实用Python程序设计

郭 炜

微信公众号

微博：http://weibo.com/guoweiofpku

学会程序和算法，走遍天下都不怕！

文件读写

文本文件读写

德国菲森新天鹅堡

文本文件读写

文本文件读写概述

- open函数打开文件，将返回值放入一个变量，例如 f
- 用f.write函数写入文件
- 用f.readlines()函数读取全部文件内容
- 用f.readline()函数读取文件一行
- 用f.close()函数关闭文件
- 用f.read()读取全部文件内容。返回一个字符串，包含文件全部内容

文本文件读写

- 创建文件并写入内容

```
a = open("c:\\tmp\\t.txt","w")
#文件夹c:\tmp 必须事先存在,open不会创建文件夹
#"w"表示写入,用此种方式打开文件，若文件本来存在，就会被覆盖
a.write("good\n")
a.write("好啊\n")
a.close()
```

运行后文件 c:\tmp\t.txt 内容：

```
good
好啊
```

文本文件读写

- 读取现有文件

```
f = open("c:\\tmp\\t.txt","r")   #"r"表示读取
lines = f.readlines() #每一行都带结尾的换行符"\n"
f.close()    #lines是个字符串列表，每个元素是一行
for x in lines:
    print(x,end="")
```

输出：

good

好啊

文本文件读写

- 读取现有文件

```
#不用readlines也行
f = open("c:\\tmp\\t.txt","r",encoding="utf-8")
for x in f:
    print(x,end="")
f.close()
```

文本文件读写

- 用readline读文件中的一行

```
infile = open("c:\\tmp\\t.txt","r")
while True:
    data1 = infile.readline() #data1带结尾的换行符 "\n"。空行也有一个字符，就是"\n"
    if data1 == "":    #此条件满足就代表文件结束
        break
    data1 = data1.strip() #去掉两头空格，包括结尾的 "\n"
    print(data1)
infile.close()
```

文本文件读写

- 如果要读取的文件不存在会引发异常：

```
try:
    f = open("c:\\tmp\\ts.txt","r")
    #若文件不存在，会产生异常，跳到 except后面执行
    lines = f.readlines()
    f.close()
    for x in lines:
        print(x,end="")
except Exception as e:
    print(e)  #>>[Errno 2] No such file or directory: 'c:\\tmp\\ts.txt'
```

文本文件读写

- 添加文件内容：

```
f = open("c:\\tmp\\t.txt","a")
#"a"要打开文件添加内容。若文件本来不存在，就创建文件
f.write("新增行\n")
f.write("ok\n")
f.close()
```

good
好啊
新增行
ok

文件打开模式

"r" ： 文本文件读

"rb"： 二进制文件读

"w" ：文本文件写

"wb" ：二进制文件写

"r+"： 文本文件既读又写

"rb+"： 二进制文件既读又写

文本文件的编码

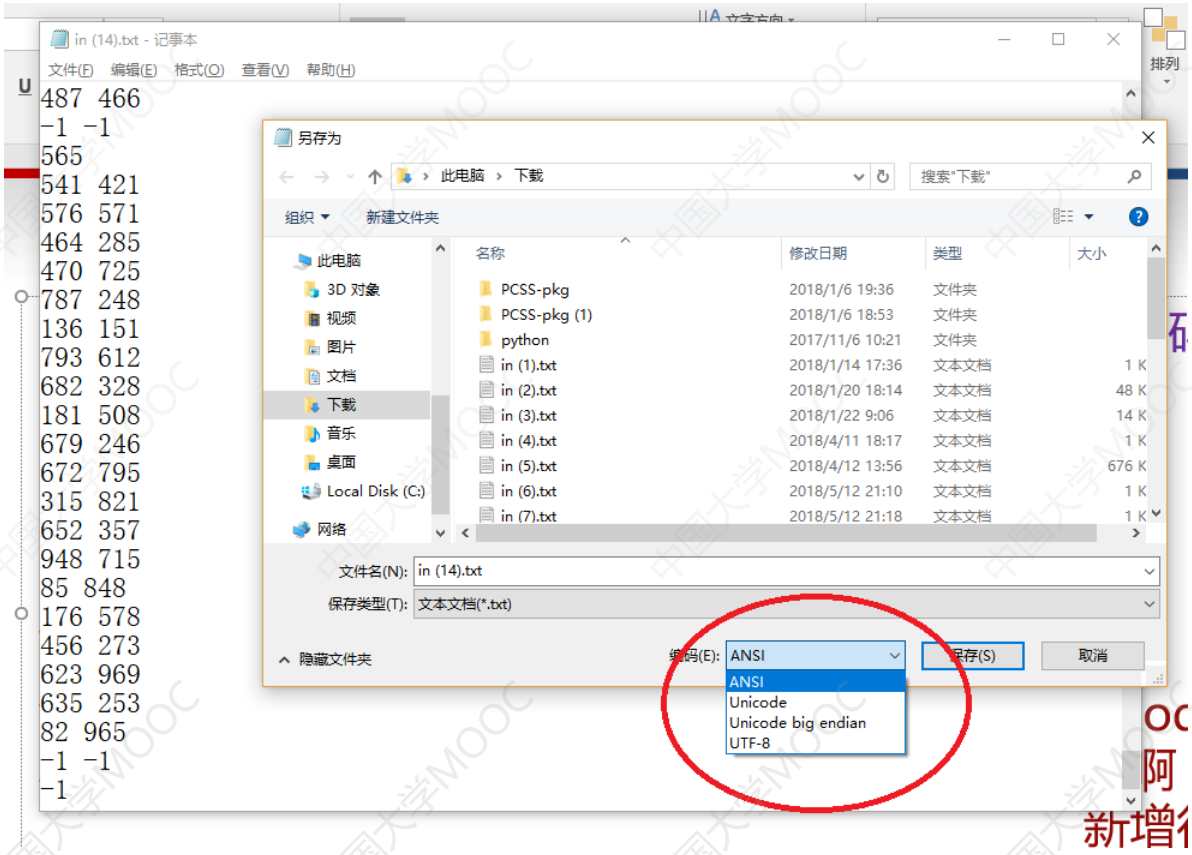
威尼斯

文本文件的编码

- 常见编码有 gbk和utf-8两种。打开文件时如果编码不对，则不能正确读取文件。
- ANSI对应 gbk
- 写入文件时，如果不指定编码，则用操作系统的缺省编码

Windows：gbk，可能从win10开始是 utf-8

Linux,MacOs：utf-8



python程序的编码

.py文件必须存成utf-8格式，才能运行

如果存成 ansi格式，则应该在文件开头写：

```
#coding=gbk
print("你好")
```

文本文件的编码

- 创建文件和读取文件时都可以指定编码

```
outfile = open("c:\\tmp\\t.txt","w",encoding="utf-8")
#若打开文件用于写入时不指定编码，则使用系统缺省编码，win10下也可能是Ansi(gbk)
outfile.write("这很好ok\n")
outfile.write("这ok")
outfile.close()

infile = open("c:\\tmp\\t.txt","r",encoding="utf-8")
lines = infile.readlines()
infile.close()
for x in lines:
    print(x.strip())
```

文本文件的编码

gbk编码的文件，如果用 encoding = "utf-8" 来open，可能会产生异常。

utf-8编码的文件，如果用 encoding = "gbk" 来open，不会产生异常，但是用read或readline,readlines读取到的信息会乱码。

文件的路径

瑞士少女峰

open文件名参数的相对路径形式和绝对路径形式

- **相对路径形式：文件名没有包含盘符**

```
open("readme.txt","r")
```
文件在当前文件夹(当前路径)下

```
open("tmp/readme.txt","r")   "/" 写成 "\\"效果也一样
```
文件在当前文件夹下的tmp文件夹里面

```
open("tmp/test/readme.txt","r")
```
文件在当前文件夹下的tmp文件夹里面的test文件夹下面

```
open("../readme.txt","r")
```
文件在当前文件夹的上一层文件夹里面

```
open("../../readme.txt","r")
```
文件在当前文件夹的上两层文件夹里面

```
open("../tmp2/test/readme.txt","r")
```
文件在当前文件夹的上一层的tmp2文件夹的test文件夹里面
tmp2和当前文件夹是平级的

```
open("/tmp3/test/readme.txt","r")
```
文件在当前盘符的根文件夹下的tmp3/test/里面

open文件名参数的相对路径形式和绝对路径形式

- **绝对路径形式：文件名包含盘符**

```
open("d:/tmp/test/readme.txt" ,"r")
```

路径也叫文件夹，或者目录(path, folder, directory)

Python程序的“当前文件夹 (当前路径，当前目录)”

- 程序运行时，会有一个“当前文件夹”，open打开文件时，如果文件名不是绝对路径形式，则都是相对于当前文件夹的。
- 一般情况下，.py文件所在的文件夹，就是程序运行时的当前文件夹。在Pycharm里面运行程序，就是如此。
- 程序可以获取当前文件夹：

```
import os
print(os.getcwd())                    #os.getcdw()获取当前文件夹

#>>c:\tmp5\test
```

Python程序的“当前文件夹 (当前路径，当前目录)”

- 在命令行方式运行程序时，cmd窗口的当前文件夹，就是程序的当前文件夹，不论程序存在哪里。

c:\tmp5\test\t1.py 如下:

```
import os
print(os.getcwd())
```

则：

命令提示符

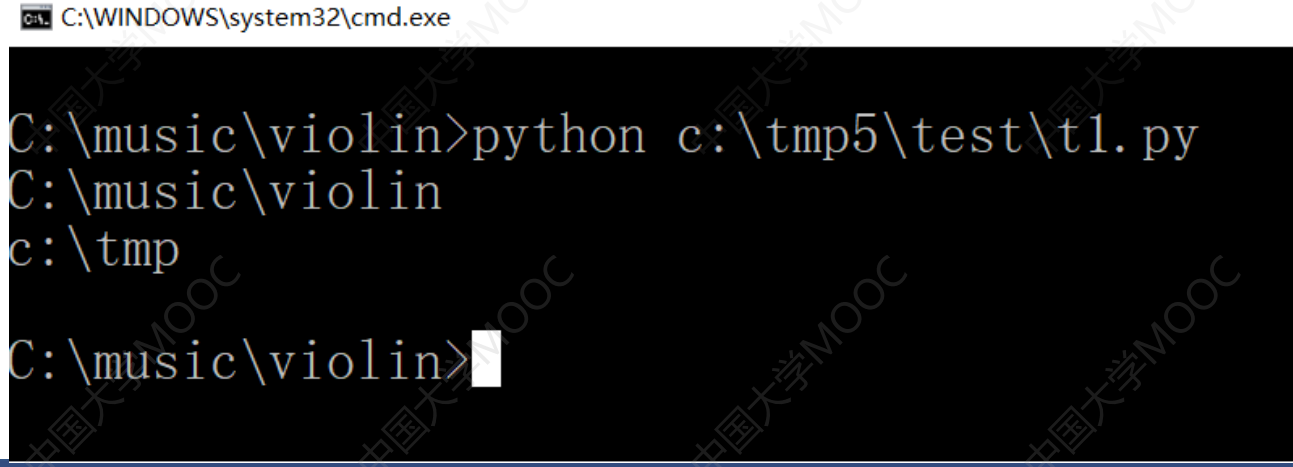
Python程序的“当前文件夹 (当前路径，当前目录)”

- 程序运行期间可以改变当前文件夹

c:\tmp5\test\t1.py 如下:

```
import os
print(os.getcwd())
os.chdir("c:/tmp")
print(os.getcwd())
```

则：



文件夹操作

祁连山

Python的文件夹操作函数

os库和shutil库中有一些函数可以用来操作文件和文件夹(文件夹也称为“目录”)

os.chdir(x)	将程序的当前文件夹设置为x
os.getcwd()	求程序的当前文件夹
os.listdir(x)	返回一个列表，里面是文件夹x中的所有文件和子文件夹的名字
os.mkdir(x)	创建文件夹x
os.path.exists(x)	判断文件或文件夹x是否存在
os.path.getsize(x)	获取文件x的大小（单位:字节）
os.path.isfile(x)	判断x是不是文件
os.remove(x)	删除文件x
os.rmdir(x)	删除文件夹x。x必须是空文件夹才能删除成功
os.rename(x,y)	将文件或文件夹x改名为y。不但可以改名，还可以起到移动文件或文件夹的作用。例如,os.rename("c:/tmp/a","c:/tmp2/b") 可以将文件夹或文件" c:/tmp/a" 移动到"c:/tmp2/"文件夹下面，并改名为b。前提是tmp2必须存在。
shutil.copyfile(x,y)	拷贝文件x到文件y。若y本来就存在，会被覆盖

Python的文件夹操作函数

删除文件夹的递归函数:(不要随便试，用它删除了文件夹就不能恢复)

```
import os
def powerRmDir(path):   #连根删除文件夹path
    lst = os.listdir(path)
    for x in lst:
        actualFileName = path + "/" + x    #x不包括路径,例如a.txt
        if os.path.isfile(actualFileName):#actualFileName是文件
            os.remove(actualFileName)
        else:
            powerRmDir(actualFileName) #actualFileName是文件夹
    os.rmdir(path)

powerRmDir("c:/tmp/ttt")
powerRmDir("tmp/ttt")   #删除当前文件夹下的tmp文件夹下的ttt文件夹
```

Python的文件夹操作函数

获取文件夹总大小的递归函数

```
import os
def getTotalSize(path):
    total = 0
    lst = os.listdir(path)
    for x in lst:
        actualFileName = path + "/" + x   #x不包括路径
        if os.path.isfile(actualFileName):
            total += os.path.getsize(actualFileName)
        else:
            total += getTotalSize(actualFileName)
    return total
```

命令行参数

冰岛杰古沙龙冰河湖

以命令行方式运行python程序

每次运行Python程序，都要从Pycharm里运行，显然不方便。
因此有时需要以命令行方式（命令脚本方式)运行python程序

具体做法：

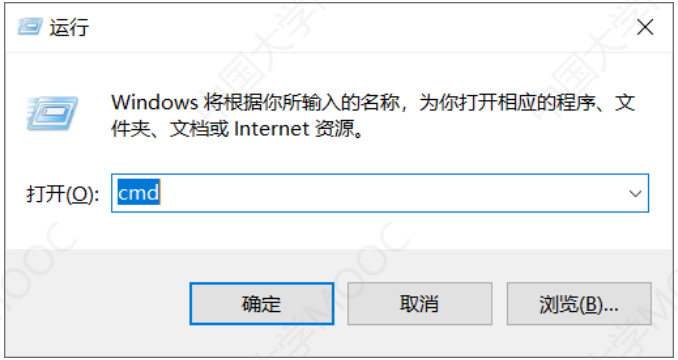
在命令行窗口（mac叫“终端”）敲：

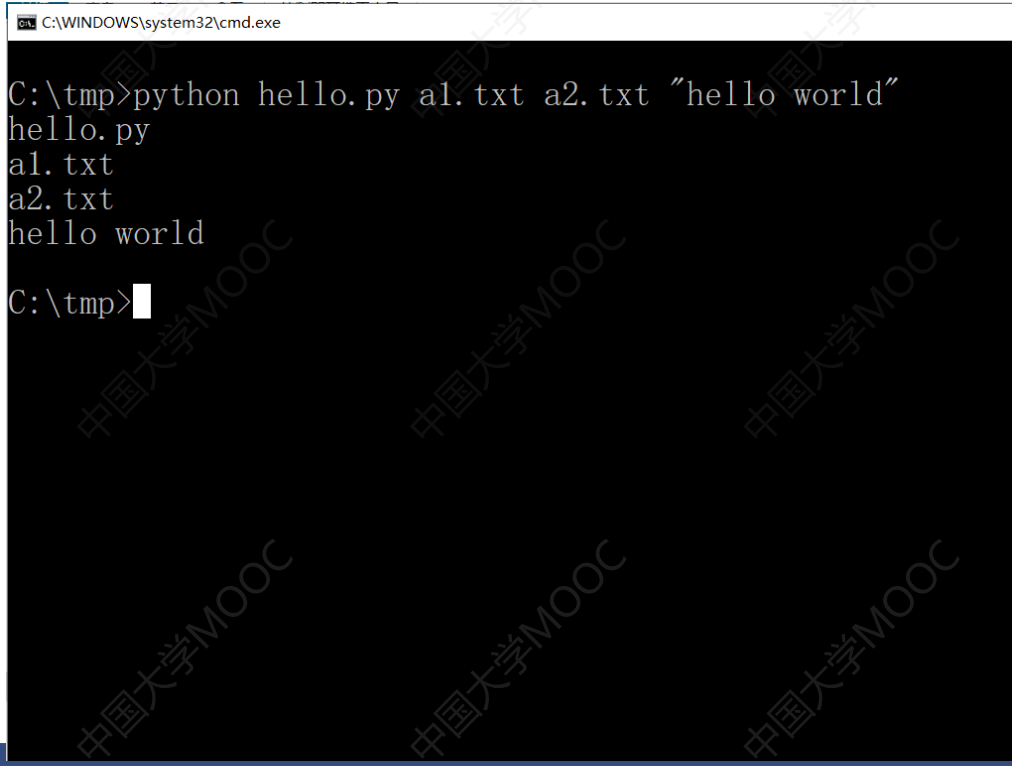
```
python xxx.py
```

就能运行xxx.py

以命令行方式运行python程序：

Windows下，Win+R 键，可以弹出左边“运行”窗口，敲"cmd"确定，就能弹出右边cmd窗口(命令行窗口)





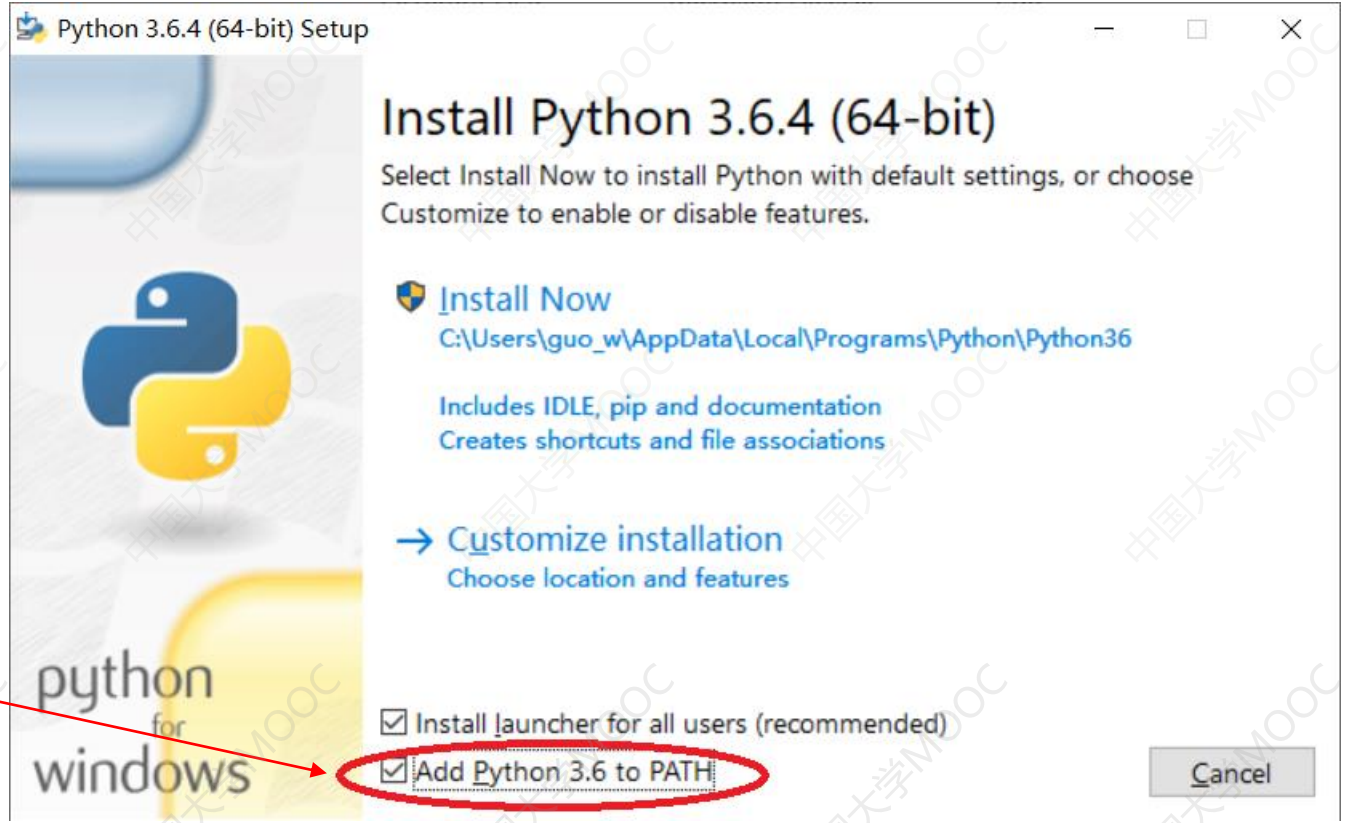
Mac上相应操作，是从launchPad里面启动“终端”

在命令行窗口，敲
python XXX.py就可以运行XXX.py

安装Python

要在命令行方式启动python，就要在安装的时候勾上

不行就卸载重装



命令行参数

如果编写了一个程序 hello.py，功能是合并两个文件

希望在命令行敲

```
python hello.py a1.txt a2.txt
```

就能完成把a2.txt合并到a1.txt上面。

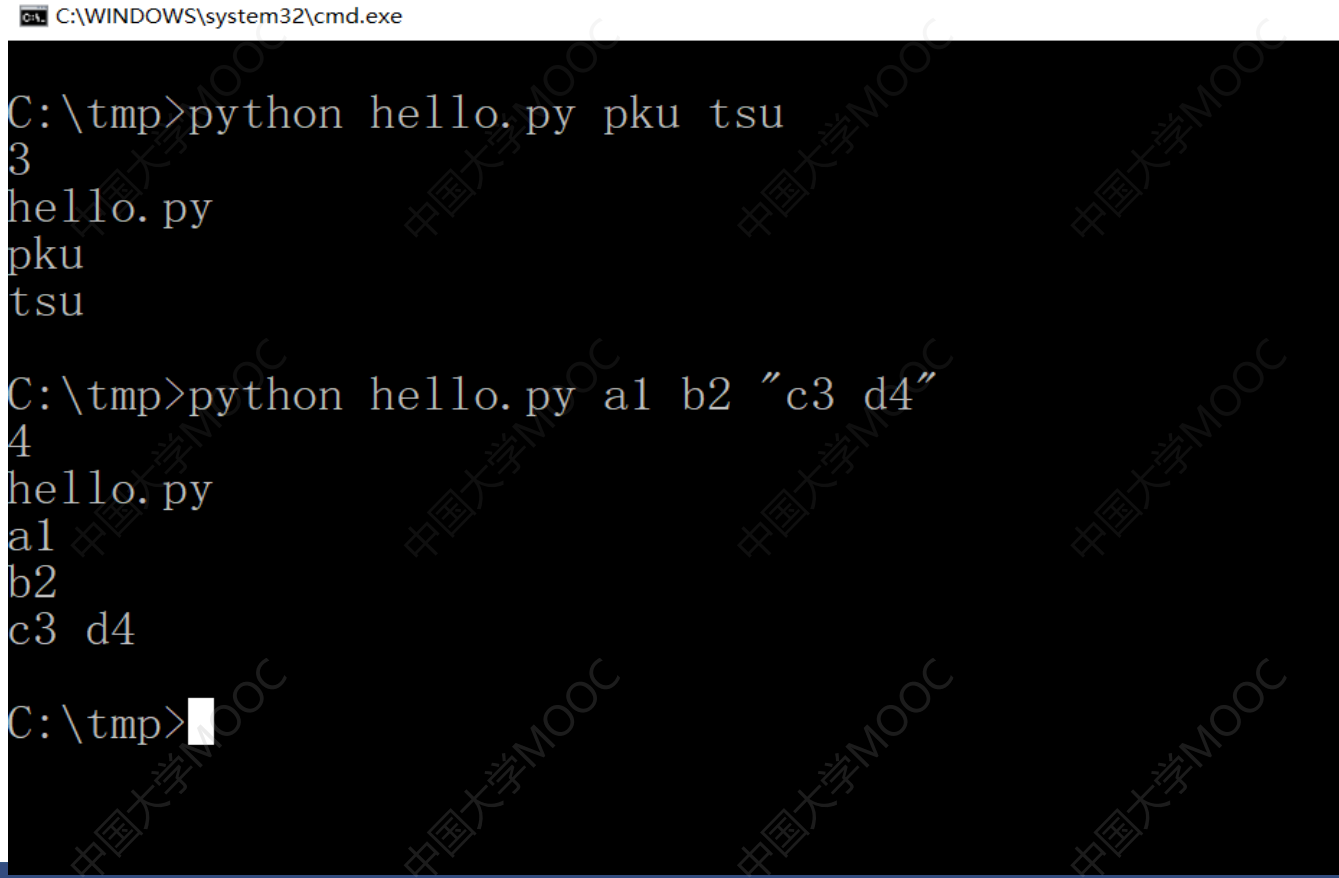
hello.py运行时，如何知道 要处理的文件是 a1.txt和a2.txt呢？

a1.txt, a2.txt 都是“命令行参数”。因此程序内应该有获得命令行参数的方法

命令行参数

程序 **hello.py**如下：

```
import sys
for x in sys.argv:
    print(x)
```



命令行参数

程序 hello.py如下：

```
import sys
for x in sys.argv:
    print(x)
```

在命令行窗口以如下方式运行该程序，假设程序存为 hello.py：

```
python hello.py this is "hello world"
```

则在程序中

sys.argv[0] 就是 'hello.py'

sys.argv[1] 就是 'this'

sys.argv[2] 就是 'is'

sys.argv[3] 就是 'hello world'

输出结果：

```
hello.py
this
is
hello world
```

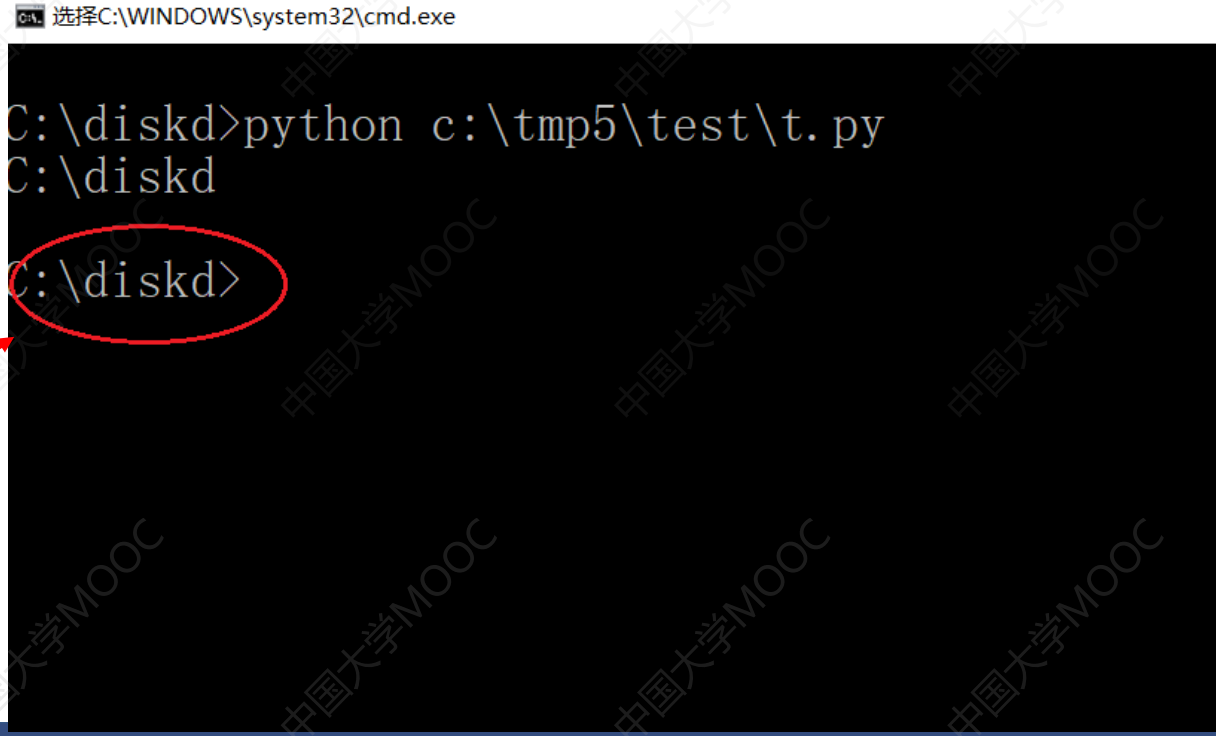
程序以命令行运行时的当前文件夹

程序以命令行方式启动时，当前文件夹就是命令提示符表示的文件夹，而不是python程序文件所在的文件夹。

c:/tmp5/test/t.py

```
import os
print(os.getcwd())
```

命令提示符



文件处理实例

美国纪念碑谷

程序1： 统计文章中的单词词频

- 程序名：countfile.py

用命令行方式启动该程序:

```
python countfile.py 源文件 结果文件
```

例如：

```
python countfile.py a1.txt r1.txt
python countfile.py c:\tmp\a4.txt d:\tmp\r4.txt
```

对 "源文件" 进行单词词频(出现次数)分析，分析结果写入 "结果文件",单词按照字典序排列

程序1： 统计文章中的单词词频

- 文章文件 a1.txt 的格式：

When many couples decide to expand their family, they often take into consideration the different genetic traits that they may pass on to their children. For example, if someone has a history of heart problems, they might be concerned about passing that on to their children as well.

Treacher Collins syndrome, or TCS,?is a rare facial disfigurement that greatly: slows: the development of bones and other tissues that make up the human face. As a result, most people living?with TCS have?underdeveloped cheek bones, a small jaw, and an undersized chin.

.....

程序1： 统计文章中的单词词频

- 统计的结果结果文件 r1.txt 格式

```
a          8
about      2
an         1
and        4
are        1
around     1
as         2
backlash   1
be         4
......
```

程序1： 统计文章中的单词词频

- 思路：

1）命令行参数sys.argv[1]就是源文件，sys.argv[2]就是结果文件。

2）要从a1.txt中分割出单词，然后用字典记录单词的出现频率。

3）分割单词时的分隔字符多种多样，因此要统计a1.txt中出现了哪些非字母的字符，非字母的字符都是分隔串。

4）要用 re.split() 来分割。

程序1： 统计文章中的单词词频

- 回顾：通过正则表达式用多个分隔串进行分割

```
import re
re.split(x,s)  :用正则表达式x里面的分隔串分割s
```

x里面不同分隔串用"|"隔开，形如：

```
';| |,|\*|\n|\?|ok|8'
```

一些特殊字符，比如： ? ! " '( ) | * $ \ [ ] ^ { } . ,

在正则表达式里出现时，前面需要加 \

程序1： 统计文章中的单词词频

➢回顾：通过正则表达式用多个分隔串进行分割

```
import re
a = 'Beautiful, is; beoktter*than\nugly'
print(re.split(';| |,|\*|\n|ok',a)) #分隔串用 | 隔开]
```

';' ' ' ',' '*' '\n' 'ok' 都被看作分隔串

```
#>>['Beautiful', '', 'is', '', 'be', 'tter', 'than', 'ugly']
```

两个相邻的分隔串之间，会隔出一个空串

程序2：统计多个文件累计单词频率

➢ 程序名 countfiles.py ：

- 用法:

python countfiles.py 结果文件

例如：

python countfiles.py result.txt

对当前文件夹(countfiles.py 文件所在文件夹）下全部文件名是字母a打头的.txt文件进行词频统计，统计的总的结果写入 "结果文件" result.txt

程序2：统计多个文件累计单词频率

- 思路:

要获得 .py 程序所在文件夹下的所有a打头，.txt结尾的文件。对每个文件，调用上面程序1的处理单个文件的函数进行处理

```
import os          #python自带 os 库
```

os.listdir() 可以获得当前文件夹下所有文件和文件夹的列表。列表中元素是文件或文件夹名字，不带路径（目录）

os.path.isfile(x)可以判断x是不是一个文件(文件夹不是文件)

程序2：统计多个文件累计单词频率

- os.listdir示例

假设 c:\tmp 文件夹下有文件

t.py, a.txt,b.txt和文件夹 hello

程序 t.py如下：

```
import os
print(os.listdir())
```

则运行t.py输出结果为：

['a.txt', 'b.txt', 'hello', 't.py']

程序2：统计多个文件累计单词频率

- 实现：

```
result = {}
lst  = os.listdir() #列出当前文件夹下所有文件和文件夹的名字
for x in lst:
    if os.path.isfile(x): #如果x是文件
        if x.lower().endswith(".txt") and x.lower().startswith("a"):
            #x是 'a'开头，  .txt结尾
            countFile(x,result) #countFile是程序1中统计一个文件的函数
```

程序3：准确统计文章中的单词词频

➢ 程序名：countfile_novary.py

- 用法：

python countfile_novary.py 源文件 结果文件

对 "源文件" 进行单词词频分析，分析结果写入 "结果文件"

如果碰到单词的变化形式，则转换成原型再统计

- 单词原型-变化 词汇表在文件 word_varys.txt 里面，格式：

```
act
        acted|acting|acts
action
        actions
active
        actively|activeness
```

程序3：准确统计文章中的单词词频

- 思路

1）同样需要一个字典来统计单词及其出现次数。

2）读取 word_varys.txt文件，构造一个字典dt。元素形式为：

{acted:act, acting:act, acts:act, actions:action,.....}

键是单词的变化形式，值是单词的原型。

3）对每个“源文件”里的单词 w，查找dt中键为w的元素x。如果x不存在，则w就是原型，统计其词频。如果x存在，则值x[1]是原型，将x[1]的出现次数加1。

程序4： countfile_nocet4.py

- 程序名： countfile_nocet4.py

- 用法：

python countfile_nocet4.py 源文件 结果文件

对 "源文件" 进行单词词频分析，只抽取不在四级单词列表中的单词，将分析结果写入 "结果文件"

- 四级单词列表在文件 cet4words.txt 中，单词都是单独一行，以$打头

$a

[e?]

art.一(个)；每一(个)

$abandon

[?'b?nd?n]

vt.遗弃；放弃；放纵(自己)

程序4：countfile_nocet4.py

- 思路：

读取cet4words.txt中的单词，存放到一个集合里面。碰到源文件里的单词，先查查在不在集合里面，如果在，则抛弃。