import math                          # 导入 math 模块
e_value = math.e                     # 欧拉数(自然数 e = 2.718281828459045)
inf_value = math.inf                 # 正无穷大浮点数
nan_value = math.nan                 # 浮点值
pi_value = math.pi                   # 圆周率
tau_value = math.tau                 # tau 是一个圆周长数,等于 2pi

print('e 为:', e_value)              # 输出 e 的值
print('正无穷大浮点数为:', inf_value)  # 输出正无穷大浮点数的值
print('浮点值为:', nan_value)        # 输出浮点数的值
print('圆周率为:', pi_value)         # 输出圆周率的值
print('圆周长为:', tau_value)        # 输出圆周长的值
```

执行上述代码,结果如图 3.9 所示。

```
e为:  2.718281828459045
正无穷大浮点数为:  inf
浮点值为:  nan
圆周率为:  3.141592653589793
圆周长为:  6.283185307179586

进程已结束,退出代码0
```

图 3.9　案例 3.7 执行结果

3.3　字符串类型

3.3.1　字符串常量

Python 字符串常量可用单引号、双引号、3 个单引号或双引号、带 r 或 R 前缀的 raw 字符串、带 u 或 U 前缀的 Unicode 字符串等多种方法表示。

字符串常量的表示方法如下。

```
# 单引号
'a'
'123'
'abc'
# 双引号
"a"
"123"
"abc"
# 3 个单引号或双引号(可以包含多行字符)
'''Python code'''
"""Python string"""
# raw 字符串
r'abc\n123'
R'abc\n123'
# Unicode 字符串(默认字符串,可以省略)
u'asdf'
U'asdf'
```

字符串都是 str 类型的对象,可用内置的 str()函数创建字符串对象。

案例 3.8　使用 str()函数创建字符串对象,示例代码如下。

```
x = str(135)                # 用数字创建字符串对象
y = str('abc135')           # 用字符串常量创建字符串对象
print(x)                    # 输出字符串:135
print(type(x))              # 判断字符串对象类型
print(y)                    # 输出字符串:abc135
```

执行上述代码,结果如图 3.10 所示。

```
135
<class 'str'>
abc135

进程已结束,退出代码0
```

图 3.10　案例 3.8 执行结果

转义字符用于表示不能直接表示的特殊字符,Python 常用转义字符如表 3.9 所示。

表 3.9　转义字符

转义字符	说　　明
\	反斜杠
\'	单引号

续表

转义字符	说　　明
\"	双引号
\a	响铃符
\b	退格符
\f	换页符
\n	换行符
\r	回车符
\t	水平制表符
\v	垂直制表符
\o	NULL
\ooo	3 位八进制表示的 Unicode 码对应字符
\xhh	两位十进制表示的 Unicode 码对应字符

3.3.2　字符串操作符

字符串是字符的有序集合,下面简单介绍 in 操作符、空格、加号(+)、星号(*)、逗号分隔符的使用。

案例 3.9　用 in 操作符判断字符串包含关系,示例代码如下。

```
x = 'Python'          # 创建字符串
print('a' in x)       # 判断字符串中是否包含'a',若包含,则返回 True,否则返回 False
print('thon' in x)    # 判断字符串中是否包含'thon',若包含,则返回 True,否则返回 False
print('1' in x)       # 判断字符串中是否包含'1',若包含,则返回 True,否则返回 False
```

执行上述代码,结果如图 3.11 所示。

```
False
True
False

进程已结束,退出代码0
```

图 3.11　案例 3.9 执行结果

案例 3.10　以空格分隔(或者没有分隔符号)的多个字符串可自动合并,示例代码如下。

```
x = '01''10''11'                # 创建字符串
print(x)                        # 字符串自动合并,输出 011011
```

执行上述代码,结果如图 3.12 所示。

```
011011

进程已结束,退出代码0
```

图 3.12　案例 3.10 执行结果

案例 3.11　用加号操作符将多个字符串合并,示例代码如下。

```
x = '11' + '10' + '00'                    # 创建字符串
print(x)                                  # 多个字符串合并,输出 111000
```

执行上述代码,结果如图 3.13 所示。

```
111000

进程已结束,退出代码0
```

图 3.13　案例 3.11 执行结果

案例 3.12　用星号操作符将字符串复制多次并构成新的字符串,示例代码如下。

```
print('10' * 4)                   # 输出构成新的字符串 10101010
```

执行上述代码,结果如图 3.14 所示。

```
10101010

进程已结束,退出代码0
```

图 3.14　案例 3.12 执行结果

案例 3.13　使用逗号分隔字符时,会创建用字符串组成的元组,示例代码如下。

```
x = 'Hello', 'Python'                     # 创建字符串
print(x)                                  # 输出字符串
print(type(x))                            # 判断字符串类型
```

执行上述代码,结果如图 3.15 所示。

```
('Hello', 'Python')
<class 'tuple'>

进程已结束,退出代码0
```

图 3.15　案例 3.13 执行结果

3.3.3　字符串的索引

字符串是一个有序的集合,其中的每个字符可通过偏移量进行索引或分片。

字符串中的字符按从左到右的顺序,其偏移量依次为 0,1,2,…,len－1(最后一个字符偏移量为字符串长度减 1)。按从右到左的顺序,偏移量取负值,依次为－len,－len＋1,…,－2,－1。

索引指通过偏移量定位字符串中的单个字符。

案例 3.14　使用字符串的索引,示例代码如下。

```
x = 'Python'
print('第 1 个字符是:', x[0])                  # 索引第 1 个字符
print('最后一个字符是:', x[-1])                # 索引最后一个字符
print('第 4 个字符是:', x[3])                  # 索引第 4 个字符
```

执行上述代码,结果如图 3.16 所示。

```
第1个字符是:  P
最后一个字符是:  n
第4个字符是:  h

进程已结束,退出代码0
```

图 3.16　案例 3.14 执行结果

索引可获得指定位置的单个字符,但不能通过索引修改字符串。

案例 3.15　字符串对象不允许被修改,示例代码如下。

```
y = 'Python'
y[0] = 'c'              # 修改字符串中的指定字符,会报错
print(y)                # 输出修改后的字符串
```

执行上述代码,结果如图 3.17 所示。

```
    y[0] = 'c'              # 修改字符串中的指定字符, 会报错
TypeError: 'str' object does not support item assignment

进程已结束,退出代码1
```

图 3.17　案例 3.15 执行结果

3.3.4　字符串的切片

字符串的切片也称为分片,它使用索引范围从字符串中获得连续的多个字符(即子字符串),字符串切片的基本语法格式如下。

```
x[ start : end ]
```

上述代码返回字符串 x 中从偏移量 start 开始到偏移量 end 前的子字符串;start 和 end 参数均可省略,start 默认为 0,end 默认为字符串长度。

案例 3.16　字符串的切片,示例代码如下。

```
x = 'Python'
print(x[1:4])           # 返回偏移量为 1~3 的字符
print(x[1:])            # 返回偏移量为 1 到末尾的字符
print(x[:4])            # 返回从字符串开头到偏移量为 3 的字符
print(x[:-1])           # 除最后一个字符,其他字符全部返回
print(x[:])             # 返回全部字符
```

执行上述代码,结果如图 3.18 所示。

3.3.5　迭代字符串

字符串是有序的字符集合,可用 for 循环迭代处理字符串。

案例 3.17　迭代字符串,示例代码如下。

```
for a in 'Python':              # 变量 a 依次表示字符串中的每个字符
    print(a)                    # 输出迭代处理后的字符串
```

```
yth
ython
Pyth
Pytho
Python

进程已结束,退出代码0
```

图 3.18　案例 3.16 执行结果

执行上述代码,结果如图 3.19 所示。

```
P
y
t
h
o
n

进程已结束,退出代码0
```

图 3.19　案例 3.17 执行结果

3.3.6　字符串处理函数

字符串长度指字符串中包含的字符个数,可用 len()函数获得字符串长度。

案例 3.18　求字符串长度,示例代码如下。

```
x = 'Python'              # 创建字符串'Python'
print(len(x))             # 输出字符串'Python'的长度
```

执行上述代码,结果如图 3.20 所示。

```
6

进程已结束,退出代码0
```

图 3.20　案例 3.18 执行结果

可用 str()函数将非字符串数据转换为字符串,示例代码如下。

案例 3.19　将非字符串数据转换为字符串,示例代码如下。

```
print(str(10))                # 将整数转换为字符串
print(str(10.2))              # 将浮点数转换为字符串
print(str(2+3j))              # 将复数转换为字符串
print(str([10,11,12]))        # 将列表转换为字符串
print(str(False))             # 将布尔常量转换为字符串
```

执行上述代码,结果如图 3.21 所示。

在转换数字时,repr()和 str()函数的效果相同。在处理字符串时,repr()函数会将一对表示字符串常量的单引号添加到转换后的字符串中。

```
10
10.2
(2+3j)
[10, 11, 12]
False

进程已结束,退出代码0
```

图 3.21　案例 3.19 执行结果

案例 3.20　repr()函数的用法,示例代码如下。

```
print(str(123456), repr(123456))          # repr()函数会直接添加到转换之后的字符串
print(str('123456'), repr('123456'))      # repr()函数会将字符串的引号显示出来
print(str("123456"), repr("123456"))      # 若是双引号,也会用单引号显示出来
```

执行上述代码,结果如图 3.22 所示。

```
123456 123456
123456 '123456'
123456 '123456'

进程已结束,退出代码0
```

图 3.22　案例 3.20 执行结果

案例 3.21　ord()函数返回字符的 Unicode 码；chr()函数返回 Unicode 码对应的字符,示例代码如下。

```
print('返回字符 Unicode 码')
print('"A"字符的 Unicode 码为:', ord('A'))
print('"字"字符 Unicode 码为:', ord('字'))
print('将 Unicode 码转换为字符')
print('将 65 转换为字符为:', chr(65))
print('将 23383 转换为字符为:', chr(23383))
```

执行上述代码,结果如图 3.23 所示。

```
返回字符Unicode码
"A"字符的Unicode码为:  65
"字"字符Unicode码为:  23383
将Unicode码转换为字符
将65转换为字符为:  A
将23383转换为字符为:  字
```

图 3.23　案例 3.21 执行结果

3.3.7　字符串处理方法

字符串是 str 类型的对象,字符串处理方法调用格式：字符串.方法()。表 3.10～表 3.19 分别列出了字符串的各种常用方法及说明。

表 3.10　字符串查询

常用方法	说　　明
str. find(s)	返回字符串 s 在字符串 str 中的位置索引，没有则返回－1
str. rfind()	类似于 find()函数，不过是从右边开始查找
str. index(s)	和 find()方法一样，但是如果 s 不存在于 str 中，则抛出异常
str. rindex()	类似于 index()函数，不过是从右边开始

表 3.11　字符串大小转换操作

常用方法	说　　明
str. upper()	将字符串中所有元素都转换为大写
str. lower()	将字符串中所有元素都转换为小写
str. swapcase()	交换大小写，大写转换为小写，小写转换为大写
str. capitalize()	把字符串的首字母大写
str. title()	将字符串中所有单词的首字母转换为大写，其余均为小写

表 3.12　合并与替换

常用方法	说　　明
str. join(sep)	以指定字符串作为分隔符，将 sep 中所有元素合并为一个新的字符串
str. replace(old，new[，max])	把字符串中的 old 替换成 new，如果指定 max，则替换不超过 max 次

表 3.13　字符串对齐

常用方法	说　　明
str. center(width[，fillchar])	返回一个指定的宽度 width 居中的字符串，fillchar 为填充的字符，默认为空格
str. ljust(width[，fillchar])	返回一个原字符串左对齐，并使用 fillchar 填充至长度 width 的新字符串，fillchar 默认为空格
str. rjust(width[，fillchar])	返回一个原字符串右对齐，并使用 fillchar(默认空格)填充至长度 width 的新字符串
str. zfill(width)	返回长度为 width 的字符串，原字符串右对齐，前面填充 0

表 3.14　分割字符串

常用方法	说　　明
str. split(seq＝""，num＝string. count())	以 seq (默认空格)为分隔符截取字符串，如果 num 有指定值，则仅截取 num＋1 个子字符串(只需 num 个 seq 分隔符)
str. rsplit()	与 split()方法类似，不过是从右边开始分割
str. splitlines()	按照行进行分割，得到新的列表
str. partition(str)	找到字符串中第 1 个 str，并以 str 为界，将字符串分割为 3 部分，返回一个新的元组
str. rpartition()	与 partition()方法类似，只不过是反向找

表 3.15　判断字符串

常用方法	说　　明
str. isidentifier()	判断字符串是否为合法标识符(字符、数字、下画线)
str. isspace()	如果 str 中只包含空格，则返回 True，否则返回 False
str. isalpha()	如果 str 至少有一个字符并且都是字母则返回 True，否则返回 False
str. isdecimal()	判断字符是否全部由十进制数字组成，不包括中文、罗马字符

续表

常用方法	说　　明
str.isdigit()	判断字符串是否只包含数字,不包括中文数字
str.isnumeric()	判断字符串是否全部由数字组成,包括中文数字
str.isalnum()	判断字符串是否由字母和数字组成
str.islower()	判断字符串中的字符是否全部为小写,字符串至少有一个字符
str.supper()	判断字符串中的字符是否全部为大写,字符串至少有一个字符
str.istitle()	判断字符串是否标题化
str.isascii()	如果字符串为空或字符串中的所有字符都是 ASCII 编码,则返回 True,否则返回 False
str.isprintable()	如果所有字符都是可打印的,则返回 True,否则返回 False

表 3.16　去除两端多余字符操作

常用方法	说　　明
str.lstrip([Chars])	去掉左边的指定字符(不是字符串),默认为空白字符
str.rstrip([Chars])	去掉右边的指定字符
str.strip([Chars])	去掉左右两边的指定字符

表 3.17　判断开头结尾字符串

常用方法	说　　明
str.startswith(strz)	检查字符串是否以 strz 开头,若是,则返回 True
str.endswith(strz)	检查字符串是否以 strz 结尾,若是,则返回 True

表 3.18　字符串计数

常用方法	说　　明
str.count(sub[,start,end])	在字符串[start,end)范围内,计算 sub 字符串的个数
len()	len()是内置函数,计算字符串中的字符个数

表 3.19　判断开头结尾字符串

常用方法	说　　明
str.encode(encoding='UTF-8',errors='strict')	以 encoding 指定的编码格式编码字符串,如果出错,默认抛出 ValueError 异常,除非 errors 指定的是'ignore'或'replace'
bytes.decode()	Python 3 中没有 decode()方法,但可以使用 bytes 对象的 decode()方法解码给定的 bytes 对象,这个 bytes 对象可以由 str.encode()方法编码返回

3.3.8　字符串的格式化

字符串格式化表达式用“%”表示,“%”之前为格式字符串,“%”之后为需要填入格式字符串中的参数,基本语法格式如下。

```
格式字符串 % (参数 1,参数 2,…)
```

Python 中常用的字符串格式化表示符如表 3.20 所示。

表 3.20　常用的字符串格式化表示符

表示符	说　　明
s	将非 str 类型的对象用 str()函数转换为字符串
r	将非 str 类型的对象用 repr()函数转换为字符串

续表

表示符	说　　明
c	参数为单个字符(包括各国文字)或字符的 Unicode 码,将 Unicode 码转换为对应的字符
d、i	参数为数字,转换为带符号的十进制整数
o	参数为数字,转换为带符号的八进制整数
x	参数为数字,转换为带符号的十六进制整数,字母小写
X	参数为数字,转换为带符号的十六进制整数,字母大写
e	将数字转换为科学记数法格式(小写)
E	将数字转换为科学记数法格式(大写)
f、F	将数字转换为十进制浮点数
g	浮点格式,如果指数小于－4 或不小于精度(默认为 6),则使用小写指数格式,否则使用十进制格式
G	浮点格式,如果指数小于－4 或不小于精度(默认为 6),则使用大写指数格式,否则使用十进制格式

3.3.9　bytes 字符串

bytes 对象是一个不可变的字节对象序列,是一种特殊的字符串,也称为 bytes 字符串,用前缀 b 表示,示例如下。

```
# 单引号
b'a'
b'111'
b'abc'
# 双引号
b"a"
b"111"
b"abc"
# 3 个单引号或双引号
b'''Python code'''
b"""Python string"""
```

在 bytes 字符串中只能包含 ASCII 字符,使用非 ASCII 字符会报错。

案例 3.22　bytes 字符串,示例代码如下。

```
x = b'字母 anc'                    # 使用非 ASCII 字符
print(x)                           # 输出非 ASCII 字符会报错
```

执行上述代码,结果如图 3.24 所示。

```
    x = b'字母anc'    #使用非ASCII字符
        ^
SyntaxError: bytes can only contain ASCII literal characters.

进程已结束,退出代码1
```

图 3.24　案例 3.22 执行结果

3.4 数据类型操作

3.4.1 类型判断

数据类型的判断,可以用 type()函数查看数据类型。

案例 3.23 type()函数的使用,示例代码如下。

```
x = 111                 # 创建整数类型字符串
y = (111.2)             # 创建浮点数类型字符串
print(type(x))          # 判断 x 的字符串类型
print(type(y))          # 判断 y 的字符串类型
```

执行上述代码,结果如图 3.25 所示。

```
<class 'int'>
<class 'float'>

进程已结束,退出代码0
```

图 3.25 案例 3.23 执行结果

3.4.2 类型转换

可以使用 int()函数将一个字符串按指定进制转换为整数。int()函数基本语法格式如下。

```
int('整数字符串', n)              # n 表示转换进制数
```

案例 3.24 int()函数的使用,示例代码如下。

```
print(int('11001'))             # 默认按十进制转换
print(int('11001',2))           # 按二进制转换
print(int('11001',8))           # 按八进制转换
print(int('1101',10))           # 按十进制转换
```

执行上述代码,结果如图 3.26 所示。

```
11001
25
4609
1101

进程已结束,退出代码0
```

图 3.26 案例 3.24 执行结果

int()函数的第 1 个参数只能是整数字符串,第 1 个字符可以是正/负号,其他字符必须是数字,不能包含小数点或其他符号,否则会报错。

float()函数可将整数和字符串转换为浮点数。

案例 3.25 float()函数的使用,示例代码如下。

```
print(float(15))                # 将整数 15 转换为浮点数
print(float('15'))              # 将整数'15'转换为浮点数
```

```
print(float('+15'))                # 将整数'+15'转换为浮点数
print(float('-15'))                # 将整数'-15'转换为浮点数
```

执行上述代码，结果如图 3.27 所示。

```
15.0
15.0
15.0
-15.0

进程已结束,退出代码0
```

图 3.27　案例 3.25 执行结果

内置函数 bin()、oct()和 hex()用于将整数转换为对应进制的字符串。

案例 3.26　内置函数 bin()、oct()和 hex()转换进制，示例代码如下。

```
print(bin(122))                    # 转换为二进制字符串
print(oct(122))                    # 转换为八进制字符串
print(hex(122))                    # 转换为十六进制字符串
```

执行上述代码，结果如图 3.28 所示。

```
0b1111010
0o172
0x7a

进程已结束,退出代码0
```

图 3.28　案例 3.26 执行结果

3.5　课业任务

扫一扫

视频讲解

课业任务 3.1　字符串索引的应用

【能力测试点】

字符串类型中索引的应用。

【任务实现步骤】

(1) 打开 PyCharm，新建一个 Python 文件，本课业任务以“任务 3.1”命名。

(2) 任务需求：对字符串'Python'进行索引。首先创建对象 x，接着按要求进行程序编写，代码如下。

```
x = 'Python'                       # 创建字符串'Python'
for i in range(len(x)):            # 根据索引遍历字符串的内容
    print(x[i],end = "")           # i 从 0 开始索引
print()                            # 默认换行
for i in range(len(x)):            # 根据索引遍历字符串的内容
    print(x[-(i+1)],end = "")      # -(i+1)是反向索引
```

运行结果如图 3.29 所示。

```
Python
nohtyP
进程已结束,退出代码0
```

图 3.29　课业任务 3.1 运行结果

扫一扫

视频讲解

课业任务 3.2　整除运算

【能力测试点】

整除运算 int()函数的应用；使用 if 语句进行判断。

【任务实现步骤】

(1) 打开 PyCharm，新建一个 Python 文件，本课业任务以“任务 3.2”命名。

(2) 任务需求：输入一个 4 位以上的整数，否则提示输入错误，输出其千位以上的数字。例如，用户输入 1234，则程序输出 1(使用 if 语句进行程序编写)。首先创建对象 x 和 y，接着按要求进行程序编写，代码如下。

```
x = eval(input('请输入一个 4 位以上的整数: '))
y = int(x / 1000)                    # int()函数进行整除运算
if(x < 999):                         # 如果 x < 999,则输出'输出错误'
    print('输入错误 ')
else:                                # 否则,输出整除结果
    print(y)
```

(3) 运行程序，输入整数 25，输出如图 3.30 所示；输入整数 1212，输出如图 3.31 所示。

```
请输入一个 4 位以上的整数: 25
输入错误

进程已结束,退出代码0
```

图 3.30　课业任务 3.2 运行结果(1)

```
请输入一个 4 位以上的整数: 1212
1

进程已结束,退出代码0
```

图 3.31　课业任务 3.2 运行结果(2)

扫一扫

视频讲解

课业任务 3.3　求矩形面积

【能力测试点】

input()函数、内置函数 map()和运算符的应用。

【任务实现步骤】

(1) 打开 PyCharm，新建一个 Python 文件，本课业任务以“任务 3.3”命名。

(2) 任务需求：输入矩形的长和宽，计算矩形的面积并输出(使用 input()函数实现同步赋值)。首先创建对象 L_W，接着按要求进行程序编写，代码如下。

```
L_W = input('请输入矩形的长和宽,以空格分开:')
length, width = map(float,L_W.split())          # 使用 map()函数映射出指定的序列
S = length * width                              # 面积 = 长 * 宽
print(" 矩形面积为:",S)                          # 输出计算后的面积
```

(3) 运行程序，输入长和宽分别为 4 和 5，结果如图 3.32 所示。

```
请输入矩形的长和宽，以空格分开：4 5
矩形面积为：  20.0

进程已结束,退出代码0
```

图 3.32　课业任务 3.3 运行结果

扫一扫

视频讲解

课业任务 3.4　课程等级评定

【能力测试点】

数字运算操作符和 if 语句的应用。

【任务实现步骤】

(1) 打开 PyCharm，新建一个 Python 文件，本课业任务以“任务 3.4”命名。

(2) 任务需求：将某课程的百分制分数 m 转换为 5 分制(优、良、中、及格、不及格)的评定等级 grade。评定条件：$m \geqslant 90$ 为优，$80 \leqslant m < 90$ 为良，$70 \leqslant m < 80$ 为中，$60 \leqslant m < 70$ 为及格，$m < 60$ 为不及格。首先创建对象 m，接着按要求进行程序编写，代码如下。

```
m = eval(input('请输入你的分数:'))
if m >= 90:                    # 如果 m >= 90,则 grade 为优
    print('grade 为:', '优')
elif 80 <= m < 90:             # 否则如果 80 <= m < 90,则 grade 为良
    print('grade 为:', '良')
elif 70 <= m < 80:             # 否则如果 70 <= m < 80,则 grade 为中
    print('grade 为:', '中')
elif 60 <= m < 70:             # 否则如果 60 <= m < 70,则 grade 为及格
    print('grade 为:', '及格')
elif m < 60:                   # 否则如果 m < 60,则 grade 为不及格
    print('grade 为:', '不及格')
```

(3) 运行程序，输入分数 93，输出如图 3.33 所示；输入分数 59，输出如图 3.34 所示。

```
请输入你的分数: 93
grade为:    优

进程已结束,退出代码0
```

图 3.33　课业任务 3.4 运行结果(1)

```
请输入你的分数: 59
grade为:    不及格

进程已结束,退出代码0
```

图 3.34　课业任务 3.4 运行结果(2)

扫一扫

视频讲解

课业任务 3.5　判断识别字符串

【能力测试点】

字符串处理函数的应用。

【任务实现步骤】

(1) 打开 PyCharm，新建一个 Python 文件，本课业任务以“任务 3.5”命名。

(2) 任务需求：变量 x = "Learning Python makes me happy"。判断"e"出现的次数；

判断字符串是否为空；判断字符串是否为大写；判断字符串是否全为数字；将字符串填充到长度为45(使用“+”进行填充)。首先创建对象x,且赋值为"Learning Python makes me happy",接着按要求进行程序编写,代码如下。

```
x = "Learning Python makes me happy"
print("某个字符串出现次数: x.count('e') = ", x.count('e'))
print("字符串是否都是空格: x.isspace() = ", x.isspace())
print("字符串是否大写: x.isupper() = ", x.isupper())
print("字符串是否全为数字: x.isalnum() = ", x.isalnum())
print("填充+值: x.rjust(45,"+") = ", x.rjust(45,"+"))
```

运行结果如图3.35所示。

```
某个字符串出现次数:  x.count('e') =  3
字符串是否都是空格:  x.isspace() =  False
字符串是否大写:  x.isupper() =  False
字符串是否全为数字:  x.isalnum() =  False
填充+值: x.rjust(45,)=  +++++++++++++++Learning Python makes me happy

进程已结束,退出代码0
```

图3.35　课业任务3.5运行结果

扫一扫

视频讲解

课业任务3.6　华氏温度转换为摄氏温度

【能力测试点】

数据类型转换的应用。

【任务实现步骤】

(1) 打开PyCharm,新建一个Python文件,本课业任务以“任务3.6”命名。

(2) 任务需求：将华氏温度F转换为摄氏温度C,转换公式为$C=(F-32)\times 5/9$。首先创建对象F和C,接着按要求进行程序编写,代码如下。

```
# 将华氏温度转换为摄氏温度
F = int(input("请输入华氏摄氏度:"))              # 输入华氏温度
C = int((F-32)*5/9)                              # 转换为摄氏温度
print("转换后的摄氏温度为:", C)                   # 输出转换后的摄氏温度
```

(3) 运行程序,输入华氏温度为65,结果如图3.36所示。

```
请输入华氏摄氏度: 65
转换后的摄氏度为:  18

进程已结束,退出代码0
```

图3.36　课业任务3.6运行结果

习题3

1. 选择题

(1) 下列关于表达式中整型、浮点型、复数型的运算结果类型的说法中正确的是(　　)。

A. 表达式中整型、浮点型、复数型的运算结果是整型

B. 表达式中浮点型、复数型的运算结果是复数型

C. 表达式中整型、浮点型、复数型的运算结果是字符型

D. 表达式中整型、浮点型、复数型的运算结果是浮点型

(2) 遍历循环语句 for i in range (n),n 的数据类型是(　　)。

A. 字符串类型　　B. 浮点型　　C. 整数类型　　D. 复数类型

(3) Python 内置函数(　　)可以返回列表、元组、字典、集合、字符串以及 range 对象中的元素个数。

A. max()　　B. len()　　C. globals()　　D. type()

(4) 下列关于 Python 复数的说法中错误的是(　　)。

A. 表示复数的语法是 real+image j

B. 实部和虚部都是浮点数

C. 虚部必须后缀 j,且必须是小写

D. conjugate()方法返回复数的共轭复数

(5) 计算机中信息处理和信息存储用(　　)。

A. 二进制代码　　B. 十进制代码　　C. 十六进制代码　　D. ASCII 码

(6) Python 中不支持的数据类型有(　　)。

A. int　　B. char　　C. list　　D. float

(7) 下列关于字符串的说法中错误的是(　　)。

A. 字符应该视为长度为 1 的字符串

B. 既可以用单引号,也可以用双引号创建字符串

C. 以\0 标志字符串的结束

D. 在三引号字符串中可以包含换行、回车等特殊字符

2. 填空题

(1) 表达式 abs(−23)的值为_____。

(2) 表达式 3 | 5 的值为_____。

(3) 表达式 [1, 2, 3] == [1, 3, 2] 的值为_____。

(4) Python 内置函数_____用来返回序列中的最小元素。

(5) 表达式 'abc1212'.isalnum() 的值为_____。

3. 判断题

(1) 在 Python 中 0xa1 是合法的八进制数字表示形式。(　　)

(2) 可以使字符串开头首字母大写的字符串函数有 capitalize()和 title()。(　　)

(3) decode()方法是编码,encode()方法是解码,如果字符串是 Unicode 编码,就可以直接编码而不需要解码。(　　)

(4) 在 Python 中,字符串没有长度限制。(　　)

(5) Python 中运算符的运算规则是优先级高的运算先执行,优先级低的运算后执行,同级则按从左到右的顺序执行。(　　)

4. 编程题

(1) 输入一个 4 位的整数,分别输出其个位、十位、百位、千位。

(2) 输入三角形 3 条边的边长,求三角形的面积。