2.2 数据类型

2023 年 9 月 8 日

数据类型

1. 关键字与标识符

（一）关键字

关键字（Keyword），是 Python 语言预设的具有特殊用途的专用词汇。Python 的常见关键字如下：

关键词					
and	del	from	not	while	as
global	or	with	assert	else	if
yield	break	except	import	print	class
in	raise	continue	finally	is	return

练习

使用 print(’’) 语句，输出 “Hello, World!”

[ ]:

（二）标识符

标识符（Identifier）是 Python 中用于变量、常量、函数、语句块等命名的一串字符。

命名标识符须遵循以下规则：1. 标识符由字母、数字、下划线 “_” 或美元符号 “$” 等组合构成；2. 标识符的首个字符不能是数字；3. 不能使用关键字；4. 标识符大小写敏感

例如 gduf2020、_gduf2020 都是合法标识符，而 2020gduf 不是标识符。由于 Python 严格区分字符的大小写，所以 Pi 和 pi 是两个不同的标识符。
为了提高程序的可读性，标识符命名要尽可能 “见其名而知其意”。

练习

请设计一个变量的名称，用来表示学生的姓名、学生的年龄、学生的专业

[ ]:

2. 变量与赋值

变量（Variable）是计算机语言中能记忆数据的抽象概念。程序通过变量名（标识符表示）来访问变量。
在程序执行过程中，一个变量名代表其记忆的数据所存储的内存区域。Python 程序通过变量名使用该变量指代的内存区域，从该内存区域读取数据或将某项数据保存到该区域中。
例如 anInteger=100 语句表示将整数 100 保存到变量 anInteger 所指代的内存区域。

[1]:
```
anInteger = 100 # 一个整数型数据
aFloat = 999.99 # 一个浮点型数据
aString = 'Hello, world!' # 一个字符串
```

[2]:
```
print(anInteger)
print(aFloat)
print(aString)
print('anInteger=', anInteger, ',aFloat=', aFloat, ',aString=', aString)
print('anInteger= %s, aFloat= %s, aString= %s' %(anInteger, aFloat, aString))
```

```
100
999.99
Hello, world!
anInteger= 100 ,aFloat= 999.99 ,aString= Hello, world!
anInteger= 100, aFloat= 999.99, aString= Hello, world!
```

以 # 开头的语句是注释，注释是给人看的，可以是任意内容，解释器会忽略掉注释。

练习

1. 定义一个变量表示学生的姓名，并赋值给它”John”
2. 定义一个变量表示学生的年龄，并赋值给它”21”
3. 定义一个变量表示学生的专业，并赋值给它”Math”
4. 使用 print 语句打印上面的三个变量

[ ]:

3. 常用数据类型

计算机顾名思义就是可以做数学计算的机器，因此，计算机程序理所当然地可以处理各种数值。

但是，计算机能处理的远不止数值，还可以处理文本、图形、音频、视频、网页等各种各样的数据，不同的数据，需要定义不同的数据类型。在 Python 中，能够直接处理的数据类型有以下几种：

（一）整数

Python 可以处理任意大小的整数，当然包括负整数，在程序中的表示方法和数学上的写法一模一样，例如：`1`，`100`，`-8080`，`0`，等等。

计算机由于使用二进制，所以，有时候用十六进制表示整数比较方便，十六进制用 `0x` 前缀和 0-9，a-f 表示，例如：`0xff00`，`0xa5b4c3d2`，等等。

对于很大的数，例如 `10000000000`，很难数清楚 0 的个数。Python 允许在数字中间以 `_` 分隔，因此，写成 `10_000_000_000` 和 `10000000000` 是完全一样的。十六进制数也可以写成 `0xa1b2_c3d4`。

（二）浮点数

浮点数也就是小数，之所以称为浮点数，是因为按照科学记数法表示时，一个浮点数的小数点位置是可变的，比如，1.23×10^9 和 12.3×10^8 是完全相等的。

浮点数可以用数学写法，如 `1.23`，`3.14`，`-9.01`，等等。但是对于很大或很小的浮点数，就必须用科学计数法表示，把 10 用 e 替代，1.23×10^9 就是 `1.23e9`，或者 `12.3e8`，0.000012 可以写成 `1.2e-5`，等等。

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

（三）字符串

字符串是以单引号`'` 或双引号`"` 括起来的任意文本，比如`'abc'`，`"xyz"` 等等。请注意，`''` 或`""` 本身只是一种表示方式，不是字符串的一部分，因此，字符串`'abc'` 只有 `a`，`b`，`c` 这 3 个字符。如果`'` 本身也是一个字符，那就可以用`""` 括起来，比如`"I'm OK"` 包含的字符是 `I`，`'`，`m`，空格，`O`，`K` 这 6 个字符。

如果需要拼接字符串，可以使用 `+` 号，例如拼接'abc' 和'xyz' 这两个字符串：

[3]:
```
'abc' + 'xyz'
```

[3]:
```
'abcxyz'
```

练习

请拼接如下三个字符串：‘123’, ‘456’, ’789’ 为一个字符串。

[ ]:

（四）列表

列表（List）是 python 语言编程使用最多的一种数据结构。
它是一种**有序**的集合，可以随时添加和删除其中的元素。

[4]:
```
l = [6, 3.2, '苹果']
```

列表是一个可变的有序表，所以，可以往列表中追加元素到末尾：

[5]:
```
l.append('香蕉')
l
```

[5]:
```
[6, 3.2, ' 苹果', ' 香蕉']
```

要删除指定位置的元素，用 `pop(i)` 方法，其中 `i` 是索引位置：

[6]:
```
l.pop(1)
l
```

[6]:
```
[6, ' 苹果', ' 香蕉']
```

要把某个元素替换成别的元素，可以直接赋值给对应的索引位置：

[7]:
```
l[1] = 1.9
l
```

[7]:
```
[6, 1.9, ' 香蕉']
```

列表中的元素的索引位置是从 0 开始，直到列表的长度-1 结束。

列表元素也可以是另一个列表，比如：

[8]:
```
l = [6, 1.9, ['苹果', '香蕉']]  # 这种称之为嵌套
len(l)
```

[8]:
```
3
```

如果一个列表中一个元素也没有，就是一个空的列表，它的长度为 0：

[9]:
```
l = []
len(l)
```

[9]: 0

列表的常用方法

列表支持以下方法：

语法	描述
L.append(x)	将元素 x 追加到列表 L 的尾部
L.count(x)	返回元素 x 在列表 L 中出现的次数
L.extend(m)	将列表 m 的项追加到 L 的结尾处，操作符 += 完成同样的功能
L.index(x, start, end)	返回元素 x 在列表 L 中 (或 L 的 start:end 分片中) 最左边出现的索引位置，否则会产生一个 ValueError 异常
L.insert(i, x)	在索引位置 i 处将元素 x 插入列表 L
L.pop()	返回并移除 list L 最右边的元素
L.pop(i)	返回并移除 L 中索引位置 int i 处的元素
L.remove(x)	从 list L 中移除最左边出现的元素 x，如果找不到 x 就产生 ValueError 异常

练习

1. 创建一个列表包含元素：‘100’ 和’hoffman’, 然后在列表后面追加一个新元素 70。
2. 计算上述列表的长度

[ ]:
```

```

（五）元祖

另一种有序列表叫元组：tuple。

> 这个单词怎么读呢？Python 之父吉多 · 范罗苏姆在 Twitter 上说：“每周的一三五我会把 tuple 念作 **too-pull**，而二四六我喜欢念作 **tub-pull**。至于礼拜天嘛，我从不会讨论这些。:)”

引自《Python 语言及其应用》

元祖和列表非常类似，但是 tuple 一旦初始化就不能修改，比如：

```
>>> t = (6, 1.9,  ' 香蕉')
>>> type(t) # 看看它的类型
```

现在，这个元祖不能变了，它也没有 append()，insert() 这样的方法。其他获取元素的方法和列表是一样的，你可以正常地使用 `t[0]`，`train[-1]`，但不能赋值成另外的元素。通常我们使用元祖来存储一些无法被修改的安全信息。

练习

创建一个元祖变量名称为 account_info, 其中包含两个元素'John', '123456',
然后尝试令 account_info[1] = '654321'。

[ ]:

（六）字典

我们要学习的第三个数据结构是字典（dict）全称 dictionary。它是 Python 中唯一的映射类型的数据结构。

举个例子，假设要根据同学的名字查找对应的成绩，如果用列表实现，需要两个列表：

```
names = ['Michael', 'Bob', 'Tracy']
scores = [95, 75, 85]
```

给定一个名字，要查找对应的成绩，就先要在 names 中找到对应的位置，再从 scores 取出对应的成绩，列表越长，耗时越长。

如果用字典实现，只需要一个 "名字"-"成绩" 的对照表，直接根据名字查找成绩，无论这个表有多大，查找速度都不会变慢。用 Python 写一个字典如下：

```
>>> d = {'Michael': 95, 'Bob': 75, 'Tracy': 85}
>>> d['Michael']
95
```

请务必注意，字典内部存放的顺序和键放入的顺序是没有关系的。

和列表比较，字典有以下几个特点：

1. 查找和插入的速度极快，不会随着 key 的增加而变慢；
2. 需要占用大量的内存，内存浪费多。

而列表相反：

1. 查找和插入的时间随着元素的增加而增加；
2. 占用空间小，浪费内存很少。

所以，字典是用空间来换取时间的一种方法。

字典可以用在需要高速查找的很多地方，在 Python 代码中几乎无处不在，正确使用字典非常重要，需要牢记的第一条就是字典的 key 必须是**不可变对象**。

字典有如下方法：

语法	描述
d.clear()	从字典 d 中移除所有项
d.copy()	返回字典 d 的浅拷贝
d.fromkeys(s, v)	返回一个 dict，该字典的键为序列 s 中的项，值为 None 或 v(如果给定了参数 v)
d.get(k)	返回键 k 相关联的值，如果 k 不在字典 d 中就返回 None
d.get(k, v)	返回键 k 相关联的值，如果 k 不在字典 d 中就返回 v
d.items()	返回字典 d 中所有 (key, value) 对的视图
d.keys()	返回字典 d 中所有键的视图
d.pop(k)	返回键 k 相关联的值，并移除键为 k 的项，如果 k 不包含在 d 中就产生 KeyError 异常
d.pop(k, v)	返回键 k 相关联的值，并移除键为 k 的项，如果 k 不包含在 d 中就返回 v
d.popitem()	返回并移除字典 d 中一个任意的 (key, value) 对，如果 d 为空就产生 KeyError 异常
d.setdefault(k, v)	与 dict.get() 方法一样，不同之处在于，如果 k 没有包含在字典 d 中就插入一个键为 k 的新项，其值为 None 或 v(如果给定了参数 v)
d.update(a)	将 a 中每个尚未包含在字典 d 中的 (key, value) 对添加到 d，对同时包含在 d 与 a 中的每个键，使用 a 中对应的值替换 d 中对应的值——a 可以是字典，也可以是 (key, value) 对的一个 iterable，或关键字参数
d.values()	返回字典 d 中所有值的视图 [*]

练习

创建一个字典变量，名为 student_info, 包含三个键-值对：

- 'name': 'John'
- 'age': 21
- 'major':'math'

[ ]:

4. 运算符与表达式

运算符包括算术运算符、关系运算符、逻辑运算符、赋值运算符欧等等。
我们先来看看逻辑运算符：and、or 和 not，逻辑运算符的操作数是逻辑值 True（真）和 False（假）。not 属于一元运算符，只有一个操作数；or 和 and 属于二元运算符，有两个操作数。

表逻辑运算符的运算法则

操作数 X	操作数 Y	X and Y	X or Y	not X
True	True	True	True	False
True	False	False	True	False
False	True	False	True	True
False	False	False	False	True

表达式是由变量、对象、方法调用和操作符等元素组成的式子。例如 `a+b`, `a>b`, `a and b` 都为表达式。

当表达式出现多个运算符时，Python 将按照运算符的优先次序从高到低进行计算。

运算符的优先次序

运算符	描述（优先次序从高到低）
**	指数
~, + , -	按位翻转，正号，负号
*, /, %, //	乘、除、取模和取整除
+, -	加法、减法
», «	右移

运算符	描述（优先次序从高到低）
&,	位与
^, |	位运算符
<,<=,>,>=,!=,==	比较运算符
is, is not	身份运算符
in, not in	成员运算符
not, or, and	逻辑运算符

练习

以下表达式的运行结果是多少?

- 1 + 2**2
- 2>1 and 3<2
- not 2>1
- 2 == 2
- 3 != 2
- a = True; (a is False) or (a is not True) or (a is not False)
- a = [1,2,3,4,5,6]; not (3 not in a)
- a=[2+2*4,3+2*1,4+1**5,4]; ((not(5 not in a) and (10 in a)) is False) == (not(5 not in a) and (10 in a) is False)

可以粘贴并运行。

[ ]: