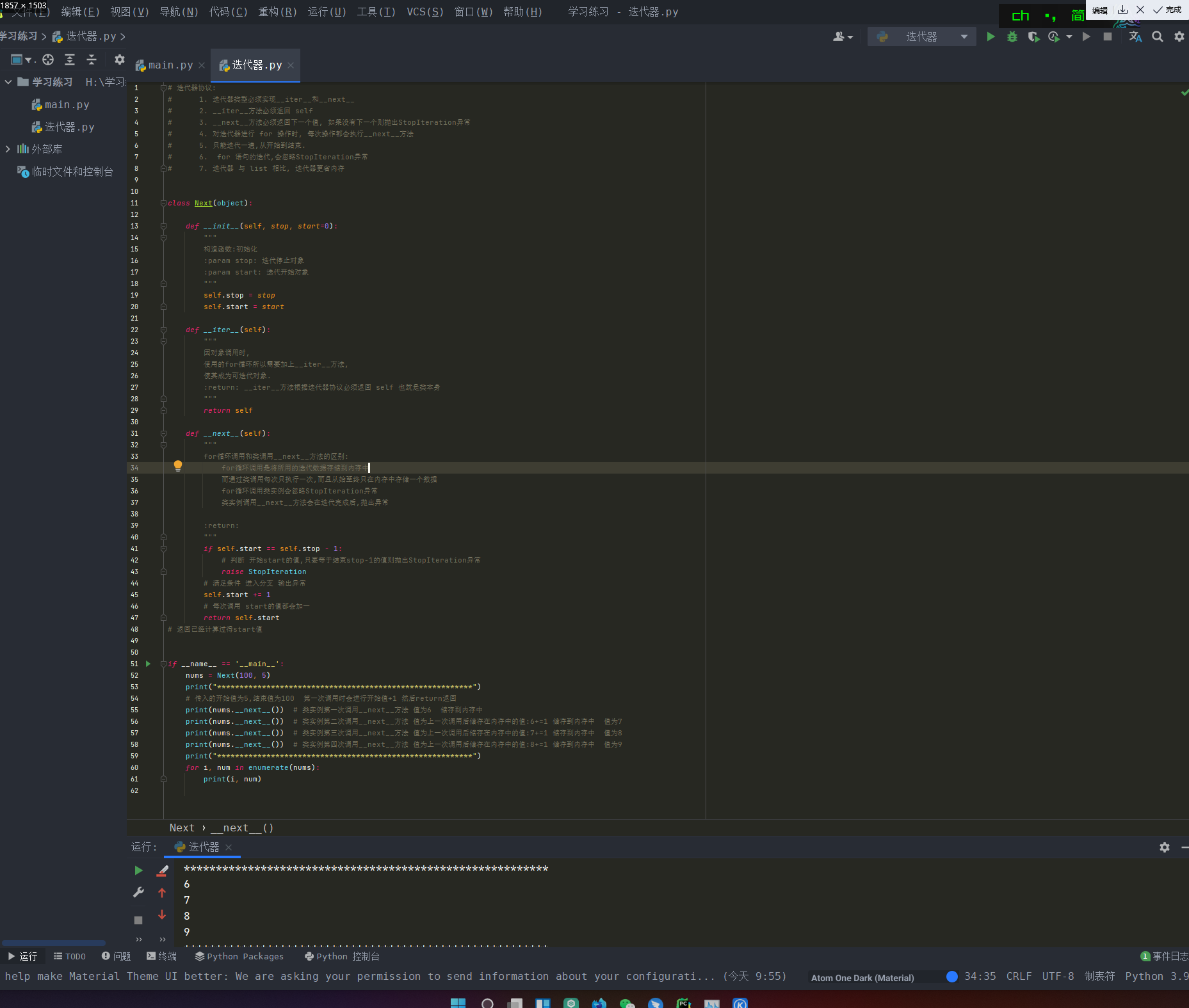
# 迭代器:



迭代器类型: Iterator

如何判断某个对象是不是迭代器?

答案: 有两种方法:

第一种:使用isinstance(对象,类型)来判断但是这个有个方法需要先导包<typing>

第二种: 看对象有没有\_\_iter\_\_属性和\_\_next\_\_属性 两个属性均具备才是迭代器

迭代器协议:

1. 迭代器类型必须实现\_\_iter\_\_和\_\_next\_\_

2. \_\_iter\_\_方法必须返回 self

3. \_\_next\_\_方法必须返回下一个值, 如果没有下一个则抛出StopIteration异常

4. 对迭代器进行 for 操作时, 每次操作都会执行\_\_next\_\_方法

5. 只能迭代一遍,从开始到结束.

6. for 语句的迭代,会忽略StopIteration异常

7. 迭代器 与 list 相比, 迭代器更省内存

如何判断一个对象是不是可迭代对象

有两种方法:

方法一: isinstance(obj,Iterable) ---🡪 obj == 对象, Iterable == 可迭代类型

方法二: 看有没有\_\_iter\_\_方法

# 生成器:

有3种方法:

第一种需要我们自己去编写迭代器.如迭代器中的图

第一种:



第二种:



当一个函数中有 yield 关键字那么这个函数就是一个普通普通的函数函数内的代码不会执行,只会只有yield代码

如果一个函数中有yield 关键字,调用函数的时候不会执行函数的内容,会返回一个对象(这个对象是生成器类)

yield return 区别:

共同点: 都是Python关键字

不同点: return是结束函数并返回值, yield是暂时离开函数

当要访问生成器的\_\_next\_\_方法时,函数会变成running(运行状态),当执行完yield时,函数变成非running状态(即挂起)只有再次执行生成器对象的\_\_next\_\_方法时,函数才会被唤醒.