本实验推荐在Linux系统下完成，Windows系统也能支持，但`5.pytorch.ipynb`中的数据下载需手动操作。

实验室Linux环境

实验室电脑建议使用Virtual Box虚拟机中的Ubuntu22或者SEED Ubuntu镜像，Ubuntu22的密码放在桌面上，SEED Ubuntu默认的账号是SEED，密码是dees。默认的python命令是`python3`，如果没有`pip`或`pip3`，请执行以下命令安装

```
sudo apt install python3-pip -y
```

conda

在深度学习开发领域，可能会用到不同的深度学习框架、同一个框架不同版本、不同版本的python依赖包，推荐使用Conda进行开发环境管理。

Conda 是一个开源跨平台的包管理、环境管理工具，支持 Windows、MacOS 和 Linux 系统，适用于 Python、Java、C/C++ 等多种开发语言，方便创建、保存、加载和切换开发环境。Conda最初专为 Python 开发设计的，现在可以打包和分发任何语言的软件。默认配置中，Conda 可以安装和管理 repo.anaconda.com 仓库中的 7,500 多个软件包，该仓库由 AnaConda 生成，审查和维护。

Conda发行版即安装方式有miniconda和anaconda两种，两者都是Python发行版，需要先安装Python，现有的Linux发行版都默认带有Python。miniConda是最小化的发行版，安装好之后会有Python，Conda，以及一些基础包，占用的空间较小。anaconda相比miniconda安装的常用包多很多，占的空间也很大。无论是miniconda安装还是anaconda安装，安装好之后，都是用conda命令进行操作，默认进入的是base环境。

实验建议安装miniconda，具体步骤请参考[miniconda-install]，根据所用的操作系统、python版本选择对应的安装文件。

conda初始化：安装之后按照输出的提示信息，执行以下命令初始化。初始化之后需关闭终端新开一个即重新进入shell环境，否则仍然找不到conda命令。如果用的是fish或者zsh，请查看教程进行对应的初始化。

```
~/miniconda3/bin/conda init
```

创建一个新的环境：以下命令创建一个名ai_lab的环境，指定python版本是3.9。过程中会提示安装哪些包，并通过键盘输入y表示安装。

```
conda create --name ai_lab python=3.9 -y
```

激活指定的环境：创建环境成功后，会在最后的输出提示怎么激活环境，执行以下命令激活。激活之后，终端提示符最开头会看到ai_lab的字样，环境中的python也是指定的版本，后续实验都在ai_lab这个环境下操作。

```
conda activate ai_lab
```

安装依赖包：在conda环境中如果安装缺失的包，优先使用conda安装，如果conda没有对应的包，再尝试pip安装。

```
conda install package_name
```

本次实验用到的基础包通过从文件读入的方式安装，mindspore、pytorch需要自行按照官方教程安装。

```
conda install --file requirements.txt
```

另需要通过pip安装download包：

```
pip install download
```

以上操作主要参考自Getting started with conda

Jupyter Notebook

notebook是基于网页的用于交互计算的应用程序，在一个文件里面，集成了文本、代码、数据展示等功能。notebook的具体实现有很多种，现在广泛使用的是jupyter notebook。jupyter是一个开源项目，该项目下jupyter notebook、jupyterLab、jupyterHub等应用，这3个应用的核心功能都是notebook。Jupyter Notebook实际也是轻量级的IDE，支持浏览器访问。

运行conda activate ai_lab 进入ai_lab环境，使用conda 安装jupyter，过程中会下载很多依赖包进行安装。

```
conda install -c anaconda jupyter
```

运行jupyter，如果能输出以下内容则安装成功。

```
(ai_lab) $ ~/hitsz-cs-ai ‹master*› »jupyter                                    1 ↲
usage: jupyter [-h] [--version] [--config-dir] [--data-dir] [--runtime-dir] [--paths]
               [--json] [--debug]
               [subcommand]

Jupyter: Interactive Computing

positional arguments:
  subcommand     the subcommand to launch

optional arguments:
  -h, --help     show this help message and exit
  --version      show the versions of core jupyter packages and exit
  --config-dir   show Jupyter config dir
  --data-dir     show Jupyter data dir
  --runtime-dir  show Jupyter runtime dir
  --paths        show all Jupyter paths. Add --json for machine-readable format.
  --json         output paths as machine-readable json
  --debug        output debug information about paths

Available subcommands: bundlerextension console dejavu events execute kernel kernelspec lab
labextension labhub migrate nbclassic nbconvert nbextension notebook qtconsole run server
serverextension troubleshoot trust

Please specify a subcommand or one of the optional arguments.
(ai_lab) $ ~/hitsz-cs-ai ‹master*› »                                           1 ↲
```

在jupyter notebook中使用conda环境，需要创建kernel

```
conda install ipykernel
python -m ipykernel install --user --name=ai_lab    # 指定kernel名字
```

切换到实验所在的路径，执行以下命令启动notebook，退出jupyter notebook的方式是ctrl+c终止任务。

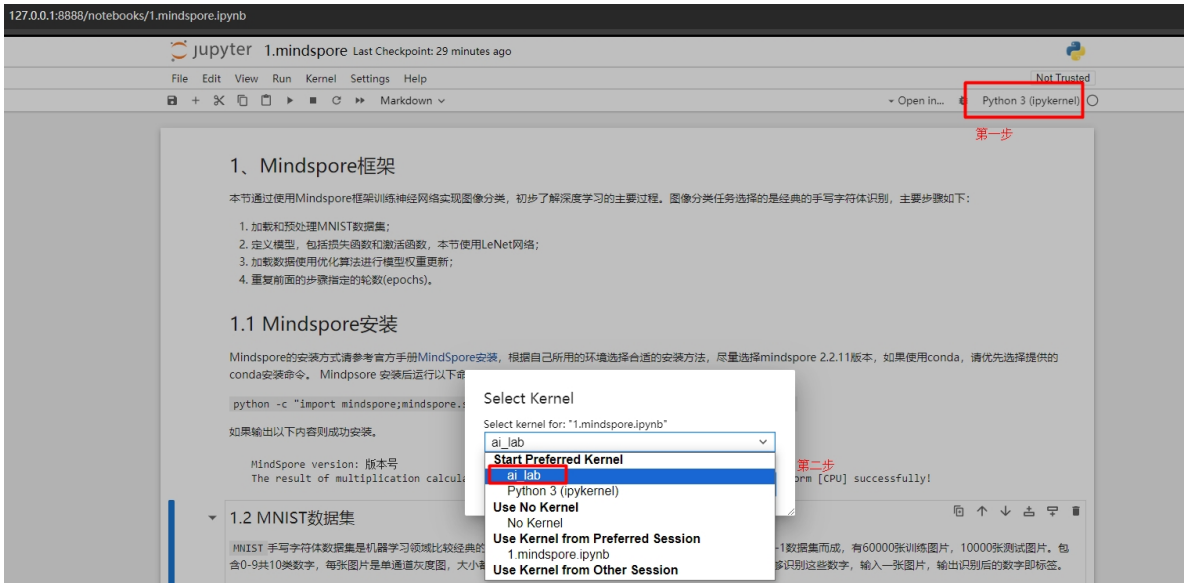
```
jupyter notebook
```

从输出信息的最后复制以下访问链接到浏览器。为了安全，默认配置下jupyter notebook每次启动生成的token都不同。

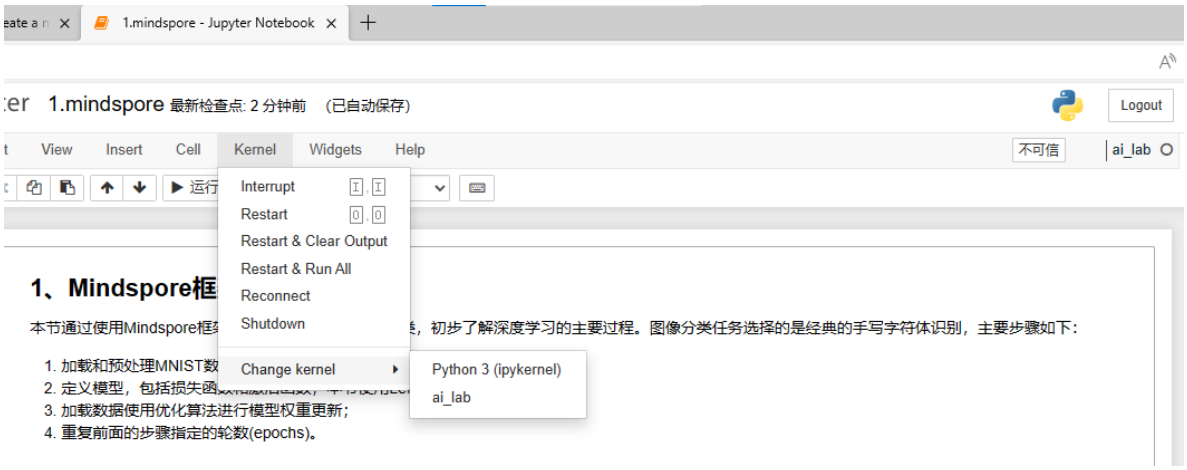
```
[I 2024-03-07 10:34:53.619 ServerApp] Use Control-C to stop this server and shut down all ker
[C 2024-03-07 10:34:53.622 ServerApp]

    To access the server, open this file in a browser:
        file:///home/zhg/.local/share/jupyter/runtime/jpserver-67137-open.html
    Or copy and paste one of these URLs:
        http://ubuntu-big:8888/tree?token=ab475b951eb6394a3f8ae2c53c6a91523312234411012ab3
        http://127.0.0.1:8888/tree?token=ab475b951eb6394a3f8ae2c53c6a91523312234411012ab3
[I 2024-03-07 10:34:54.373 ServerApp] Skipped non-installed server(s): bash-language-server,
```

在浏览器中打开链接，可以看到启动jupyter notebook所在目录的所有文件。双击`1.mindspore.ipynb`文件打开，按下图的两种方式之一选择前面创建的ai_lab kernel。新打开一个文件要注意切换kernel，否则容易出现明明已经安装了对应的Python包仍然会提示找不到。

kernel选择方式一：



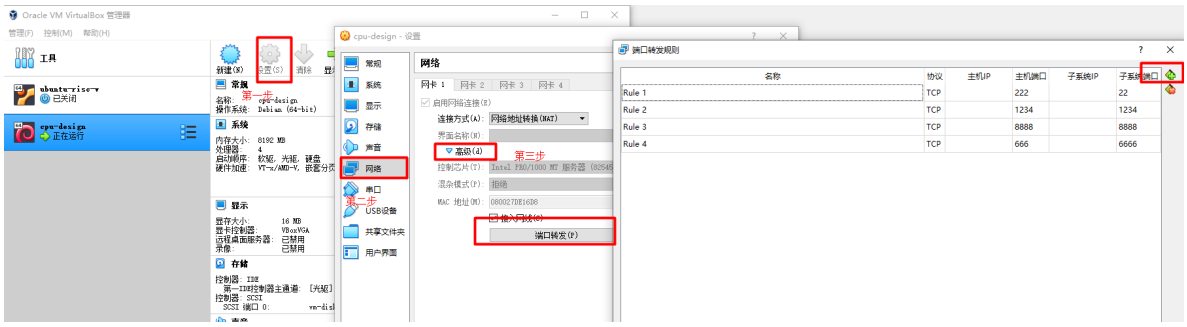
kernel选择方式二：



notebook的基本单元是cell，每个cell可以配置为code或markdown，点击顶部的运行按钮可执行选中的cell，如果是代码，执行结果会输出在cell下方，如果是markdown则会显示渲染后的文本。双击某个cell，进入编辑模式，可以进行修改。更多的操作请查看官方教程。

如果是虚拟机中启动notebook，想在宿主机Windows中的浏览器访问，启动时需要指定ip以允许远端ip访问，然后参考下图配置一条8888端口的转发规则。

```
jupyter notebook --ip 0.0.0.0
```



numpy

NumPy是Python中科学计算的基础包。它是一个Python库，提供多维数组对象以及用于数组快速操作的各种API。原生Python Array（数组）即list类型也可以提供多维数组，为什么还需要numpy？numpy性能更优，接口更丰富。相对于原生数组，具体有以下区别：

1. NumPy 数组在创建时具有固定的大小，与Python的原生数组对象（可以动态增长）不同。更改ndarray的大小将创建一个新数组并删除原来的数组。
2. NumPy 数组中的元素都需要具有相同的数据类型，因此在内存中的大小相同。 例外情况：Python的原生数组里包含了NumPy的对象的时候，这种情况下就允许不同大小元素的数组。
3. NumPy 数组有助于对大量数据进行高级数学和其他类型的操作。通常，这些操作的执行效率更高，比使用Python原生数组的代码更少。

NumPy包的核心是 ndarray 对象。它封装了python原生的同数据类型的 n 维数组，为了保证其性能优良，其中有许多操作都是代码在本地进行编译后执行的。

深度学习中另一个经常碰到的概念是张量（Tensor），即多维数组。PyTorch和TensorFlow有张量类Tensor，都与Numpy的ndarray类似。但深度学习框架又比Numpy的ndarray多一些重要功能，比如张量类支持自动微分。python的原生数组是list，numpy是ndarry，各框架的张量类，类型之间都可以互相转换。

实验中用到框架代码是基于numpy实现的，mindspore、pytorch等框架用到的数据可能也涉及到与numpy互相转换。numpy的详细学习可以参考以下资料：

1. https://numpy.org/devdocs/user/quickstart.html
2. https://cs231n.github.io/python-numpy-tutorial/