

파이썬으로 배우는 AI 로봇

핑퐁 파이썬 활용 1

김정래 (Solrootec)

주요 내용

- ▶ 핑퐁 가상환경 설정
- ▶ Example1: chapter 1
- ▶ Example2: chapter 3
- ▶ Example3: chapter 7

● 아나콘다 - 의존성 패키지 설치

준비 2

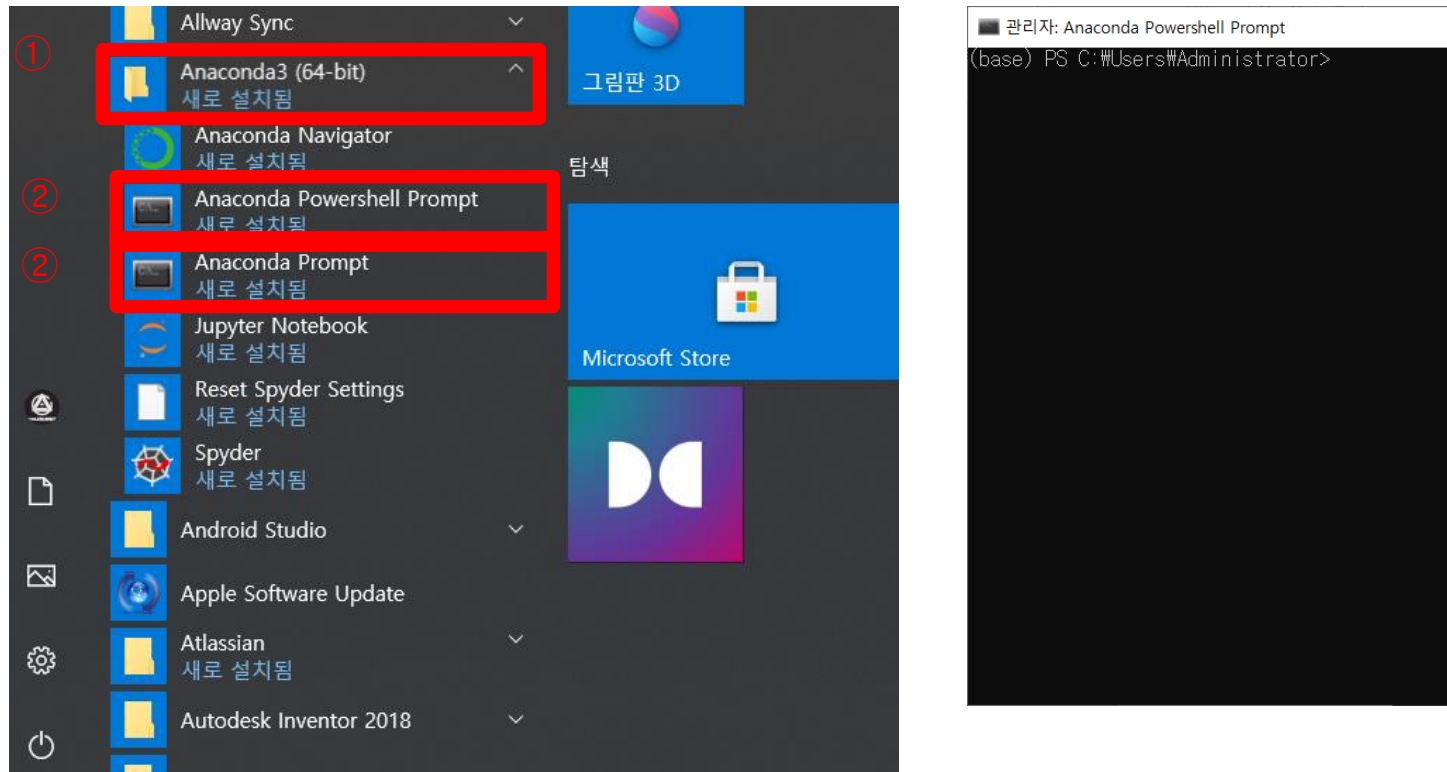
- ▶ 프로젝트를 실행하기 위해, pingpongthread의 파이썬 의존성 패키지를 설치해야 한다.
- ▶ pyserial==3.4
- ▶ Pillow==7.0.0
- ▶ psutil==5.7.3
- ▶ keyboard==0.13.5
- ▶ opencv-python==4.4.0.46
- ▶ matplotlib==3.1.1
- ▶ playsound==1.2.2
- ▶ gtts==2.2.1
- ▶ SpeechRecognition==3.8.1
- ▶ PyAudio==0.2.11
- ▶ tensorflow==2.3.1 (numpy 1.18.5)

● 아나콘다 - 의존성 패키지 설치

준비 2

2-1.

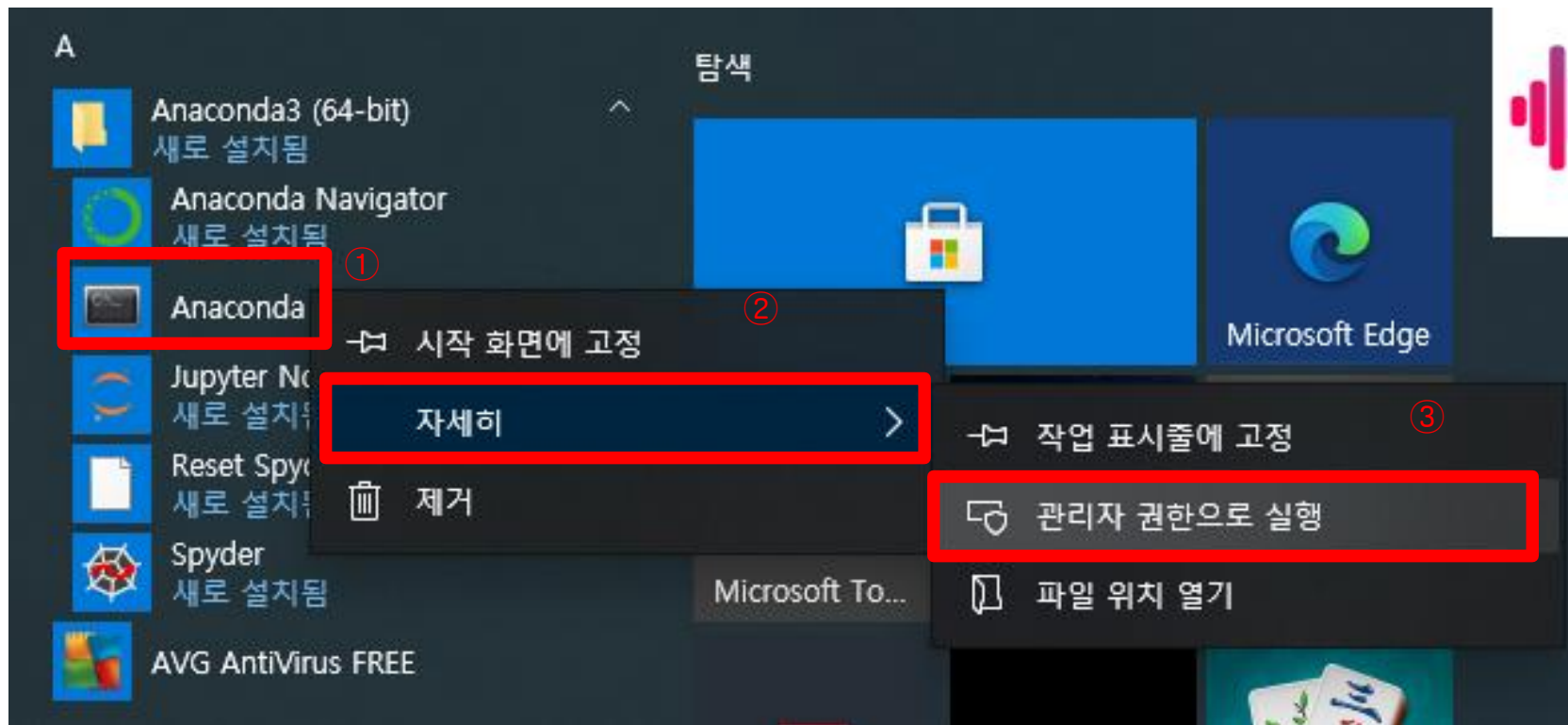
- ▶ 시작 프로그램에서 'Ananconda3' 폴더의 'Anaconda Prompt', 또는 'Anaconda Powershell Prompt'를 관리자 권한으로 열어 아나콘다 전용 셸을 실행한다.



● 아나콘다 - 의존성 패키지 설치

준비 2

- ▶ 관리자 권한으로 실행하는 방법은, 오른쪽 클릭 - '자세히' - '관리자 권한으로 실행'에 있다.



● 아나콘다 - 의존성 패키지 설치

준비 2

2-2.

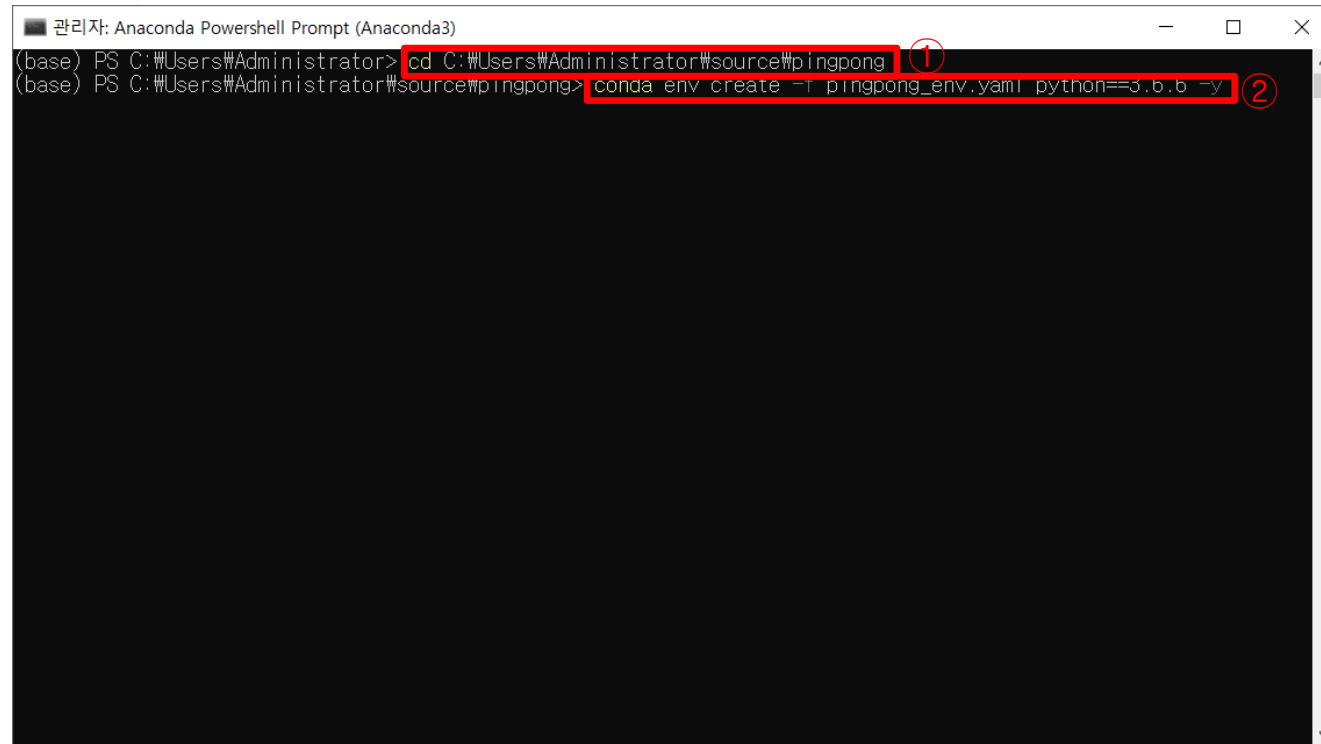
- ▶ 만약 아나콘다 2020.11 버전을 사용한다면, 가상환경 설정을 불러오면 의존성 패키지를 일일이 설치하지 않고 한꺼번에 설치할 수 있다.
 - ▶ 다른 아나콘다 버전의 가상환경 설정 설치 대한 작동 여부는 확인되지 않았다.
 - ▶ 가상환경 설정 파일의 확장자는 '.yaml'이다.
- ▶ 다음 코드를 입력하면 파이썬 3.6.6 버전의 pingpong 가상환경을 불러올 수 있다.

```
conda env create -f {pingpong_env.yaml} python==3.6.6
```

- ▶ {pingpong_env.yaml}는 이 파일의 위치를 찾아 입력해주면 된다. 아니면 이 파일이 있는 위치로 이동하여 'pingpong_env.yaml'라고 그대로 써주면 된다. 이 파일은 pingpong 라이브러리 최상위 폴더에 있다.

● 아나콘다 - 의존성 패키지 설치

준비 2



```
관리자: Anaconda PowerShell Prompt (Anaconda3)
(base) PS C:\Users\Administrator> cd C:\Users\Administrator\source\pingpong
(base) PS C:\Users\Administrator\source\pingpong> conda env create -f pingpong_env.yaml python==3.6.6 -y
```

● 아나콘다 - 의존성 패키지 설치

준비 2

- ▶ "conda activate pingpong"라는 명령어를 입력하면 설치한 pingpong 가상환경을 활성화시킬 수 있다.
- ▶ 가상환경이 활성화 되었으면 가장 왼쪽에 (pingpong)이라는 문자가 뜰 것이다.
- ▶ 셸 프롬프트를 꺾다 다시 켜면 기본 환경인 (base)으로 이동할 것이다. 따라서 셸 프롬프트를 켤 때마다 "conda activate pingpong"를 실행해 주어야 한다.
- ▶ 만약 셸 프롬프트를 끄지 않고 가상환경에서 나가고 싶으면, "deactivate"라고 입력하면 된다.

● 아나콘다 - 의존성 패키지 설치

준비 2

2-3.

- ▶ 2-2의 방법으로 예제가 잘 실행되지 않는다면, 가상환경을 직접 만들고 패키지를 모두 설치해야 한다.
- ▶ 가상환경의 이름은 pingpong으로 정할 것이다. 쉘 프롬프트에 다음 코드를 순차적으로 입력한다.
 - 1) `conda create -n pingpong python==3.6.6 -y`
 - 2) `conda activate pingpong`
- ▶ 가상환경이 활성화 되었으면 가장 왼쪽에 (pingpong)이라는 문자가 뜰 것이다.
- ▶ 쉘 프롬프트를 꺾다 다시 켜면 기본 환경인 (base)으로 이동할 것이다. 따라서 쉘 프롬프트를 켤 때마다 "conda activate pingpong"를 실행해 주어야 한다.
- ▶ 만약 쉘 프롬프트를 끄지 않고 가상환경에서 나가고 싶으면, "deactivate"라고 입력하면 된다.

● 아나콘다 - 의존성 패키지 설치

준비 2

- ▶ 쉘에 다음페이지의 명령어들을 순차적으로 입력한다.
(pingpong) 환경에 다음 패키지들을 설치할 것이다.
- ▶ 현재 설치된 패키지들을 확인하고 싶으면, "pip freeze"라고 입력하면 된다.

● 아나콘다 - 의존성 패키지 설치

준비 2

- 1) `conda install jupyter -y`
- 2) `pip install msgpack`
- 3) `pip install pyserial==3.4`
- 4) `pip install Pillow==7.0.0`
- 5) `pip install psutil==5.7.3`
- 6) `pip install keyboard==0.13.5`
- 7) `pip install opencv-python==4.4.0.46`
- 8) `pip install matplotlib==3.1.1`
- 9) `pip install playsound==1.2.2`
- 10) `pip install gtts==2.2.1`
- 11) `pip install SpeechRecognition==3.8.1`
- 12) `pip install PyAudio==0.2.11`
- 13) `pip install tensorflow==2.3.1`

● 아나콘다 - 의존성 패키지 설치

준비 2

2-3.

- ▶ 텐서플로를 실행할 수 있게 만들기 위해, "Microsoft Visual C++ 2015-2019 Redistributable" 패키지를 설치해야 한다. 다음 사이트에 들어가 자신의 컴퓨터(32비트, 64비트)에 맞는 버전을 다운로드 받아 설치하라.

<https://support.microsoft.com/ko-kr/help/2977003/the-latest-supported-visual-c-downloads>

Visual Studio 2015, 2017 및 2019

Visual Studio 2015, 2017 및 2019용 Microsoft Visual C++ 재배포 가능 패키지를 다운로드합니다. 다음 업데이트는 Visual Studio 2015, 2017 및 2019용으로 지원되는 최신 Visual C++ 재배포 가능 패키지입니다. Universal C Runtime의 기본 버전이 포함되어 있습니다. 자세한 내용은 MSDN을 참조하세요.

- x86: [vc_redist.x86.exe](#)
- x64: [vc_redist.x64.exe](#)
- ARM64: [vc_redist.arm64.exe](#)

참고 Visual C++ 2015, 2017 및 2019 모두 동일한 재배포 가능 파일을 공유합니다.

| | | |
|--------------|--|-------------|
| configure.py | Upgrading Bazel to 4.2.1 | 14 days ago |
| models.BUILD | Make models.BUILD filegroup include everything but metadata files a... | 5 years ago |

☰ README.md



python 3.7 | 3.8 | 3.9 pypi package 2.7.0 DOI 10.5281/zenodo.4724125

Documentation

api reference

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of [tools](#), [libraries](#), and [community](#) resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML-powered applications.

TensorFlow was originally developed by researchers and engineers working on the Google Brain team within Google's Machine Intelligence Research organization to conduct machine learning and deep neural networks research. The system is general enough to be applicable in a wide variety of other domains, as well.

3,029 contributors

Languages



● 주피터 노트북 실행

준비 2

3-1.

- ▶ 셸 프롬프트의 (pingpong) 환경에서 주피터 노트북을 실행한다. 다음 명령어를 입력하면 된다.

```
cd {예제_폴더_위치}
```

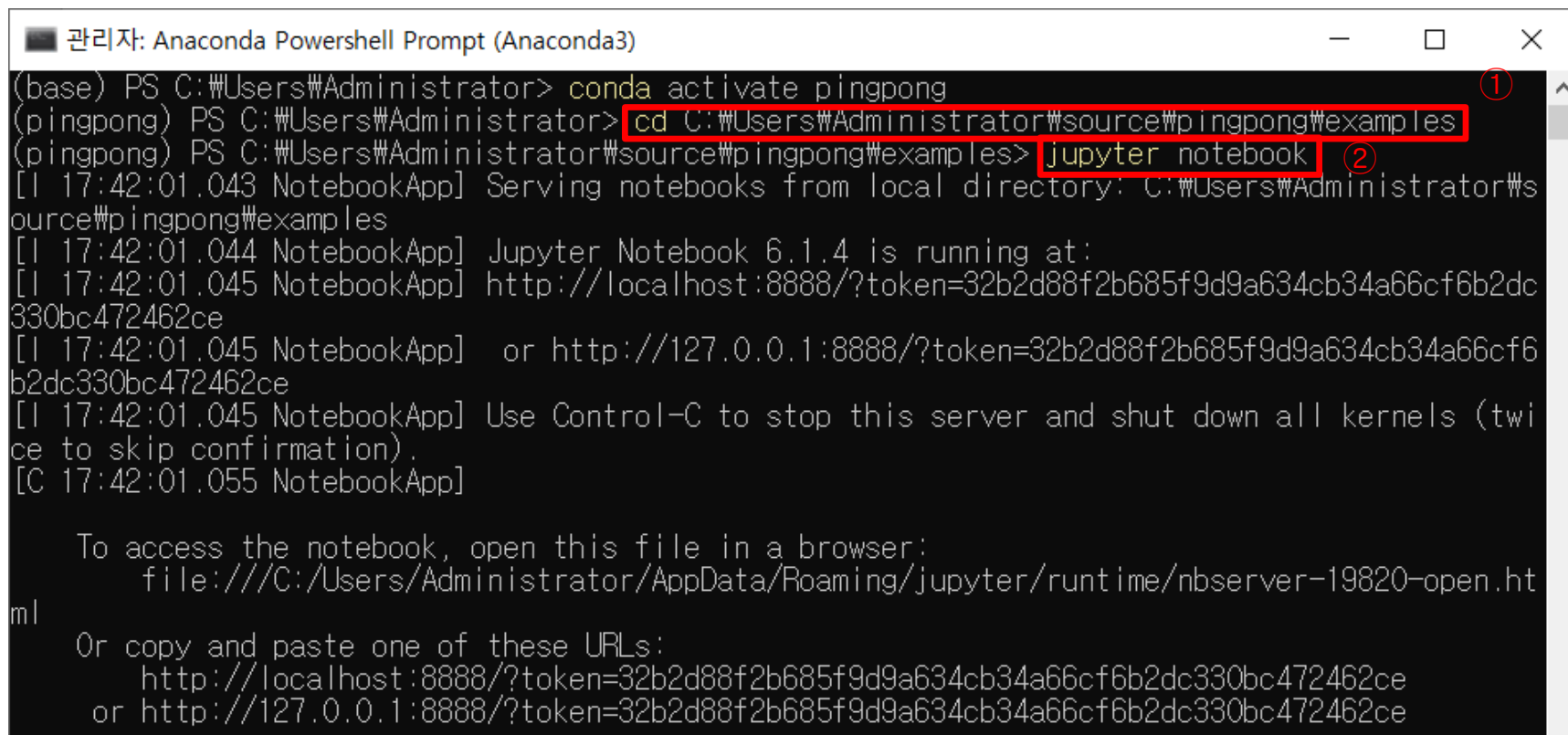
```
jupyter notebook
```

- ▶ 만약 pingpongthread 라이브러리의 예제 위치가 'C:\Users\Administrator\source\pingpong\examples'에 있으면,
"cd C:\Users\Administrator\source\pingpong\examples" 라고 입력 하면 된다.

● 주피터 노트북 실행

준비 2

* 주의: 주피터 노트북이 켜져 있는 동안, 이 프롬프트는 끄면 안 된다.



```
관리자: Anaconda Powershell Prompt (Anaconda3)
(base) PS C:\Users\Administrator> conda activate pingpong
(pingpong) PS C:\Users\Administrator> cd C:\Users\Administrator\source\pingpong\examples
(pingpong) PS C:\Users\Administrator\source\pingpong\examples> jupyter notebook
[I 17:42:01.043 NotebookApp] Serving notebooks from local directory: C:\Users\Administrator\source\pingpong\examples
[I 17:42:01.044 NotebookApp] Jupyter Notebook 6.1.4 is running at:
[I 17:42:01.045 NotebookApp] http://localhost:8888/?token=32b2d88f2b685f9d9a634cb34a66cf6b2dc330bc472462ce
[I 17:42:01.045 NotebookApp] or http://127.0.0.1:8888/?token=32b2d88f2b685f9d9a634cb34a66cf6b2dc330bc472462ce
[I 17:42:01.045 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 17:42:01.055 NotebookApp]

To access the notebook, open this file in a browser:
    file:///C:/Users/Administrator/AppData/Roaming/jupyter/runtime/nbserver-19820-open.html
Or copy and paste one of these URLs:
    http://localhost:8888/?token=32b2d88f2b685f9d9a634cb34a66cf6b2dc330bc472462ce
    or http://127.0.0.1:8888/?token=32b2d88f2b685f9d9a634cb34a66cf6b2dc330bc472462ce
```

● 주피터 노트북 실행

준비 2

* 주의: 일반 cmd 프롬프트에서, 다른 드라이브로 (현재 C 드라이브 → D 드라이브로 이동)하기 위해서는, 'D:'라고 입력한 뒤에 'cd' 명령어를 입력해 이동.

- ▶ 파워셸의 경우에는 다른 드라이브로 이동하려면 cmd처럼 해도 되고, 아니면 바로 'cd' 명령어로 이동할 수도 있다.
- ▶ 파워셸 명령어에 대한 자세한 내용은 파이썬 설치 & 사용법 - 3. 파워셸 열기 - 3-4. 파워셸 기초 참고.

```
관리자: C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.18363.1198]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Windows\System32>D:

D:\>cd D:/pingpong/examples

D:\pingpong\examples>
```

```
관리자: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

새로운 크로스 플랫폼 PowerShell 사용 https://aka.ms/pscore6

PS C:\Users\Administrator> cd D:/pingpong/examples
PS D:\pingpong\examples>
```


● 주피터 노트북 실행

준비 2

▶ 주피터 노트북을 실행하면 다음과 같이 나타난다.

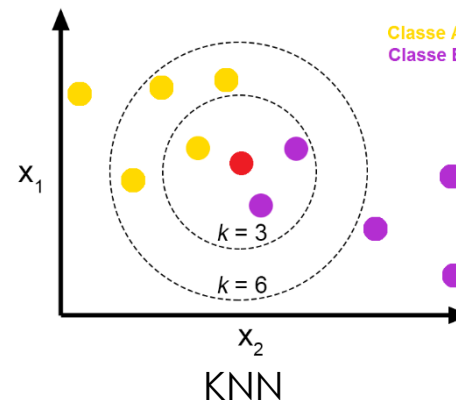
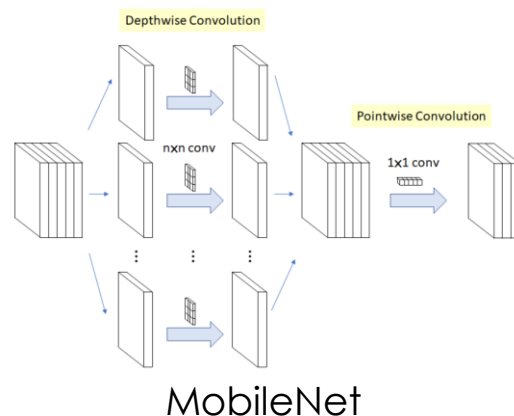
The screenshot displays the JupyterLab web interface in a browser window. The address bar shows 'localhost:8888/tree'. The interface includes a top bar with the Jupyter logo, 'Quit', and 'Logout' buttons. Below this is a tabbed interface with 'Files', 'Running', and 'Clusters' tabs. The 'Files' tab is active, showing a file browser view. At the top of the file browser, there are buttons for 'Upload', 'New', and a refresh icon. Below these is a table of files and directories. The table has columns for 'Name', 'Last Modified', and 'File size'. The files listed include directories like 'chapter5' and 'ex_dev', and files like 'ai_chapter2.ipynb', 'ai_chapter3.ipynb', 'ai_chapter5.ipynb', 'ai_chapter7.ipynb', 'ai_chapter9.ipynb', 'ai_chapter2.py', 'ai_chapter3.py', 'ai_chapter5.py', 'ai_chapter7.py', 'ai_chapter9.py', 'example_base.py', 'get_sensor_example.py', and 'ledmatrix_example.py'.

| Name | Last Modified | File size |
|-----------------------|---------------|-----------|
| 0 | | |
| / | | |
| chapter5 | 14일 전 | |
| ex_dev | 18일 전 | |
| ai_chapter2.ipynb | 18시간 전 | 7.03 kB |
| ai_chapter3.ipynb | 18시간 전 | 4.37 kB |
| ai_chapter5.ipynb | 18시간 전 | 8.06 kB |
| ai_chapter7.ipynb | 18시간 전 | 6.7 kB |
| ai_chapter9.ipynb | 18시간 전 | 8.28 kB |
| ai_chapter2.py | 8일 전 | 2.57 kB |
| ai_chapter3.py | 9일 전 | 1.54 kB |
| ai_chapter5.py | 8일 전 | 3.29 kB |
| ai_chapter7.py | 12일 전 | 1.65 kB |
| ai_chapter9.py | 9일 전 | 3.27 kB |
| example_base.py | 한 달 전 | 204 B |
| get_sensor_example.py | 18일 전 | 943 B |
| ledmatrix_example.py | 한 달 전 | 1.93 kB |

● 이미지 분류의 원리

출처: <https://observablehq.com/@nsthorat/how-to-build-a-teachable-machine-with-tensorflow-js>

- ▶ 본 프로젝트에서 사용한 이미지 분류의 원리는 ' MobileNet + 전이학습 + KNN ' 이다.
- ▶ MobileNet은 CNN(Convolutional Neural Network) 기반의 이미지 분류 신경망이다. 연산량이 빠르다는 장점이 있다.
- ▶ 전이학습(Transfer Learning)은 이미 학습시킨 신경망을 이용하여 다른 역할을 수행하도록 만드는 알고리즘을 말한다.
- ▶ KNN (K-Nearest Neighbor)은 한 점에서 distance가 가까운 순서로 k개의 점을 골라 해당하는 점의 클래스는 예측하는 알고리즘이다.



● 이미지 분류의 원리

- 1) 'ImageNet'의 데이터로 학습시킨 MobileNet 모델을 불러온다.
- 2) 레퍼런스용 이미지 n 개를 MobileNet에 통과시켜 레퍼런스 행렬을 받는다.
 - ▶ 한 개의 이미지를 MobileNet에 통과시키면 길이가 1000인 정규화된 벡터(normalized vector)로 나온다.
 - ▶ n 개의 이미지를 MobileNet에 통과시키면 벡터가 n 개 쌓여 있는 형태인 $1000 \times n$ 의 행렬로 반환된다.
- 3) 예측용 이미지를 MobileNet에 통과시켜 길이가 1000인 벡터를 받는다. 그리고 레퍼런스 행렬과 예측 벡터를 행렬 곱셈하여 길이가 n 인 유사도 벡터를 받는다.
 - ▶ 임의의 두 정규화된 벡터를 내적(inner product)시키면, 두 벡터의 코사인 유사도(cosine similarity)가 나오게 된다. 두 벡터의 각도가 비슷할 수록 유사도가 크고, 각도가 멀수록 유사도가 작다.
 - ▶ 레퍼런스 행렬과 예측 벡터를 행렬 곱셈하면, 레퍼런스 행렬의 각각의 행과 예측 벡터를 내적시킨 것과 같다. 내적시킨 결과들을 벡터로 나타낸 것이 유사도 벡터이다.
 - ▶ 유사도 벡터는 예측용 이미지와 각각의 레퍼런스용 이미지들 사이의 유사도를 리스트로 나타낸 것이다.
- 4) 거리 벡터 중 가장 큰 k 개를 고른다. 즉, 예측용 이미지와 가장 유사한 레퍼런스 이미지 k 개를 고른다. 그리고 어느 클래스에 속해 있는지 확인한다. 가장 많이 속해 있는 클래스가 예측용 이미지가 속한 클래스라고 예측한다.

● 이미지 분류의 원리

참고: KNN을 사용할 때, 거리 대신 유사도를 이용하였다. 유사도는 거리와 반대 개념으로, 두 객체의 거리는 작을수록 비슷한 것이지만, 두 객체의 유사도는 클수록 비슷한 것이다.

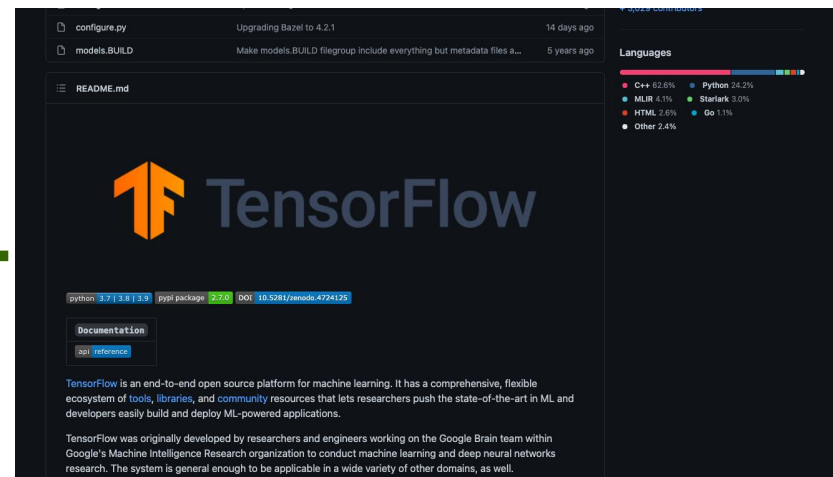
- ▶ 본 프로젝트에서 사용한 알고리즘에서는 인공신경망에서 사용하는 학습인 '가중치들의 수정'을 적용하지 않았다. 따라서 인공신경망을 학습시키지는 않는 알고리즘이며, 인공신경망을 이용한 이미지 분류라고 보는 것이 적절하다.

핑퐁 파이썬 활용

- ▶ 핑퐁 가상환경 불러오기
- ▶ Examples 실행

핑퐁 가상환경 설정

- ▶ 텐서플로를 실행할 수 있게 만들기 위해,
- ▶ "Microsoft Visual C++ 2015-2019 Redistributable" 패키지를 설치
- ▶ <https://support.microsoft.com/ko-kr/help/2977003/the-latest-supported-visual-c-downloads>



Visual Studio 2015, 2017 및 2019

Visual Studio 2015, 2017 및 2019용 Microsoft Visual C++ 재배포 가능 패키지를 다운로드합니다. 다음 업데이트는 Visual Studio 2015, 2017 및 2019용으로 지원되는 최신 Visual C++ 재배포 가능 패키지입니다. Universal C Runtime의 기본 버전이 포함되어 있습니다. 자세한 내용은 MSDN을 참조하세요.

- x86: [vc_redist.x86.exe](#)
- x64: [vc_redist.x64.exe](#)
- ARM64: [vc_redist.arm64.exe](#)

참고 Visual C++ 2015, 2017 및 2019 모두 동일한 재배포 가능 파일을 공유합니다.

핑퐁 가상환경 설정

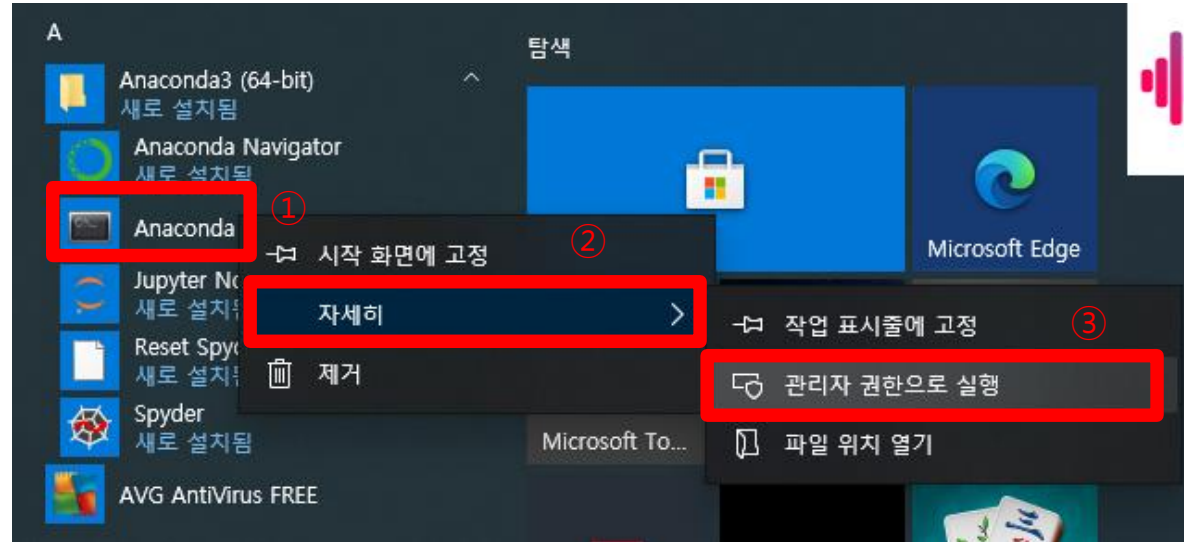
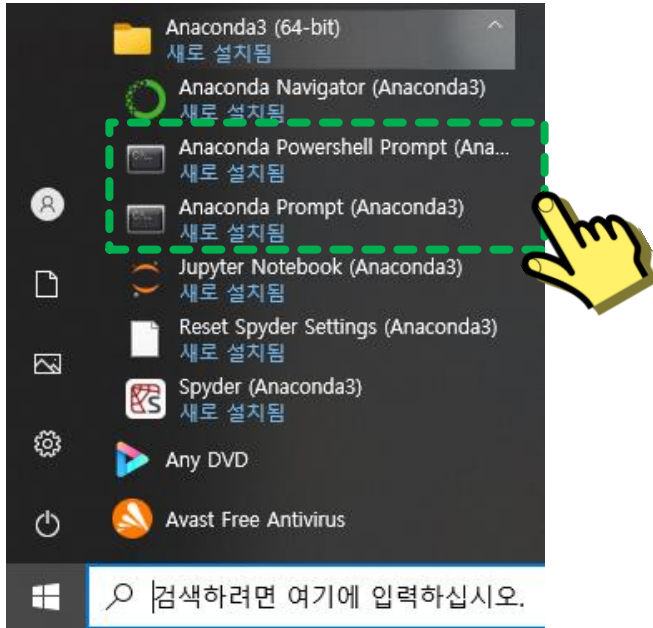
압축해제하고 D:에 pingpong_test_yaml 폴더를 생성하고 저장
(pingpong_env.yaml 파일이 있는 폴더)

| 6_202112_한양_파이썬으로_배우... > 3_2_핑퐁_파이썬_활용 | | |
|---|--------------------|------------|
| 이름 | 수정한 날짜 | 유형 |
| pingpong-master | 2021-12-21 오후 4:27 | 파일 폴더 |
| pingpong-master 9.9.zip | 2021-12-16 오후 9:59 | 압축(ZIP) 파일 |

압축풀기

| 내 PC > Solrootec (D:) > pingpong_test_yaml | | | |
|--|--------------------|---------------------|------|
| 이름 | 수정한 날짜 | 유형 | 크기 |
| .git | 2021-12-16 오전 1:11 | 파일 폴더 | |
| .vscode | 2021-12-16 오전 1:11 | 파일 폴더 | |
| __pycache__ | 2021-12-16 오전 1:12 | 파일 폴더 | |
| ai | 2021-12-16 오전 1:11 | 파일 폴더 | |
| connection | 2021-12-16 오전 1:11 | 파일 폴더 | |
| dev | 2021-12-16 오전 1:11 | 파일 폴더 | |
| examples | 2021-12-16 오전 1:30 | 파일 폴더 | |
| operations | 2021-12-16 오전 1:11 | 파일 폴더 | |
| protocols | 2021-12-16 오전 1:11 | 파일 폴더 | |
| 파이썬_핑퐁_교육자료 | 2021-12-16 오전 1:11 | 파일 폴더 | |
| check_dependencies.py | 2021-07-09 오후 9:31 | JetBrains PyChar... | 3KB |
| dependency.txt | 2021-07-09 오후 9:39 | 텍스트 문서 | 1KB |
| LICENSE.txt | 2021-07-09 오후 9:39 | 텍스트 문서 | 2KB |
| main.py | 2021-07-09 오후 9:39 | JetBrains PyChar... | 1KB |
| note.txt | 2021-07-09 오후 9:39 | 텍스트 문서 | 10KB |
| pingpong_env.yaml | 2021-07-09 오후 9:39 | YAML 파일 | 4KB |
| pingpong_test_env.yaml | 2021-07-31 오전 7:51 | YAML 파일 | 4KB |
| pingpongthread.py | 2021-08-15 오전 8:39 | JetBrains PyChar... | 10KB |
| README.md | 2021-07-09 오후 9:39 | MD 파일 | 1KB |
| robotstatus.py | 2021-07-09 오후 9:39 | JetBrains PyChar... | 3KB |

핑퐁 가상환경 설정



핑퐁 가상환경 설정

pingpong 가상환경 생성

```
(base) PS D:\> cd .\pingpong_test_yaml\
(base) PS D:\pingpong_test_yaml> dir
```

디렉터리: D:\pingpong_test_yaml

| Mode | LastWriteTime | Length | Name |
|--------|--------------------|--------|------------------------|
| d---- | 2021-12-16 오후 9:59 | | .git |
| d---- | 2021-12-16 오후 9:59 | | .vscode |
| d---- | 2021-12-16 오후 9:59 | | ai |
| d---- | 2021-12-16 오후 9:59 | | connection |
| d---- | 2021-12-16 오후 9:59 | | dev |
| d---- | 2021-12-16 오후 9:59 | | examples |
| d---- | 2021-12-16 오후 9:59 | | operations |
| d---- | 2021-12-16 오후 9:59 | | protocols |
| d---- | 2021-12-16 오후 9:59 | | __pycache__ |
| d---- | 2021-12-16 오후 9:59 | | 파이썬_핑퐁_교육자료 |
| -a---- | 2021-12-16 오후 9:59 | 2123 | check_dependencies.py |
| -a---- | 2021-12-16 오후 9:59 | 328 | dependency.txt |
| -a---- | 2021-12-16 오후 9:59 | 1343 | LICENSE.txt |
| -a---- | 2021-12-16 오후 9:59 | 95 | main.py |
| -a---- | 2021-12-16 오후 9:59 | 9348 | note.txt |
| -a---- | 2021-12-16 오후 9:59 | 9750 | pingpongthread.py |
| -a---- | 2021-12-16 오후 9:59 | 3889 | pingpong_env.yaml |
| -a---- | 2021-12-16 오후 9:59 | 3894 | pingpong_test_env.yaml |
| -a---- | 2021-12-16 오후 9:59 | 10 | README.md |
| -a---- | 2021-12-16 오후 9:59 | 3055 | robotstatus.py |

```
(base) PS D:\pingpong_test_yaml> conda env create -f pingpong_env.yaml python==3.6.6
```

(base) > pingpong_env.yaml 파일이 있는 폴더로 이동

(base) > conda env create -f pingpong_env.yaml python==3.6.6

핑퐁 가상환경 설정

pingpong 가상환경 실행

```
Anaconda Powershell Prompt (Anaconda3)

)) (2.0.0)
Requirement already satisfied, skipping upgrade: python-dateutil>=2.1 in c:\users\solro\anaconda\envs\pingpong\lib\site-packages (from matplotlib==3.1.1->r D:\pingpong_test_yaml\condaenv.9h6cdt5l.requirements.txt (line 21)) (2.8.1)
Requirement already satisfied, skipping upgrade: pyparsing!=2.0.4,!2.1.2,!2.1.6,>=2.0.1 in c:\users\solro\anaconda\envs\pingpong\lib\site-packages (from matplotlib==3.1.1->r D:\pingpong_test_yaml\condaenv.9h6cdt5l.requirements.txt (line 21)) (2.4.7)
Requirement already satisfied, skipping upgrade: certifi>=2017.4.17 in c:\users\solro\anaconda\envs\pingpong\lib\site-packages (from requests==2.25.0->r D:\pingpong_test_yaml\condaenv.9h6cdt5l.requirements.txt (line 35)) (2020.11.8)
Collecting wrapt>=1.11.1
  Using cached wrapt-1.13.3-cp36-cp36m-win_amd64.whl (34 kB)
Requirement already satisfied, skipping upgrade: zipp>=0.5 in c:\users\solro\anaconda\envs\pingpong\lib\site-packages (from importlib-metadata; python_version < "3.8"->markdown==3.3.3->r D:\pingpong_test_yaml\condaenv.9h6cdt5l.requirements.txt (line 20)) (3.4.0)
Installing collected packages: absl-py, astunparse, soupsieve, beautifulsoup4, cachetools, chardet, click, cyclo, gast, pyasn1, pyasn1-modules, rsa, google-auth, urllib3, idna, requests, oauthlib, requests-oauthlib, google-pasta, grpcio, gtts, gtts-token, numpy, h5py, keras-preprocessing, keyboard, kiwisolver, markdown, matplotlib, msgpack, opencv-python, opt-einsum, pillow, playsound, protobuf, psutil, pyaudio, pyserial, speechrecognition, tensorboard-plugin-wit, werkzeug, tensorboard, termcolor, wrapt, tensorflow-estimator, tensorflow
Successfully installed absl-py-0.11.0 astunparse-1.6.3 beautifulsoup4-4.9.3 cachetools-4.1.1 chardet-3.0.4 click-7.1.2 cyclo-0.10.0 gast-0.3.3 google-auth-1.23.0 google-auth-oauthlib-0.4.2 google-pasta-0.2.0 grpcio-1.33.2 gtts-2.2.1 gtts-token-1.1.4 h5py-2.10.0 idna-2.10 keras-preprocessing-1.1.2 keyboard-0.13.5 kiwisolver-1.3.1 markdown-3.3.3 matplotlib-3.1.1 msgpack-1.0.0 numpy-1.18.5 oauthlib-3.1.0 opencv-python-4.4.0.46 opt-einsum-3.3.0 pillow-7.0.0 playsound-1.2.2 protobuf-3.14.0 psutil-5.7.3 pyasn1-0.4.8 pyasn1-modules-0.2.8 pyaudio-0.2.11 pyserial-3.4 requests-2.25.0 requests-oauthlib-1.3.0 rsa-4.6 soupsieve-2.0.1 speechrecognition-3.8.1 tensorboard-2.4.0 tensorboard-plugin-wit-1.7.0 tensorflow-2.3.1 tensorflow-estimator-2.3.0 termcolor-1.1.0 urllib3-1.26.2 werkzeug-1.0.1 wrapt-1.13.3

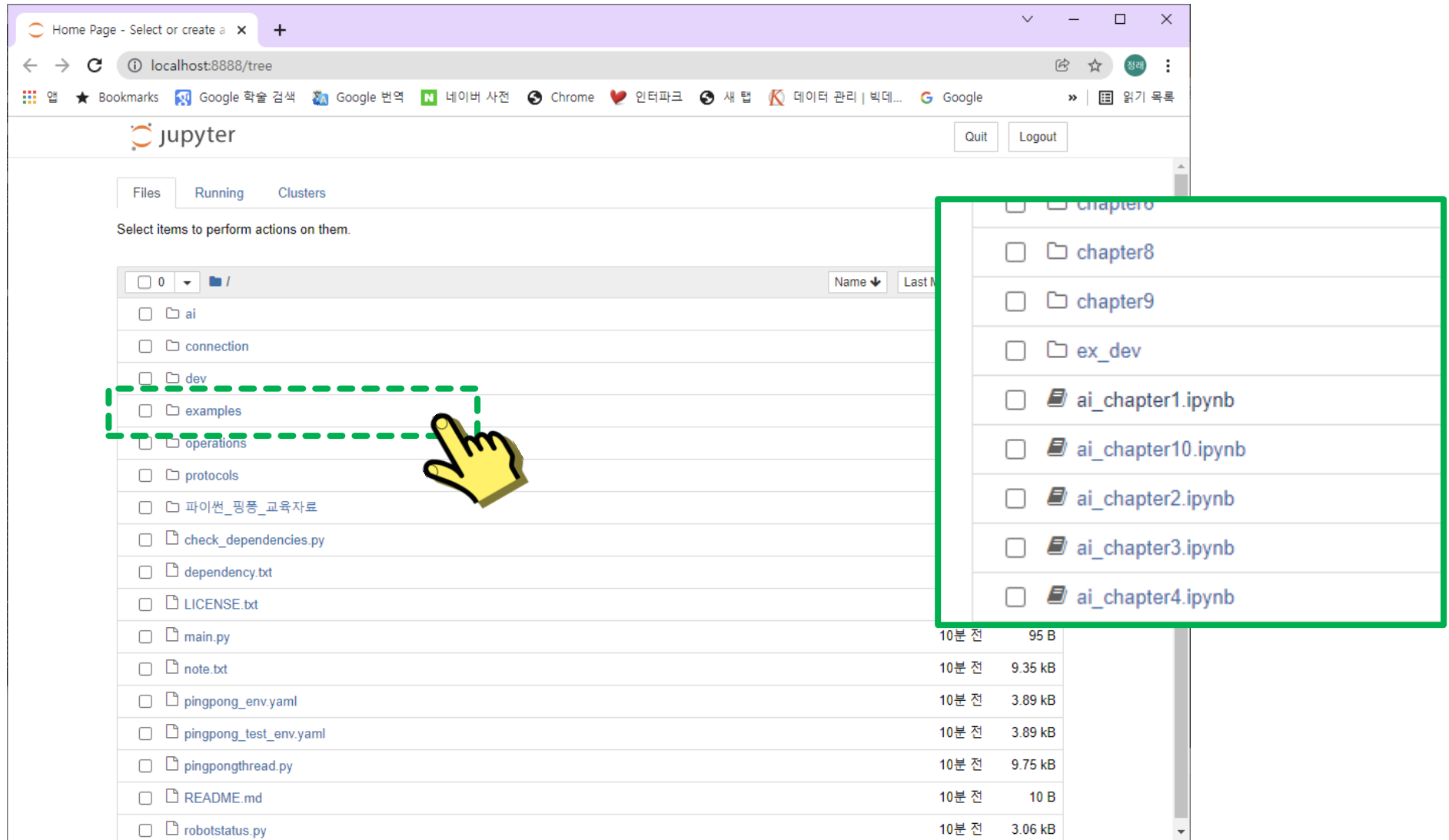
done
#
# To activate this environment, use
#
#     $ conda activate pingpong
#
# To deactivate an active environment, use
#
#     $ conda deactivate

(base) PS D:\pingpong_test_yaml>
(base) PS D:\pingpong_test_yaml> conda activate pingpong
(pingpong) PS D:\pingpong_test_yaml>
(pingpong) PS D:\pingpong_test_yaml> jupyter notebook
```

(base) > conda activate pingpong
(pingpong) > jupyter notebook

핑퐁 가상환경 설정

Jupyter notebook 실행

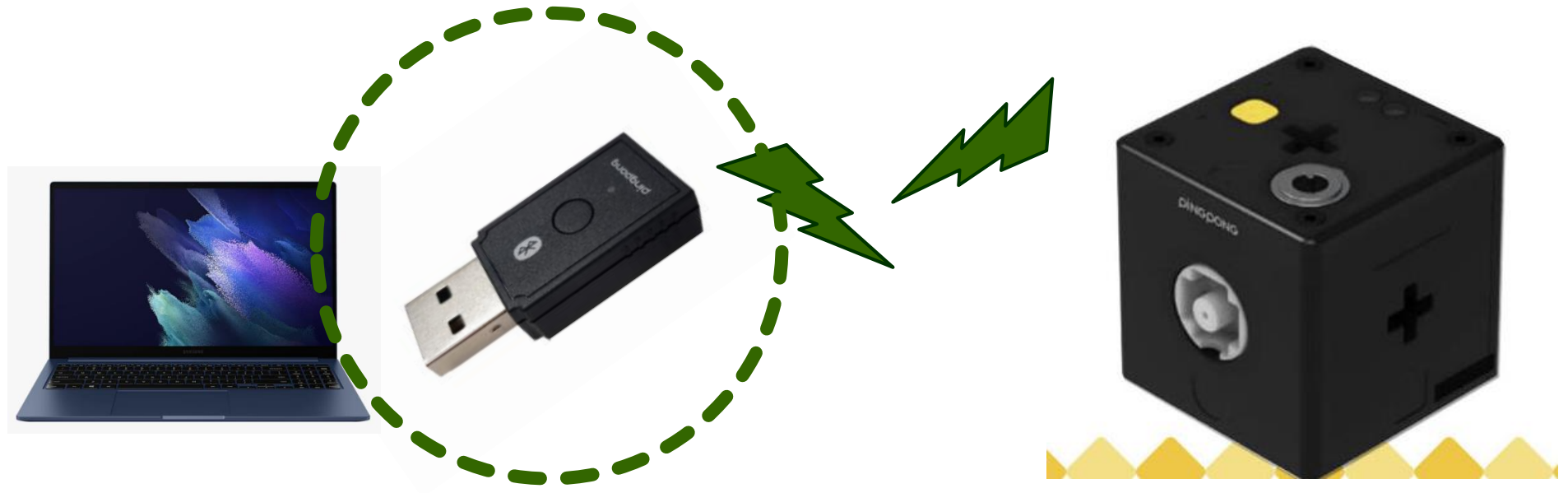


The screenshot shows the JupyterLab interface in a web browser. The address bar indicates the URL is `localhost:8888/tree`. The interface includes a top bar with the Jupyter logo and navigation tabs for Files, Running, and Clusters. Below the tabs, there is a message: "Select items to perform actions on them." The main area displays a file browser view. A green dashed box highlights the 'dev' folder in the file list. A yellow hand icon points to the 'dev' folder. On the right side of the interface, a green box highlights a list of files, including 'chapter8', 'chapter9', 'ex_dev', and several 'ai_chapter' files.

| File Name | Size |
|--------------------|------|
| chapter8 | |
| chapter9 | |
| ex_dev | |
| ai_chapter1.ipynb | |
| ai_chapter10.ipynb | |
| ai_chapter2.ipynb | |
| ai_chapter3.ipynb | |
| ai_chapter4.ipynb | |

핑퐁 큐브 연결

핑퐁 USB 동글 활용
핑퐁 큐브와 노트북 연결



Chapter 1

● 1. 핑퐁로봇과 인공지능

1. 파워셸을 열어 'pingpong-master\examples' 디렉터리로 이동한다.
2. 파워셸에 'jupyter notebook'이라고 입력하여 주피터 노트북을 켜다.

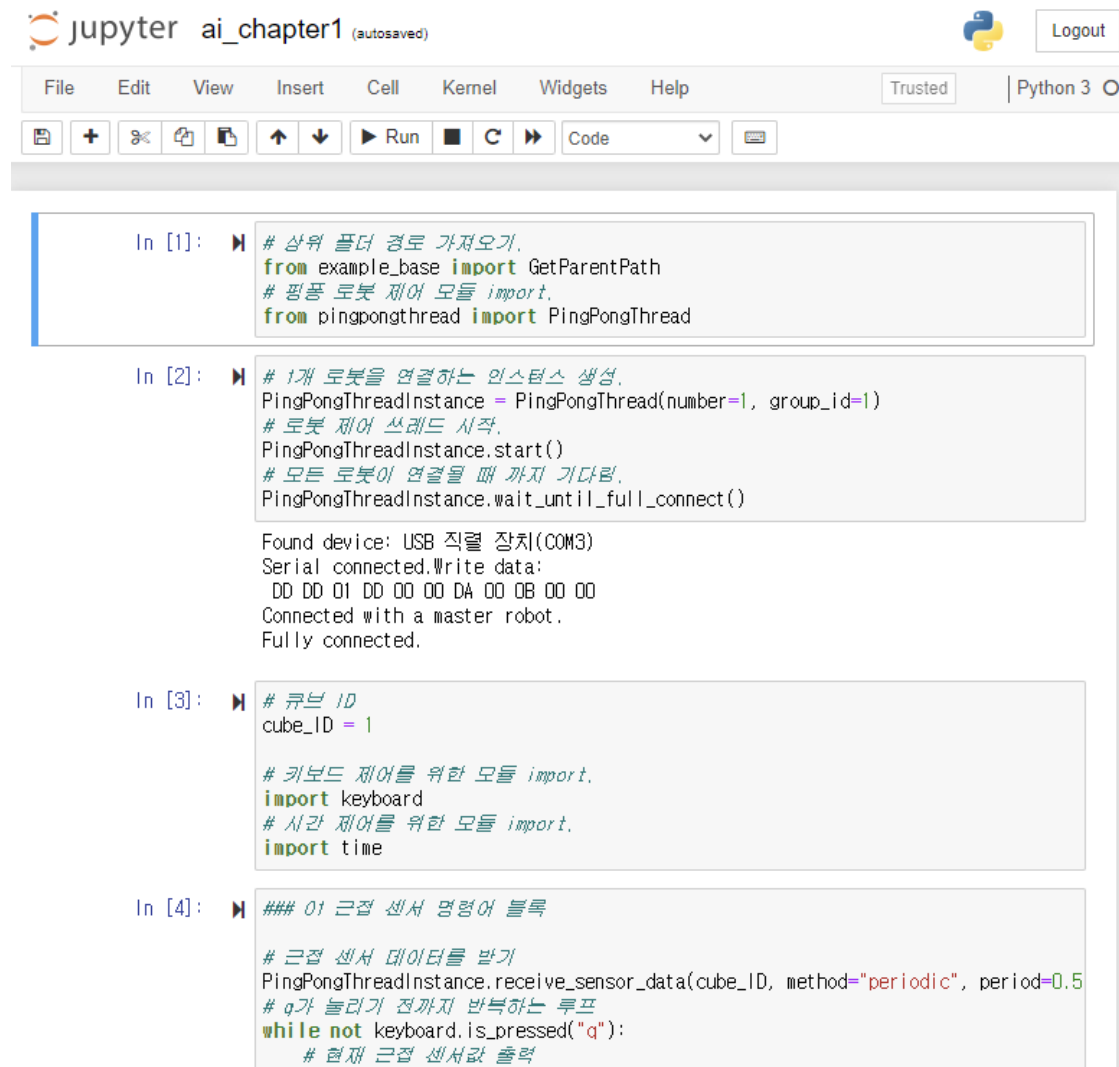
● 1. 핑퐁로봇과 인공지능

| | | |
|--|---------------|----------|
| Files | Running | Clusters |
| Select items to perform actions on them. | | |
| Upload New ↻ | | |
| 0 / Name Last Modified File size | | |
| chapter10 21일 전 | | |
| chapter2 3시간 전 | | |
| chapter4 3시간 전 | | |
| chapter5 21일 전 | | |
| chapter6 2시간 전 | | |
| chapter8 2시간 전 | | |
| chapter9 2시간 전 | | |
| ex_dev 한 달 전 | | |
| ai_chapter1.ipynb | Running 2시간 전 | 6.81 kB |
| ai_chapter10.ipynb | Running 6분 전 | 154 kB |
| ai_chapter2.ipynb | Running 3시간 전 | 6.91 kB |
| ai_chapter3.ipynb | Running 2시간 전 | 4.1 kB |
| ai_chapter4.ipynb | Running 2시간 전 | 155 kB |
| ai_chapter5.ipynb | Running 2시간 전 | 7.93 kB |
| ai_chapter6.ipynb | Running 2시간 전 | 131 kB |
| ai_chapter7.ipynb | Running 한 달 전 | 6.59 kB |
| ai_chapter8.ipynb | Running 2시간 전 | 170 kB |
| ai_chapter9.ipynb | Running 2시간 전 | 8.21 kB |
| ai_chapter1.py | 한 달 전 | 2.42 kB |
| ai_chapter10.py | 2시간 전 | 3.71 kB |
| ai_chapter2.py | 3시간 전 | 2.53 kB |

● 1. 핑퐁로봇과 인공지능

3.

- ▶ 주피터 노트북에서 'ai_chapter1.ipynb'를 클릭하여 연다. 또는, New - Python 3를 눌러 새로운 노트북을 만들고, 다음 순서에 나오는 코드들을 순차적으로 입력하고 실행한다.



```
jupyter ai_chapter1 (autosaved) Logout
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
In [1]: # 상위 폴더 경로 가져오기,
from example_base import GetParentPath
# 핑퐁 로봇 제어 모듈 import,
from pingpongthread import PingPongThread

In [2]: # 1개 로봇을 연결하는 인스턴스 생성,
PingPongThreadInstance = PingPongThread(number=1, group_id=1)
# 로봇 제어 쓰레드 시작,
PingPongThreadInstance.start()
# 모든 로봇이 연결될 때 까지 기다림,
PingPongThreadInstance.wait_until_full_connect()

Found device: USB 직렬 장치(COM3)
Serial connected.Write data:
DD DD 01 DD 00 00 DA 00 0B 00 00
Connected with a master robot.
Fully connected.

In [3]: # 큐브 ID
cube_ID = 1

# 키보드 제어를 위한 모듈 import,
import keyboard
# 시간 제어를 위한 모듈 import,
import time

In [4]: ### 01 근접 센서 명령어 블록

# 근접 센서 데이터를 받기
PingPongThreadInstance.receive_sensor_data(cube_ID, method="periodic", period=0.5)
# q가 눌러지기 전까지 반복하는 루프
while not keyboard.is_pressed("q"):
    # 현재 근접 센서값 출력
```


● 1. 핑퐁로봇과 인공지능

* 주의: 실행한 셀이 끝나기 전에 다른 셀을 실행하지 않는다. 실행 중인 셀은 셀 왼쪽에 'In[*]'이라고 뜨며, 실행이 완료되면 'In[숫자]'라고 뜬다.

▶ 주피터 노트북 설명

- 1) 현재 노트북을 저장한다.
- 2) 선택한 셀 다음에 새로운 셀을 생성한다.
- 3) 선택한 셀을 잘라낸다.
- 4) 선택한 셀을 실행한다.
- 5) 실행되고 있는 셀을 중지한다.
- 6) 실행되는 셀을 중지하고 모든 메모리를 정리한다.



● 1. 핑퐁로봇과 인공지능

4.

- ▶ 상위 경로를 가져오고, 핑퐁 제어 클래스 모듈인 PingPongThread를 가져온다.

```
In [1]: # 상위 폴더 경로 가져오기.  
        from example_base import GetParentPath  
        # 핑퐁 로봇 제어 모듈 import.  
        from pingpongthread import PingPongThread
```

● 1. 핑퐁로봇과 인공지능

5.

▶ 큐브 1개 연결하는 인스턴스 생성하고, 로봇 제어 쓰레드 시작. 'Serial connected.' → 큐브 노란 버튼 두 번 눌러 연결. 'Fully connected.' → 큐브 연결 완료.

- 연결할 큐브의 수를 number 파라미터에, 자신의 그룹 ID를 group_id 파라미터에 넣어 설정. 위 예시는 4번 그룹에 연결한 예.
- 텐서플로우 라이브러리 로딩 시간 다소 소요됨.
- 텐서플로우 MobileNet 모델 처음 생성시 인터넷 연결 필요.

```
In [2]: # 1개 로봇을 연결하는 인스턴스 생성.
PingPongThreadInstance = PingPongThread(number=1, group_id=4)
# 로봇 제어 쓰레드 시작.
PingPongThreadInstance.start()
# 모든 로봇이 연결될 때 까지 기다림.
PingPongThreadInstance.wait_until_full_connect()

# 큐브 ID
cube_ID = 1
```

Found device: Silicon Labs CP210x USB to UART Bridge(COM5)
Serial connected.Write data: DD DD 04 DD 00 00 DA 00 0B 00 00

Connected with a master robot.
Fully connected.

● 1. 핑퐁로봇과 인공지능

6.

▶ 키보드 제어, 시간 제어를 위한 모듈을 import한다.

```
In [3]: # 큐브 ID  
cube_ID = 1  
  
# 키보드 제어를 위한 모듈 import.  
import keyboard  
# 시간 제어를 위한 모듈 import.  
import time
```

● 1. 핑퐁로봇과 인공지능

7.

- ▶ 근접 센서 데이터를 받아 출력하는 작업을 수행한다.
- ▶ 'q'가 눌리기 전까지 0.5초 간격으로 근접 센서 값을 출력한다.
- ▶ 물체가 가까이 있을수록 출력 값이 커지고, 물체가 멀수록 출력 값이 작아진다.

In [4]: `### 01 근접 센서 명령어 블록`

```
# 근접 센서 데이터를 받기
PingPongThreadInstance.receive_sensor_data(cube_ID, method="periodic", period=0.5)
# q가 눌리기 전까지 반복하는 루프
while not keyboard.is_pressed("q"):
    # 현재 근접 센서값 출력
    print(PingPongThreadInstance.get_current_proxy(cube_ID))
    # 0.5초 기다리기
    time.sleep(0.5)
# 센서 데이터 받기 종료
PingPongThreadInstance.stop_sensor_data(cube_ID)
```

● 1. 핑퐁로봇과 인공지능

▶ 출력값:

Write data: FF FF FF 00 00 C8 B8 00 0B 32 01

0

79

74

74

74

74

74

74

76

87

100

108

132

133

76

74

74

73

73

73

73

73

73

Write data: FF FF FF 00 00 C8 B8 00 0B 00 01

● 1. 핑퐁로봇과 인공지능

코드(알고리즘)설명

- 1) 근접 센서 값을 0.5초 단위로 받는다.
- 2) 'q' 키가 눌리기 전까지 계속 반복해서 데이터를 출력한다.
데이터를 출력하면 0.5초를 쉰다.
- 3) 루프에서 나오면 센서 데이터를 그만 받는다.

● 1. 핑퐁로봇과 인공지능

8.

- ▶ 스텝 모터를 회전시키는 작업을 수행한다.
- ▶ 회전 신호를 보낼 때마다 회전하는 시간($\text{angle}/\text{speed} \times 60$) 초만큼 회전을 기다려준다. 자세한 내용은 후술.

In [5]: *### 02 스텝 모터 명령어 블록*

```
# 모터를 반시계 방향으로 90도 회전하기
angle = 90/360 # 회전의 단위는 바퀴(rotation)
speed = 15 # 속력의 단위는 RPM.
PingPongThreadInstance.run_motor_step(cube_ID, speed, -angle)
time.sleep(angle/speed*60)
# 4초 기다리기
time.sleep(4)
# 모터를 시계방향으로 90도 회전하기
PingPongThreadInstance.run_motor_step(cube_ID, speed, angle)
time.sleep(angle/speed*60)
```

Write data: FF FF FF 00 10 00 C1 00 13 02 01 00 02 FC 7C 00 00 01 F4

Write data: FF FF FF 00 10 00 C1 00 13 02 01 00 02 03 84 00 00 01 F4

● 1. 핑퐁로봇과 인공지능

코드 알고리즘 설명

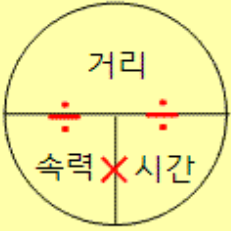
- ▶ 1) 반시계 방향 15RPM으로 90도 회전한다.
- ▶ 2) 4초를 쉰다.
- ▶ 3) 시계 방향 15RPM으로 90도 회전한다.

* 참고: 'run_motor_step' 함수에서, 입력하는 회전 값이 양수이면 시계 방향, 음수이면 반시계 방향으로 회전한다.

● 1. 핑퐁로봇과 인공지능

* 참고: 모터를 돌리는 시간이 왜 ' $\text{angle/speed} \times 60$ '인가?

- ▶ ' run_motor_step ' 함수는 로봇에 회전하라는 신호만 보내는 것이라, 기다려주지 않으면 다음 코드로 넘어가게 된다. 따라서 다음 코드가 실행되는 것을 방지하기 위해 모터를 돌리는 시간 동안 정지하는 것이다.
- ▶ 모터를 돌리는 시간은 ' 거리/속력 '으로 구할 수 있다. 모터가 돌아가는 거리(회전)은 ' angle[Rotation] '이며, 속력은 ' speed[RPM] '이다. RPM은 [Rotation Per Minute]을 의미하며, 1분에 도는 바퀴의 수를 나타낸다. 그러면 시간은 ' angle/speed[min] '이 된다. 단위가 '분'인 것을 '초'가 되도록 60을 곱해주면 ' $\text{angle/15} \times 60[\text{sec}]$ '가 된다.



거리 = 속력 \times 시간

속력 = $\frac{\text{거리}}{\text{시간}}$

시간 = $\frac{\text{거리}}{\text{속력}}$

\sqrt{x} 수학방(mathbang.net)

● 1. 핑퐁로봇과 인공지능

9.

- ▶ 핑퐁 큐브 노란색 버튼 눌리는 것을 감지하는 작업 수행
- ▶ 버튼이 눌리면 1이 출력, 눌리지 않으면 0이 출력.
- ▶ 버튼이 눌리면 "안녕!"을 출력하고 종료.

```
In [6]: ### 03 버튼 명령어 블록

# 버튼 데이터를 받기
PingPongThreadInstance.receive_sensor_data(cube_ID, method="periodic", period=0.5)
# 버튼을 누르는 것을 기다리는 루프
while True:
    # 버튼이 눌리면 "안녕!"을 출력
    if PingPongThreadInstance.get_current_button(cube_ID) == 1:
        print("안녕!")
        # 4초 기다리기
        time.sleep(4)
        break
    # 0.5초 기다리기
    time.sleep(0.5)
# 센서 데이터 받기 종료
PingPongThreadInstance.stop_sensor_data(cube_ID)
```

Write data: FF FF FF 00 00 C8 B8 00 0B 32 01

안녕!

Write data: FF FF FF 00 00 C8 B8 00 0B 00 01

● 1. 핑퐁로봇과 인공지능

코드(알고리즘) 설명

- ▶ 1) 버튼 센서 값을 0.5초 단위로 받는다.
- ▶ 2) 버튼이 눌리기 전까지 0.5초씩 쉬면서 루프를 돌며 기다린다.
- ▶ 3) 버튼이 눌리면 "안녕!"을 출력하고 4초를 쉰다. 그리고 루프를 나온다.
- ▶ 4) 센서 데이터를 그만 받는다.

● 1. 핑퐁로봇과 인공지능

10.

- ▶ 0.1초 간격으로 기울기 센서 값 받는 작업 수행.
- ▶ 네모 방향으로 기울이면 "네모 방향으로 회전하였습니다!"를 출력하고 종료.
- ▶ $\text{abs}(\text{gyro_value}[1]) < 50$, $\text{abs}(\text{gyro_value}[2]) < 50$ 은 네모방향외 다른 방향으로 많이 움직였을 경우 제외.

In [7]: `### 04 기울기 센서 명령어 블록`

```
# 센서 데이터를 받기
PingPongThreadInstance.receive_sensor_data(cube_ID, method="periodic", period=0.1)
# 네모 방향으로 회전하는 것을 기다리는 루프
while True:
    gyro_value = PingPongThreadInstance.get_current_gyro(cube_ID)
    # 네모 방향으로 빠르게 움직이면 "네모 방향으로 회전하였습니다!"를 출력
    if gyro_value[0] < -90 and abs(gyro_value[1]) < 50 and abs(gyro_value[2]) < 50:
        print("네모 방향으로 회전하였습니다!")
        break
    # 0.1초 기다리기
    time.sleep(0.1)
# 센서 데이터 받기 종료
PingPongThreadInstance.stop_sensor_data(cube_ID)
```

Write data: FF FF FF 00 00 C8 B8 00 0B 0A 01

네모 방향으로 회전하였습니다!

Write data: FF FF FF 00 00 C8 B8 00 0B 00 01

● 1. 핑퐁로봇과 인공지능

코드(알고리즘) 설명

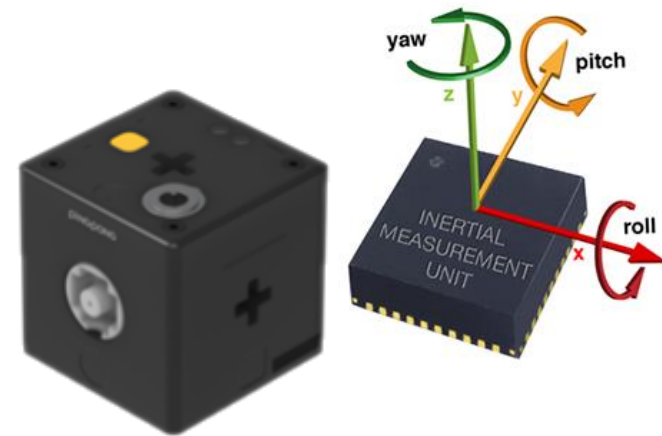
- 1) 기울기 센서 값을 0.1초 단위로 받는다.
- 2) 네모 방향으로 회전하기 전까지 0.1초씩 쉬면서 루프를 돌며 기다린다.
- 3) 네모 방향으로 회전했다는 신호가 나오면 "네모 방향으로 회전하였습니다!"을 출력하고 루프를 나온다.
- 4) 센서 데이터를 그만 받는다.

● 1. 핑퐁로봇과 인공지능

* 참고

- ▶ 기울기 센서 값은 빠르게 변하므로 주기를 0.1초로 설정.
- ▶ 기울기는 3개의 축 방향으로 존재. 따라서 gyro_value의 데이터도 3개. 각 순서대로 roll, pitch, yaw 명칭.
- ▶ 핑퐁 큐브의 하트 모양이 위로 왔을 때를 기준으로, 각 방향으로 기울었을 때 출력되는 센서값은 아래 표 확인.

| 회전 방향 | 출력값 |
|----------------|-----------|
| 별 | roll +90 |
| 네모 | roll -90 |
| 세모 | pitch +90 |
| 동그라미 | pitch -90 |
| (위에서 봤을 때) 우회전 | yaw +90 |
| (위에서 봤을 때) 좌회전 | yaw -90 |



● 1. 핑퐁로봇과 인공지능

11.

- ▶ 로봇 제어 쓰레드를 종료한다.
- ▶ 셀이 종료되면 큐브 노란색 버튼을 2초 이상 눌러 전원을 끈다.
- ▶ 메모리를 비우기 위해 Restart kernel 버튼을 누른다.

* 주의: 중간에 오류가 생겼으면, 반드시 이 버튼을 눌러준다. 이 버튼을 누르지 않으면 다음 연결 시 문제가 발생할 수 있다.

```
In [8]: # 로봇 제어 쓰레드 종료.  
PingPongThreadInstance.end()
```

```
Write data: FF FF FF FF 00 00 A8 00 0A 01
```

```
Disconnected with a master robot.
```

```
Reconnecting with serial...
```

```
Disconnect master robot.
```

```
End thread.
```

```
Found device: Silicon Labs CP210x USB to UART Bridge(COM5)
```

```
Write data: DD DD 00 DD 00 00 DA 00 0B 00 00
```


Chapter 3

● 3. 핑퐁! 현재 시간을 알려줘

1. 파워셸을 열어 'pingpong-master\examples' 디렉터리로 이동한다.
2. 파워셸에 'jupyter notebook'이라고 입력하여 주피터 노트북을 켜다.
3. 주피터 노트북에서 'ai_chapter3.ipynb'를 클릭하여 연다.
또는, New - Python 3를 눌러 새로운 노트북을 만들고, 다음 순서에 나오는 코드들을 순차적으로 입력하고 실행한다.

● 3. 핑퐁! 현재 시간을 알려줘

4.

- ▶ 상위 경로를 가져오고, 핑퐁 제어 클래스 모듈인 PingPongThread를 가져온다.

```
In [1]: # 상위 폴더 경로 가져오기.  
        from example_base import GetParentPath  
        # 핑퐁 로봇 제어 모듈 import.  
        from pingpongthread import PingPongThread
```

● 3. 핑퐁! 현재 시간을 알려줘

5.

- ▶ 큐브 1개 연결하는 인스턴스 생성하고, 로봇 제어 쓰레드 시작. 'Serial connected.' → 큐브 노란 버튼 두 번 눌러 연결. 'Fully connected.' → 큐브 연결 완료.
- ▶ 텐서플로우 라이브러리 로딩 시간 다소 소요됨.
- ▶ 텐서플로우의 MobileNet 모델 처음 생성시 인터넷 연결 필요.
- ▶ 첫 번째로 연결한 로봇이 시침, 두 번째 연결한 로봇이 분침

```
In [2]: # 2개 로봇을 연결하는 인스턴스 생성.  
PingPongThreadInstance = PingPongThread(number=2)  
# 로봇 제어 쓰레드 시작.  
PingPongThreadInstance.start()
```

```
Found device: Silicon Labs CP210x USB to UART Bridge(COM5)  
Serial connected.  
Write data: FF FF 00 FF 20 00 AD 00 0B 0A 00  
Connected with a master robot.  
Connected robots: 2  
Fully connected.
```

● 3. 핑퐁! 현재 시간을 알려줘

6.

- ▶ `get_audio_list()` 함수를 통해 내 컴퓨터에 있는 오디오 장치의 리스트 확인 후 사용할 마이크의 인덱스 확인.

```
In [3]: # 오디오 장치의 리스트를 확인.  
PingPongThreadInstance.get_audio_list()
```

```
Microsoft 사운드 매퍼 - Input  
마이크(Realtek High Definition  
마이크(Steam Streaming Micropho  
Microsoft 사운드 매퍼 - Output  
스피커(Realtek High Definition  
스피커(Steam Streaming Speakers  
Realtek Digital Output(Realtek  
스피커(Steam Streaming Micropho
```

● 3. 핑퐁! 현재 시간을 알려줘

7.

In [4]:

```
# 키보드 제어를 위한 모듈 import
import keyboard
# 시간 제어를 위한 모듈 import
import time
# 시간 정보를 받아오기 위한 모듈 import
import datetime

# text 변수를 TTS(Text To Speech)를 하는 tts_ko 함수로 말하기.
PingPongThreadInstance.tts_ko('스페이스 키를 누른 뒤 "지금 몇시야?"라고 말해보세요!', True)
# 음성 인식을 하는 루프.
while True:
    # 스페이스 바 키가 눌렸으면 목소리 인식.
    if keyboard.is_pressed(" "):
        print("마이크를 인식하는 중입니다.")
        # 목소리 인식을 하는 voice_recognize_ko 함수로 목소리 인식. audio_input_index는 오디오 장치 리스트에서 확인한 마이크의 인덱스.
        result = PingPongThreadInstance.voice_recognize_ko(audio_input_index=0)
        print("결과:", result)
        # result에서 공백 지우기.
        result = result.replace(" ", "")
        # 결과에 '지금 몇 시야'가 있으면 현재 시간을 읽어주고, 아니면 목소리 인식 다시.
        if "지금 몇 시야".replace(" ", "") in result:
            # 루프에서 나감.
            break
        else:
            PingPongThreadInstance.tts_ko("스페이스 키를 누르고 다시 말씀해주세요!", True)
    else:
        time.sleep(0.1)
# 현재 시간 가져와서 읽기.
now = datetime.datetime.now()
now_hour = now.hour
now_minute = now.minute
text = "{}시 {}분 입니다.".format(now_hour, now_minute)
PingPongThreadInstance.tts_ko(text, True)
```

● 3. 핑퐁! 현재 시간을 알려줘

```
# 현재 시각을 시계 각도로 변환. 단위는 바퀴.  
hour_cycle = (now_hour%12)/12  
minute_cycle = now_minute/60  
# 두 모터를 시계 각도만큼 돌리기.  
PingPongThreadInstance.run_motor_step(cube_ID_list=[1, 2], speed_list=15, step_list=[hour_cycle, minute_cycle])  
# 모터를 돌리는 시간인 3초 동안 쉬기.  
time.sleep(3)
```

스페이스 키를 누른 뒤 "지금 몇시야?"라고 말해보세요!
마이크를 인식하는 중입니다.
결과: Voice recognition failed.

WARNING: Logging before flag parsing goes to stderr.
W1116 14:32:35.692564 19012 tts.py:134] Unable to get language list: 'NoneType' object is not subscriptable

스페이스 키를 누르고 다시 말씀해주세요!
마이크를 인식하는 중입니다.
결과: 지금 몇 시야
14 시 32 분입니다.
Write data: FF FF FF AA 20 00 CD 00 33 02 01 00 00 FF FF FF 00 20 00 C1 00 13 02 01 00 02 03 84 00 00 01 4D FF FF FF 01 20 00 C1 00 13 02 01 00 0
2 03 84 00 00 04 2B
Aggregator set.

* 참고: 파이썬에서 '%'는 나머지를 의미한다.
따라서 'now_hour%12'는 현재 시간을 12로 나눈 나머지이다.

● 3. 핑퐁! 현재 시간을 알려줘

* 참고: 'run_motor_step' 함수에서, 입력하는 회전 값이 양수이면 시계 방향, 음수이면 반시계 방향으로 회전한다.

* 주의: tts_ko 함수와 voice_recognize_ko 함수는 인터넷이 연결되어 있어야만 사용이 가능하다.

* 주의: tts_ko 함수는 한국어만 말할 수 있다.

voice_recognize_ko 함수는 한국어만 인식할 수 있다.

* 참고: tts_ko 함수의 두 번째 인자는 출력(print)를 할 것인지에 대한 것이다. True이면 출력을 하고, False이면 출력하지 않는다.

● 3. 핑퐁! 현재 시간을 알려줘

- 1) 스페이스 바 키를 누르면, 음성 인식을 시작한다.
- 2) 음성 인식에서 '지금 몇 시야'라는 문장이 인식되면, 현재 시각을 알려준다. 인식되지 않으면 스페이스 바 키를 누르고 음성 인식을 다시 한다.
- 3) 현재 시각은 24시간제로 알려주며, 음성으로 말해준다.
- 4) 현재 시각을 시계의 시침과 분침 각도로 변환하여, 핑퐁 로봇의 스테퍼 모터를 해당하는 각도만큼 돌린다.

● 3. 핑퐁! 현재 시간을 알려줘

8.

- ▶ 연결을 끊고 스레드를 종료한다.
- ▶ 셀이 종료되면 각 큐브 노란 버튼을 2초 이상 눌러 전원 OFF.
- ▶ 메모리를 비우기 위해 Restart kernel 버튼을 누른다.
- ▶ 이 버튼을 누르지 않으면 다음 연결 시 문제가 발생할 수 있다.

```
In [5]: PingPongThreadInstance.end()
```

```
Write data: FF FF FF FF 00 00 A8 00 0A 01  
Disconnect master robot.  
Serial disconnected. Sleep 3 seconds.  
End thread.
```

Chapter 7

● 7. 자율 주행 오토카

1. 파워셸을 열어 'pingpong-master\examples' 디렉터리로 이동한다.
2. 파워셸에 'jupyter notebook'이라고 입력하여 주피터 노트북을 켜다.
3. 주피터 노트북에서 'ai_chapter7.ipynb'를 클릭하여 연다.
또는, New - Python 3를 눌러 새로운 노트북을 만들고, 다음 순서에 나오는 코드들을 순차적으로 입력하고 실행한다.

● 7. 자율 주행 오토카

4.

- ▶ 상위 경로를 가져오고, 핑퐁 제어 클래스 모듈인 PingPongThread를 가져온다.

```
In [1]: # 상위 폴더 경로 가져오기.  
from example_base import GetParentPath  
# 핑퐁 로봇 제어 모듈 import.  
from pingpongthread import PingPongThread
```

● 7. 자율 주행 오토카

5.

- ▶ 자신의 그룹 ID에 큐브 2개 연결하는 인스턴스 생성하고, 로봇제어 쓰레드 시작. 'Serial connected.' -> 각 큐브 노란 버튼 두 번씩 눌러 연결. 'Fully connected.' -> 큐브 연결 완료
- ▶ 텐서플로우 라이브러리 로딩 시간 다소 소요됨.
- ▶ 텐서플로우 MobileNet 모델 처음 생성시 인터넷 연결 필요.

```
In [2]: # 2개 로봇을 연결하는 인스턴스 생성.  
PingPongThreadInstance = PingPongThread(number=2)  
# 로봇 제어 쓰레드 시작.  
PingPongThreadInstance.start()  
# 모든 로봇이 연결될 때 까지 기다림.  
PingPongThreadInstance.wait_until_full_connect()
```

```
Found device: Silicon Labs CP210x USB to UART Bridge(COM5)  
Serial connected.  
Write data: FF FF 00 FF 20 00 AD 00 0B 0A 00  
Connected with a master robot.  
Connected robots: 2  
Fully connected.
```

● 7. 자율 주행 오토카

6.

```
In [3]: # 시간 제어를 위한 모듈 import
import time

### 320도 앞으로 회전.
PingPongThreadInstance.tts_ko("안전벨트를 매주세요. 320 미터 직진입니다.", True)
time.sleep(2)
angle = 320/360 # 회전의 단위는 바퀴(rotation).
PingPongThreadInstance.run_motor_step([1, 2], 15, [-angle, angle]) # 15RPM으로 모터 돌리기.
time.sleep(angle/15*60) # 모터 돌리는 시간 동안 정지.
### 2초 정지.
PingPongThreadInstance.tts_ko("2초간 멈추세요.", True)
time.sleep(2)
### 200도 앞으로 회전.
PingPongThreadInstance.tts_ko("200미터 앞 우회전입니다.", True)
time.sleep(1)
angle = 200/360 # 회전의 단위는 바퀴(rotation).
PingPongThreadInstance.run_motor_step([1, 2], 15, [-angle, angle]) # 15RPM으로 모터 돌리기.
time.sleep(angle/15*60) # 모터 돌리는 시간 동안 정지.
### 107도를 돌려 우회전.
PingPongThreadInstance.tts_ko("우회전해 주세요.", True)
time.sleep(1)
angle = 107/360 # 회전의 단위는 바퀴(rotation).
PingPongThreadInstance.run_motor_step([1, 2], 15, [-angle, -angle]) # 15RPM으로 모터 돌리기.
time.sleep(angle/15*60) # 모터 돌리는 시간 동안 정지.
### 500도 앞으로 회전.
PingPongThreadInstance.tts_ko("500미터 직진입니다.", True)
time.sleep(1)
angle = 500/360 # 회전의 단위는 바퀴(rotation).
PingPongThreadInstance.run_motor_step([1, 2], 15, [-angle, angle]) # 15RPM으로 모터 돌리기.
time.sleep(angle/15*60) # 모터 돌리는 시간 동안 정지.
```

* 참고: 모터를 돌리는 시간이 왜 'angle/15*60'인지는 이후 페이지 설명 참조

● 7. 자율 주행 오토카

```
### 5초 정지.  
PingPongThreadInstance.tts_ko("신호등이 있습니다. 잠시 기다려주세요.", True)  
time.sleep(5)  
### 300도 앞으로 회전.  
PingPongThreadInstance.tts_ko("300미터 전방 목적지가 있습니다.", True)  
time.sleep(1)  
angle = 300/360 # 회전의 단위는 바퀴(rotation).  
PingPongThreadInstance.run_motor_step([1, 2], 15, [-angle, angle]) # 15RPM으로 모터 돌리기.  
time.sleep(angle/15*60) # 모터 돌리는 시간 동안 정지.  
### 도착.  
PingPongThreadInstance.tts_ko("목적지에 도착했습니다.", True)
```

안전벨트를 매주세요. 320 미터 직진입니다.

Write data: FF FF FF AA 20 00 CD 00 33 02 01 00 00 FF FF FF 00 20 00 C1 00 13 02 01 00 02 FC 7C 00 00 06 F2 FF FF FF 01 20 00 C1 00 13 02 01 00 02 03 84 00 00 06 F2

Aggregator set.

2초간 멈추세요.

200미터 앞 우회전입니다.

Write data: FF FF FF AA 20 00 CD 00 33 02 01 00 00 FF FF FF 00 20 00 C1 00 13 02 01 00 02 FC 7C 00 00 04 57 FF FF FF 01 20 00 C1 00 13 02 01 00 02 03 84 00 00 04 57

Aggregator set.

우회전해 주세요.

Write data: FF FF FF AA 20 00 CD 00 33 02 01 00 00 FF FF FF 00 20 00 C1 00 13 02 01 00 02 FC 7C 00 00 02 52 FF FF FF 01 20 00 C1 00 13 02 01 00 02 2 FC 7C 00 00 02 52

Aggregator set.

500미터 직진입니다.

Write data: FF FF FF AA 20 00 CD 00 33 02 01 00 00 FF FF FF 00 20 00 C1 00 13 02 01 00 02 FC 7C 00 00 0A DA FF FF FF 01 20 00 C1 00 13 02 01 00 02 03 84 00 00 0A DA

Aggregator set.

신호등이 있습니다. 잠시 기다려주세요.

300미터 전방 목적지가 있습니다.

Write data: FF FF FF AA 20 00 CD 00 33 02 01 00 00 FF FF FF 00 20 00 C1 00 13 02 01 00 02 FC 7C 00 00 06 83 FF FF FF 01 20 00 C1 00 13 02 01 00 02 03 84 00 00 06 83

Aggregator set.

목적지에 도착했습니다.

● 7. 자율 주행 오토카

- * 참고: 'run_motor_step' 함수에서, 입력하는 회전 값이 양수이면 시계 방향, 음수이면 반시계 방향으로 회전한다.
- * 참고: 조립된 로봇의 모양에 의해, 1번 큐브가 반시계 방향, 2번 큐브가 시계 방향으로 회전하면 로봇은 앞으로 간다. 1번, 2번 큐브 둘 다 반시계 방향으로 회전하면 로봇은 우회전한다.
- * 주의: tts_ko 함수는 인터넷이 연결되어 있어야만 사용이 가능하다. 또한 tts_ko 함수는 한국어만 말할 수 있다.
- * 참고: tts_ko 함수의 두 번째 인자는 출력(print)를 할 것인지에 대한 것이다. True이면 출력을 하고, False이면 출력하지 않는다.

● 7. 자율 주행 오토카

- 1) 2초를 쉬고 15RPM으로 320도(deg)를 움직인다.
- 2) 2초 동안 정지한다.
- 3) 1초를 쉬고 15RPM으로 200도(deg)를 움직인다.
- 4) 1초를 쉬고 15RPM으로 107도(deg)만큼 오른쪽으로 회전한다.
- 5) 1초를 쉬고 15RPM으로 500도(deg)를 움직인다.
- 6) 5초 동안 정지한다.
- 7) 1초를 쉬고 15RPM으로 300도(deg)를 움직인다.

● 7. 자율 주행 오토카

- * 참고: 모터를 돌리는 시간이 왜 ' $\text{angle}/\text{speed} \times 60$ '인가?
- ' run_motor_step ' 함수는 로봇에 회전하라는 신호만 보내는 것이라, 기다려주지 않으면 다음 코드로 넘어가게 된다. 따라서 다음 코드가 실행되는 것을 방지하기 위해 모터를 돌리는 시간 동안 정지하는 것이다.
- 모터를 돌리는 시간은 ' $\text{거리}/\text{속력}$ '으로 구할 수 있다. 모터가 돌아가는 거리(회전)은 ' $\text{angle}[\text{Rotation}]$ '이며, 속력은 ' $\text{speed}[\text{RPM}]$ '이다. RPM은 [Rotation Per Minute]을 의미하며, 1분에 도는 바퀴의 수를 나타낸다. 그러면 시간은 ' $\text{angle}/\text{speed}[\text{min}]$ '이 된다. 단위가 '분'인 것을 '초'가 되도록 60을 곱해주면 ' $\text{angle}/15 \times 60[\text{sec}]$ '가 된다.

거리 = 속력 \times 시간

속력 = $\frac{\text{거리}}{\text{시간}}$

시간 = $\frac{\text{거리}}{\text{속력}}$

\sqrt{x} 수학방(mathbang.net)

● 7. 자율 주행 오토카

6.

- ▶ 로봇 제어 스레드를 종료한다.
- ▶ 셀이 종료되면 각 큐브 노란 버튼을 2초 이상 눌러 전원 OFF.
- ▶ 메모리를 비우기 위해 Restart kernel 버튼을 누른다.

▶ In [4]: `# 로봇 제어 스레드 종료.`
`PingPongThreadInstance.end()` 있다.

```
Write data: FF FF FF FF 00 00 A8 00 0A 01
Disconnect master robot.
Serial disconnected. Sleep 3 seconds.
End thread.
```

Thank you