nsd1905_py01_day05

列表

- 列表属于容器、可变、序列

```
>>> alist = [13, 2, 356, 23]
>>> alist[-1] = 100
>>> alist
[13, 2, 356, 100]
>>> alist[1:3]
[2, 356]
>>> alist[1:3] = [10, 20, 30]
>>> alist
[13, 10, 20, 30, 100]
```

字符串的方法

```
# 追加对象到列表的结尾
>>> alist.append(200)
# 统计列表中有几个10
>>> alist.count(10)
# 在下标为1的位置插入100
>>> alist.insert(1, 100)
# 反转列表
>>> alist.reverse()
# 清空列表
>>> alist.clear()
# extend用于将序列对象中的内容合并到列表
>>> alist.extend('abc')
>>> alist
['a', 'b', 'c']
>>> alist.append('abc')
>>> alist
['a', 'b', 'c', 'abc']
>>> alist.extend(['bob', 'alice'])
>>> alist
['a', 'b', 'c', 'abc', 'bob', 'alice']
# pop默认弹出最后一项
>>> alist.pop()
# 弹出下标为2的项
>>> alist.pop(2)
# 升序排序
>>> alist.sort()
# 降序排序
>>> alist.sort(reverse=True)
# 把alist的值复制给blist
>>> blist = alist.copy()
```

```
# 返回abc的下标
>>> alist.index('abc')
# 删除abc
>>> alist.remove('abc')
```

元组

- 元组是容器、不可变、顺序
- 元组相当于是静态的列表

```
>>> atu = (10, 35, 20, 1, 8)
>>> atu.
atu.count(  atu.index(
>>> atu.count(35)
1
>>> atu.index(20)
2
```

- 单元素元组必须有逗号，否则不是元组

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

列表练习

1. 思考程序的运行方式

```
# python stack.py
(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:
```

```
(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): 0
数据: china
(0) 压栈
(1) 出栈
(2) 查询
(3) 退出
请选择(0/1/2/3): 2
['hello', 'china']
(0) 压栈
(1) 出栈
(2) 查询
(3) 退出
请选择(0/1/2/3): 1
从栈出弹出了: china
(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
从栈中弹出了: hello
(0) 压栈
(1) 出栈
(2) 查询
(3) 退出
请选择(0/1/2/3): 1
```

```
栈已经为空
(0) 压栈
(1) 出栈
(2) 查询
(3) 退出
请选择(0/1/2/3): 3
Bye-bye
```

2. 分析有哪些功能，编写功能函数
3. 编写主程序代码

```
def push_it():
    print('push')

def pop_it():
    print('pop')

def view_it():
    print('view')

def show_menu():


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

4. 编写具体的函数语句块

字典

- 字典属于容器、可变、映射类型
- 字典的key不能重复，value可重复
- 字典的key必须是不可变对象

```
>>> adict = dict(['ab', ['name', 'bob'], ('age', 20)])
>>> adict
{'a': 'b', 'name': 'bob', 'age': 20}
>>> bdict = {}.fromkeys(['bob', 'alice', 'zs', 'ls'], 20)
>>> bdict
{'bob': 20, 'alice': 20, 'zs': 20, 'ls': 20}

>>> adict
{'a': 'b', 'name': 'bob', 'age': 20}

>>> for key in adict:
...   print(key, adict[key])

>>> '%s is %s years old' % (adict['name'], adict['age'])
'bob is 20 years old'
>>> '%(name)s is %(age)s years old' % adict
```

```
'bob is 20 years old'

# 更新字典。字典中有key为修改，没有为添加
>>> adict['email'] = 'bob@tedu.cn'
>>> adict['age'] = 22
# 判断字典有没有某个key
>>> 'email' in adict
True
>>> 22 in adict
False
```

字典的方法

```
>>> adict.get('name')  # 取出value
'bob'
>>> print(adict.get('qq'))  # key不存在，默认返回None
None
>>> adict.get('name', 'not found')
'bob'
>>> adict.get('phone', '110')  # value不存在，返回指定值
'110'
>>> adict.keys()
dict_keys(['a', 'name', 'age', 'email'])
>>> list(adict.keys())  # 返回所有的key
['a', 'name', 'age', 'email']
>>> adict.values()   # 返回所有的value
>>> adict.items()   # 返回(key, val)组成的列表

>>> adict.pop('a')  # 弹出key为a的项目
>>> adict.popitem()  # 弹出某个项目
# 合并字典
>>> adict.update({'name': 'tom', 'phone': '12363456'})
```

集合

- 集合是一个数学上的概念，它由不同元素构成
- 集合中的元素必须是不可变的
- 集合无序
- 集合就像是一个无值的字典

```
>>> s1 = set('abc')
>>> s2 = set('bccd')
>>> s1
{'a', 'b', 'c'}
>>> s2
{'d', 'b', 'c'}
>>> 'a' in s1
True
>>> len(s1)
```

```
3
>>> for i in s1:
...   print(i)
# 取交集
>>> s1 & s2
{'b', 'c'}
# 取并集
>>> s1 | s2
{'b', 'a', 'd', 'c'}
# 取差补，s1中有，s2中没有的
>>> s1 - s2
{'a'}
```

集合的方法

```
>>> s1.add('new')
>>> s1
{'a', 'new', 'b', 'c'}
>>> s1.update('new')
>>> s1
{'e', 'new', 'w', 'n', 'b', 'a', 'c'}
>>> s1.remove('b')
>>> s3 = set('abcde')
>>> s4 = set('bcd')
>>> s3.issuperset(s4)  # s3是s4的超集吗？
True
>>> s4.issubset(s3)   # s4是s3的子集吗？
True
>>> s3.union(s4)   # s3 | s4
{'b', 'e', 'a', 'd', 'c'}
>>> s3.intersection(s4)   # s3 & s4
{'d', 'b', 'c'}
>>> s3.difference(s4)  # s3 - s4
{'a', 'e'}
```

集合的应用

```
# 去重
>>> nums = [randint(1, 30) for i in range(20)]
>>> nums
[1, 29, 1, 4, 6, 30, 5, 21, 10, 13, 4, 4, 19, 17, 10, 10, 28, 21, 4, 17]
>>> set(nums)
{1, 4, 5, 6, 10, 13, 17, 19, 21, 28, 29, 30}
>>> list(set(nums))
[1, 4, 5, 6, 10, 13, 17, 19, 21, 28, 29, 30]
```

文件去重，找出mima中有，passwd中没有的行

```
(nsd1905) [root@room8pc16 day05]# cp /etc/passwd /tmp/mima
# 修改mima文件，使之与passwd不同
>>> with open('/etc/passwd') as f1:
```

```
...     s1 = set(f1)
...
>>> with open('/tmp/mima') as f2:
...     s2 = set(f2)
...
>>> s2 - s1
{'admin:x:3:4:adm:/var/adm:/sbin/nologin\n', 'hello world!\n'}
>>> with open('/tmp/result.txt', 'w') as f3:
...     f3.writelines(s2 - s1)
(nsd1905) [root@room8pc16 day05]# cat /tmp/result.txt
```