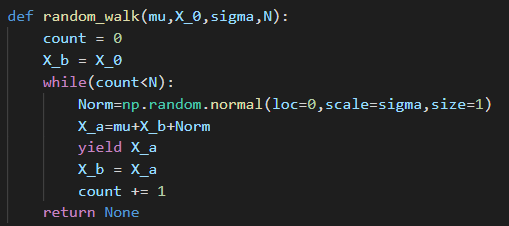
# 第九次作业

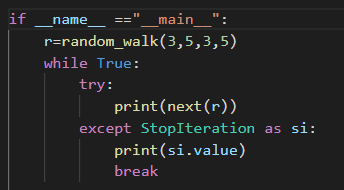
王舵 19377163

生成器和迭代器有两种常见的使用场景。

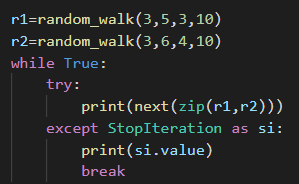
1. 后项需要前项导出，且无法通过列表推导式生成。例如，时间序列中的“随机游走”便是一种满足上述条件的序列数据。其公式为$$X\_t = \mu + X\_{t-1} + w\_t$$，其中$\mu$为漂移量，$w\_{t}$是满足某种条件的独立同分布的随机变量，这里假设其服从正态分布N(0, $\sigma^2$)。本题要求写出实现该功能的迭代器函数。具体要求如下：

1. 实现random\_walk生成器，输入参数$\mu$, $X\_0$, $\sigma^2$，$N$，函数将迭代返回N个随机游走生成的变量。

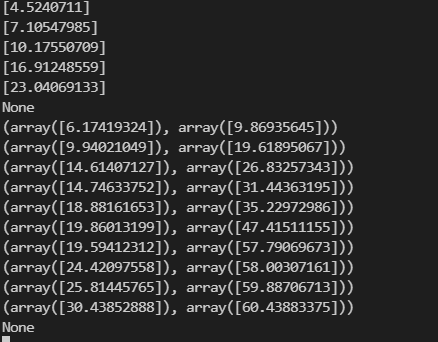




2. 利用zip，实现拼合多个random\_walk的生成器，以生成一组时间上对齐的多维随机游走序列



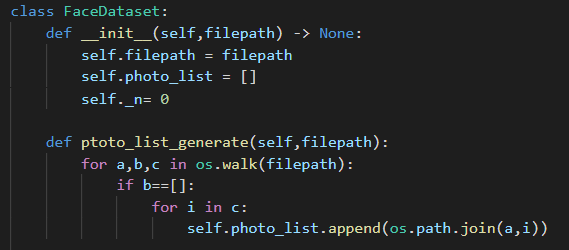
测试结果：



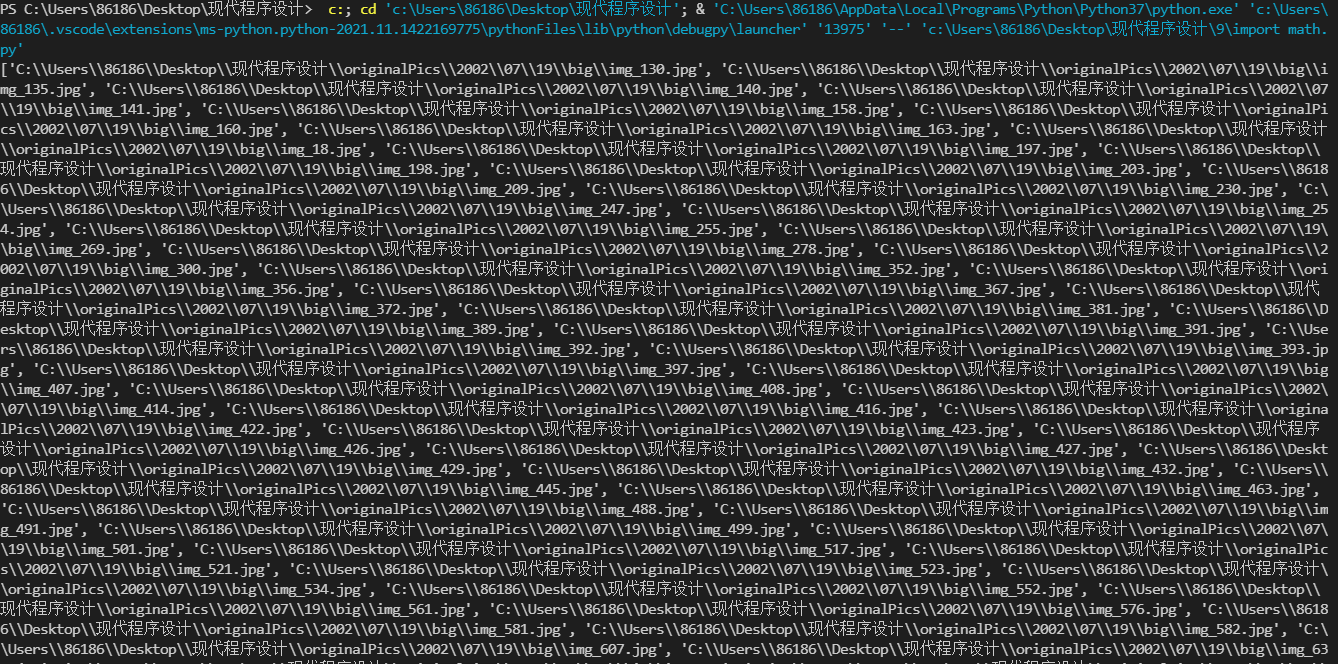
二. 需要迭代的内容数据量过大，无法一次性加载。例如，在图像相关的深度学习任务中，由于数据总量过大，一次性加载全部数据耗时过长、内存占用过大，因此一般会采用批量加载数据的方法。（注：实际应用中由于需要进行采样等操作，通常数据加载类的实现原理更接近字典，例如pytorch中的Dataset类。）现提供文件FaceImages.zip(http://vis-www.cs.umass.edu/fddb/originalPics.tar.gz，其中包含5000余张人脸图片。要求设计FaceDataset类，实现图片数据的加载。具体要求：

1. 类接收图片路径列表

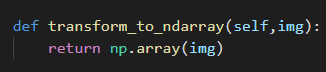
根据路径遍历所有目录，并将所有图片路径添加到列表中



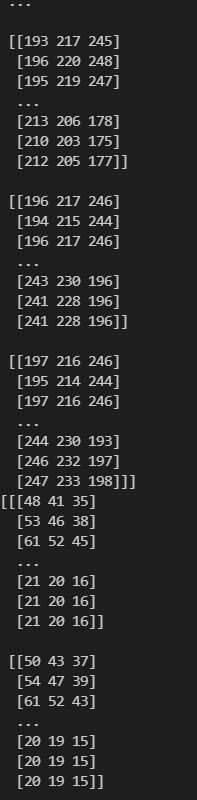
列表如下：共有28000张左右图片



1. 类支持将一张图片数据以ndarray的形式返回（可以利用PIL库实现）。



运行结果：

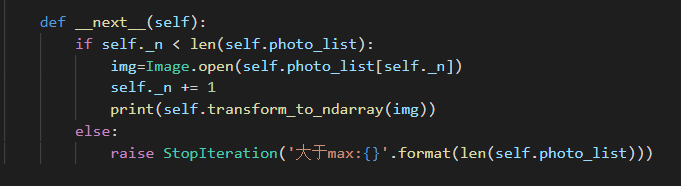


1. 实现\_\_iter\_\_方法。

返回定义了\_\_next\_\_的对象



1. 实现\_\_next\_\_方法，根据类内的图片路径列表，迭代地加载并以ndarray形式返回图片数据。



测试结果（只输出前100张）：

