# 第 2 章

# 数据类型与运算符

本章主要介绍 Python 的字符串、数值、布尔值、空值等数据类型，以及变量、常量和基本运算符的使用方式等。

本章要求掌握字符串类型的变量的定义和常用操作，掌握数值类型的定义和类型转换，掌握运算符的使用方式和优先级，理解变量和常量。

学习目标：

- 字符串类型及其操作
- 数字类型及其操作
- 计算机中数的表示及进制转换
- 布尔类型及其操作
- 空值类型
- 基本数据类型之间的转换
- 变量和常量
- 基本运算符

## 2.1 字符串类型

Python 的基本数据类型包括字符串、数字、布尔值和空值。其中，字符串是最常用的基本数据类型之一。

字符串是一个有序的字符集合，即字符序列。在 Python 中，字符串属于不可变序列类型，使用单引号、双引号、三单引号或三双引号作为界定符，不同界定符之间可以互相嵌套。

说明：

☑ 单引号界定的字符串，可以嵌套双引号字符串，如'Python programming "games" '。

☑ 双引号界定的字符串，可以嵌套单引号字符串，如"Python programming 'games' "。

☑ 三单引号或三双引号界定的字符串，如'''Python'''，可以跨行用于表示长字符串。

☑ Python 中的各种界定符都应该在英文输入状态下输入。

### 2.1.1 字符串的输入

Python3.x 中输入函数 input( )接收用户输入的数据，input( )返回值即为字符串类型。例如下面代码中，input("Please input name:")的返回值为用户输入的字符串，如“Guido van Rossum”，表达式中变量 name 的类型为字符串，值为“Guido Van Rossum”。

```
name=input("Please input name:")
```

**说明：**

☑ 吉多·范罗苏姆（Guido van Rossum），荷兰计算机程序员，他作为 Python 程序设计语言的作者而为人们熟知，被誉为“Python 之父”。

### 2.1.2 转义字符串

在 Python 中如果要表示控制字符或特殊意义的符号，主要使用转义字符串。转义字符串以反斜杠“\”开始，紧跟一个转义字母，如“\n”表示换行符，“\t”表示制表符，“\\”表示这里是一个反斜杠。

Python 中主要的转义字符串如表 2.1 所示。

**表 2.1 转义字符串**

| 转义字符串 | 代表的特殊符号 | 转义字符串 | 代表的特殊符号 |
|---|---|---|---|
| \′ | 单引号 | \f | 换页 |
| \″ | 双引号 | \n | 换行 |
| \\ | 反斜杠 | \r | 回车 |
| \a | 响铃 | \t | 水平制表符 |
| \b | 退格 | \v | 垂直制表符 |

### 2.1.3 字符串的格式化

字符串格式化方法可以将其他类型数据或者表达式转换为字符串或另一种数据格式，嵌入到字符串或模板中输出。

1. 字符串的 f-string 格式化

在 Python 3.6 之前，将 Python 表达式嵌入到字符串中进行格式化的主要方法是：%格式化方法和 str.format( )函数。

从 Python 3.6 开始，引入了更易读、更简洁、不易出错的 f-string 格式化方法。f-string 格式化字符串以字母“f”开头，后面紧跟着字符串，字符串中的表达式用大括号{}括起来，它会将变量或表达式计算后的值替换进去。

本节主要介绍字符串的 f-string 格式化和%格式化。

**【实例 2.1】字符串的 f-string 格式化。**

```
#Example2.1.py

name1="Guido Van Rossum"
year1= 1956
program_lang1="Python"
name2="Dennis MacAlistair Ritchie"
year2= 1941
program_lang2="C"
print(f"\t {name1} is the author of the { program_lang1} language . He
  was born in {year1}.")
print(f"\t {name2} is the author of the { program_lang2} language . He
  was born in {year2}. ")
```

**【运行结果】**

```
Guido Van Rossum is the author of the Python language . He was born in
  1956.
Dennis MacAlistair Ritchie is the author of the C language . He was born
  in 1941.
```

**说明：**

☑ f-string 格式化字符串以字母"f"开头，后面紧跟着字符串。

☑ 程序运行时，f-string 中的{ }被相应的变量值替代。

☑ 本例中使用了转义字符"\t"。

2. 字符串的%格式化

Python 字符串%格式化的一般格式如图 2.1 所示。

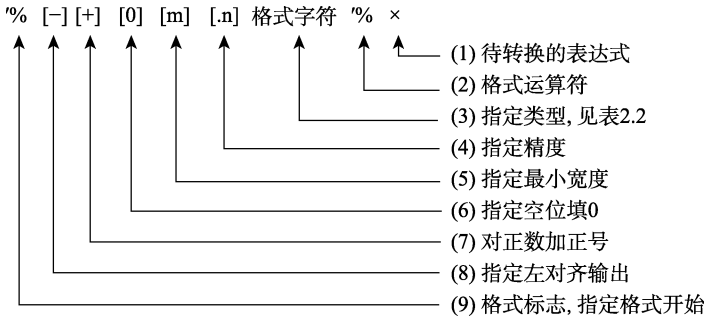



图 2.1　字符串格式化

**说明：**

☑ %格式化的一般格式有可选项，但各项顺序不能改变。

☑ 第 1 个%符号是格式化字符串的开始符号。

☑ 转换标志–、+、0 是可选项，–表示左对齐；+表示在转换值之前要加上正负号；0 表示如果转换值位数不够就用 0 填充。

☑ 最小宽度 m 是可选项，转换后的字符串如果小于这个宽度补 0 或空格，如果大于等于这个宽度则按照实际宽度输出。

☑ 精度 n 是可选项，表示出现在小数点后的位数。

☑ 格式字符规定以什么形式显示表达式，常见的格式字符如表 2.2 所示。

☑ 第 2 个%符号是待转换表达式的开始符号。

☑ 待转换的表达式 x 可以是常量、变量或者表达式。

**表 2.2　格式字符**

| 格式字符 | 说　明 | 格式字符 | 说　明 |
|---|---|---|---|
| % s | 字符串（采用 str( )的显示） | % X | 十六进制整数（大写） |
| % r | 字符串（采用 repr( )的显示） | % e | 指数（基底写为 e） |
| % c | 单个字符 | % E | 指数（基底写为 E） |
| % d | 十进制整数 | % f、% F | 浮点数 |
| % o | 八进制整数 | % g | 指数（e）或浮点数 |
| % u | 无符号整数 | % G | 指数（E）或浮点数 |
| % x | 十六进制整数 | % % | 字符% |

**【实例 2.2】%格式化输出。**

```
#Example2.2.py

print('''根据调查，84%的 Python 用户已经在用 Python 3。''')
print('''根据调查，%f 的 Python 用户已经在用%s。'''%(0.84,'Python 3'))
print('''根据调查， %.2f%%的 Python 用户已经在用%s。'''%(0.84*100,'Python 3'))
print('''根据调查， %8.2f%%的 Python 用户已经在用%.6s。'''%(0.84*100,'Python
  3'))
print('''根据调查，%-8.2f%%的 Python 用户已经在用%-.6s。'''%(0.84*100,'
  Python 3'))
```

**【运行结果】**

```
根据调查，84%的 Python 用户已经在用 Python 3。
根据调查，0.840000 的 Python 用户已经在用 Python 3。
根据调查，84.00%的 Python 用户已经在用 Python 3。
根据调查，    84.00%的 Python 用户已经在用 Python。
根据调查，84.00   %的 Python 用户已经在用 Python。
```

**说明：**

☑ %格式化后面如果多个表达式，格式用%(表达式 1，表达式 2，……，表达式 n)。

☑ %f 格式字符默认保留小数点之后 6 位。

☑ 转换标志-符号表示左对齐。

☑ %%格式字符代表%。

☑ .n 格式字符对于数值来说，代表小数点后的位数。

☑ .n 格式字符对于字符串来说，代表能显示的最大字符个数。

### 2.1.4 字符串运算符

1. 连接运算符“+”

连接运算符形式与加法符号相同，用于将两个字符串连接成一个新的字符串。

2. 重复运算符“*”

重复运算符用于字符串的重复输出，重复的次数由与“*”运算符搭配的操作数指定。

**【实例 2.3】字符串连接运算符和重复运算符。**

```
#Example2.3.py

program_lang='Python'
print('Hello'+' '+program_lang+'\n')
print('Hello'+' '+program_lang*3+'!')
```

**【运行结果】**

```
Hello Python

Hello PythonPythonPython!
```

**说明：**

☑ 重复运算符“*”不能用于字符串与字符串相乘，只用于字符串和整数相乘。

☑ 思考：为什么运行结果中有一个空行？

### 2.1.5 字符串函数

1. 字符串检测函数

- type()：查看数据类型。
- len()：计算字符串长度。
- .startswith('特定字符串')：检测字符串是否以特定字符串开头。
- .endswith('特定字符串')：检测字符串是否以特定字符串结尾。
- .isalnum()：检测字符串是否全为字母或数字组成。
- .isalpha()：检测字符串是否只由字母组成。
- .isdigit()：检测字符串是否只由数字组成。
- .islower()：检测字符串是否全由小写字母组成。
- .isupper()：检测字符串是否全由大写字母组成。
- .isspace()：检测字符串是否只由空格组成。
- .istitle()：检测字符串中所有单词是否首字母大写，且其他字母小写。

**【实例 2.4】字符串检测函数的使用。**

```
#Example2.4.py

sentence='''Python is a programming language that lets you work quickly
  and integrate systems
more effectively.'''
```

```
word='''Python'''
print('type(sentence):',type(sentence))
print('len(sentence):',len(sentence))
print("sentence.startswith('Python'):",sentence.startswith('Python'))
print("sentence.endswith('ly'):",sentence.endswith('ly'))
print("word.isalnum():",word.isalnum())
print("sentence.isalpha():",sentence.isalpha())
print("word.isalpha():",word.isalpha())
print("word.isdigit():",word.isdigit())
print("word.islower():",word.islower())
print("word.isupper():",word.isupper())
print("sentence.istitle():",sentence.istitle())
print("word.istitle():",word.istitle())
print("word.isspace():",word.isspace())
```

【运行结果】

```
type(sentence): <class 'str'>
len(sentence): 100
sentence.startswith('Python'): True
sentence.endswith('ly'): False
word.isalnum(): True
sentence.isalpha(): False
word.isalpha(): True
word.isdigit(): False
word.islower(): False
word.isupper(): False
sentence.istitle(): False
word.istitle(): True
word.isspace(): False
```

 思考：

- ☑ sentence.endswith('ly')的结果为什么是 False?
- ☑ sentence.isalpha()的结果为什么是 False?
- ☑ sentence.istitle()的结果为什么是 False?

2. 字符串字母处理函数

- .upper()：字符串所有字母变为大写。
- .lower()：字符串所有字母变为小写。
- .swapcase()：字符串所有字母大小写互换。
- .capitalize()：字符串首字母变为大写，其余小写。
- .title()：字符串所有单词首字母变为大写，其余小写。

【实例 2.5】字符串的字母处理。

```
#Example2.5.py

sentence='''software quality is a vital ingredient to success in industry
  and science. Ubiquitous IT systems control the business processes of
  the global economy. '''
print('sentence.upper():',sentence.upper())
print('sentence.swapcase():',sentence.swapcase())
```

```
print("sentence.capitalize():",sentence.capitalize())
print("sentence.title():",sentence.title())
print("sentence:",sentence)
```

【运行结果】

```
sentence.upper(): SOFTWARE QUALITY IS A VITAL INGREDIENT TO SUCCESS IN
  INDUSTRY AND SCIENCE. UBIQUITOUS IT SYSTEMS CONTROL THE BUSINESS
  PROCESSES OF THE GLOBAL ECONOMY.
sentence.swapcase(): SOFTWARE QUALITY IS A VITAL INGREDIENT TO SUCCESS
  IN INDUSTRY AND SCIENCE. uBIQUITOUS it SYSTEMS CONTROL THE BUSINESS
  PROCESSES OF THE GLOBAL ECONOMY.
sentence.capitalize(): Software quality is a vital ingredient to success
  in industry and science. ubiquitous it systems control the business
  processes of the global economy.
sentence.title(): Software Quality Is A Vital Ingredient To Success In
  Industry And Science. Ubiquitous It Systems Control The Business
  Processes Of The Global Economy.
sentence: software quality is a vital ingredient to success in industry
  and science. Ubiquitous IT systems control the business processes of
  the global economy.
```

 思考：

☑ 注意比较 sentence.capitalize()与 sentence.title()的结果。

☑ 观察各个函数是否改变了 sentence 的初始值。

3. 字符串查找与替换函数

● .find()

语法：str.find(sub_str, begin=0, end=len(string))

从字符串 str 中 begin 开始的位置到 end 结束的位置，查找指定子串 sub_str 出现的位置，如果找到返回子串 sub_str 所在的索引值（从字符串第一个字符开始计算），如果没有找到指定子串，则返回–1。

☑ 字符串的索引值从 0 开始。

☑ begin 值可以省略，默认为 0，即从字符串第一个字符开始。

☑ end 值可以省略，默认为字符串的长度，即到字符串结束为止。

● .rfind()

语法：str.rfind(sub_str, begin=0, end=len(string))

在 str 字符串 begin 至 end 范围内，从右边开始查找指定子串 sub_str 出现的位置，即子串最后一次出现的位置。如果找到返回子串 sub_str 所在的索引值（从字符串第一个字符开始计算），如果没有找到指定子串，则返回–1。

☑ 如果 begin 值大于 end，函数返回–1。

☑ begin 值可以省略，默认为 0，即从字符串第一个字符开始。

☑ end 值可以省略，默认为字符串的长度，即到字符串结束为止。

● .index()

语法：str.index(sub_str, begin=0, end=len(string))

从字符串 str 中 begin 开始的位置到 end 结束位置，查找指定子串 sub_str 出现的位置，

如果找到返回子串 sub_str 所在的索引值（从字符串第一个字符开始计算），如果没有找到指定子串，则抛出 VaueError 异常。

☑ find()与 index()的区别在于：index 如果查找不到指定子串抛出 VaueError 错误异常，而 find 查找不到子串时返回值–1。

☑ begin 值可以省略，默认为 0，即从字符串第一个字符开始。

☑ end 值可以省略，默认为字符串的长度，即到字符串结束为止。

● .count()

语法：str.count(sub_str, begin=0, end=len(string))

从字符串 str 中 begin 开始的位置到 end 结束位置，查找指定子串 sub_str 出现的次数并作为返回值。

☑ begin 值可以省略，默认为 0，即从字符串第一个字符开始。

☑ end 值可以省略，默认为字符串的长度，即到字符串结束为止。

● .replace()

语法：str.replace(子字符串 1, 子字符串 2[, 最大替换次数])

将字符串 str 中的子字符串 1 替换为子字符串 2，可以指定最大替换次数，默认替换所有符合条件的子串。

☑ 函数返回值为字符串中的子字符串 1 替换成子字符串 2 后生成的新字符串。

☑ 第三个参数“最大替换次数”可以省略，如果指定，则替换不超过指定的最大替换次数。

**【实例 2.6】字符串的查找与替换。**

```
#Example2.6.py

sentence='''Software quality is a vital ingredient to success in industry
  and science. Ubiquitous IT systems control the business processes of
  the global economy. '''
print("sentence.find('the'):",sentence.find('the'))
print("sentence.find('the',106):",sentence.find('the',106))
print("sentence.find('the',140):",sentence.find('the',140))
print("sentence.rfind('the'):",sentence.rfind('the'))
print("sentence.rfind('the',100,140):",sentence.rfind('the',100,140))
print("sentence.rfind('the',130,100):",sentence.rfind('the',130,100))
print("sentence.index('the'):",sentence.index('the'))
print("sentence.count('the'):",sentence.count('the'))
print("sentence.replace('the','THE'):",sentence.replace('the','THE'))
```

**【运行结果】**

```
sentence.find('the'): 105
sentence.find('the',106): 131
sentence.find('the',140): -1
sentence.rfind('the'): 131
sentence.rfind('the',100,140): 131
sentence.rfind('the',130,100): -1
sentence.index('the'): 105
sentence.count('the'): 2
sentence.replace('the','THE'): Software quality is a vital ingredient
  to success in industry and science. Ubiquitous IT systems control THE
  business processes of THE global economy.
```

思考：

☑ 请解释 sentence.find('the',140)的输出结果。

☑ 请解释 sentence.rfind('the',130,100)的输出结果。

【实例 2.7】字符串 **index** 查找不成功抛出异常。

```
#Example2.7.py

sentence='''Software quality is a vital ingredient to success in industry
  and science. Ubiquitous IT systems control the business processes of
  the global economy. '''
print("sentence.index('the'):",sentence.index('the'))
print("sentence.index('the',140):",sentence.index('the',140))
```

【运行结果】

```
File "D:/教材配套代码/第二章/Example2_7.py", line 10, in <module>
    print("sentence.index('the',140):",sentence.index('the',140))

ValueError: substring not found
```

4. 字符串的分割与连接函数

● .split()

语法：str.split(分隔符,分隔次数)

分隔符为可选参数，split()函数通过指定分隔符对字符串进行切片，默认的分隔符包括所有的空字符，如空格、换行(\n)、制表符(\t)等。

第二个可选参数“分隔次数”表示分割的次数，默认值为–1，表示不限制分割的次数；如果指定特定的值，分割后即得到分隔次数+1 个子字符串。

.split()函数的返回值为分割后的字符串列表。

● .rsplit()

语法：str.rsplit(分隔符,分隔次数)

从右侧开始分割字符串，语法同 split()。

● .partition()

语法：str.partition(分隔符)

分隔符不可省略，如果字符串包含指定的分隔符，则返回值为一个 3 元的元组，第一个为分隔符左边的子串，第二个为分隔符本身，第三个为分隔符右边的子串。

● .rpartition()

语法：str.rpartition(分隔符)

分隔符不可省略，从右侧开始分割字符串，如果字符串包含指定的分隔符，则返回值为一个 3 元的元组，第一个为分隔符左边的子串，第二个为分隔符本身，第三个为分隔符右边的子串。

● .join()

语法：连接字符.join(字符序列)

用于将序列中的元素以指定的字符连接生成一个新的字符串。返回值为指定的连接字符在连接字符序列后生成的新字符串。

【实例 2.8】字符串的分割与连接。

```
#Example2.8.py

sentence='''Software quality is a vital ingredient to success in industry
  and science. Ubiquitous IT systems control the business processes of
  the global economy.'''
print("sentence.split( ):",sentence.split( ))
print("sentence.split('.'):",sentence.split('.'))
print("sentence.split('.',1):",sentence.split('.',1))
print("sentence.rsplit('the',1 ):",sentence.rsplit('the',1))
print("sentence.partition('science'):",sentence.partition('science'))
link1 = "-"
link2 = ""
seq = ("p", "y", "t", "h", "o", "n")
print (link1.join( seq ))
print (link2.join( seq ))
```

【运行结果】

```
sentence.split( ): ['Software', 'quality', 'is', 'a', 'vital', 'in-
  gredient', 'to', 'success', 'in', 'industry', 'and', 'science.',
  'Ubiquitous', 'IT', 'systems', 'control', 'the', 'business',
  'processes', 'of', 'the', 'global', 'economy.']
sentence.split('.'): ['Software quality is a vital ingredient to success
  in industry and science', ' Ubiquitous IT systems control the business
  processes of the global economy', '']
sentence.split('.',1): ['Software quality is a vital ingredient to
  success in industry and science', ' Ubiquitous IT systems control the
  business processes of the global economy.']
sentence.rsplit('the',1 ): ['Software quality is a vital ingredient to
  success in industry and science. Ubiquitous IT systems control the
  business processes of ', ' global economy.']
sentence.partition('science'): ('Software quality is a vital ingredient
  to success in industry and ', 'science', '. Ubiquitous IT systems control
  the business processes of the global economy.')
p-y-t-h-o-n
python
```

**思考：**

☑ 请思考如何对一段英文文本进行分句。

5. 字符串填充对齐和删除指定字符

● .ljust()

语法：str.ljust(宽度[, 填充字符])

使用指定的填充字符填充字符串 str 到指定长度的新字符串，原字符串左对齐，填充的字符放在原字符串的右边。

填充字符可以省略，默认为填充空格。

返回值为新生成的字符串，如果指定的宽度小于原字符串的宽度度，则返回原字符串。

● .rjust()

语法：str.rjust(宽度[, 填充字符])

使用指定的填充字符，填充字符串 str 到指定长度的新字符串，原字符串右对齐，填充的字符放在原字符串的左边。

填充字符可以省略，默认为填充空格。

返回值为新生成的字符串，如果指定的宽度小于原字符串的宽度度，则返回原字符串。

● .center()

语法：str.center(宽度[, 填充字符])

使用指定的填充字符填充字符串 str 两端，字符串 str 在指定宽度中居中对齐。

填充字符可以省略，默认为填充空格。

返回值为新生成的字符串，如果指定的宽度小于原字符串的宽度度，则返回原字符串。

● .zfill(width)：获取固定长度，右对齐，左边不足用 0 补齐

语法：str.zfill(宽度)

使用字符“0”填充字符串 str 到指定长度的新字符串，原字符串右对齐，填充的字符“0”放在原字符串的左边。

返回值为新生成的字符串，如果指定的宽度小于原字符串的宽度度，则返回原字符串。

● .strip()

语法：str.strip([指定字符或字符序列])

用于移除字符串 str 前后的指定字符或字符序列，指定字符或字符序列可以省略，默认为空格。

返回值为移除字符串前后指定的字符序列生成的新字符串。

● .lstrip()

语法：str.lstrip([指定字符或字符序列])

用于移除字符串 str 前面的指定字符或字符序列，指定字符或字符序列可以省略，默认为空格。

返回值为移除字符串前后指定的字符序列生成的新字符串。

● .rstrip()

语法：str.rstrip([指定字符或字符序列])

用于移除字符串 str 后面的指定字符或字符序列，指定字符或字符序列可以省略，默认为空格。

返回值为移除字符串前后指定的字符序列生成的新字符串。

**【实例 2.9】填充对齐和删除指定字符。**

```
#Example2.9.py

str = "Python!!!"
print ('str.ljust(13):',str.ljust(13))
print ('str.ljust(13, "*"):',str.ljust(13, '*'))
print ('str.rjust(13):',str.rjust(13))
print ('str.rjust(13, "*"):',str.rjust(13, '*'))
print ('str.rjust(2, "*"):',str.rjust(2, '*'))
print ('str.center(13, "*"):',str.center(13, '*'))
print ('str.zfill(13):',str.zfill(13))
str1 = "*****Hello Python!!!*****"
print('str1.strip("*"):',str1.strip('*'))
print('str1.lstrip("*"):',str1.lstrip('*'))
```

```
print('str.rstrip("!"):',str.rstrip('!'))
```

【运行结果】

```
str.ljust(13): Python!!!
str.ljust(13, "*"): Python!!!****
str.rjust(13):     Python!!!
str.rjust(13, "*"): ****Python!!!
str.rjust(2, "*"): Python!!!
str.center(13, "*"): **Python!!!**
str.zfill(13): 0000Python!!!
str.strip("*"): Hello Python!!!
str.lstrip("*"): Hello Python!!!*****
str.rstrip("!"): Pythonpython
```

**说明：**

☑ 字符串类型是不可变的，因此上述方法的返回值是字符串的副本，并没有改变原来的字符串。

6. 字符串的转换

● .maketrans( )

语法：str.maketrans(源字符表,目标字符表)

用于创建字符映射的转换表，第一个参数是字符串，表示需要转换的源字符，第二个参数也是字符串，表示转换的目标字符。

●. translate()

语法：str.translate(转换表)

根据参数给出的转换表一一转换字符串 str 中的字符。

返回值为字符串转换后生成的新字符串。

【实例 2.10】字符串的转换。

```
#Example2.10.py

source_lang= "Python"
target_lang = "我们是好朋友"
trantab = str.maketrans(source_lang, target_lang) # 制作转换表
str = "Hey, Python!!!"
print(str.translate(trantab))
```

【运行结果】

```
Hey, 我们是好朋友!!!
```

**说明：**

☑ 两个字符串的长度必须相同，为一一对应的关系。

## 2.2 数 值 类 型

Python 3 支持的数字类型包括整数 int、浮点数 float 和复数 complex。Python 内置的 type()

函数可以用来查询对象所属的类型。

### 2.2.1 整数 int

在 Python 3 中，整数类型取消了 Python2 中整型与长整型的区别，只有一种整型数。Python3 的整数类型的取值范围仅与机器支持的内存大小有关，可以超过机器位数所能表示的数值范围表示很大的数。

Python 3 的整数类型可以是正整数或负整数，不带小数位数。在 Python 程序中数值类型的赋值和计算都是很直观的，例如：1024，–8086，0，等等。在 Python 中，可对整数执行加法（+）减法（–）乘法（*）除法（/、//）运算。

**【实例 2.11】Python3 整数的基本运算。**

```
#Example2.11.py

a=10
b=20
print("a+b=",a+b)
print("a-b=",a-b)
print("a*b=",a*b)
print("a/b=",a/b)
print("a//b=",a//b)
```

**【运行结果】**

```
a+b= 30
a-b= -10
a*b= 200
a/b= 0.5
a//b= 0
```

**整数除法的说明：**

☑ 在 Python 3 中，无论运算符“/”前后两个操作数是否浮点数，除法结果总是返回一个浮点数。

☑ 在 Python 3 中，运算符“//”除法结果向下取整，结果类型与分母分子的数据类型有关。

☑ 在 Python 2 中，运算符“//”效果与 Python 3 相同，如果运算符“/”前后两个操作数都是整数，除法结果将向下取整，结果是整数；如果两个操作数有浮点数，就是浮点数除法，结果是浮点数。

### 2.2.2 计算机中数的进制

日常生活中，数通常以十进制形式表示。在计算机的内部，数据的表示、处理和存储都基于电子器件的“0”或“1”状态，因此以二进制形式为基础。二进制数往往要用一长串“0”或“1”代码表示，为了提高表示效率，在计算机中数的处理引入了八进制和十六进制。

在计算机的数制中，数码、基数和位权是数制的三要素。

☑ 数码：一个数制中表示基本数值大小的不同数字符号。例如，十进制有 0、1、2、3、4、5、6、7、8、9 共 10 个数码。

☑ 基数：一个数值所使用数码的个数。例如，二进制的基数为 2，十进制的基数为 10。

☑ 位权：一个数值中某一位上的 1 所表示数值的大小。如十进制的 123，百位上的 1 的位权是 100，十位上的 2 的位权是 10，个位上的 3 的位权是 1。

1. 二进制

由于计算机的物理实现上，具有两种不同稳定状态且能相互转换的电子器件是很容易找到，例如：电位的高低、晶体管的导通截止、磁化的正反方向、脉冲的有无、开关的闭合断开等，都可以用 0 和 1 两种状态对应。同时，这些物理器件的状态稳定可靠，抗干扰能力强。

因此，从易得性、可靠性、可行性、逻辑性等各方面考虑，计算机内部选择二进制数字系统，通过两种不同状态值的组合来表示计算机内部的所有信息。

### 二进制的要素：

☑ 数码：二进制有 0、1 共两个数码。

☑ 基数：二进制的基数即使用数码的个数为 2。

☑ 位权：二进制数个位上的位权是 $2^0$=1，从右往左第二位上的位权是 $2^1$=2，第三位上的位权是 $2^2$=4，以此类推。

☑ 运算规则：二进制运算规则简单，加法运算“逢二进一”，减法运算“借一当二”。二进制的加法、乘法的基本运算规则如下：

0+0=0　　0+1=1　　1+0=1　　1+1=10

0×0=0　　0×1=0　　1×0=0　　1×1=1

☑ 二进制转换为十进制的方法：按权展开。

☑ 在 Python 中二进制数用“0b”加相应数字来表示，如“0b1001”表示二进制数“1001”。

**【实例 2.12】二进制数转换为十进制数。**

$(1101.001)_2=1\times2^3+1\times2^2+0\times2^1+1\times2^0+0\times2^{-1}+0\times2^{-2}+1\times2^{-3}$

$=8+4+1+0.125$

$=(13.125)_{10}$

$(1101100.111)_2=1\times2^6+1\times2^5+1\times2^3+1\times2^2+1\times2^{-1}+1\times2^{-2}+1\times2^{-3}$

$=64+32+8+4+0.5+0.25+0.125$

$=(108.875)_{10}$

2. 八进制

为了简化计算机中二进制冗长的表示，引入了八进制。

### 八进制的要素：

☑ 数码：八进制有 0、1、2、3、4、5、6、7 共八个数码。

☑ 基数：八进制的基数即使用数码的个数为 8。

☑ 位权：八进制数个位上的位权是 80=1，从右往左第二位上的位权是 81=8，第三位上的位权是 82=64，以此类推。

☑ 运算规则：八进制加法运算“逢八进一”，减法运算“借一当八”。

☑ 八进制转换为十进制的方法：按权展开。

☑ 在 Python 中八进制数用“0o”加相应数字来表示，如“0o71”表示八进制数“71”。

**【实例 2.13】八进制数转换为十进制数。**

$(318)_8=3\times8^2+1\times8^1+8\times8^0=192+8+8=(208)_{10}$

$(652.4)_8=6\times8^2+5\times8^1+2\times8^0+4\times8^{-1}$

$=384+40+2+0.5$

$=(426.5)_{10}$

3. 十六进制

十六进制能够进一步简化计算机中二进制冗长的表示。

### 十六进制的要素：

☑ 数码：十六进制有 0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F 共十六个数码，其中数码 A 对应十进制的 10，B 对应十进制的 11，以此类推，数码 F 对应十进制的 15。

☑ 在 Python 中，十六进制的字母数码不区分大小写。

☑ 基数：十六进制的基数即使用数码的个数为 16。

☑ 位权：十六进制数个位上的位权是 $16^0=1$，从右往左第二位上的位权是 $16^1=16$，第三位上的位权是 $16^2=256$，以此类推。

☑ 运算规则：十六进制加法运算“逢十六进一”，减法运算“借一当十六”。

☑ 十六进制转换为十进制的方法：按权展开。

☑ 在 Python 中十六进制数用“0x”加相应数字来表示，如“0xA9”表示十六进制数“A9”。

**【实例 2.14】十六进制数转换为十进制数。**

$(3C7)_{16}=3\times16^2+12\times16^1+7\times16^0=768+192+7=(967)_{10}$

$(AB9.5)_{16}=10\times16^2+11\times16^1+9\times16^0+5\times16^{-1}$

$=2560+176+9+0.3125$

$=(2745.3125)_{10}$

4. 进制转换

### 其他进制转换为十进制

☑ 转换规则：以进制为基数按权展开并相加，例如【实例 2.12】【实例 2.13】和【实例 2.14】。

☑ 整数与小数部分同规则。

## 十进制转换为二进制、八进制和十六进制

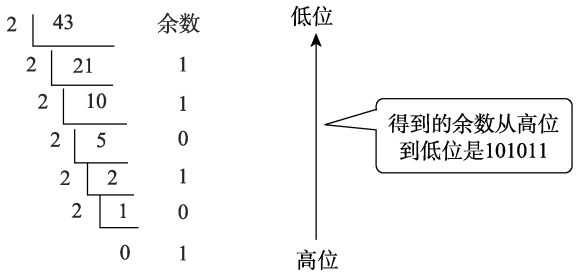
☑ 整数部分转换规则：除基数取余数、直到商为 0，余数由下而上取，得到整数部分的由高位到低位排列。

☑ 小数部分转换规则：乘基数取整数，直到小数的当前值为 0，或者满足精度要求为止，取的整数由上而下取，得到小数部分的由高位到低位排列。

**【实例 2.15】十进制数转换为二进制数。**

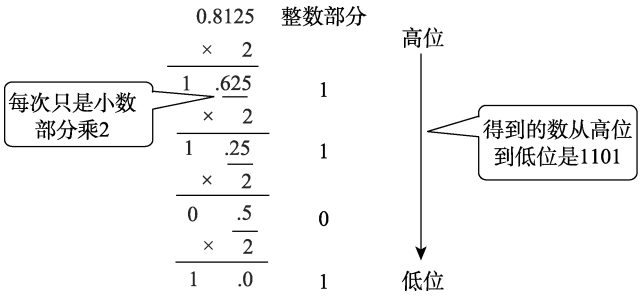
填空题：$(43.8125)_{10}=$ (　　)$_2$

（1）先转换整数部分：




计算得到$(43)_{10}=(101011)_2$

（2）再转换小数部分：




计算得到$(0.8125)_{10}=(0.1101)_2$

综合整数部分和小数部分，最终得到转换结果：

$(43.8125)_{10}=(101011.1101)_2$

## 思考

☑ 试分析十进制转换为二进制时，为什么整数部分和小数部分的转换一个是由下而上取，一个是由上而下取？

☑ 十进制转换为八进制、十六进制同上例题，基数换为相应进制的基数 8 或 16 即可。

## 二进制与八进制、十六进制间的转换

☑ 观察表 2.3，可以发现有 1 位八进制对应 3 位二进制，1 位十六进制对应 4 位二进

制的规律。

☑ 二进制转换为八进制：二进制整数部分从右向左 3 位并 1 位转换成八进制，不够 3 位补 0；二进制小数部分从左向右 3 位并 1 位转换成八进制，不够 3 位补 0。

☑ 二进制转换为十六进制：二进制整数部分从右向左 4 位并 1 位转换成十六进制，不够 4 位补 0；二进制小数部分从左向右 4 位并 1 位转换成十六进制，不够 4 位补 0。

☑ 八进制转换为二进制：八进制 1 位转换成 3 位二进制。

☑ 十六进制转换为二进制：十六进制 1 位转换成 4 位二进制。

☑ 八进制与十六进制之间的转换：可以以二进制为中间进制，如八进制转换为二进制，再转换为十六进制。

**表 2.3　十进制、二进制、八进制、十六进制的对应关系**

| 十进制 | 二进制 | 八进制 | 十六进制 | 十进制 | 二进制 | 八进制 | 十六进制 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 16 | 10000 | 20 | 10 |
| 1 | 1 | 1 | 1 | 17 | 10001 | 21 | 11 |
| 2 | 10 | 2 | 2 | 18 | 10010 | 22 | 12 |
| 3 | 11 | 3 | 3 | 19 | 10011 | 23 | 13 |
| 4 | 100 | 4 | 4 | 20 | 10100 | 24 | 14 |
| 5 | 101 | 5 | 5 | 21 | 10101 | 25 | 15 |
| 6 | 110 | 6 | 6 | 22 | 10110 | 26 | 16 |
| 7 | 111 | 7 | 7 | 23 | 10111 | 27 | 17 |
| 8 | 1000 | 10 | 8 | 24 | 11000 | 30 | 18 |
| 9 | 1001 | 11 | 9 | 25 | 11001 | 31 | 19 |
| 10 | 1010 | 12 | A | 26 | 11010 | 32 | 1A |
| 11 | 1011 | 13 | B | 27 | 11011 | 33 | 1B |
| 12 | 1100 | 14 | C | 28 | 11100 | 34 | 1C |
| 13 | 1101 | 15 | D | 29 | 11101 | 35 | 1D |
| 14 | 1110 | 16 | E | 30 | 11110 | 36 | 1E |
| 15 | 1111 | 17 | F | 31 | 11111 | 37 | 1F |

**【实例 2.16】二进制与十六进制之间的转换。**

**填空题一**：$(4A3C.8D25)_{16}=$ (　　)$_2$

十六进制转换为二进制规则是 1 位转换成 4 位，对照表 2.3 可得：

$$(4A3C.8D25)_{16}=(100'1010'0011'1100.1000'1101'0010'0101)_2$$

**填空题二**：$(110100111.100010011)_2=$ (　　)$_{16}$

二进制转换为十六进制规则是 4 位转换成 1 位，可以从小数点往两端划分 4 位一组，不够 4 位补 0。对照表 2.3 可得：

$$(0001'1010'0111.1000'1001'1000)_2=(1A7.898)_{16}$$

**【实例 2.17】二进制与八进制之间的转换。**

**填空题一**：$(453.25)_8=$ (　　)$_2$

八进制转换为二进制规则是 1 位转换成 3 位，对照表 2.3 可得：

$$(453.25)_8=(100'101'011.010'101)_2$$

**填空题二**：$(10100111.10001001)_2=$ (　　)$_8$

二进制转换为八进制规则是 3 位转换成 1 位，可以从小数点往两端划分 3 位一组，不够 3 位补 0。对照表 2.3 可得：

$$(010'100'111.100'010'010)_2=(247.422)_8$$



## Python 的进制转换函数

☑ 转换为十进制数

int（其他进制数），返回值为相应的十进制数。

int（数字组成的字符串，进制基数），将数字组成的字符串按照进制基数读取，然后返回为相应的十进制数。

☑ 转换为二进制数

bin（其他进制数），返回值为相应的二进制数。

☑ 转换为八进制数

oct（其他进制数），返回值为相应的八进制数。

☑ 转换为十六进制数

hex（其他进制数），返回值为相应的十六进制数。

**【实例 2.18】进制转换函数。**

```
#Example2.18.py

bin_number = 0b10111111
print("二进制数 bin_number=","0b10111111")
print("转换为十进制 int(bin_number)=",int(bin_number))
print("转换为十进制 int('10111111',2)=",int('10111111',2))
print("转换为八进制 oct(bin_number)=",oct(bin_number))
print("转换为十六进制 hex(bin_number)=",hex(bin_number))
oct_number = 0o277
print("八进制数 oct_number=","0o277")
print("转换为十进制 int(oct_number)=",int(oct_number))
print("转换为二进制 bin(oct_number)=",bin(oct_number))
print("转换为十六进制 hex(oct_number)=",hex(oct_number))
hex_number = 0xbf
print("十六进制数 hex_number=","0xbf")
print("转换为十进制 int(hex_number)=",int(hex_number))
print("转换为二进制 bin(hex_number)=",bin(hex_number))
print("转换为八进制 oct(hex_number)=",oct(hex_number))
```

**【运行结果】**

```
二进制数 bin_number= 0b10111111
转换为十进制 int(bin_number)= 191
转换为十进制 int('10111111',2)= 191
转换为八进制 oct(bin_number)= 0o277
转换为十六进制 hex(bin_number)= 0xbf
```

```
八进制数 oct_number= 0o277
转换为十进制 int(oct_number)= 191
转换为二进制 bin(oct_number)= 0b10111111
转换为十六进制 hex(oct_number)= 0xbf
十六进制数 hex_number= 0xbf
转换为十进制 int(hex_number)= 191
转换为二进制 bin(hex_number)= 0b10111111
转换为八进制 oct(hex_number)= 0o277
```

### 说明

☑ int('10111111',2)是 int（数字组成的字符串，进制基数）形式的实例，返回值为将 10111111 看做二进制数字时，相应的十进制数值。

## 2.2.3 浮点数（float）

浮点数由整数部分与小数部分组成。由于按照科学记数法表示时，一个浮点数的小数点位置是可变的，因此称为浮点数。整数和浮点数在计算机内部存储的方式不同，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。

浮点数可以使用科学计数法表示，例如 0.0000001024 可以用科学记数法表示为 1.024e–7。

**【实例 2.19】圆周率的格式化输出。**

```
#Example2.19.py

pi=3.1415926
print('圆周率的值为%d '%pi)          #十进制整数形式输出
print('圆周率的值为%e '%pi)          #指数形式输出
print('圆周率的值为%f '%pi)          #浮点数形式输出
print('圆周率的值为%.2f '%pi)        #浮点数形式输出，指定精度为小数点后 2 位
print('圆周率的值为%10.2f '%pi)      #浮点数形式输出，指定宽度为 10，其中包括小数
                                      点后 2 位
```

**【运行结果】**

```
圆周率的值为 3
圆周率的值为 3.141593e+00
圆周率的值为 3.141593
圆周率的值为 3.14
圆周率的值为       3.14
```

## 2.2.4 复数（complex）

复数是由一个实数和一个虚数组合构成，表示为：a+bj

其中，有序浮点数(a,b)中，a 表示实数部分，b 表示虚数部分。

在 Python 语言中，复数对象拥有数据属性，分别为该复数的实数部分和虚数部分。

Python 语言中还可以调用 conjugate 方法返回该复数的共轭复数对象。复数的表示需要用到复合类型，本章不做详述。



## 2.3 布 尔 类 型

布尔类型只有两种值：True 和 False，分别表示逻辑上的真和假，在数值上下文环境中，分别表示为 1 和 0。

在 Python 中，可以直接用 True、False 表示布尔值（注意首字母大写）。

### 说明：

☑ 参与算术运算时，True 被当作 1，False 被当作 0，如 True+5 的结果是 6。

☑ 其他类型值转换为布尔值时除了 ''、""、''''、""""""、0、()、[]、{}、None、0.0、0L、0.0+0.0j、False 为 False 外，其他都为 True。

## 2.4 数据类型转换函数

本节介绍基本数据类型：字符串、整数、浮点数之间数据类型转换的函数。

### Python 的数据类型转换函数

☑ 转换为字符串

str（其他数据类型），返回值为相应的字符串。

☑ 转换为十进制整数

int（其他数字类型或由数字组成的字符串），返回值为相应的十进制整数。

☑ 转换为浮点数

float（其他数据类型），返回值为相应的浮点数。

☑ 转换为布尔值

bool（其他数据类型），返回值为相应的布尔值。

**【实例 2.20】数据类型转换函数。**

```
#Example2_20.py

#其他数据类型转换为整数
print("字符串'37'转换为十进制整数 int('37',10)=",int('37',10))
print("浮点数 8848.8 转换为十进制整数 int(8848.8)=",int(8848.8))
print("布尔值 False 转换为十进制整数 int(False)=",int(False))
#其他数据类型转换为浮点数
print("字符串'37'转换为浮点数 float('37')=",float('37'))
print("整数 1024 转换为浮点数 float(1024)=",float(1024))
print("type(float(1024))=",type(float(1024)))
print("布尔值 False 转换为浮点数 float(False)=",float(False))
#其他数据类型转换为布尔值
```

```
print("字符串'37'转换为布尔值 bool('37')=",bool('37'))
print("整数 1024 转换为布尔值 bool(1024)=",bool(1024))
print("浮点数 8848.8 转换为布尔值 bool(8848.8)=",bool(8848.8))
#其他数据类型转换为字符串
print("布尔值 False 转换为字符串 str('37')=",str('37'))
print("整数 1024 转换为字符串 str(1024)=",str(1024))
print("浮点数 8848.8 转换为字符串 str(8848.8)=",str(8848.8))
print("type(str(8848.8))=",type(str(8848.8)))
```

【运行结果】

```
字符串'37'转换为十进制整数 int('37',10)= 37
浮点数 8848.8 转换为十进制整数 int(8848.8)= 8848
布尔值 False 转换为十进制整数 int(False)= 0
字符串'37'转换为浮点数 float('37')= 37.0
整数 1024 转换为浮点数 float(1024)= 1024.0
type(float(1024))= <class 'float'>
布尔值 False 转换为浮点数 float(False)= 0.0
字符串'37'转换为布尔值 bool('37')= True
整数 1024 转换为布尔值 bool(1024)= True
浮点数 8848.8 转换为布尔值 bool(8848.8)= True
布尔值 False 转换为字符串 str('37')= 37
整数 1024 转换为字符串 str(1024)= 1024
浮点数 8848.8 转换为字符串 str(8848.8)= 8848.8
type(str(8848.8))= <class 'str'>
```

**说明：**

☑ 实例中 type()函数返回值为数据对象的类型，如<class 'str'>表示字符串类型。

# 2.5 运 算 符

## 2.5.1 算术运算符

Python 中的算术运算符如表 2.4 所示。

**表 2.4 算术运算符**

| 运算符 | 描述 |
|---|---|
| + | 两个操作数相加 |
| – | 负数或是一个数减去另一个数 |
| * | 两个数相乘或是返回一个被重复若干次的字符串 |
| / | 除法运算，如 x/y 返回 x 除以 y 的结果 |
| % | 取模运算，返回除法的余数 |
| ** | 幂运算，如：x**y 返回 x 的 y 次幂 |
| // | 取整除运算，返回值为商的向下取整的整数部分 |

【实例 2.21】算术运算符。

```
#Example2.21.py

a = 21
b = 4
c = 0
c = a + b
print ("a+b 的值为：", c)
c = a - b
print ("a-b 的值为：", c)
c = a * b
print ("a*b 的值为：", c)
c = a / b
print ("a/b 的值为：", c)
c = a % b
print ("a%b 的值为：", c)
c = a**b
print ("a**b 的值为：", c)
c = a//b
print ("a//b 的值为：", c)
```

【运行结果】

```
a+b 的值为： 25
a-b 的值为： 17
a*b 的值为： 84
a/b 的值为： 5.25
a%b 的值为： 1
a**b 的值为： 194481
a//b 的值为： 5
```

### 2.5.2 赋值运算符

Python 中的赋值运算符如表 2.5 所示。

**表 2.5 赋值运算符**

| 运算符 | 描述 |
|---|---|
| = | 简单的赋值运算符 |
| += | 加法赋值运算符，a+=b 等效于 a=a+b |
| -= | 减法赋值运算符，a-=b 等效于 a=a-b |
| *= | 乘法赋值运算符，a*=b 等效于 a=a*b |
| /= | 除法赋值运算符，a/=b 等效于 a=a/b |
| %= | 取模赋值运算符，a%=b 等效于 a=a%b |
| **= | 幂赋值运算符，a**=b 等效于 a=a**b |
| //= | 取整除赋值运算符，a//=b 等效于 a=a//b |

【实例 2.22】赋值运算符。

```
#Example2.22.py

a = 21
b = 4
a += b
print ("a+=b 后，a 的值为：",a )
a -= b
print ("a-=b 后，a 的值为：",a )
a *= b
print ("a*=b 后，a 的值为：",a )
a /= b
print ("a/=b 后，a 的值为：", a)
a %= b
print ("a%=b 后，a 的值为：", a)
a**=b
print ("a**=b 后，a 的值为：", a)
a//=b
print ("a//=b 后，a 的值为：", a) a//=b
```

【运行结果】

```
a+=b 后，a 的值为： 25
a-=b 后，a 的值为： 21
a*=b 后，a 的值为： 84
a/=b 后，a 的值为： 21.0
a%=b 后，a 的值为： 1.0
a**=b 后，a 的值为： 1.0
a//=b 后，a 的值为： 0.0
```

**说明：**

☑ 尝试改写【实例 2.21】，输出 a 值的变化，并比较【实例 2.21】和【实例 2.22】中 a 值的变化。

☑ 如果给多个变量赋同样的值，可以采用链式赋值，如：a=b=c=d=1。

### 2.5.3 比较运算符

比较运算符也称为关系运算符，Python 中的比较运算符如表 2.6 所示。

**表 2.6 比较运算符**

| 运算符 | 描述 |
|---|---|
| == | 比较对象是否相等，返回布尔值 True 或 False |
| != | 比较两个对象是否不相等，返回布尔值 True 或 False |
| > | 比较 a>b 中，a 是否大于 b，返回布尔值 True 或 False |
| < | 比较 a<b 中，a 是否小于 b，返回布尔值 True 或 False |
| >= | 比较 a>=b 中，a 是否大于等于 b，返回布尔值 True 或 False |
| <= | 比较 a<=b 中，a 是否小于等于 b，返回布尔值 True 或 False |

【实例 2.23】比较运算符。

```
#Example2.23.py

a = 21
b = 4
if ( a == b ):
   print ("第一次比较：a 等于 b")
else:
   print ("第一次比较：a 不等于 b")
if ( a != b ):
   print ("第二次比较： a 不等于 b")
else:
   print ("第二次比较： a 等于 b")
if ( a < b ):
   print ("第三次比较： a 小于 b")
else:
   print ("第三次比较： a 大于等于 b")
if ( a > b ):
   print ("第四次比较： a 大于 b")
else:
   print ("第四次比较： a 小于等于 b")
if ( a <= b ):
   print ("第五次比较： a 小于等于 b")
else:
   print ("第五次比较： a 大于 b")
if (  a>= b ):
   print ("第六次比较： a 大于等于 b")
else:
   print ("第六次比较： a 小于 b")
```

【运行结果】

```
第一次比较： a 不等于 b
第二次比较： a 不等于 b
第三次比较： a 大于等于 b
第四次比较： a 大于 b
第五次比较： a 大于 b
第六次比较： a 大于等于 b
```

**说明：**

☑ 注意，代码中的缩进和符号“:”不能缺失，后续章节将详细介绍 if-else 语句。

### 2.5.4 逻辑运算符

逻辑运算符用来连接布尔值进行逻辑运算，运算结果也是布尔值。Python 语言支持的逻辑运算符如表 2.7 所示。

表 2.7　逻辑运算符

| 运算符 | 描述 | 运算规则 | |
|---|---|---|---|
| and | "与"运算 | 真 and 真 | 真 |
| | | 真 and 假 | 假 |
| | | 假 and 真 | 假 |
| | | 假 and 假 | 假 |
| or | "或"运算 | 真 or 真 | 真 |
| | | 真 or 假 | 真 |
| | | 假 or 真 | 真 |
| | | 假 or 假 | 假 |
| not | "非"运算 | not 真 | 假 |
| | | not 假 | 真 |

【实例 2.24】逻辑运算符。

```
#Example2.24.py

a = 21
b = 0
if ( a and b ):
   print ("a and b：a 和 b 都为 True")
else:
   print ("a and b：a 和 b 有一个不为 True")
if ( a or b ):
   print (" a or b: a 和 b 都为 True，或其中一个变量为 True")
else:
   print (" a or b: a 和 b 都不为 True")
if not a :
   print (" not a: a 为 False ")
else:
   print (" not a: a 为 True")
```

【运行结果】

```
a and b：a 和 b 有一个不为 True
a or b: a 和 b 都为 True，或其中一个变量为 True
not a: a 为 True
```

 说明：

☑ 注意代码中的缩进和符号“:”不能缺失，后续章节将详细介绍 if-else 语句。

### 2.5.5　运算符的优先级

Python 运算符从高到低的优先级顺序如表 2.8 所示。

表 2.8　运算符优先级

| 优先级 | 运算符 | 描述 |
| --- | --- | --- |
| 最高 | ** | 指数运算 |
| | ~ + − | 按位翻转，一元加号和减号 |
| | * / % // | 乘、除、取模和取整除法 |
| | + − | 加法、减法 |
| | >> << | 右移左移运算符 |
| | & | 位‘AND’ |
| | ^ | 位运算符 |
| | <= >= > < | 比较运算符：小于等于、大于等于、大于、小于 |
| | == != | 比较运算符：等于、不等于 |
| | = %= /= //= -= += *= **= | 赋值运算符 |
| | is is not | 身份运算符 |
| | in not in | 成员运算符 |
| 低 | Not or and | 逻辑运算符 |

# 2.6　变量和常量

## 2.6.1　Python 变量

变量是计算机内存中的一块区域，每个变量在内存中的信息包括变量的标识、变量的名称和变量中存储的数据。

Python 中的变量不需要声明，变量的赋值操作就完成了变量的声明和定义。Python 变量名在引用前必须先赋值。

Python 允许你同时为多个变量赋值。例如：

```
a = b = c = 5
d, e, f = 1, 2, 3
```

上面第一行代码变量 a、b、c 的赋值均为 5。这三个变量被分配到相同的内存空间上。第二行代码变量 d、e、f 的赋值分别为 1、2、3。

1. Python 变量命名规则

Python 中变量的名称由字母、数字、下画线组成，并且不能以数字开头。

变量名不能用 Python 关键字和函数名，这些是 Python 用于特殊用途的保留字。Python3.7 有 35 个保留字，具体包括：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。

2. 变量作用域

变量作用域是指在程序中命名的变量，在什么范围内能被访问到。根据变量作用域，可分为局部变量和全局变量。

局部变量是只能在函数或代码块内部使用的变量，函数或者代码块结束，局部变量的生命周期也结束。

全局变量是在函数之外定义的变量，能够被文件内不同的函数、类或者外部文件访问的变量。

在程序中应尽量避免使用全局变量，避免由于不同模块自由访问全局变量而带来的全局变量不可预知性。此外，全局变量还会降低函数或模块之间的通用性。

### 2.6.2 Python 常量

常量是一块只读的内存区域，常量一旦被初始化就不能被改变。因此，常量是不能修改的固定值，例如字符串常量"Hello Python"在运行时一直都不会发生变化。

## 2.7 空　　值

空值是 Python 中一个特殊的值，在 Python 中用 None 表示。None 不能理解为 0，因为 0 是有意义的，而 None 则是一个特殊的空值。

## 2.8 本 章 小 结

本章首先介绍了 Python 基本数据类型中的字符串，包括字符串的输入、转义字符串、字符串的格式化、字符串运算符、字符串函数等。然后讲解了 Python 中的数字类型，包括整数、浮点数和复数，还介绍了计算机中数的进制，包括二进制、八进制、十六进制以及进制转换。接着介绍了 Python 中的布尔类型，以及 Python 的数据类型转换函数。

Python 的运算符主要有算术运算符、赋值运算符、比较运算符和逻辑运算符，本章还介绍了各类运算符的优先级。

本章还介绍了变量和常量的概念，以及 Python 中的变量命名规则。

## 2.9 习　　题

### 一、单项选择题

1. 执行语句 print( 2*"Today")的输出结果是（　　）。
   A. Today　　B. To　　C. ay　　D. TodayToday
2. 执行语句 print( "Today".upper())的输出结果是（　　）。
   A. Today　　B. tODAY　　C. today　　D. TODAY
3. 执行语句 print( "Today".swapcase())的输出结果是（　　）。
   A. Today　　B. tODAY　　C. today　　D. TODAY
4. 执行语句 print( "today".title())的输出结果是（　　）。
   A. Today　　B. tODAY　　C. today　　D. TODAY
5. 执行语句 print( "today".replace('to', 'mon'))的输出结果是（　　）。
   A. Today　　B. mONDAY　　C. monday　　D. MONDAY

6. 执行语句 print( "today".zfill(8))的输出结果是（　　）。

A. Today000　　B. to000day　　C. today000　　D. 000today

7. 执行语句 print( "today".center(9,'*'))的输出结果是（　　）。

A. Today****　　B. to****day　　C. today****　　D. **today**

8. 执行语句 print( "Today is Monday.".split())的输出结果是（　　）。

A. Today is Monday.　　B. ['Today', 'is', 'Monday.']
C. Today, is, Monday.　　D. [Today] [is] [Monday.]

9. 执行语句 print('\nToday \t is \t a\t sunny\n day!'.split(None, 3))的输出结果是（　　）。

A. ['Today is a sunny day!']　　B. ['Today', 'is', 'a', 'sunny', 'day!']
C. ['Today', 'is', 'a', 'sunny\n day!']　　D. ['\nToday \t is \t a\t sunny\n day!']

10. 执行语句 print( "Today is Monday.".rsplit(' ',1))的输出结果是（　　）。

A. [Today is Monday.]　　B. ['Today', 'is', 'Monday.']
C. [Today, is, Monday.]　　D. ['Today is', 'Monday.']

11. 执行语句 print( "Today is Monday.".partition('is '))的输出结果是（　　）。

A. [Today is Monday.]　　B. ('Today', 'is', 'Monday.')
C. (Today, is, Monday.)　　D. ['Today' , 'is', 'Monday.']

12. 执行语句 print( "Today is Monday.".replace('Monday','Tuesday'))的输出结果是（　　）。

A. [Today is Tuesday.]　　B. ('Today', 'is', 'Tuesday.')
C. (Today, is, Tuesday.)　　D. Today is Tuesday.

13. 执行语句 print( "Today is "+"a sunny"+" Monday")的输出结果是（　　）。

A. [Today is a sunny Monday.]　　B. ('Today', 'is', 'a', 'sunny', 'Monday.')
C. (Today, is, a, sunny , Monday.)　　D. Today is a sunny Monday.

14. 执行语句 print( "Today is a sunny day".startswith('Mon'))的输出结果是（　　）。

A. True　　B. False　　C. 0　　D. 1

15. 执行语句 print( "Today".rjust(7,'*'))的输出结果是（　　）。

A. **Today　　B. Today**　　C. 0　　D. 1

16. 执行语句 print( "**Today**".strip('*'))的输出结果是（　　）。

A. **Today　　B. Today**　　C. Today　　D. *Today*

17. 执行以下代码的输出结果是（　　）。

```
sentence= "Today is a sunny Monday"
words=sentence.split()
print("Sentence:\' %s\' \nLength of sentence: %s" % (sentence,
  len(words)))
```

A. Sentence:' Today is a sunny Monday'\n Length of sentence: 5
B. Sentence:\' %s\' \nLength of sentence: %s
C. Sentence:\' %s\' \nLength of sentence: %s % (sentence, len(words)
D. Sentence:' Today is a sunny Monday'
Length of sentence: 5

18. 在 Python 3 中 3/2 的结果是（　　）。

A. 1.5　　B. 1　　C. 报错　　D. 2

19. 下列变量名中错误的是（　　）。

A. name1　　B. name_1　　C. 1_name　　D. name_family

20. 下列可以用作变量名的是（　　）。

A. False　　B. None　　C. and　　D. test

21. 以下不是赋值运算符的是（　　）。

A. +=　　B. =　　C. *=　　D. ==

22. (3+ 6) ** 2 的运行结果是（　　）。

A. 30　　B. 81　　C. 20　　D. 90

23. 算术运算符//的含义是（　　）。

A. 两次除法　　B. 取整除运算　　C. 斜线运算　　D. 双斜线运算

24. x,y, z = 7, 13, "Python"，z 的值是（　　）。

A. 7　　B. 13　　C. "Python"　　D. 空

25. 下列代码执行的结果是（　　）。

```
a=1
b=2
print (a==b)
```

A. 1　　B. 0　　C. False　　D. True

26. 下列代码执行的结果是（　　）。

```
a=True
b=False
print (a and b)
```

A. 1　　B. 2　　C. False　　D. True

27. 下列代码执行的结果是（　　）。

```
a=1
b=2
print (a%b)
```

A. 1　　B. 0　　C. False　　D. True

28. 执行语句 print('-' .join('hello python'))的输出结果是（　　）。

A. -h-e-l-l-o- -p-y-t-h-o-n-　　B. h-e-l-l-o p-y-t-h-o-n

C. -h-e-l-l-o -p-y-t-h-o-n　　D. h-e-l-l-o- -p-y-t-h-o-n

29. 下列代码执行的结果是（　　）。

```
a=4
b=27
b%=a
a//=b
print ("%d,%d"%(a,b))
```

A. 1,3　　B. 3,1　　C. 0,3　　D. 3,0

30. 100-50*3%4 的运行结果是（　　）。

A. 2　　B. 98　　C. –2　　D. 0

31. 要将 5.2874 变成 00005.28 需要进行的格式化输出为（　　）。

A. "%.2f"%5.2874　　B. "%08.2f"% 5.2874

C. "%0.2f"%5.2874　　D. "%8.2f"%5.2874

32. 下列代码执行的结果是（　　）。

```
a=b=c=50
```

```
a=c%3*4
b=-2 and 2
print ("%d,%d"%(a,b))
```

A. 2,8　　B. 4,0　　C. 8,2　　D. 0,4

33. 下列那个语句在 Python 中是非法的？（　　）

A. a**=b　　B. a=b=c=6

C. a=(b+1=c)　　D. a,b=c,a

34. 下列代码的运行结果是（　　）。

```
a='a'
print(a>'b' or 'c')
```

A. a　　B. b　　C. c　　D. True

35. 下列字符串格式化语法正确的是（　　）。

A. 'Jenny's%d%' book'　　B. 'Jenny\'s%c%' book'

C. 'Jenny's%s%' book'　　D. 'Jenny\'s%s'%' book'

36. 判断 char 型变量 c 是否为小写字母的正确表达式是（　　）。

A. 'a'<=c'='z'　　B. (c>=A)&&(c<=z)

C. ('a'>=c)||('z'<=c)　　D. (c>='a')&&(c<='z')

## 二、多项选择题

1. 下列关于 Python 字符串界定方式说法正确的是（　　）。

A. 单引号界定的字符串不可以嵌套双引号字符串

B. 双引号界定的字符串可以嵌套单引号字符串

C. 三单引号或三双引号界定的字符串可以用于跨行表示字符串

D. Python 中各界定符只能英文输入状态下输入

2. 下列字符列中，用来表达转义字符的是（　　）。

A. \′　　B. \0　　C. \n　　D. \\′

3. 下列函数中能够实现字符串查找的有（　　）。

A. find()　　B. partition()　　C. .index()　　D. replace()

4. 下列属于格式字符的是（　　）。

A. %%　　B. %s　　C. %F　　D. %d

5. 下列对于函数的说明错误的有（　　）。

A. find()函数在查找不到指定子字符串时不返回值

B. index 如果查找不到指定子字符串时抛出异常

C. split()函数的返回值为分割后的字符串列表

D. islower()函数用于检测字符串是否含有小写字母

6. 下列运算结果输出正确的是（　　）。

A. a=80,b=20　a/b=4　　B. a=80.0,b=20　a/b=4.0

C. a=80,b=20.0　a//b=4.0　　D. a=80,b=20　a//b=4.0

7. 下列数据类型转换结果正确的是（　　）。

A. int('24',10)= 24　　B. int(False)= 0

C. bool(455)= False　　D. float(104)= 104.0

8. 下列表达式结果为 Ture 的有（　　）。

A. 'a'<'b'　　B. 'abc'<'z'　　C. 6>4>4　　D. 'ax'>'x'

9. 下列关于运算优先级比较的叙述正确的是（　　）。

A. 成员运算符>身份运算符

B. 指数运算符>比较运算符

C. 赋值运算符>成员运算符

D. 比较运算符>赋值运算符

10. 下列属于 Python 保留字的有（　　）。

A. None　　B. lambda　　C. for　　D. def

11. Python 支持的数据类型有（　　）。

A. Int　　B. float　　C. char　　D. list

12. 下列哪种说法错误的是（　　）。

A. 空字符串的布尔值是 True

B. 除字典类型外，所有标准对象均可以用于布尔测试

C. 值为 0 的任何数字对象的布尔值是 False

D. 空列表对象的布尔值是 False

13. 下列关于 Python 变量说法正确的是（　　）。

A. Python 中的变量不需要声明

B. Python 变量名在引用前必须先赋值

C. Python 允许同时为多个变量赋值

D. Python 中变量名称由字母、数字、下画线组成

## 二、判断题

1. 字符串属于不可变序列类型。（　　）
2. 字符串的不同界定符之间可以互相嵌套。（　　）
3. %%格式字符代表%。（　　）
4. 表达式'python.jpg'.endswith(( '.png', '.jpg')) 的值 False。（　　）
5. isspace()函数用于检测字符串是否含有空格。（　　）
6. 由于引号表示字符串的开始和结束，所以字符串本身不能包含引号。（　　）
7. 不同类型的数据不能相互运算。（　　）
8. 表达式'a city of USA'.istitle()的值为 True。（　　）
9. 表达式 'pppython'.lstrip('p')的值为'python'。（　　）
10. 运算符“//”的结果类型与分母分子的数据类型有关。（　　）
11. '%10.2f ' % pi 的含义是指定宽度为 10 指定精度为小数点后 2 位的 pi。（　　）
12. 'aaa/bbb'.split('/')得到的结果为 ['aaa','/', 'bbb']。（　　）
13. a= '456',b= '789',表达式 a+b 的值为'456789'。（　　）
14. bool(0)和 bool(None)的值都为 False。（　　）
15. a=4,b=6,a=+b，得到 a 的值为 10。（　　）
16. None 是一个特殊的空值，不能理解为 0。（　　）
17. 表达式 'appe' in ['apples'] 的值为 True。（　　）
18. 'Hello python'.swapcase().swapcase() 的值为'Hello python'。（　　）
19. 表达式 '%c%65==str(65)'的值为 False。（　　）

20. Python 变量使用前必须先声明，一旦声明就不能在当前作用域内改变其类型。（　　）

21. Python 变量名必须以字母或下画线开头，并且区分字母大小写。（　　）

22. 8888**8888 这样的命令在 Python 中无法运行。（　　）

23. 0o14f 在 Python 中是合法的八进制数字表示形式。（　　）

24. 0xbf 在 Python 中是合法的十六进制数字表示形式。（　　）

25. 在 Python 中可以使用 implement 做变量名。（　　）

26. Python 中 type()函数用来查看变量类型。（　　）

27. 表达式 int('101',2)的值为 6。（　　）

28. 假设 a 为整数，那么表达式 n&1==n%2 的值为 True。（　　）

29. 已知 a=3，b=5，在执行语句 a,b=b,a 后 a 的值为 3。（　　）

30. 表达式'Hellow python'.lower().upper()的值为'HELLOW PYTHON'。（　　）

## 四、计算题，写出计算过程

1. $(11001.11)_2=(?)_{10}=(?)_{16}=(?)_8$

2. $(205.5)_{16}=(?)_{10}=(?)_2=(?)_8$

3. $(215.75)_{10}=(?)_{16}=(?)_2=(?)_8$

## 五、思考题

写出判断变量 year 代表的年份是否为闰年的逻辑表达式。闰年是能被 4 整除但不能被 100 整除，或者能被 400 整除的年份。