1. 说明列表的浅拷贝和深拷贝的区别

  浅拷贝可以使用列表自带的copy()函数（如list.copy()），或者使用copy模块的copy()函数。深拷贝只能使用copy模块的deepcopy(),所以使用前要导入：from copy import deepcopy

  如果拷贝的对象里的元素只有值，没有引用，那浅拷贝和深拷贝没有差别，都会将原有对象复制一份，产生一个新对象，对新对象里的值进行修改不会影响原有对象，新对象和原对象完全分离开。

  如果拷贝的对象里的元素包含引用（像一个列表里储存着另一个列表，存的就是另一个列表的引用），那浅拷贝和深拷贝是不同的，浅拷贝虽然将原有对象复制一份，但是依然保存的是引用，所以对新对象里的引用里的值进行修改，依然会改变原对象里的列表的值，新对象和原对象完全分离开并没有完全分离开。而深拷贝则不同，它会将原对象里的引用也新创建一个，即新建一个列表，然后放的是新列表的引用，这样就可以将新对象和原对象完全分离开。

1. 说明列表和元组的相同点和不同点

列表和元组是Python中最常用的两种数据结构，字典是第三种。 相同点：

* 都是序列
* 都可以存储任何数据类型
* 可以通过索引访问

**语法差异**

使用方括号[]创建列表，而使用括号()创建元组。 请看以下示例：

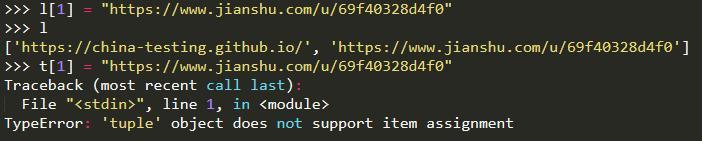


**是否可变**

列表是可变的，而元组是不可变的，这标志着两者之间的关键差异。

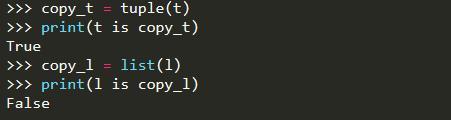
我们可以修改列表的值，但是不修改元组的值。

由于列表是可变的，我们不能将列表用作字典中的key。 但可以使用元组作为字典key。



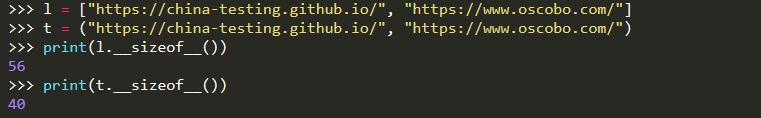
**重用与拷贝**

元组无法复制。 原因是元组是不可变的。 如果运行tuple(tuple\_name)将返回自己。



**大小差异**

Python将低开销的较大的块分配给元组，因为它们是不可变的。 对于列表则分配小内存块。 与列表相比，元组的内存更小。 当你拥有大量元素时，元组比列表快。列表的长度是可变的。



**同构与异构**

习惯上元组多用于用于存储异构元素，异构元素即不同数据类型的元素，比如(ip,port)。 另一方面，列表用于存储异构元素，这些元素属于相同类型的元素，比如[int1,in2,in3]。

1. 请写出字符串支持的所有方法及说明（例如: lower 返回字符串的小写）

|  |  |
| --- | --- |
| **方法** | **描述** |
| [string.capitalize()](https://www.runoob.com/python/att-string-capitalize.html) | 把字符串的第一个字符大写 |
| [string.center(width)](https://www.runoob.com/python/att-string-center.html) | 返回一个原字符串居中,并使用空格填充至长度 width 的新字符串 |
| [**string.count(str, beg=0, end=len(string))**](https://www.runoob.com/python/att-string-count.html) | 返回 str 在 string 里面出现的次数，如果 beg 或者 end 指定则返回指定范围内 str 出现的次数 |
| [string.decode(encoding='UTF-8', errors='strict')](https://www.runoob.com/python/att-string-decode.html) | 以 encoding 指定的编码格式解码 string，如果出错默认报一个 ValueError 的 异 常 ， 除非 errors 指 定 的 是 'ignore' 或 者'replace' |
| [string.encode(encoding='UTF-8', errors='strict')](https://www.runoob.com/python/att-string-encode.html) | 以 encoding 指定的编码格式编码 string，如果出错默认报一个ValueError 的异常，除非 errors 指定的是'ignore'或者'replace' |
| [**string.endswith(obj, beg=0, end=len(string))**](https://www.runoob.com/python/att-string-endswith.html) | 检查字符串是否以 obj 结束，如果beg 或者 end 指定则检查指定的范围内是否以 obj 结束，如果是，返回 True,否则返回 False. |
| [string.expandtabs(tabsize=8)](https://www.runoob.com/python/att-string-expandtabs.html) | 把字符串 string 中的 tab 符号转为空格，tab 符号默认的空格数是 8。 |
| [**string.find(str, beg=0, end=len(string))**](https://www.runoob.com/python/att-string-find.html) | 检测 str 是否包含在 string 中，如果 beg 和 end 指定范围，则检查是否包含在指定范围内，如果是返回开始的索引值，否则返回-1 |
| [**string.format()**](https://www.runoob.com/python/att-string-format.html) | 格式化字符串 |
| [**string.index(str, beg=0, end=len(string))**](https://www.runoob.com/python/att-string-index.html) | 跟find()方法一样，只不过如果str不在 string中会报一个异常. |
| [string.isalnum()](https://www.runoob.com/python/att-string-isalnum.html) | 如果 string 至少有一个字符并且所有字符都是字母或数字则返  回 True,否则返回 False |
| [string.isalpha()](https://www.runoob.com/python/att-string-isalpha.html) | 如果 string 至少有一个字符并且所有字符都是字母则返回 True,  否则返回 False |
| [string.isdecimal()](https://www.runoob.com/python/att-string-isdecimal.html) | 如果 string 只包含十进制数字则返回 True 否则返回 False. |
| [string.isdigit()](https://www.runoob.com/python/att-string-isdigit.html) | 如果 string 只包含数字则返回 True 否则返回 False. |
| [string.islower()](https://www.runoob.com/python/att-string-islower.html) | 如果 string 中包含至少一个区分大小写的字符，并且所有这些(区分大小写的)字符都是小写，则返回 True，否则返回 False |
| [string.isnumeric()](https://www.runoob.com/python/att-string-isnumeric.html) | 如果 string 中只包含数字字符，则返回 True，否则返回 False |
| [string.isspace()](https://www.runoob.com/python/att-string-isspace.html) | 如果 string 中只包含空格，则返回 True，否则返回 False. |
| [string.istitle()](https://www.runoob.com/python/att-string-istitle.html) | 如果 string 是标题化的(见 title())则返回 True，否则返回 False |
| [string.isupper()](https://www.runoob.com/python/att-string-isupper.html) | 如果 string 中包含至少一个区分大小写的字符，并且所有这些(区分大小写的)字符都是大写，则返回 True，否则返回 False |
| [**string.join(seq)**](https://www.runoob.com/python/att-string-join.html) | 以 string 作为分隔符，将 seq 中所有的元素(的字符串表示)合并为一个新的字符串 |
| [string.ljust(width)](https://www.runoob.com/python/att-string-ljust.html) | 返回一个原字符串左对齐,并使用空格填充至长度 width 的新字符串 |
| [string.lower()](https://www.runoob.com/python/att-string-lower.html) | 转换 string 中所有大写字符为小写. |
| [string.lstrip()](https://www.runoob.com/python/att-string-lstrip.html) | 截掉 string 左边的空格 |
| [string.maketrans(intab, outtab])](https://www.runoob.com/python/att-string-maketrans.html) | maketrans() 方法用于创建字符映射的转换表，对于接受两个参数的最简单的调用方式，第一个参数是字符串，表示需要转换的字符，第二个参数也是字符串表示转换的目标。 |
| [max(str)](https://www.runoob.com/python/att-string-max.html) | 返回字符串 *str* 中最大的字母。 |
| [min(str)](https://www.runoob.com/python/att-string-min.html) | 返回字符串 *str* 中最小的字母。 |
| [**string.partition(str)**](https://www.runoob.com/python/att-string-partition.html) | 有点像 find()和 split()的结合体,从 str 出现的第一个位置起,把 字 符 串 string 分 成 一 个 3 元 素 的 元 组 (string\_pre\_str,str,string\_post\_str),如果 string 中不包含str 则 string\_pre\_str == string. |
| [**string.replace(str1, str2,  num=string.count(str1))**](https://www.runoob.com/python/att-string-replace.html) | 把 string 中的 str1 替换成 str2,如果 num 指定，则替换不超过 num 次. |
| [string.rfind(str, beg=0,end=len(string) )](https://www.runoob.com/python/att-string-rfind.html) | 类似于 find()函数，不过是从右边开始查找. |
| [string.rindex( str, beg=0,end=len(string))](https://www.runoob.com/python/att-string-rindex.html) | 类似于 index()，不过是从右边开始. |
| [string.rjust(width)](https://www.runoob.com/python/att-string-rjust.html) | 返回一个原字符串右对齐,并使用空格填充至长度 width 的新字符串 |
| [string.rpartition(str)](https://www.runoob.com/python/att-string-rpartition.html) | 类似于 partition()函数,不过是从右边开始查找 |
| [string.rstrip()](https://www.runoob.com/python/att-string-rstrip.html) | 删除 string 字符串末尾的空格. |
| [**string.split(str="", num=string.count(str))**](https://www.runoob.com/python/att-string-split.html) | 以 str 为分隔符切片 string，如果 num 有指定值，则仅分隔 num+ 个子字符串 |
| [string.splitlines([keepends])](https://www.runoob.com/python/att-string-splitlines.html) | 按照行('\r', '\r\n', \n')分隔，返回一个包含各行作为元素的列表，如果参数 keepends 为 False，不包含换行符，如果为 True，则保留换行符。 |
| [string.startswith(obj, beg=0,end=len(string))](https://www.runoob.com/python/att-string-startswith.html) | 检查字符串是否是以 obj 开头，是则返回 True，否则返回 False。如果beg 和 end 指定值，则在指定范围内检查. |
| [**string.strip([obj])**](https://www.runoob.com/python/att-string-strip.html) | 在 string 上执行 lstrip()和 rstrip() |
| [string.swapcase()](https://www.runoob.com/python/att-string-swapcase.html) | 翻转 string 中的大小写 |
| [string.title()](https://www.runoob.com/python/att-string-title.html) | 返回"标题化"的 string,就是说所有单词都是以大写开始，其余字母均为小写(见 istitle()) |
| [**string.translate(str, del="")**](https://www.runoob.com/python/att-string-translate.html) | 根据 str 给出的表(包含 256 个字符)转换 string 的字符,  要过滤掉的字符放到 del 参数中 |
| [string.upper()](https://www.runoob.com/python/att-string-upper.html) | 转换 string 中的小写字母为大写 |
| [string.zfill(width)](https://www.runoob.com/python/att-string-zfill.html) | 返回长度为 width 的字符串，原字符串 string 右对齐，前面填充0 |

1. 使用选择排序算法实现排序[3, 5, 1, 7, 9, 6, 8]

5. 有如下一个字符串变量logs，请统计出每种请求类型的数量（提示：空格分割的第2列是请求类型），得到如下输出：

POST 2

GET 3

下边是logs变量：

logs = '''

111.30.144.7 "POST /mock/login/?t=GET HTTP/1.1" 200

111.30.144.7 "Get /mock/users/?t=POST HTTP/1.1" 200

111.13.100.92 "Post /mock/login/ HTTP/1.1" 200

223.88.60.88 "GET /mock/users/?t=POST HTTP/1.1" 200

111.30.144.7 "GET /mock/users/ HTTP/1.1" 200

'''