



04 Python语言基础(2)

西安科技大学 牟琦
muqi@xust.edu.cn



4.1 内置数据结构



4.1.1 序列数据结构

■ 序列数据结构 (sequence)

- 成员是**有序**排列的
- 每个元素的位置称为**下标**或**索引**
- 通过索引访问序列中的成员
- Python中的序列数据类型有**字符串**、**列表**、**元组**
"abc" \neq "bca"
- Python中的列表和元组，可以存放**不同类型**的数据
- **列表**使用方括号[]表示，**元组**使用小括号()表示。

列表: [1,2,3]

元组: (1,2,3)



□ 列表 (list)

```
lst_1=[1,2,3]
lst_2=[4]
lst_3=[[1,2,3],[4,5,6]]

lst_mix=[160612,"张明",18,[92,76,85]]
lst_empty=[ ]
```

□ 元组 (tuple) : 一经定义, 元组的内容不能改变。

```
>>>tup_1=(1,2,3)
```

```
>>>tup_empty=()
```

```
>>>tup_empty
```

```
()
```

```
>>>t1=(1)
```

```
>>>t2=(1,)
```

```
>>> print("t1=",t1,type(t1))
```

```
t1= 1 <class 'int'>
```

```
>>> print("t2=",t2,type(t2))
```

```
t2= (1,) <class 'tuple'>
```



索引（下标）：通过它访问序列中的元素

-6 -5 -4 -3 -2 -1 逆向索引
Python
正向索引: 0 1 2 3 4 5

-3 -2 -1 逆向索引
[1,2,3]
正向索引: 0 1 2

```
# 字符串索引
>>>str_py = "Python"
>>>print(str_py[0])
P
>>>print(str_py[-1])
n
```

```
#列表索引
>>> lst_1=[1,2,3]
>>> print(lst_1[1])
2
>>> print(lst_1[-2])
2
```



■ 切片：一次性从序列中获取多个元素

[开始位置:结束位置]

前闭后开: 切片不包括结束位置的元素

Python

正向索引: 0 1 2 3 4 5

字符串切片

```
>>>str_py="Python"
>>>print(str_py[1:5])
'ytho'
print(str_py[1:])
'ython'
print(str_py[:5])
'Pytho'
```

[1,2,3]

正向索引: 0 1 2

#列表切片

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

开始位置缺省, 从**第一个元素**开始
结束位置缺省, 到**最后一个元素**为止
开始位置和结束位置都缺省, 取到**整个序列**



例：混合列表切片

正向索引: 0 1 2 3
 [160612, "张明", 18, [92, 76, 85]]

```
#列表切片
>>>lst_stud=[160612,"张明",18,[92,76,85]]

>>> list_stud[1:]
["张明",18,[92,76,85]]

>>> list_stud[:3]
[160612,"张明",18]
```



■ 打印——print()

```
>>>lst_1=[1,2,3]
>>>print(lst_1)           #打印整个列表
[1,2,3]

>>>tup_1=(1,2,3)
>>>print(tup_1)           #打印整个元组
(1,2,3)

>>>print(lst_1[0])        #打印列表中的元素
1

>>>print(tup_1[0:2])      #打印元组中的元素
(1,2)
```



■ 获取序列的长度——len(序列名称)

```
#获取字符串的长度
```

```
>>>len("Python")
```

```
6
```

```
>>>str="Python"
```

```
>>>len(str)
```

```
6
```

```
#获取列表的长度
```

```
>>>lst_1=[1,2,3]
```

```
>>>len(lst_1)
```

```
3
```

```
#获取二维列表的长度
```

```
lst_3=[[1,2,3],[4,5,6]]
```

```
>>>len(lst_3)
```

```
2
```

```
#获取混合列表的长度
```

```
>>>lst_mix=[100612,"张明",18,[92,76,85]]
```

```
>>>len(lst_mix)
```

```
4
```

```
# 获取元组的长度
```

```
>>>tup_1=(1,2,3)
```

```
>>>len(tup_1)
```

```
3
```



■ 更新列表 ——向列表中**添加元素**

□ append()

#向列表中追加元素

```
>>>lst_1=[1,2,3]
>>>lst_1.append(4)
>>>lst_1
[1,2,3,4]
```

□ insert()

#向列表中追加元素

```
>>>lst_1=[1,2,3]
>>>lst_1.insert(1,5)
>>>lst_1
[1,5,2,3]
```

由于元组一经定义后就不能更改了，因此**元组不支持更新操作**。



■ 更新列表——合并列表

□ extend()

#合并列表

```
>>>lst_1=[1,2,3]
>>>lst_2=[4]
>>>lst_1.extend(lst_2)
>>>lst_1
[1,2,3,4]
```

□ "+" 运算符

```
>>>lst_1=[1,2,3]
>>>lst_2=[4]
>>>lst_3=lst_1+lst_2
>>>lst_3
[1,2,3,4]
```



■ 更新列表——删除列表中的元素

□ del 语句

```
>>>lst_1=[1,2,3,4]
>>>del lst_1[1]          #删除下标为1的元素
>>>lst_1
[1,3,4]
```



■ 更新列表——排序

- `sort()`：对列表中的元素排序
- `reverse()`：对列表中的元素倒排序

```
>>>lst_1=[2,3,1,4]
>>>lst_1.sort()           #将lst_1中的元素按从小到大的顺序排列
>>>lst_1
[1,2,3,4]

>>>lst_1.reverse()       #将lst_1中的元素原地逆序
>>>lst_1
[4,3,2,1];
```



■ 遍历列表中的元素

```
#遍历列表中的元素
>>>lst_1=[1,2,3,4]
>>>for i in lst_1:
    print(i,end=" ")
1 2 3 4
```

