Python 八股文

Python为我们提供了非常完善的基础代码库，覆盖了网络、文件、GUI、数据库、文本等大量内容，被形象地称作“内置电池（batteries included）”。用Python开发，许多功能不必从零编写，直接使用现成的即可。

除了内置的库外，Python还有大量的第三方库，也就是别人开发的，供你直接使用的东西。当然，如果你开发的代码通过很好的封装，也可以作为第三方库给别人使用。

什么是 Python 生成器?

generator，有两种产生生成器对象的方式：一种是列表生成式加括号：

g1 = (x for x in range(10))

一种是在函数定义中包含yield关键字：

```
def fib(max):
    n, a, b = 0, 0, 1
    while n < max:        yield b
        a, b = b, a + b
        n = n + 1
    return 'done'g2 = fib(8)

```

对于generator对象g1和g2，可以通过next(g1)不断获得下一个元素的值，如果没有更多的元素，就会报错StopIteration

也可以通过for循环获得元素的值。

生成器的好处是不用占用很多内存，只需要在用的时候计算元素的值就行了。

什么是 Python 迭代器?

Python中可以用于for循环的，叫做可迭代Iterable，包括list/set/tuple/str/dict等数据结构以及生成器；可以用以下语句判断一个对象是否是可迭代的：

```
from collections import Iterableisinstance(x, Iterable)
```

迭代器Iterator，是指可以被next()函数调用并不断返回下一个值，直到StopIteration；生成器都是Iterator，而列表等数据结构不是；

可以通过以下语句将list变为Iterator：

```
iter([1,2,3,4,5])

```

生成器都是Iterator，但迭代器不一定是生成器。

list 和 tuple 有什么区别?

- list 长度可变，tuple不可变；
- list 中元素的值可以改变，tuple 不能改变；
- list 支持append; insert; remove; pop等方法，tuple 都不支持

Python 中的 list 和 dict 是怎么实现的?

List： 本质是顺序表，只不过每次表的扩容都是指数级，所以动态增删数据时，表并不会频繁改变物理结构，同时受益于顺序表遍历的高效性（通过角标配合表头物理地址，计算目标元素的位置），使得python的list综合性能比较优秀

dict： 本质上是顺序表，不过每个元素存储位置的角标，不是又插入顺序决定的，而是由key经过hash算法和其他机制，动态生成的，即key通过hash散列，生成value应该存储的位置，然后再去存储这个value；所以dict的查询时间复杂度是o（1）；

因此，dict的key只能为可hash的对象，即不可变类型；

Python 中使用多线程可以达到多核CPU一起使用吗?

Python中有一个被称为Global Interpreter Lock（GIL）的东西，它会确保任何时候你的多个线程中，只有一个被执行。

线程的执行速度非常之快，会让你误以为线程是并行执行的，但是实际上都是轮流执行。经过GIL这一道关卡处理，会增加执行的开销。

可以通过多进程实现多核任务。

py3和py2的区别

print在py3里是一个函数，在py2里只是一个关键字

py3文件的默认编码是utf8，py2文件的默认编码是ascii

py3的str是unicode字符串，而py2的str是bytes

py3的range()返回一个可迭代对象，py2的 range()返回一个列表，xrange()返回一个可迭代对象,py3的除法返回float，py2的除法返回int

可变对象与不可变对象

可变对象: list，dict，set

不可变对象: bool，int，float，tuple，str...

迭代器与可迭代对象的区别

可迭代对象类，必须自定义__iter__()魔法方法，range，list类的实例化对象都是可迭代对象

迭代器类，必须自定义__iter__()和__next__()魔法方法，用iter()函数可以创建可迭代对象的迭代器

闭包

闭包就是一个嵌套函数，它的内部函数 使用了 外部函数的变量或参数,它的外部函数 返回了 内部函数

可以保存外部函数内的变量，不会随着外部函数调用完而销毁

什么是装饰器?

装饰器是一个接收函数作为参数的闭包函数

它可以在不修改函数内部源代码的情况下，给函数添加额外的功能

```
import time
def calc_time(func):
    def inner():
        t1 = time.time()
        func()
        t2 = time.time()
        print('cost time: {}s'.format(t2-t1))
    return inner

```

什么是元类? 使用场景

元类是创建类的类，type还有继承自type的类都是元类

作用: 在类定义时（new, init）和 类实例化时(call) 可以添加自定义的功能

使用场景: ORM框架中创建一个类就代表数据库中的一个表，但是定义这个类时为了统一需要把里面的类属性全部改为小写，这个时候就要用元类重写new方法，把attrs字典里的key转为小写

GIL(Global Interpreter Lock)

全局解释器锁

全局解释器锁(Global Interpreter Lock)是计算机程序设计语言解释器用于同步线程的一种机制，它使得任何时刻仅有一个线程在执行。

即便在多核处理器上，使用 GIL 的解释器也只允许同一时间执行一个线程，常见的使用 GIL 的解释器有CPython与Ruby MRI。可以看到GIL并不是Python独有的特性，是解释型语言处理多线程问题的一种机制而非语言特性。

GIL的设计初衷?

单核时代高效利用CPU, 针对解释器级别的数据安全(不是thread-safe 线程安全)。首先需要明确的是GIL并不是Python的特性，它是在实现Python解析器(CPython)时所引入的一个概念。

当Python虚拟机的线程想要调用C的原生线程需要知道线程的上下文，因为没有办法控制C的原生线程的执行，所以只能把上下文关系传给原生线程，同理获取结果也是线 程在python虚拟机这边等待。那么要执行一次计算操作，就必须让执行程序的线程组串行执行。

为什么要加在解释器,而不是在其他层?

展开 GIL锁加在解释器一层，也就是说Python调用的Cython解释器上加了GIL锁，因为你python调用的所有线程都是原生线程。原生线程是通过C语言提供原生接口，相当于C语言的一个函数。

你一调它，你就控制不了了它了，就必须等它给你返回结果。只要已通过python虚拟机 ，再往下就不受python控制了，就是C语言自己控制了。

加在Python虚拟机以下加不上去，只能加在Python解释器这一层。

GIL的实现是线程不安全?为什么?

python2.x和3.x都是在执行IO操作的时候，强制释放GIL，使其他线程有机会执行程序。

Python2.x Python使用计数器ticks计算字节码，当执行100个字节码的时候强制释放GIL，其他线程获取GIL继续执行。ticks可以看作是Python自己的计数器，专门作用于GIL，释放后归零，技术可以调整。

Python3.x Python使用计时器，执行时间达到阈值后，当前线程释放GIL。总体来说比Python3.x对CPU密集型任务更好，但是依然没有解决问题。

什么是 lambda 表达式?

简单来说，lambda表达式通常是当你需要使用一个函数，但是又不想费脑袋去命名一个函数的时候使用，也就是通常所说的匿名函数。

lambda表达式一般的形式是：关键词lambda后面紧接一个或多个参数，紧接一个冒号“：”，紧接一个表达式

什么是深拷贝和浅拷贝?

赋值（=），就是创建了对象的一个新的引用，修改其中任意一个变量都会影响到另一个。

浅拷贝 copy.copy：创建一个新的对象，但它包含的是对原始对象中包含项的引用（如果用引用的方式修改其中一个对象，另外一个也会修改改变）

深拷贝：创建一个新的对象，并且递归的复制它所包含的对象（修改其中一个，另外一个不会改变）{copy模块的deep.deepcopy()函数}