
tensorflow / tensorflow
TensorFlow: Open sou
Clone this repo:
git clone https://tensor
Branches
master

4.4 文件

打开文件

文件对象 = open (文件名, 访问模式)

- **绝对路径**：从**盘符**开始的路径

  C:\windows\system32\cmd.exe

  C:\Users\Administrator\.jupyter_notebook_config.py

  D:\Program Files\Anaconda3\lib\site-packages\ipykernel__main__.py

- **相对路径**：从**当前目录（工作目录）**的路径

  D:\jupyter\example \python\file.txt → **D:\jupyter\example**python\file.txt

  D:\jupyter\example ..\file.txt → **D:\jupyter**file.txt

  D:\jupyter\example ..\tensorflow\file.txt → **D:\jupyter**tensorflow\file.txt

获取当前路径

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

current working directory

访问模式

```
文件对象 = open（文件名，访问模式）
```

参数“访问模式”的可取值

访问模式	执行操作
'r'	以只读方式打开文件
'w'	以写入方式打开文件，会覆盖已经存在的文件
'a'	以写入方式打开文件，在末尾追加写入
'+'	以可读写的方式打开文件
'b'	以二进制模式打开文件
't'	以文本模式打开文件（默认）

访问模式	执行操作
'rb'	二进制读模式
'wb'	二进制写模式
'ab'	二进制追加模式

例：在C盘根目录下，创建文本文件myfile.txt，然后使用open()函数打开它：

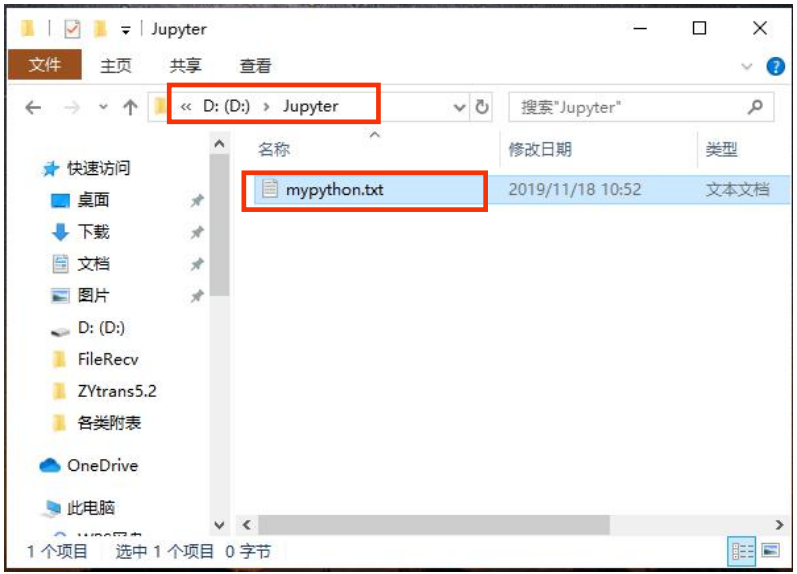
```
f = open("c:/myfile.txt")
```

使用open()函数成功打开一个文件之后，它会返回一个**文件对象**；否则，就会返回一个错误。

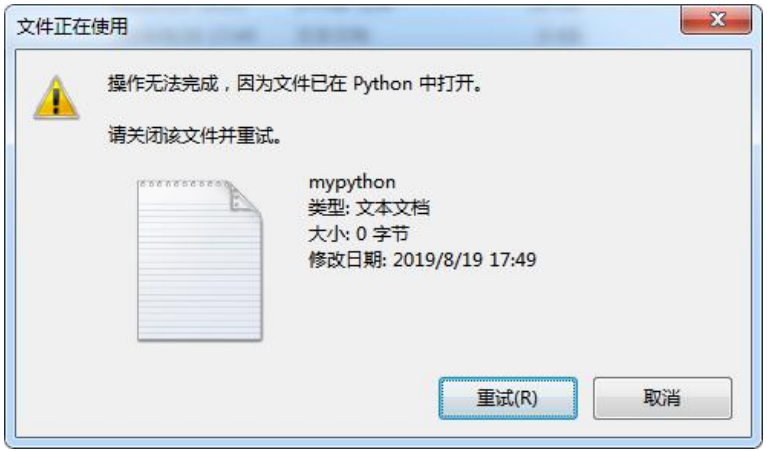
只有文件名，默认为**当前工作目录**

```
f = open("mypython.txt",'w')
```

访问模式是 "w"，如果在工作目录下找不到这个文件，就会自动创建它。



在资源管理器中**删除**上一步创建的文件mypython.txt，会出现错误提示：



关闭文件

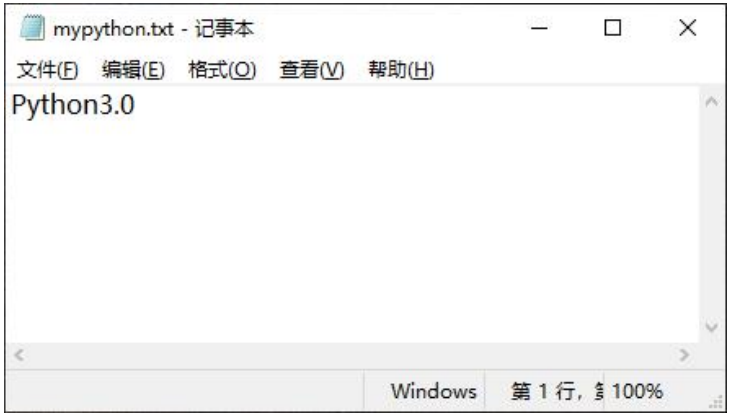
```
文件对象.close()
```

```
f.close()
```

- Python有**垃圾回收机制**，会自动关闭不再使用的文件
- 在对文件进行了**写入操作**后，应该立刻关闭文件，以避免意外事故造成的错误

读取文件的内容

在资源管理器中，打开mypython.txt文本文件，写入“Python3.0”



read()方法：读取整个文档

文件对象.read()

```
>>>f = open("mypython.txt")
>>>f.read()
'Python3.0'
```

例：读取文件

step1:

在资源管理中，在当前工作目录下，创建文本文件The Zen of Python.txt，输入文本内容。

The Zen of Python.txt

```
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

step2: 打开并读取文件内容

```
>>>fp=open("the zen of python.txt")
>>>fp.read()
"The Zen of Python, by Tim Peters\n\nBeautiful is better than ugly.
\nExplicit is better than implicit.\nSimple is better than complex.\n
Complex is better than complicated.\nFlat is better than nested.\n
Sparse is better than dense.\nReadability counts.\nSpecial cases
aren't special enough to break the rules.\nAlthough practicality
beats purity.\nErrors should never pass silently.\nUnless explicitly
silenced.\nIn the face of ambiguity, refuse the temptation to guess.
\nThere should be one-- and preferably only one --obvious way to
do it.\nAlthough that way may not be obvious at first unless you're
Dutch.\nNow is better than never.\nAlthough never is often better
than *right* now.\nIf the implementation is hard to explain, it's a
 bad idea.\nIf the implementation is easy to explain, it may be a
good idea.\nNamespaces are one honking great idea -- let's do
more of those!"
```

readline()方法：每次只读取文件中的一行

文件对象.readline()

```
>>>fp=open("the zen of python.txt")
>>>fp.readline()
"The Zen of Python, by Tim Peters\n"

>>>fp.readline()
"\n"

>>>fp.readline()
"Beautiful is better than ugly.\n"
>>>fp.readline()
"Explicit is better than implicit.\n"
>>>fp.readline()
"Simple is better than complex.\n"
```

文件指针

read()

The Zen of Python.txt

```
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

文件指针

readline()

The Zen of Python.txt

```
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

读取文件的内容——指定字节数

```
文件对象.read(字节数)
```

```
文件对象.readline(字节数)
```

```
"The Zen of Python, by Tim Peters\n\nBeautiful is better than ugly.
\nExplicit is better than implicit.\nSimple is better than complex.\n
omplex is better than complicated.\nFlat is better than nested.\n
Sparse is better than dense.\nReadability counts.\nSpecial cases
aren't special enough to break the rules.\nAlthough practicality
beats purity.\nErrors should never pass silently.\nUnless explicitly
silenced.\nIn the face of ambiguity, refuse the temptation to guess.
\nThere should be one-- and preferably only one --obvious way to
do it.\nAlthough that way may not be obvious at first unless you're
Dutch.\nNow is better than never.\nAlthough never is often better
than *right* now.\nIf the implementation is hard to explain, it's a
 bad idea.\nIf the implementation is easy to explain, it may be a
good idea.\nNamespaces are one honking great idea -- let's do
more of those!"
```

```
>>>fp.readline(10)
'Complex is'

>>>fp.read(8)
' better '
```

向文件中写入数据

write(写入内容)

- 在使用write()函数之前，要确保open()函数的**访问模式**，是**支持写入**的。
- 写入成功之后，会返回写入的**字符数**，是一个整数。

```
f = open("myfile.txt","w")
f.write("Hello, World!")
```

例：完整实例——打开文件，读取文件，写入文件，关闭文件

```
>>>f = open("myfile.txt","w")
>>>f.write("Hello!")
>>>f.close()

>>>f = open("myfile.txt")
>>>f.read()
'Hello!'
>>>f.close()

>>>f = open("myfile.txt","a")
>>>f.write("World!")
6
>>>f.close()
```

```
>>>f = open("myfile.txt")
>>>f.read()
'Hello!World!'
>>>f.close()

>>>f = open("myfile.txt","w")
>>>f.write("Python3.0")
9
>>>f.close()

>>>f = open("myfile.txt")
>>>f.read()
Python3.0
```