**张治卓 517111910078**

# 1 String

## 1.1 回文序列

```
def palindrome(str):
    if len(str) == 1:
        return True
    if str[0]!=str[-1]:
        return False
    return palindrome(str[1:-1])

string=input("Please enter the string you want:")
if palindrome(string):
    print("%s is palindrome."%string)
else:
    print("%s is not palindrome."%string)
```

## 1.2 字符串比较

字符串之所以能进行比较，是因为在str类中定义了这些进行比较的方法，我们可以输入dir( str )看看str类的内嵌方法。下面列出一些与字符串比较相关的：
(a) a. eq (b)相当于a == b
(b) ge 相当于a >= b
(c) gt 相当于a > b
(d) le 相当于a <= b
(e) lt 相当于a < b
(f) ne 相当于a != b

## 1.3 字符串的其他一些常规操作

字符串有很多其他操作，如取元、切片、连接。如对于字符串a=” abcdef”
(a) a. getitem (3)对应于a[3]
(b) a. getslice (1,3)相当于a [1:3]
(c) a. add (b)相当于a+b
(d) a. mul (2)相当于a∗2
(e) a. rmul (2)相当于2∗a

## 1.4 字符串的方法

(a) split ()与 rsplit ()的区别是什么？举例说明

> split()从左向右寻找，以某个元素为中心将左右分割成两个元素并放入列表中
> rsplit()从右向左寻找，以某个元素为中心将左右分割成两个元素并放入列表中

```
>> string="abcdfcdeg"
>> string.split("c", 1)
['ab', 'dfcdeg']
>> string.rsplit("c", 1)
['abcdf', 'deg']
```

(b) splitlines ()的主要用途是什么？举例说明

> splitlines()根据换行符（\n）分割并将元素放入列表中

```
>> string = "hello\nworld\nha!"
>> string.splitlines()
['hello', 'world', 'ha!']
```

(c) index(), find (), rfind ()有什么区别？举例说明

> find()从左向右寻找子序列的位置，如存在多个相同子序列只返回第一个查找到的位置，如果子序列不存在返回-1
> rfind()从右向左寻找子序列的位置，如存在多个相同子序列只返回第一个查找到的位置，如果子序列不存在返回-1.
> index()从左向右寻找子序列的位置，如果子序列不存在报错

```
>> string = "hello world!"
>> string.find("l")
2
>> string.rfind("l")
9
>> string.index("l")
2
>> string.index("lo")
3
>> string.find("b")
-1
>> string.index("b")
Traceback (most recent call last):

  File "<pyshell#22>", line 1, in <module>
    string.index("b")
ValueError: substring not found
```

(d) partition (), rpartition ()与 split (), rsplit ()的区别是什么?

> partition()从左向右寻找，以字符串中的某个元素为中心将左右分割共分割成三个元素并放入到元组中
> rpartition()从右向左寻找，以字符串中的某个元素为中心将左右分割共分割成三个元素并放入到元组中
> split()从左向右寻找，以某个元素为中心将左右分割成两个元素并放入列表中
> rsplit()从右向左寻找，以某个元素为中心将左右分割成两个元素并放入列表中
> 说明前两个返回元组，并且保留分割片段；后两个返回列表，不保留分割片段

```
>> string
'hello world!'
>> string.partition("l")
('he', 'l', 'lo world!')
>> string.rpartition("l")
('hello wor', 'l', 'd!')
>> string.split("l")
['he', '', 'o wor', 'd!']
>> string.rsplit("l")
['he', '', 'o wor', 'd!']
>> string.split("l",1)
['he', 'lo world!']
>> string.rsplit("l",1)
['hello wor', 'd!']
```

(e) startswith (), endswith()的主要作用是什么?

> startswith、endswith用来查找开头或结尾条件的元素

```
>> string
'hello world!'
>> string.startswith("he")
True
>> string.endswith("d!")
True
>> string.startswith("He")
False
>> string.endswith("d.")
False
```

(f) isalpha (), isalnum(), isdigit (), islower (), isupper (), istitle ()

> isalpha()如果字符串只包含字母，并且非空返回True;否则返回False
> isalnum()如果字符串只包含字母和数字，并且非空返回True;否则返回False
> isdigit()如果字符串只包含数字字符，并且非空返回True;否则返回False
> islower()如果字符串只由小写字母组组成，并且非空返回True;否则返回False

> isupper()如果字符串只由大写字母组组成，并且非空返回True;否则返回False
> istitle()如果字符串仅包含以大写字母开头、后面都是小写字母的单词返回True;否则返回False

(g) capitalize (), lower (), upper(), swapcase()等方法可对大小写进行调整

> capitalize()首字母大写
> lower()全部小写
> upper()全部大写
> swapcase()大小写反转

(h) 谈谈replace (), translate ()的用法

> translate()只处理单个字符，而且可以同时进行多个替换。
> replace()函数可以替换string中的单个字符，也可以替换连续的字符，但无法生成字符替换映射表
> replace针对的是字符串，而translate针对的是单个字符。

(i) 解释s=’ ghi’; s. join (’ abc’)的返回结果

> ```
> s='ghi'; s. join ('abc')
> 'aghibghic'
> ```
>
> 以abc为连接符，将s插入abc中得到了该结果

# 2 List

## 2.1 寻找和为固定值的数对

有两个列表a和b，分别存储了数量不等的但是列表内部唯一的整数。写一个函数，统计所有值对x ∈ a和y ∈ b，令其和x + y = v，其中v是固定值，也就是f(a, b, v)。
请写出函数，并举例说明。

```
def num_sum(a, b, v):
    for i in a:
        for j in b:
            if (i + j) == v:
                print("(%d,%d)"%(i,j))

a = [1,2,3,4,5,8,9,12,10]
b = [2,3,6,4,5,8,7,9,1,0,-1]

num_sum(a,b,9)
```

测试：

```
================ RESTART: C:/Users/Reckoning/Desktop/test.py ================
(1,8)
(2,7)
(3,6)
(4,5)
(5,4)
(8,1)
(9,0)
(10,-1)
```

## 2.2 列表的排序

我们课上介绍了列表的排序函数list.sort，由3个关键的参数：key是一个函数，指定用函数返回的什么属性进行比较；cmp则是key指定的函数返回的任意两个元素之间进行比较的函数，返回值可为-1, 0, 1；reverse是一个布尔值，表示是否按照降序排列。
现在有一个列表的列表，每个元素都是长度为2的整数列表。现需要对其进行排序，首先按照每个元素列表的和，然后根据第二个值，最后是第一个值，降序排列。请写出你的方法，并举例说明。

```
def cmp(x,y):
    if (x[0]+x[1]) > (y[0]+y[1]):
        return 1
    elif (x[0]+x[1]) < (y[0]+y[1]):
        return -1
    else:

        if x[1] > y[1]:
            return 1
        elif x[1] < y[1]:
            return -1
        else:
            if x[0] > y[0]:
                return 1
            elif x[0] < y[0]:
                return -1
            else:
                return 0

list_test = [[1,2],[2,3],[1,5],[4,6],[7,4],[8,8],[9,12],[-5,15],[0,-5],[5,5]]
list_test.sort(cmp=cmp,reverse=True)
print(list_test)
```

测试：

```
[[9, 12], [8, 8], [7, 4], [-5, 15], [4, 6], [5, 5], [1, 5], [2, 3], [1, 2], [0, -5]]
```