1 Configuring the Windows Experiment Environment

1.1 Installing Anaconda

1.1.1 Introduction to Anaconda

Anaconda is a distribution of Python for scientific computing. It supports Linux, macOS, and Windows. It provides simplified package management and environment management, and easily deals with the installation issues when the system has multiple Python versions and third-party packages. Anaconda uses the conda tool/command or pip to implement package and environment management. It also comes with Python and related tools. Anaconda is a Python tool for enterprise-level big data analytics. It contains more than 720 open-source data-science packages, including data visualization, machine learning, and deep learning. It can be used for data analysis, big data and AI fields.

Anaconda provides the following features for developers:

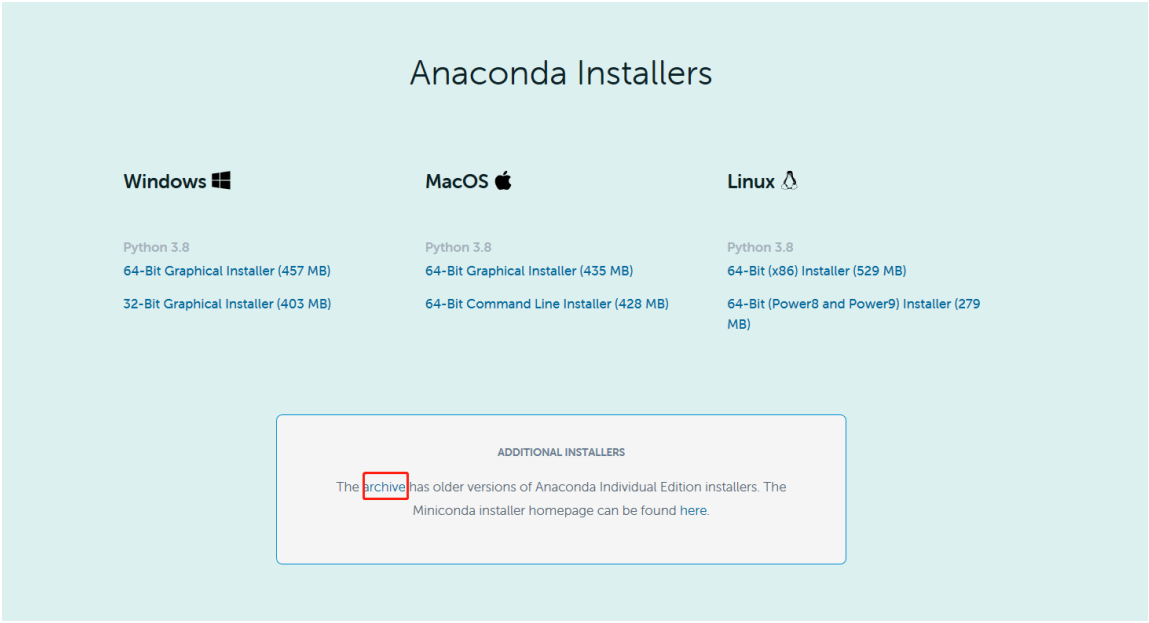
- With Anaconda, you do not need to install Python. You only need to select the Python version when downloading Anaconda.
- With Anaconda, you only need to add a virtual environment to Anaconda when different frameworks are required to support the development. You can conduct development in different environments without worrying about compatibility issues. You can also configure environments for special projects to facilitate management.

1.1.2 Installation Procedure

Step 1 Download Anaconda.

Log in to the official Anaconda website and download the installation package **https://www.anaconda.com/distribution/#download-section**.

Select the version for Windows, macOS, or Linux. In this example, select **Windows**. But because of the current anaconda update, the current version has been used python3.8, so we choose to enter the historical version here ,(32-bit computer can not match the course).



Then the pop-up page finds the version we need

We choose the 2020.02 windows64 bit version, under which python3.7 is installed by default

Anaconda3-2020.02-Linux-x86_64.sh	521.6M	2020-03-11 10:32:37	17600d1f12b2b047b62763221f29f2bc
Anaconda3-2020.02-MacOSX-x86_64.pkg	442.2M	2020-03-11 10:32:57	d1e7fe5d52e5b3ccb38d9af262688e89
Anaconda3-2020.02-MacOSX-x86_64.sh	430.1M	2020-03-11 10:32:34	f0229959e0bd45dee0c14b20e58ad916
Anaconda3-2020.02-Windows-x86.exe	423.2M	2020-03-11 10:32:58	64ae8d0e5095b9a878d4522db4ce751e
Anaconda3-2020.02-Windows-x86_64.exe	466.3M	2020-03-11 10:32:35	6b02c1c91049d29fc65be68f2443079a
Anaconda3-2020.07-Linux-ppc64le.sh	290.4M	2020-07-23 12:16:47	daf3de1185a390f435ab80b3c2212205
Anaconda3-2020.07-Linux-x86_64.sh	550.1M	2020-07-23 12:16:50	1046c40a314ab2531e4c099741530ada
Anaconda3-2020.07-MacOSX-x86_64.pkg	462.3M	2020-07-23 12:16:42	2941ddbaf0cdb49b342c18cde51fee43
Anaconda3-2020.07-MacOSX-x86_64.sh	454.1M	2020-07-23 12:16:44	50f20c90b8b5bfdc09759c09e32dce68
Anaconda3-2020.07-Windows-x86.exe	397.3M	2020-07-23 12:16:51	aa7dcf4d02baa25d14baf5728e29d067
Anaconda3-2020.07-Windows-x86_64.exe	467.5M	2020-07-23 12:16:46	7c718535a7dd89fa46b147626ada9e46
Anaconda3-2020.11-Linux-ppc64le.sh	278.9M	2020-11-18 16:45:36	bc09710e65cdbba68688061b149281dc
Anaconda3-2020.11-Linux-x86_64.sh	528.8M	2020-11-18 16:45:36	4cd48ef23a075e8555a8b6d0a8c4bae2
Anaconda3-2020.11-MacOSX-x86_64.pkg	435.3M	2020-11-18 16:45:35	2f96bb47eb5a949da6f99a71d7d66420
Anaconda3-2020.11-MacOSX-x86_64.sh	427.8M	2020-11-18 16:45:35	918de9a9936908fe62514e0ca6873a21
Anaconda3-2020.11-Windows-x86.exe	403.0M	2020-11-18 16:45:34	ca1f6f3e75eb346f5ab2d87bab005878
Anaconda3-2020.11-Windows-x86_64.exe	457.2M	2020-11-18 16:45:34	0841ffcb927a3c2edbd682520f52e546
Anaconda3-4.0.0-Linux-x86.sh	336.9M	2016-03-29 11:15:03	c88cbe27cc8fb4976e6bd38068cc57d6
Anaconda3-4.0.0-Linux-x86_64.sh	398.4M	2016-03-29 11:15:02	546d1f02597587c685fa890c1d713b51
Anaconda3-4.0.0-MacOSX-x86_64.pkg	341.5M	2016-03-29 11:16:08	b25796c49f9d3b47561c6eac9bbc77f0
Anaconda3-4.0.0-MacOSX-x86_64.sh	292.7M	2016-03-29 11:16:21	efd870aa3fabc8f4865a1b9567e69b69
Anaconda3-4.0.0-Windows-x86.exe	283.1M	2016-03-29 11:16:22	ae5c9ba0c6f4639fbf94848f81c3d4b4
Anaconda3-4.0.0-Windows-x86_64.exe	345.4M	2016-03-29 11:16:22	a6b7a787c6c574867cee3f2d12ecfc50
Anaconda3-4.1.0-Linux-x86.sh	328.4M	2016-06-28 11:28:32	302fddc310233f5e6f120753ec3e392d

Step 2 Install Anaconda.

Double-click the downloaded **Anaconda3-x.x.x-Windows-x86_64.exe** file. In the dialog box that is displayed as shown in Figure 1-1, click **Next**.

Figure 1-1

Click **I Agree**.

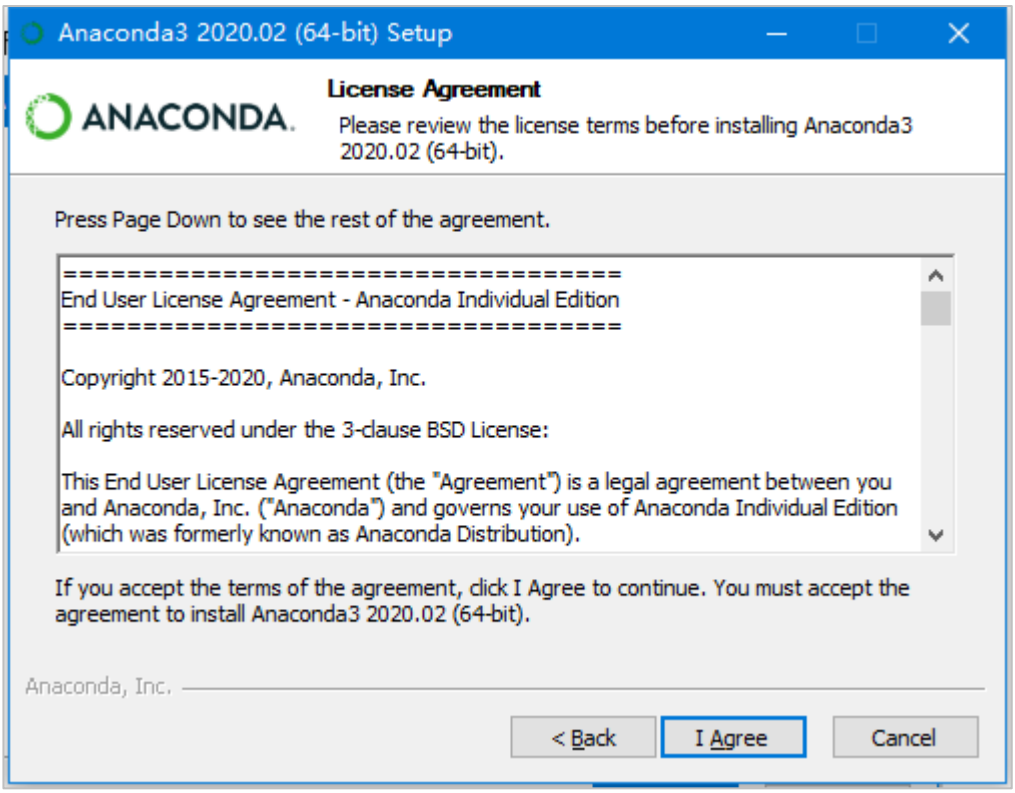


Figure 1-2

Select **Just Me** and click **Next**.

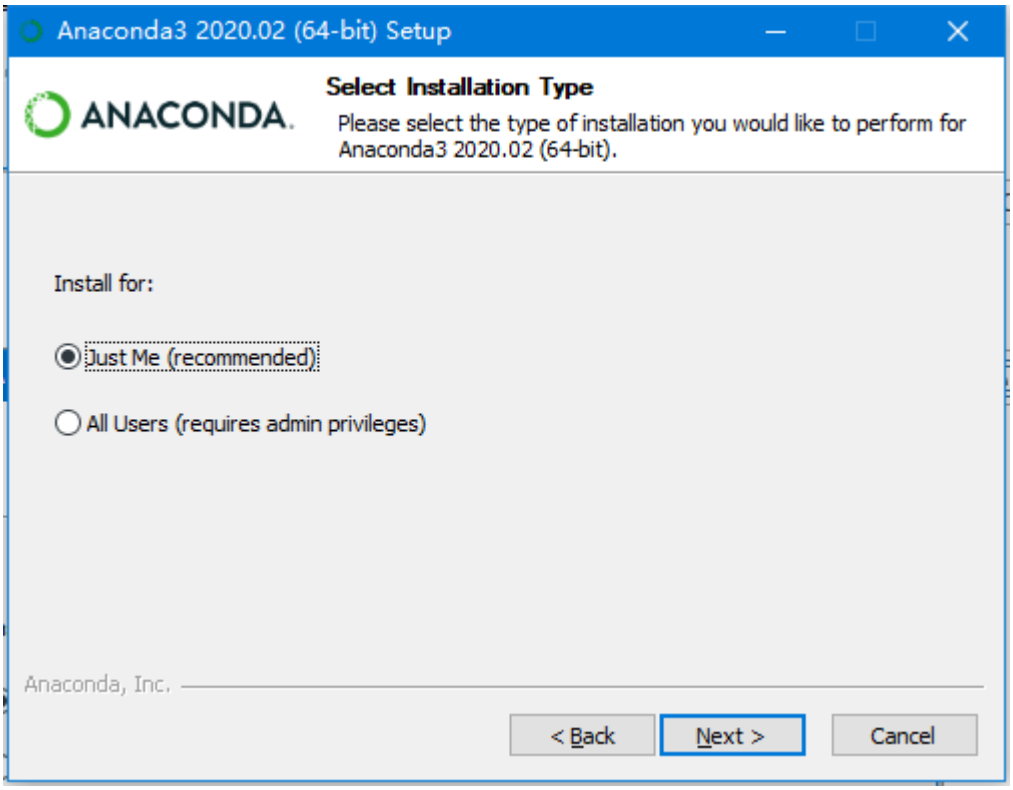


Figure 1-3

Step 3 Set the installation path.

Specify the software installation path and click **Next**.

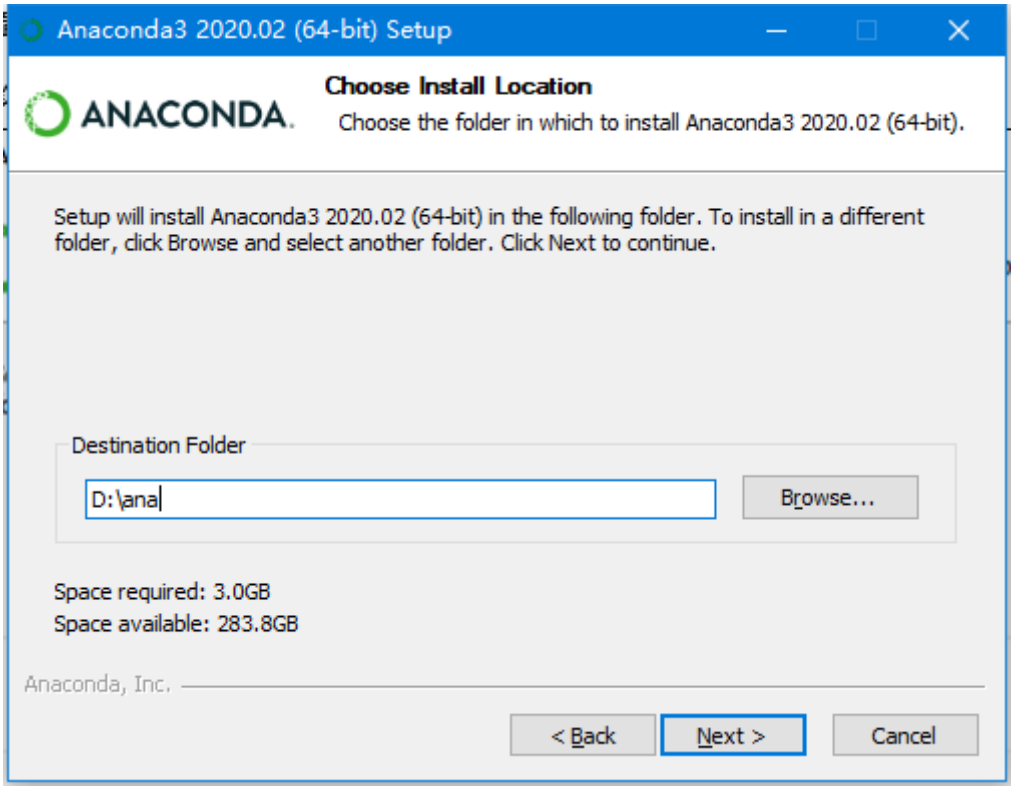


Figure 1-4

Step 4 Configure environment variables.

Select **Add Anaconda to my PATH environment variable** and **Register Anaconda as my default Python 3.7** to reduce subsequent configurations, and click **Install**. (Note: If Python of another version has been installed on the computer, you are advised to remove it and install Anaconda. If it is not removed, do not select the two options. Otherwise, a path error may occur.)

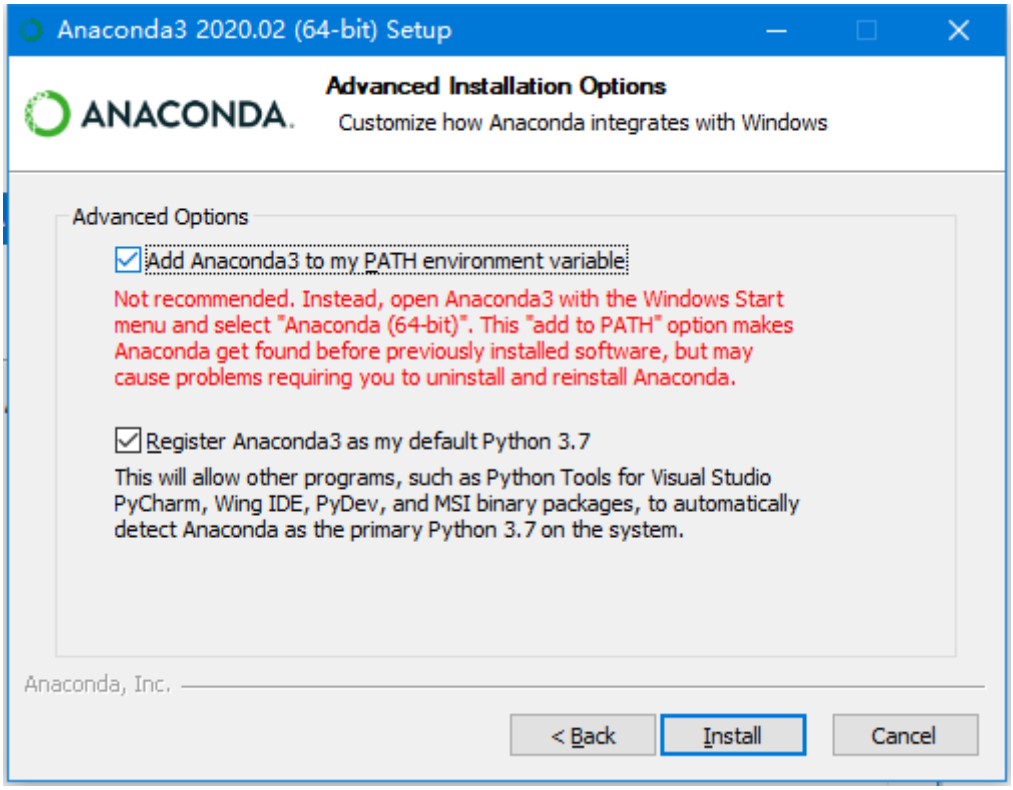


Figure 1-5

After the installation is complete, click **Finish**.

1.2 Third-party library installation

pip install missingno

pip install scikit-learn

pip install imblearn

pip install seaborn

pip install nltk

pip install wordcloud

pip install xgboost

1.3 Installing TensorFlow

1.3.1 Introduction to TensorFlow

TensorFlow is Google's second-generation artificial intelligence learning system based on DistBelief. Its name derives from its operating principle. A tensor indicates an N-dimensional array, and a flow indicates calculation based on the dataflow graph. A tensor flow is a calculation process of a tensor from one end of the graph to the other

end. TensorFlow is a system that transmits complex data structures to the AI neural network for analysis and processing.

The latest version TensorFlow 2.1.0 focuses on simplicity and ease of use and provides the following new features:

- Keras module added
- Eager execution of the dynamic graph mechanism
- Support for more platforms and languages
- Removal of discarded APIs and reduction of duplicate APIs
- Compatibility and continuity

TensorFlow 2.0 and 2.1.0 are dominant versions used currently. TensorFlow 2.0 is stable, and TensorFlow 2.1.0 is an optimized version released in November 2019. There are minor differences between the two versions. You can select a version as required. The following describes how to install the 2.0 and 2.1.0 releases for CPU and GPU.

1.3.2 Installing TensorFlow 2.1.0 for CPU

Step 1 Install TensorFlow 2.1.0 for CPU.

Open the Anaconda Terminal window from the start menu.

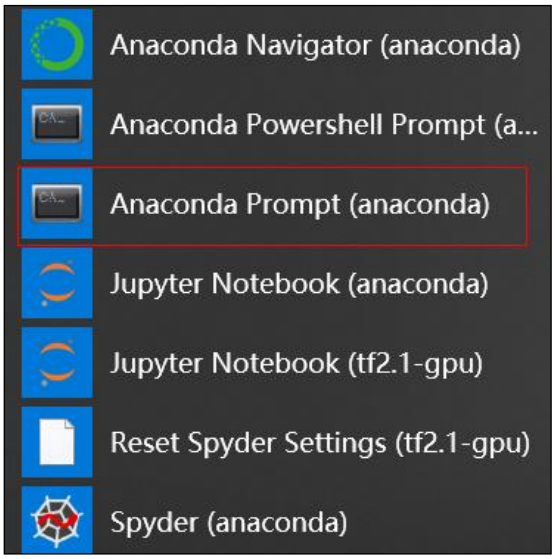


Figure 1-6

Before the installation, ensure that the computer is properly connected to the Internet.

Step 2 Run the pip install **tensorflow-cpu** command.

```
(base) PS C:\Users\11611> pip install tensorflow==2.1.0
Requirement already satisfied: tensorflow==2.1.0 in e:\software\anaconda\lib\site-packages (2.1.0)
Requirement already satisfied: scipy==1.4.1; python_version >= "3" in c:\users\11611\appdata\roaming\python\python37\s
e-packages (from tensorflow==2.1.0) (1.4.1)
Requirement already satisfied: six>=1.12.0 in e:\software\anaconda\lib\site-packages (from tensorflow==2.1.0) (1.12.0)
Requirement already satisfied: keras-applications>=1.0.8 in e:\software\anaconda\lib\site-packages (from tensorflow==2
.0) (1.0.8)
```

Figure 1-7

The installation is successful if the following information is displayed:

```
Successfully installed absl-py-0.9.0 astor-0.8.1 cachetools-4.0.0 chardet-3.0.4 gast-0.2.2 google-auth-1.11.2 google-aut
h-oauthlib-0.4.1 google-pasta-0.1.8 grpcio-1.27.2 h5py-2.10.0 idna-2.9 keras-applications-1.0.8 keras-preprocessing-1.1.
0 markdown-3.2.1 numpy-1.18.1 oauthlib-3.1.0 opt-einsum-3.2.0 protobuf-3.11.3 pyasn1-0.4.8 pyasn1-modules-0.2.8 requests
-2.23.0 requests-oauthlib-1.3.0 rsa-4.0 scipy-1.4.1 six-1.14.0 tensorboard-2.1.1 tensorflow-2.1.0 tensorflow-estimator-2
.1.0 termcolor-1.1.0 urllib3-1.25.8 werkzeug-1.0.0 wrapt-1.12.0
```

Figure 1-8

Anaconda installed comes with a Python module required in this experiment. You do not need to install pandas, numpy, or scikit-image.

Step 3 If you need to install a module, run the **pip install** *software package* command. For example, run **pip install matplotlib**.

```
(tf2) C:\Users\WWX697589>pip install matplotlib
Looking in indexes: https://pypi.tuna.tsinghua.edu.cn/simple/
Collecting matplotlib
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/93/22/a3cef48d25ad94f540bb3dd91490fb22afe0911acc3390b1929527a
e4f/matplotlib-3.1.3-cp37-cp37m-win_amd64.whl (9.1 MB)
     |████████████████████████████████████████| 9.1 MB 3.3 MB/s
Collecting kiwisolver>=1.0.1
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/c6/ea/e5474014a13ab2dcb5056608e0716c600c3d8a8bcffb10ed55ccd6a
eb0/kiwisolver-1.1.0-cp37-none-win_amd64.whl (57 kB)
     |████████████████████████████████████████| 57 kB 4.1 MB/s
Collecting python-dateutil>=2.1
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/d4/70/d60450c3dd48ef87586924207ae8907090de0b306af2bce5d134d78
5cb/python_dateutil-2.8.1-py2.py3-none-any.whl (227 kB)
     |████████████████████████████████████████| 227 kB 6.8 MB/s
Requirement already satisfied: numpy>=1.11 in d:\anaconda\envs\tf2\lib\site-packages (from matplotlib) (1.18.1)
Collecting cycler>=0.10
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/f7/d2/e07d3ebb2bd7af696440ce7e754c59dd546ffe1bbe732c8ab68b9c8
e61/cycler-0.10.0-py2.py3-none-any.whl (6.5 kB)
```

Figure 1-9

Step 4 Check the installation.

Run the **pip list** command. All Python modules installed are displayed, as shown in the following figure.

```
Package            Version
------------------ ---------------
matplotlib         3.1.3
numpy              1.18.1
oauthlib           3.1.0
opt-einsum         3.2.0
pip                20.0.2
protobuf           3.11.3
pyasn1             0.4.8
pyasn1-modules     0.2.8
pyparsing          2.4.6
python-dateutil    2.8.1
requests           2.23.0
requests-oauthlib  1.3.0
rsa                4.0
scipy              1.4.1
setuptools         45.2.0.post20200210
six                1.14.0
tensorboard        2.1.1
tensorflow         2.1.0
tensorflow-estimator 2.1.0
termcolor          1.1.0
urllib3            1.25.8
Werkzeug           1.0.0
wheel              0.34.2
wincertstore       0.2
wrapt              1.12.0
```

Figure 1-10

1.3.3 Installing TensorFlow 2.0.0 for CPU

Step 1 Install TensorFlow 2.0.0 for CPU.

Open the Anaconda Terminal window from the start menu.

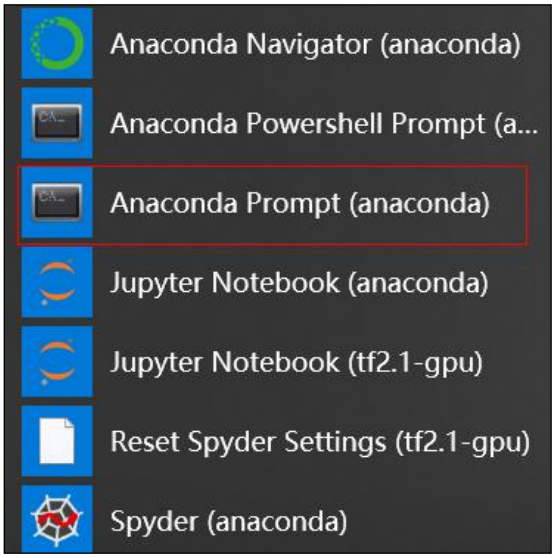


Figure 1-11

Before the installation, ensure that the computer is properly connected to the Internet.

Step 2 Run the **pip install tensorflow==2.0.0 command**.

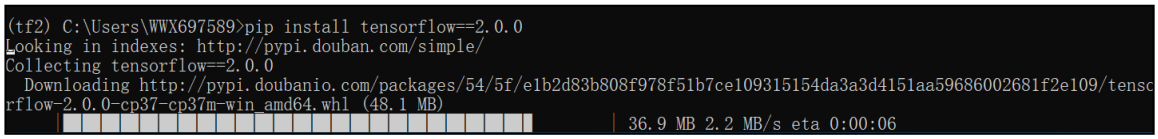


Figure 1-12

The installation is successful if the following information is displayed:

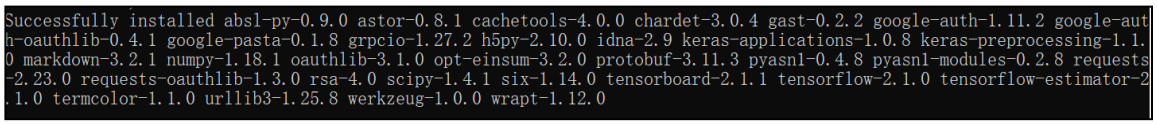


Figure 1-13

1.3.4 Installing TensorFlow 2.1.0 for GPU

Step 1 Install TensorFlow 2.1.0 for GPU.

Open the Anaconda Terminal window from the start menu.

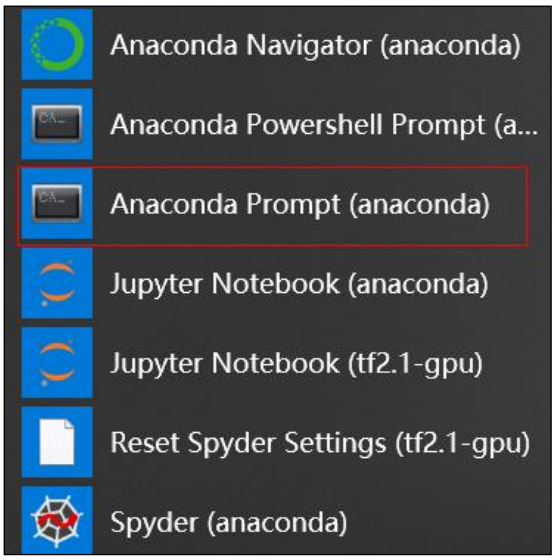


Figure 1-14

Before the installation, ensure that the computer is properly connected to the Internet.

Step 2 Run the **pip install tensorflow** command.

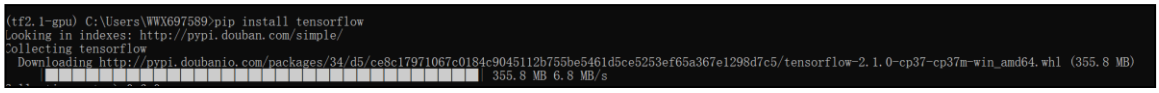


Figure 1-15

The installation is successful if the following information is displayed:

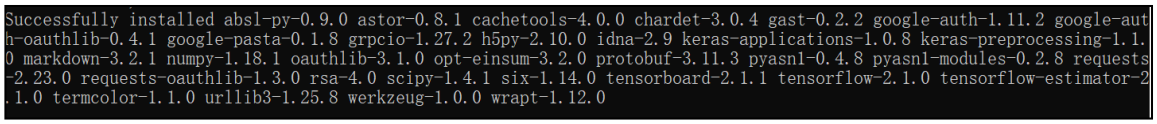


Figure 1-16

Step 3 GPU acceleration can be implemented only when the local host has an independent graphics card and CUDA and cuDNN are installed. Install CUDA first. Download the required CUDA version from https://developer.nvidia.com/cuda-10.1-download-archive-update2?target_os=Windows&target_arch=x86_64&target_version=10&target_type=exenetwork.

Version	Python version	Compiler	Build tools	cuDNN	CUDA
tensorflow_gpu-2.4.0	3.6-3.8	MSVC 2019	Bazel 3.1.0	8.0	11.0
tensorflow_gpu-2.3.0	3.5-3.8	MSVC 2019	Bazel 3.1.0	7.6	10.1
tensorflow_gpu-2.2.0	3.5-3.8	MSVC 2019	Bazel 2.0.0	7.6	10.1
tensorflow_gpu-2.1.0	3.5-3.7	MSVC 2019	Bazel 0.27.1-0.29.1	7.6	10.1
tensorflow_gpu-2.0.0	3.5-3.7	MSVC 2017	Bazel 0.26.1	7.4	10

Install install according to your tensorflow version and CUDA

Of course see your graphics branch doesn't support CUDA.Here the graphics card we use is

Nvidia RTX2060 as Display

This URL is directed to CUDA 10.1. To download other versions, click **Legacy Releases**.

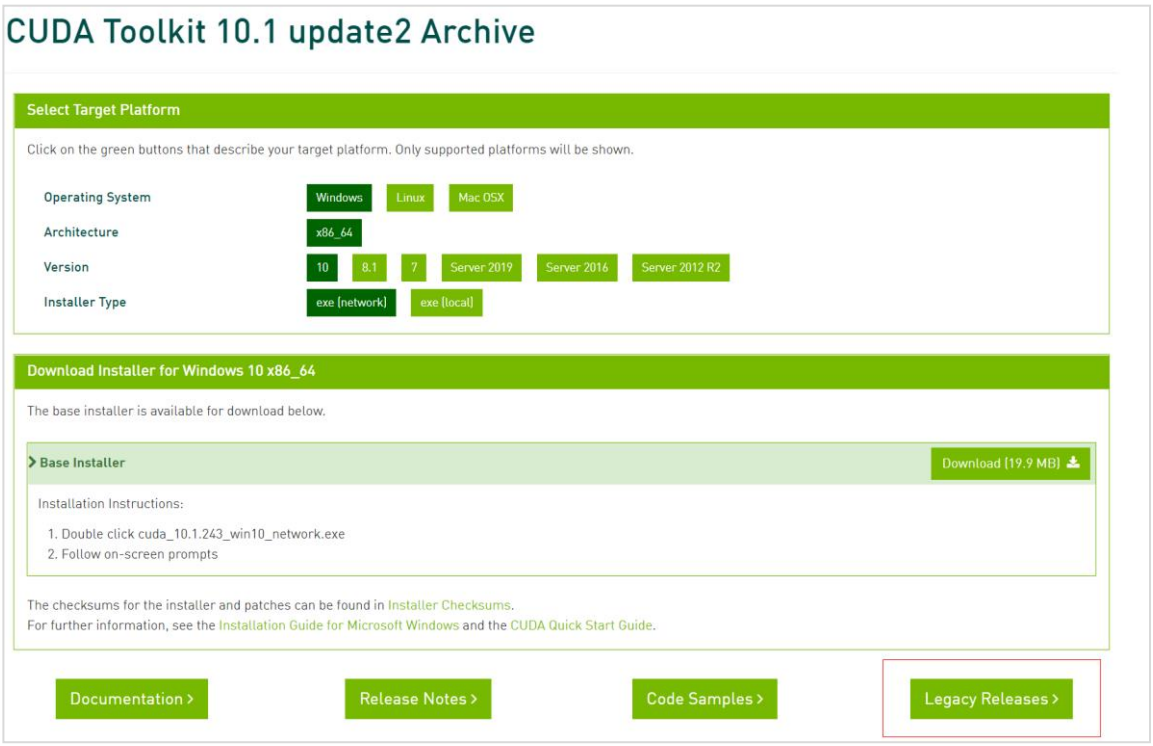


Figure 1-17

Step 4 Double-click the downloaded program to install CUDA. At the beginning, the program automatically decompresses the package to the default directory. Click **OK** to go to the next step.

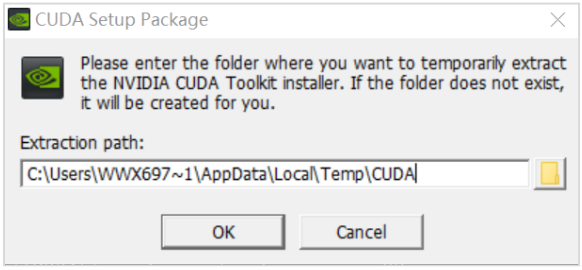


Figure 1-18

Step 5 After the decompression is complete, the installation starts. Click **AGREE AND CONTINUE**.

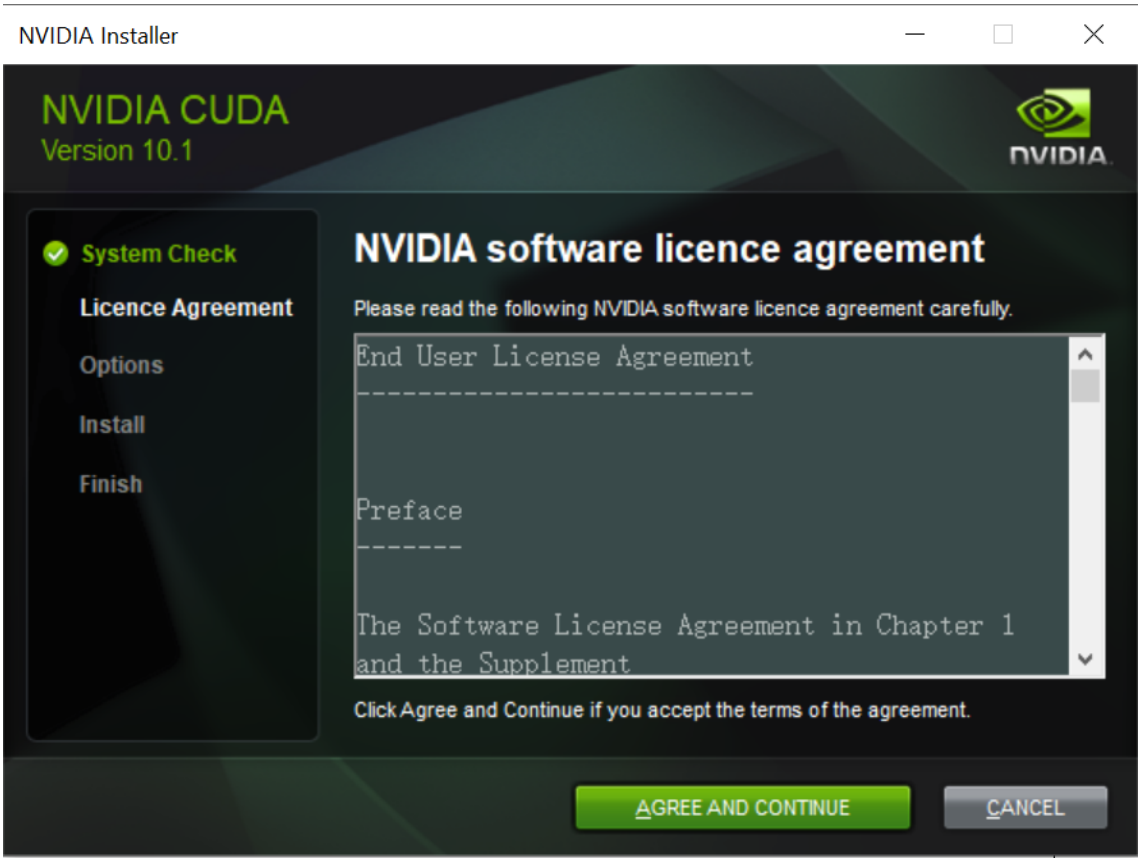


Figure 1-19

Step 6 Select **Custom** and click **NEXT**.

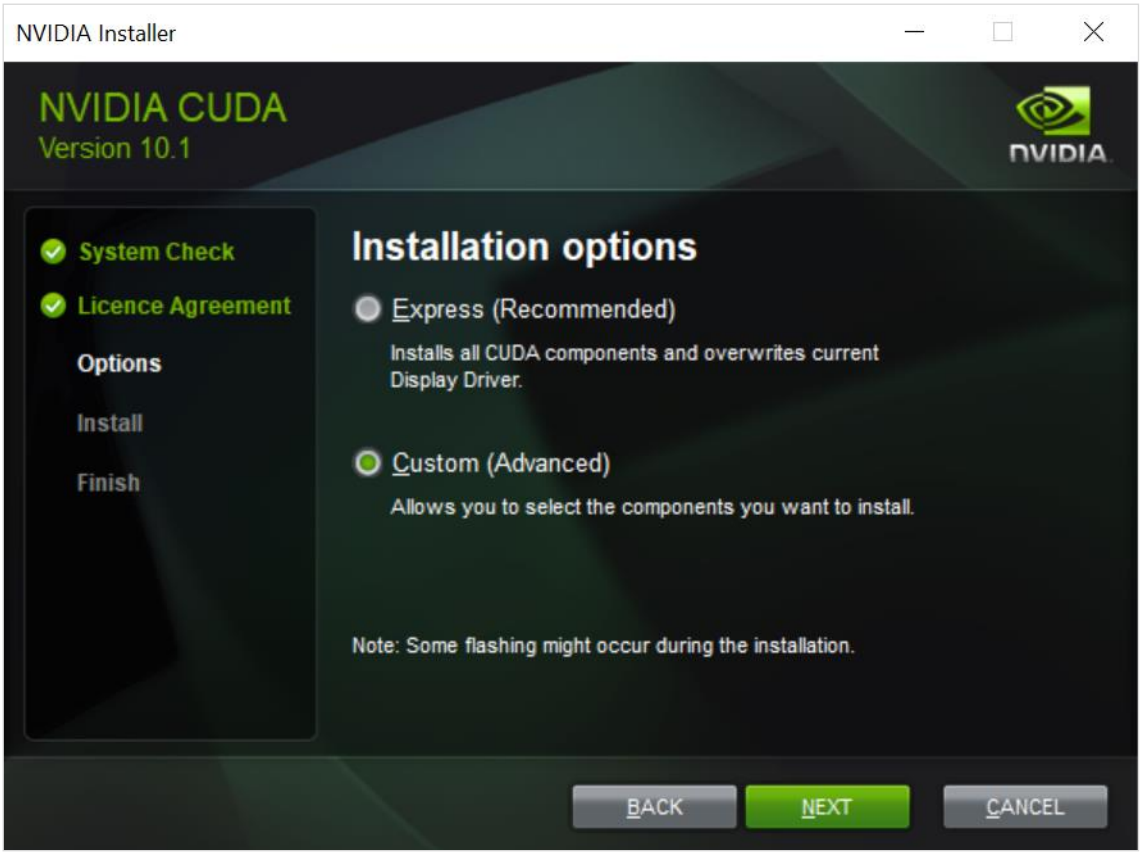


Figure 1-20

The window shown in Figure 1-21 is displayed.

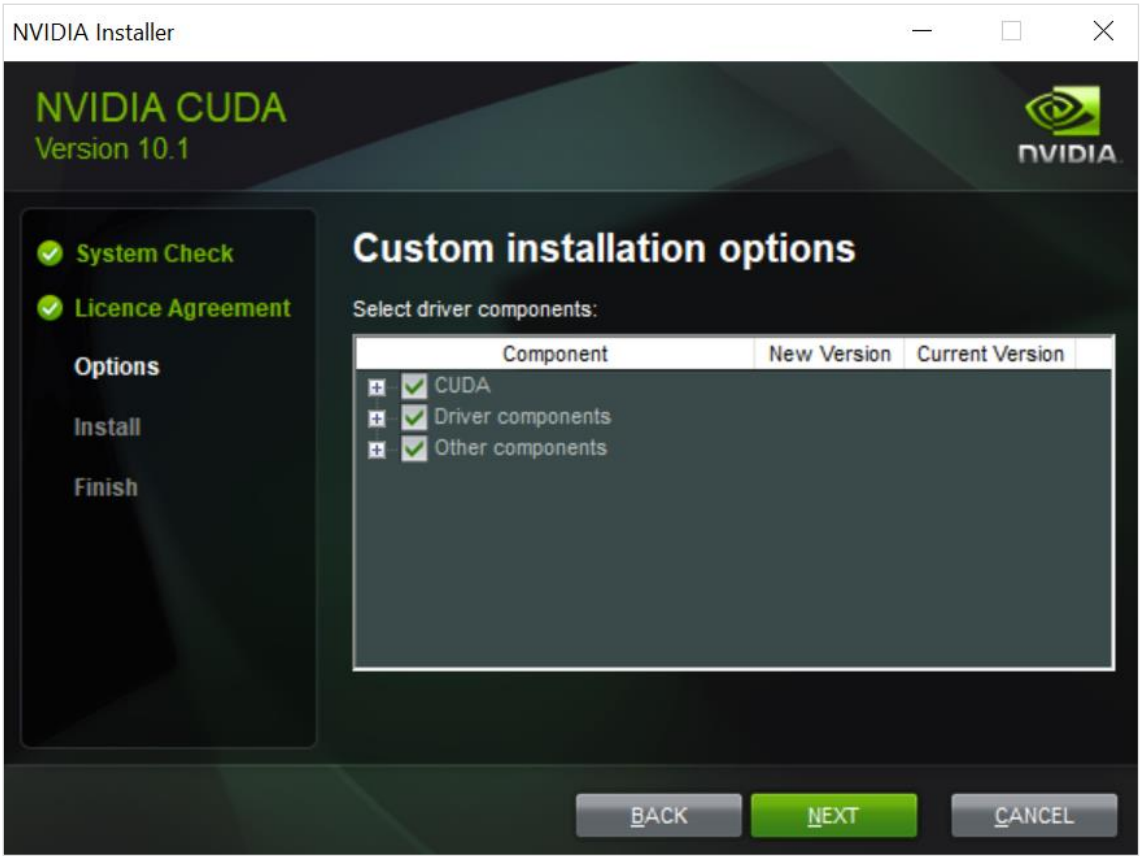


Figure 1-21

Note: If "No Visual Studio Integration" is displayed after you click **NEXT**, click the link and download Visual Studio.

Wait until the installation is complete.

Step 7 Install cuDNN. Register with the official website (https://developer.nvidia.com/cudnn) and download the cuDNN software.

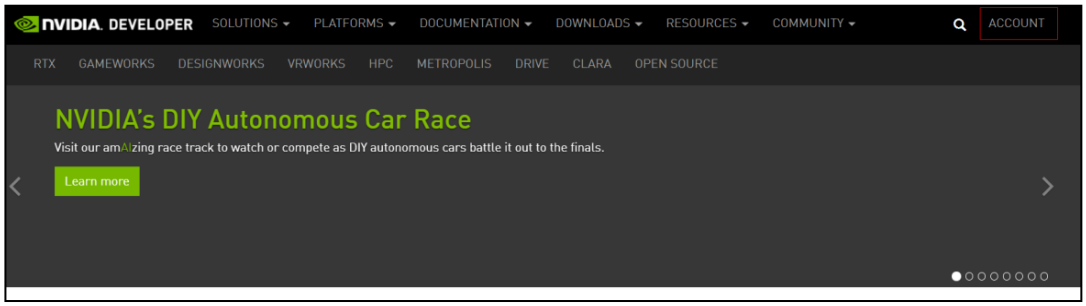


Figure 1-22

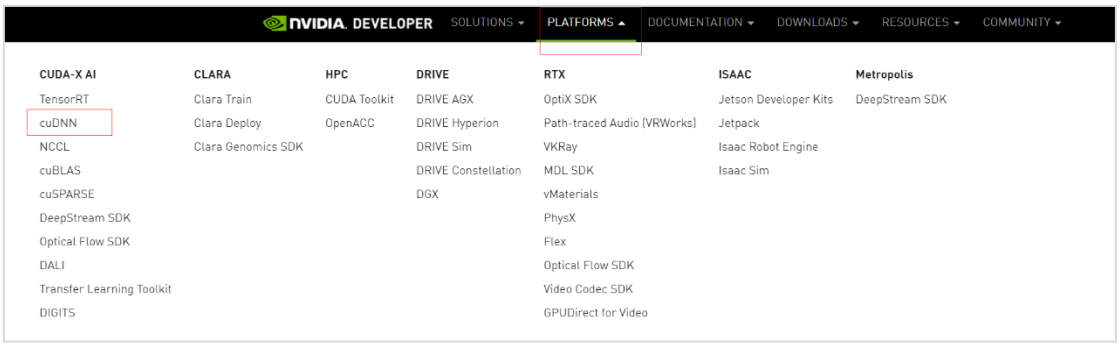


Figure 1-23

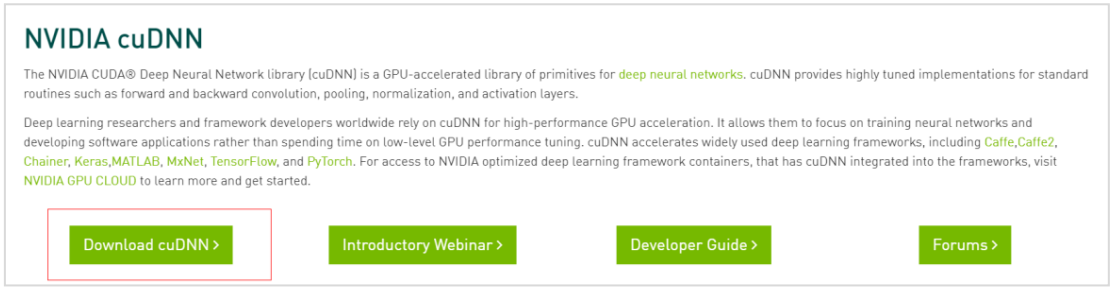


Figure 1-24

Step 8 The cuDNN version must match the CUDA version. Since CUDA 10.1 is installed, download cuDNN 7.6.5.

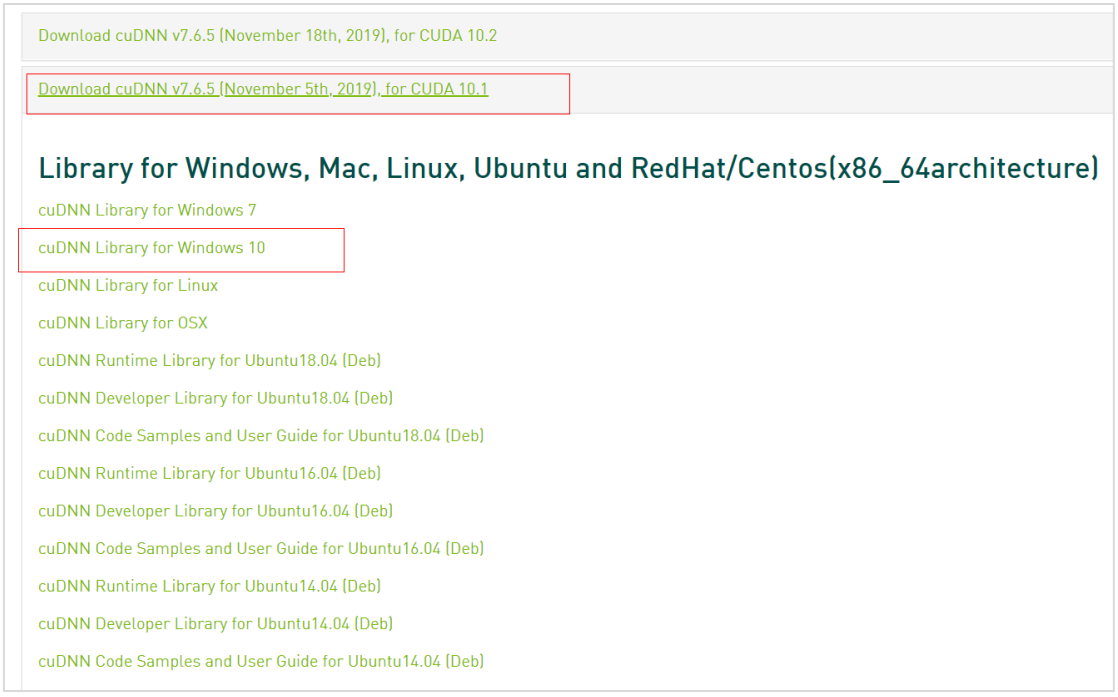


Figure 1-25

Step 9 Decompress the package downloaded.

Three folders are generated, as shown in Figure 1-26.

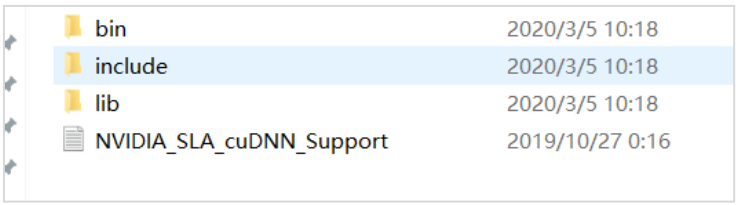


Figure 1-26

Step 10 Copy the three folders to the **CUDA** directory. If you do not change the paths, the default path is **C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.1**. Replace the folders in the directory with the three folders.

Step 11 Configure the CUDA environment variables. Search for "environment variables" in the search box and click **Open**.

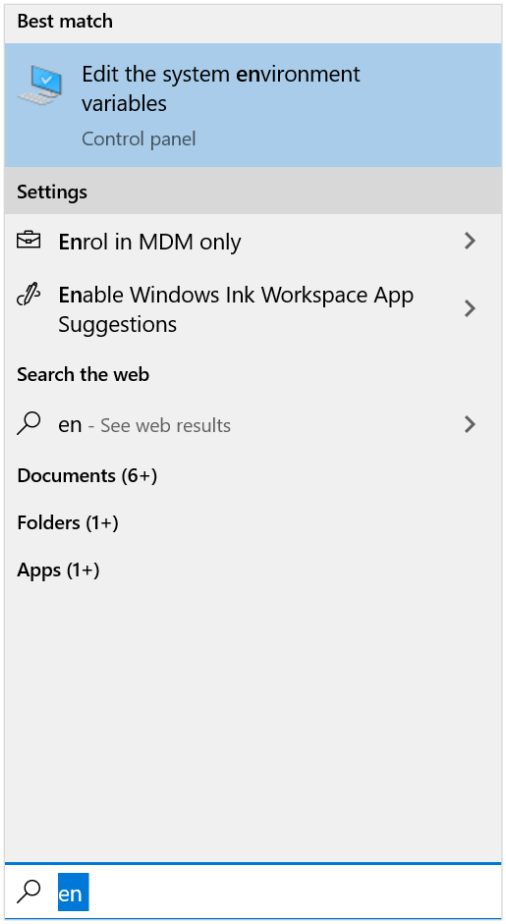


Figure 1-27

Step 12 Check whether the CUDA system variables exist, as shown in Figure 1-28.

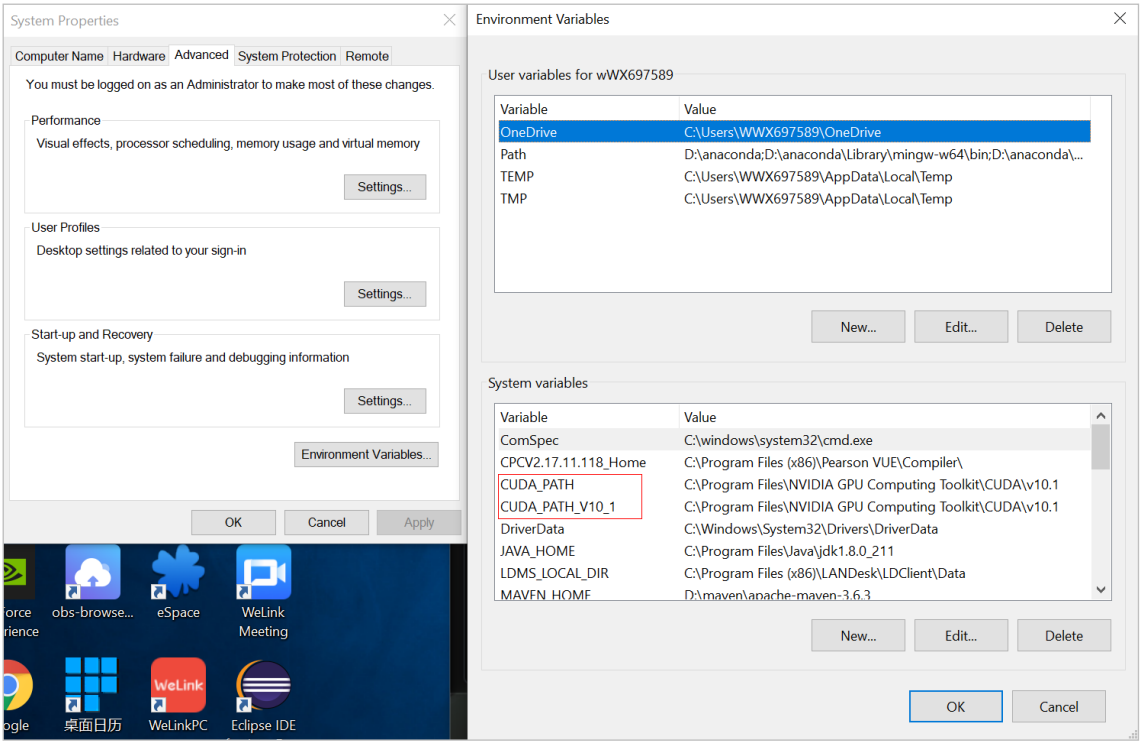


Figure 1-28

Step 13 Add the CUDA paths to **Path**. If you never change the paths, add the following paths:

- C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.1
- C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.1\bin
- C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.1\lib\x64
- C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.1\libnvvp

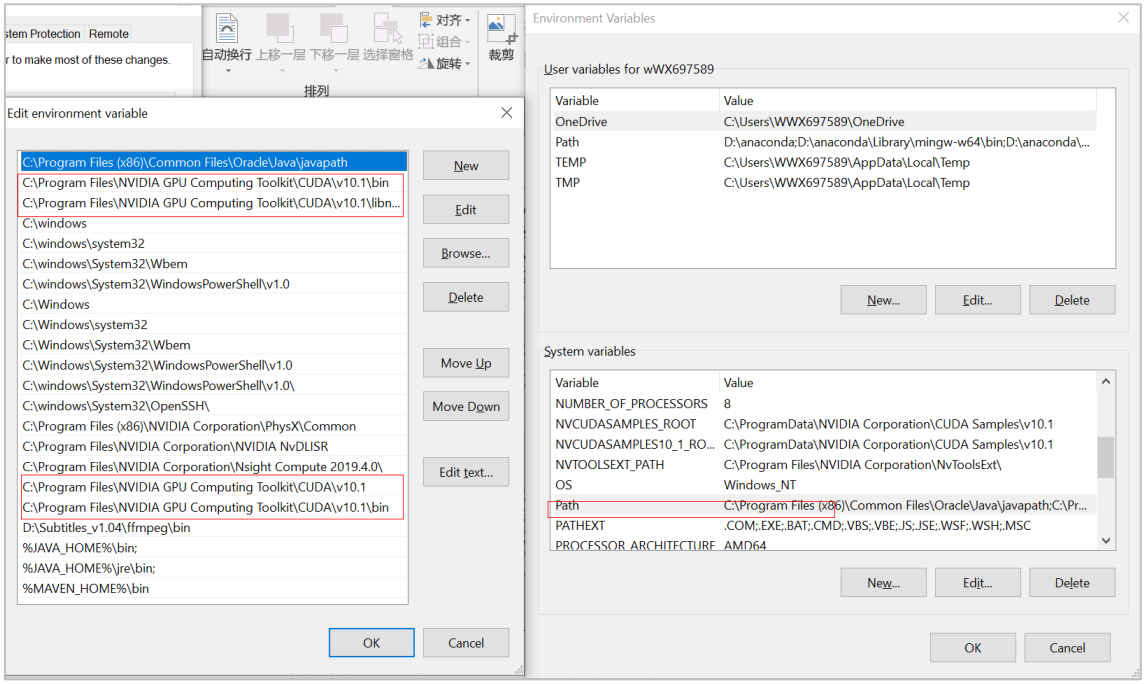


Figure 1-29

Step 14 On the CLI, run **nvcc –V**.

If information similar to Figure 1-30 is displayed, the installation is successful.

```
C:\Users\[redacted]>nvcc -V
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2019 NVIDIA Corporation
Built on Sun_Jul_28_19:12:52_Pacific_Daylight_Time_2019
Cuda compilation tools, release 10.1, V10.1.243
```

Figure 1-30

You can also run **import tensorflow** in the GPU environment and TensorFlow for GPU version. If no error is reported, the installation is successful.

Then we open the python

GPU test after importing tensorflow , True, indicates that tensorflowGPU installation was successful

```
import tensorflow as tf
tf.test.is_gpu_available()
```

```
WARNING:tensorflow:From <ipython-input-2-4a3f57d5652e>:2: is_gpu_available (fro
m tensorflow.python.framework.test_util) is deprecated and will be removed in a
future version.
Instructions for updating:
Use `tf.config.list_physical_devices('GPU')` instead.

True
```

1.3.5 Installing TensorFlow 2.0.0 for GPU

Step 1 Install TensorFlow 2.0.0 for GPU.

Open the Anaconda Terminal window from the start menu.

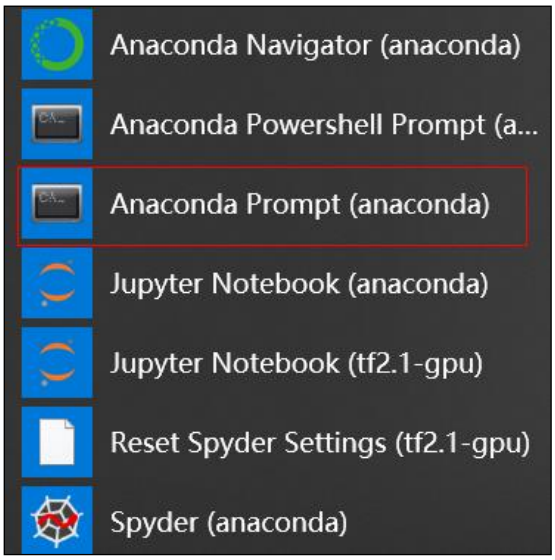


Figure 1-31

Before the installation, ensure that the computer is properly connected to the Internet.

Step 2 Run the **pip install tensorflow-gpu==2.0.0** command, as shown in Figure 1-32.

Figure 1-32

The installation is successful if information shown in Figure 1-33 is displayed.

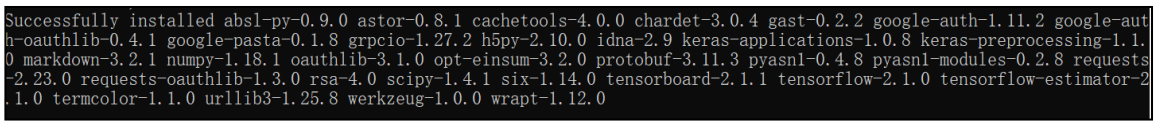


Figure 1-33

Step 3 GPU acceleration can be implemented only when the local host has an independent graphics card and CUDA and cuDNN are installed. Install CUDA first. Download the CUDA version from https://developer.nvidia.com/cuda-10.0-download-archive?target_os=Windows&target_arch=x86_64&target_version=10&target_type=exelocal.

This URL is directed to CUDA 10.0. To download other versions, click **Legacy Releases**.

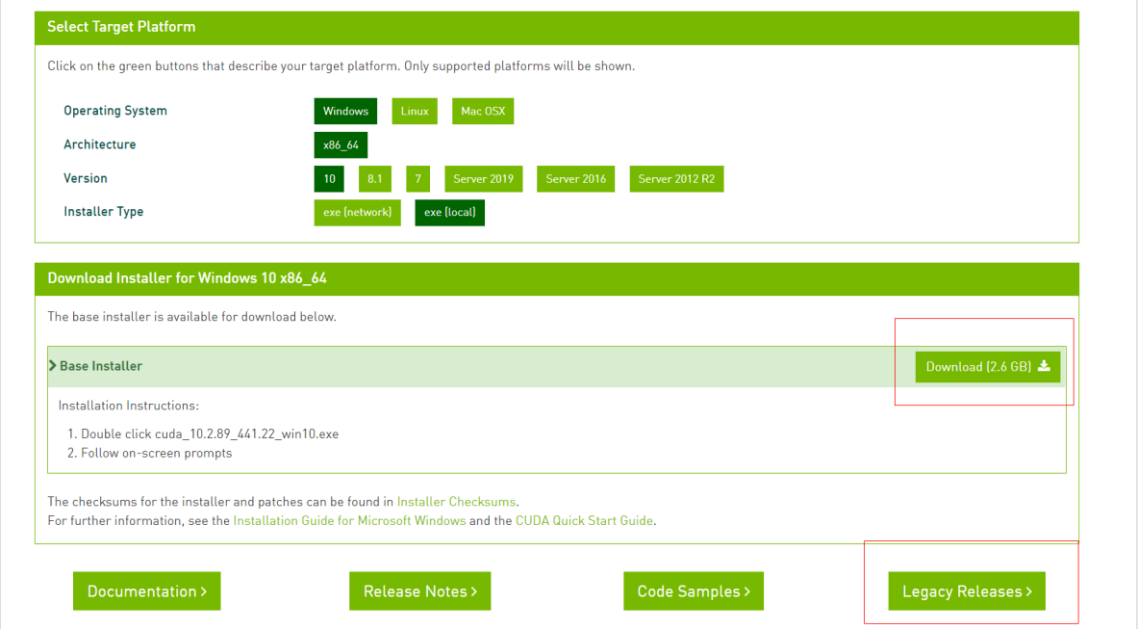


Figure 1-34

Step 4 Install CUDA. At the beginning, the program automatically decompresses the package to the default directory. Click **OK** to go to the next step.

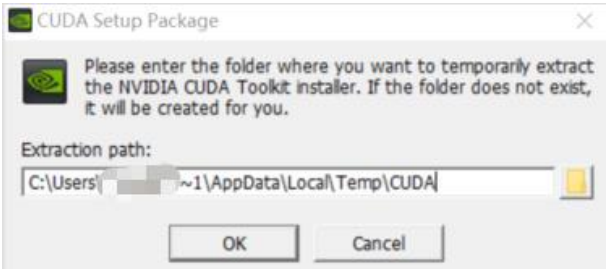


Figure 1-35

Step 5 Select **Custom** and click **NEXT**.

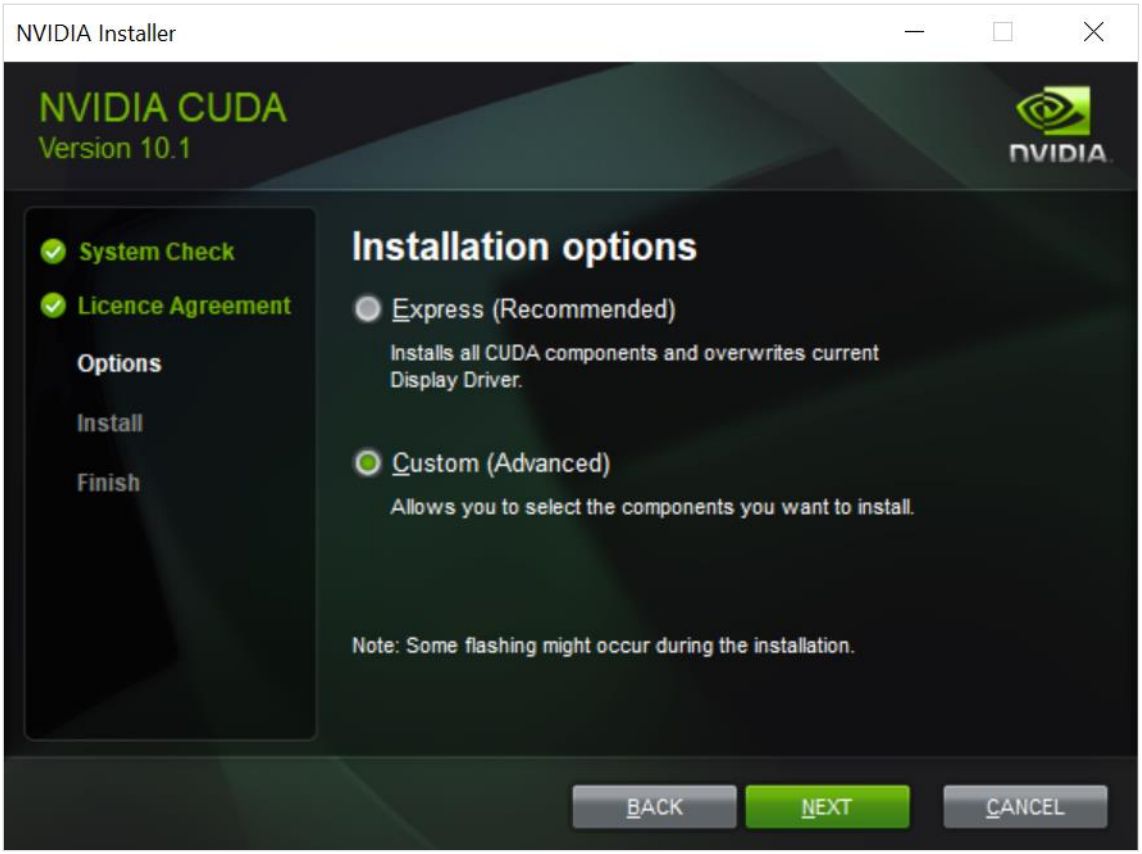


Figure 1-36

Wait until the installation is complete.

Step 6 Install cuDNN. Register with the official website (https://developer.nvidia.com/cudnn) and download the cuDNN software.

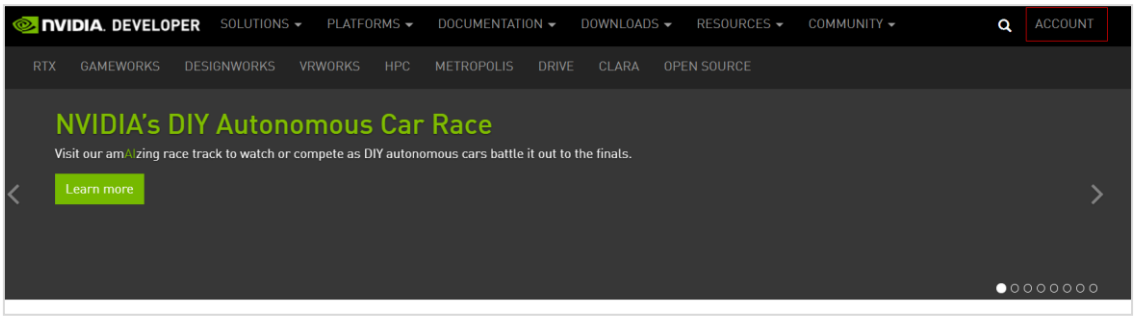


Figure 1-37

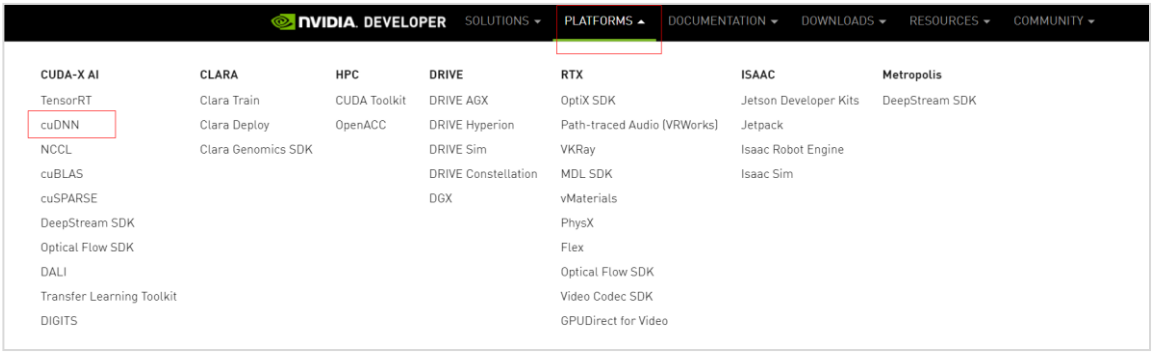


Figure 1-38

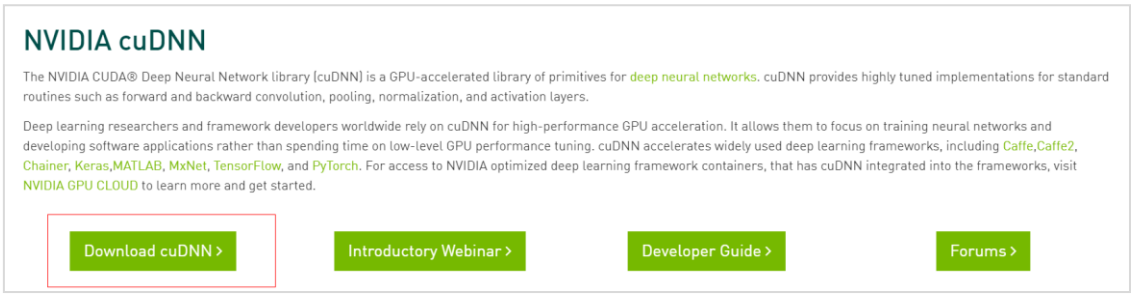


Figure 1-39

Step 7 The cuDNN version must match the CUDA version. Since CUDA 10.1 is installed, download cuDNN 7.6.5.

Figure 1-40

Step 8 Decompress the package downloaded.

Three folders are generated, as shown in Figure 1-41.

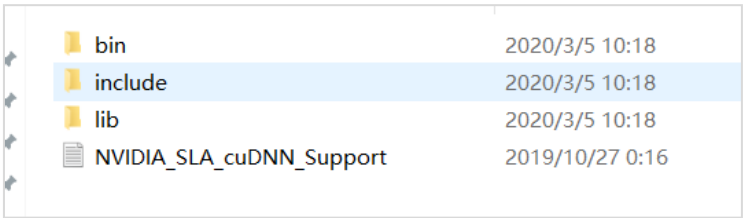


Figure 1-41

Step 9 Copy the three folders to the **CUDA** directory. If you do not change the paths, the default path is **C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.0**. Replace the folders in the directory with the three folders.

Step 10 Configure CUDA environment variables. For details, see the operation of adding environment variables in section 1.3.4 "Installing TensorFlow 2.1.0 for GPU". Note that the version number must be 10.0.

Step 11 On the CLI, run **nvcc –V**.

If information similar to Figure 1-42 is displayed, the installation is successful.

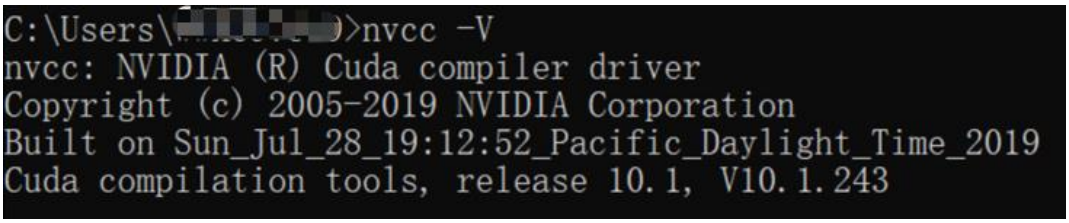


Figure 1-42

You can also run **import tensorflow** in the GPU environment and TensorFlow for GPU version. If no error is reported, the installation is successful.

Then we open the python

GPU test after importing tensorflow , True, indicates that tensorflowGPU installation was successful

```
import tensorflow as tf
tf.test.is_gpu_available()
```

```
WARNING:tensorflow:From <ipython-input-2-4a3f57d5652e>:2: is_gpu_available (fro
m tensorflow.python.framework.test_util) is deprecated and will be removed in a
future version.
Instructions for updating:
Use `tf.config.list_physical_devices('GPU')` instead.

True
```

1.4 Compiling Test Scripts in Real Time

1.4.1 Test Approach

TensorFlow introduces Keras in version 2.1.0. The test is to check whether Keras can be successfully imported and used.

1.4.2 Test Procedure

Step 1 Start Jupyter Notebook.

Start Jupyter Notebook of Anaconda from the start menu.

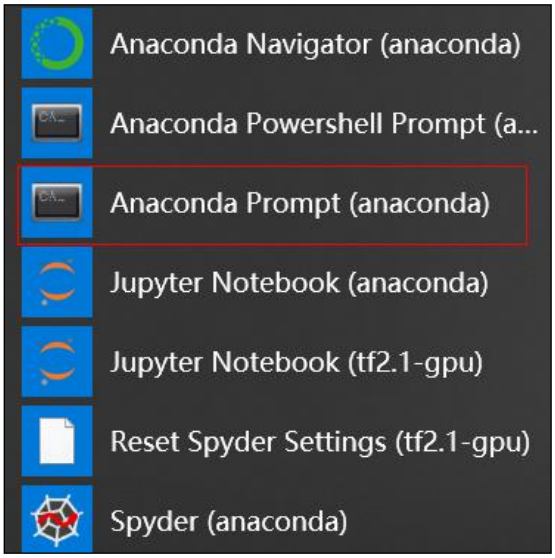


Figure 1-43

Step 2 Edit the script.

Click **New** on the right and select **Python 3**.

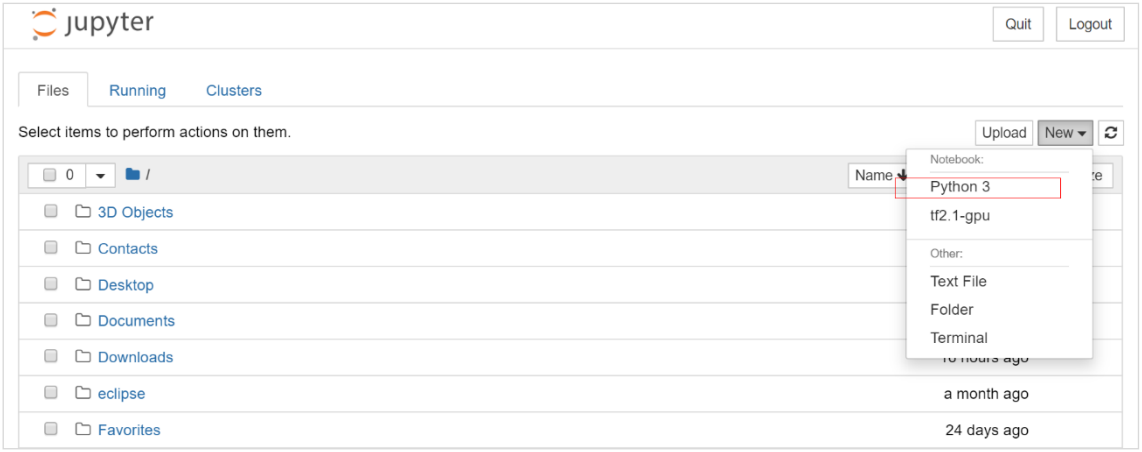


Figure 1-44

Enter the following content in the text box:

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0
model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation='softmax')])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

```
model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

Step 3 Start the test.

Click **Run** to run the Python script.

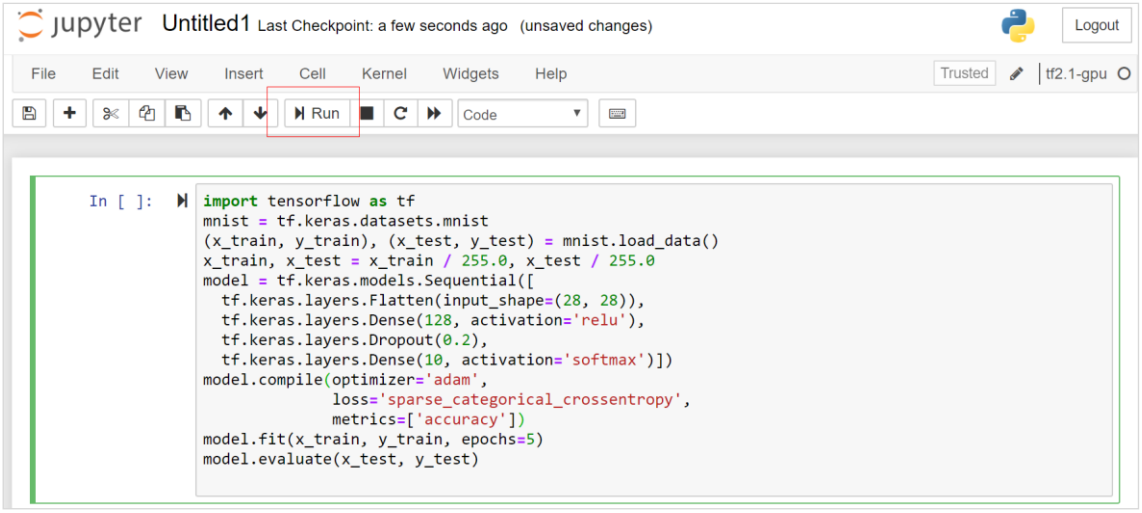


Figure 1-45

The output is as follows:

Figure 1-46

As shown in Figure 1-46, Keras is successfully imported and a basic neural network is built. After the mnist dataset is used for training, the test is complete. The test result indicates that TensorFlow 2.1.0 is successfully installed.

1.5 Anaconda Virtual Environments

1.5.1 Introduction to Virtual Environments

Anaconda is popular because of virtual environments. It allows multiple independent Python environments to be created on a host for developers to use. When the dependencies between different Python modules are disordered or different development framework applications exist, virtual environments keep these dependencies in separate sandboxes so users can switch between both applications easily and get them running.

Note: Modules in the virtual environments are independent of each other. If you have installed TensorFlow in environment A and want to use TensorFlow in environment B, you need to install TensorFlow in environment B. This also applies to Spyder and Jupyter Notebook.

1.5.2 Creating a Virtual Environment

Step 1 Start Anaconda Navigator.

Start Anaconda Navigator from the start menu.

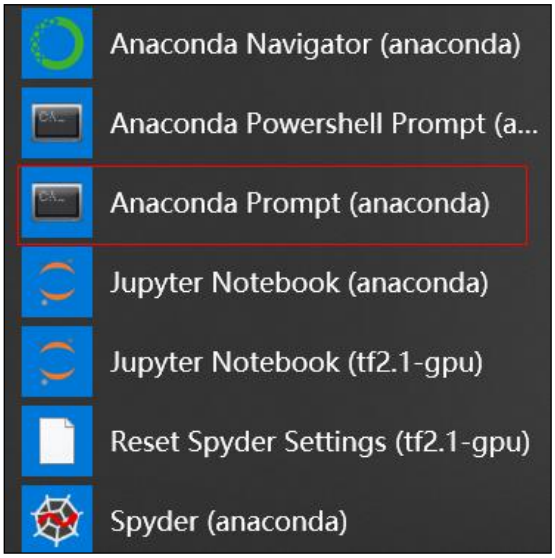


Figure 1-47

Step 2 Create a virtual environment.

Click **Environments** on the main menu, as shown in Figure 1-48.

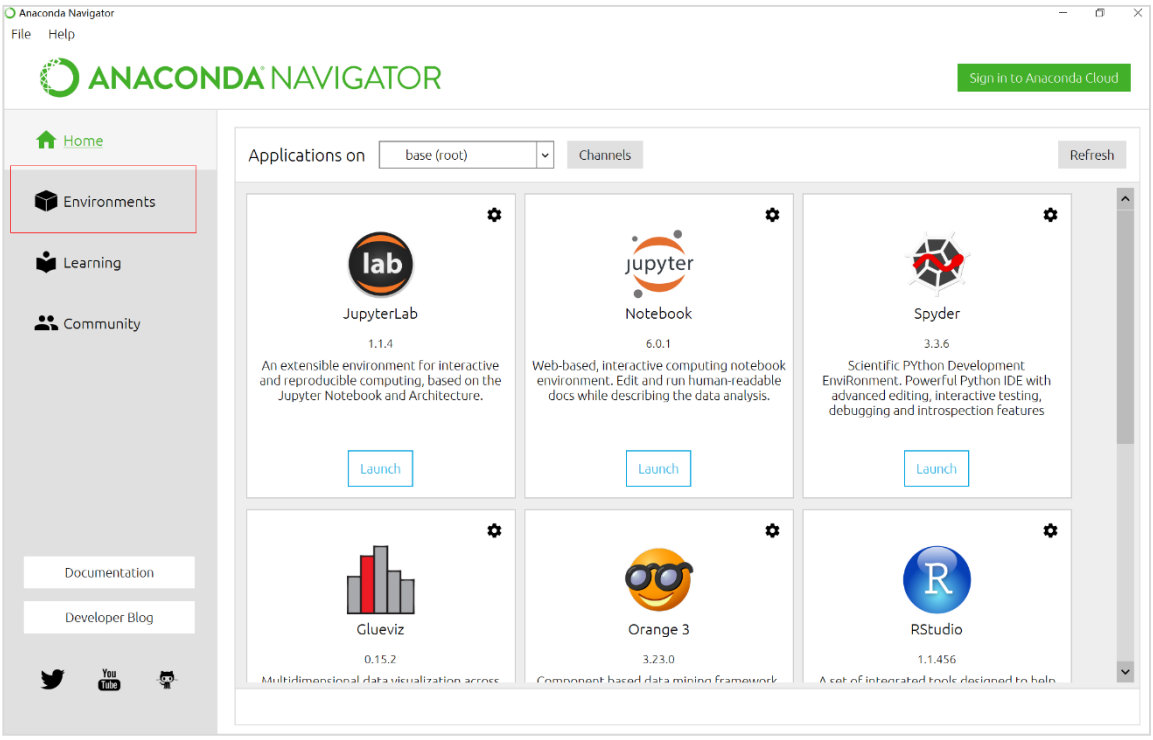


Figure 1-48

Click **Create**.

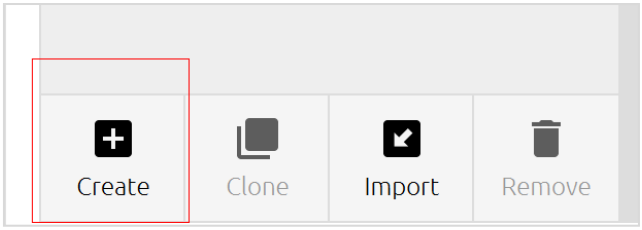


Figure 1-49

Step 3 In the dialog box that is displayed, set **Name**, and select **Python** and the version required by the development environment, for example, **3.6** or **3.7**.

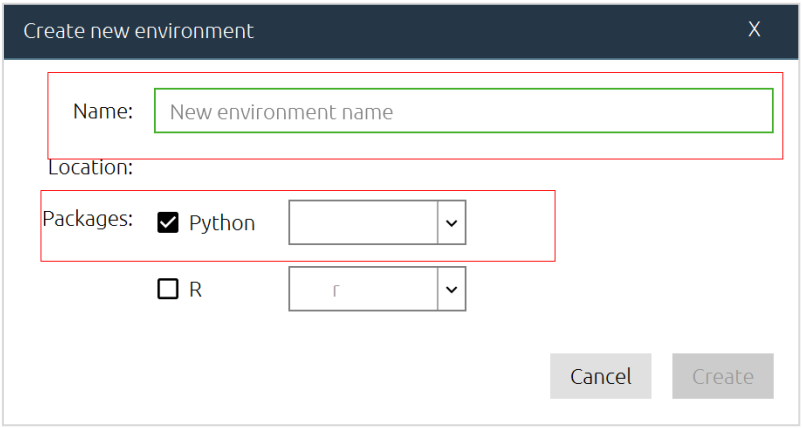


Figure 1-50

Click **Create**.

1.5.3 Creating a Virtual Environment Using the CLI

Step 1 Search for **cmd** in the search box on the taskbar and open the CLI.

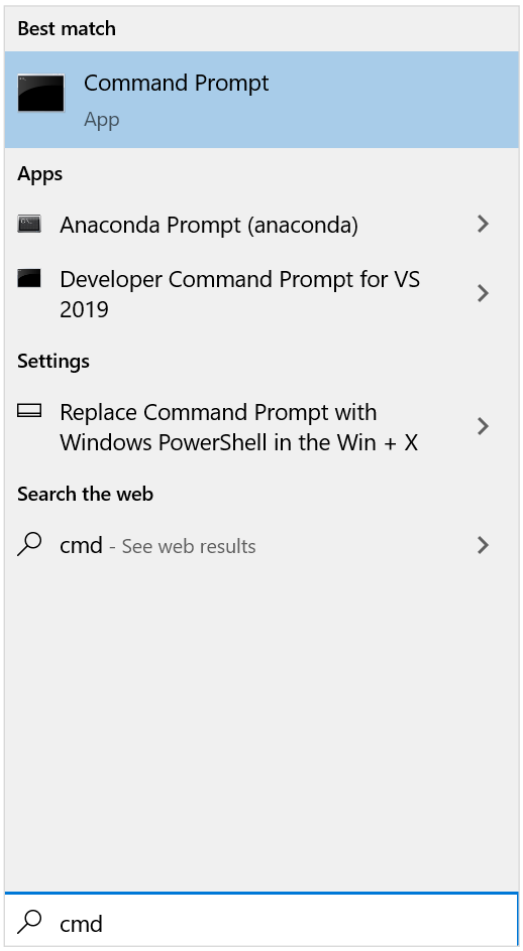


Figure 1-51

Step 2 Run the **conda create –n** *Environment name* **python==***x.x* command. Specify the environment name and Python version as required.

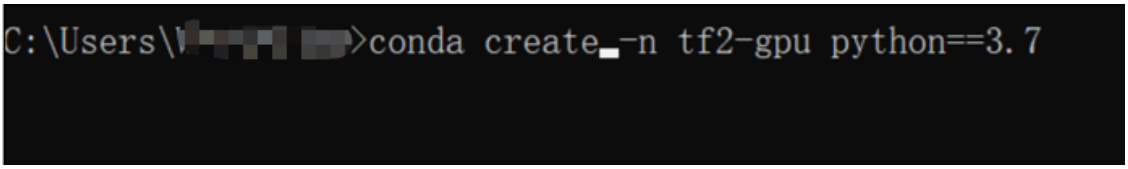


Figure 1-52

Step 3 When the information shown in Figure 1-53 is displayed, enter **y**.

```
The following NEW packages will be INSTALLED:

  certifi            pkgs/main/win-64::certifi-2019.11.28-py37_0
  pip                pkgs/main/win-64::pip-20.0.2-py37_1
  python             pkgs/main/win-64::python-3.7.0-hea74fb7_0
  setuptools         pkgs/main/win-64::setuptools-45.2.0-py37_0
  vc                 pkgs/main/win-64::vc-14.1-h0510ff6_4
  vs2015_runtime     pkgs/main/win-64::vs2015_runtime-14.16.27012-hf0eaf9b_1
  wheel              pkgs/main/win-64::wheel-0.34.2-py37_0
  wincertstore       pkgs/main/win-64::wincertstore-0.2-py37_0


Proceed ([y]/n)? y_
```

Figure 1-53

Wait until the installation is complete.

```
Downloading and Extracting Packages
python-3.7.0         | 16.6 MB   | ##########
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate tf2-gpu
#
# To deactivate an active environment, use
#
#     $ conda deactivate


C:\Users\WWX697589>_
```

Figure 1-54

1.5.4 Activating a Virtual Environment

On the CLI, run the **activate** ***Environment name*** command to activate the specified virtual environment. If the name in the brackets before the command line is changed, the virtual environment is started and entered.

```
C:\Users\[redacted]>activate tf2-gpu

(tf2-gpu) C:\Users\[redacted]>
```

Figure 1-55

You can run the **pip** or **conda** command to install modules.

1.5.5 Viewing Virtual Environments

You can query created virtual environments on Anaconda Navigator.

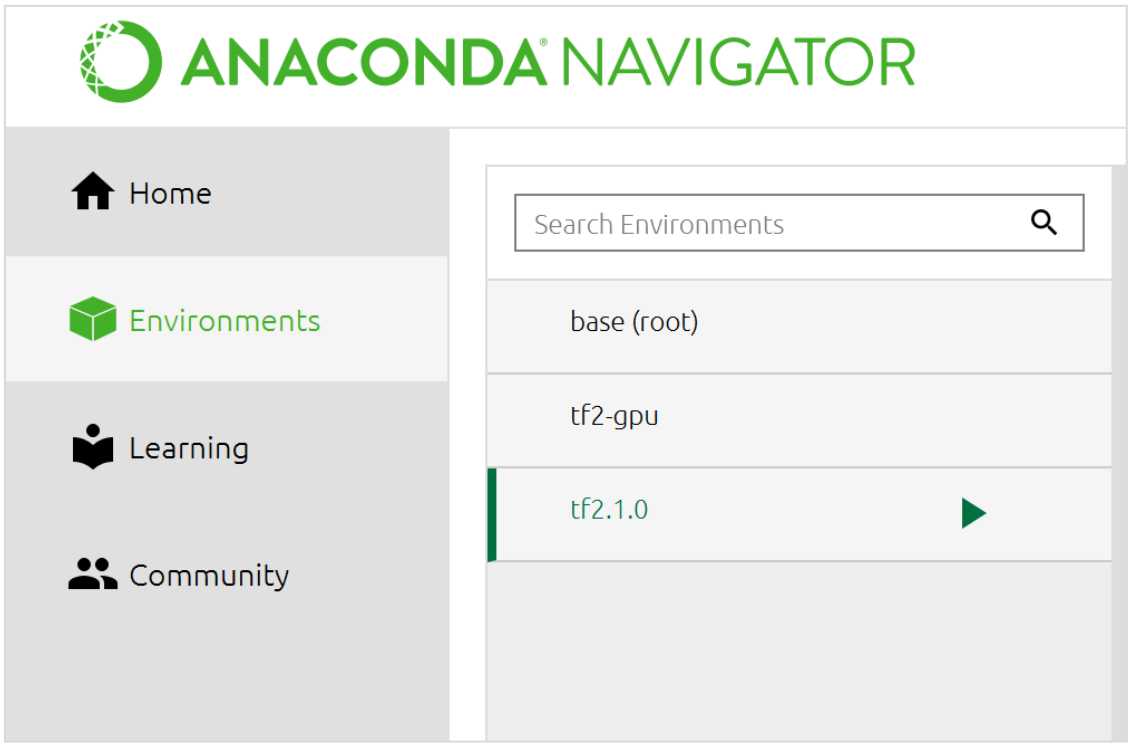


Figure 1-56

You can click the triangle on the right of the environment name to access the CLI of the virtual environment.

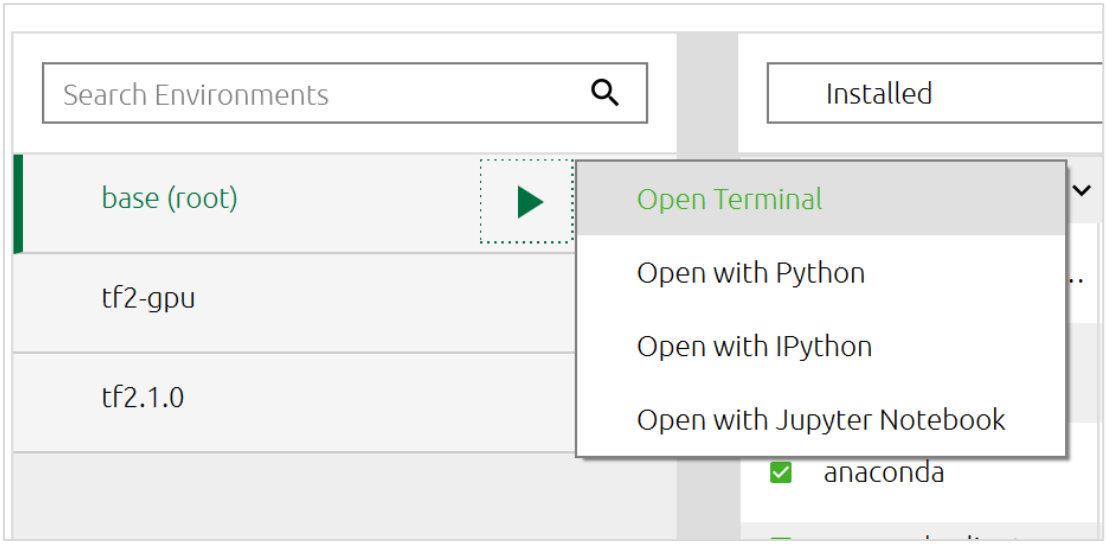


Figure 1-57

In this case, you can install modules in the current environment without activating the environment.

1.5.6 Deleting a Virtual Environment

Run the **conda remove –n** *Environment name* **–all** command and enter **y** to delete the environment.

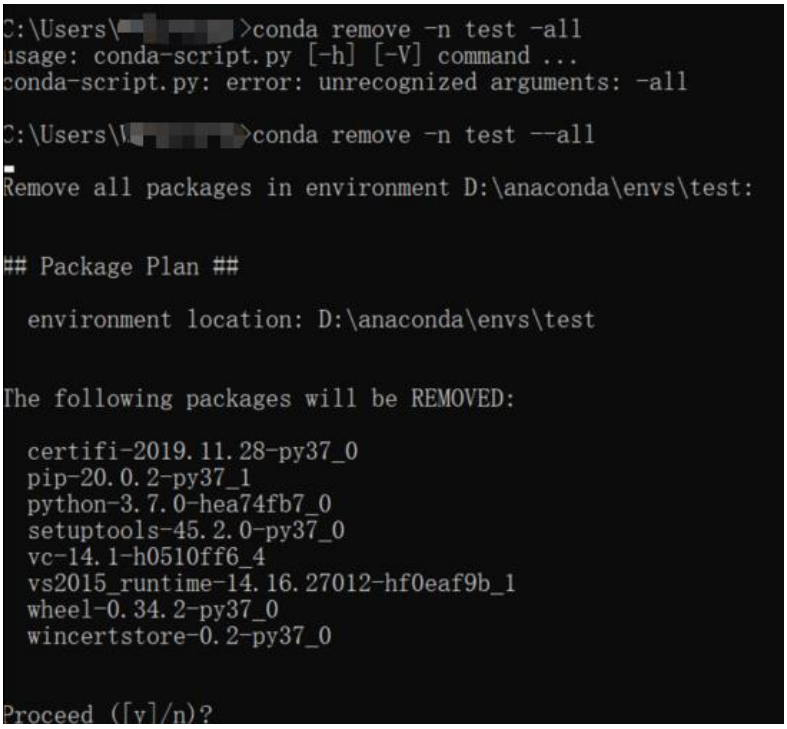


Figure 1-58

2 Configuring the macOS Experiment Environment

2.1 Introduction

2.1.1 About This Experiment

In this experiment, you need to set up a development environment for all HCIA-AI experiments based on macOS. You need to download and install Anaconda, select the Python version, install dependencies, and install Jupyter Notebook.

2.1.2 Objectives

Set up an HCIA-AI experiment environment of the CPU version based on macOS.

2.1.3 Modules Required

- Anaconda 3.7
- Python 3.7
- TensorFlow 2.1.0 or 2.0.0

2.2 Downloading Anaconda and Configuring the Python Environment

Anaconda is a distribution of Python for scientific computing. It supports Linux, macOS, and Windows. It provides simplified package management and environment management, and easily deals with the installation issues when the system has multiple Python versions and third-party packages. Anaconda uses the Conda command to manage packages and environments. It contains Python and related tools. Anaconda is a Python tool for enterprise-level big data analytics. It contains more than 720 open-source data-science packages, including data visualization, machine learning, and deep learning. It can be used for data analysis, big data and AI fields.

After installing Anaconda, you do not need to install Python.

2.2.1 Downloading Anaconda

Step 1 Visit https://www.anaconda.com/, click **Download**, and download the macOS version, as shown in Figure 2-1.

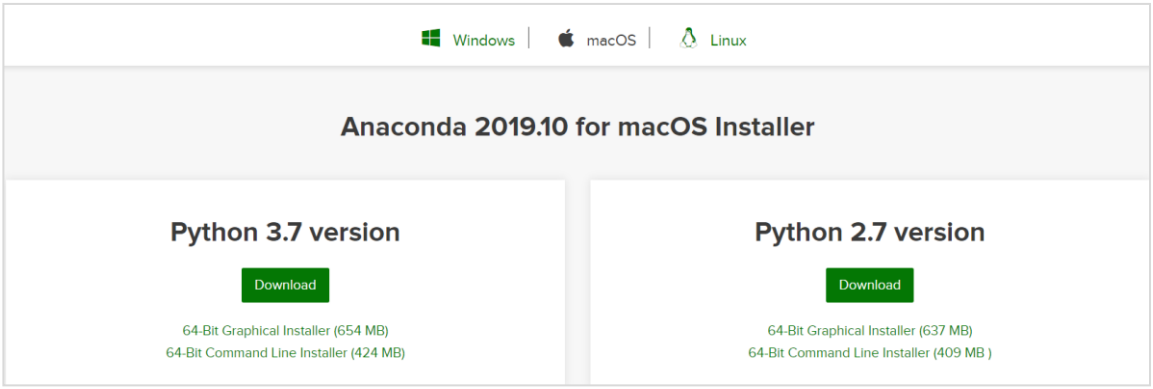


Figure 2-1

Download the Python 3.7 installation package of 654 MB.

2.2.2 Installing Anaconda

Step 1 Double-click the downloaded Anaconda installation package in .pkg format to install it.

Step 2 Click **Continue** and **Install** buttons repeatedly.

Step 3 Enter the system password.

Wait until the installation is complete.

2.2.3 Creating a Python Virtual Environment

Step 1 Start Anaconda Navigator.

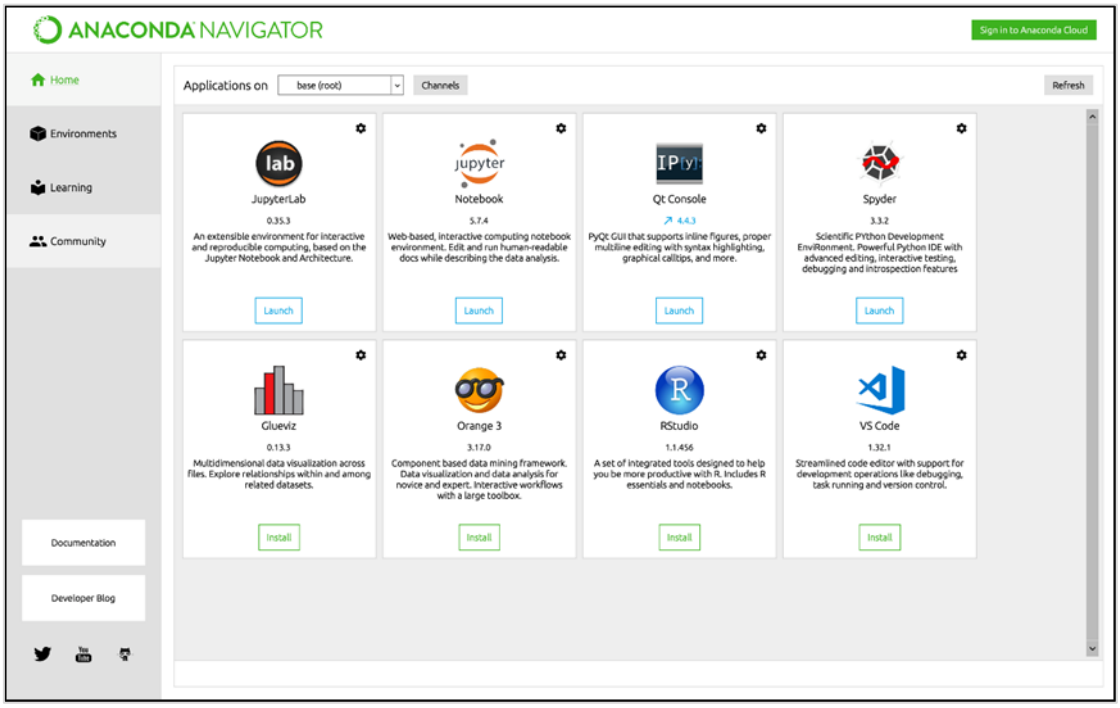


Figure 2-2

Step 2 Click **Environments** on the left. On the **Environments** page that is displayed, click **Create** in the lower left corner.

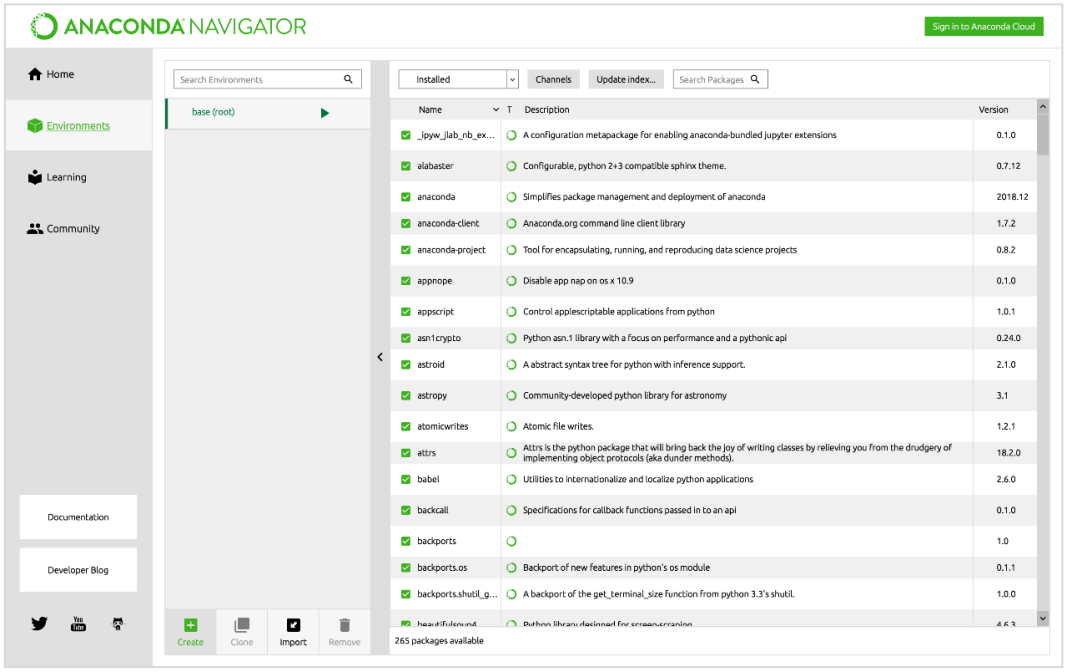
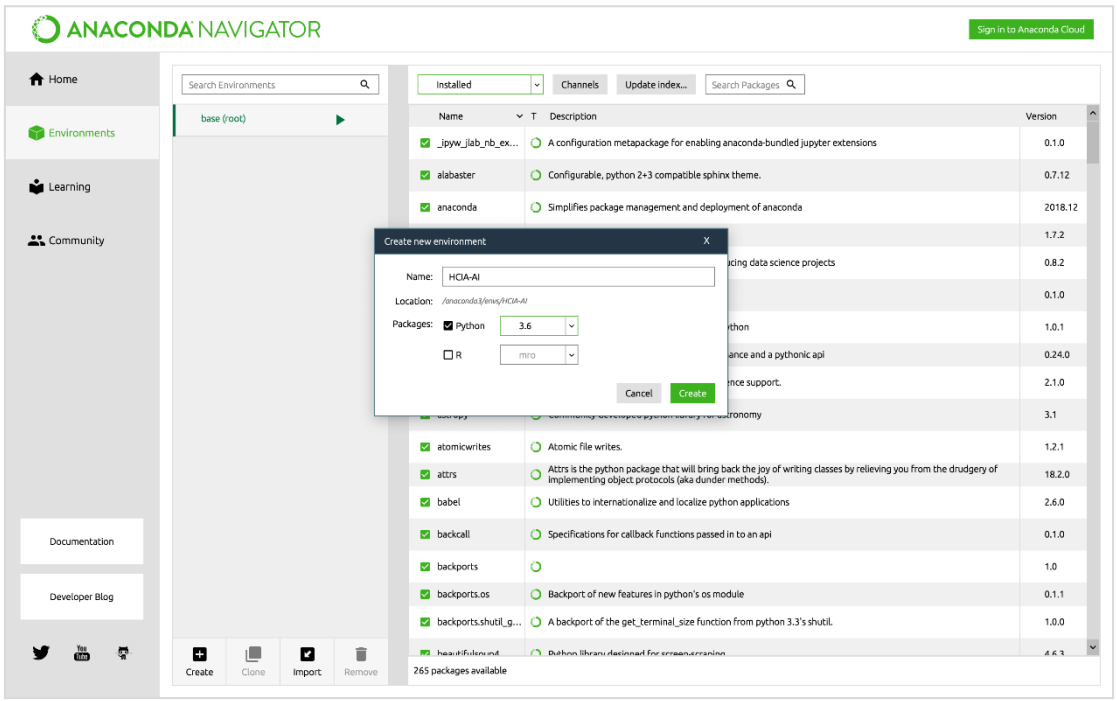


Figure 2-3

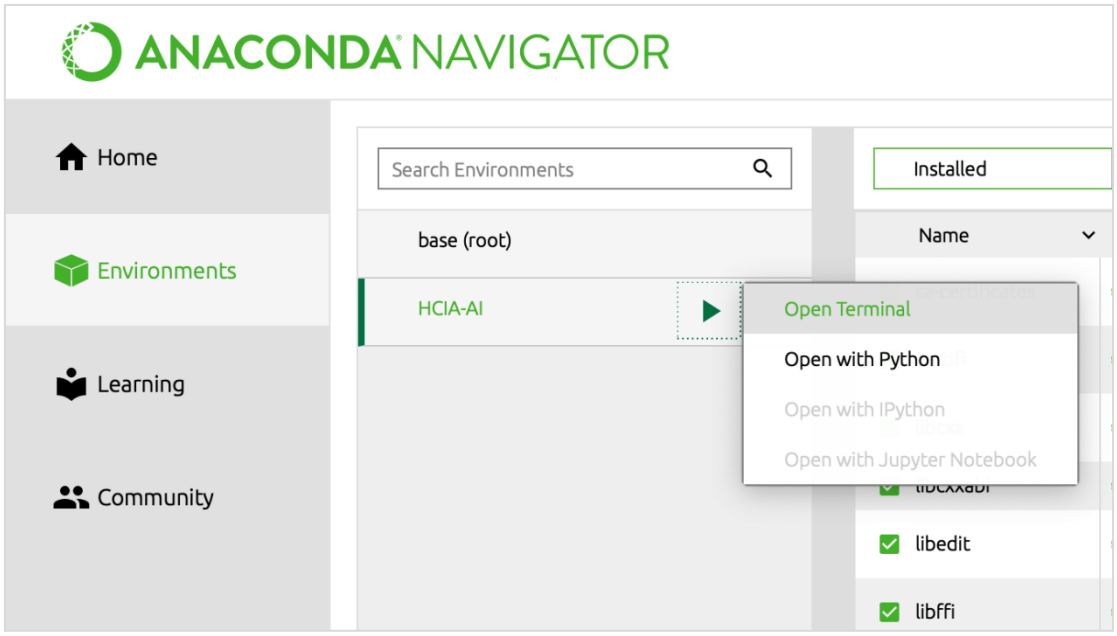
Step 3 Set the environment name, and select **Python** and its version. Version 3.6 is recommended.



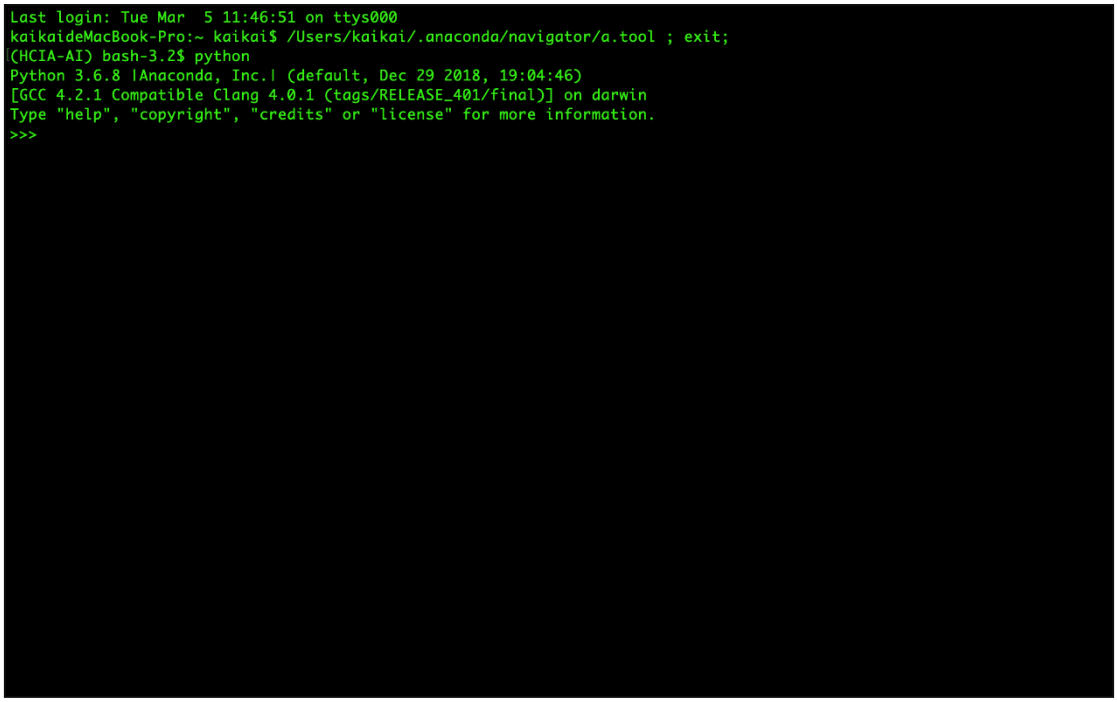
Wait until the environment is created successfully.

2.2.4 Testing the Environment

Step 1 Click the triangle next to the environment name to access the newly created environment and select **Open Terminal**.



Step 2 Enter **python** to check the Python version in the environment.



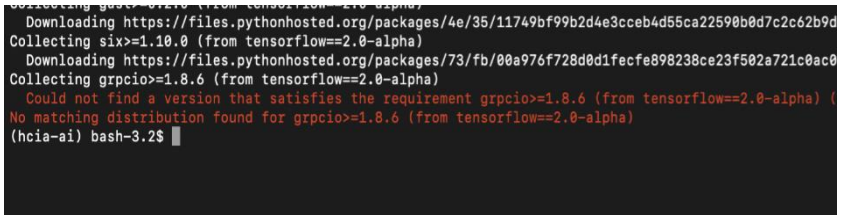
"Python 3.6.8" is displayed, which indicates that the installation is successful.

2.3 Installing TensorFlow

2.3.1 Installing TensorFlow 2.1.0

Step 1 Run the **pip install tensorflow-cpu** command to install the TensorFlow2.1.0 framework.

Step 2 If the following information is displayed, there are package missing. Manually install the corresponding packages.



Step 3 After the packages are installed, run **pip install tensorflow==2.1.0**, and wait until the installation is complete.

2.3.2 Installing TensorFlow 2.0.0

Run the **pip install tensorflow==2.0.0** command to install TensorFlow 2.0.0. For details, see "Installing TensorFlow 2.1.0".

2.4 Installing Jupyter Notebook

Jupyter Notebook can be installed by using the Terminal or Anaconda Navigator.

2.4.1 Installing Jupyter Notebook Using the Terminal

Run the **pip install jupyter notebook** command on the Terminal.

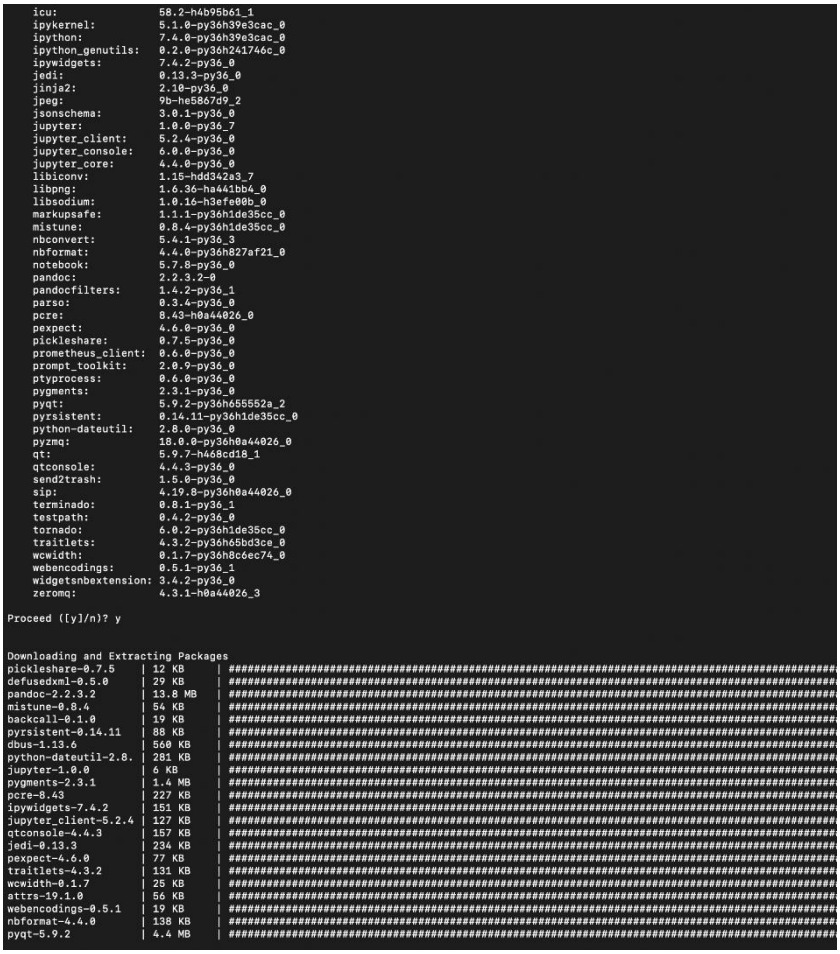


Figure 2-4

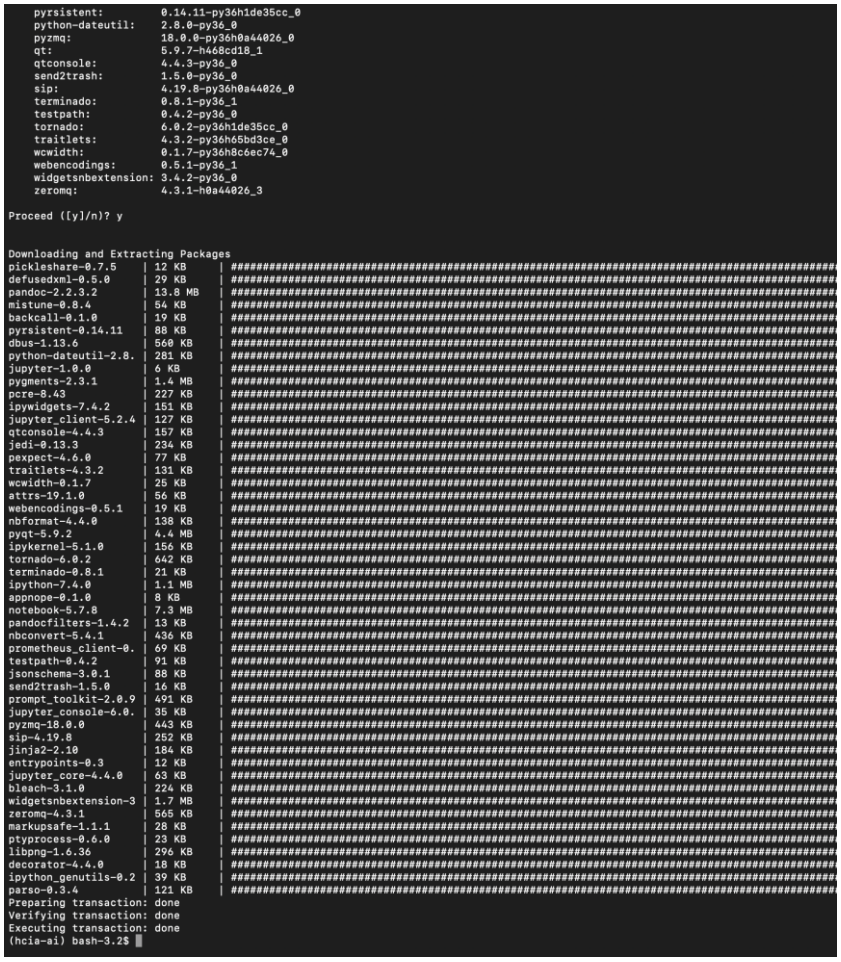


Figure 2-5

Wait until the installation is complete.

2.4.2 Installing Jupyter Notebook Using Anaconda Navigator

Step 1 Start Anaconda.

On the Anaconda home page, click **Home** on the left.

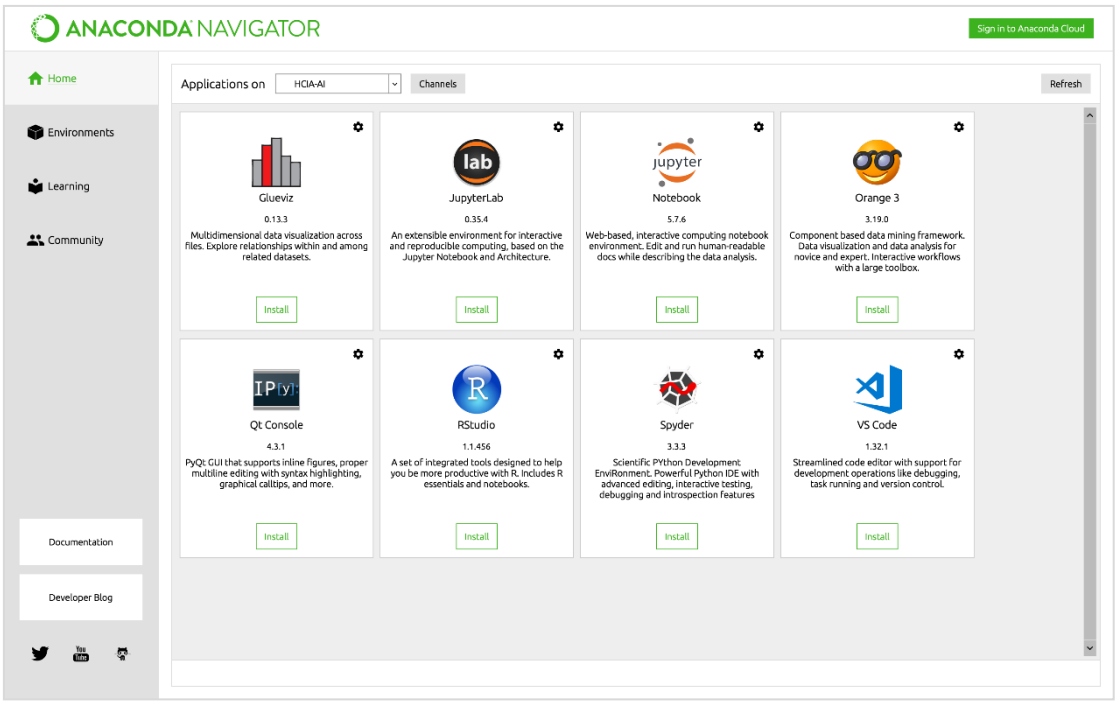


Figure 2-6

Step 2 Install Jupyter Notebook.

Set **Application on** to **HCIA-AI** and click **Install** under **Jupyter Notebook** on the right. After the installation is complete, the page shown in Figure 2-7 is displayed.

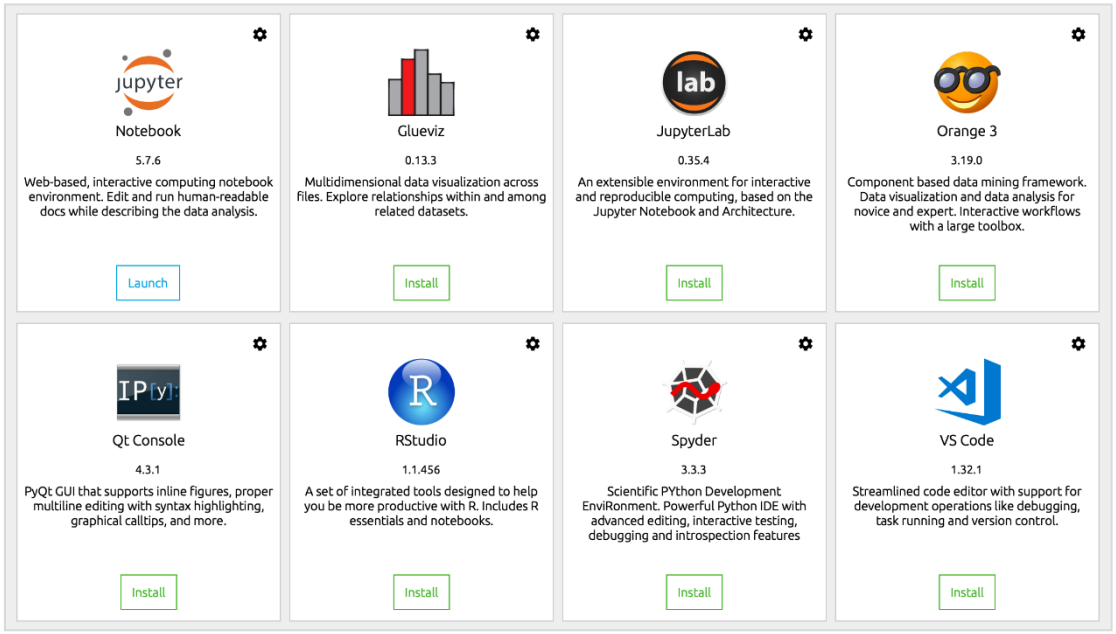


Figure 2-7

The **Install** button under **Jupyter Notebook** changes to **Launch**.

Step 3 Verify the installation.

Click **Launch** under **Jupyter Notebook**. The Jupyter home page is displayed.

Figure 2-8

Click **New** in the upper right corner of the page and select **Python 3** to create a Jupyter file.

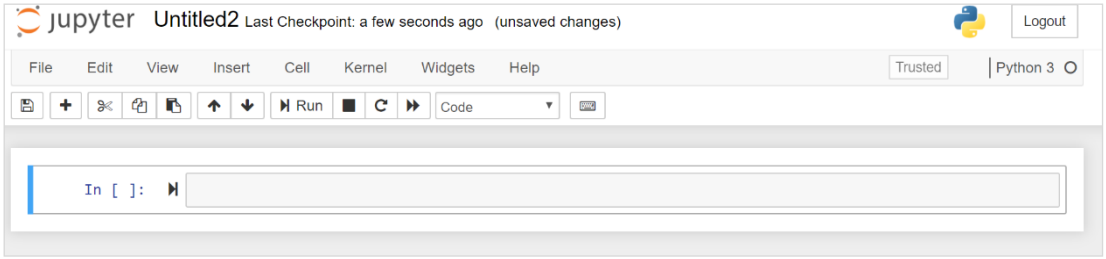


Figure 2-9

Enter the following content in the text box:

```
for i in range(100):
if i%2 == 0:
    print(i)
```

Click **Run**, as shown in Figure 2-10.

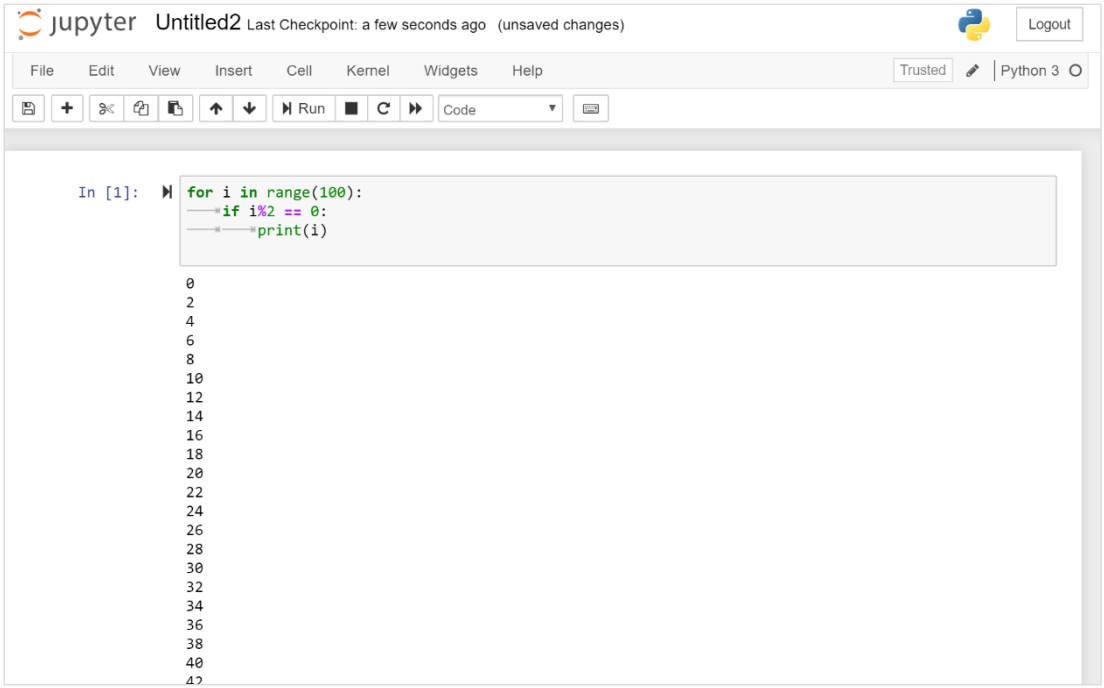


Figure 2-10

2.5 Performing Test Cases

Verify that Python 3.6 and TensorFlow development environments created based on Anaconda can run properly.

To be more specific, verify that the graph mechanism can be used to perform constant additions based on Python 3.6, Jupyter Notebook, and TensorFlow frameworks.

2.5.1 Test Case

Step 1 Start Jupyter.

Click **Launch**.

Figure 2-11

Step 2 Create a Jupyter project.

Click **New** in the upper right corner and select **Python 3** to create a script.

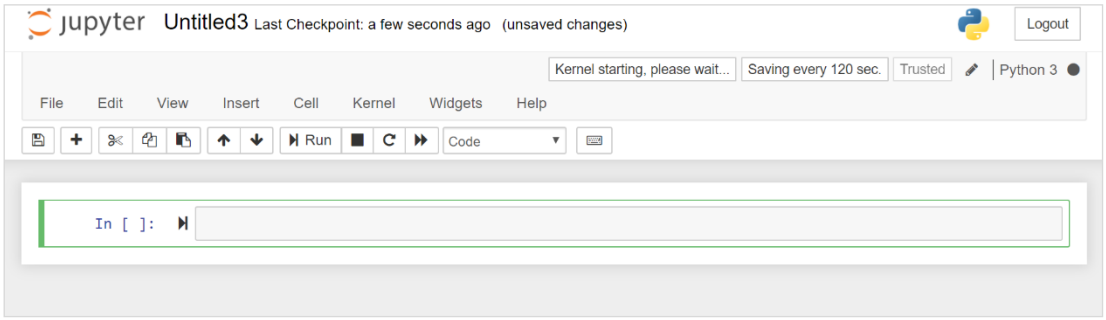


Figure 2-12

Click **Untitled3** in the upper left corner and name the script **tf_demo**.

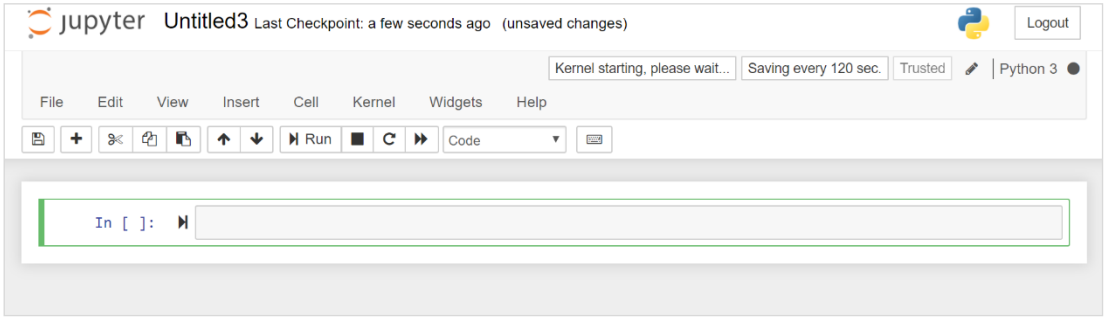


Figure 2-13

Enter the following code:

```
# Import modules.
import tensorflow as tf
import numpy as np

# Create two constant tensors.
mat1 = tf.constant(np.array([[1.0, 2.0]]))
mat2 = tf.constant(np.array([[2.],[2.]]))

# Create a multiplication operation.
mul = tf.matmul(mat1,mat2)

print(mul)
```

Result:

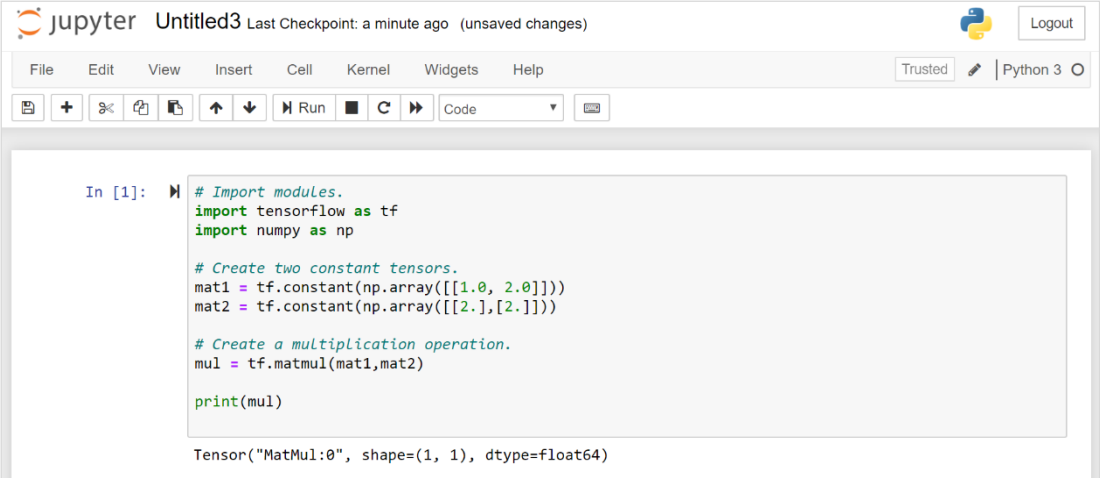


Figure 2-14

Step 3 Verify Keras.

Verify the Keras module, which is a new feature introduced in TensorFlow 2.0.

Enter the following code:

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0
model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss=['sparse_categorical_crossentropy'],
              metrics=['accuracy'])
model.fit(x_train, y_train, epochs=5)

model.evaluate(x_test, y_test)
```

Click **Run** above the code. The information shown in Figure 2-15 is displayed.

```
In [4]: import tensorflow as tf
        mnist=tf.keras.datasets.mnist
        (x_train,y_train),(x_test,y_test)=mnist.load_data()
        x_train,x_test=x_train/255.0,x_test/255.0

        Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
        11493376/11490434 [==============================] - 16s 1us/step

In [9]: model=tf.keras.models.Sequential([
            tf.keras.layers.Flatten(input_shape=(28,28)),
            tf.keras.layers.Dense(128,activation='relu'),
            tf.keras.layers.Dropout(0.2),
            tf.keras.layers.Dense(10,activation='softmax'),
        ])

In [19]: model.compile(optimizer='adam',
                       loss=['sparse_categorical_crossentropy'],
                       metrics=['accuracy'],)
         model.fit(x_train,y_train,epochs=5)

         Epoch 1/5
         60000/60000 [==============================] - 2s 37us/sample - loss: 0.2982 - accuracy: 0.9131
         Epoch 2/5
         60000/60000 [==============================] - 2s 33us/sample - loss: 0.1423 - accuracy: 0.9582
         Epoch 3/5
         60000/60000 [==============================] - 2s 33us/sample - loss: 0.1093 - accuracy: 0.9667
         Epoch 4/5
         60000/60000 [==============================] - 2s 33us/sample - loss: 0.0893 - accuracy: 0.9722
         Epoch 5/5
         60000/60000 [==============================] - 2s 33us/sample - loss: 0.0758 - accuracy: 0.9758

Out[19]: <tensorflow.python.keras.callbacks.History at 0x10971f898>

In [20]: model.evaluate(x_test,y_test)

         10000/10000 [==============================] - 0s 24us/sample - loss: 0.0814 - accuracy: 0.9743

Out[20]: [0.08144361303178593, 0.9743]

In [ ]:
```

Figure 2-15

This experiment verifies whether the HCIA-AI experiment environment is successfully set up. If no error is reported during code running, the experiment environment is successfully set up.