Python核心数据类型-列表与元组

一. 列表的特性:

1. 列表是任意对象的有序集合, 列表中可以包含任何种类的对象, 如数字 字符串 或者是其他列表, 列表所包含的每一项都保持了从左到右的位置顺序.
2. 列表可以通过偏移来读取其中的元素, 也可以通过分片读取列表中的某一部分元素.
3. 变长 异构 任意嵌套: 列表可以增长或者缩短(长度可变), 并且可以包含任何类型的对象(异构), 因为列表能够包含其他复杂的对象, 又支持任意嵌套, 所以可以在列表中嵌套多层列表.
4. 可变类型: 列表支持在原处修改.

In [48]:
```
#示例代码1.

#1.列表是任意对象的有序集合
l = [1, 'a', [1, 2,3], {"x": 1}]
print l

#2.列表可以任意嵌套
l = [[1, 2,3,4], [2,3, [4, [5, 6]]]]
print l

#3.列表中的元素是可变的
l = [1, 2, 3]
print l
l[0] = 'apple'
print l
```

```
[1, 'a', [1, 2, 3], {'x': 1}]
[[1, 2, 3, 4], [2, 3, [4, [5, 6]]]]
[1, 2, 3]
['apple', 2, 3]
```

二. 列表的常用操作

In [1]:
```
l = [1,2,3,4]

#1.len()获取列表的长度
print len(l) #获取列表的长度
```

```
4
```

In [3]:
```
#2.list()将字符串 元组 集合转换成列表

x = 'apple'
print list(x)

x = (1,2,3,4)
print list(x)

x = set([1,2,3,4])
print list(x)
```

```
['a', 'p', 'p', 'l', 'e']
[1, 2, 3, 4]
[1, 2, 3, 4]
```

三. 列表的常用方法

In [1]:
```
print(dir([]))
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__
gt__', '__hash__', '__iadd__', '__imul__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__', '__reduce__',
'__reduce_ex__', '__repr__', '__reversed__', '__rmul__', '__setattr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'append', 'clear', 'copy', 'coun
t', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']
```

In [20]:
```
#1.append向列表的尾部添加一个对象
help(l.append)
l = [1,2,3,4]
l.append(1)
l.append('apple')
l.append([1,2,3])
l.append((1,2,4))
print l
```

```
Help on built-in function append:

append(...)
    L.append(object) -- append object to end

[1, 2, 3, 4, 1, 'apple', [1, 2, 3], (1, 2, 4)]
```

In [23]:
```
#2.extend把一个可迭代对象中的每个元素添加到列表的尾部
help(l.extend)
l = [1,2,3]
l.extend('apple')
l.extend(['a', 'b', 'c'])
l.extend(("x", "y", "z"))
print l
```

```
Help on built-in function extend:

extend(...)
    L.extend(iterable) -- extend list by appending elements from the iterable

[1, 2, 3, 'a', 'p', 'p', 'l', 'e', 'a', 'b', 'c', 'x', 'y', 'z']
```

In [25]:
```
#3.insert向列表指定索引位置之前添加一个对象
help(l.insert)
l = [1,2,3,4]
l.insert(1, 'apple')
print l
```

```
Help on built-in function insert:

insert(...)
    L.insert(index, object) -- insert object before index

[1, 'apple', 2, 3, 4]
```

In [28]:
```
#4.pop从列表中删除指定索引位置元素，并将来其返回，如果列表为空或者索引超出边界则报indexerror
help(l.pop)
l.pop(0)
l.pop(10)
```

```
Help on built-in function pop:

pop(...)
    L.pop([index]) -> item -- remove and return item at index (default last).
    Raises IndexError if list is empty or index is out of range.


---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-28-de19ba3427cd> in <module>()
      2 help(l.pop)
      3 l.pop(0)
----> 4 l.pop(10)

IndexError: pop index out of range
```

In [31]:
```
#5.remove从列表中删除你所指定的元素，如果该元素有多个，则删除它第一次出现时的那个
help(l.remove)
l = [2,1,3,1,4]
l.remove(1)
print l
```

```
Help on built-in function remove:

remove(...)
    L.remove(value) -- remove first occurrence of value.
    Raises ValueError if the value is not present.

[2, 3, 1, 4]
```

In [33]:
```
#6.count列表中的某一个元素出现的次数
help(l.count)
l = [2, 1, 3, 1, 4]
print l.count(1)
print l.count(2)
```

```
Help on built-in function count:

count(...)
    L.count(value) -> integer -- return number of occurrences of value

2
1
```

In [35]:
```
#7.index列表中的某个元素第一次出现时的索引
help(l.index)
l = [2,1,3,1,4]
print l.index(1)
```

```
Help on built-in function index:

index(...)
    L.index(value, [start, [stop]]) -> integer -- return first index of value.
    Raises ValueError if the value is not present.

1
```

In [42]:
```
#8.sort对列表中的元素进行排序
help(l.sort)

l = [1, 4, 2, 5, 10]
l.sort() #升序排列
print l

l = [1, 4, 2, 5, 10]
l.sort(reverse=True) #降序排列
print l
```

```
Help on built-in function sort:

sort(...)
    L.sort(cmp=None, key=None, reverse=False) -- stable sort *IN PLACE*;
    cmp(x, y) -> -1, 0, 1

[1, 2, 4, 5, 10]
[10, 5, 4, 2, 1]
```

In [40]:
```
#9.reverse对列表中的元素进行反转
l = ["x", 1, "y", "a", 'e']
help(l.reverse)
l.reverse()
print l
```

```
Help on built-in function reverse:

reverse(...)
    L.reverse() -- reverse *IN PLACE*

['e', 'a', 'y', 1, 'x']
```

四.元组的特点

任意对象的有序集合: 与列表相同, 元组是一个位置有序的对象的集合, 内容维持着从左到右的顺序, 支持异构(也就是元组中可以嵌套其他的核心数据类型) Python元组与列表类似, 但是元组属于不可变类型.

五.元组的常用操作

In [46]:
```
l = (1,2,3,4)

#1.len()获取元组的长度
print len(l)
```

```
4
```

In [49]:
```
#2.tuple()将列表 字符串 集合转换为元组
l = [1,2,3]
print tuple(l)

l = 'apple'
print tuple(l)

l = set([1,2,3,4])
print tuple(l)

#3.元组中的元素是不可变的
l = (1,2,3)
l[0] = 'apple'
```

```
(1, 2, 3)
('a', 'p', 'p', 'l', 'e')
(1, 2, 3, 4)

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-49-eed90179e8d9> in <module>()
     11 #3.元组中的元素是不可变的
     12 l = (1,2,3)
---> 13 l[0] = 'apple'

TypeError: 'tuple' object does not support item assignment
```

六.元组的常用方法

In [3]:
```
print(dir(()))
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getnewargs__',
'__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__r
epr__', '__rmul__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', 'count', 'index']
```

In [52]:
```
#1.count元组中的某一个元素出现的次数
l = (1,2,3, 1)
help(l.count)
print l.count(1)
```

```
Help on built-in function count:

count(...)
    T.count(value) -> integer -- return number of occurrences of value

2
```

In [54]:
```
#2.index元组中的某个元素第一次出现时的索引
l = (1,2,3, 1)
help(l.index)
print l.index(1)
```

```
Help on built-in function index:

index(...)
    T.index(value, [start, [stop]]) -> integer -- return first index of value.
    Raises ValueError if the value is not present.

0
```