一、python解释器

编写完python后会生成一个.py的文件，需要用python解释器去执行.py文件

python解释器五类：

CPython：c语言实现，使用字节码的解释器，运行代码前都要先编译成字节码

IPython:交互方式强，执行方式和CPytho完全一样，是基于CPython的交互式解释器

PyPy:执行速度快，对python代码进行动态编译

JPython:运行在Java平台上的解释器，可以把python代码编译成Java字节码执行

IronPython:运行在微软.net平台的解释器，可以把python代码编译成.net字节码

二、python模块

四种：解释器内建模块、python标准库、第三方模块、自定义模块

常用的几大模块：

time、romdom、sys（与python解释器交互的一个接口）、os（与操作系统交互的一个接口）

#随机小数 random.random()      # 大于0且小于1之间的小数

sys.path、sys.exit(0)、sys.version

os.makedir(‘dirname’)

三、python装饰器应用场景

装饰本质上是一个Python函数，它能使其他函数在没有任何代码变化的情况下增加额外的功能

可以把装饰器理解为一个包装函数的函数，它一般将传入的函数或者是类做一定的处理，返回修改之后的对象．所以，我们能够在不修改原函数的基础上，在执行原函数前后执行别的代码

可以应用的场景：身份的认证、日志记录、合理性检查、缓存装饰器、时间模块、数据库的打开和关闭

<https://wenku.baidu.com/view/69a14dcde309581b6bd97f19227916888586b95f.html>

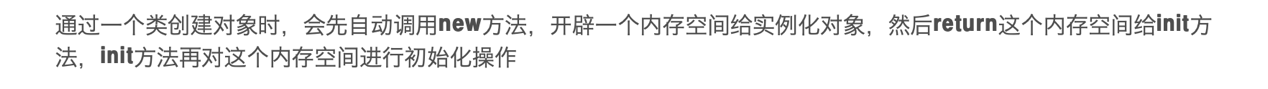
<http://c.biancheng.net/view/5410.html>

日志记录同样是很常见的一个案例。在实际工作中，如果你怀疑某些函数的耗时过长，导致整个系统的延迟增加，想在线上测试某些函数的执行时间，那么，装饰器就是一种很常用的手段。

在大型公司的机器学习框架中，调用机器集群进行模型训练前，往往会用装饰器对其输入（往往是很长的 json 文件）进行合理性检查。这样就可以大大避免输入不正确对机器造成的巨大开销。

LRU cache，在 Python 中的表示形式是 @lru\_cache。@lru\_cache 会缓存进程中的函数参数和结果，当缓存满了以后，会删除最近最久未使用的数据。正确使用缓存装饰器，往往能极大地提高程序运行效率。举个例子，大型公司服务器端的代码中往往存在很多关于设备的检查，比如使用的设备是安卓还是 iPhone，版本号是多少。这其中的一个原因，就是一些新的功能，往往只在某些特定的手机系统或版本上才有

四、init和new的区别



五、生成器迭代器的区别

当你需要一个类除了有生成器的特性之外还要有一些自定义的方法时，可以使用自定义的迭代器，一般来说生成器更方便，更简单