


```
1. #这件没问题，正常与八J义什。
8.
9. #-----
10. s = "ShowMeAI是数据科学与人工智能领域的资料库和学习社区。\\
11. 我喜欢ShowMeAI!!\\
12. "
13. b = bytes(s,encoding='utf-8')
14.
15. f = open('test_b.txt','wb')  #注意多了个b
16. f.write(s)
17.
18. #报错
19. TypeError: a bytes-like object is required, not 'str'
20. #意思是它需要一个bytes类型数据，你却给了个字符串
21.
22. #-----
23. s = "ShowMeAI是数据科学与人工智能领域的资料库和学习社区。\\
24. 我喜欢ShowMeAI!!\\
25. "
26. b = bytes(s,encoding='utf-8')
27.
28. f = open('test_b.txt','wb')  # 注意多了个b
29. f.write(b)                # 将变量b传给它，b是个bytes类型
```

+模式

对于w+模式，在读写之前都会清空文件的内容，建议不要使用！

对于a+模式，永远只能在文件的末尾写入，有局限性，建议不要使用！

对于r+模式，也就是读写模式，配合seek()和tell()方法，可以实现更多操作。

文件编码

要读取非UTF-8编码的文件，需要给open()函数传入encoding参数，例如，读取GBK编码的文件：

```
1. >>> f = open('gbk.txt', 'r', encoding='gbk')
2. >>> f.read()
3. 'GBK'
```

遇到有些编码不规范的文件，可能会抛出 `UnicodeDecodeError` 异常，这表示在文件中可能夹杂了一些非法编码的字符。遇到这种情况，可以提供errors参数，表示如果遇到编码错误后如何处理。

```
1. >>> f = open('gbk.txt', 'r', encoding='gbk', errors='ignore')
```

文件对象操作

每当我们用open方法打开一个文件时，将返回一个文件对象。这个对象内置了很多操作方法。下面假设，已经打开了一个f文件对象。

read函数

读取一定大小的数据，然后作为字符串或字节对象返回。size是一个可选的数字类型的参数，用于指定读取的数据量。当size被忽略了或者为负值，那么该文件的所有内容都将被读取并且返回。

```
1. f = open("test.txt", "r")
2.
3. str = f.read()
4. print(str)
5.
6. f.close()
```

如果文件体积较大，请不要使用read()方法一次性读入内存，而是read(512)这种一点一点的读。

readline函数

从文件中读取一行n内容。换行符为'\n'。如果返回一个空字符串，说明已经已经读取到最后一行。这种方法，通常是读一行，处理一行，并且不能回头，只能前进，读过的行不能再读了。

```
1. f = open("test.txt", "r")
2. str = f.readline()
3. print(str)
4. f.close()
```

readlines函数

将文件中所有的行，一行一行全部读入一个列表内，按顺序一个一个作为列表的元素，并返回这个列表。readlines方法会一次性将文件全部读入内存，所以也存在一定的风险。但是它有个好处，每行都保存在列表里，可以随意存取。

```
1. f = open("test.txt", "r")
2. a = f.readlines()
3. print(a)
4. f.close()
```

遍历文件

实际上，更多的时候，我们将文件对象作为一个迭代器来使用。

```
1. # 打开一个文件
2. f = open("test.txt", "r")
3.
4. for line in f:
5.     print(line)
```

```
4. for line in f:
5.     print(line, end=")
6.
7. # 关闭打开的文件
8. f.close()
```

这个方法很简单, 不需要将文件一次性读出, 但是同样没有提供一个很好的控制, 与readline方法一样只能前进, 不能回退。

几种不同的读取和遍历文件的方法比较：

- 如果文件很小, read()一次性读取最方便。
- 如果不能确定文件大小, 反复调用read(size)比较保险。
- 如果是配置文件, 调用readlines()最方便。
- 普通情况, 使用for循环更好, 速度更快。

write函数

将字符串或bytes类型的数据写入文件内。write()动作可以多次重复进行, 其实都是在内存中的操作, 并不会立刻写回硬盘, 直到执行close()方法后, 才会将所有的写入操作反映到硬盘上。在这过程中, 如果想将内存中的修改, 立刻保存到硬盘上, 可以使用f.flush()方法, 但这可能造成数据的不一致。

```
1. # 打开一个文件
2. f = open("/tmp/foo.txt", "w")
3.
4. f.write("ShowMeAI是数据科学与人工智能领域的资料库和学习社区。\\
5. 我喜欢ShowMeAI!!\\
6. ")
7.
8. # 关闭打开的文件
9. f.close()
```

tell函数

返回文件读写指针当前所处的位置,它是从文件开头开始算起的字节数。一定要注意了, 是字节数, 不是字符数。

seek函数

如果要改变位置指针的位置, 可以使用 f.seek(offset, from_what) 方法。seek()经常和tell()方法配合使用。

from_what 的值, 如果是0表示从文件开头计算, 如果是1表示从文件读写指针的当前位置开始计算, 2表示从文件的结尾开始计算, 默认为0, 例如：

offset：表示偏移量。

- seek(x,0)： 从起始位置即文件首行首字符开始移动 x 个字符
- seek(x,1)： 表示从当前位置往后移动x个字符
- seek(-x,2)：表示从文件的结尾往前移动x个字符

参考下述代码示例

```
1. >>> f = open("test.txt", "rb+")
2. >>> f.write(b"ShowMeAI is born for AI and data science.\\
3. I love ShowMeAI!!\\
4. ")
5. 60
6. >>> f.tell()
7. 60
8. >>> f.seek(5)
9. 5
10. >>> f.read(6)
11. b'eAI is'
12. >>> f.seek(-3, 2)
13. 92
14. >>> f.read(2)
15. b'!!'
```

close函数

关闭文件对象。当处理完一个文件后, 调用f.close()来关闭文件并释放系统的资源。文件关闭后, 如果尝试再次调用该文件对象, 则会抛出异常。如果忘记调用close(), 可能会导致只写了一部分数据到磁盘而丢失其他的内容。也就是说「大象塞进冰箱后, 一定不要忘记关上冰箱的门」。

with关键字

with关键字用于Python的上下文管理器机制。为了防止诸如open这一类文件打开方法在操作过程出现异常或错误, 或者最后忘了执行close方法, 文件非正常关闭等可能导致文件泄露、破坏的问题。Python提供了with这个上下文管理器机制, 保证文件会被正常关闭。在它的管理下, 不需要再写close语句。注意缩进。

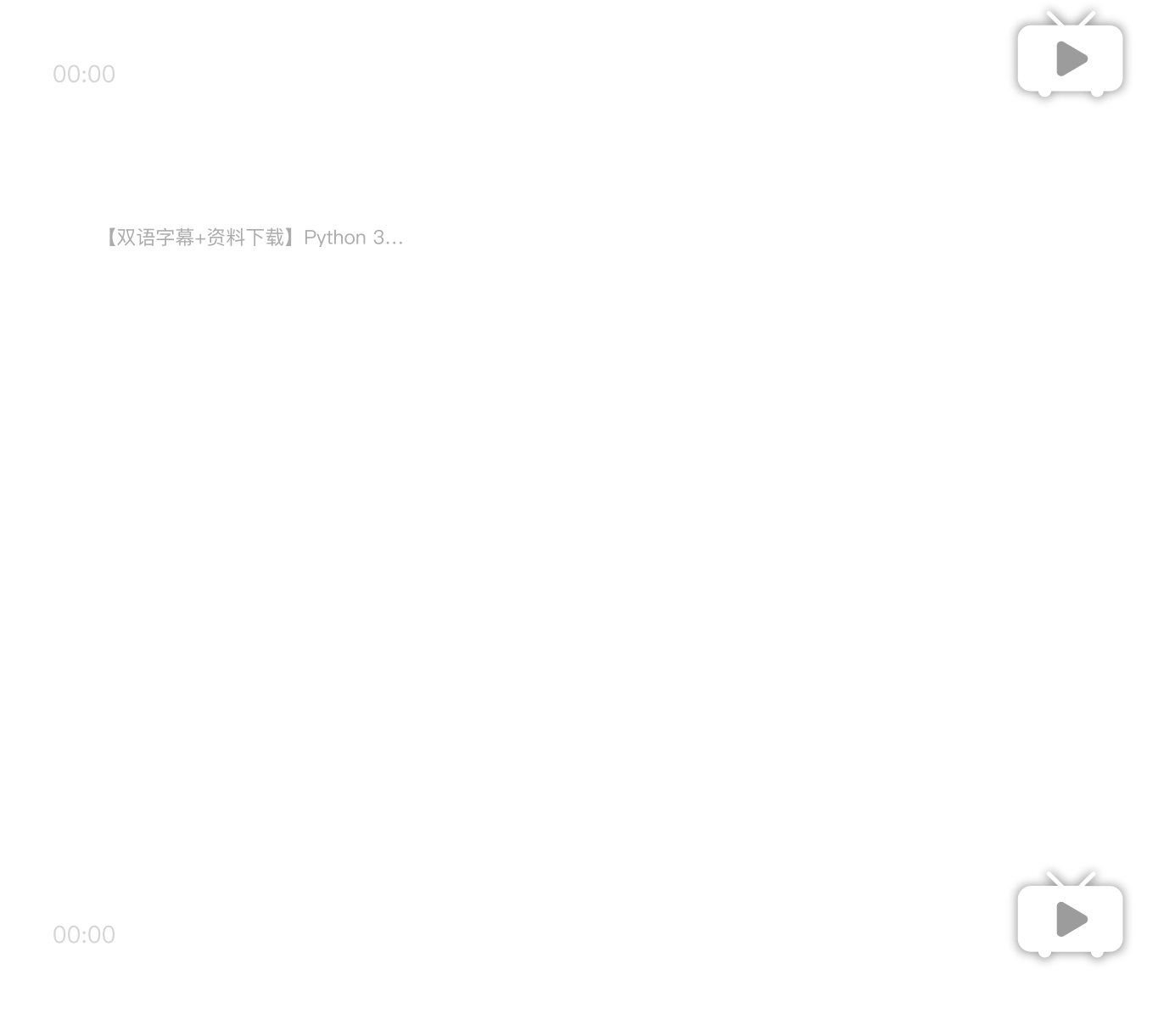
```
1. with open('test.txt', 'w') as f:
2.     f.write('Hello, ShowMeAI!')
```

with支持同时打开多个文件：

```
1. with open('log1') as obj1, open('log2', 'w') as obj2:
2.     s=obj1.read()
3.     obj2.write(s)
```

视频教程

也可以点击 [这里](#) 到B站查看有【中英字幕】的版本




一键运行所有代码

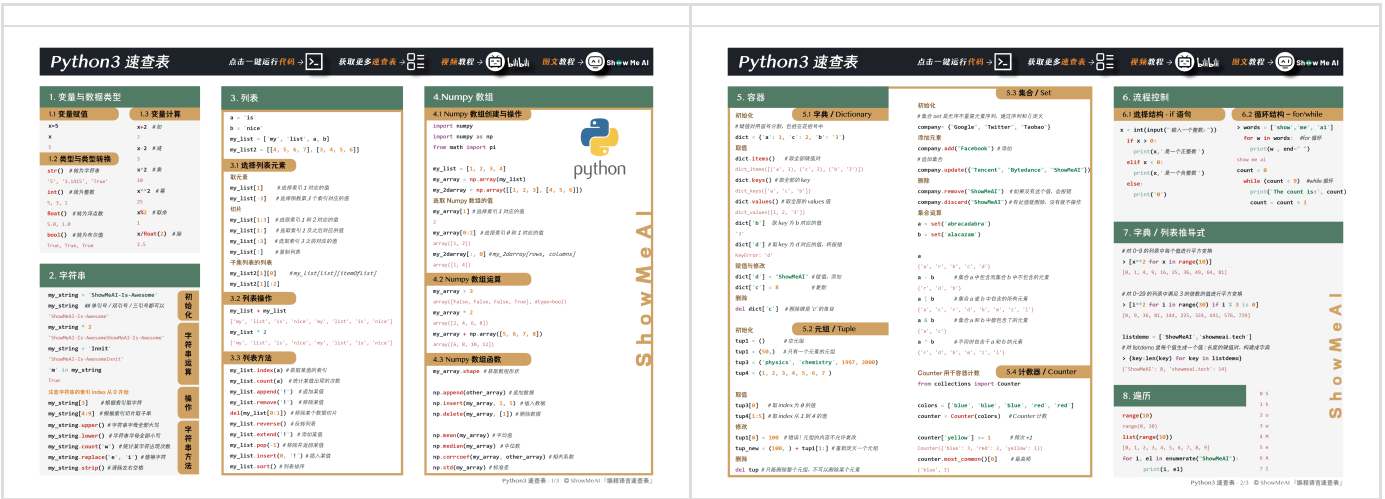
图解Python编程系列 配套的所有代码，可前往ShowMeAI 官方 **GitHub**，下载后即可在本地 Python 环境中运行。能访问 Google 的宝宝也可以直接借助 Google Colab一键运行与交互学习！

下载Python要点速查表

Awesome cheatsheets | **ShowMeAI**速查表大全 系列包含『编程语言』『AI技能知识』『数据科学工具库』『AI垂直领域工具库』四个板块，追平到工具库当前最新版本，并跑通了所有代码。点击 [官网](#) 或 [GitHub](#) 获取~

 ShowMeAI速查表大全

 **Python** 速查表（部分）



拓展参考资料

- Python教程 - Python3文档
- Python教程 - 廖雪峰的官方网站

ShowMeAI图解Python编程系列推荐（要点速查版）

- ShowMeAI 图解 Python 编程(1) | 介绍
- ShowMeAI 图解 Python 编程(2) | 安装与环境配置
- ShowMeAI 图解 Python 编程(3) | 基础语法
- ShowMeAI 图解 Python 编程(4) | 基础数据类型
- ShowMeAI 图解 Python 编程(5) | 运算符
- ShowMeAI 图解 Python 编程(6) | 条件控制与if语句
- ShowMeAI 图解 Python 编程(7) | 循环语句
- ShowMeAI 图解 Python 编程(8) | while循环
- ShowMeAI 图解 Python 编程(9) | for循环
- ShowMeAI 图解 Python 编程(10) | break语句
- ShowMeAI 图解 Python 编程(11) | continue语句
- ShowMeAI 图解 Python 编程(12) | pass语句
- ShowMeAI 图解 Python 编程(13) | 字符串及操作
- ShowMeAI 图解 Python 编程(14) | 列表
- ShowMeAI 图解 Python 编程(15) | 元组
- ShowMeAI 图解 Python 编程(16) | 字典
- ShowMeAI 图解 Python 编程(17) | 集合
- ShowMeAI 图解 Python 编程(18) | 函数
- ShowMeAI 图解 Python 编程(19) | 迭代器与生成器

- [ShowMeAI 图解 Python 编程\(20\) | 数据结构](#)
- [ShowMeAI 图解 Python 编程\(21\) | 模块](#)
- [ShowMeAI 图解 Python 编程\(22\) | 文件读写](#)
- [ShowMeAI 图解 Python 编程\(23\) | 文件与目录操作](#)
- [ShowMeAI 图解 Python 编程\(24\) | 错误与异常处理](#)
- [ShowMeAI 图解 Python 编程\(25\) | 面向对象编程](#)
- [ShowMeAI 图解 Python 编程\(26\) | 命名空间与作用域](#)
- [ShowMeAI 图解 Python 编程\(27\) | 时间和日期](#)

ShowMeAI系列教程精选推荐

- [大厂技术实现：推荐与广告计算解决方案](#)
- [大厂技术实现：计算机视觉解决方案](#)
- [大厂技术实现：自然语言处理行业解决方案](#)
- [图解Python编程：从入门到精通系列教程](#)
- [图解数据分析：从入门到精通系列教程](#)
- [图解AI数学基础：从入门到精通系列教程](#)
- [图解大数据技术：从入门到精通系列教程](#)
- [图解机器学习算法：从入门到精通系列教程](#)
- [机器学习实战：手把手教你玩转机器学习系列](#)
- [深度学习教程：吴恩达专项课程 · 全套笔记解读](#)
- [自然语言处理教程：斯坦福CS224n课程 · 课程带学与全套笔记解读](#)
- [深度学习与计算机视觉教程：斯坦福CS231n · 全套笔记解读](#)

图解 Python 编程(22) | 文件读写

[< 上一篇](#)

[图解 Python 编程\(21\) | 模块](#)

[下一篇 >](#)

[图解 Python 编程\(23\) | 文件与目录操作](#)