nsd1903_py01_day05

列表

```
>>> alist = [10, 5, 32, 1, 8, 20]
>>> alist[0] = 100
>>> alist[1:3] = [45, 88, 12, 24]
>>> alist
[100, 45, 88, 12, 24, 1, 8, 20]
>>> alist[2:2]
[]
>>> alist[2:2] = [12, 8]
>>> alist
[100, 45, 12, 8, 88, 12, 24, 1, 8, 20]

# 列表的方法
>>> alist.append(12)   # 追加
>>> alist.extend([55, 32, 1])   # 加入多项
>>> alist.remove(8)   # 移除第一个8
>>> alist.index(12)   # 返回第一个12的下标
>>> alist.reverse()   # 翻转
>>> blist = alist.copy()   # 将alist的值copy后，赋值给blist
>>> alist.insert(2, 88)   # 在下标为2的位置插入88
>>> alist.sort()   # 升序排列
>>> alist.sort(reverse=True)   # 降序
>>> alist.count(12)   # 统计12出现的次数
>>> alist.pop()   # 将最后一项弹出
>>> alist.pop(2)   # 弹出下标为2的项目
```

元组

相当于是静态的列表。

```
>>> atuple = (10, 20, 15)
>>> atuple.count(10)   # 统计10出现的次数
1
>>> atuple.index(15)   # 获取15的下标
2
>>> a = (10)
>>> type(a)
<class 'int'>
>>> a
10
>>> b = (10,)   # 单元素元组必须有逗号
>>> type(b)
<class 'tuple'>
>>> len(b)
1
```

练习：模拟栈结构

1. 发呆。思考程序的运作方式

```
(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): 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: world
(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. 思考程序有哪些功能，将功能定义成函数

```
def push_it():


def pop_it():

def view_it():

def show_menu():


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

字典

- 映射、可变、容器
- 字典key不能重复
- 为字典赋值时，key存在则修改，不存在则新建
- 字典的key必须是不可变对象

```
>>> dict(['ab', ['name', 'bob'], ('age', 20)])
{'a': 'b', 'name': 'bob', 'age': 20}
>>> dict([('name', 'tom'), ('age', 20), ('mail', 'tom@tedu.cn')])
{'name': 'tom', 'age': 20, 'mail': 'tom@tedu.cn'}

>>> {}.fromkeys(['tom', 'jerry', 'bob'], 20)
{'tom': 20, 'jerry': 20, 'bob': 20}

>>> info = dict([('name', 'tom'), ('age', 20), ('mail', 'tom@tedu.cn')])
>>> info
{'name': 'tom', 'age': 20, 'mail': 'tom@tedu.cn'}
>>> for key in info:
...   print(key, info[key])
...
name tom
age 20
mail tom@tedu.cn

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

```
>>> info['age'] = 22
>>> info['phone'] = '15012345678'
>>> info
{'name': 'tom', 'age': 22, 'mail': 'tom@tedu.cn', 'phone': '15012345678'}
>>> 'tom' in info
False
>>> 'name' in info
True
>>> len(info)
4

>>> info.keys()   # 取出所有的key
dict_keys(['name', 'age', 'mail', 'phone'])
>>> info.values()  # 取出所有的value
dict_values(['tom', 22, 'tom@tedu.cn', '15012345678'])
>>> info.items()   # 取出键值对
dict_items([('name', 'tom'), ('age', 22), ('mail', 'tom@tedu.cn'), ('phone',
'15012345678')])
>>> list(info.items())   # 将键值对转成列表
[('name', 'tom'), ('age', 22), ('mail', 'tom@tedu.cn'), ('phone', '15012345678')]

>>> info.popitem()   # 弹出字典中的一项
('phone', '15012345678')
>>> info.pop('mail')   # 弹出key是mail的项目
>>> info.update({'mail': 'tom@qq.com'})   # 更新字典

>>> {(1, 2): 'tom'}   # 元组作为key
{(1, 2): 'tom'}
>>> {[1, 2]: 'tom'}   # 列表是可变的，不能成为key，报错

# **字典中最生要的方法**
>>> info.get('name')   # 在字典中取出key为name的值
'tom'
>>> print(info.get('phone'))   # key不在字典中，默认返回None
None
>>> print(info.get('phone', 'not found'))  # 找不到key返回not found
not found
>>> info.get('name', 'not found')
'tom'
>>> print(info.get('phone', '110'))
110
```

集合

集合是一个数学上的概念

- 它由不同元素构成
- 集合元素必须是不可变对象
- 集合是无序的
- 集合就像是一个无值的字典
- 集合分成可变集合和不可变集合

```
```

```
>>> frozenset('abc')  # 不可变集合，集合一旦创建，不能做增删改操作
frozenset({'b', 'a', 'c'})

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

>>> aset & bset   # 交集，两个集合中都有的元素
{'b', 'c'}
>>> aset | bset   # 并集，两个集合中所有的元素
{'b', 'd', 'a', 'c'}
>>> aset - bset   # 差补，aset中有，bset中无
{'a'}
>>> bset - aset
{'d'}
>>> aset
{'b', 'a', 'c'}
>>> len(aset)
3
>>> 'a' in aset
True
>>> for ch in aset:
...   print(ch)

>>> aset.add(10)  # 向集合中加入新项目
>>> aset.pop()   # 弹出任意一个元素
'b'
>>> aset.remove(10)   # 删除指定的元素
>>> aset.update(['tom', 'jerry', 'bob'])
>>> aset
{'tom', 'a', 'jerry', 'bob', 'c'}

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

# 经常使用集合实现去重操作
>>> from random import randint
```

```
>>> nums = [randint(1, 20) for i in range(20)]
>>> nums
[6, 12, 18, 19, 1, 1, 16, 15, 5, 6, 18, 19, 14, 11, 17, 13, 2, 5, 20, 16]
>>> result = []
>>> for i in nums:
...   if i not in result:
...     result.append(i)
>>> result
[6, 12, 18, 19, 1, 16, 15, 5, 14, 11, 17, 13, 2, 20]
>>> list(set(nums))
[1, 2, 5, 6, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
```

url1.txt : 第一天用户访问过的url

url2.txt：第二天用户访问过的url

需求：找出第二天新的url

```
(nsd1903) [root@room8pc16 day05]# cp /etc/passwd /tmp/
(nsd1903) [root@room8pc16 day05]# cp /etc/passwd /tmp/mima
# 修改mima，使它与/tmp/passwd有所区别
>>> with open('/tmp/passwd') as f1:
...   set1 = set(f1)   # set1中的元素是文件的每一行

>>> with open('/tmp/mima') as f2:
...   set2 = set(f2)
>>> set2 - set1
{'how are you?\n', 'hello world\n'}

>>> with open('/tmp/result.txt', 'w') as f3:
...   f3.writelines(set2 - set1)
```

练习：

- 模拟unix2dos程序
  - windows系统把\r\n作为行结束标志
  - 非windows系统把\n作为行结束标志

python l2w.py userdb.py =>生成新文件，已经具有windows的换行符