Python for Senior Lesson 5

v1.0.0

2016.10 by David.Yi

复习

- 文件和目录操作之一：文件和目录操作
- 思考一下: 搜索硬盘上指定路径指定类型的文件

本次内容要点

- 列表生成器用法
- 文件和目录操作之二：读写文本文件
- 思考一下

列表生成式

列表生成式是 Python 内置的非常简单却强大的可以用来创建list的方法。

大家都知道，要生成一个这样的 list：[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

可以用 `list(range(1, 11))`

那么如果要生成这样的 list：[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]，应该怎么办呢?

In [1]:

```
# 用循环来生成

l = []
for x in range(1, 11):
    l.append(x * x)

print(l)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

In [9]:

```
# 用列表生成式

l = [ x * x for x in range(1, 11)]
print(l)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

列表生成式用法:

写列表生成式时，把要生成的元素 `x * x` 放到前面，后面跟 `for` 循环，就可以把 list 创建出来，十分有用。

在列表生成式后面还可以加上判断，过滤出结果为偶数的结果

[x * x for x in range(1, 11) if x % 2 == 0]

In [11]:

```
# 在列表生成式后面加上判断，过滤出结果为偶数的结果

[x * x for x in range(1, 11) if x % 2 == ]
```

Out[11]:

```
[4, 16, 36, 64, 100]
```

In [14]:

```
# 可以在列表生成式中使用双重循环
# 输出一对元组，每个数在10以内，且加在一起等于5

l = [(x, y) for x in range(10) for y in range(10) if x + y == 5 if x > y]



print(l)
```

```
[(2, 0), (3, 1), (4, 2), (5, 3), (6, 4), (7, 5), (8, 6), (9, 7)]
```

In [16]:

```
# 改进之前寻找目录下指定字母开头的文件的判断方式
# 修改为使用列表生成式

import os

# 可以指定路径参数，来列出该目录下所有文件
# l = os.listdir('/Users/yijun')

# 可以判断各类情况，比如第一个是大写的 P 字母，用列表生成式的方式，代码精简了很多
l1 =  [l for l in os.listdir('/Users/yijun') if l[0:1] == 'P']
print(l1)
```

```
['Pictures', 'Public']
```

文件和目录操作之二

读写文件是最常见的IO操作。Python内置了读写文件的函数，用法和C是兼容的。

读写文件前，我们先必须了解一下，在磁盘上读写文件的功能都是由操作系统提供的，现代操作系统不允许普通的程序直接操作磁盘，所以，读写文件就是请求操作系统打开一个文件对象，然后，通过操作系统提供的接口从这个文件对象中读取数据，或者把数据写入这个文件对象。

读文件

函数 `open()` 返回 文件对象，通常的用法需要两个参数：`open(filename, mode)`。分别是文件名和打开模式

在做下面的例子前，我们要创建一个 `test.txt` 文件，并且保证其中的内容是如下样式，包含三行内容：

> hello
>
> hi
>
> byebye

文件保存在可以访问的目录，我这里就保存在和 notebook 同样的目录

> 使用 jupyter 可以直接新建 Text File，来完成建立和编辑文本文件

In [5]:

```
import os

# 获得当前路径
cd = os.getcwd()

print(cd)

# 拼接完整文件名
filename = os.path.join('/Users/Feng', 'test.txt')

print(filename)

try:
    # 打开文件
    f = open(filename, 'r')
    print(f.read())
finally:
    if f:
        f.close()
```

```
/Users/yijun/Documents/dev_python/python_beginner/python_senior/less
on5
/Users/yijun/Documents/dev_python/python_beginner/python_senior/less
on5/test.txt
Hello, World!
Hello, Shanghai!
Hello, Beijing!
```

In [14]:

```
# 简化调用方式
# 省却了 try...finally, 会有 with 来自动控制

with open(filename, 'r') as f:
    print(f.read())
```

```
hello
hi
byebye
```

In [23]:

```
with open(filename, 'r') as f:
    lines = f.readlines()

print(type(lines))
print(lines)
```

```
<class 'list'>
['Hello\n', 'byebye']
```

In [24]:

```
for i in lines:
    print(i)
```

```
Hello

byebye
```

In [6]:

```
# 更简单的按行读取文件内容方法
with open(filename, 'r') as f:
    for eachline in f:
        print(eachline)
```

```
Hello, World!

Hello, Shanghai!

Hello, Beijing!
```

写文件

写文件和读文件是一样的，唯一区别是调用 open() 函数时，传入标识符 'w' 或者 'wb' 表示写文本文件或写二进制文件。

r 以读方式打开 w 以写方式打开 a 以追加模式打开（必要时候创建新文件）

In [20]:

```
# 写文件
import os

# 获得当前路径
cd = os.getcwd()

# 拼接完整文件名
filename= os.path.join(cd, 'test2.txt')

# 换行符
br = os.linesep

# 写文件
with open(filename, 'w') as f:
    f.write('Hello, World!' + br)
    f.write('Hello, Shanghai!' + br)
    f.write('Hello, CHINA!' + br)

with open(filename, 'r') as f:
    print(f.read())
```

```
Hello, World!
Hello, Shanghai!
Hello, CHINA!
```

操作系统和文件系统差异处理

`linesep` 文件中分隔行的字符串 `path.sep` 分割文件路径名的字符串 `curdir` 当前工作目录的字符串 `pardir` 当前工作目录的父目录字符串

In [21]:

```
import os

# 换行符会显示不出，  在 macOS 下是：  \n
print(os.linesep)
print(os.path.sep)
print(os.path.curdir)
print(os.path.pardir)
```

```
/
.
..
```

使用 glob 包查找文件

glob 是 python 自己带的一个文件操作相关模块，很简洁，用它可以查找符合自己目的的文件，就类似于 Windows下的文件搜索，而且也支持通配符: `*,?,[]` 这三个通配符，* 代表0个或多个字符，？代表一个字符，[] 匹配指定范围内的字符，如[0-9]匹配数字。

glob 的主要方法也叫 glob，该方法返回所有匹配的文件路径列表，该方法需要一个参数用来指定匹配的路径字符串

In [25]:

```
# 使用 glob 来遍历指定路径下的指定类型文件
import glob

# notebook 写法
glob.glob('/Users/yijun/dev_python/*/*.py')

# IDLE 写法
l = glob.glob('/Users/yijun/dev_python/*/*.py')
for i in l:
    print(i)
```

In [23]:

```
l = glob.glob('/Users/yijun/dev_python/*/e2*.py')
for i in l:
    print(i)
```

In [27]:

```
# python 有趣灵活的变量定义

first, second, *rest = (1,2,3,4,5,6,7,8)
print(first)
print(second)
print(*rest)
```

```
1
2
3 4 5 6 7 8
```

In [28]:

```
# python 交换变量

a, b = 3, 4
a, b = b, a
print(a)
print(b)
```

```
4
3
```

In [ ]:

```

```