# 浅拷贝与深拷贝

浅拷贝和深拷贝都会创建一个新的对象，但是前者不会递归地拷贝内部的嵌套对象。也即对于嵌套对象，新的对象会**引用原始对象中的相通嵌套对象**。具体可以参考下面这段代码：

import copy

original\_list = [1, 2, [3, 4]]

shallow\_copied\_list = copy.copy(original\_list)

# 也可以写shallow\_copied\_list = original\_list[:] 用切片浅拷贝

# 深拷贝使用deep\_copied\_list = copy.deepcopy(original\_list)

shallow\_copied\_list[2][0] = 99

print("Original List:", original\_list)

# 输出: [1, 2, [99, 4]] 如果是深拷贝则输出[1, 2, [3, 4]]

print("Shallow Copied List:", shallow\_copied\_list)

# 输出: [1, 2, [99, 4]]

另外，浅拷贝是引用，深拷贝才是另外开辟一块空间。

# Git的使用

参考：[Git 基本操作 | 菜鸟教程](https://www.runoob.com/git/git-basic-operations.html)

## 上传本地文件

$ git add . # 也可以git add file\_name

$ git commit -m "241029" # 备注这里随便写什么都行

$ git push origin master

三者关系可以参考下面这张图：



注意如果文件夹为空，则不会被add到暂存区，至少需要在文件夹中新建一个文件。

# I/O

## open函数

最基本的使用形式：

with open("test.txt", "r") as file:

pass

# 或者

with open("test.txt", "r") as f:

pass

在打开时需要注意不同的模式：

|  |  |
| --- | --- |
| **模式** | **描述** |
| t | 文本模式 (默认)。 |
| x | 写模式，新建一个文件，如果该文件已存在则会报错。 |
| b | 二进制模式。 |
| + | 打开一个文件进行更新(可读可写)。 |
| U | 通用换行模式（不推荐）。 |
| r | 以只读方式打开文件。文件的指针将会放在文件的开头。这是默认模式。 |
| rb | 以二进制格式打开一个文件用于只读。文件指针将会放在文件的开头。这是默认模式。一般用于非文本文件如图片等。 |
| r+ | 打开一个文件用于读写。文件指针将会放在文件的开头。 |
| rb+ | 以二进制格式打开一个文件用于读写。文件指针将会放在文件的开头。一般用于非文本文件如图片等。 |
| w | 打开一个文件只用于写入。如果该文件已存在则打开文件，并从开头开始编辑，即原有内容会被删除。如果该文件不存在，创建新文件。 |
| wb | 以二进制格式打开一个文件只用于写入。如果该文件已存在则打开文件，并从开头开始编辑，即原有内容会被删除。如果该文件不存在，创建新文件。一般用于非文本文件如图片等。 |
| w+ | 打开一个文件用于读写。如果该文件已存在则打开文件，并从开头开始编辑，即原有内容会被删除。如果该文件不存在，创建新文件。 |
| wb+ | 以二进制格式打开一个文件用于读写。如果该文件已存在则打开文件，并从开头开始编辑，即原有内容会被删除。如果该文件不存在，创建新文件。一般用于非文本文件如图片等。 |
| a | 打开一个文件用于追加。如果该文件已存在，文件指针将会放在文件的结尾。也就是说，新的内容将会被写入到已有内容之后。如果该文件不存在，创建新文件进行写入。 |
| ab | 以二进制格式打开一个文件用于追加。如果该文件已存在，文件指针将会放在文件的结尾。也就是说，新的内容将会被写入到已有内容之后。如果该文件不存在，创建新文件进行写入。 |
| a+ | 打开一个文件用于读写。如果该文件已存在，文件指针将会放在文件的结尾。文件打开时会是追加模式。如果该文件不存在，创建新文件用于读写。 |
| ab+ | 以二进制格式打开一个文件用于追加。如果该文件已存在，文件指针将会放在文件的结尾。如果该文件不存在，创建新文件用于读写。 |

## File对象

当一个文件被打开，就获得了一个file对象，其有常用的函数为：

|  |  |
| --- | --- |
| **序号** | **方法及描述** |
| 1 | [file.close()](https://www.runoob.com/python/file-close.html)  关闭文件。关闭后文件不能再进行读写操作。 |
| 2 | [file.flush()](https://www.runoob.com/python/file-flush.html)  刷新文件内部缓冲，直接把内部缓冲区的数据立刻写入文件, 而不是被动的等待输出缓冲区写入。 |
| 3 | [file.fileno()](https://www.runoob.com/python/file-fileno.html)  返回一个整型的文件描述符(file descriptor FD 整型), 可以用在如os模块的read方法等一些底层操作上。 |
| 4 | [file.isatty()](https://www.runoob.com/python/file-isatty.html)  如果文件连接到一个终端设备返回 True，否则返回 False。 |
| 5 | [file.next()](https://www.runoob.com/python/file-next.html)  返回文件下一行。 |
| 6 | [file.read([size])](https://www.runoob.com/python/python-file-read.html)  从文件读取指定的字节数，如果未给定或为负则读取所有。 |
| 7 | [file.readline([size])](https://www.runoob.com/python/file-readline.html)  读取整行，包括 "\n" 字符。 |
| 8 | [file.readlines([sizeint])](https://www.runoob.com/python/file-readlines.html)  读取所有行并返回列表，若给定sizeint>0，则是设置一次读多少字节，这是为了减轻读取压力。 |
| 9 | [file.seek(offset[, whence])](https://www.runoob.com/python/file-seek.html)  设置文件当前位置 |
| 10 | [file.tell()](https://www.runoob.com/python/file-tell.html)  返回文件当前位置。 |
| 11 | [file.truncate([size])](https://www.runoob.com/python/file-truncate.html)  截取文件，截取的字节通过size指定，默认为当前文件位置。 |
| 12 | [file.write(str)](https://www.runoob.com/python/python-file-write.html)  将字符串写入文件，返回的是写入的字符长度。 |
| 13 | [file.writelines(sequence)](https://www.runoob.com/python/file-writelines.html)  向文件写入一个序列字符串列表，如果需要换行则要自己加入每行的换行符。 |

## 常用到读取文件方法

### 使用csv包

使用这种方法一般是适合CSV等结构化文件，会自动解析和分割每一行，不需要手动指定分隔符：

import csv

with open('example.csv', 'r') as file:

    reader = csv.reader(file, delimiter=',')

    for row in reader:

        print(row)  # 每行会是一个列表，例如 ['col1', 'col2', 'col3']

### 使用file对象方法

这更适合读取普通文本文件，或者不需要逐列解析的情况下使用：

with open('example.csv', 'r') as file:

    for line in file:

        line\_data = line.strip().split(',')  # 手动按逗号分隔

        print(line\_data)