

01.修改Python运行路径【绝对路径与相对路径】

2020年6月11日 19:21

路径加载连接的三种方式: /、\、r

```
import os
```

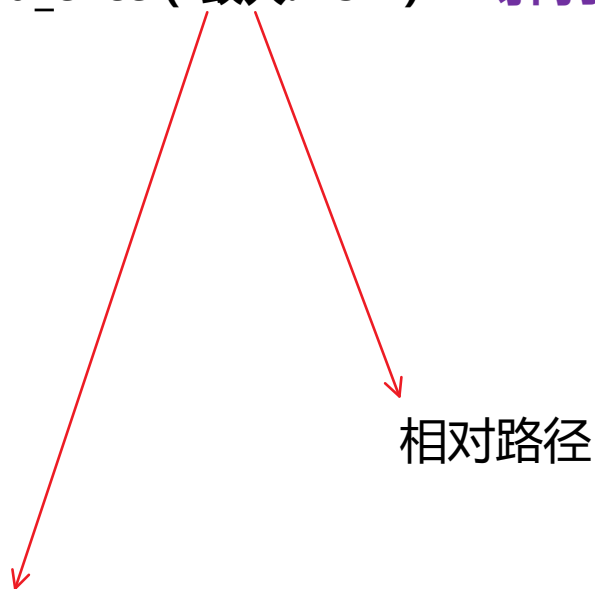
```
import pandas as pd
```

```
os.chdir('C:/超大娃/大娃/中娃/小娃/超小娃') # 建议加上r养成习惯
```

```
# os.chdir('C:\超大娃\大娃\中娃\小娃\超小娃')
```

```
数据 = pd.read_excel("蚁人.xlsx") # 等同于("./蚁人.xlsx")
```

```
print(数据)
```



'C:/超大娃/大娃/中娃/小娃/超小娃/蚁人.xlsx'

绝对路径

02.获得当前python程序运行路径

2020年6月17日 9:01

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

03.Python自动路径连接

2020年6月17日 9:03

os.path.join()函数：连接两个或更多的路径名组件

- 1.如果各组件名首字母不包含' /' ， 则函数会自动加上**
- 2.第一个以"/" 开头的参数开始拼接， 之前的参数全部丢弃,当有多个时， 从最后一个开始**
- 3.如果最后一个组件为空， 则生成的路径以一个' /' 分隔符结尾**

```
import os
```

```
P1 = '\超大娃'
```

```
P2 = '大娃'
```

```
P3 = '中娃'
```

```
P4 = '小娃'
```

```
P5 = '超小娃'
```

```
路径1 = P1 + P2 + P3 + P4 + P5
```

```
路径2 = os.path.join(P1,P2,P3,P4,P5)
```

```
print('c:',路径1)
```

```
print('c:',路径2)
```

04.列出当前程序文件夹下所有内容

2020年6月17日 9:27

```
import os
os.chdir('C:/超大娃/大娃/中娃/小娃/超小娃')
print(os.listdir())
```

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

如果指定了运行路径就返回运行路径文件夹下所有内容（以列表形式返回文件和文件夹）
如果不指定运行路径，就返回Py文件所在文件夹下所有的内容

那么怎么知道输出的文件是文件还是文件夹呢？

这是绝对路径

```
import os
列表 = os.listdir()
for 元素 in 列表:
    print(元素, os.path.isdir(元素))
```

结果输出：文件夹名称，False（不是文件夹） True（是文件夹）

如果是相对路径，要特别注意

```
import os
列表 = os.listdir('c:/')
for 元素 in 列表:
    路径 = 'c:/' + 元素
    print(元素, os.path.isdir(路径))
```

★ 05.os.scandir遍历目录

2020年6月17日 9:36


在 Python 3.5版本中，新添加了 `os.scandir()`方法，它是一个目录迭代方法。低于3.5版本的Py是无法使用的。

建议使用场景：遍历一个文件夹下面所有的文件

参数	说明
name	条目的文件名，相对于 scandir path 参数 (对应于 os.listdir的返回值)
path	输入路径 NAME (不一定是绝对路径) 与 os.path.join(scandir_path, entry.name)
is_dir(*, follow_symlinks=True)	类似于 pathlib.Path.is_dir(), 但返回值在 DirEntry 对象上是缓存； 大多数情况下不需要系统调用； 如果 follow_symlinks 是 false, 则不要跟随符号链接。
is_file(*, follow_symlinks=True)	类似于 pathlib.Path.is_file(), 但返回值在 DirEntry 对象上是缓存； 大多数情况下不需要系统调用； 如果 follow_symlinks 是 false, 则不要跟随符号链接。
is_symlink()	类似 pathlib.Path.is_symlink(), 但返回值缓存在 DirEntry 对象上；大多数情况下不需要系统调用
stat(*, follow_symlinks=True)	类似 os.stat(), 但返回值缓存在 DirEntry 对象上； 不需要对 Windows (。除了符号符号外) 进行系统调用； 如果 follow_symlinks 是 false, 则不跟随符号链接(像 os.lstat())。
inode()	返回项的节点数；返回值在 DirEntry 对象上缓存

```
import os
for file in os.scandir():
    print(file.name, file.path, file.is_dir())
```

```
import os
for 元素 in os.scandir('c:/'):
    print(元素.name, 元素.path,元素.is_dir())
```

 相对路径

结果输出的是：文件夹名称，路径和是否是文件夹的判断

5.1 查询文件的信息

2020年6月18日 9:50

部分参数的说明见下表：

名称	说明
st_size	文件的体积大小（单位：bytes），除以1024就是KB
st_atime	文件的最近访问时间
st_mtime	文件的最近修改时间
st_ctime	windows下创建的时间
st_birthtime	在Mac、linux下使用，表示创建时间

```
import os
```

```
可遍历的对象 = os.scandir('c:/练习1')
```

```
for 文件 in 可遍历的对象:
```

```
    print(文件.name,文件.stat())
```

5.2 查询一个文件的信息

2020年6月18日 10:08

```
import os  
print(os.stat('c:/练习1/李小龙08.jpg'))
```

5.3 处理以秒计算的时间

2020年6月18日 10:11

笔记5.2中:

```
, st_atime=1592442400,
```

```
Thu Jun 18 09:06:40 2020
```

```
import time  
print(time.ctime(1592442400))
```

更详细的应用请看练习2

5.4 datetime模块 【Pandas笔记 附01】

2020年6月18日 10:11

```
import datetime as dt
日期时间 = dt.datetime.fromtimestamp(1592442400)
print(日期时间)
print(日期时间.year,日期时间.month,日期时间.day)
print(日期时间.hour,日期时间.minute,日期时间.second)
```

练习1:

2020年6月17日 10:07

要求: 在指定的文件夹下面

- (1) 找到有多少个文件, 有多少个文件夹, 分别都是什么
- (2) 找到包含哪些文字的文件, 分别都是什么
- (3) 找到指定后缀名的文件 (例如excel文件), 分别都是什么
- (4) 忽略大小写问题

```
import os
```

```
路径 = r'c:/练习1'
```

```
os.chdir(路径)
```

```
文件 = []
```

```
文件夹 = []
```

```
for 元素 in os.scandir():
```

```
    if 元素.is_dir():
```

```
        文件夹.append(元素.name)
```

```
    else:
```

```
        文件.append(元素.name)
```

```
print(f' (1) 文件夹共{len(文件夹)}个。分别是{文件夹}')
```

```
print(f' (2) 文件共{len(文件)}个。分别是{文件}')
```

```
暂存列表 = []
```

```
for 文件名 in 文件:
```

```
    if ('孙兴华' in 文件名) or ('小甲鱼' in 文件名):
```

```
        暂存列表.append(文件名)
```

```
print(f' (3) 孙兴华和小甲鱼的文件共{len(暂存列表)}个, 分别是{暂存列表}')
```

```
if 文件名.endswith('.xlsx'):
```

Python基础篇PPT笔记P24

```
暂存列表2 = []
```

```
for 文件名 in 暂存列表:
```

```
    点的位置 = 文件名.rfind('.')
```

```
    # print(文件名[点的位置+1:])
```

```
    if 文件名[点的位置+1:] == 'xlsx':
```

```
        暂存列表2.append(文件名)
```

```
print(f' (4) Excel文件共{len(暂存列表2)}个, 分别是{暂存列表2}')
```

正则笔记1.2和2.6和2.9

match 从字符串的第一个字符开始匹配, 如果未匹配到返回None, 匹配到则返回一个对象

re.I不区分大小写

group()分组返回整个字符串

```
暂存列表3 = []
```

```
for 文件名 in 文件:
```

```
    正则 = '.*pandas.*'
```

```
    r = re.match(正则, 文件名, re.I)
```

```
try:
    导入re库
```

```
    暂存列表3.append(r.group())
```

```
except:
```

```
    pass
```

```
print(f'包含pandas单词的文件有{len(暂存列表3)}个')
```

异常的写法:

try:

可能发生错误的代码

Python笔记P178页

try:
切记要导入re库
暂存列表3.append(r.group())
except:
pass
print(f'包含pandas单词的文件有{ler

异常的写法:

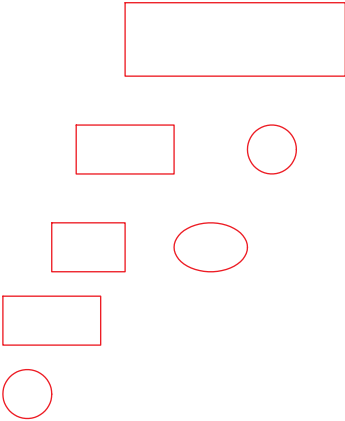
try: 可能发生错误的代码
except: 如果出现异常执行的代码

Python笔记P178页

06.遍历文件夹

2020年6月17日 11:48

- 一、使用os.listdir()函数+递归的方式实现
- 二、使用os.walk () 函数实现 **【推荐】**
- 三、os.scandir方法是一种比较快速的方法，但仍然需要递归来实现，从办公角度讲，我不推荐，办公自动化速度第二，代码简洁第一。【纯属个人想法】



6.1 使用os.listdir()函数+递归的方式实现

【不推荐，此方法较复杂】

2020年6月17日 11:51

```
import os
image_path = 'c:/超大娃'
# 遍历文件夹及其子文件夹中的文件，并存储在一个列表中
# 输入文件夹路径、空文件列表[]
# 返回 文件列表Filelist,包含文件名（完整路径）
def get_filelist(dir, Filelist):
    newDir = dir
    if os.path.isfile(dir):
        Filelist.append(dir)
        # # 若只是要返回文件文，使用这个
        # Filelist.append(os.path.basename(dir))
    elif os.path.isdir(dir):
        for s in os.listdir(dir):
            # 如果需要忽略某些文件名，使用以下代码
            # if s == "xxx":
            # continue
            newDir = os.path.join(dir, s)
            get_filelist(newDir, Filelist)
    return Filelist
if __name__ == '__main__':
    list = get_filelist('c:/超大娃', [])
    print(len(list))
    for e in list:
        print(e)
```

补充：

os.path.basename()函数用于返回路径路径最后的文件名。若路径以/或\结尾，那么就会返回空值。例如：

```
>>> import os
>>> 路径 = 'c:/超大娃'
>>> os.path.basename(路径)
'超大娃'
>>> 路径 = 'c:/超大娃/'
>>> os.path.basename(路径)
''
```

6.2 使用os.walk()函数实现

2020年6月17日 11:53

os.walk(top, topdown=True, onerror=None, followlinks=False)

建议使用场景：遍历一个目录下面若干个目录中所有的文件

参数	说明
top	要遍历的目录的地址 返回的是一个三元组(root,dirs,files)。 (1) root 所指的是当前正在遍历的这个文件夹的本身的地址 (2) dirs 是一个 list , 内容是该文件夹中所有的目录的名字(不包括子目录) (3) files 同样是 list , 内容是该文件夹中所有的文件(不包括子目录)
topdown	可选, 为 True, 则优先遍历 top 目录, 否则优先遍历 top 的子目录(默认为开启)。 如果 topdown 参数为 True, walk 会遍历top文件夹, 与top 文件夹中每一个子目录。
onerror	需要一个 callable 对象, 当walk需要异常时, 会调用
followlinks	可选, 如果为 True, 则会遍历目录下的快捷方式, 实际所指的目录(默认关闭), 如果为 False, 则优先遍历 top 的子目录。

```
import os
j = 0
for 文件夹路径,子文件夹列表,文件列表 in os.walk('c:/超大娃'):
    # print(文件夹路径)
    # print(子文件夹列表)
    # print(文件列表)
    for i in 文件列表:
        print(文件夹路径+i)
        j += 1
print(f'共发现{j}个文件')
```

07.搜索、匹配文件名称

2020年6月17日 13:44

- (1) 利用字符串内置的方法: `.startswith()`和`.endswith()`
- (2) `glob`模块
- (3) `fnmatch`模块

7.1 字符串内置的方法：startswith()和endswith()

2020年6月18日 8:45

```
import os
```

```
for 元素 in os.scandir('c:/练习1'):
```

```
    # if 元素.name.startswith('孙兴华'):
```

```
        if 元素.name.endswith('.xlsx') or 元素.name.endswith('.xls'):
```

```
            print(元素.name, 元素.path,元素.is_dir())
```


7.2 glob模块

2020年6月18日 8:45

glob是python自带的一个文件操作相关模块，用它可以查找符合自己要求的文件，类似于Windows下的文件搜索，支持通配符操作，有 “*”、 “?” 、 “[]” 这三个通配符，

“*” ：代表0个或者多个字符；

“? ” ：代表一个字符；

“[]” ：匹配指定范围内的字符，如[0-9]匹配数字；主要有以下2个主要方法。

★ 1.glob方法:

glob模块的主要方法就是glob，该方法返回所有匹配的文件路径列表(list)；该方法需要一个参数用来制定匹配的路径字符串(字符串可以为绝对路径也可以为相对路径)，其返回文件名只包括当前目录里的文件名，不包括子文件夹里的文件。

```
import os
import glob
os.chdir('C:/练习1')
# print(glob.glob('*.txt'))
# print(glob.glob('孙*.txt'))
# print(glob.glob('孙兴华.分析[0-9].*'))
# print(glob.glob('孙兴华.分析[!0-9].*'))
列表 = glob.glob('*.jpg')
print(列表)
for 文件 in 列表:
    print(文件)
```

返回一个列表

遍历文件夹深层目录下所有的指定文件：

```
import os
import glob
os.chdir('C:/练习1')
列表 = glob.glob('**/*.xlsx',recursive=True)
print(列表)
```

用**表示任意层文件或文件夹，recursive=True会不断进入文件夹内。查找该路径下面的所有.xlsx文件，包括深层的文件

2.iglob方法:

获取一个可遍历对象，使用它可以逐个获取匹配的文件路径名。与glob.glob()的区别是：glob.glob同时获取所有的匹配路径，而 glob.iglob一次只获取一个匹配路径。这有点类似于.NET中操作数据库用到的DataSet与DataReader。下面是一个简单的例子：

```
import os
import glob
os.chdir('C:/练习1')
可遍历对象 = glob.iglob('*.jpg')
print(可遍历对象)
for 文件 in 可遍历对象:
    print(文件)
```

模式	意义
*	匹配所有
?	匹配任意字符
[seq]	匹配seq中的任何字符
[!seq]	匹配任何不在seq中的字符

7.3 fnmatch模块

2020年6月18日 9:30

使用os.listdir来列出当前目录中的文件，如果匹配的后缀为.txt，那么就打印出来，fnmatch.fnmatch是一个布尔函数，返回为True或者False。

import os

import fnmatch

目录列表 = os.listdir('c:/练习1')

for 文件名 in 目录列表:

if fnmatch.fnmatch(文件名, '*.txt'): # 匹配模式为星号，表示任意的字符。当fnmatch.fnmatch(文件名, '*.txt')返回True时，执行下方缩进代码块

print(文件名)

模式	意义
*	匹配所有
?	匹配任意字符
[seq]	匹配seq中的任何字符
[!seq]	匹配任何不在seq中的字符

import os

import fnmatch

目录列表 = os.listdir('c:/练习1')

for 文件名 in 目录列表:

if fnmatch.fnmatch(文件名, '?el.zip'):

if fnmatch.fnmatch(文件名, '*el.zip'):

if fnmatch.fnmatch(文件名, '[a-z]el.zip'):

if fnmatch.fnmatch(文件名, '[!a-z]el.zip'):

print(文件名)

练习2:

2020年6月18日 10:30

编写程序, 要求:

- (1) 搜索整个文件夹, 包括文件夹中所有的文件夹
- (2) 筛选体积大于2MB的压缩包.zip文件
- (3) 筛选这些文件中日期早于17日之前的文件
- (4) 输出这些文件的路径

文件夹选取电脑中某一个文件, 编写原代码如下:

(1) 文件大于2M,时间是2020年以前的

```
import os
import glob
import datetime as dt
os.chdir('C:/练习1')
所有mp4文件的列表 = glob.glob('**/*.mp4',recursive=True)
for 文件 in 所有mp4文件的列表:
    文件大小 = os.stat(文件).st_size/1024/1024    # 笔记5.1, 除1024是kb, 再除1024就是mb
    年 = dt.datetime.fromtimestamp(os.stat(文件).st_ctime).year
    if (文件大小 > 2) and (年 < 2020):
        print(f'路径:{文件},大小{int(文件大小)}mb,于{年}年创建')
```

(3) 文件大于2M,时间是2020.6.17以前的

```
import os
import glob
import time
os.chdir('C:/练习1')
所有mp4文件的列表 = glob.glob('**/*.mp4',recursive=True)
for 文件 in 所有mp4文件的列表:
    文件大小 = os.stat(文件).st_size/1024/1024
    时间数组 = time.localtime(os.stat(文件).st_ctime)
    日期 = time.strftime('%Y-%m-%d', 时间数组)
    if (文件大小 > 2) and (日期 < '2020-06-17'):
        print(f'路径:{文件},大小{int(文件大小)}mb,于{日期}日创建')
```

(3) 删除文件 (没有指定文件会报错)

os.remove(目标文件名)

详见Python基础PPT笔记P121

把os.remove(文件)写到if语句下方缩进代码中

08.创建文件夹【单层】

2020年6月18日 20:42

一、创建文件夹

```
import os  
os.mkdir('c:/孙兴华')
```

二、如果文件夹不存在就创建

```
import os  
# os.path.exists(路径)  
# 如果路径存在，返回True；如果path不存在，返回False。  
if not os.path.exists('c:/孙兴华'):  
    os.mkdir('c:/孙兴华')
```

二、在当前Py文件的路径下创建文件夹

```
import os  
# 获取当前路径  
当前路径 = os.getcwd()  
路径 = 当前路径 + r'\abc' # '\\abc'  
if os.path.exists(路径):  
    print('路径已经存在')  
else:  
    os.mkdir(路径)
```

三、建议大家用函数的方式写，随时调用【通用模版】

```
# 导入内置os模块  
import os  
# 创建单层目录
```

```
def 创建文件夹(路径):  
    # basename:返回目录路径中的最后一个元素  
    目录名称 = os.path.basename(路径)  
    # 判断路径是否存在  
    是否存在 = os.path.exists(路径)  
    if not 是否存在:  
        # 如果不存在, 则创建单层目录  
        os.mkdir(路径)  
        print('目录创建成功: ' + 目录名称)  
        return True  
    else:  
        # 如果目录存在则不创建, 并提示目录已存在  
        print('目录已存在: ' + 目录名称)  
        return False  
  
if __name__ == "__main__":  
    创建文件夹('c:/孙兴华/华兴孙')
```

09.创建文件夹【多层】

2020年6月18日 20:57

一、创建多层文件夹

```
import os
os.makedirs('c:/大/中/小/超小')
```

二、建议写成函数，随时调用

```
import os
def 创建多层文件夹(路径):
    # 判断路径是否存在
    是否存在 = os.path.exists(路径)
    if not 是否存在:
        # 如果不存在，则创建目录（多层）
        os.makedirs(路径)
        print('目录创建成功! ')
        return True
    else:
        # 如果目录存在则不创建，并提示目录已存在
        print('目录已存在! ')
        return False
if __name__ == "__main__":
    创建多层文件夹('c:/超大/大/中/小')
```

10.python操作文件，强大的shutil模块

2020年6月18日 21:15

1、找到pip3.exe所在的文件夹，复制路径

我的路径是： C:\Users\孙艺航\AppData\Local\Programs\Python\Python37\Scripts

2、按Win+R,输入CMD确定

3、进入后，先输入cd 路径 回车

4、输入 pip3 install shutil 回车

10.1 复制文件 【copy与copy2】

2020年6月18日 21:35

```
import shutil
```

```
原地址 = 'c:/1.zip'
```

```
新地址1 = 'c:/孙兴华'
```

```
新地址2 = 'c:/孙兴华/No1.zip'
```

```
shutil.copy(原地址,新地址2)
```

```
# 复制后的结果保留了原来的所有信息（包括状态信息）
```

```
新地址3 = 'c:/孙兴华/No2.zip'
```

```
shutil.copy2(原地址,新地址2)
```

注意文件夹要保证存在



复制文件：shutil.copy(要复制的文件，要复制到的文件位置)

10.2 复制文件夹 **【copytree】**

2020年6月18日 21:35

```
import shutil  
shutil.copytree('c:/孙兴华','c:/练习2')
```

→ 不能是已经存在的文件夹
但是 'c:/练习2/练习3'

复制文件夹: `shutil.copytree`(要复制的文件夹, 要复制到的新的文件夹位置)

10.3 移动文件/文件夹 **【move】**

2020年6月18日 21:35

移动文件/文件夹: shutil.move(要移动的文件/文件夹, 要复制到的文件位置)

```
import shutil
shutil.move('c:/1.zip', "c:/练习1")
shutil.move('c:/1.zip', "c:/练习1/2.zip")
shutil.move('c:/练习2', "c:/练习1")
```

10.4 删除文件/文件夹

2020年6月18日 22:36

一、删除文件，直接使用os模块下的remove方法

```
import os  
os.remove('c:/2.zip')
```

二、删除所有的文件夹

```
import shutil  
shutil.rmtree('c:/超大')
```

10.4 重命名文件/文件夹

2020年6月18日 21:35

os.rename(要重命名的文件/文件夹,新的名字)

一、修改文件和文件夹的名字

```
import os
os.rename('c:/改名','c:/新名字')
os.rename('c:/改名.zip','c:/新名字.zip')
```

二、改名同时进行移动

```
import os
os.rename('c:/新名字','c:/练习1/改名')
os.rename('c:/新名字.zip','c:/练习1/改名.zip')
```

练习3:

2020年6月18日 22:54

- (1) 在练习1文件夹下找到所有mp4文件
- (2) 重命名, 要求是使用每个文件的创建日期+原始名称
- (3) 将重命名的文件移动到指定文件夹

```
import os
import datetime as dt

os.chdir('c:/练习1')
if not os.path.exists('c:/文件存放'):
    os.mkdir('c:/文件存放')
# 使用os.walk将练习1下面所有的文件夹都找到
for 文件夹路径,子文件夹列表,文件列表 in os.walk('c:/练习1'): # 笔记6.2
    # 遍历所有文件夹中的所有文件
    for 文件 in os.listdir(文件夹路径):
        if 文件.name.endswith(".mp4"):
            日期 = dt.datetime.fromtimestamp(文件.stat().st_mtime)
            新名字 = str(日期.year)+'-'+str(日期.month)+'-'+str(日期.day)+'-'+文件.name
            print(f'{文件.name}, 重命名为{新名字}')
            os.rename(文件夹路径 + '/' + 文件.name, 'c:/文件存放/'+新名字)
```

11.读写文件内容

2020年6月18日 23:24

11.1 复习访问模式

2020年6月18日 23:43

(1) 访问模式 (三个主访问模式)

'r' 只读: 如果文件不存在报错, 不支持写

'w' 写入: 如果文件不存在新建文件, 写入时覆盖原有内容

'a' 追加: 如果文件不存在新建文件, 写入时在原有内容基础上追加新内容

总结: 访问模式可以省略, 默认为'r'模式

(2) 访问模式特点 ('b' 二进制、'+' 可读可写)

r、rb、r+、rb+: 只要文件不存在都报错, 文件指针 (光标的位置) 放在文件开头

w、wb、w+、wb+: 只要文件不存在就新建文件, 文件指针在开头, 用新内容覆盖原内容

a、ab、a+、ab+: 只要文件不存在新建文件, 文件指针在结尾,

Py基础篇PPT笔记 P115页

11.2 读取文件的方法

2020年6月18日 23:47

文件对象.read(num)

num表示要从文件中读取数据的长度（单位是字节）【换行\n占一个字节】，省略就表示读取所有数据

文件对象.readlines() # 需要赋值给一个变量

将整个文件中的内容一次性读取，并返回一个列表，原文件中每一行的数据为一个元素，例如['aaa\n','bbb\n',ccc]
每一行都有换行自带\n，最后一行没有换行不带\n

文件对象.readline() # 需要赋值给一个变量

一次性读取一行内容，第一次调用读取第一行，第二次调用读取第二行，不带换行符\n

Py基础篇PPT笔记 P117页

11.4 with语句

2020年6月15日 20:08

Python基础课程PPT笔记184页 with语句

第15课 异常处理和存储数据

**with语句：打开文件不用手动关闭，
但要注意缩进**

```
try:
    变量名 = open('1999绝秘档案.txt','w')
    for 遍历文件 in 变量名:
        print(遍历文件)
except OSError as 错误信息:
    print(f'错了，出错信息是{错误信息}')
finally:
    变量名.close()
```

finally

AAAAA x

C:\Users\...\AppData\Local\Programs\Python\Pyth
错了，出错信息是not readable

```
try:
    with open('1999绝秘档案.txt','w') as 变量名:
        for 遍历文件 in 变量名:
            print(遍历文件)
except OSError as 错误信息:
    print(f'错了，出错信息是{错误信息}')
```

try › with open("1999绝秘档案.txt","w") a... › for 遍历文件 in 变量名

AAAAA x

C:\Users\...\AppData\Local\Programs\Python\Pyth
错了，出错信息是not readable

语法：

**with 表达式 as 变量：
代码块**

11.5 with 语句读取

2020年6月18日 23:46

```
with open('c:/测试.txt','r',encoding='utf-8') as 文件:  
    数据 = 文件.readlines()  
    print(数据)
```

11.6 with语句写入

2020年6月19日 0:10

with open('c:/测试.txt','a',encoding='utf-8') as 文件:

数据 = '第一行内容\n第二行内容\n'

文件.write(数据)

文件.write('第三行内容')

11.7 使用Pandas读取文件

2020年6月18日 23:42

操作excel文件时，使用read_excel和to_excel

```
import pandas as pd
```

```
路径 = 'c:/测试.txt'
```

```
数据 = pd.read_csv(路径)
```

```
数据.to_csv('c:/测试2.txt',index=False)
```

详见Pandas课程：[点击这里查看](#)

跟着孙兴华学习Pandas 大数据分析入门课 python教程进阶篇 【本季完】办公自动化 Excel

知识 > 野生技术协会 2020-05-23 14:06:52

—播放— 弹幕 未经作者授权，禁止转载



12. TemporaryFile 模块

2020年6月19日 0:17

一、创建临时文件存储数据 TemporaryFile 模块

```
from tempfile import TemporaryFile
```

```
文件 = TemporaryFile('w+')
```

```
文件.write('孙兴华'*1000000)
```

```
文件.seek(0)
```

```
数据 = 文件.read(12)
```

```
print(数据)
```

```
print(文件)
```

```
文件.close()
```

建议在爬虫时使用

二、建议写成with...as形式

```
from tempfile import TemporaryFile
```

```
with TemporaryFile('w+') as 文件:
```

```
    文件.write('孙兴华'*1000000)
```

```
    文件.seek(0)
```

```
    数据 = 文件.readlines()
```

```
    print(数据)
```

三、创建临时文件夹存储数据

```
from tempfile import TemporaryDirectory
```

```
with TemporaryDirectory() as 文件夹:
```

```
    print(f'临时文件夹是: {文件夹}')
```

13.Python解压ZIP文件【zipfile模块】

2020年6月19日 3:54

一、读取压缩包 zipobj.namelist()

```
import zipfile
with zipfile.ZipFile('c:/练习1.zip','r') as 文件:
    for 文件名 in 文件.namelist():
        print(文件名.encode('cp437').decode('gbk'))
```

```
import zipfile
with zipfile.ZipFile('c:/练习2.zip','r') as 文件:
    for 文件名 in 文件.namelist():
        print(文件名)
# 文件中没有中文就不用设置编码
```

二、查看压缩包里面的信息 zipobj.getinfo()

```
import zipfile
with zipfile.ZipFile('c:/练习1.zip','r') as 文件:
    for 文件名 in 文件.namelist():
        信息 = 文件.getinfo(文件名)
        文件名 = 文件名.encode('cp437').decode('gbk')
        # 输出结果: 文件名称、文件大小、压缩后的大小(单位是字节)
        print(文件名, 信息.file_size, 信息.compress_size)
```

三、解压单个文件 zipobj.extract()

```
import zipfile
with zipfile.ZipFile('c:/练习2.zip','r') as 文件:
    文件.extract('WinRAR.rar', 'c:/练习2')
```

四、解压全部文件 zipobj.extractall()

```
import zipfile
with zipfile.ZipFile('c:/练习2.zip','r') as 文件:
    文件.extractall(path = 'c:/练习2')
```

五、解压有中文名字的zip文件

正常解压文件。

使用正确的文件名重命名解压的文件。

对应的代码如下，这里使用了 pathlib 库，强烈推荐该库！

```
from pathlib import Path
import zipfile
import os
os.chdir('c:/练习2') # 保证有这个文件夹
with zipfile.ZipFile('c:/练习1.zip', 'r') as 文件:
    for 文件名 in 文件.namelist():
        解压后的文件 = Path(文件.extract(文件名, pwd=b'1234'))
        解压后的文件.rename(文件名.encode('cp437').decode('gbk'))
```

有密码时在这里设置密码

原因

原因很简单，zipfile 会将所有文件名用 CP437 来编码，官方说明如下：

There is no official file name encoding for ZIP files. If you have unicode file names, you must convert them to byte strings in your desired encoding before passing them to write(). WinZip interprets all file names as encoded in CP437, also known as DOS Latin.

知道文件名的编码后，就可以使用对应的编码来解码了。也就是先用 CP437 编码 encode 成 bytes，再以 gbk 格式解码成中文 string。

使用 extract 方法而不是 extractall 方法，对压缩文件内的文件名进行遍历，逐个解压。

14.Python创建ZIP文件

2020年6月19日 5:00

```
import os
import zipfile
```

压缩文件

```
def zipDir(dirPath, outFileName=None):
```

```
    ...
```

```
    :param dirPath: 目标文件或文件夹
```

```
    :param outFileName: 压缩文件名,默认和目标文件相同
```

```
    :return: none
```

```
    ...
```

```
    # 如果不校验文件也可以生成zip文件, 但是压缩包内容为空, 所以最好有限校验路径是否正确
```

```
    if not os.path.exists(dirPath):
```

```
        raise Exception("文件路径不存在: ", dirPath)
```

```
    # zip文件名默认和被压缩的文件名相同, 文件位置默认在待压缩文件的同级目录
```

```
    if outFileName == None:
```

```
        outFileName = dirPath + ".zip"
```

```
    if not outFileName.endswith("zip"):
```

```
        raise Exception("压缩文件名的扩展名不正确! 因以.zip作为扩展名")
```

```
    zip = zipfile.ZipFile(outFileName, "w", zipfile.ZIP_DEFLATED)
```

```
    # 遍历文件或文件夹
```

```
    for path, dirnames, filenames in os.walk(dirPath):
```

```
        fpath = path.replace(dirPath, "")
```

```
        for filename in filenames:
```

```
            zip.write(os.path.join(path, filename), os.path.join(fpath, filename))
```

```
    zip.close()
```

```
if __name__ == "__main__":
```

```
    zipDir(r"C:\练习1")
```

文件校验, 目标文件夹是否存在。

创建ZIP文件对象。

遍历目标文件夹, 将文件写入到ZIP对象中。

遍历完毕, 关闭文件流。