nsd1909-py01-day05

列表

- 属于容器、顺序、可变类型

```
>>> l1 = [10, 20, 30, 10, 15, 8]
>>> len(l1)
6
>>> l1[-1] = 80   # 因为列表可变，所以可以对某个元素重新赋值
>>> l1
[10, 20, 30, 10, 15, 80]
>>> l1[1:3] = [1, 2, 3, 4, 5]   # 将下标为1、2的元素替换
>>> l1
[10, 1, 2, 3, 4, 5, 10, 15, 80]
```

- 列表的方法

```
>>> l1.append(10)  # 向列表尾部追加数据
>>> l1.insert(2, 80)  # 向下标为2的位置插入数字80
>>> l1
[10, 1, 80, 2, 3, 4, 5, 10, 15, 80, 10]
>>> l1.extend([10, 20, 30])   # 批量向列表尾部增加数据
>>> l1
[10, 1, 80, 2, 3, 4, 5, 10, 15, 80, 10, 10, 20, 30]
>>> l1.remove(80)  # 将列表中的第一个80删除
>>> l1.pop()       # 默认弹出列表中最后一项
30
>>> l1.index(80)   # 获取80的下标
8
>>> l1.pop(8)      # 弹出下标为8的数据项
80
##############################
# 方法其实就是函数。 remove没有返回值，它默认返回 None；pop有返回值，返回的是删除项
>>> a = l1.remove(1)  # 把列表中出现的第一个 1删除
>>> print(a)
None
>>> l1
[10, 2, 3, 4, 5, 10, 15, 10, 10, 20]
>>> b = l1.pop(1)     # 把列表中下标为1的项目弹出
>>> print(b)
2
>>> l1
[10, 3, 4, 5, 10, 15, 10, 10, 20]
##############################
>>> l1.index(10, 2)   # 从下标为2的位置开始查找10的下标
4
>>> l1.count(10)      # 统计列表中有几个10
4
>>> l1.sort()         # 升序排列
>>> l1
[3, 4, 5, 10, 10, 10, 10, 15, 20]
>>> l1.reverse()      # 翻转列表
>>> l1
[20, 15, 10, 10, 10, 10, 5, 4, 3]

>>> l3 = l1.copy()    # 将l1的值赋值给l3，它们使用不同的内存地址
>>> l3
[3, 4, 5, 10, 10, 10, 10, 15]
>>> l3.clear()        # 清空l3
>>> l3
[]
>>> l1
[3, 4, 5, 10, 10, 10, 10, 15]
```

元组

- 属于容器、序列、不可变类型
- 单元素元组，数据项后面必须有逗号，否则它不表示元组

```
>>> t1 = (10)   # t1是数字，不是元组
>>> type(t1)
<class 'int'>
>>> t1
10
>>> t2 = (10,)  # t2是元组，长度为1
>>> type(t2)
<class 'tuple'>
>>> len(t2)
1
>>> t2
(10,)
```

- 元组的方法

```
>>> t2.<tab><tab>
t2.count(  t2.index(
```

练习：模拟栈结构

1. 发呆。思考程序的运行方式

```
(0) push
(1) pop
(2) view
(3) quit
Please input your choice(0/1/2/3): push
无效的选择，请重试。
(0) push
(1) pop
(2) view
(3) quit
Please input your choice(0/1/2/3): 2
[]
(0) push
(1) pop
(2) view
(3) quit
Please input your choice(0/1/2/3): 0
data:
没有获取到数据
(0) push
(1) pop
(2) view
(3) quit
Please input your choice(0/1/2/3): 2
[]
(0) push
(1) pop
(2) view
(3) quit
Please input your choice(0/1/2/3): 0
data: hello
(0) push
(1) pop
(2) view
(3) quit
Please input your choice(0/1/2/3): 0
data: nihao
(0) push
(1) pop
(2) view
(3) quit
Please input your choice(0/1/2/3): 2
['hello', 'nihao']
(0) push
(1) pop
(2) view
(3) quit
Please input your choice(0/1/2/3): 1
从栈中，弹出： niaho
(0) push
(1) pop
(2) view
(3) quit
Please input your choice(0/1/2/3): 1
从栈中，弹出： hello

(0) push
```

```
(1) pop
(2) view
(3) quit
Please input your choice(0/1/2/3): 1
空栈
(0) push
(1) pop
(2) view
(3) quit
Please input your choice(0/1/2/3): 2
[]
(0) push
(1) pop
(2) view
(3) quit
Please input your choice(0/1/2/3): 3
Bye-bye
```

2. 分析程序有哪些功能，编写成功能函数

```
def push():
    '用于实现压栈功能'

def pop():
    '用于实现出栈功能'

def view():
    '实现查询功能'

def show_menu():
    '用于实现菜单，根据用户的选择调用相应的函数 '
```

3. 编写主程序代码

```
def push():
    '用于实现压栈功能'

def pop():
    '用于实现出栈功能'

def view():
    '实现查询功能'

def show_menu():
    '用于实现菜单，根据用户的选择调用相应的函数 '

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

4. 编写每个具体的函数

字典

- 属于容器、可变、映射类型

```
>>> d1 = {'name': 'tom', 'age': 20}
>>> d2 = dict(['ab', (10, 20), ['email', 'tom@tedu.cn']])
>>> d2
{'a': 'b', 10: 20, 'email': 'tom@tedu.cn'}
>>> dict([('name', 'tom'), ('age', 20), ('email', 'tom@tedu.cn')])
{'name': 'tom', 'age': 20, 'email': 'tom@tedu.cn'}
# 为字典的每个key设置相同的value
>>> d3 = {}.fromkeys(['tom', 'jerry', 'bob', 'alice'], 20)
>>> d3
{'tom': 20, 'jerry': 20, 'bob': 20, 'alice': 20}
```

- 通过key来访问字典的val

```
>>> len(d1)
2
>>> 'tom' in d1   # 'tom'是字典的key吗？
False
>>> 'name' in d1
True
>>> d1['name']
'tom'
>>> for k in d1:
...   print('%s: %s' % (k, d1[k]))
...
name: tom
age: 20
```

- 字典的Key不能重复。通过key进行赋值时，如果key已在字典中，则更新value，key不在字典中是增加新项

```
>>> d1['age'] = 22
>>> d1['email'] = 'tom@tedu.cn'
>>> d1
{'name': 'tom', 'age': 22, 'email': 'tom@tedu.cn'}
```

- 字典的key必须使用不可变对象

```
>>> d4 = {1: 'aaaa', 'name': 'tom', (10, 20): 'jerry'}
```

- 字典常用方法

```
# get是字典最重要的方法，必须熟练掌握
>>> d1.get('name')       # 在d1中取出name对应的值
'tom'
>>> print(d1.get('qq'))  # d1中没有qq这个key，默认返回None
None
>>> d1.get('age', 'not found')  # 在字典中取出age对应的值，没有age返回not found
22
>>> d1.get('qq', 'not found')  # 在字典中取出qq对应的值，没有qq返回not found
'not found'

>>> d1.keys()     # 返回所有的key
dict_keys(['name', 'age', 'email'])
>>> list(d1.keys())
['name', 'age', 'email']
>>> d1.values()   # 返回所有的value
dict_values(['tom', 22, 'tom@tedu.cn'])
>>> list(d1.values())
['tom', 22, 'tom@tedu.cn']
>>> d1.items()    # 返回所有的(key, value)
dict_items([('name', 'tom'), ('age', 22), ('email', 'tom@tedu.cn')])
>>> list(d1.items())
[('name', 'tom'), ('age', 22), ('email', 'tom@tedu.cn')]

>>> d1.pop('email')  # 弹出key为email的项目
'tom@tedu.cn'
>>> d1.popitem()     # 弹出字典中某一项的(key, value)
('age', 22)
>>> d1.update({'aaa': 'bbb', 'ccc': 'ddd'})
>>> d1
{'name': 'tom', 'aaa': 'bbb', 'ccc': 'ddd'}
```

集合

- 数学概念，由不同元素构成的数据类型
- 集合也是容器类型
- 它有可变集合set和不可变集合frozenset
- 集合元素不能重复
- 集合元素必须是不可变对象
- 集合没有顺序
- 集合就像是一个无值的字典，所以也使用{}来表示

```
>>> s1 = {10, 20, 'tom', 'jerry'}
>>> type(s1)
<class 'set'>
>>> len(s1)
4
>>> 10 in s1
True
>>> 'o' in s1
False
>>> for data in s1:
...   print(data)
```

- 集合常用操作

```
>>> s1 = set('abc')
>>> s2 = set('bcd')
>>> s1
{'b', 'c', 'a'}
>>> s2
{'b', 'c', 'd'}
>>> s1 & s2    # 取交集
{'b', 'c'}
>>> s1 | s2    # 取并集
{'c', 'd', 'b', 'a'}
>>> s1 - s2    # 差补，在s1中有但s2中无的元素
{'a'}
>>> s2 - s1
{'d'}
# 集合常用的场景是去重
>>> from random import randint
>>> nums = [randint(1, 50) for i in range(20)]
>>> nums
[28, 31, 27, 28, 43, 11, 2, 13, 4, 50, 4, 13, 12, 39, 23, 42, 35, 26, 19, 6]
>>> set(nums)
{2, 35, 4, 6, 39, 42, 11, 43, 13, 12, 50, 19, 23, 26, 27, 28, 31}
>>> list(set(nums))
[2, 35, 4, 6, 39, 42, 11, 43, 13, 12, 50, 19, 23, 26, 27, 28, 31]
```

- 集合方法

```
>>> s1.intersection(s2)   # s1 & s2
{'b', 'c'}
>>> s1.union(s2)          # s1 | s2
{'c', 'd', 'b', 'a'}
>>> s1.difference(s2)     # s1 - s2
{'a'}
>>> s1.add('hello')       # 增加一个元素
>>> s1
{'b', 'c', 'hello', 'a'}
>>> s1.update(s2)         # 批量增加元素
>>> s1
{'c', 'd', 'b', 'hello', 'a'}
>>> s1.issuperset(s2)     # s1是s2的超集吗？
True
>>> s2.issubset(s1)       # s2是s1的子集吗？
True
```

练习：找到mima文件中有，但是passwd文件中没有的行

```
[root@localhost day05]# cp /etc/passwd /tmp/
[root@localhost day05]# cp /etc/passwd /tmp/mima
[root@localhost day05]# vim /tmp/mima  # 修改，使之与/tmp/passwd有不一样的行
>>> with open('/tmp/mima') as f1:
...     s1 = set(f1)   # 把文件每一行，放到集合中
...
>>> with open('/tmp/passwd') as f2:
...     s2 = set(f2)
>>> s1 - s2   # 取差补，即mima中有，passwd中没有的行
{'how are you?\n', 'hello world!\n'}
>>> with open('/tmp/a.txt', 'w') as f3:
...     f3.writelines(s1 - s2)
[root@localhost day05]# cat /tmp/a.txt
how are you?
hello world!
```