实验**1**：**PyTorch** 使用简介

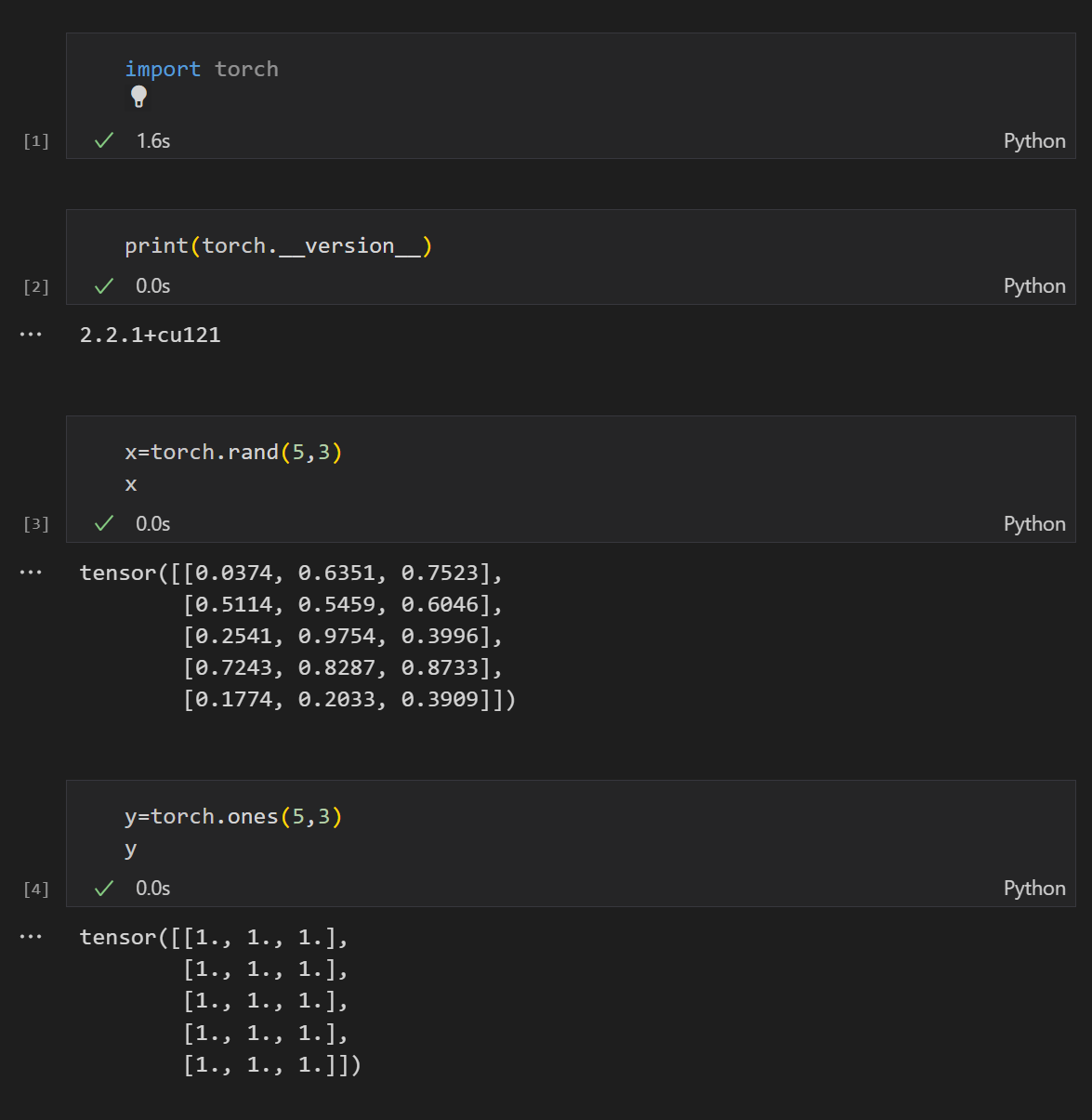
1. 实验内容

实验内容：学习PyTorch 中Tensor的用法，以及简单的自动微分变量原理。

实验环境：Python-3.11;PyTorch-cu121;CUDA-12.1;VS code/JupyterNotebook

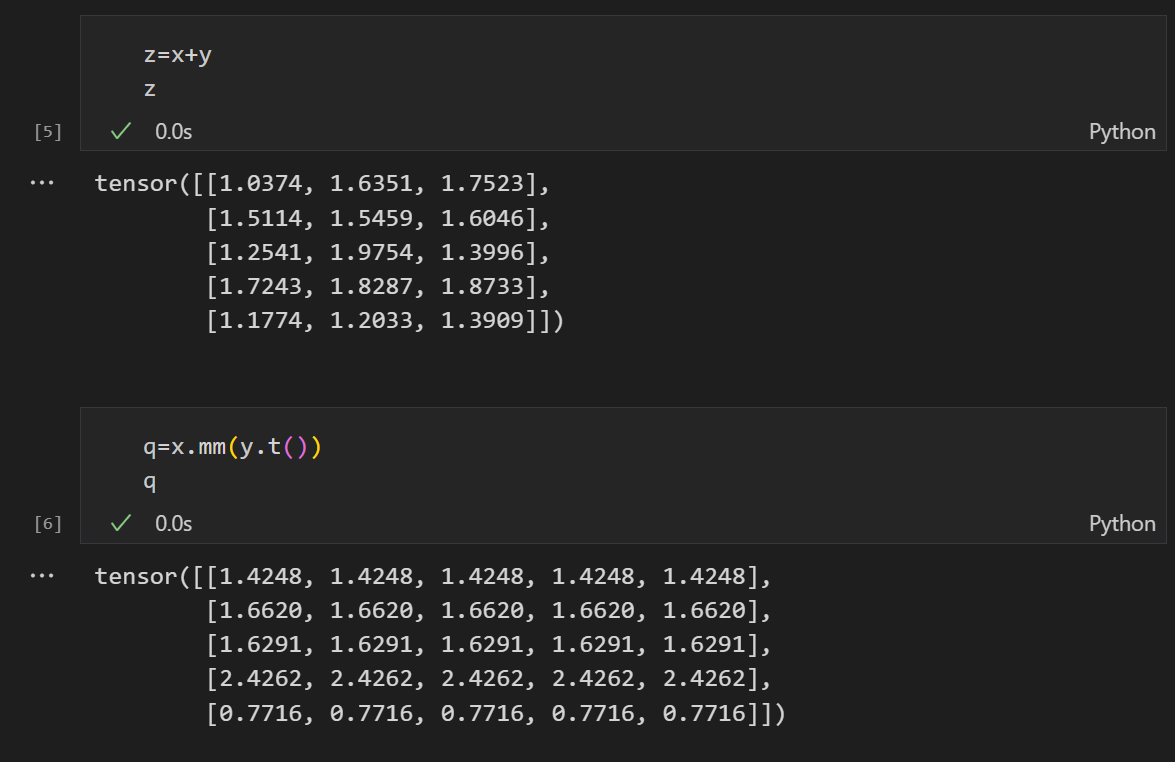
1. 实验过程和结果
2. 张量的运算练习

1.1使用Tensor



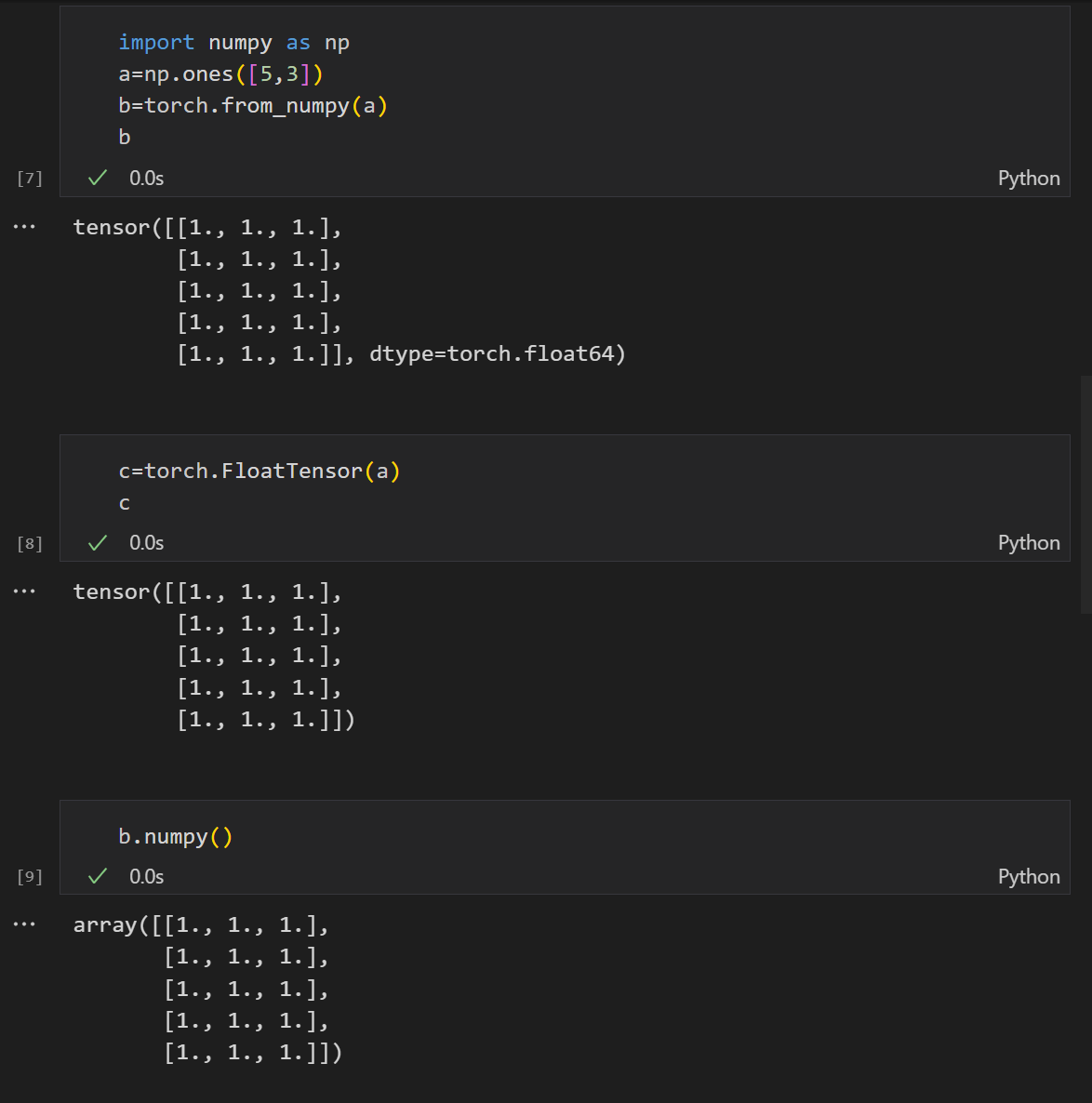
这段代码首先引入了torch库输出torch的版本，并使用torch.random()和.ones()生成了5×3大小的二维张量，生成随机数矩阵，和全1矩阵。

1.2基本Tensor运算

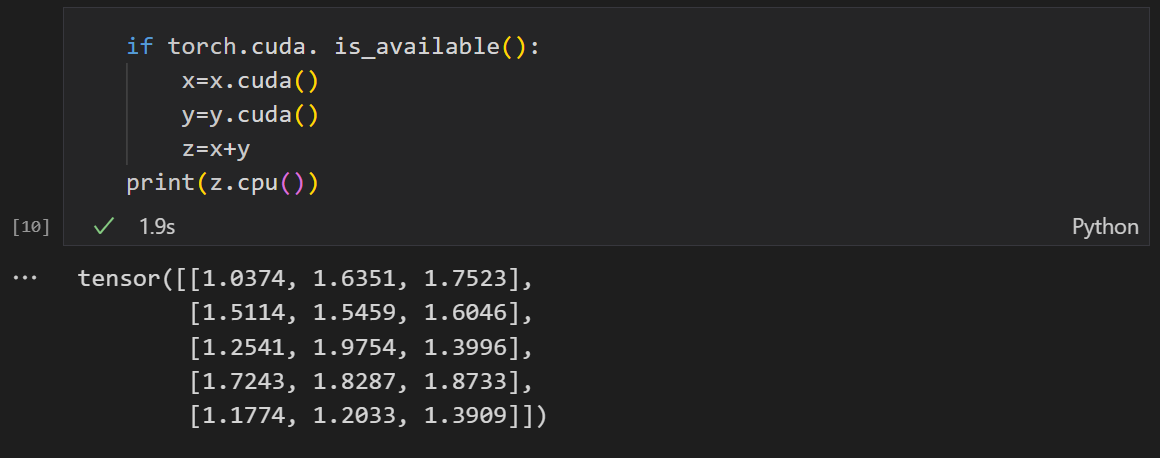


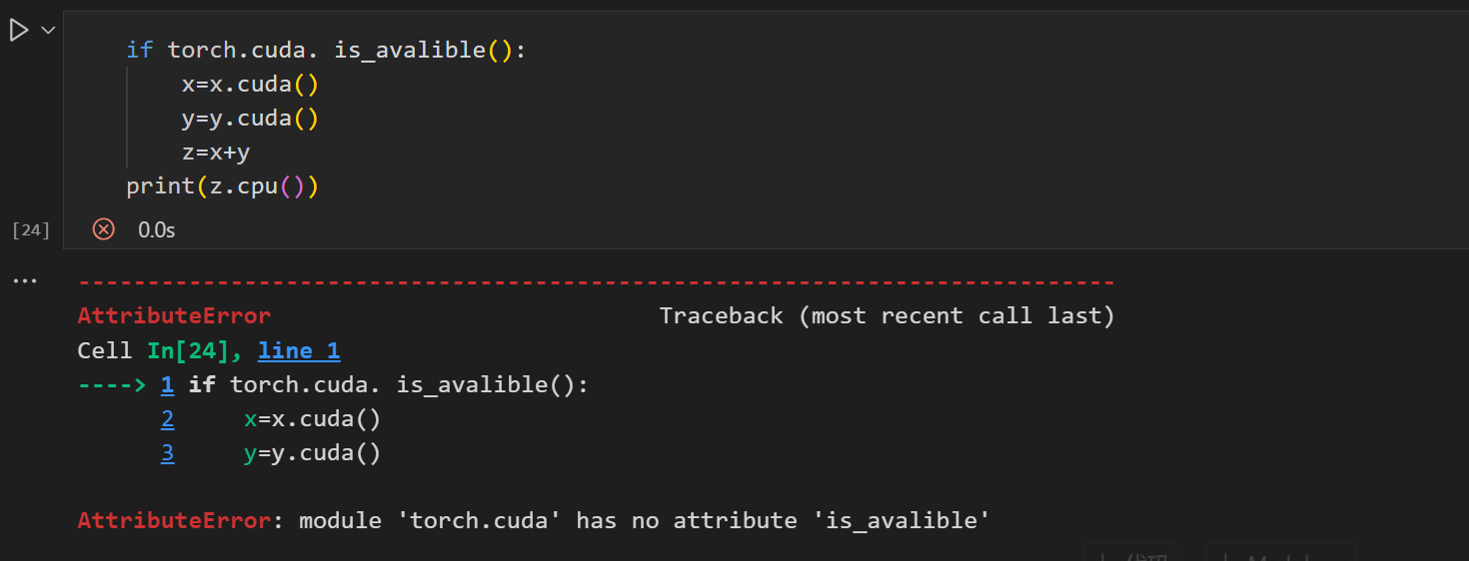
这段代码运用了张量的基本运算，同维度张量矩阵的加法，和张量矩阵的乘法。.mm()和.t()方法分别代表了矩阵乘法和矩阵的转置。

1.3Tensor与numpy.ndarry之间的转换



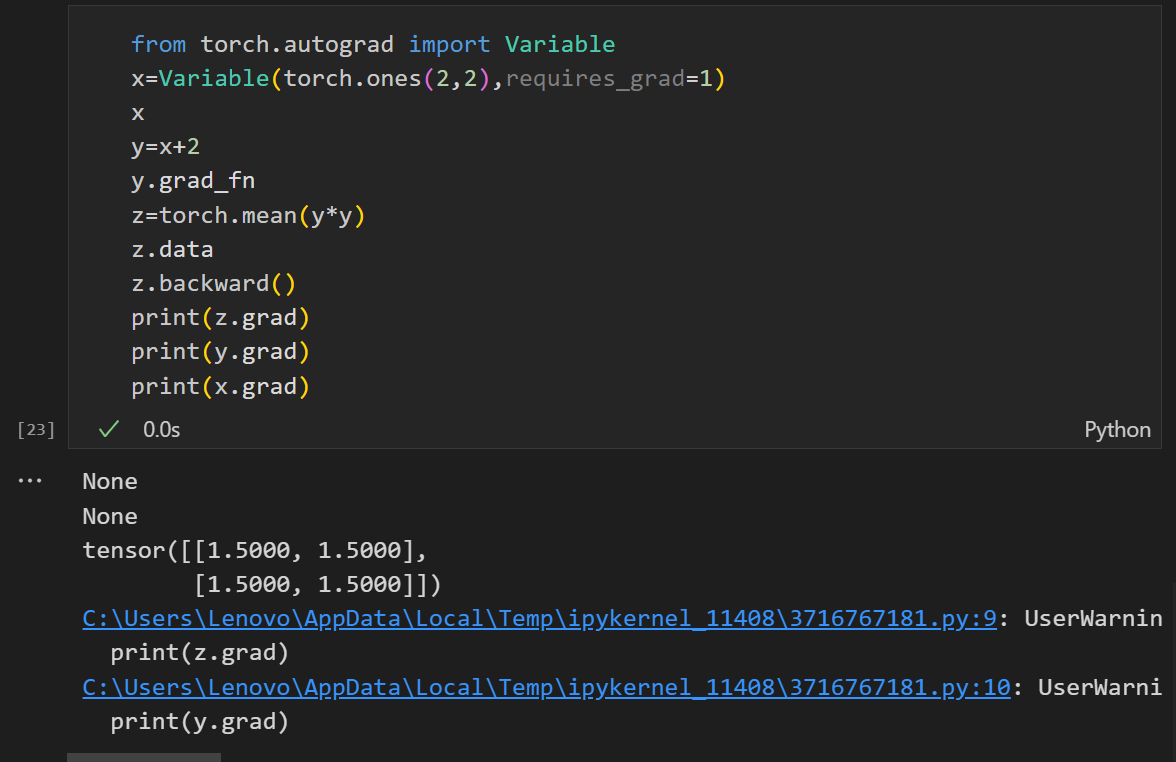
该段代码引入了numpy库，a为使用np库生成的ndarry矩阵，b、c为由a转换的torch张量，采用了不同的转换方法，用numpy()方法将张量转换为numpy数组。



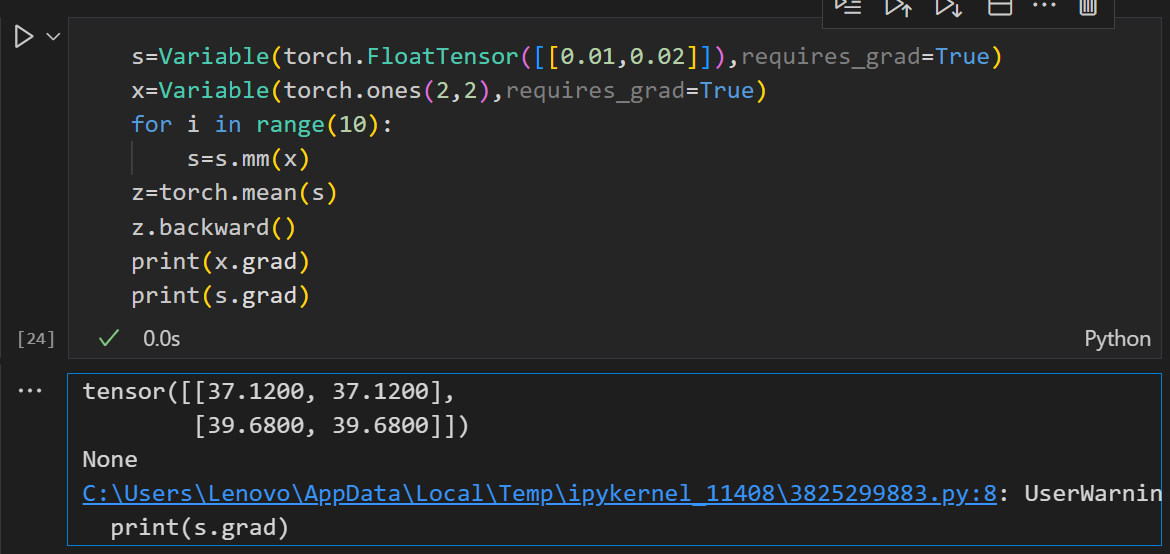


该段代码，首先判断了torch版本是否可用CUDA，在判断语句中将x、y张量转换到GPU中，并做加法计算，再将结果z转换到CPU内存中。上图两个结果是在两个不同的Python内核环境中的运行结果，分别使用了GPU/CUDA版本的torch和CPU版本的torch。

2.自动微分变量的练习



该段代码从torch.autograd中引入Variable函数，用于生成自动微分变量，生成一个变量x设置为需要求梯度，z由x和y计算得到，对z进行反向传播算法，得到各节点的梯度信息，z和y不是子节点没有梯度信息。



该段代码展示了更复杂的一组张量运算并进行反向传播算法，并求梯度。将张量s与x进行十次矩阵乘法，再求s的均值z，对z进行反向传播。对x和s求梯度，s不是叶节点没有梯度信息。

三、心得体会

本次实验主要练习了PyTorch张量的基本用法，对张量的计算，numpy数组的相互转化，反向传播、梯度计算进行了练习。对PyTorch的使用方法有了初步的了解，了解了动态计算图，动态计算图是pytorch最主要的特性，可以清楚记录算法的每一步流程，自动微分变量，可以通过反向传播算法得到其梯度信息。动态计算图使自动微分变量在计算梯度和进行反向传播更加方便简单。

实验过程中初次尝试没有选择合适的环境，导致安装的CUDA不可用。原因是Anaconda安装时没有添加到环境变量，VS code没有识别出该环境，采用了默认的python环境没有安装正确版本的torch。通过重新安装anaconda解决问题。