nsd1906_py01_day05

列表

- 属于：容器、可变、序列

```
>>> alist = [10, 8, 20, 365, 23, 4]
>>> alist[0] = 100
>>> alist
[100, 8, 20, 365, 23, 4]
>>> alist[1:3]
[8, 20]
>>> alist[1:3] = [9, 6, 9, 5, 200]
>>> alist
[100, 9, 6, 9, 5, 200, 365, 23, 4]

# 列表方法
>>> alist.append(1000)  # 追加
>>> alist.append((1, 2))  # 把元组追加到列表
>>> alist
[100, 9, 6, 9, 5, 200, 365, 23, 4, 1000, (1, 2)]
>>> alist.extend((1, 2))  # 将序列对象中的每一项作为列表项加入
>>> alist
[100, 9, 6, 9, 5, 200, 365, 23, 4, 1000, (1, 2), 1, 2]
>>> alist.remove((1, 2))  # 删除列表中的某一项
>>> alist
[100, 9, 6, 9, 5, 200, 365, 23, 4, 1000, 1, 2]
>>> alist.index(1000)   # 取出1000的下标
9
>>> alist.reverse()     # 反转列表
>>> alist
[2, 1, 1000, 4, 23, 365, 200, 5, 9, 6, 9, 100]

>>> alist.insert(2, 9)  # 在下标为2的位置插入数据9
>>> alist
[2, 1, 9, 1000, 4, 23, 365, 200, 5, 9, 6, 9, 100]
>>> alist.sort()   # 升序排列
>>> alist
[1, 2, 4, 5, 6, 9, 9, 9, 23, 100, 200, 365, 1000]
>>> alist.sort(reverse=True)   # 降序排列
>>> alist
[1000, 365, 200, 100, 23, 9, 9, 9, 6, 5, 4, 2, 1]
>>> alist.count(9)   # 统计9出现的次数
3
>>> alist.pop()    # 默认将最后一个数据弹出
1
>>> alist.pop(2)   # 弹出下标为2的数据
200
```

```
>>> blist = alist.copy()  # 将alist的值拷贝出来，赋值给blist
>>> blist
[1000, 365, 100, 9, 9, 9, 5, 4, 2]
>>> blist.clear()   # 清空列表
>>> blist
[]
>>> alist
[1000, 365, 100, 9, 9, 9, 5, 4, 2]
```

元组

- 相当于是静态的列表

```
>>> atu = (10, 3, 20)
>>> atu.
atu.count(  atu.index(
>>> atu.count(100)
0
>>> atu.index(3)
1
```

- 单元素元组必须加逗号

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

练习：模拟栈

1. 思考程序的运作方式

```
# python stack.py
(0) 压栈
(1) 出栈
(2) 查询
(3) 退出
请选择(0/1/2/3): 压栈
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： 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: tedu
(0) 压栈
(1) 出栈
(2) 查询
(3) 退出
请选择(0/1/2/3): 2
['hello', 'tedu']
(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: tedu
(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. 分析程序有哪些功能，将功能编写成函数
3. 编写主程序代码，按相关的规则调用函数

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

def pop_it():
    print('pop')

def view_it():
    print('view')

def show_menu():

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

字典

- 字典属于：容器、可变、映射
- 字典的key不能重复
- 字典的key必须是不可变类型

```
>>> adict = dict(['ab', ('name', 'bob'), ['age', 20]])
>>> adict
{'a': 'b', 'name': 'bob', 'age': 20}
# 创建具有相同值的字典
>>> bdict = {}.fromkeys(['bob', 'alice', 'tom'], 7)
>>> bdict
{'bob': 7, 'alice': 7, 'tom': 7}

# 访问字典
>>> for key in adict:
...   print(key, adict[key])
...
a b
name bob
age 20
>>> '%(name)s is %(age)d years old' % adict
'bob is 20 years old'

# 赋值时，key在字典里是更新，不在字典里是加新值
>>> adict['age'] = 22
>>> adict['email'] = 'bob@tedu.cn'
>>> 'bob' in adict
False
>>> 'name' in adict  # name是字典的key吗？
```

```
True
>>> len(adict)
4

# 字典最常用的方法是get
>>> adict.get('age')   # 根据key取值
22
>>> adict['qq']
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'qq'
>>> print(adict.get('qq'))  # 字典中无qq这个key，默认返回None
None
>>> adict.get('qq', 'not found')
'not found'
>>> adict.get('age', 'not found')
22

# 取出所有的key
>>> list(adict.keys())
['a', 'name', 'age', 'email']
# 取出所有的value
>>> adict.values()
dict_values(['b', 'bob', 22, 'bob@tedu.cn'])
# 取出key,value对
>>> list(adict.items())
[('a', 'b'), ('name', 'bob'), ('age', 22), ('email', 'bob@tedu.cn')]
# 根据key弹出字典的一项
>>> adict.pop('a')
'b'
# 批量向字典加入数据
>>> adict.update({'qq': '123455678'})
```

集合

- 集合是由不同元素构成的
- 集合元素必须是不可变类型
- 集合没有顺序
- 集合就像是一个无值的字典
- 集合有可变集合和不可变集合

```
# 创建集合
>>> s1 = set('abc')
>>> s2 = set('bcd')
>>> s3 = set(['tom', 'jerry', 'bob', 'alice'])
>>> s1
{'c', 'b', 'a'}
>>> s2
{'c', 'b', 'd'}
>>> s3
{'alice', 'tom', 'jerry', 'bob'}
```

```
>>> set('hello')
{'l', 'o', 'h', 'e'}

>>> len(s3)
4
>>> for name in s3:
...   print(name)
...
alice
tom
jerry
bob
>>> 'tom' in s3
True
>>> 'zhangsan' in s3
False
>>> 'zhangsan' not in s3
True

# 交集，两个集合中都包含的元素
>>> s1 & s2
{'c', 'b'}
# 并集，两个集合中全部的元素
>>> s1 | s2
{'c', 'd', 'a', 'b'}
# 差补，s1中有，s2中没有
>>> s1 - s2
{'a'}

# 创建空集合
>>> s3 = set()
# 添加元素
>>> s3.add(10)
>>> s3.add([20, 30])   # 错误，因为列表是可变的
>>> s3.update([20, 30])
>>> s3
{10, 20, 30}
# 删除元素
>>> s3.remove(30)
>>> s3
{10, 20}
>>> s4 = set((10, 20, 30, 40))
>>> s4
{40, 10, 20, 30}

# s3是s4的字集吗？
>>> s3.issubset(s4)
True
# s4是s3的超集吗？
>>> s4.issuperset(s3)
True

>>> s1.intersection(s2)   # s1 & s2
```

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

集合的应用

```
# 去重
>>> from random import randint
>>> nums = [randint(1, 20) for i in range(20)]
>>> nums
[9, 7, 13, 5, 2, 10, 14, 12, 18, 9, 12, 19, 16, 1, 18, 1, 11, 9, 20, 5]
>>> set(nums)
{1, 2, 5, 7, 9, 10, 11, 12, 13, 14, 16, 18, 19, 20}
>>> list(set(nums))
[1, 2, 5, 7, 9, 10, 11, 12, 13, 14, 16, 18, 19, 20]
```

取出第2个文件中有，第1个文件中没有的行

```
(nsd1906) [root@room8pc16 day05]# cp /etc/passwd /tmp/mima1
(nsd1906) [root@room8pc16 day05]# cp /etc/passwd /tmp/mima2
# 修改mima2，使之与mima1有不一样的行，还会出现重复
(nsd1906) [root@room8pc16 day05]# vim /tmp/mima2

>>> with open('/tmp/mima1') as f1:
...   s1 = set(f1)
>>> with open('/tmp/mima2') as f2:
...   s2 = set(f2)
>>> s2 - s1
{'hello world!\n', 'mail:x:8:12:mail:/var/spool/mail:/sbin/nologin/abcd\n'}
>>> with open('/tmp/mima3', 'w') as f3:
...   f3.writelines(s2 - s1)
```