内置数据类型包括哪几种?

目录

内置类型汇总

- 内置类型
  - 逻辑值检测
  - 布尔运算 --- and, or, not
  - 比较运算
  - 数字类型 --- int, float, complex
  - 迭代器类型
  - 序列类型 --- list, tuple, range
  - 文本序列类型 --- str
  - 二进制序列类型 --- bytes, bytearray, memoryview
  - 集合类型 --- set, frozenset
  - 映射类型 --- dict
  - 上下文管理器类型
  - 类型注解的类型 --- Generic Alias 、 Union
  - 其他内置类型
  - 特殊属性

https://docs.python.org/zh-cn/3.10/library/index.html

常见数据类型

数字类型：int、float、complex

文本类型：str

序列类型：list、tuple、range

int, int, int, int…

int, str, int, int…

映射类型：dict

int:str, int:str, int:str…

int:list, int:list, int:list…

常见数据类型的主要用途

从学过的 int、str 来思考数据类型的用途

- 内置常用函数
- 运算符复用
- 语义明确，不容易出错

常见数据类型的主要用途

- 自己编写类型需要更丰富的编程经验
- 需要通用、高效、功能齐全的其他数据类型

常见数据类型的主要用途

哪类问题比较普遍?

从非结构化数据到结构化数据——列表、元组

从非结构化数据到半结构化数据——字典

总结

1. 常用的数据类型包括数字、字符串、序列、映射
2. 序列中列表和元组最常用，映射中字典最常用
3. 从非结构化数据到半结构化数据再到结构化数据，满足日常的数据处理需求

课后作业

请你根据以下需求，选择一个合适的数据类型：

1. 开发通讯录工具，需要存储姓名和手机号码；

2. 开发电商程序，需要存储商品的种类；

3. 开发游戏，需要存储一段与 NPC 的对话。

列表：如何处理同类数据?

目录

列表的定义

- 列表中可以存放一系列对象
- 存放的对象可以是数字、字符串也可以是列表
- 列表用方括号“[]”表示，每个对象称作列表的元素，元素之间使用逗号“,”分隔
- 列表中的元素可以在定义列表之后进行修改，也可以使用索引来访问一个或多个元素
- 列表也支持了丰富的内置函数

列表的定义

[1, 2, 3, 4, 5]

['aa', 'cc', 'dd']

[123, 'abc']

[[1, 'a'], [2, 'b']]

以上列表的定义都是合法的

创建列表

```
colours = [ "red", "blue", "green" ]

print( colours )  # 将列表打印出来

print( type(colours) ) # 打印 colours 变量的类型
```

创建列表

使用内置函数 list() 也可以将字符串创建为列表，如：

```
>>> list('red')

['r', 'e', 'd']
```

创建列表

使用列表推导式创建列表，如：

```
>>> [ x for x in range(1, 10)]

[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

* 本内容在循环章节详细讲解

访问列表的元素

列表是有序的数据类型，可以通过索引访问到列表的每一个元素：

```
list1 = [ 'a', 'b', 'c', 'd' ]

list1[0]   # a

list1[1]   # b

list1[–1]   # d
```

访问列表的元素

```
list1 = [ 'a', 'b', [ 'c', 'd' ] ]

list1[2] [0]    # c
```

如果我想得到字符串 'd'， 要怎么使用索引呢?

删除列表的元素和删除列表

删除列表中的元素

```
del  list1[0]
```

删除列表

```
del  list1
```

总结

1. 列表用于存放有序的并列数据
2. 列表创建和删除方法
3. 列表中的元素需要使用索引来访问

课后作业

如果我想创建如下列表，需要如何成功创建?

[[1, 2, 3],

  [4, 5, 6],

  [7, 8, 9]]

常见常新：列表常见操作

目录

添加元素

list.insert(索引, 元素) 在索引位置插入元素

list.append(元素) 在列表结尾添加元素

list.extend(可迭代对象*) 为列表扩展元素

*append 和 extend 参数为列表时，增加的结果不同

修改元素

list.remove(元素) 移除列表的元素

list.reverse() 反转列表元素的顺序

list.pop(索引) 移除索引对应的元素并返回该元素，不指定索引移除最后一个元素

list.copy() 复制列表

list.clear() 清空列表

计算列表长度

len(list) 得到列表的长度

len(list[0]) 得到列表中元素的长度

list.count(元素) 元素出现的次数

列表排序

list.sort(reverse=True) 列表原地排序

sort(list) 列表排序后返回新的列表

总结

1. 列表常见操作包括添加、修改、统计、排序
2. 注意列表内置函数的参数
3. 注意列表的函数是否会原地修改

课后作业

['a', 1, 'b', 2, 'c', 'c', 3]

请将上面列表中重复的元素 'c' 移除，并在命令行输出。

元组：如何处理一次性数据?

目录

序列和元组

基本序列包括列表、元组、range

- 元组和列表类似，但是差别是创建后不可修改
- 二进制数据和文本字符串属于特别定制的附加序列

序列和元组

序列的通用操作

运算	结果
x in s	如果 s 中的某项等于 x 则结果为 True，否则为 False
x not in s	如果 s 中的某项等于 x 则结果为 False，否则为 True
s + t	s 与 t 相拼接
s * n 或 n * s	相当于 s 与自身进行 n 次拼接
s[i]	s 的第 i 项，起始为 0
s[i:j]	s 从 i 到 j 的切片
s[i:j:k]	s 从 i 到 j 步长为 k 的切片
len(s)	s 的长度
min(s)	s 的最小项
max(s)	s 的最大项
s.index(x[, i[, j]])	x 在 s 中首次出现项的索引号（索引号在 i 或其后且在 j 之前）
s.count(x)	x 在 s 中出现的总次数

序列和元组

序列分为：可变序列和不可变序列

- 列表属于可变序列
- 元组、字符串属于不可变序列
- 不可变序列中， append()、pop()、insert() 等修改序列元素的函数均无法使用

元组和列表的区别

1. 元组和列表创建之后，元组不可修改，列表可修改

2. 元组执行效率高，列表执行效率低

创建元组

- 可以使用圆括号“()”定义元组
- 可以使用 tuple() 函数创建元组
- 将 range()、列表、字符串转换为元组

删除元组

使用 del 可以删除元组

课后作业

请你创建元组，元组中包含三个元素，分别为：'x'、'y'、'z'。

请为该元组执行增加元素和删除元素操作，执行访问第四个元素索引操作，

执行后请分别观察出错信息，并区分不同错误产生的不同提示信息内容。

常见常新：集合的常见操作

目录

集合与 set 对象

集合类型包括 set 和 frozenset 两种对象

- set 对象可变，frozenset 对象不可变
- set 对象在程序设计中较常使用

创建 set 对象

集合可以用多种方式创建：

1. 使用花括号内，以逗号分隔元素的方式 { 'wilson' , 'yin' }

2. 使用集合推导式

3. 使用类型构造器 set()

set 对象的常用操作

len(s)	返回集合 s 的元素数量
x in s	检测 x 是否为 s 中的成员
set <= other	检测集合 set 中的每个元素是否在 other 中
set < other	检测集合 set 是否为 other 的真子集

其他内置函数： add()、remove()、pop()、clear() 都能实现对集合的原地修改

删除 set 对象

- 使用 del 可以删除 set 对象

总结

1. 集合的唯一性是经常使用该数据类型的主要原因

2. 从序列中去除重复项等操作，可以通过数据类型强制转换实现

课后作业

给定列表 list1 = ['r', 'g', 'b', 'g', 'b', 'r', 'g'],

请使用集合删除列表中重复的元素，并将其转换为元组在命令行进行输出。

字典：如何处理映射类型的数据?

目录

映射与字典

正整数模 7 运算后，得到了 0–6 的空间

映射与字典

- 映射是可变类型
- 映射只有一种数据类型，即：字典
- 字典里的一个元素由键和值两部分组成
- 键不能重复，因此无法哈希的类型，如列表、字典等，不可作为键来使用
- 整数 1 和浮点数 1.0 会被当作相同的键

字典的定义

定义一个新字典：

1. 使用花括号定义：{ 'one': 1, 'two' : 2 }

2. 使用类型构造器：dict(one=1, two=2)

3. 使用字典推导式：{ x: x**2 for x in range(10) }

字典的删除

- 使用 del 可以删除字典
- 使用 del 字典[键] 可以删除字典中的指定元素

总结

1. 字典用于存储键值对，键值对之间有关联
2. 字典的键要求可哈希，一般采用字符串、元组做字典的键
3. 可以使用 dict() 函数、推导式和花括号{} 三种方式实现字典的创建

课后作业

已知有两个列表，分别为:

['name1', 'name2', 'name3']

['1111', '2222', '3333']

现需要将这两个列表组成一个如下字典，请编写程序实现:

{ 'name1':'1111', 'name2':'2222', 'name3':'3333' }

常见常新：字典的常见操作

目录

字典的内置函数

- **访问字典的内容：**

```
>>> mail_list
{'tom': 'tom@gmail.com', 'jerry': 'jerry@foxmail.com', 'john': 'john@163.com'}
>>> mail_list.items()
```

- **访问字典里指定的键：**

```
>>> mail_list.get('tom')
'tom@gmail.com'
>>> mail_list['tom']
‘tom@gmail.com'
```

字典的内置函数

- **遍历字典**

```
>>> for key, value in mail_list.items():
...     print(key)
...     print(value)
...
tom
tom@gmail.com
jerry
jerry@foxmail.com
john
john@163.com
```

字典的内置函数

- 修改字典的内容：

  为字典添加新的键值对：mail_list['wilson'] = 'wilson@163.com'

  如果字典中已经包含键 'wilson'，则更新该键对应的值

字典的内置函数

- 返回字典的项数

  len(字典)

- 判断键是否在字典中

  key in 字典

- 如果键存在且在字典中，移除该键，返回该键对应的值

  pop()

- 如果键存在且在字典中，移除该键，返回键值对

  popitem()

字典的高级用法

- 如果字典存在键 key，返回它对应的值
- 如果不存在，插入 default 的键 key，并返回 default

字典.setdefault(key[, default])

字典的高级用法

- 用新字典的“键”和“值”更新旧的字典，新字典的“键”和“值”优先

```
mail_list |= new_mail_list
```

* |= 需 Python3.9 及以上版本支持

字典与其他数据类型的混合使用

- 数据类型间的转换

  列表 < — > 字典 < — > 字符串

  list()　　dict()　　str()

字典与其他数据类型的混合使用

- 利用 zip() 函数合并两个列表为字典

  字典 = dict(zip(列表1, 列表2))

字典与其他数据类型的混合使用

含有多个字典的列表

```
{

'red':[255,0,0],

'green':[0,255,0],

'blue':[0,0,255],

}
```

```
[

{ 'color':'red', 'value':[255,0,0] },

{ 'color':'green', 'value':[0,255,0] },

{ 'color':'blue', 'value':[0,0,255] }

]
```

总结

1. 字典能支持丰富的内部函数，方便对字典进行查询和修改

2. 字典和列表、字符串经常混合使用，表示更复杂的数据类型

课后作业

请你设计一个字典数据类型用于存储通讯录，通讯录中包含：

必须填写的姓名、可以为空的备注名、1 个邮箱、至少 2 个手机号码，

并尝试增加和删除联系人。

小试牛刀：如何利用类型转换实现手机通讯录?

目录

数据处理的思路

- 数据需要汇总才能体现出更多的价值
- 大部分数据无法直接处理
- 数据需要去除噪声
- 数据需要从非结构化转换为半结构化或者结构化数据

需求分析

数据分析需求：

1. 统计文件行数

2. 统计不包含空行的行数

3. 统计单词 I 出现的次数

4. 统计单词 you 和 You 出现的次数

需求分析

样例文章的结构如下：

```
[
'... ... \n',
'\n',
'... ... \n',
'\n'
]
```

使用类型转换进行数据的临时存储

- 需要统计个数时，要处理好分隔符
- 需要计算时，要转换为数字

使用类型转换进行通讯录的存储

- 需要将以“,”分隔的字段读取并转换为字典
- 需要支持查询功能

总结

1. 数据处理就是将非结构化数据转换为结构化数据的过程
2. 利用不同的数据类型可以方便对文章进行数据分析
3. 编程除了结果正确外，可读性也是好的程序标准，可以为复杂的程序适当增加变量

课后作业

请你基于本讲的代码，统计样例文件中每个单词出现的次数，并将每个单词和出现次数存入字典中。形式如下：

```
{ "I":5,
  "You":3,
  …
}
```

常见的内置数据类型都何时使用?

目录

常见数据类型有哪些

常量：None

逻辑值：True、False

空集：''、()、[]、{}、set()、range(0)

数字：int、float

序列：list、tuple、range

文本序列：str

集合：set

映射：dict

常见数据类型的主要用途

以下这些类型用来表示“假”值：

1. 常量 None

2. 布尔值的 False

3. 数值类型的 0、0.0

4. 空的序列、映射

常见数据类型的主要用途

不同类型的比较都可以使用 >、<、== ， 但是比较的规则不同

1. 数字之间按照数值大小比较

2. 字符串之间按照首字母 ASCII 编码大小比较

3. 列表、元组从第一个元素开始比较

常见数据类型的主要用途

- 列表、元组、集合、字典可以和字符串、数字组成复杂的数据类型
- 当你有多种组合方法时，要多考虑每种类型的内置方法

常见数据类型的主要用途

利用不同类型的优点，可以有效减少编程的难度，如：

- 集合可以自动去重复，用于避免重复元素最便捷
- 元组不可变，利用元组可以避免误修改

总结

1. 掌握不同数据类型的用途可以帮你有效降低代码复杂度
2. 利用不同数据类型的优势，可以减少编程的难度
3. 数据类型之间的组合可以实现更复杂的结构

课后作业

请根据本讲学习的数据类型，回答以下问题：

1. 当你需要存储不重复的数据时，可以采用哪种数据类型?

2. 当你既需要存储数字，又要存储字符时，可以采用那种数据类型?

3. 当你需要存储汉字时，可以采用哪种数据类型?

避坑指南：内置数据类型的常见错误

目录

访问错误

使用内置方法，访问到某个类型不存在的元素

- 列表：IndexError: list index out of range
- 字典： KeyError: 'xxx'

访问错误

解决办法：

先判断该元素是否存在，再进行操作

元素 in 对象

不同数据类型之间操作报错

- 不同的数据类型之间，能够支持不同功能但相同写法的运算符，如：

>、<、==、+、−

- 不同类型之间操作会出现报错，如：

```
>>> 1 + 'a'

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

不同数据类型之间操作报错

解决办法：

转换为相同的类型

对只读类型进行写入报错

这种错误经常出现在元组类型中，如：

```
>>> t.append('new_element')

AttributeError: 'tuple' object has no attribute 'append'
```

对只读类型进行写入报错

解决办法：

写入前，先判断好类型

type(t)

引用错误

常见于用不可哈希对象作为字典的键

典型报错：TypeError: unhashable type: 'list'

引用错误

解决办法：

使用元组、字符串、数字作为字典的键

总结

1. 数据类型产生的错误主要源于对变量类型的理解不清晰
2. 数据类型错误可以避免
3. 可以在程序逻辑增加判断语句，避免错误

课后作业

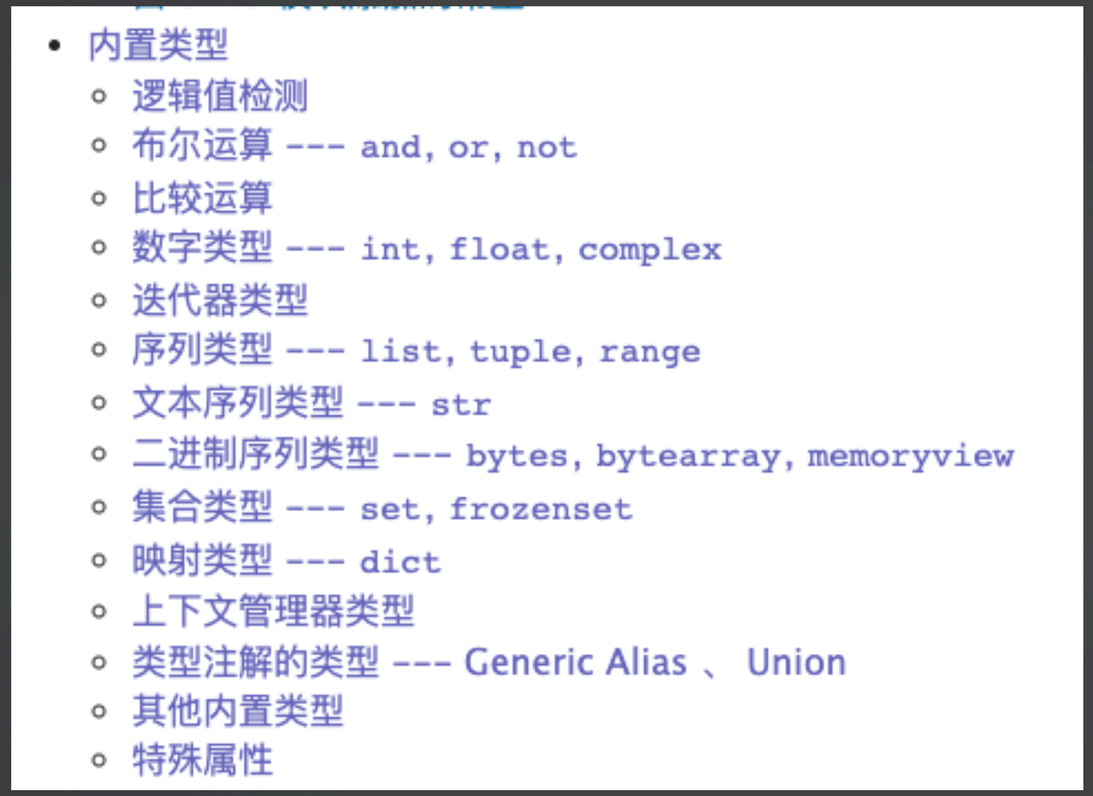
定义一个列表 list1 和元组 tuple1，当 tuple1 作为字典 dict1 的键时，是否能够成功定义字典？请说明原因。执行后，观察执行结果。

```
list1 = [ 1, 2, 3]

tuple1 = ( 'abc', list1 )

print( type( tuple1 ))

dict1 = { tuple1, 1 }   # 此语句是否能够正确执行?
```

内置数据类型参考：如何使用官方文档与帮助?

目录

使用官方文档



- 内置类型
  - 逻辑值检测
  - 布尔运算 --- and, or, not
  - 比较运算
  - 数字类型 --- int, float, complex
  - 迭代器类型
  - 序列类型 --- list, tuple, range
  - 文本序列类型 --- str
  - 二进制序列类型 --- bytes, bytearray, memoryview
  - 集合类型 --- set, frozenset
  - 映射类型 --- dict
  - 上下文管理器类型
  - 类型注解的类型 --- Generic Alias 、 Union
  - 其他内置类型
  - 特殊属性

https://docs.python.org/zh-cn/3.10/library/stdtypes.html

使用官方文档

`str.count(sub[, start[, end]])`

返回子字符串 *sub* 在 [*start*, *end*] 范围内非重叠出现的次数。 可选参数 *start* 与 *end* 会被解读为切片表示法。

str → 对象

count() → 对象的方法

sub → 参数（必须）

[start] → 参数（可选）

使用帮助

IDE 自带帮助，可以帮你补全命令

```
>>> list.
list.append(  list.count(   list.insert(  list.remove(
list.clear(   list.extend(  list.mro(     list.reverse(
list.copy(    list.index(   list.pop(     list.sort(
```

使用帮助

help（对象）

```
class list(object)
 |  list(iterable=(), /)
 |
 |  Built-in mutable sequence.
 |
 |  If no argument is given, the constructor creates a new empty list.
 |  The argument must be an iterable if specified.
 |
 |  Methods defined here:
 |
 |  __add__(self, value, /)
 |      Return self+value.
 |
 |  __contains__(self, key, /)
 |      Return key in self.
 |
 |  __delitem__(self, key, /)
 |      Delete self[key].
```

```
 |
 |  append(self, object, /)
 |      Append object to the end of the list.
 |
 |  clear(self, /)
 |      Remove all items from list.
 |
 |  copy(self, /)
 |      Return a shallow copy of the list.
 |
 |  count(self, value, /)
 |      Return number of occurrences of value.
 |
 |  extend(self, iterable, /)
 |      Extend list by appending elements from the iterable.
 |
```

总结

1. 官方文档是体系化的帮助文档，help() 是提醒式的帮助文档
2. 使用一门编程语言前应该提前阅读官方文档
3. 读懂文档中的表达方式非常有必要

课后作业

在学习 Python 的过程中，发现一条以前没有学习过的语句：

it = iter([1, 2, 3, 4])

请结合官方文档，判断 it 对象是否为迭代器类型。

以下为官方文档链接：

https://docs.python.org/zh-cn/3.10/library/stdtypes.html#iterator-types

THANKS