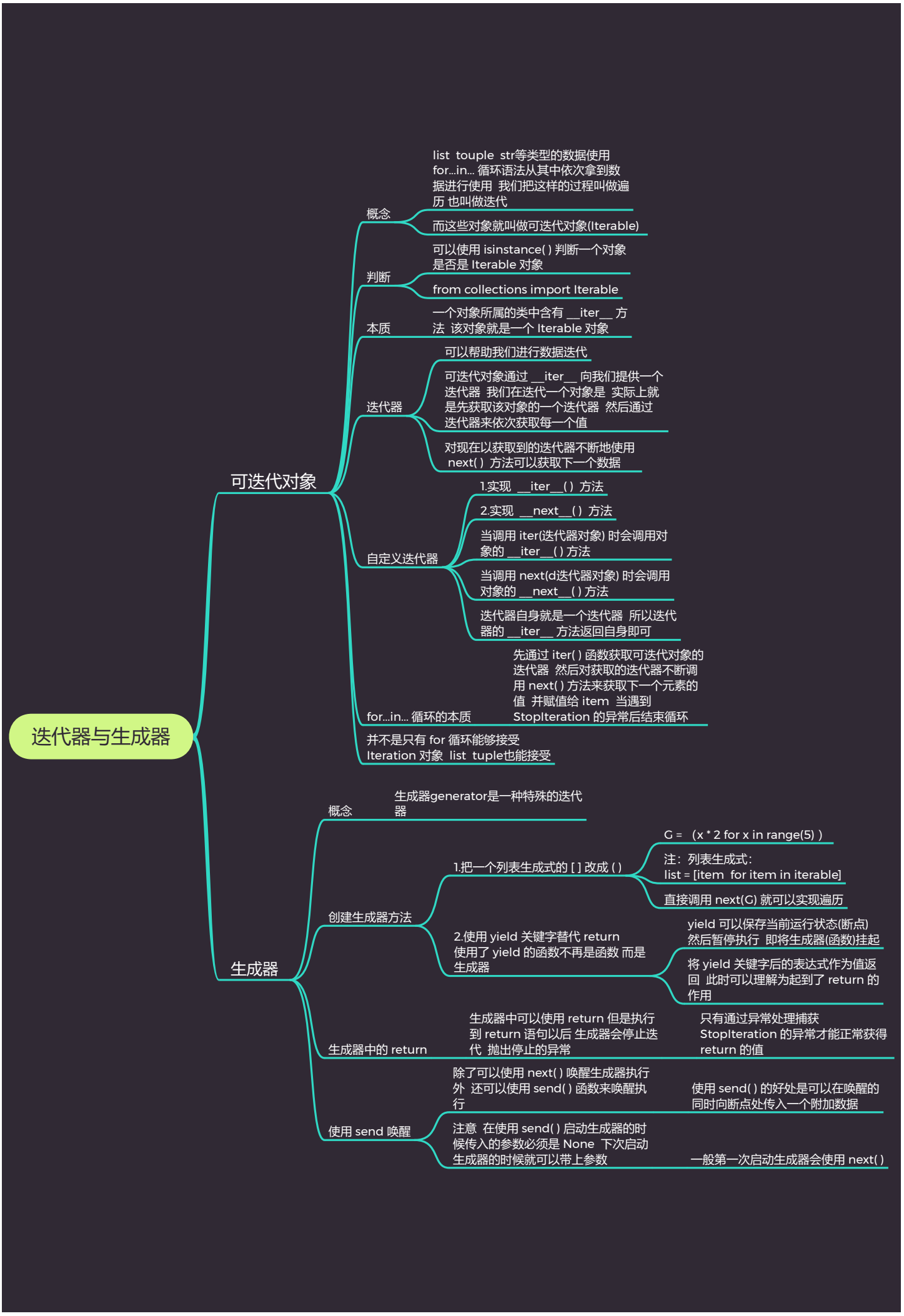
迭代器与生成器
可迭代对象
概念
list touple str等类型的数据使用for...in... 循环语法从其中依次拿到数据进行使用 我们把这样的过程叫做遍历 也叫做迭代
而这些对象就叫做可迭代对象(Iterable)
判断
可以使用 isinstance( ) 判断一个对象是否是 Iterable 对象
from collections import Iterable
本质
一个对象所属的类中含有 __iter__ 方法 该对象就是一个 Iterable 对象
迭代器
可以帮助我们进行数据迭代
可迭代对象通过 __iter__ 向我们提供一个迭代器 我们在迭代一个对象是 实际上就是先获取该对象的一个迭代器 然后通过迭代器来依次获取每一个值
对现在以获取到的迭代器不断地使用 next( ) 方法可以获取下一个数据
自定义迭代器
1.实现 __iter__( ) 方法
2.实现 __next__( ) 方法
当调用 iter(迭代器对象) 时会调用对象的 __iter__( ) 方法
当调用 next(d迭代器对象) 时会调用对象的 __next__( ) 方法
迭代器自身就是一个迭代器 所以迭代器的 __iter__ 方法返回自身即可
for...in... 循环的本质
先通过 iter( ) 函数获取可迭代对象的迭代器 然后对获取的迭代器不断调用 next( ) 方法来获取下一个元素的值 并赋值给 item 当遇到 StopIteration 的异常后结束循环
并不是只有 for 循环能够接受 Iteration 对象 list tuple也能接受
生成器
概念
生成器generator是一种特殊的迭代器
创建生成器方法
1.把一个列表生成式的 [ ] 改成 ( )
G = (x * 2 for x in range(5))
注：列表生成式：
list = [item for item in iterable]
直接调用 next(G) 就可以实现遍历
2.使用 yield 关键字替代 return 使用了 yield 的函数不再是函数 而是生成器
yield 可以保存当前运行状态(断点) 然后暂停执行 即将生成器(函数)挂起
将 yield 关键字后的表达式作为值返回 此时可以理解为起到了 return 的作用
生成器中的 return
生成器中可以使用 return 但是执行到 return 语句以后 生成器会停止迭代 抛出停止的异常
只有通过异常处理捕获 StopIteration 的异常才能正常获得 return 的值
使用 send 唤醒
除了可以使用 next( ) 唤醒生成器执行外 还可以使用 send( ) 函数来唤醒执行
使用 send( ) 的好处是可以在唤醒的同时向断点处传入一个附加数据
注意 在使用 send( ) 启动生成器的时候传入的参数必须是 None 下次启动生成器的时候就可以带上参数
一般第一次启动生成器会使用 next( )