nsd1904_py01_day05

列表

- 列表属于容器、可变、序列类型

```
>>> alist = [15, 8, 30]
>>> len(alist)
3
>>> alist[-1] = 100
>>> alist
[15, 8, 100]
>>> alist[2:2]
[]
>>> alist[2:2] = [10, 20, 30]
>>> alist
[15, 8, 10, 20, 30, 100]
>>> alist[1:3]
[8, 10]
>>> alist[1:3] = [1, 2, 3]
>>> alist
[15, 1, 2, 3, 20, 30, 100]
```

- 列表的常用方法

```
>>> alist.append(200)   # 追加元素
>>> alist.append(20)
>>> alist.count(20)   # 统计20的个数
>>> alist.insert(2, 100)   # 下标为2的位置，插入100
>>> alist.reverse()   # 翻转列表
>>> alist.extend([100, 300])   # 将序列对象合并到列表
>>> alist.index(30)   # 取出30的下标
>>> alist.pop()    # 弹出最后一个元素
>>> alist.pop(4)   # 弹出下标为4的元素
>>> alist.sort()   # 升序排列
>>> alist.sort(reverse=True)   # 降序排列
>>> alist.remove(15)   # 删除列表中的第一个15
>>> clist = alist.copy()  # 将alist的值拷贝给clist，两个列表不是同一内存空间
>>> alist.clear()   # 清空列表
```

元组

- 元组属于容器、不可变、序列类型
- 元组常用的方法

```
>>> atup = (10, 20, 15, 20)
>>> atup.count(20)    # 统计20出现的次数
2
>>> atup.index(20)    # 返回第一个20的下标
1
```

- 创建单元素元组时，必须加上逗号，否则不是元组

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

列表练习：模拟栈结构

1. 思考程序运作方式

```
# python stack.py
(0) push it
(1) pop it
(2) view it
(3) quit
Please inupt your choice(0/1/2/3): 2
[]
(0) push it
(1) pop it
(2) view it
(3) quit
Please inupt your choice(0/1/2/3): 0
item to push: hello
(0) push it
(1) pop it
(2) view it
(3) quit
Please inupt your choice(0/1/2/3): 2
['hello']
(0) push it
(1) pop it
(2) view it
(3) quit
Please inupt your choice(0/1/2/3): 0
item to push: world
(0) push it
```

```
(1) pop it
(2) view it
(3) quit
Please inupt your choice(0/1/2/3): 2
['hello', 'world']
(0) push it
(1) pop it
(2) view it
(3) quit
Please inupt your choice(0/1/2/3): 1
from stack popped: world
(0) push it
(1) pop it
(2) view it
(3) quit
Please inupt your choice(0/1/2/3): 1
from stack popped: hello
(0) push it
(1) pop it
(2) view it
(3) quit
Please inupt your choice(0/1/2/3): 1
empty stack
(0) push it
(1) pop it
(2) view it
(3) quit
Please inupt your choice(0/1/2/3): 2
[]
(0) push it
(1) pop it
(2) view it
(3) quit
Please inupt your choice(0/1/2/3): abc
Invalid choice. Try again.
(0) push it
(1) pop it
(2) view it
(3) quit
Please inupt your choice(0/1/2/3): 3
Bye-bye
```

2. 分析程序有哪些功能，将功能编写为函数

```
def push_it():
    "压栈"

def pop_it():
    "出栈"

def view_it():
    "查询"

def show_menu():
    "显示菜单"
```

3. 程序主体

```
def push_it():
    "压栈"

def pop_it():
    "出栈"

def view_it():
    "查询"

def show_menu():
    "显示菜单"

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

字典

- 字典属于容器、可变、映射类型
- 字典的元素是键-值对(key: val)的形式
- 字典的key必须是不可变对象
- 字典的key不能重复
- 通过key进行赋值是，如果元素在字典中则修改值；如果不在字典中，则新加

```
>>> adict = {'name': 'bob', 'age': 20}
>>> dict(['ab', ['name', 'bob'], ('age', 20)])
{'a': 'b', 'name': 'bob', 'age': 20}
>>> dict(('ab', ['name', 'bob'], ('age', 20)))
{'a': 'b', 'name': 'bob', 'age': 20}
# 创建具有相同值的字典
>>> {}.fromkeys(['tom', 'jerry', 'bob'], 18)
{'tom': 18, 'jerry': 18, 'bob': 18}

>>> adict[(10, 20)] = 'beijing'
>>> adict[[10, 20]] = 'beijing'    # 错误，列表可变，不能作为key
>>> adict['age'] = 22
>>> adict['email'] = 'bob@tedu.cn'

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

>>> "He is %s, he is %s years old. His email is %s" % (adict['name'], adict['age'],
adict['email'])

>>> "He is %(name)s, he is %(age)s years old. His email is %(email)s" % adict
'He is bob, he is 22 years old. His email is bob@tedu.cn'
```

字典常有方法

```
# 最常用的方法是get，用于通过key取出val。字典中没有该项，返回值默认是None，也可以由用户手工指定
>>> adict.get('name')
>>> print(adict.get('qq'))
None
>>> adict.get('qq', 'not found')
'not found'
>>> adict.get('email', 'not found')
'bob@tedu.cn'

>>> adict.keys()   # 取出所有的key
dict_keys(['name', 'age', (10, 20), 'email'])
>>> list(adict.keys())  # 将key转为列表
['name', 'age', (10, 20), 'email']
>>> adict.values()   # 取出所有的val
dict_values(['bob', 22, 'beijing', 'bob@tedu.cn'])
>>> list(adict.values())
['bob', 22, 'beijing', 'bob@tedu.cn']
>>> list(adict.items())
[('name', 'bob'), ('age', 22), ((10, 20), 'beijing'), ('email', 'bob@tedu.cn')]
>>> adict.update({'qq': '123456', 'phone': '110'})  # 合并字典
>>> adict.popitem()   # 随机弹出一项
>>> adict.pop((10, 20))   # 弹出key是(10, 20)的项目
```

字典练习：模拟用户登陆注册

```
# python userdb.py
(0) 注册
(1) 登陆
(2) 退出
请选择(0/1/2/): 0
用户名: tom
密码: abc
(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: tom
密码:
用户名或密码错误
(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/): 3
无效输入，请重试。
(0) 注册
(1) 登陆
(2) 退出
请选择(0/1/2/): 2
Bye-bye
```

集合

- 集合是一个数学上的概念，由不同元素构成
- 集合元素必须是不可变的
- 集合元素不同能重复
- 集合元素没有顺序
- 集合有可变集合set和不可变集合frozenset
- 集合可以看作是一个无值的字典

```
>>> set('abc')
{'c', 'a', 'b'}
>>> set(['aaa', 'bbb', 'ccc'])
{'ccc', 'bbb', 'aaa'}
>>> set(('aaa', 'bbb', 'ccc'))
{'ccc', 'bbb', 'aaa'}

>>> aset = set('hello')
>>> aset
{'e', 'h', 'l', 'o'}
>>> len(aset)
4
>>> for ch in aset:
...   print(ch)
>>> 'h' in aset
True

>>> aset = set('abc')
>>> bset = set('bcd')
>>> aset & bset   # 交集
{'c', 'b'}
>>> aset | bset   # 并集
{'d', 'a', 'b', 'c'}
>>> aset - bset   # 差补，aset有，bset无
{'a'}
>>> bset - aset
{'d'}
```

集合的方法

```
>>> aset.intersection(bset)  # aset & bset
{'c', 'b'}
>>> aset.union(bset)    # aset | bset
{'d', 'a', 'b', 'c'}
>>> aset.difference(bset)   # aset - bset
{'a'}
>>> aset.add('hello')    # 添加一项
>>> aset
{'c', 'a', 'hello', 'b'}
>>> aset.update('hello')   # 添加多项
>>> aset
{'e', 'l', 'a', 'hello', 'b', 'c', 'h', 'o'}
>>> aset.update(['ni', 'hao'])
>>> aset
{'e', 'l', 'a', 'hello', 'b', 'c', 'h', 'ni', 'o', 'hao'}
>>> aset.pop()   # 弹出一项
>>> aset.remove('hello')   # 从集合中删除hello
>>> aset
{'l', 'a', 'b', 'c', 'h', 'ni', 'o', 'hao'}
>>> cset = set('abc')
>>> cset
{'c', 'a', 'b'}
>>> cset.issubset(aset)   # cset是aset的子集吗
True
>>> aset.issuperset(cset)   # aset是cset的超集吗？
True
```

集合的应用

```
# 去重
>>> from random import randint
>>> nums = [randint(1, 20) for i in range(10)]
>>> nums
[13, 12, 10, 1, 10, 13, 14, 18, 4, 12]
# 新建一个列表，遍历nums，如果数字没有在result中，则追加
>>> result = []
>>> for i in nums:
...   if i not in result:
...     result.append(i)
...
>>> result
```

```
[13, 12, 10, 1, 14, 18, 4]
>>> set(nums)
{1, 4, 10, 12, 13, 14, 18}
>>> list(set(nums))
[1, 4, 10, 12, 13, 14, 18]
```

集合练习：

1. 有两个文件:a.log和b.log
2. 两个文件中有大量重复内容
3. 取出只有在b.log中存在的行

```
(nsd1904) [root@room8pc16 day05]# cp /etc/passwd /tmp/mima
# 将mima文件内容复制出重复行，添加或修改一些行
(nsd1904) [root@room8pc16 day05]# vim /tmp/mima
>>> with open('/etc/passwd') as f1:
...   set1 = set(f1)
...
>>> with open('/tmp/mima') as f2:
...   set2 = set(f2)
...
>>> set2 - set1
{'how are you?\n', 'ni hao\n'}

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

(nsd1904) [root@room8pc16 day05]# cat /tmp/sss.txt
how are you?
ni hao
```

127-提取字符串: https://www.jianshu.com/p/af2b3cc6ff8f

118-ip地址与10进制数的转换: https://www.jianshu.com/p/6541dca9f414

114-百鸡百钱问题: https://www.jianshu.com/p/a915980dadd0

113-模拟字符串rstrip用法: https://www.jianshu.com/p/a001a103def8

112-模拟字符串lstrip用法: https://www.jianshu.com/p/3e62945b4600