nsd1902_py01_day05

markdown 编辑器：typora : https://www.typora.io/

列表和元组

列表操作

```
>>> from random import randint
>>> alist = [randint(1, 100) for i in range(10)]
>>> alist
[95, 58, 74, 39, 42, 32, 41, 35, 40, 65]
>>> alist[0] = 100
>>> alist
[100, 58, 74, 39, 42, 32, 41, 35, 40, 65]
>>> alist[3:5] = [25, 46, 89]
>>> alist
[100, 58, 74, 25, 46, 89, 32, 41, 35, 40, 65]
>>> alist[2:2] = [10, 20]
>>> alist
[100, 58, 10, 20, 74, 25, 46, 89, 32, 41, 35, 40, 65]
```

列表方法

```
>>> alist.append(100)   # 追加
>>> alist
[100, 58, 10, 20, 74, 25, 46, 89, 32, 41, 35, 40, 65, 100, [10, 20]]
>>> alist.append([10, 20])
>>> alist.extend([10, 20])   # 把序列对象的每一项逐个放到列表中
>>> alist
[100, 58, 10, 20, 74, 25, 46, 89, 32, 41, 35, 40, 65, 100, [10, 20], 10, 20]
>>> alist.remove(10)   # 删除列表中的第一个10
>>> alist.pop()   # 默认弹出最后一项
20
>>> alist.pop(-2)   # 弹出列表下标为-2的项
[10, 20]
>>> alist
[100, 58, 20, 74, 25, 46, 89, 32, 41, 35, 40, 65, 100, 10]
>>> alist.index(46)   # 获取46的下标
5
>>> alist.reverse()   # 将列表原地翻转
>>> alist
[10, 100, 65, 40, 35, 41, 32, 89, 46, 25, 74, 20, 58, 100]
>>> alist.insert(4, 100)   # 在下标为4的位置插入100
>>> alist.count(100)   # 统计100出现的次数
>>> alist.sort()  # 默认升序排列
>>> alist.sort(reverse=True)   # 降序排列
```

```
>>> alist
[100, 100, 100, 89, 74, 65, 58, 46, 41, 40, 35, 32, 25, 20, 10]
>>> blist = alist.copy()  # 将alist的值拷贝一份后，赋值给blist
>>> blist
[100, 100, 100, 89, 74, 65, 58, 46, 41, 40, 35, 32, 25, 20, 10]
>>> blist.clear()   # 清空列表
>>> blist
[]
>>> alist
[100, 100, 100, 89, 74, 65, 58, 46, 41, 40, 35, 32, 25, 20, 10]
```

元组

元组相当于是静态的列表，不可改变。

```
>>> atuple = (10, 20, 30, 20, 40, 20)
>>> atuple.<tab><tab>
atuple.count(  atuple.index(
>>> atuple.count(20)
3
>>> atuple.index(20)
1
```

单元素元组注意，一定要在结尾有个逗号

```
>>> a = (10)
>>> a
10
>>> type(a)   # 查看a的类型
<class 'int'>
>>> b = (10,)
>>> b
(10,)
>>> type(b)
<class 'tuple'>
>>> len(b)
1
```

列表模拟栈结构

1. 思考程序运行方式

```
(0) 压栈
(1) 出栈
(2) 查询
(3) 退出
请选择(0/1/2/3): 5
无效输入，请重试。
(0) 压栈
(1) 出栈
(2) 查询
(3) 退出
```

```
请选择(0/1/2/3): 2
[]
(0) 压栈
(1) 出栈
(2) 查询
(3) 退出
请选择(0/1/2/3): 0
数据: hello
(0) 压栈
(1) 出栈
(2) 查询
(3) 退出
请选择(0/1/2/3): 2
['hello']
(0) 压栈
(1) 出栈
(2) 查询
(3) 退出
请选择(0/1/2/3): 1
从栈中弹出：hello
(0) 压栈
(1) 出栈
(2) 查询
(3) 退出
请选择(0/1/2/3): 1
空栈
(0) 压栈
(1) 出栈
(2) 查询
(3) 退出
请选择(0/1/2/3): 3
bye-bye
```

2. 把功能编写成函数

```python
def push_it():
    pass

def pop_it():
    pass

def view_it():
    pass

def show_menu():
    pass

if __name__ == '__main__':
    show_menu()
```

3. 编写每一个功能函数

字典

属于容器、可变、映射

```
>>> adict = dict(['ab', ['name', 'bob'], ('age', 20)])
>>> adict
{'a': 'b', 'name': 'bob', 'age': 20}

# 创建值相同的字典
>>> bdict = {}.fromkeys(['jerry', 'tom', 'bob'], 20)
>>> bdict
{'jerry': 20, 'tom': 20, 'bob': 20}

# 字典中的key不能重复
>>> adict = {'name': 'tom', 'age': 20}
>>> adict['age']
20
# 赋值时，key已存在则修改它的值
>>> adict['age'] = 22
>>> adict
{'name': 'tom', 'age': 22}
# 赋值时，key不存在，则增加新项目
>>> adict['email'] = 'tom@tedu.cn'
>>> adict
{'name': 'tom', 'age': 22, 'email': 'tom@tedu.cn'}

# tom是字典的key吗？
>>> 'tom' in adict
False
>>> for key in adict:
...     print(key, adict[key])

>>> '%(name)s: %(age)s %(email)s' % adict
'tom: 22 tom@tedu.cn'
```

字典的key只能是不可变对象

```
>>> adict[[1, 2]] = 10
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unhashable type: 'list'
>>> bdict[(1, 2)] = 10
>>> bdict
{(1, 2): 10}
```

字典的相关方法：

```
>>> adict.items()
dict_items([('name', 'tom'), ('age', 22), ('email', 'tom@tedu.cn')])
# 遍历字典
>>> for key, val in adict.items():
```

```
...     print(key, val)
...
name tom
age 22
email tom@tedu.cn

>>> adict.keys()   # 只取出key
dict_keys(['name', 'age', 'email'])
>>> adict.values()  # 只取出value
dict_values(['tom', 22, 'tom@tedu.cn'])  # 更新字典
>>> adict.update({'qq': '13542632', 'phone': '15088997766'})
>>> adict
{'name': 'tom', 'age': 22, 'email': 'tom@tedu.cn', 'qq': '13542632', 'phone':
'15088997766'}

>>> adict.popitem()   # 随机移出一项
('phone', '15088997766')
>>> adict.pop('qq')   # 通过key删除项
'13542632'

>>> adict['birth_date']   # 没有相应的key，则报错
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'birth_date'

# get是字典中用得最多的一个方法
>>> print(adict.get('birth_date'))  # key不在字典中默认返回None
None
>>> adict.get('age')
22
>>> adict.get('age', 30)   # 如果get到值则返回值，get不到则返回30
22
>>> adict.get('birth_date', 'not found')
'not found'
```

集合

集合用set表示，由不同（不可变）元素构成。

集合分为不可变集和forzenset和可变集合set。

```
>>> fs = frozenset('abc')   # 创建不可变集合
>>> s1 = set('abc')         # 创建可变集合
>>> fs
frozenset({'a', 'c', 'b'})
>>> s1
{'a', 'c', 'b'}
>>> s10 = set(['aaa', 'bbb', 'ccc'])
>>> s10
{'ccc', 'bbb', 'aaa'}

>>> len(s10)
```

```
3
>>> for word in s10:
...     print(word)
>>> 'aaa' in s10
True
>>> 'fff' in s10
False
```

集合和字典都是无序的，字典的key和集合元素都是不可变的、不能重复的。因此，集合就像是一个没有value的字典。

集合相关方法

```
>>> aset = set('abc')
>>> bset = set('bcd')
>>> aset
{'a', 'c', 'b'}
>>> bset
{'c', 'd', 'b'}
>>> aset & bset   # 交集
{'c', 'b'}
>>> aset | bset   # 并集
{'c', 'b', 'a', 'd'}
>>> aset - bset   # 差补，在aset中有，bset中没有
{'a'}
>>> bset - aset
{'d'}

>>> aset.add('new')   # 将new作为整体添加到集合
>>> aset
{'new', 'a', 'c', 'b'}
>>> aset.update('new')   # 将序列对象中的每个元素逐一添加到集合
>>> aset
{'new', 'c', 'a', 'n', 'w', 'b', 'e'}
>>> aset.update(['hello', 'world'])
>>> aset
{'new', 'c', 'a', 'w', 'n', 'hello', 'world', 'b', 'e'}
>>> aset.remove('new')   # 删除一项
>>> aset
{'c', 'a', 'w', 'n', 'hello', 'world', 'b', 'e'}

# 作为了解的方法
>>> s1 = set('abcde')
>>> s2 = set('bcd')
>>> s1
{'c', 'a', 'b', 'e', 'd'}
>>> s2
{'c', 'd', 'b'}
>>> s1.issuperset(s2)   # s1是s2的超集吗？
True
>>> s2.issubset(s1)     # s2是s1的子集吗？
True
```

```
>>> s1.union(s2)    # s1 | s2
{'c', 'b', 'a', 'e', 'd'}
>>> s1.intersection(s2)    # s1 & s2
{'c', 'd', 'b'}
>>> s1.difference(s2)    # s1 - s2
{'a', 'e'}
```

集合的应用

```
# 去重操作
>>> num_list = [randint(1, 20) for i in range(10)]
>>> num_list
[1, 7, 19, 13, 18, 16, 19, 15, 9, 19]
>>> set(num_list)
{1, 7, 9, 13, 15, 16, 18, 19}
>>> list(set(num_list))
[1, 7, 9, 13, 15, 16, 18, 19]
```

练习：

1. 有两个文件:a.log和b.log
2. 两个文件中有大量重复内容
3. 取出只有在b.log中存在的行

```
[root@room8pc16 day05]# cp /etc/passwd /tmp/mima
[root@room8pc16 day05]# vim /tmp/mima   # 修改mima文件
# 将文件内容放到列表中
>>> with open('/etc/passwd') as f1:
...     aset = set(f1)
>>> with open('/tmp/mima') as f2:
...     bset = set(f2)
>>> bset - aset   # 获取只在mima文件中存在的行
# 把结果写入新文件
>>> with open('/tmp/result.txt', 'w') as f3:
...     f3.writelines(bset - aset)
# cat /tmp/result.txt
```

114-百鸡百钱问题: https://www.jianshu.com/p/a915980dadd0

111-配置IP地址: https://www.jianshu.com/p/436bad220b5d

123-进度条 ： https://www.jianshu.com/p/581885d9bd1c

124-带进度条的文件拷贝：https://www.jianshu.com/p/da813405f582

119-比较文件的差异：https://www.jianshu.com/p/d3d4f6165f64

118-ip地址与10进制数的转换：https://www.jianshu.com/p/6541dca9f414

可以练习的例子：1 – 54