nsd1907_py01_day05

列表

- 属于容器、可变、序列类型

```
>>> import random
# 创建具有5个随机数字的列表
>>> alist = [random.randint(1, 100) for i in range(5)]
>>> alist
[85, 21, 95, 85, 30]
# 赋值
>>> alist[-1] = 21
>>> alist
[85, 21, 95, 85, 21]
>>> alist[1:3] = [1, 3, 5, 7, 9]
>>> alist
[85, 1, 3, 5, 7, 9, 85, 21]

>>> alist.append(5)  # 追加
>>> alist.insert(0, 5)   # 在下标为0的位置插入5
>>> alist.count(5)   # 统计5出现的次数
3
>>> alist.index(5)   # 返回数字5的下标
0

>>> alist.append([1, 2])   # 将列表作为整体加入alist
>>> alist
[5, 85, 1, 3, 5, 7, 9, 85, 21, 5, [1, 2]]
>>> alist.extend([1, 2])   # 将列表合并到alist
>>> alist
[5, 85, 1, 3, 5, 7, 9, 85, 21, 5, [1, 2], 1, 2]
>>> alist.pop()   # 弹出列表中最后一项
2
>>> alist.pop(-2)   # 弹出下标为-2的项
[1, 2]
>>> alist.remove(85)   # 删除第一个85
>>> alist
[5, 1, 3, 5, 7, 9, 85, 21, 5, 1]
>>> a = alist.pop()   # 将pop的返回值赋值给a
>>> a
1
>>> b = alist.remove(85)   # remove没有返回值，默认返回None
>>> print(b)
None

>>> alist
[5, 1, 3, 5, 7, 9, 21, 5]
>>> alist.reverse()    # 翻转
```

```
>>> alist
[5, 21, 9, 7, 5, 3, 1, 5]
>>> alist.sort()   # 默认升序排列
>>> alist
[1, 3, 5, 5, 5, 7, 9, 21]
>>> alist.sort(reverse=True)  # 降序排列
>>> alist
[21, 9, 7, 5, 5, 5, 3, 1]

>>> blist = alist.copy()   # 将alist的值赋值给blist，但采用不同内存空间
>>> alist
[21, 9, 7, 5, 5, 5, 3, 1]
>>> blist
[21, 9, 7, 5, 5, 5, 3, 1]
>>> blist.clear()   # 清空列表
```

元组

- 属于不可变、容器、序列类型
- 相当于是不可变的列表

```
>>> atuple = (15, 20, 15, 5, 15)
>>> atuple.count(15)   # 统计15出现的次数
3
>>> atuple.index(5)    # 5的下标
3

# 单元素元组，必须在结尾加逗号，否则创建的不是元组
>>> a = ('hello')
>>> type(a)
<class 'str'>
>>> a
'hello'
>>> b = ('hello',)
>>> type(b)
<class 'tuple'>
>>> b
('hello',)
>>> len(b)
1
```

列表练习

1. 程序的运行方式

```
(0) 压栈
(1) 出栈
(2) 查询
(3) 退出
请选择(0/1/2/3): abc
无效的输入，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:
输入为空
(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): 0
数据: world
(0) 压栈
(1) 出栈
(2) 查询
(3) 退出
请选择(0/1/2/3): 2
['hello', 'world']
(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: world
(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
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. 框架

```
def push_it():


def pop_it():


def view_it():


def show_menu():


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

3. 完成每个函数

字典

- 属于容器、可变、映射类型
- 字典的键不能重复
- 字典的key只能是不可变对象

```
>>> dict(['ab', 'cd', 'ef'])
{'a': 'b', 'c': 'd', 'e': 'f'}
>>> dict([('name', 'tom'), ('age', 20)])
{'name': 'tom', 'age': 20}
>>> dict([['name', 'tom'], ['age', 20]])
{'name': 'tom', 'age': 20}
>>> dict(['ab', ['name', 'tom'], ('age', 20)])
{'a': 'b', 'name': 'tom', 'age': 20}
>>> {}.fromkeys(('tom', 'jerry', 'bob', 'alice'), 7)
{'tom': 7, 'jerry': 7, 'bob': 7, 'alice': 7}

>>> adict = dict(['ab', ['name', 'tom'], ('age', 20)])
>>> adict
{'a': 'b', 'name': 'tom', 'age': 20}
>>> 'tom' in adict
False
>>> 'name' in adict
True
>>> for key in adict:
...   print(key, adict[key])
...
a b
```

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

>>> adict['age'] = 22   # 已有键，则更新值
>>> adict
{'a': 'b', 'name': 'tom', 'age': 22}
>>> adict['email'] = 'tom@tedu.cn'  # 没有键，则增加一项
>>> adict
{'a': 'b', 'name': 'tom', 'age': 22, 'email': 'tom@tedu.cn'}
>>> del adict['a']  # 使用关键字del删除字典的一项
>>> adict
{'name': 'tom', 'age': 22, 'email': 'tom@tedu.cn'}
# 元组可以作为key，列表不行，因为列表可变
>>> {(10, 15): 'tom'}
{(10, 15): 'tom'}


>>> adict.get('name')   # 通过key取出value
'tom'
>>> print(adict.get('phone'))  # key不存在，默认返回None
None
>>> adict.get('name', 'not found')    # key存在，返回Value
'tom'
>>> adict.get('phone', 'not found')   # key不存在，返回指定值
'not found'
>>> adict.keys()  # 返回所有的key
dict_keys(['name', 'age', 'email'])
>>> adict.values()  # 返回所有的value
dict_values(['tom', 22, 'tom@tedu.cn'])
>>> adict.items()   # 返回key-val对
dict_items([('name', 'tom'), ('age', 22), ('email', 'tom@tedu.cn')])
>>> list(adict.items())
[('name', 'tom'), ('age', 22), ('email', 'tom@tedu.cn')]

>>> adict.pop('email')  # 弹出key为email的项
'tom@tedu.cn'
>>> adict.update({'phone': '15088997788'})
>>> adict
{'name': 'tom', 'age': 22, 'phone': '15088997788'}
```

字典练习

```
(0) 注册
(1) 登陆
(2) 退出
请选择(0/1/2): 0
用户名: tom
```

```
密码: 123
(0) 注册
(1) 登陆
(2) 退出
请选择(0/1/2): 0
用户名:
用户名不能为空
(0) 注册
(1) 登陆
(2) 退出
请选择(0/1/2): 0
用户名: tom
用户已存在
(0) 注册
(1) 登陆
(2) 退出
请选择(0/1/2): 1
用户名: jerry
密码:
登陆失败
(0) 注册
(1) 登陆
(2) 退出
请选择(0/1/2): 1
用户名: tom
密码:
登陆成功
(0) 注册
(1) 登陆
(2) 退出
请选择(0/1/2): 2
Bye-bye
```

集合

- 集合：由不同的元素构成
- 元素不能重复
- 元素必须是不可变对象
- 元素没有顺序
- 集合就像是一个无值的字典
- 集合分为可变集合set和不可变集合frozenset

```
>>> aset = set('abc')
>>> aset
{'b', 'a', 'c'}
>>> set(['tom', 'jerry', 'bob'])
{'tom', 'bob', 'jerry'}
>>> bset = set('bcd')
>>> bset
{'b', 'd', 'c'}
>>> 'a' in aset
True
```

```
>>> len(aset)
3
>>> for ch in aset:
...   print(ch)
...
b
a
c

>>> aset & bset   # 交集
{'b', 'c'}
>>> aset | bset   # 并集
{'d', 'a', 'b', 'c'}
>>> aset - bset   # 差补，aset中有bset中没有
{'a'}

>>> cset.add('hello')
>>> cset
{'hello', 'b', 'a', 'c'}
>>> cset.update('hello')
>>> cset
{'o', 'b', 'l', 'hello', 'a', 'h', 'e', 'c'}
>>> cset.remove('e')

>>> cset = set('abcde')
>>> dset = set('bcd')
>>> cset.issuperset(dset)   # cset是dset的超集吗？
True
>>> dset.issubset(cset)     # dset是cset的子集吗？
True
>>> aset.union(bset)   # aset | bset
{'d', 'a', 'b', 'c'}
>>> aset.intersection(bset)  # aset & bset
{'b', 'c'}
>>> aset.difference(bset)   # aset - bset
{'a'}
```

集合的应用

```
# 去重
>>> from random import randint
>>> nums = [randint(1, 10) for i in range(20)]
>>> nums
[19, 12, 6, 2, 17, 18, 12, 15, 1, 10, 9, 5, 10, 9, 8, 14, 1, 5, 2, 1]
>>> set(nums)
{1, 2, 5, 6, 8, 9, 10, 12, 14, 15, 17, 18, 19}
>>> list(set(nums))
[1, 2, 5, 6, 8, 9, 10, 12, 14, 15, 17, 18, 19]
```

5.log -> url -> url5

6.log -> url -> url6

```
(nsd1907) [root@room8pc16 day05]# cp /etc/passwd /tmp/mima
(nsd1907) [root@room8pc16 day05]# cp /etc/passwd /tmp/mima2
# 修改mima2，使之与mima不一样
(nsd1907) [root@room8pc16 day05]# vim /tmp/mima2
>>> with open('/tmp/mima') as f1:
...   aset = set(f1)
>>> with open('/tmp/mima2') as f2:
...   bset = set(f2)
>>> bset - aset  # bset中有aset中没有的行
>>> with open('/tmp/result.txt', 'w') as f3:
...   f3.writelines(bset - aset)
```