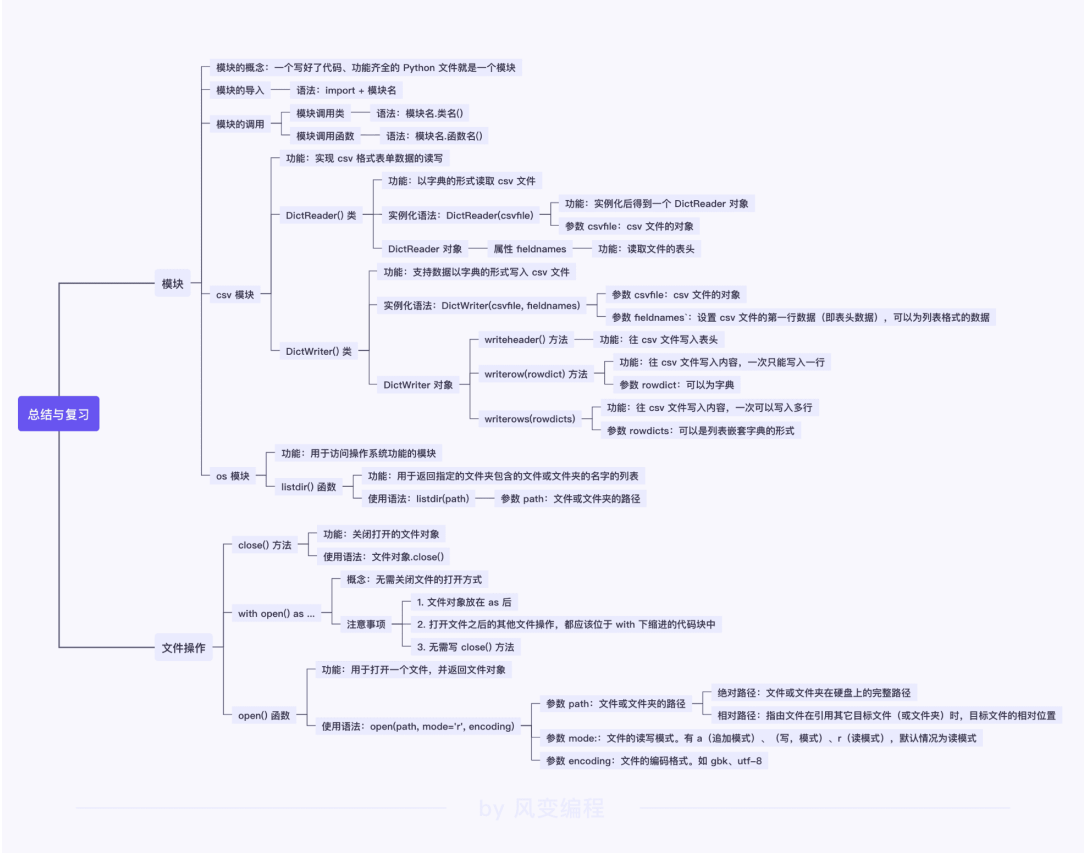
基础语法第8关笔记



一、文件操作

1.open() 函数

`open()` 函数用于打开一个文件，并返回一个 file 对象（即文件对象）。

使用该函数的语法：`open(file, mode, encoding)`。

- 参数file：表示我们要打开的文件的路径。
- 参数mode：决定打开文件的模式，最主要的模式有：只读（r），写（w），追加（a）等。默认值为只读（r）。
- 参数encoding：表示文件的编码方式，课程中使用到的文件编码方式一般为 'utf-8'。

open(file, mode='r', encoding)

功能：打开一个文件，并返回一个 file 对象（即文件对象）

参数	说明	示例
file	表示我们要打开的文件的路径，可以传入相对路径或绝对路径，根据实际需求进行选择	source_path = open('../工作/总年级成绩单.csv', 'r', encoding='utf-8')
mode	决定了打开文件的模式，最主要的模式有：只读（r），写（w），追加（a）等。默认情况为只读模式	
encoding	文件的编码方式	

参数 mode 并不是强制的，它有个默认值 'r'。也就是说，打开文件时，默认只读模式。

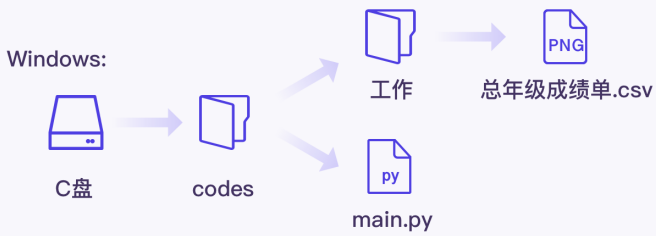
当 mode='w'，即写模式时，open() 函数会打开一个文件只用于写入，如果该文件已存在则打开文件，并从开头开始编辑，即原有内容会被清空。如果该文件不存在，创建新文件。

需要记住的是，当我们要对文件进行读取时，需要使用到 mode='r'，即只读模式打开文件。当需要对文件进行写入时，需要使用到 mode='w' 或 mode='a' 的模式打开文件。

文件路径

文件是通过文件路径定位的，在 Python 中，路径可以用字符串来表示。文件路径分为**绝对路径**和**相对路径**。

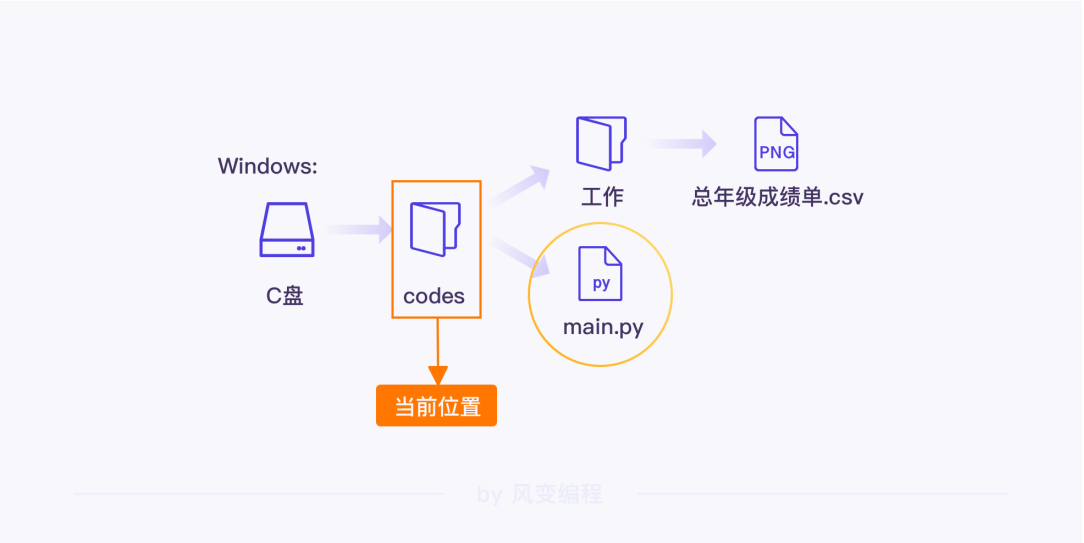
绝对路径就是文件或文件夹在硬盘上的完整路径。永远都是根目录开头，具体的文件或文件夹名称做结尾。




```
# Windows 系统中，C 盘根目录中有个名为 【codes】 的文件夹，该文件夹内有 【工作】 文件夹和 【main.py】 文件，【工作】 文件夹中有 csv 文件 【总年级成绩单.csv】。
# 【工作】文件夹的绝对路径
```

```
'C:/codes/工作/'
# 【main.py】文件的绝对路径
'C:/codes/main.py'
# 【总年级成绩单.csv】文件的绝对路径
'C:/codes/工作/总年级成绩单.csv'
```

相对路径是指相对于当前目录的位置。相对路径使用两个特殊符号：点【.】和双点【..】。点【.】表示文件或文件夹所在的当前目录，双点【..】表示当前目录的上一级目录。




```
# 【工作】 文件夹与 【main.py】 文件都在 【codes】 文件夹中，所以 【工作】 文件夹和 【main.py】 文件的相对路径就可以表示为：
# 【工作】文件夹的相对路径
'./工作/'
# 【main.py】文件的相对路径
'./'
# 【总年级成绩单.csv】文件的绝对路径
'./工作/总年级成绩单.csv'
```

2.close() 函数

close() 函数常与 open() 函数配套使用，用于关闭打开的文件对象。

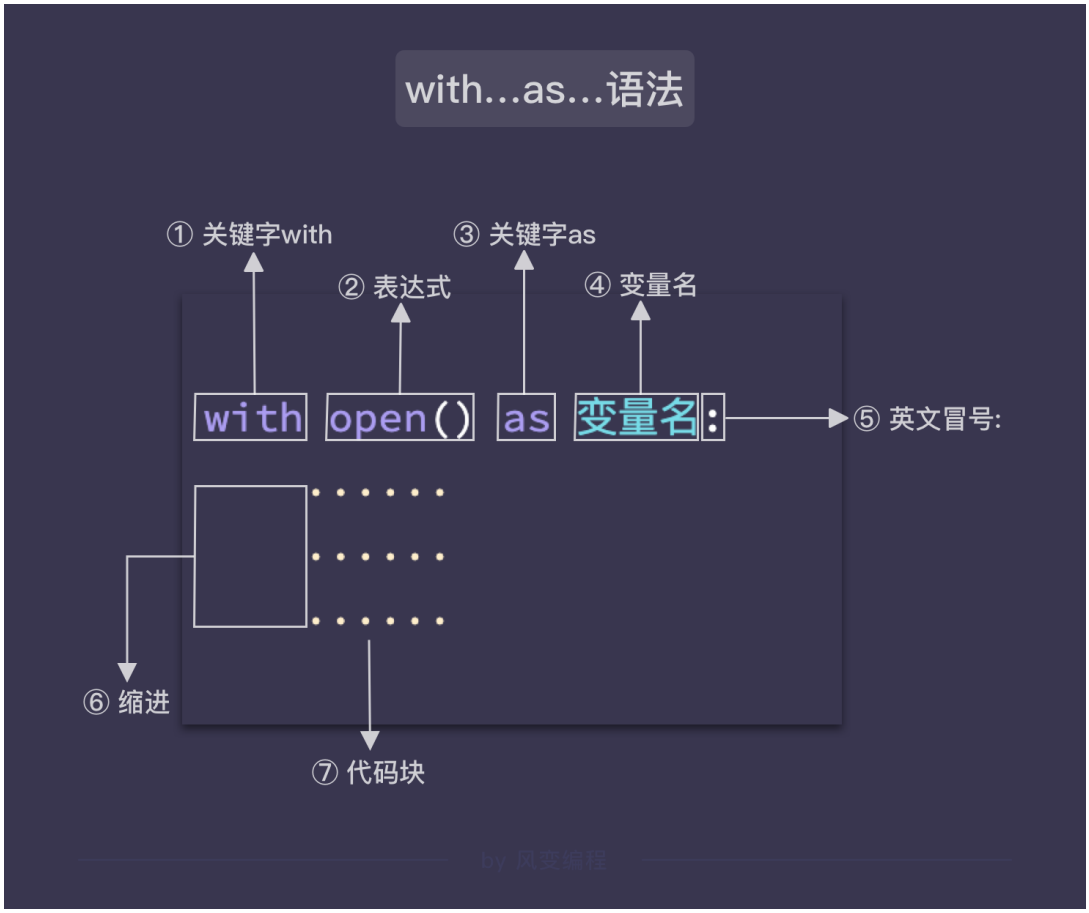
使用语法：`文件对象.c` `lo` `se()`

```
# 设置【1班成绩单.csv】文件的路径
file_path = './工作/各班级成绩/1班成绩单.csv'
# 使用 open() 函数打开 【1班成绩单.csv】 文件
source_path = open(file_path, 'r', encoding='utf-8')
# 关闭 file_path 文件对象
file_path.close()
```

2.with open() as ...

`with open() as ...` 是对原有 open() 和 close() 的优化。 使用 `with open() as ...` 语句后，在 with 下面的代码块结束时，会自动执行 close() 关闭文件。

用法是把 `open()` 函数放在 with 后面，把变量名放在 as 后面，结束时要加冒号 `:`，然后把要执行的文件处理语句缩进到 `with open() as ...` 下方的代码块中。



请务必记住缩进，打开文件之后的其他文件操作，都应该位于 `with` 下缩进的代码块中。

二、模块与库

1.模块/库的概念

模块本质上就是一个单独的 Python 文件，一个写好了代码、功能齐全的 Python 文件就是一个模块。模块中可以包含多个变量、函数、类等。

而具有相关功能模块的集合，就形成了**库**，库分为标准库和第三方库。

- 标准库：指 Python 中内置的模块的集合，不需要安装就可以导入使用。
- 第三方库：指由第三方发布的库，需要下载安装到本地的 Python 中，才能够导入使用。

2.模块的使用方式

导入模块

模块的导入语法为：`import + 模块名`。（需要留意的是，语法中 import 语句与模块名之间有个空格）

```
import os
```

直接导入模块内的类或函数：`from 模块名 import 类/函数名`。（温馨提示：import 之后的类名或函数名，只需要类或函数的名称，不需要加上 `()`）

```
from os import listdir
```

调用模块内容

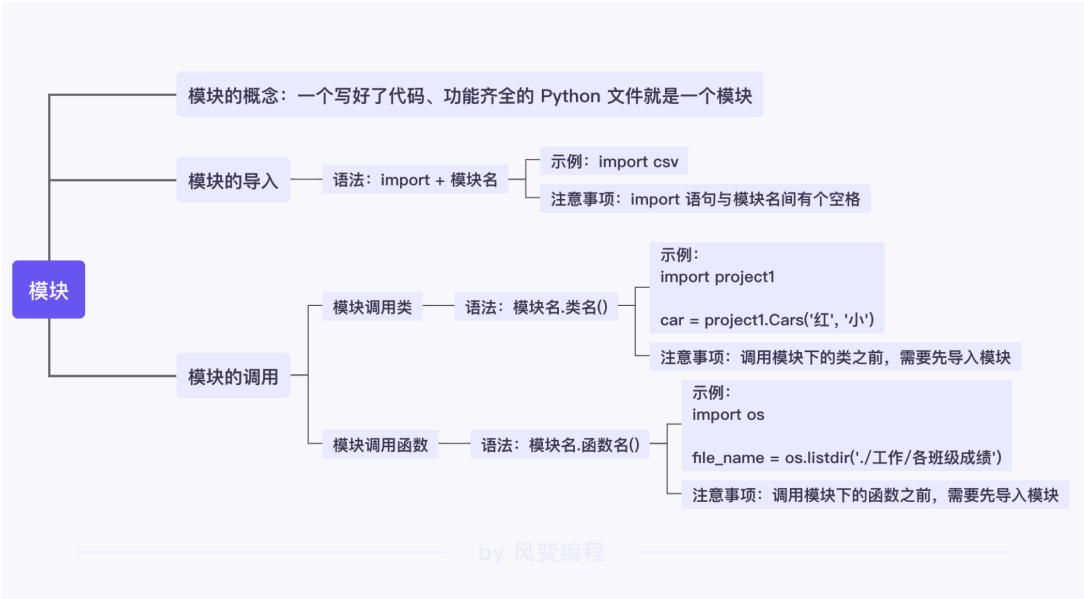
如果想要调用模块中的某一个类、函数或变量，其调用语法如下：

1）调用模块下的类：`模块名.类名()`；

2）调用模块下的函数：`模块名.函数名()`；

3）调用模块下的变量：`模块名.变量名`。

```
import os
file_name = os.listdir('./工作/各班级成绩')
print(file_name)
```



3.os模块

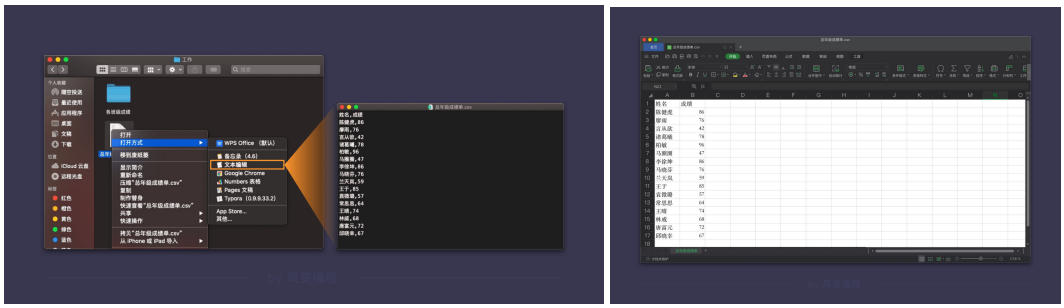
os 模块多用于访问操作系统功能。模块下的 `listdir()` 函数用于返回指定文件夹包含的文件或文件夹的名字的列表。

`listdir()` 函数 的语法是：os.listdir(path)，作用是获取某个文件夹中的所有文件、文件夹名称，需要传入指定文件夹的文件路径。

4.csv模块

csv文件

csv 是一种文件格式，是“Comma-Separated Values（逗号分割的值）”的首字母缩写，其文件以纯文本形式存储表格数据。可以通过纯文本打开，也可以通过 Excel 软件打开。

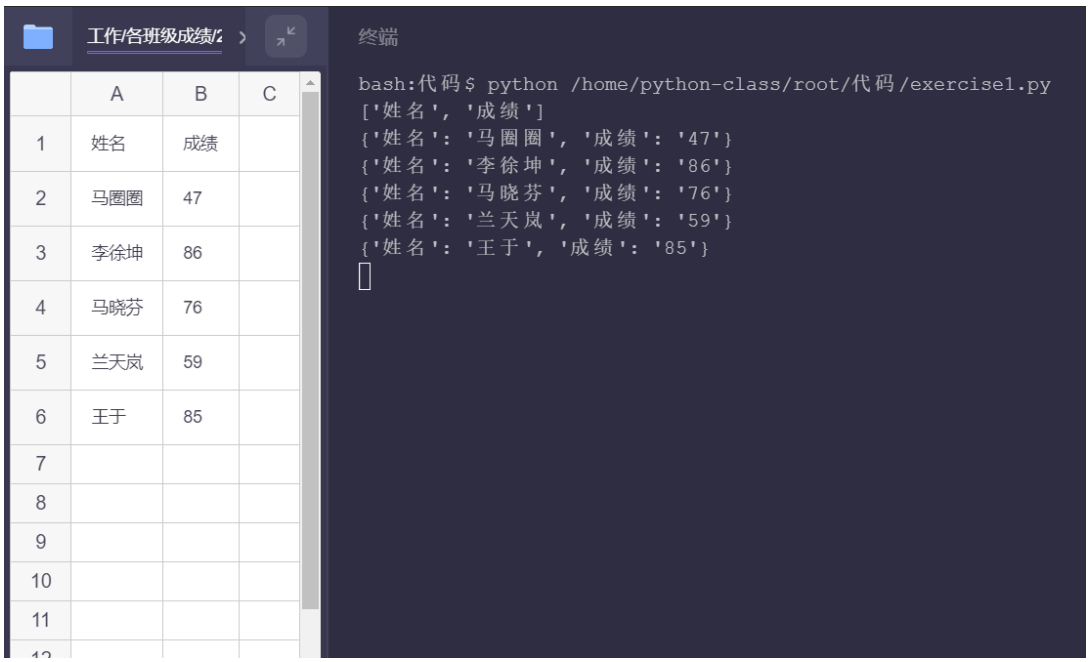


csv 文件与 Excel 文件主要区别在于 csv 文件只是纯文本文件，不包含文字排版格式、单元格样式等信息，所以比 Excel 文件占用的内存更少，读取速度也更快。

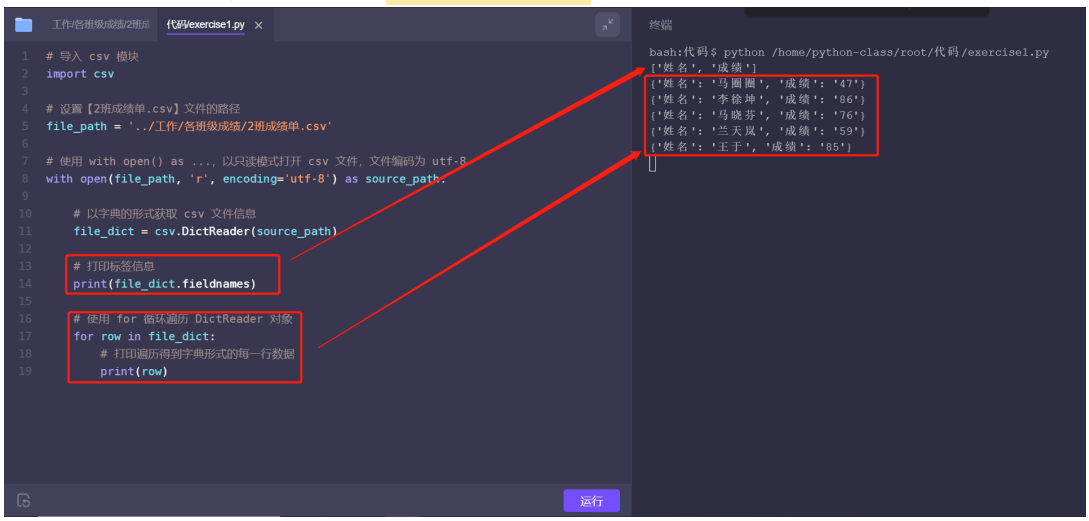
Python 自带 csv 模块，专门用于处理 csv 文件的读取和写入。

DictReader() 类

DictReader() 类可以以字典的形式读取 csv 文件。以 csv 文件首行标签为键，以每一行的内容作为各个值的形式。



类 DictReader() 的实例化语法为：`DictReader(f)`，其中参数 `f` 为 csv 文件的对象。实例化 DictReader() 后，会得到一个 DictReader 对象。




```
# 导入 csv 模块
```

```
import csv
# 设置【2班成绩单.csv】文件的路径
file_path = '../工作/各班级成绩/2班成绩单.csv'
# 使用 with open() as ...，以只读模式打开 csv 文件，文件编码为 utf-8
with open(file_path, 'r', encoding='utf-8') as source_path:
    # 以字典的形式获取 csv 文件信息
    file_dict = csv.DictReader(source_path)
    # 打印标签信息
    print(file_dict.fieldnames)

    # 使用 for 循环遍历 DictReader 对象
    for row in file_dict:
        # 打印遍历得到字典形式的每一行数据
        print(row)
```

需要注意的是：DictReader 对象无法直接打印显示，需要通过 for 循环遍历每一行信息显示；`DictReader 对象.fi` `e` `ldn` `a` `mes` 可以获取 csv 文件的标签信息，也是 DictReader 对象的键信息。

DictWriter() 类

DictWriter() 类支持数据以字典的形式写入 csv 文件。

类 DictWriter() 的实例化语法为：`DictWriter(f, fieldnames)` 。

- 参数 `f` 为 csv 文件的对象；
- 参数 `fieldnames` 的作用是定义文件的表头，其数据可以为列表格式。

```
# 导入 csv 模块
import csv
# 新建两个字典
dict1 = {'姓名': '王大帅', '身高': '176'}
dict2 = {'姓名': '许漂亮', '身高': '177'}
# 设置文件的表头
head = ['姓名', '身高']
# 设置总文件路径
file_path = '../工作/学生体检表.csv'
# 以自动关闭的方式打开文件
with open(file_path, 'w', encoding='utf-8') as f:

    # 实例化类 DictWriter()，得到 DictWriter 对象
    dict_write = csv.DictWriter(f, fieldnames=head)
    # 写入文件的表头
    dict_write.writeheader()
    # 写入文件的多行内容
    dict_write.writerows([dict1, dict2])
```

DictWriter() 类的写入流程可以分为：

1）定义文件表头；

2）以字典的形式定义需要写入的内容；

3）以‘w’模式打开指定路径文件，文件以.csv为后缀名；
4）实例化 DictWriter 对象，传入文件表头信息参数，定义表头信息；
5）通过 DictWriter对象.writeheader() 方法写入表头信息；
6）通过 DictWriter对象.writerow(dict) 或 DictWriter对象.writerows(list) 方法写入内容。