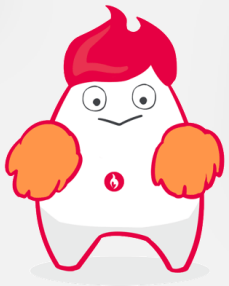
列表元组的操作符



本节课内容

- ◆len 在列表与元组上的使用
- ◆列表（元组）之间的累加与乘法
- ◆in 和 not in 在列表（元组）中的用法

len在列表元组中的使用

```
In [4]: names = ['xiaomu', 'dewei', 'xiaowang']

In [5]: length = len(names)

In [6]: print(length)
3
```

len 函数可以计算出 除了数字类型以外，其他所有的数据类型的长度

列表（元组）之间的累加与乘法

```
In [7]: names = ['xiaomu', 'dewei', 'xiaowang']

In [8]: new_names = names + names

In [9]: print(new_names)
['xiaomu', 'dewei', 'xiaowang', 'xiaomu', 'dewei', 'xiaowang']
```

+= ?
*= ?

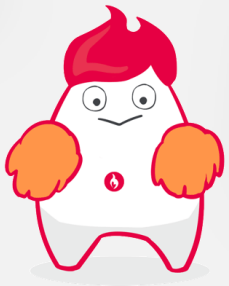
```
In [10]: names = ['xiaomu', 'dewei', 'xiaowang']

In [11]: new_names = names * 2

In [12]: print(new_names)
['xiaomu', 'dewei', 'xiaowang', 'xiaomu', 'dewei', 'xiaowang']
```

in 和 not in 在列表（元组）中的用法

- in 是判断 某个成员（元素）是否在该数据结构中
- not in 就是判断某个成员（元素）是否不在该数据结构中

```
In [1]: names = ['xiaomu', 'dewei', 'xiaoming']

In [2]: bool('xiaomu' in names)
Out[2]: True

In [3]: bool('xiaowang' not in names)
Out[3]: True
```

列表的添加-append函数



本节课内容

- append的功能
- append的用法
- append的注意事项

append的功能

- 将一个元素添加到当前列表中

append的用法

用法:

list.append(new_item)

参数:

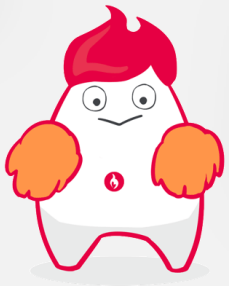
new_item: 添加进列表的新的元素(成员)

```
In [19]: names = ['xiaomu']

In [20]: names.append('dewei')

In [21]: print(names)
['xiaomu', 'dewei']
```

append的注意事项

- 被添加的元素只会被添加到末尾
- append函数是在原有列表的基础上添加，不需要额外添加新的变量

列表的添加-insert函数



本节课内容

- insert的功能
- insert的用法
- insert与append的区别

insert的功能

- 将一个元素添加到当前列表的制定位置中

insert的功能与用法

用法：

list.insert(index，new_item)

参数：

index： 新的元素放在哪个位置(数字)

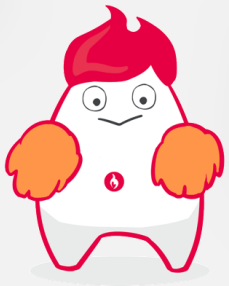
new_item： 添加的新元素 (成员)

```
In [22]: fruits = ['苹果', '西瓜', '水蜜桃']

In [23]: fruits.insert(1, '水晶梨')

In [24]: fruits
Out[24]: ['苹果', '水晶梨', '西瓜', '水蜜桃']
```

insert与append的区别

- append只能添加到列表的结尾，而insert可以选择任何一个位置
- 如果insert传入的位置列表中不存在，则将新元素添加到列表结尾
- 字符串， 元组， 字符串 元素的位置是从 0 开始 计算的

列表（元组）的count函数



慕课网 imooc.com

本节课内容

- count 的功能
- count 的用法
- count 的注意事项

count的功能

- 返回当前字符串中某个成员的个数

慕课网 imooc.com

count的用法

用法：

inttype = list.count(item)

参数：

item：你想查询个数的元素

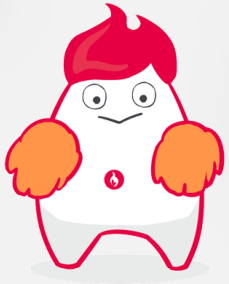
```
In [25]: fruits = ['苹果', '西瓜', '水蜜桃', '西瓜', '雪梨']

In [26]: count = fruits.count('西瓜')

In [27]: print(count)
2
```

count 的注意事项

- 如果查询的成员（元素）不存在，则返回 0
- 列表只会检查完整元素是否存在需要计算的内容

```
In [28]: fruits = ['苹果', '西瓜', '水蜜桃', '西瓜', '雪梨']

In [29]: count = fruits.count('西')

In [30]: count
Out[30]: 0
```

列表的remove函数



本节课内容

- remove的功能
- remove的用法
- remove 的注意事项
- Python内置函数 del

remove功能

- 删除列表中的某个元素

remove用法

用法:

list.remove(item)

参数:

item: 准备删除的列表元素

```
In [31]: drinks = ['雪碧', '可乐', '矿泉水']

In [32]: drinks.remove('矿泉水')

In [33]: drinks
Out[33]: ['雪碧', '可乐']
```

remove 的注意事项

- 如果删除的成员（元素）不存在，会直接报错
- 如果被删除的元素有多个，只会删除第一个
- remove函数不会返回一个新的列表，而是在原先的列表中对元素进行删除

Python的内置函数 del

◆ del 把变量完全删除

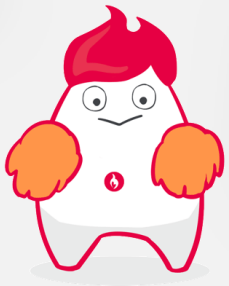
```
In [34]: drinks = ['雪碧', '可乐', '矿泉水']

In [35]: del drinks

In [36]: print(drinks)
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-36-75a320656267> in <module>
----> 1 print(drinks)

NameError: name 'drinks' is not defined
```

pop和del我们稍后见

- pop是列表中另一个删除函数
- del也可以删除列表中的指定元素
- 我们会在学习索引的时候为大家讲解这两个函数的用法

慕课网 imooc.com

列表的reverse函数



本节课内容

- reverse 的功能
- reverse 的用法

reverse的功能

- 对当前列表顺序进行反转

reverse的用法

用法：

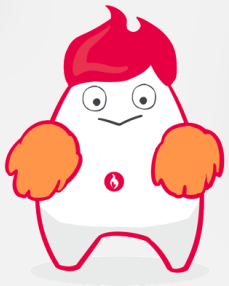
list.reverse()

参数：

无参数传递

```
In [37]: drinks = ['雪碧', '可乐', '矿泉水']

In [38]: drinks.reverse()

In [39]: print(drinks)
['矿泉水', '可乐', '雪碧']
```

列表的sort函数



本节课内容

- sort 的功能
- sort 的用法
- sort 的主意事项

sort的功能

- 对当前列表按照一定规律进行排序

慕课网 imooc.com

sort的用法

用法：

list.sort(key=None, reverse=False)

参数：

key – 参数比较

reverse -- 排序规则，reverse = True 降序， reverse = False 升序 （默认）

key涉及函数学习，我们在日后讲解当前默认不传即可

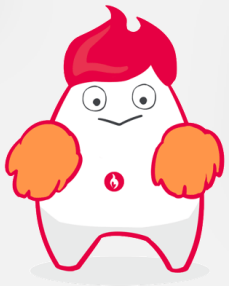
```
In [44]: books = ['python', 'django', 'web', 'flask', 'tornado']

In [45]: books.sort()

In [46]: print(books)
['django', 'flask', 'python', 'tornado', 'web']
```

sort的注意事项

◆ 列表中的元素类型必须相同，否则无法排序（报错）

```
In [47]: mixs = ['python', 1, 1.5, 'django']

In [48]: mixs.sort()
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-48-72498a9dd70d> in <module>
----> 1 mixs.sort()

TypeError: '<' not supported between instances of 'int' and 'str'
```

列表的clear函数



本节课内容

- clear的功能
- clear的用法

clear的功能

- 将当前列表中的数据清空

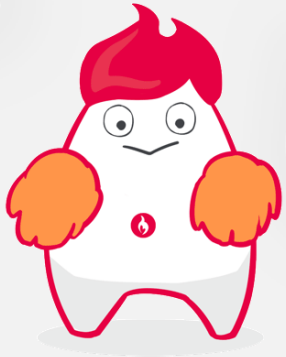
clear用法

用法:

list.clear() -> 该函数无参数，无返回值

```
In [49]: target = [1, 2, 3, 4, 5, 6]

In [50]: target.clear()

In [51]: print(target)
[]
```

列表的copy函数



本节课内容

- copy 的功能
- copy 的用法
- copy 与 2次赋值的区别

copy功能

将当前的列表复制一份相同的列表，新列表与旧列表内容相同，但内存空间不同

copy用法

用法：

list.copy() -> 该函数无参数，返回一个一模一样的列表

```
In [52]: old_list = ['a', 'b', 'c']

In [53]: new_list = old_list.copy()

In [54]: print(new_list)
['a', 'b', 'c']
```

copy与二次赋值的区别

a = [1,2,3]

b = a

- 二次赋值的变量与原始变量享有相同内存空间
- copy函数创建的新列表与原始列表不是一个内存空间，不同享数据变更
- copy 属于 浅拷贝

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

b = a.copy()

b.append(4)

b
[1, 2, 3, 4]

a
[1, 2, 3]
```

浅拷贝

◆ 通俗的说，我们有一个列表a，列表里的元素还是列表，当我们拷贝出新列表b后，无论是a还是b的内部的列表中的数据发生了变化后，相互之间都会受到影响，– 浅拷贝

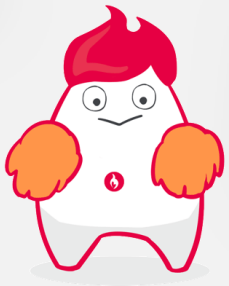
```
a = [[1, 2, 3], [5, 6, 7]]

b = a.copy()

b
[[1, 2, 3], [5, 6, 7]]

b[0].append(10)

b
[[1, 2, 3, 10], [5, 6, 7]]

a
[[1, 2, 3, 10], [5, 6, 7]]
```

深拷贝

- 不仅对第一层数据进行了copy，对深层的数据也进行copy，原始变量和新变量完完全全不共享数据 – 深拷贝

```
a = [[1,2,3], [4,5,6]]

b = copy.deepcopy(a)

b
[[1, 2, 3], [4, 5, 6]]

b[0].append(10)

b
[[1, 2, 3, 10], [4, 5, 6]]

a
[[1, 2, 3], [4, 5, 6]]
```

列表的extend函数



本节课内容

- extend 功能
- extend 用法

extend的功能

- 将其他列表或元组中的元素倒入到当前列表中

extend的功能

用法：

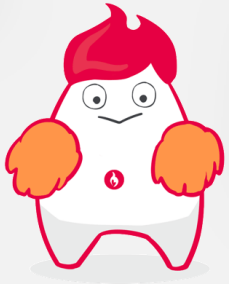
list.extend(iterable) ->

参数：

iterable 代表列表或元组，该函数无返回值

```
In [55]: students = ['dewei', 'xiaomu', 'xiaogang']

In [56]: new_students = ('xiaowang', 'xiaohong')

In [57]: students.extend(new_students)

In [58]: students
Out[58]: ['dewei', 'xiaomu', 'xiaogang', 'xiaowang', 'xiaohong']
```

索引与切片之列表



本节课内容

- 什么是索引
- 什么是切片
- 列表的索引，获取与修改
- 通过 pop 删除索引
- 通过 del 删除索引
- 索引在元组中的特殊性

什么是索引

- 字符串，列表和元组
- 从最左边记录的位置就是索引
- 索引用数字表示，起始从 0 开始

- 字符串，列表（元组）的最大索引是他们的长度 – 1

```
In [59]: I = ['name']

In [60]: I[0]
Out[60]: 'name'
```

```
In [61]: I[1]
---------------------------------
IndexError
<ipython-input-61-86e9f237d85c> in
----> 1 I[1]

IndexError: list index out of range
```

什么是切片

- 索引用来对单个元素进行访问，切片则对一定范围内的元素进行访问
- 切片通过冒号在中括号内把像个的两个索引查找出来 [0: 10]

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9 ,10]
print(numbers[3: 8])
>> [4, 5, 6, 7, 8]
```

- 切片规则为： 左含，右不含

列表的索引，获取与修改

- list[index] = new_item
- 数据的修改只能在存在的索引范围内
- 列表无法通过添加新的索引的方式赋值
- list.index(item)

tests = ['a', 'b', 'c']

tests[2] = 's' ok

tests[3] = 'o' -> X

```
In [65]: names = ['dewei', 'xiaoman']

In [66]: names.index('dewei')
Out[66]: 0

In [67]: names.index('xiaomu')
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-67-8d084ff66f96> in <module>
----> 1 names.index('xiaomu')

ValueError: 'xiaomu' is not in list
```

pop的功能

- 通过索引删除并获取列表的元素

pop的用法

用法:

list.pop(index)

参数:

index: 删除列表的第几个索引

-> 函数会删除该索引的元素并返回

-> 如果传入的index索引不存在则报错

```
In [75]: names = ['dewei', 'xiaomu']

In [76]: pop_item = names.pop(0)

In [77]: print('pop item:', pop_item, 'names:', names)
pop item: dewei names: ['xiaomu']
```

通过del删除索引

del list[index]

- 直接删除 无返回值
- 如果index（索引）不存在则报错

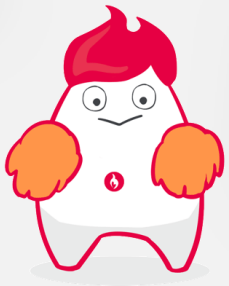
```
In [78]: names = ['dewei', 'xiaomu']

In [79]: del names[1]

In [80]: names
Out[80]: ['dewei']
```

索引切片在元组中的特殊性

- 可以和列表一样获取索引与切片索引
- 元组函数 index 和列表用法完全一致
- 无法通过索引修改与删除元素

慕课网 imooc.com

索引与切片之字符串



本节课内容

- 字符串的索引，获取
- 字符串的find与index函数

慕课网 imooc.com

字符串的索引与获取

- ◆索引规则与列表相同
- ◆切片和索引的获取与列表相同
- ◆无法通过索引修改与删除
- ◆**字符串不可修改**

name = ’ dewei’

name[0] -> d

name[:2] -> de

字符串的find与index函数

功能：

获取元素的索引位置

用法：

string.index(item) -> item: 查询个数的元素, 返回索引位置

string.find(item) -> item: 查询个数的元素, 返回索引位置

```
In [85]: info = 'my name is dewei'

In [86]: info.index('dewei')
Out[86]: 11

In [87]: info.find('dewei')
Out[87]: 11
```

find与index的区别

- find如果获取不到，返回-1
- index如果获取不到，直接报错

```
In [88]: info = 'my name is dewei'

In [89]: info.find('xiaomu')
Out[89]: -1

In [90]: info.index('xiaomu')
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-90-ac642654eb73> in <module>
----> 1 info.index('xiaomu')

ValueError: substring not found
```