



Python 培训第二期

By 李亚军

目 录

- ['列表', 'list']
- ('元祖', 'tuple')
- '字符串string'
- {'字典': 'dictionary'}
- {'集合', 'set'}
- for 循环和 while 循环
- with语句

列表list

- 列表可以如下几种方式创建：

方括号

- `[]`
- `[1, 2, 3]`

列表推导式

- In [1]: `[x for x in range(3)]`
- Out[1]: `[0, 1, 2]`

`list()`函数

- In [2]: `list('abc')`
- Out[2]: `['a', 'b', 'c']`

列表list

方法	说明	示例
<code>list.sort(<i>key=None, reverse=False</i>)</code>	对列表进行排序，默认是升序	<code>a.sort()</code>
<code>list.append(x)</code>	在列表最后添加一个元素	<code>a.append(10)</code>
<code>list.insert(i, x)</code>	在给定位置处插入一个元素	<code>a.insert(0, x)</code>
<code>list.count(x)</code>	返回元素 x 出现的次数	<code>a.sort(1)</code>
<code>list.pop([i])</code>	移除给定位置的元素	<code>a.pop(1)</code>
<code>list.index(x[, start[, end]])</code>	返回列表中第一个值为 x 的元素的下标	<code>a.index(1)</code>
<code>list.reverse()</code>	反转列表元素	<code>a.reverse()</code>

元组tuple

- 元组可以使用以下方式创建

圆括号

- `()`
- `(1,)`

逗号分隔

- In [2]: `1, 2, 3` # `(1, 2, 3)`
- Out[2]: `(1, 2, 3)`

`tuple()`函数

- In [3]: `tuple('abc')`
- Out[3]: `('a', 'b', 'c')`

元组tuple

- 元组是一种不可变类型，所以没有类似列表的 `append`、`insert` 等方法，但是有 `count` 和 `index` 方法。
- 元组和列表可以互相嵌套。

```
([1, 2, 3], [3, 2, 1])  
[(1, 2, 3), (3, 2, 1)]
```

- 元组解包

```
In [15]: t = 12345, 54321, 'hello!'  
In [16]: x, y, z = t  
In [17]: x  
Out[17]: 12345  
In [18]: y  
Out[18]: 54321  
In [19]: z  
Out[19]: 'hello!'
```

字符串string

- 字符串也是一种不可变类型，常用的方法有如下几种：

方法	说明	示例
<code>str.encode(encoding="utf-8", errors="strict")</code>	使用指定编码格式对字符串进行编码。	<code>'中国'.encode('utf-8')</code>
<code>str.find(sub[, start[, end]])</code>	返回第一个出现 sub 字符串的位置。	<code>'中国'.find('中')</code>
<code>str.join(iterable)</code>	连接字符串	<code>''.join(['中', '国'])</code>
<code>str.lstrip([chars])</code>	去掉字符串左侧的指定字符	<code>' 中国'.lstrip()</code>
<code>str.split(sep=None, maxsplit=-1)</code>	拆分字符串	<code>'中 国'.split(' ')</code>
<code>str.replace(old, new[, count])</code>	用新字符串替换旧字符串	<code>'中国'.replace('中', '美')</code>

字典dictionary

- 字典可以如下创建：

```
In [33]: a = dict(one=1, two=2, three=3)
```

```
In [34]: b = {'one': 1, 'two': 2, 'three': 3}
```

```
In [35]: c = dict(zip(['one', 'two', 'three'], [1, 2, 3]))
```

```
In [36]: d = dict([('two', 2), ('one', 1), ('three', 3)])
```

```
In [37]: e = dict({'three': 3, 'one': 1, 'two': 2})
```

```
In [38]: a == b == c == d == e
```

```
Out[38]: True
```

```
In [39]: a
```

```
Out[39]: {'one': 1, 'three': 3, 'two': 2}
```


字典dictionary

```
In [42]: a.keys()
```

```
Out[42]: dict_keys(['two', 'three', 'one'])
```

```
In [43]: a.values()
```

```
Out[43]: dict_values([2, 3, 1])
```

```
In [44]: a.items()
```

```
Out[44]: dict_items([('two', 2), ('three', 3), ('one', 1)])
```

```
In [45]: 'four' in a
```

```
Out[45]: False
```

```
In [46]: a['one']
```

```
Out[46]: 1
```

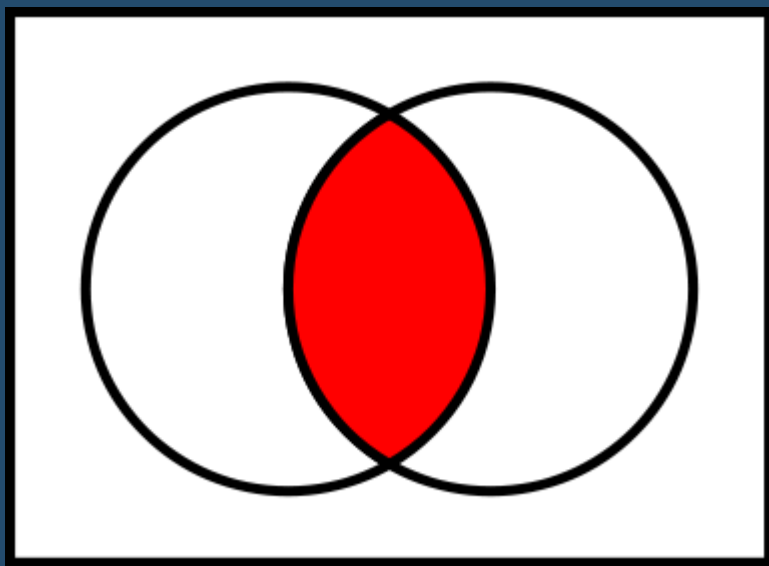
```
In [47]: a['four'] = 4
```

```
In [48]: a
```

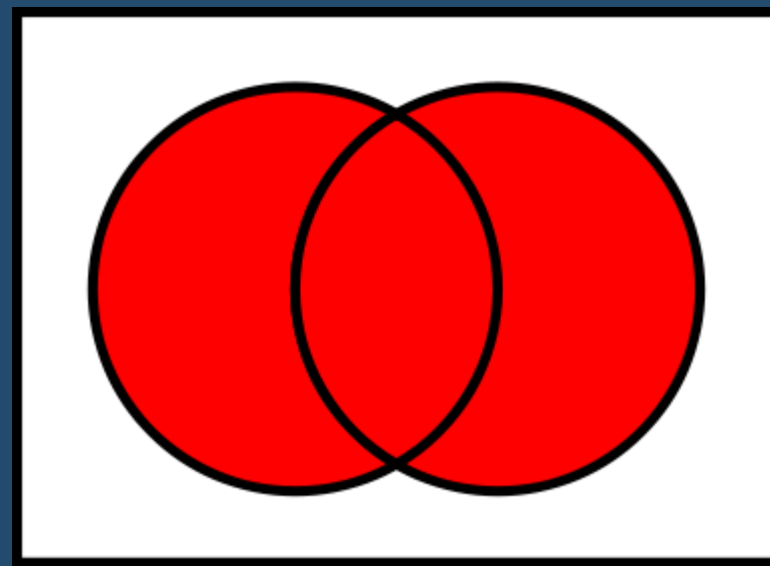
```
Out[48]: {'four': 4, 'one': 1, 'three': 3, 'two': 2}
```

集合set

- 集合是一种无序类型，其元素都是不重复的，经常用于去除重复项。
- 同数学中的集合一样，这里的集合也有交、并、差等操作。



$$A \cap B$$



$$A \cup B$$

集合set

```
In [49]: basket = {'apple', 'orange', 'apple', 'pear', 'orange', 'banana'}
```

```
In [50]: print(basket)
```

```
{'apple', 'pear', 'orange', 'banana'}
```

```
In [51]: 'orange' in basket
```

```
Out[51]: True
```

```
In [53]: a = set('abracadabra')
```

```
In [54]: b = set('alacazam')
```

```
In [55]: a - b # 在 a 中但不在 b 中的字符
```

```
Out[55]: {'b', 'd', 'r'}
```

```
In [56]: a | b # a 与 b 的并
```

```
Out[56]: {'a', 'b', 'c', 'd', 'l', 'm', 'r', 'z'}
```

```
In [57]: a & b # a 与 b 的交
```

```
Out[57]: {'a', 'c'}
```

```
In [58]: a ^ b # 除去 a 与 b 都有的元素
```

```
Out[58]: {'b', 'd', 'l', 'm', 'r', 'z'}
```

for 循环和 while 循环

```
odd = []
even = []
for i in range(10):
    if i == 0:
        continue
    elif i == 8:
        break
    elif i % 2 == 0:
        even.append(i)
    else:
        odd.append(i)
print(odd) # [1, 3, 5, 7]
print(even) # [2, 4, 6]
```



for

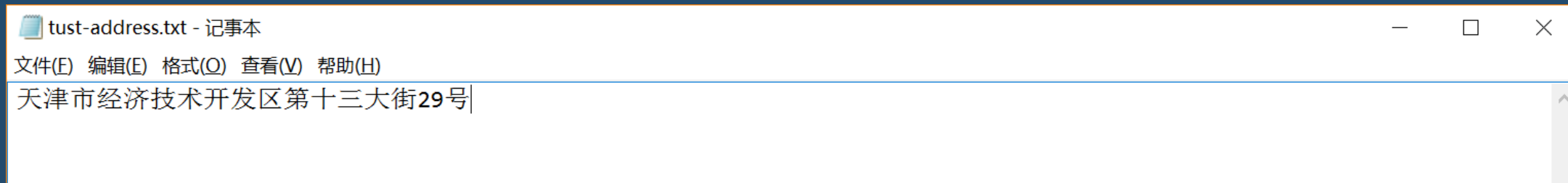
while 

```
odd = []
even = []
i = 0
while i < 10:
    if i == 0:
        i += 1
        continue
    elif i == 8:
        break
    elif i % 2 == 0:
        even.append(i)
    else:
        odd.append(i)
    i += 1
print(odd) # [1, 3, 5, 7]
print(even) # [2, 4, 6]
```

with 语句

- with 语句定义了一个上下文管理器（context managers），也就是当某程序运行时所需要的环境。最常见的是文件的相关操作。

```
s = '天津市经济技术开发区第十三大街29号'  
with open('tust-address.txt', 'w') as f:  
    f.write(s)
```



THANK YOU
FOR LISTENING

