

ai-3. Python, TensorFlow, Keras のインストール

(人工知能を実践と演習で学ぶシリーズ)

<https://www.kkaneko.jp/cc/ai/index.html>

金子邦彦



はじめに



- **TensorFlow, Keras を使いたい, それらのインストールがどのようなものか知りたい人向け**
- **NVIDIA グラフィックボードを使わない人は, 次を無視**
 - ビルドツール
 - NVIDIA CUDA ツールキット **10.1**
 - NVIDIA cuDNN **7**

注意点



- 単にインストールするだけ

注意点



1. 複数のソフトが関係する
2. Window のコマンドプロンプト
3. Python のパッケージ管理
4. バージョン指定
NVIDIA CUDA ツールキット 10.1, NVIDIA cuDNN 7 は, **バージョン指定** (最新バージョンは NG)
5. NVIDIA cuDNN 7 のダウンロードには, **登録が必要**
6. **いったん, アンインストールが必要かも**

複数のソフトが関係する



Python 開発環境

TensorFlow, Keras

NVIDIA cuDNN 7

pip

NVIDIA CUDA ツールキット 10.1

Python

ビルドツール

NVIDIA グラフィックスボード・ドライバ

Windows

プロセッサ

GPU
(グラフィックスボード)

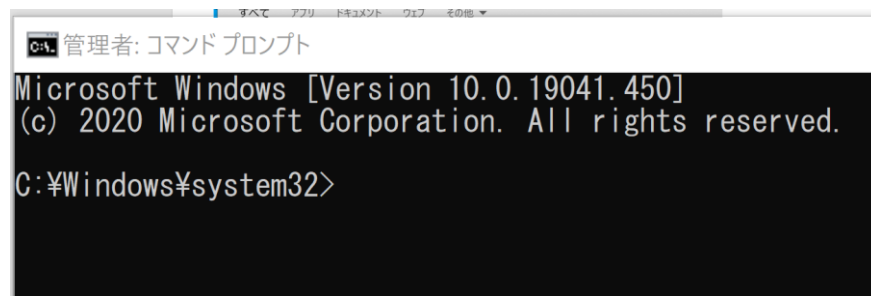
Windows のコマンドプロンプト



さまざまなコマンドを操作を行うため



cmd.exe



コマンドプロンプトを管理者として実行

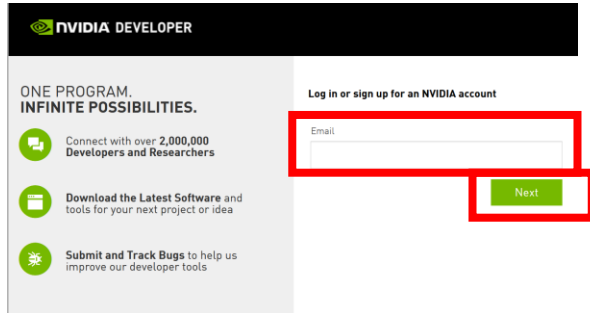
- TensorFlow や Keras は Python のパッケージ
- Python は, プログラミングのためのシステム
- インストールには pip コマンドを使用
- Python のインストール時に, パッケージをシステム領域に置くか, ユーザ領域に置くかを選ぶ.
システム領域に置く方がトラブルが少ない

バージョン指定

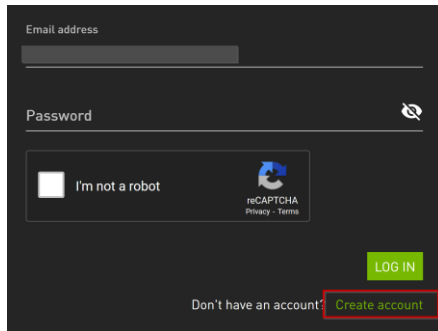
- NVIDIA CUDA ツールキット **10.1**
 - NVIDIA cuDNN **7** は
- バージョン指定** (2020年 9月時点)

最新バージョンは NG

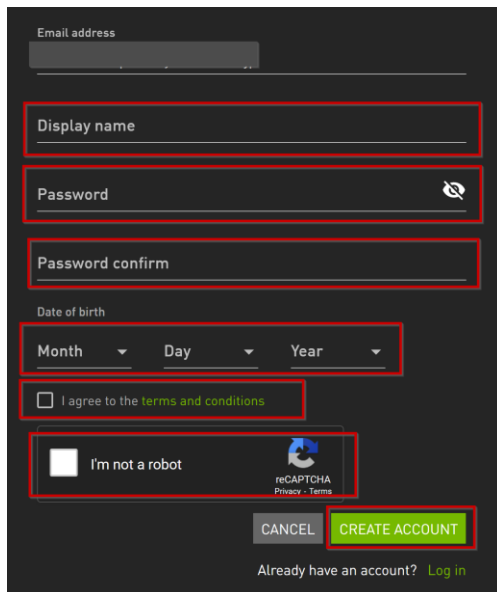
NVIDIA cuDNN のダウンロードには、登録が必要



電子メールアドレス



「Create account」
をクリック



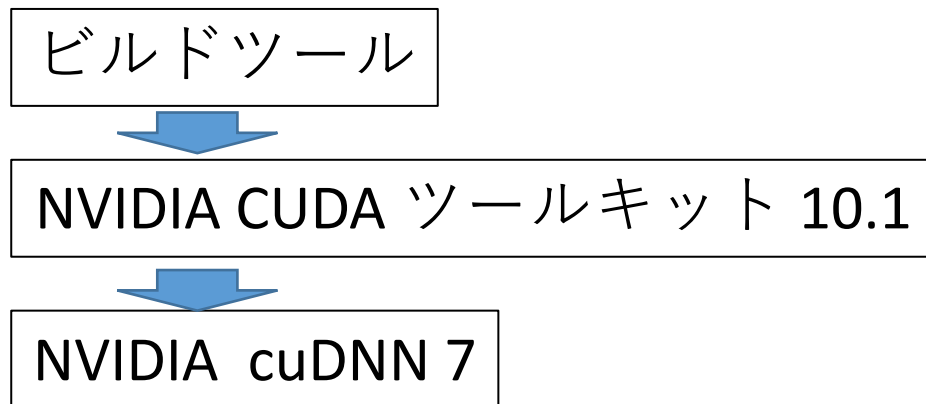
表示名 (Display name)
パスワード (password)
パスワード (password)
生年月日 (Date of Birth)
チェック 2か所

「CREATE ACCOUNT」をクリック

アンインストールが必要かも



- 次の順序でインストールする必要がある。

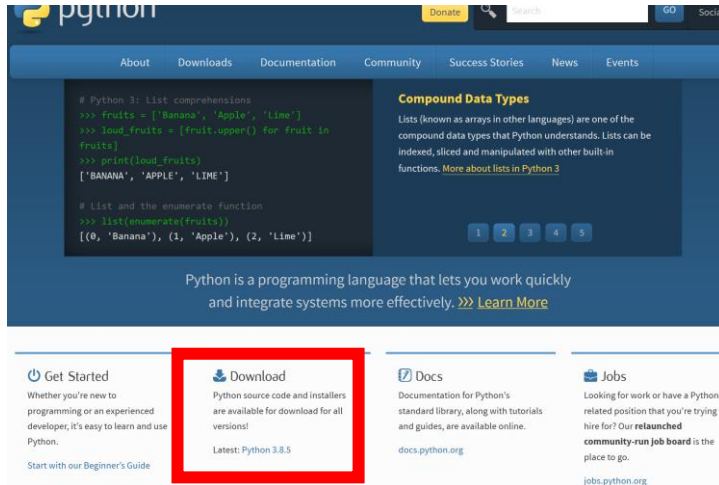


- バージョンの違うものをインストールしている場合には、アンインストールした方がよい
 - NVIDIA CUDA ツールキット **10.1**
 - NVIDIA cuDNN **7** は **バージョン指定** (2020年 9月時点)

① Python のインストール



Python の URL: <http://www.python.org/>



Download

| Version | Operating System |
|--|------------------|
| Gzipped source tarball | Source release |
| XZ compressed source tarball | Source release |
| macOS 64-bit installer | Mac OS X |
| Windows help file | Windows |
| Windows x86-64 embeddable zip file | Windows |
| Windows x86-64 executable installer | Windows |
| Windows x86-64 web-based installer | Windows |
| Windows x86 embeddable zip file | Windows |
| Windows x86 executable installer | Windows |
| Windows x86 web-based installer | Windows |

**Windows x86-64
executable installer**

① Python のインストール



1. Python 3.8 のインストーラを管理者として実行
2. 「**Install launcher for all users**」をチェック.
3. 「**Add Python 3.8 to PATH**」をチェック.
4. 「**Customize installation**」を選ぶ.
5. 次の次の画面では, 「Install for all users」をチェック.
6. そして, **Python のインストールディレクトリ**は, 「**C:¥Program Files¥Python38**」のように**自動設定**されることを確認.
7. 「Disable path length limit」をチェック.
8. インストールのあと、**Windows のスタートメニュー**に「Python 3.8」が**増えている**ことを確認.

詳しくは

<https://www.kkaneko.jp/tools/win/python.html>

② pip と setuptools の更新, Python 開発環境 (JupyterLab, spyder) のインストール

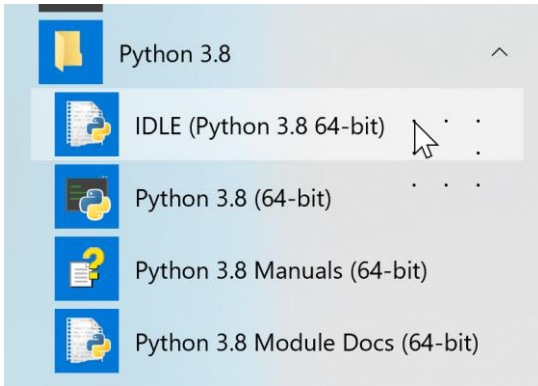


コマンドプロンプトを管理者として実行し, 次のコマンドを実行

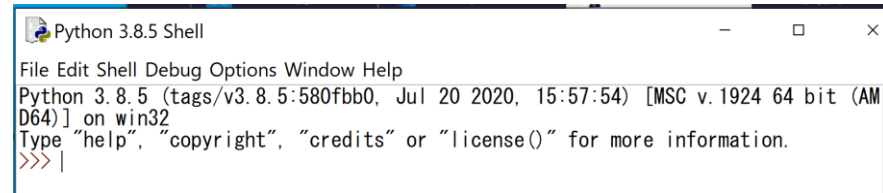
```
python -m pip install -U pip
```

```
python -m pip install -U jupyterlab jupyter jupyter-console jupyter-text spyder
```

Python の IDLE



起動

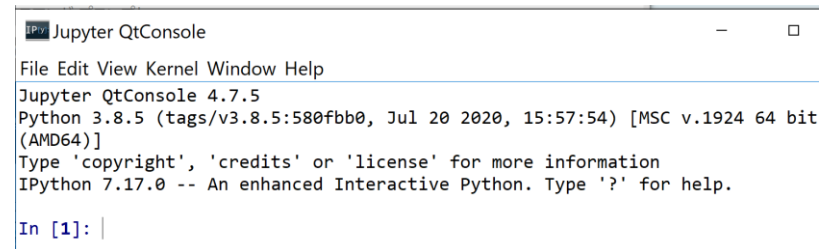


画面

jupyter qtconsole



起動



画面

③ ビルドツールのインストール



「Visual C++ ビルドツール」, 「Build Tools for Visual Studio 2019」と呼んだりもする.

URL:

<https://visualstudio.microsoft.com/ja/downloads/>

▼ Visual Studio 2019 のツール

Build Tools for Visual
Studio 2019

これらのビルド ツールを使用すると、コマンドライン インターフェイスから Visual Studio をビルドできます。 サポートされるプロジェクト: ASP.NET、Azure、C++ デスクトップ、ClickOnce、コンテナ、.NET Core、.NET Desktop、Node.js、Office と SharePoint、Python、TypeScript、単体テスト、UWP、WCF、Xamarin。

ダウンロード ↓

詳しくは

<https://www.kkaneko.jp/tools/win/buildtool.html>

④ NVIDIA グラフィックスボードドライバのインストール



NVIDIA ドライバの URL: <https://www.nvidia.co.jp/drivers>

NVIDIAドライバダウンロード

オプション1: エヌビディア製品用ドライバを手動検索する

製品のタイプ:

製品シリーズ:

製品ファミリー:

オペレーティングシステム:

ダウンロードタイプ:

言語:

検索

詳しくは

<https://www.kkaneko.jp/tools/win/nvidiadriver.html>

⑤ NVIDIA CUDA ツールキットバージョン 10.1 のインストール



NVIDIA CUDA ツールキットの URL:

<https://developer.nvidia.com/cuda-toolkit-archive>

Latest Release

CUDA Toolkit 11.0 Update1 (Aug 2020), [Versioned Online Documentation](#)

Archived Releases

CUDA Toolkit 11.0 (May 2020), [Versioned Online Documentation](#)

CUDA Toolkit 10.2 (Nov 2019), [Versioned Online Documentation](#)

CUDA Toolkit 10.1 update2 (Aug 2019), [Versioned Online Documentation](#)

CUDA Toolkit 10.1 update1 (May 2019), [Versioned Online Documentation](#)

CUDA Toolkit 10.1 (Feb 2019), [Online Documentation](#)

CUDA Toolkit 10.0 (Sept 2018), [Online Documentation](#)

CUDA Toolkit 9.2 (May 2018), [Online Documentation](#)

**CUDA 10.1
の最新版**

詳しくは

<https://www.kkaneko.jp/tools/win/cuda.html>

⑥ NVIDIA cuDNN バージョン 7 のインストール



NVIDIA cuDNN のダウンロード URL:

<https://developer.nvidia.com/cudnn>

NVIDIA cuDNN is a GPU-accelerated library of primitives for deep

Download cuDNN v8.0.3 (August 26th, 2020), for CUDA 11.0

Download cuDNN v8.0.3 (August 26th, 2020), for CUDA 10.2

Download cuDNN v8.0.3 (August 26th, 2020), for CUDA 10.1

Archived cuDNN Releases

Download cuDNN v8.0.2 (July 24th, 2020), for CUDA 11.0

Download cuDNN v8.0.2 (July 24th, 2020), for CUDA 10.2

Download cuDNN v8.0.2 (July 24th, 2020), for CUDA 10.1

Download cuDNN v8.0.1 RC2 (June 26th, 2020), for CUDA 11.0

Download cuDNN v8.0.1 RC2 (June 26th, 2020), for CUDA 10.2

Download cuDNN v7.6.5 (November 18th, 2019), for CUDA 10.2

Download cuDNN v7.6.5 (November 5th, 2019), for CUDA 10.1

Download cuDNN v7.6.5 (November 5th, 2019), for CUDA 10.0

Archived cuDNN Releases

CUDA 10.1 用の cuDNN バージョン 7 の最新版

- NVIDIA cuDNN 7 のダウンロードには、**登録が必要**
- システム環境変数 **CUDNN_PATH** と **PATH** の設定が必要

詳しくは

<https://www.kkaneko.jp/tools/win/cudnn.html>

⑦ TensorFlow, Keras の旧バージョンのアンインストール



コマンドプロンプトを管理者として実行し, 次のコマンドを実行

```
python -m pip uninstall -y tensorflow tensorflow-cpu tensorflow-gpu tensorflow_datasets tensorflow-hub keras
```

詳しくは

<https://www.kkaneko.jp/tools/win/keras.html>

⑧ TensorFlow, Keras のインストール



コマンドプロンプトを管理者として実行し, 次のコマンドを実行

```
python -m pip install -U tensorflow-gpu
tensorflow_datasets tensorflow-hub keras matplotlib

python -m pip install -q
git+https://github.com/tensorflow/docs

python -m pip install -q
git+https://github.com/tensorflow/examples.git
```

関連ソフトもインストールしている

詳しくは

<https://www.kkaneko.jp/tools/win/keras.html>

⑨ GPU が認識できているかの確認

コマンドプロンプトを管理者として実行し、次のコマンドを実行

```
python -c "from tensorflow.python.client import device_lib; print(device_lib.list_local_devices())"
```

```
0:~\heroku> python -c "from tensorflow.python.client import device_lib; print(device_lib.list_local_devices())"
2020-09-20 21:55:25.373217: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library cudart64_101.dll
2020-09-20 21:55:25.947922: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX2
To enable these instructions, TensorFlow was built with the appropriate compiler flags.
2020-09-20 21:55:25.987930: I tensorflow/compiler/xla/service/service.cc:180] XLA service 0x1de90638850 initialized for platform Host (this does not guarantee that XLA will be used). Devices:
2020-09-20 21:55:25.987930: I tensorflow/compiler/xla/service/service.cc:179] StreamExecutor device (0): Host, default version
2020-09-20 21:55:25.972875: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library nvcuda.dll
2020-09-20 21:55:25.918819: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1170] Found device 0 with properties:
pciBusId: 0000:01:00.0 name: GeForce GTX 970 computeCapability: 5.2
coreClock: 1.175GHz coreCount: 13 deviceMemory: 4.0001675GiB memoryLocation: PCI bus: 0000:01:00.0
2020-09-20 21:55:25.918759: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library cudart64_101.dll
2020-09-20 21:55:25.918922: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library cublas64_10.dll
2020-09-20 21:55:25.918939: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library curand64_10.dll
2020-09-20 21:55:25.918739: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library cusolver64_10.dll
2020-09-20 21:55:25.918939: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library cusparse64_10.dll
2020-09-20 21:55:25.924244: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library cudnn64_7.dll
2020-09-20 21:55:25.923939: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1853] Adding visible gpu devices: 0
2020-09-20 21:55:25.931172: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1821] Device interconnect StreamExecutor with strength 1 edge matrix:
2020-09-20 21:55:25.931402: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1823] 0
2020-09-20 21:55:25.932819: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1378] 0: M
2020-09-20 21:55:25.935414: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1402] Created TensorFlow device (/device:GPU:0 with 2995 MB memory) -> physical GPU (device: 0, name: GeForce GTX 970, pci bus id: 0000:01:00.0, compute capability: 5.2)
2020-09-20 21:55:25.935414: I tensorflow/compiler/xla/service/service.cc:180] XLA service 0x1de9af6f60 initialized for platform CUDA (this does not guarantee that XLA will be used). Devices:
2020-09-20 21:55:25.935414: I tensorflow/compiler/xla/service/service.cc:179] StreamExecutor device (0): GeForce GTX 970, compute Capability: 5.2
{
  "device": "/device:GPU:0",
  "memory_limit": 3130739918,
  "locality": {
    "bus_id": 1,
    "links": {}
  }
}
{
  "device": "/device:XLA_CPU:0",
  "memory_limit": 268435456,
  "locality": {}
}
{
  "incarnation": 148525784010917106,
  "name": "/device:XLA_CPU:0",
  "device_type": "XLA_CPU",
  "memory_limit": 17778669184,
  "locality": {}
}
{
  "incarnation": 16393820837522826676,
  "physical_device_desc": "device: XLA_CPU device",
  "name": "/device:GPU:0",
  "device_type": "GPU",
  "memory_limit": 3130739918,
  "locality": {
    "bus_id": 1,
    "links": {}
  }
}
{
  "incarnation": 1553489151792931868,
  "physical_device_desc": "device: 0, name: GeForce GTX 970, pci bus id: 0000:01:00.0, compute capability: 5.2",
  "name": "/device:XLA_CPU:0",
  "device_type": "XLA_CPU",
  "memory_limit": 17778669184,
  "locality": {}
}
{
  "incarnation": 15418523888694444031,
  "physical_device_desc": "device: XLA_GPU device",
  "name": "/device:XLA_GPU:0",
  "device_type": "XLA_GPU",
  "memory_limit": 17778669184,
  "locality": {}
}
```

```
incarnation: 16393820837522826676
physical_device_desc: "device: XLA_CPU device"
name: "/device:GPU:0"
device_type: "GPU"
memory_limit: 3130739918
locality {
  bus_id: 1
  links {
  }
}
```

詳しくは

<https://www.kkareko.jp/tools/win/keras.html>

⑩ TensorFlow を扱う Python プログラム



```
import tensorflow as tf
matrix1 = tf.constant([[3., 3.]])
matrix2 = tf.constant([[2.],[2.]])
print( tf.matmul(matrix1, matrix2) )
```

結果として 「**[[12.]]**」 のように表示される

詳しくは

<https://www.kkaneko.jp/tools/win/keras.html>