Spring 2022

Algorithms

Assignment #5 (Open Source SW Project)

Prof. Eunwoo Kim
School of Computer Science and Engineering
Chung-Ang University

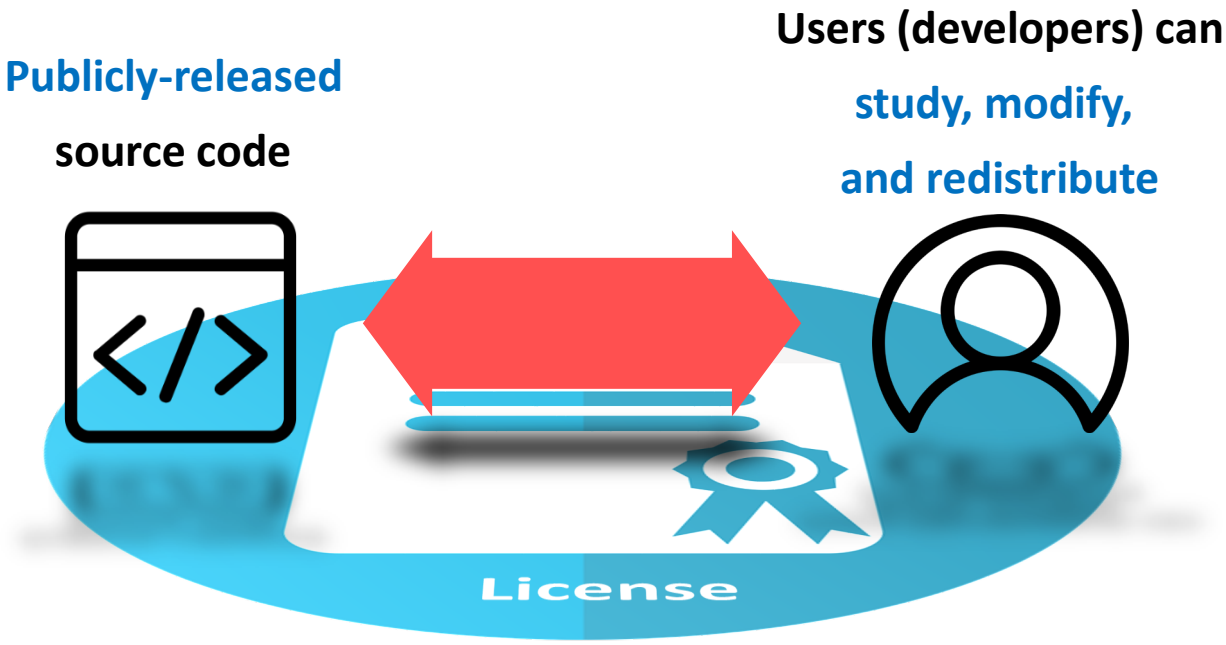
Contents

- Introduction to Open Source SW
- Open Source SWs
  - Anaconda
  - TensorFlow
  - Jupyter Notebook
  - Github
- Introduction to Deep Learning and Classification Problems
- Project Description

Introduction to Open Source SW

Credit: Hyosu Kim (CSE@CAU), https://sites.google.com/view/hyosukim, Open Source SW Project (Undergraduate: Spring 2019)

What is OSS (Open Source SW)?

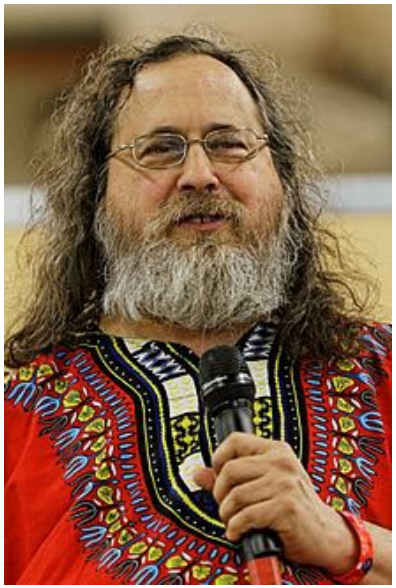


OSS is a type of computer software in which **source code is released under a license** in which the copyright holder grants users **the rights to study, change, distribute the software**

[St. Laurent et al., "Understanding Open Source and Free Software Licensing", O'Reilly Media]

History of OSS

Richard Stallman (Activist, programmer)



"Software users should have the freedom to share with their neighbors and be able to study and make changes to the SW that they use"

"Attempts by proprietary SW vendors to prohibit these acts are antisocial and unethical"

History of OSS

FREE SOFTWARE FOUNDATION

Free SW Movement

Liberating every one (every computer users) in cyberspace

- Users should have the freedom to run, copy, distribute, study, change, and improve SW
- Freedom to distribute, not freedom from cost (freedom of speech vs free beer)

Copyleft

A form of licensing which forces an author to give every person permission to reproduce, adapt, or distribute SW

- Any resulting copies or adaptations are also bound by the same licensing rules (exception: SW libraries)

History of OSS

GNU (GNU is Not Unix) Project (1983, https://www.gnu.org)

"Give computer users freedom and control in their use of their computers and computing devices"

"The entire SW of a computer should grant its users all freedom rights"

The founding goal of the project: Building a free operating system instead of Unix!!!

GNU/Linux

GNU Emacs

GCC

glibc

ls, grep, make, …

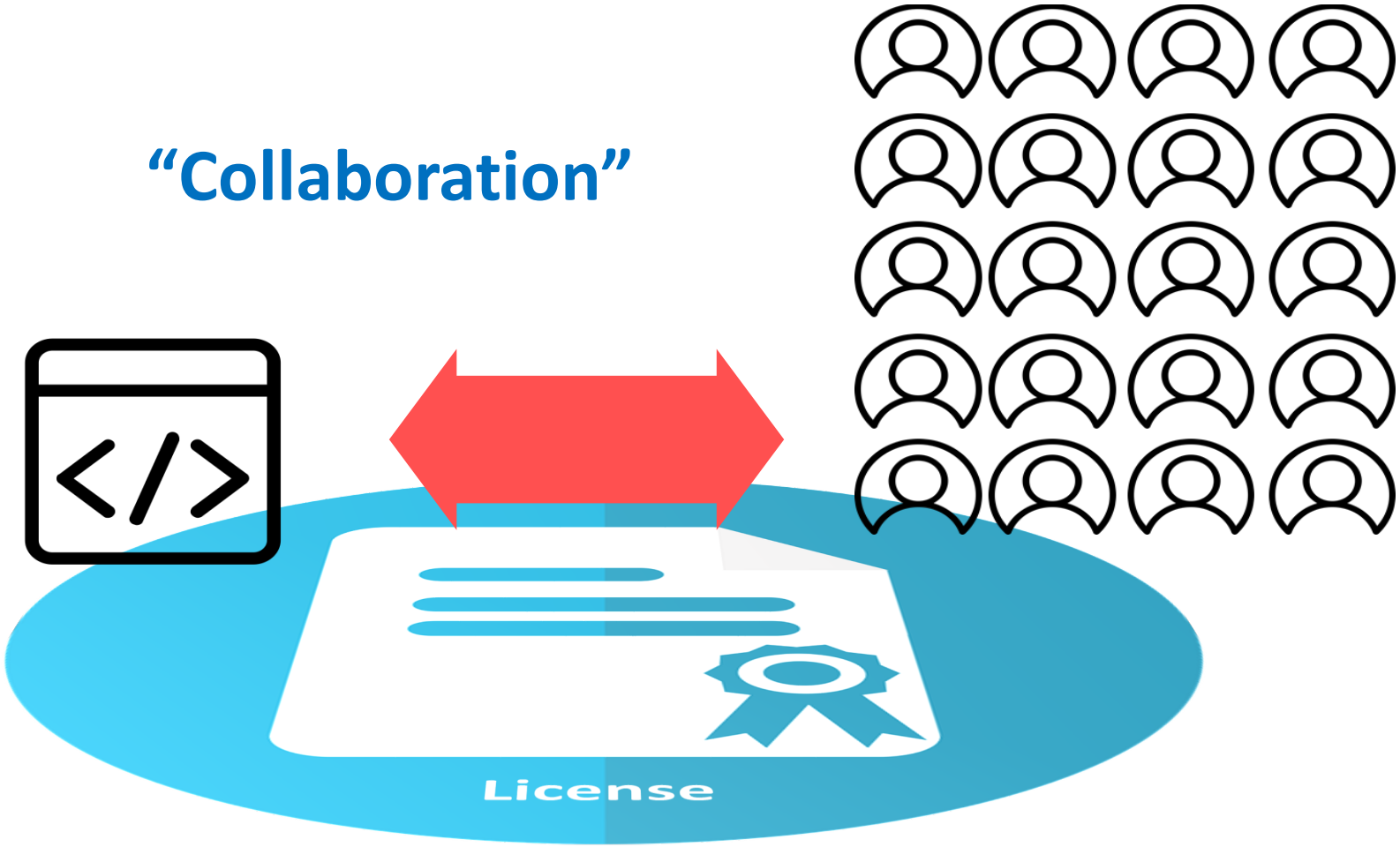
History of OSS

Free software, free society: Richard Stallman at TEDxGeneva 2014 (13min)

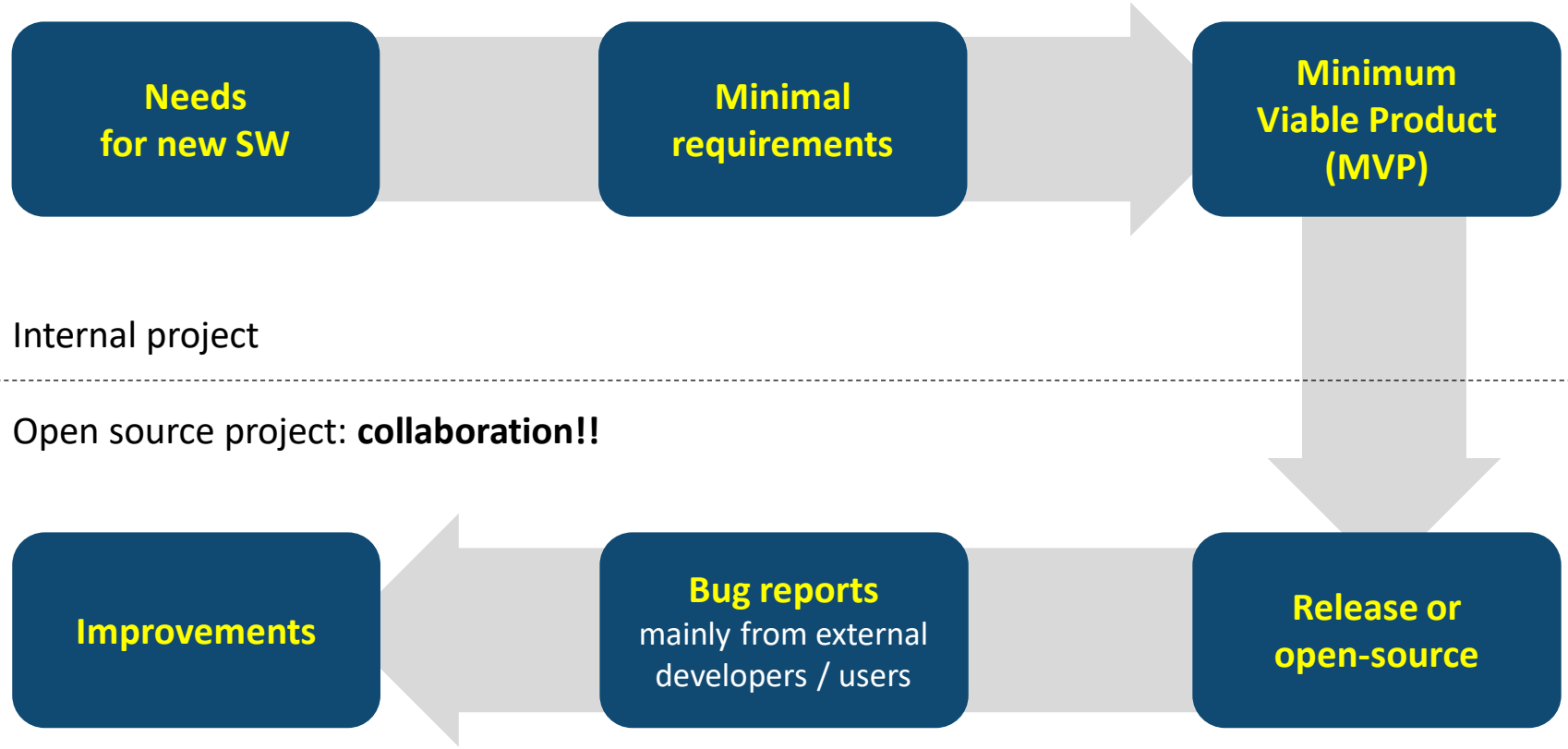
(https://www.youtube.com/watch?v=Ag1AKIl_2GM&t=43s)

OSS also means

“Collaboration”



OSS Project Life Cycle



OSS: Pros and Cons

- Pros
  - Easy-to-access / obtain
  - Flexibility
  - Globalization of market

- Cons
  - High entry barriers
  - No guarantee of development/update
  - High security threats
  - Little financial incentives (less motivation) for developers

OSS in Real-World

- **Most popular OSS** (Top 10 OSS projects on GitHub In 2018)

Rank	OSS projects	# of contributors
1	Microsoft / vscode	19K
2	Facebook / react-native	10K
3	Tensorflow / tensorflow	9.3K
4	Angular / angular-cli	8.8K
5	MicrosoftDocs / azure-docs	7.8K
6	Angular / angular	7.6K
7	Ansible / ansible	7.5K
8	Kubernetes / kubernetes	6.5K
9	Npm / npm	6.1K
10	DefinitelyTyped / DefinitelyTyped	6.0K

How to Participate

- The way to find open source SW projects
  1. Open source projects which you're currently using
  2. Open source campaigns/competitions
  3. Open source organizations
  4. A famous company's open source projects

Famous company's open source projects

Adobe	https://github.com/adobe/adobe.github.com
Netflix	https://netflix.github.io/
Twitter	https://twitter.github.io/
IBM	http://ibm.github.io/
Microsoft	https://opensource.microsoft.com/
Google	https://opensource.google.com
Apple	https://developer.apple.com/opensource/
Facebook	https://opensource.facebook.com/
Amazon	https://aws.amazon.com/ko/opensource/

Open Source SWs

- Anaconda
- TensorFlow
- Jupyter Notebook
- Github

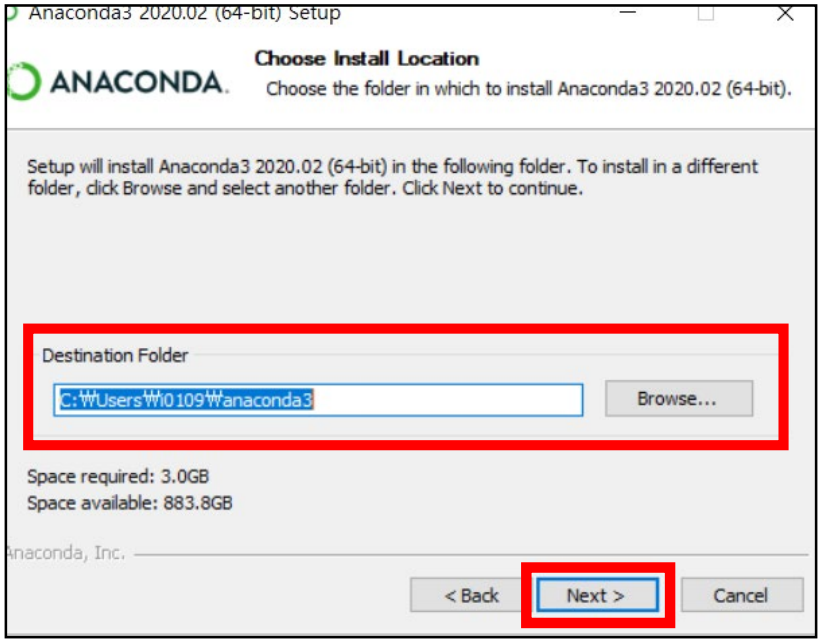
Anaconda

- Anaconda installation on Windows OS
- Check your system out (32 bit or 64 bit)
  - Control panel – System and Security – System

- Download link
  - https://www.anaconda.com/products/individual#download-section
  - Install with respect to your machine's OS and system environment

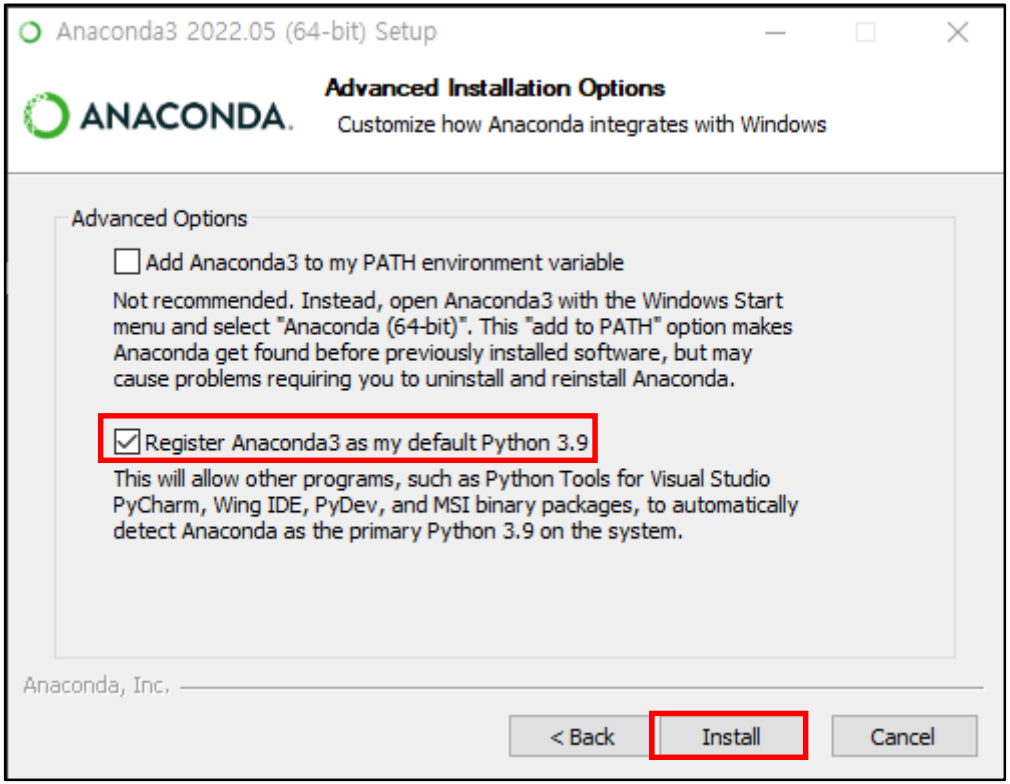
Anaconda installation

- "Destination Folder" means the location where the Anaconda is downloaded.
- Note: you should remember "Destination Folder".
  - It may be different from your computer's Destination Folder.



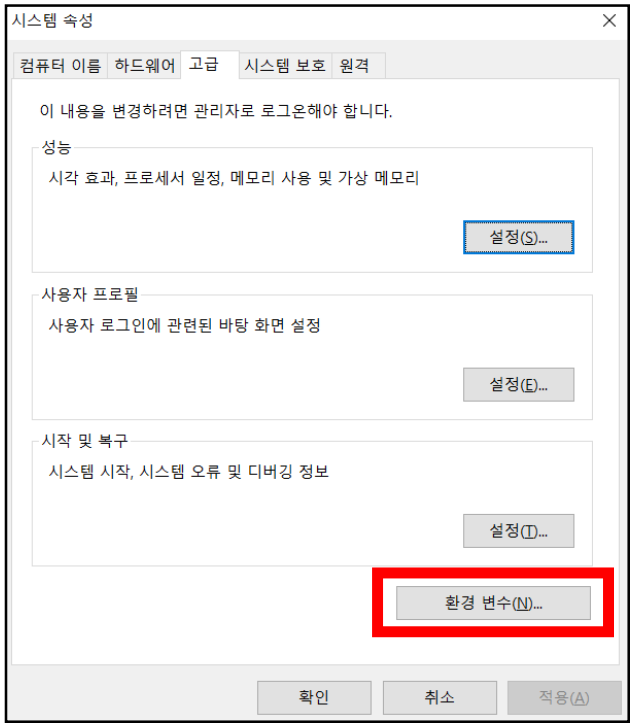
Anaconda installation

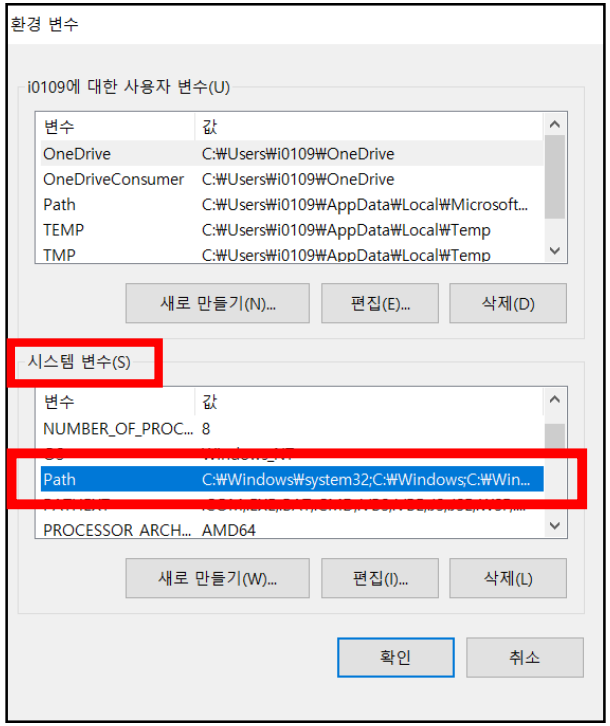
- Check “Register Anaconda3 as my default Python 3.9” and Install



Setting environment variables

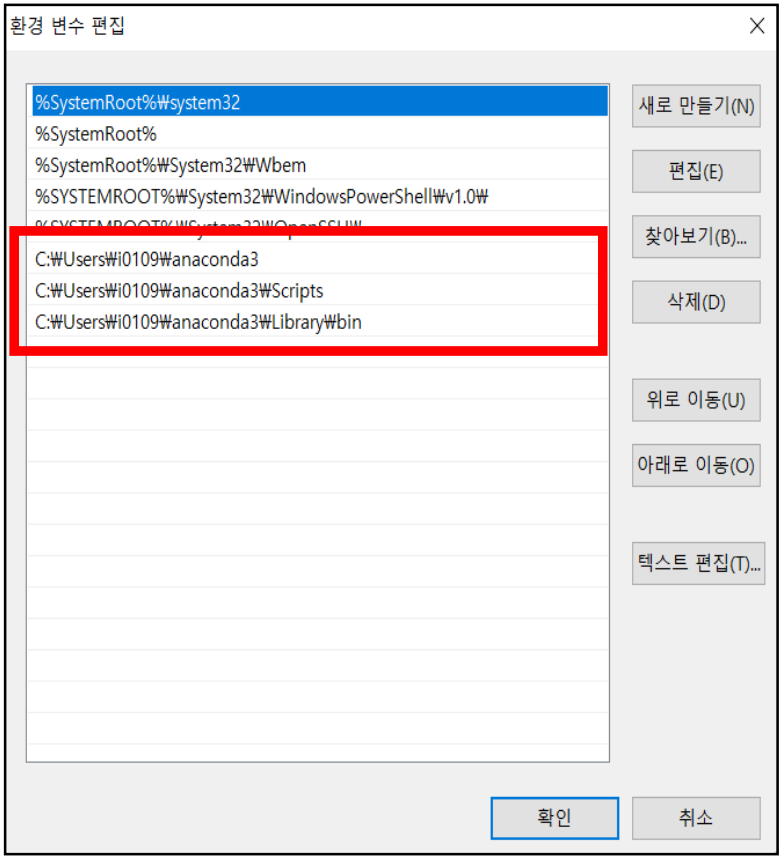
- Go to [Control Panel – System and Security – System] – [Advanced System Settings]
- Click [Environment Variables] (see bottom left)
- Double click the path (see bottom right)





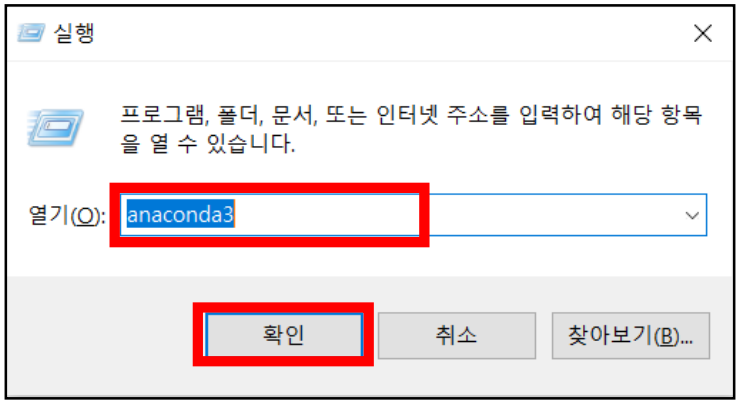
Setting environment variables

- Click [New]
- Add all the three addresses
  - Destination Folder
  - Destination Folder\Scripts
  - Destination Folder\Library\bin
- Destination Folder should be your one
  - See the previous slide



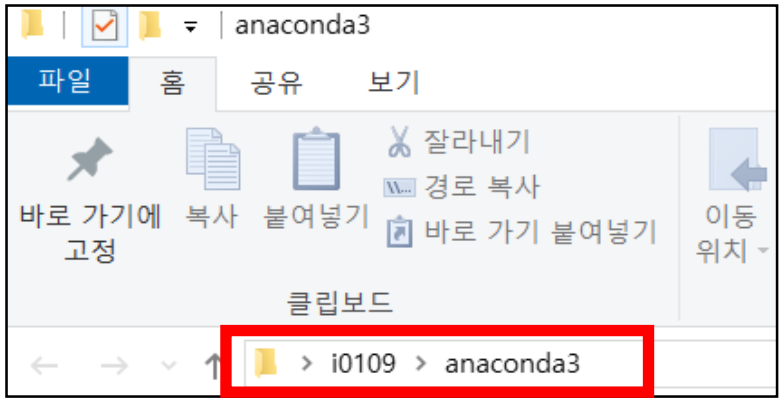
Setting environment variables

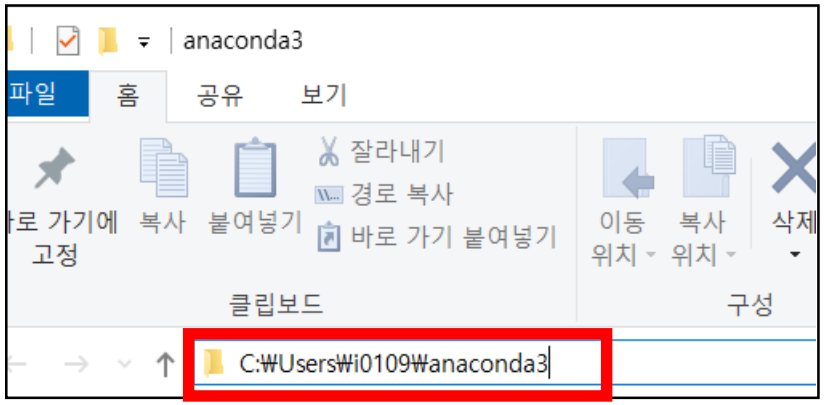
- Put [Win key + R] and enter “anaconda3”



Setting environment variables

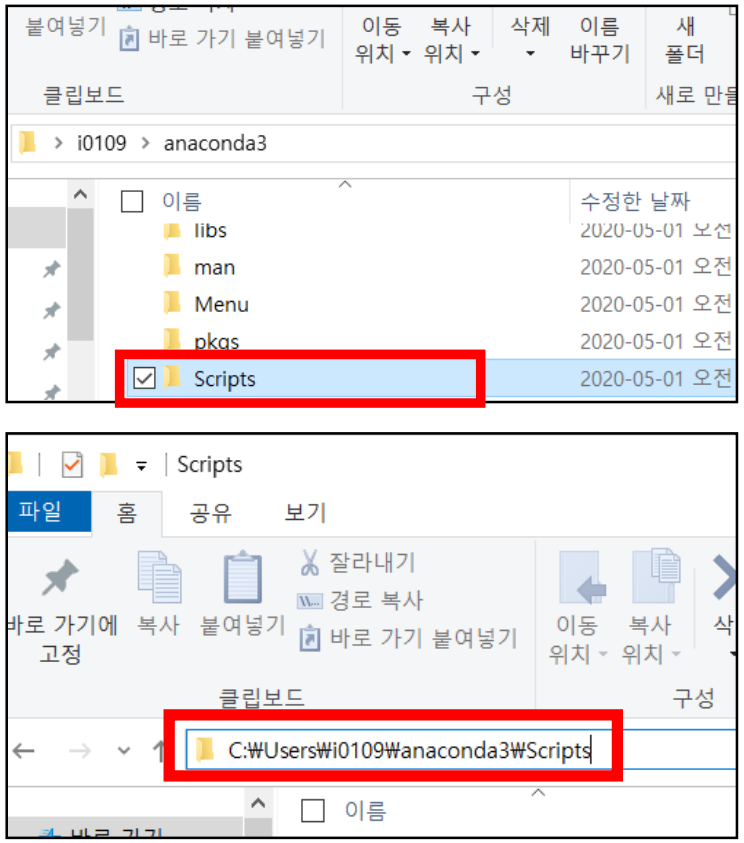
- Click the red box
- Then there is the “Destination Folder”
- Copy and Paste the “Destination Folder” to the environment variables





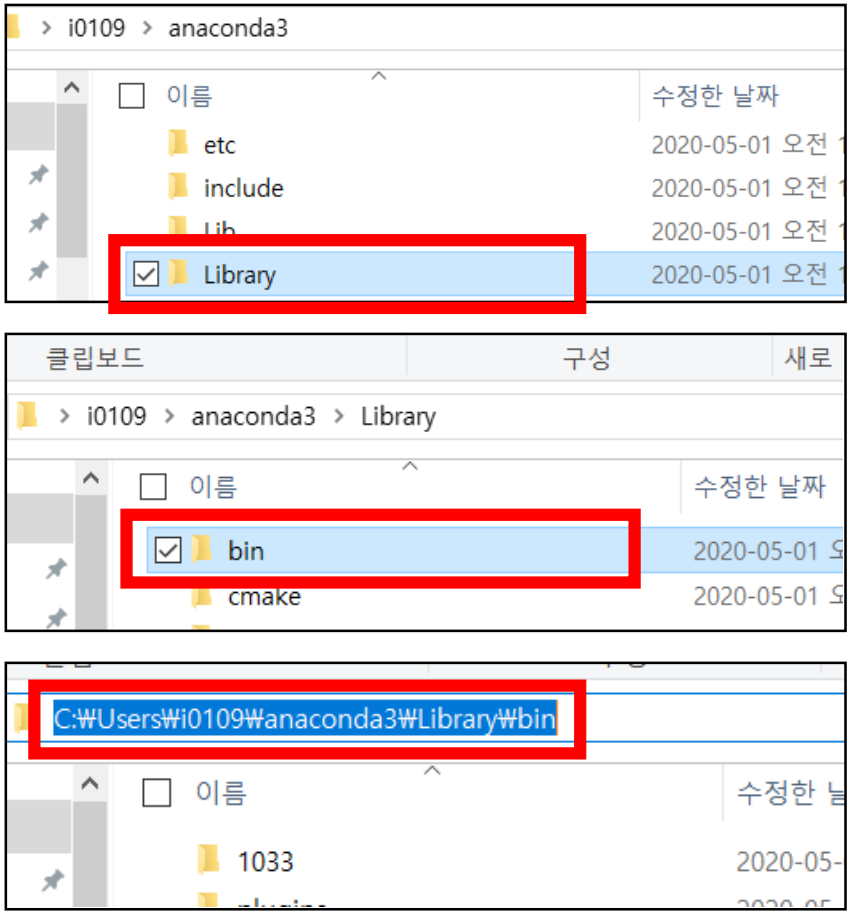
Setting environment variables

1. Scroll down and find “Scripts”
2. Double click “Scripts”
3. Then there is the Destination Folder\Scripts
3. Copy and paste the Destination Folder\Scripts to the environment variables



Setting environment variables

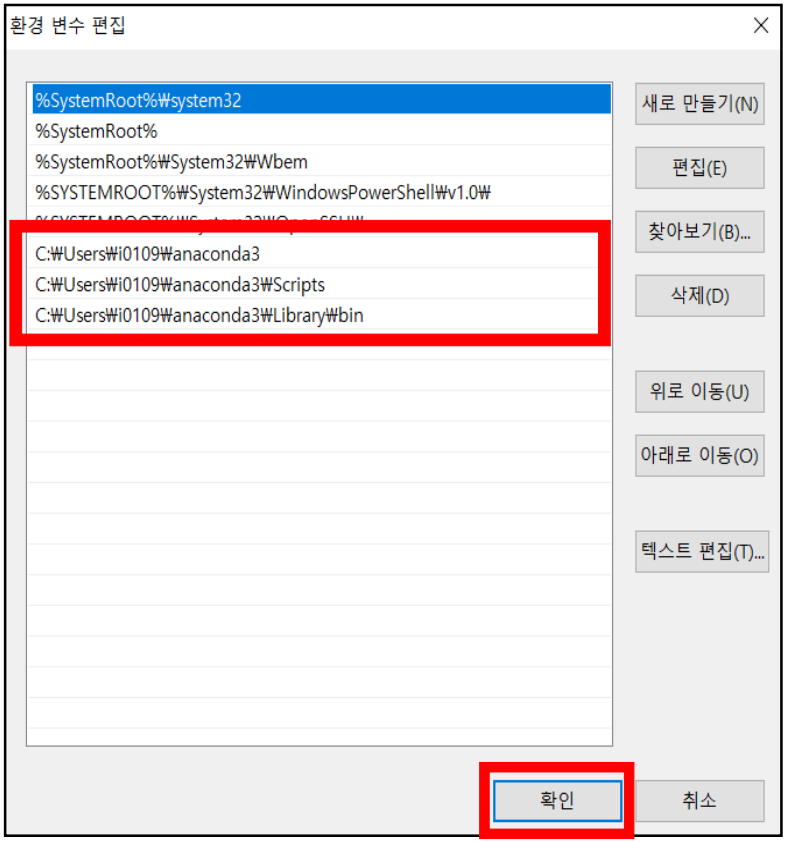
1. Exit the “Scripts” folder
2. Scroll down and find “Library”
3. Double click “Library”
4. Scroll down and find “bin”
5. Double click “bin”
6. Then there is the Destination Folder\Library\bin
7. Copy and paste the Destination Folder\Library\bin to the environment variables



Setting environment variables

1. Check the environment variables are added
2. If everything is Ok, push the confirm button.
3. Anaconda installation is done.

If you reboot the machine,
the environment variables will be applied.

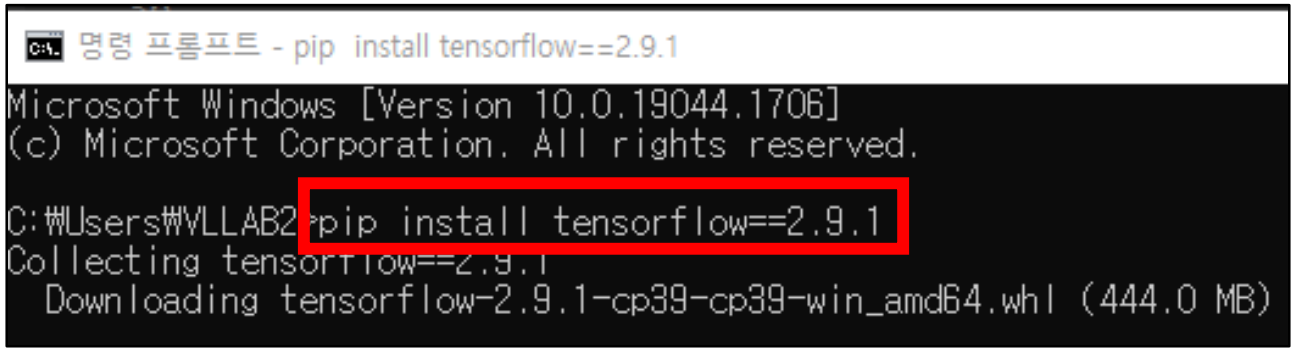


TensorFlow

- Prerequisite : install the Anaconda
- Use the keyboard Shortcut: [Win + R] or [Win + S]
- Enter “cmd”

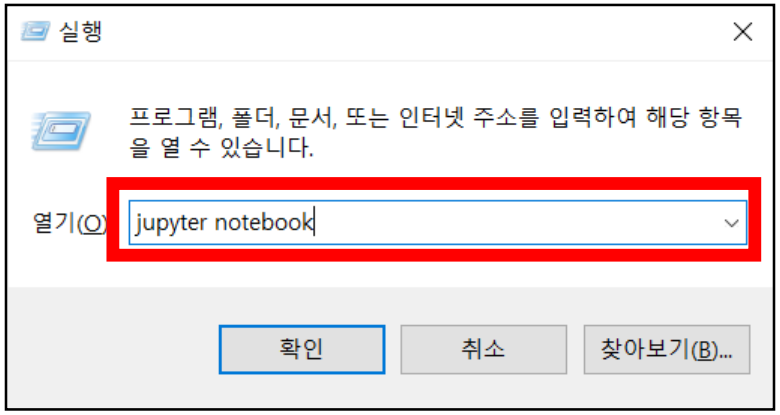
- Type the TensorFlow CPU version installation command

“pip install tensorflow==2.9.1”



Jupyter Notebook

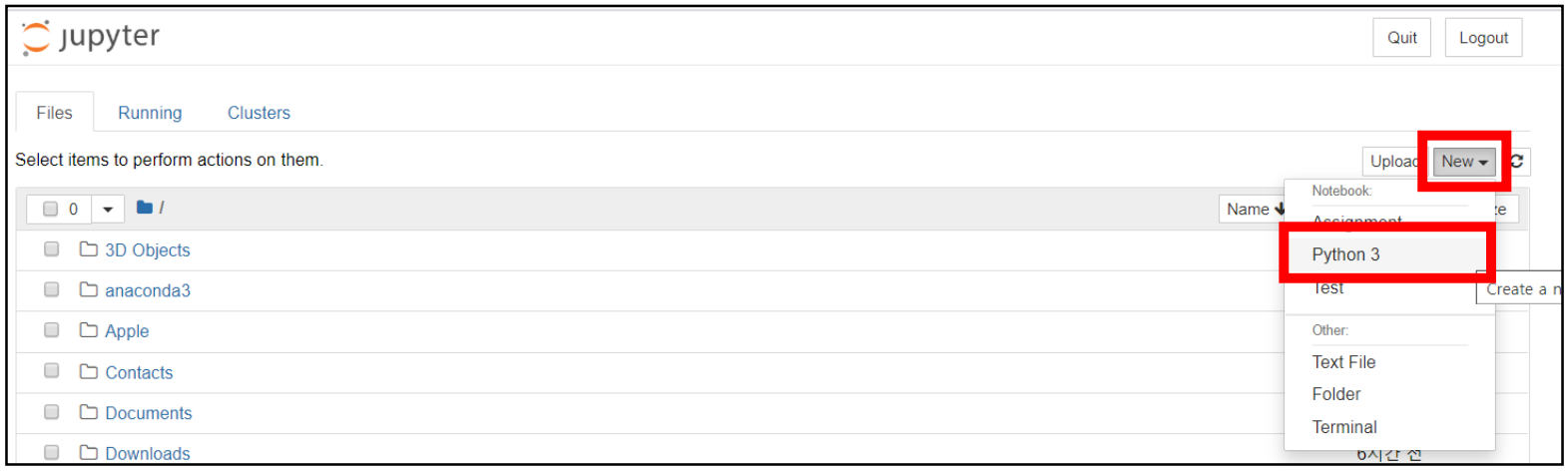
- Use the keyboard Shortcut: [Win + R] or [Win + S]
- Enter “jupyter notebook”



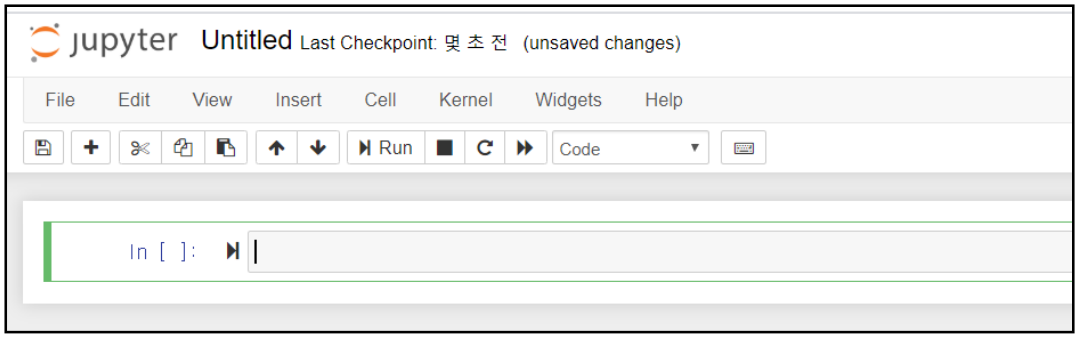
- In this project, we do NOT use other IDEs, such as Pycharm.

Jupyter Notebook

- Click the [New] – [Python 3]

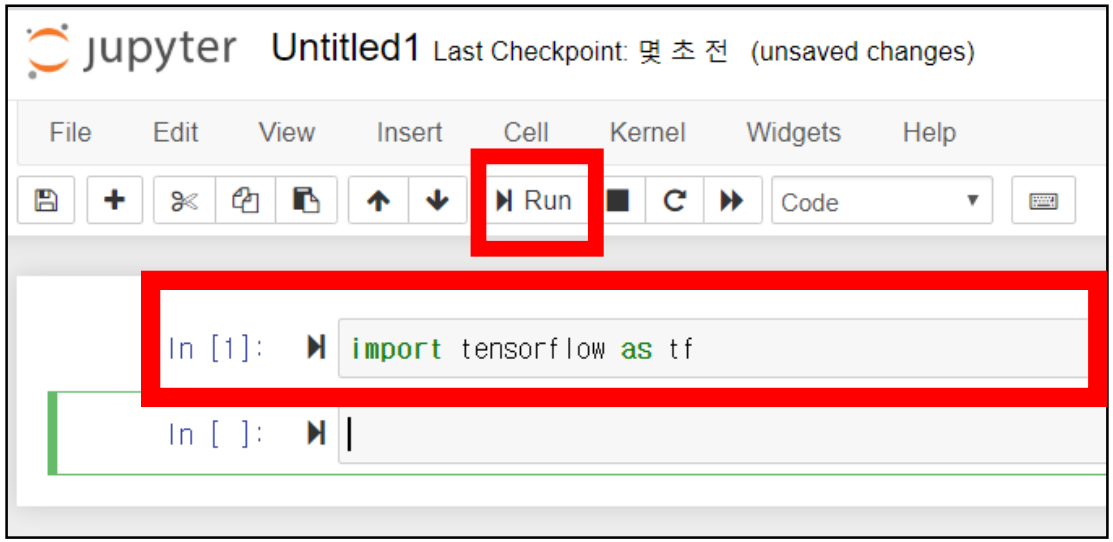


- Initial screen



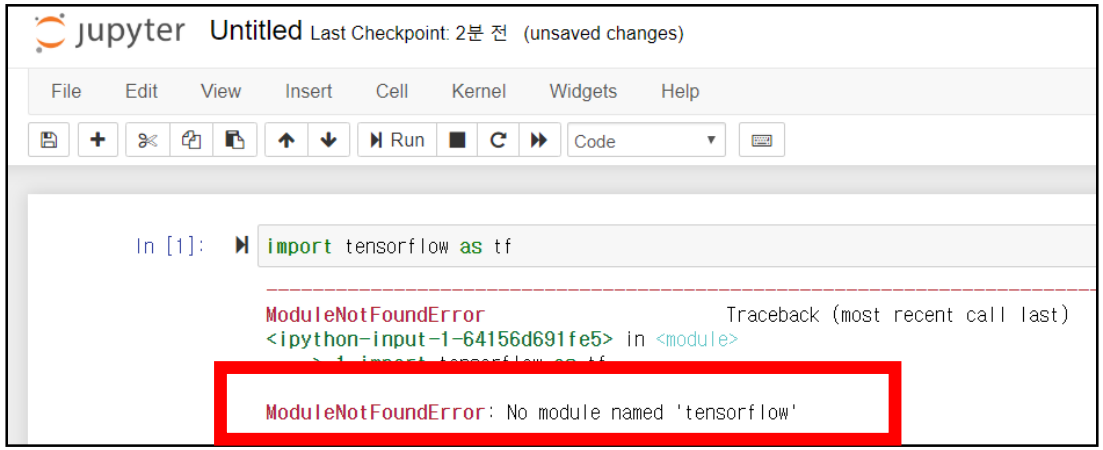
Jupyter Notebook

- Here, you can check whether the TensorFlow is installed or not
  - Put the command: “import tensorflow as tf”
  - Push the “Run” button
  - If any error message doesn’t show up, TensorFlow has been successfully installed



Jupyter Notebook

- This is one of error messages appeared.



- If this error message shows up, follow the instruction again.
- Or, the other error messages show up, I recommend you search in Google or ask TAs.

Github

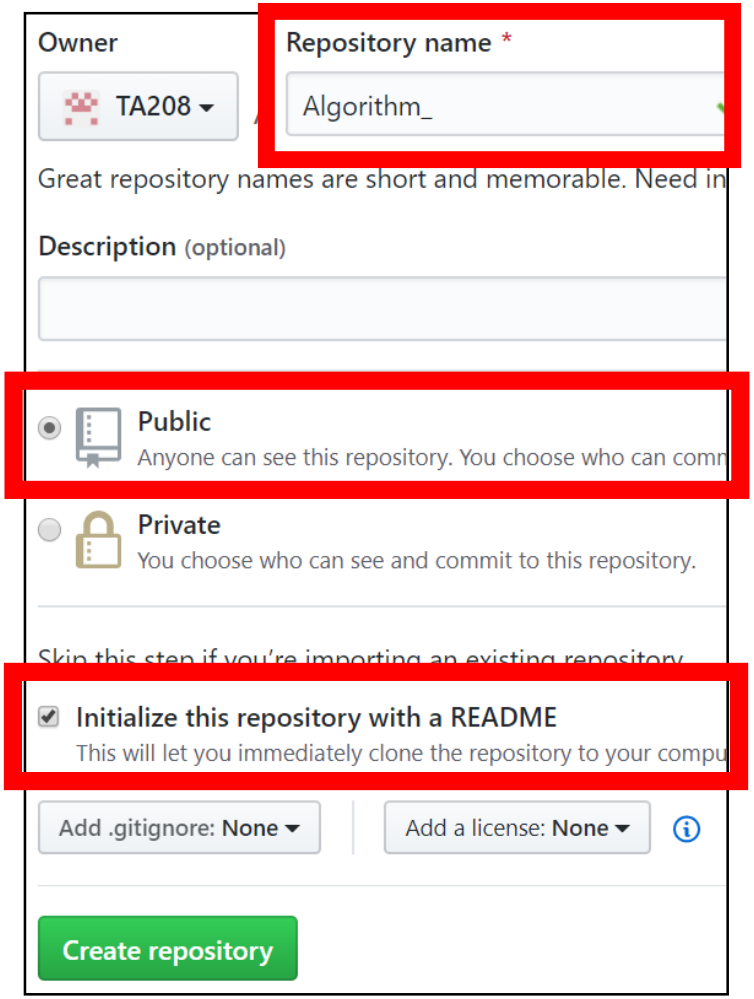
- World's leading software development platform
- https://github.com/

- If you don't have an account on Github, sign up for Github

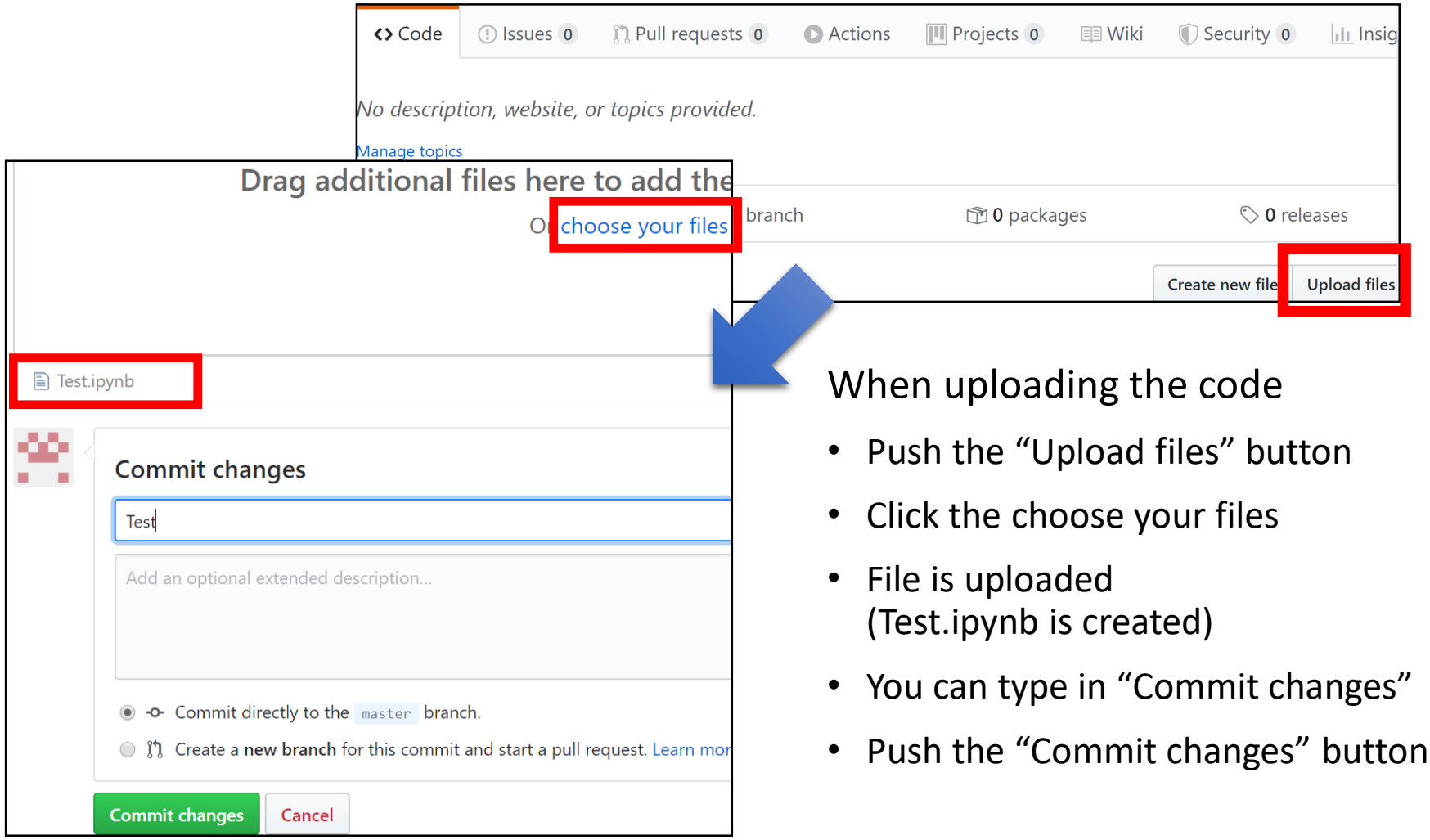
Github

Click the Create a repository

1. Type the repository name
2. Repository name:
   - If you are in class-03: Algorithm_03
   - If you are in class-04: Algorithm_04
3. You create the repository with “Public” in order to upload your assignment
4. Check “Initialize this repository with a README”
5. Push the “Create repository” button



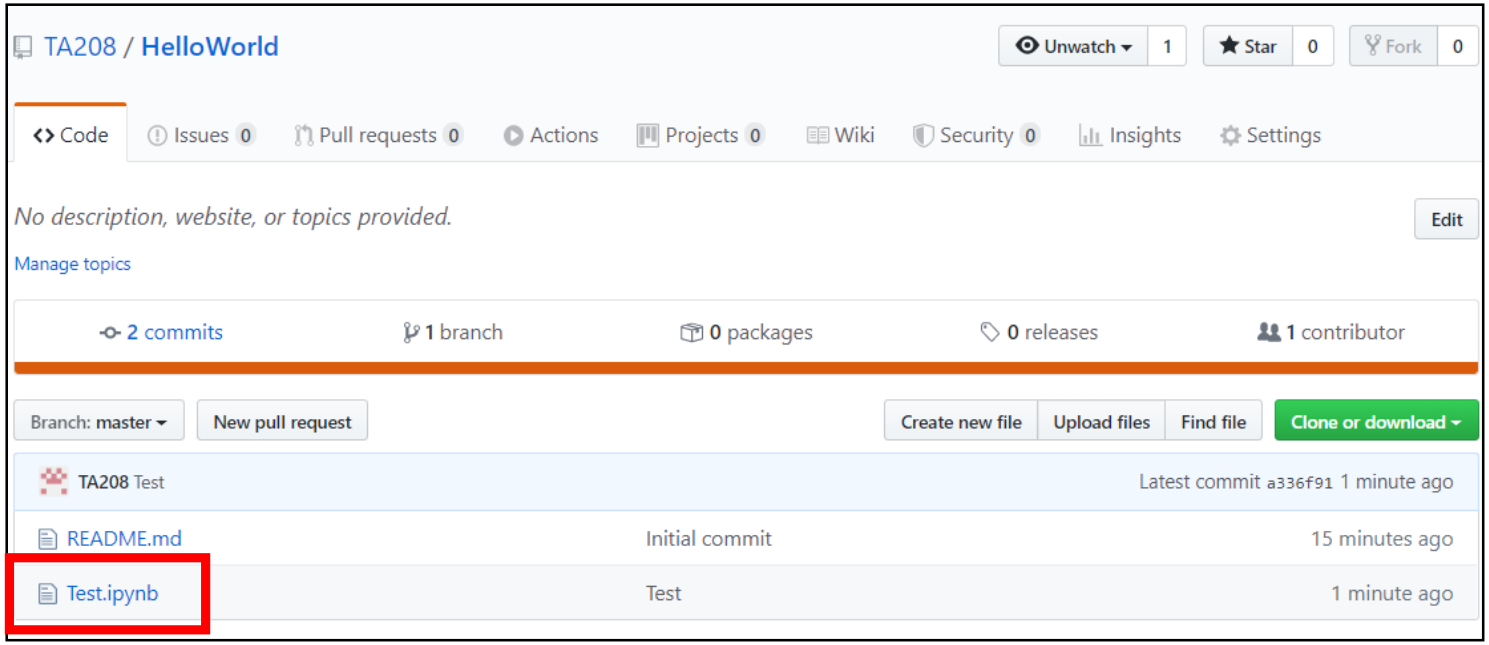
Github



When uploading the code

- Push the “Upload files” button
- Click the choose your files
- File is uploaded
  (Test.ipynb is created)
- You can type in “Commit changes”
- Push the “Commit changes” button

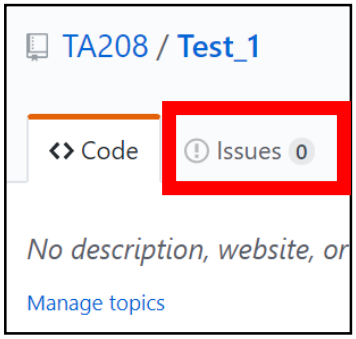
Github


TA208 / HelloWorld
Unwatch 1
Star 0
Fork 0
Code
Issues 0
Pull requests 0
Actions
Projects 0
Wiki
Security 0
Insights
Settings
No description, website, or topics provided.
Edit
Manage topics
2 commits
1 branch
0 packages
0 releases
1 contributor
Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download
TA208 Test
Latest commit a336f91 1 minute ago
README.md
Initial commit
15 minutes ago
Test.ipynb
Test
1 minute ago

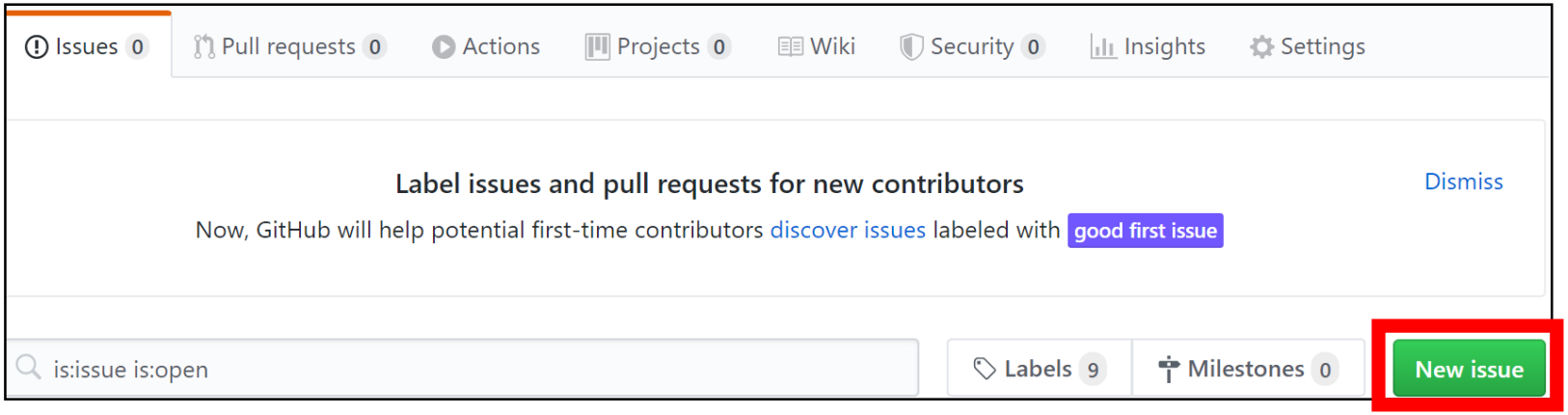
Github – Upload images

Upload images on ReadMe in Github

- Click “Issue”

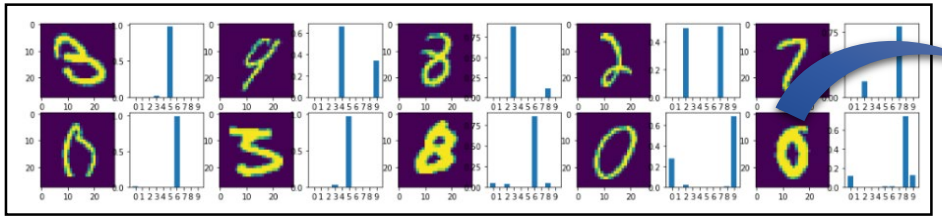


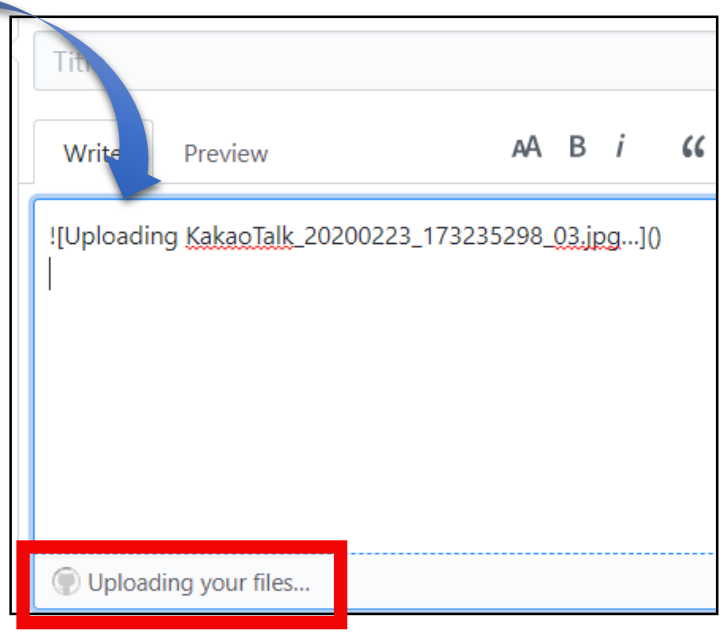
- Push the “New issue” Button



Github – Upload images

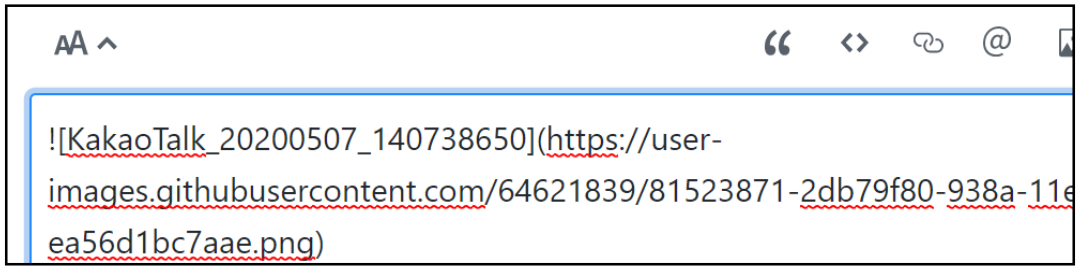
- Drag and drop your image to the blank
- Wait until the “Uploading your files…” message disappears



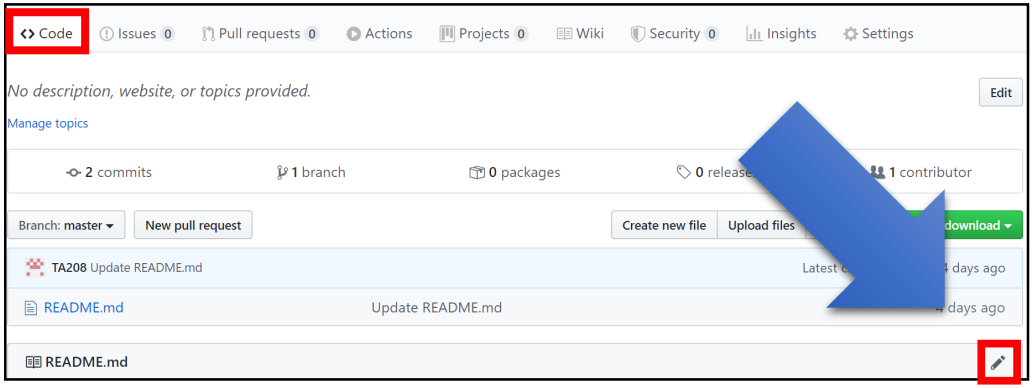


Github – Upload images

- New sentence is created automatically (see below)
- Copy the whole sentence



- Go back to the “Code” section
- Click the pencil image



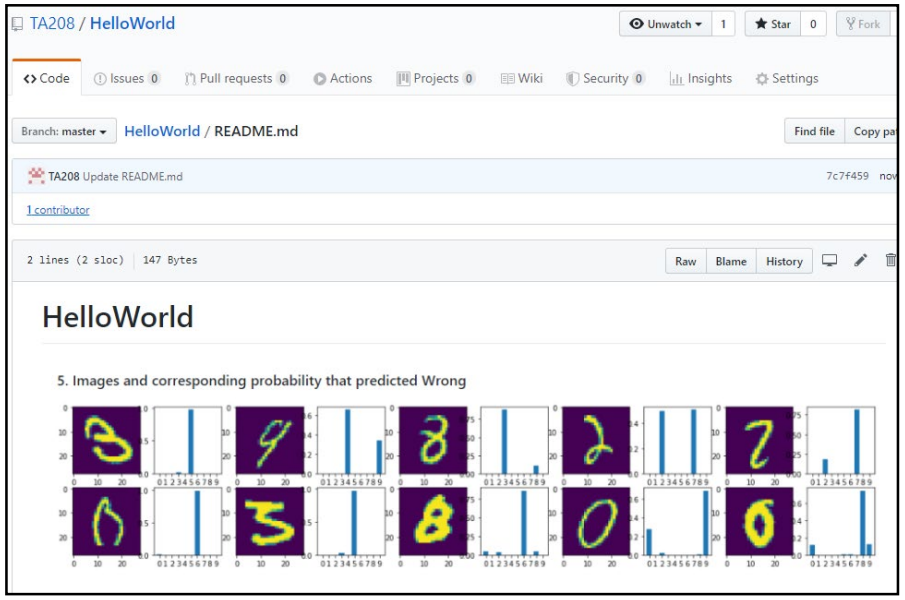
Github – Upload images

- Paste what you've copied

Edit file | Preview changes

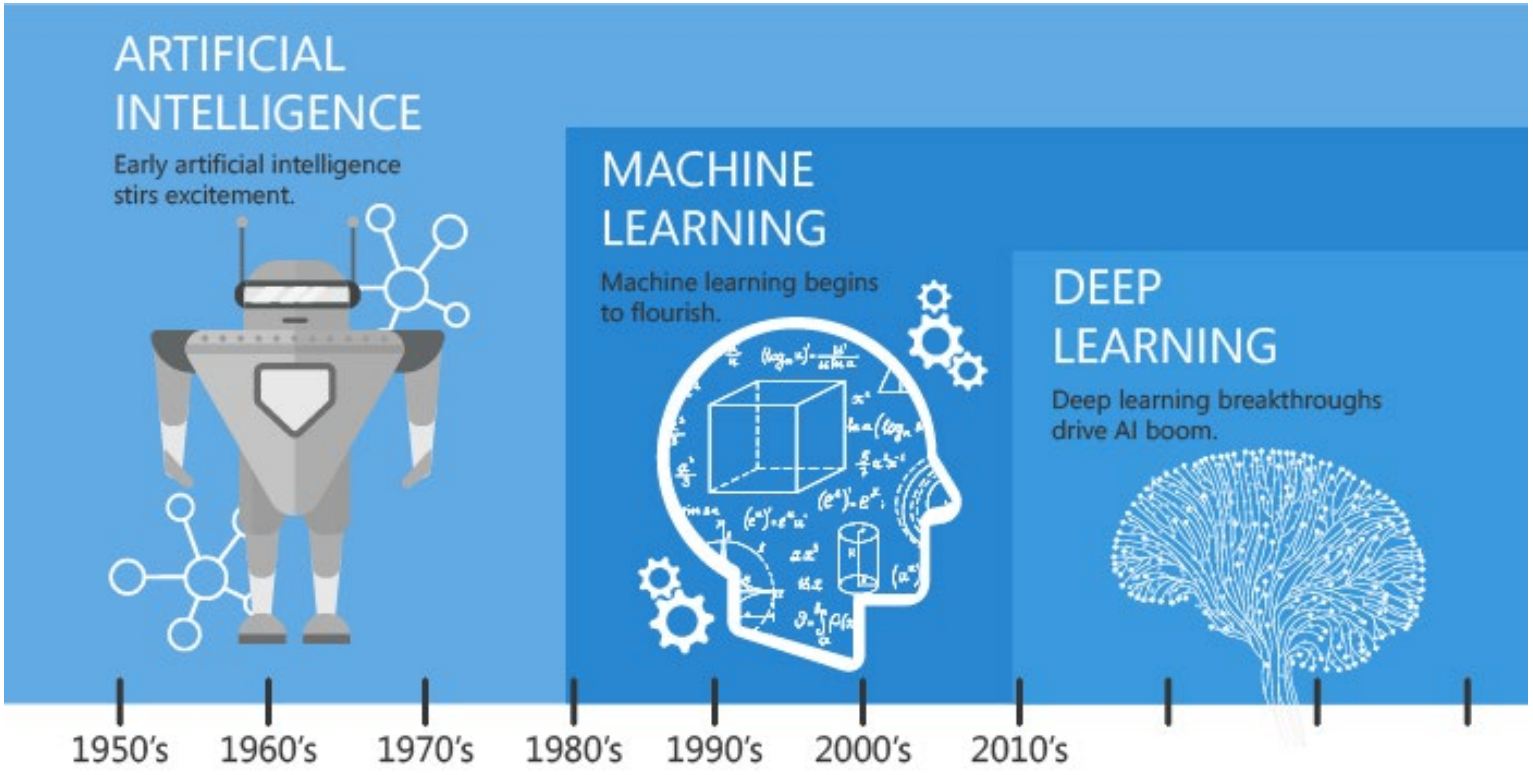
```
1  # HelloWorld
2  ![KakaoTalk_20200507_140738650](https://user-images.githubusercontent.com/64621839/8152387
```

- You will get the following one in your "README.md" page



Introduction to Deep Learning & Classification Problems

Deep Learning as an AI approach



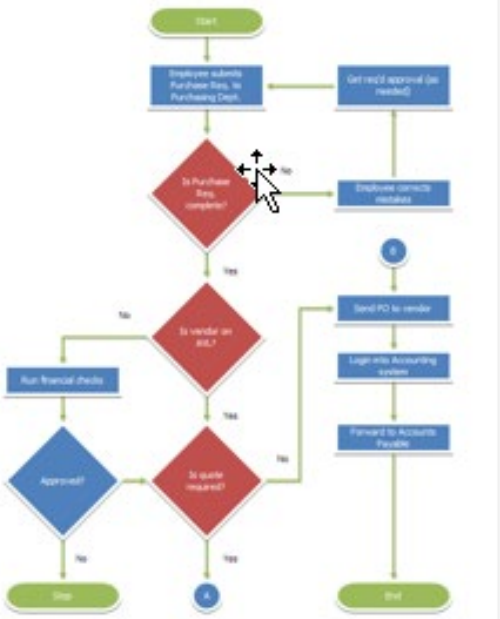
Since an early flush of optimism in the 1950's, smaller subsets of artificial intelligence - first machine learning, then deep learning, a subset of machine learning - have created ever larger disruptions.

Image Source: Medium

Deep Learning as an AI approach

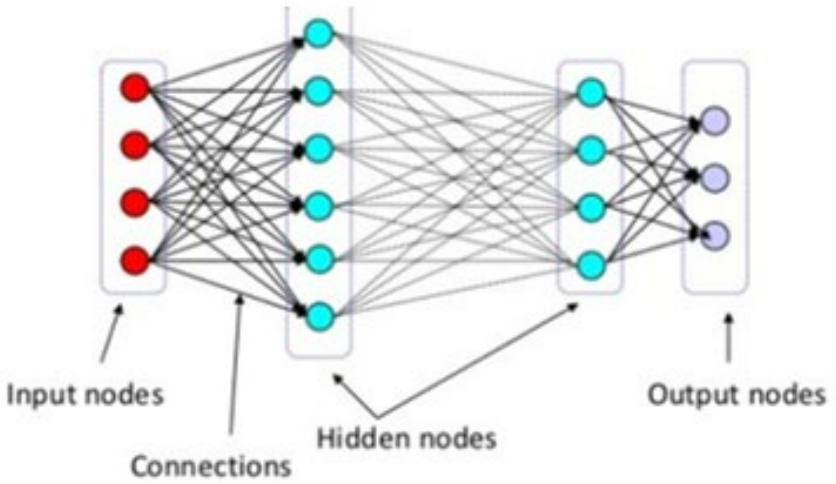
First Wave

Traditional Programming



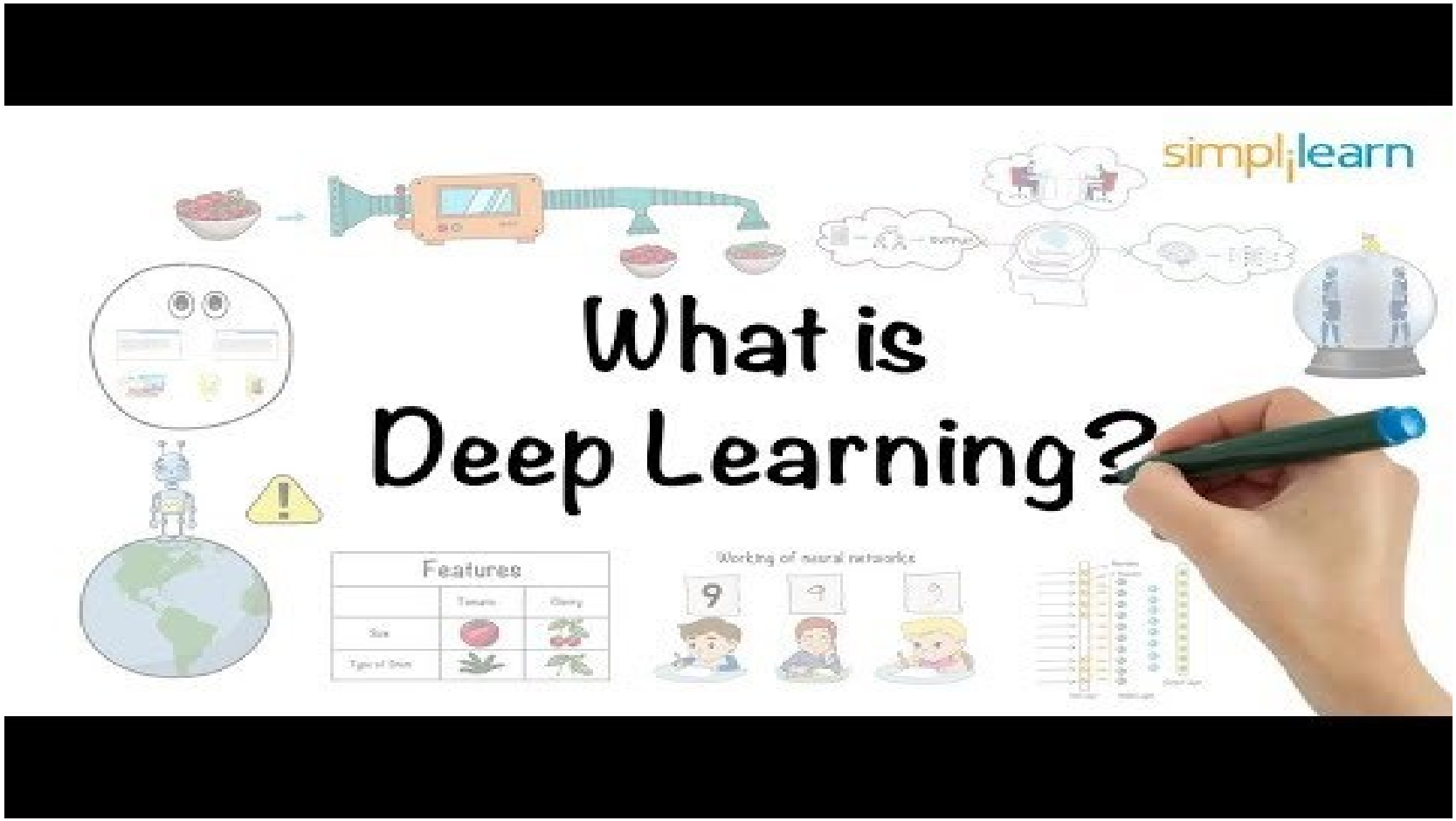
Second Wave

Neural Nets – Deep Learning



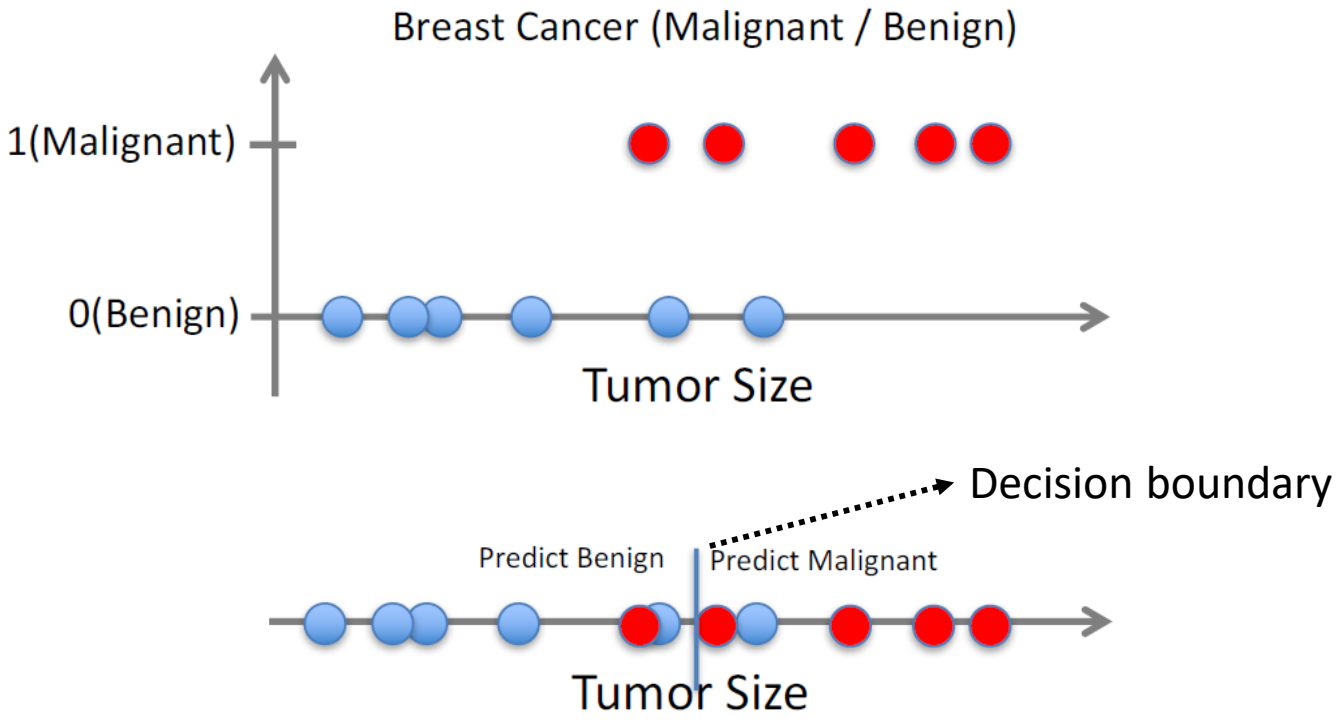
Deep Learning in 5 Minutes

- Link: https://www.youtube.com/watch?v=6M5VXKLf4D4



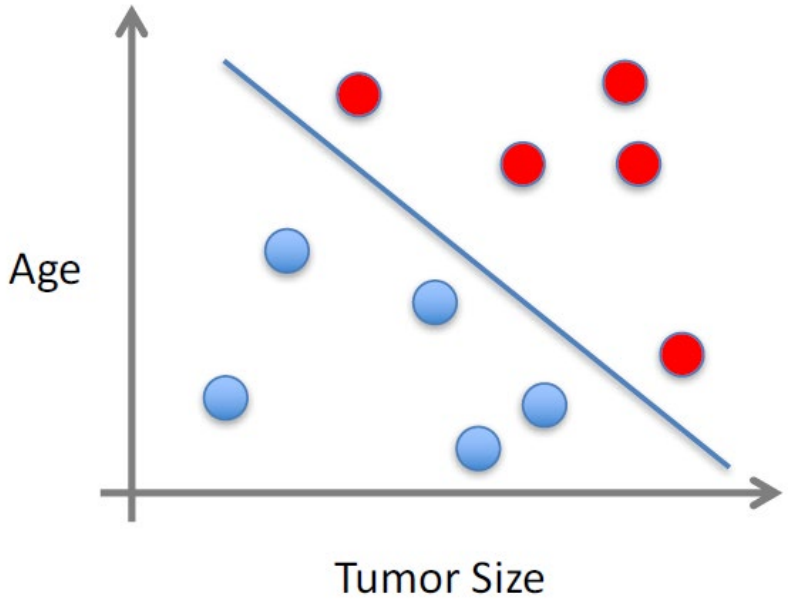
Supervised learning: Classification

- Given $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$
- Learn a function $f(x)$ to predict y given x
  - If y is **categorical**, then the problem is a **classification** problem



Supervised learning: Classification

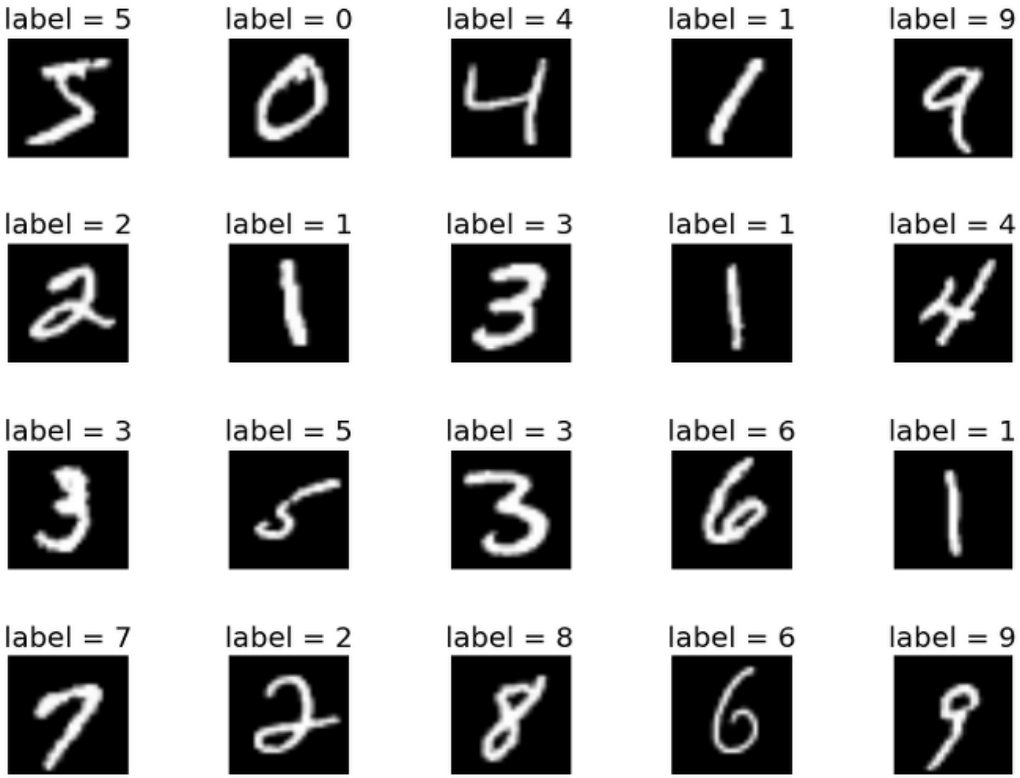
- x can be multi-dimensional
  - Each dimension corresponds to an attribute



- Clump Thickness
- Uniformity of Cell Size
- Uniformity of Cell Shape
⋮

Handwritten digit classification

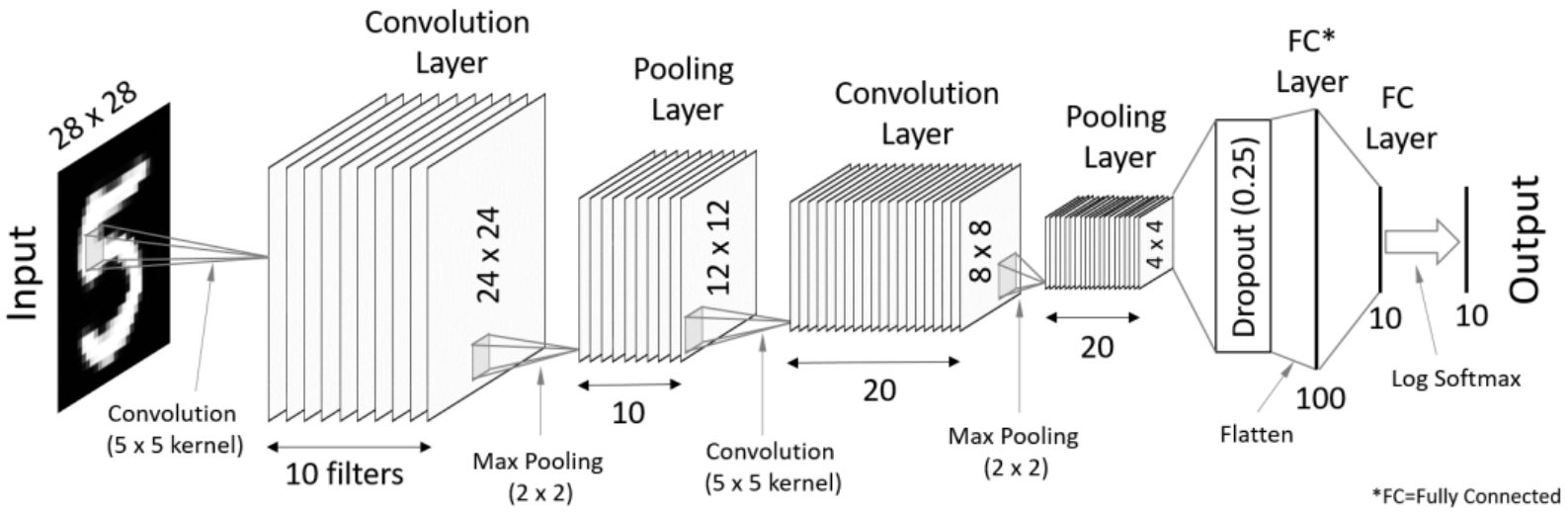
- MNIST dataset



http://yann.lecun.com/exdb/mnist/

Handwritten digit classification

- 10-class classification on the MNIST dataset using deep learning



https://codetolight.wordpress.com/2017/11/29/getting-started-with-pytorch-for-deep-learning-part-3-neural-network-basics/

Output: 10-dim vector giving probability among the classes (0-9) for the input

Project Description

Project description

1. Source code (download .ipynb file in the below page)
   - https://github.com/Jin0316/Algorithm
   - Collecting dataset is included in the code

Project description

2. Import library
   - tensorflow : for deep learning
   - numpy: for mathematical computations in Python
   - matplotlib.pyplot : for drawing images

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, models
import numpy as np
import matplotlib.pyplot as plt
```

Project description

3. Load the MNIST dataset
   - There is a built-in function named ‘load_data’ in keras.datasets.mnist
   - Split the dataset into train and test sets
   - Each set consists of images (28x28 size) and labels (classes)

```
mnist = keras.datasets.mnist
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
```

Project description

4. Split into train/test datasets
   - Change data shape to 60,000 x 28 x 28 x 1
   - 60,000 : Total number of images
   - 28 x 28 x 1 : Height, Width, # Channels of an image

```
train_images = tf.reshape(train_images, [-1, 28, 28, 1])
test_images = tf.reshape(test_images, [-1, 28, 28, 1])
```

Project description

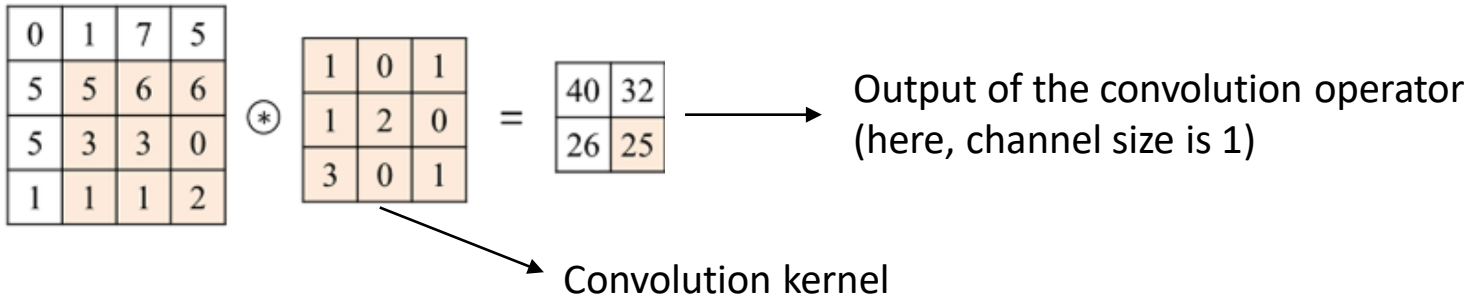
5. Model (three models whose # layers are 3, 5, and 7)
   - Easiest way to build a model is to use function “Sequential”
   - You run the models one by one by the hyperparameter “model-number”

```
def select_model(model_number):
    if model_number == 1:
        model = keras.models.Sequential([
                    keras.layers.Conv2D(32, (3,3), activation = 'relu', input_shape = (28, 28,1)),  # Layer 1
                    keras.layers.MaxPool2D((2,2)),                                                  # Layer 2
                    keras.layers.Flatten(),
                    keras.layers.Dense(10, activation = 'softmax')])                                # Layer 3

    if model_number == 2:
        model = keras.models.Sequential([
                    keras.layers.Conv2D(32, (3,3), activation = 'relu', input_shape=(28,28,1)),     # Layer 1
                    keras.layers.MaxPool2D((2,2)),                                                  # Layer 2
                    keras.layers.Conv2D(64, (3,3), activation = 'relu'),                            # Layer 3
                    keras.layers.MaxPool2D((2,2)),                                                  # Layer 4
                    keras.layers.Flatten(),
                    keras.layers.Dense(10, activation = 'softmax')])                                # Layer 5

    if model_number == 3:
        model = keras.models.Sequential([
                    keras.layers.Conv2D(32, (3,3), activation = 'relu', input_shape = (28, 28,1)),  # Layer 1
                    keras.layers.MaxPool2D((2,2)),                                                  # Layer 2
                    keras.layers.Conv2D(64, (3,3), activation = 'relu'),                            # Layer 3
                    keras.layers.Conv2D(64, (3,3), activation = 'relu'),                            # Layer 4
                    keras.layers.MaxPool2D((2,2)),                                                  # Layer 5
                    keras.layers.Conv2D(128, (3,3), activation = 'relu'),                           # Layer 6
                    keras.layers.Flatten(),
                    keras.layers.Dense(10, activation = 'softmax')])                                # Layer 7

    return model
```

Project description

Note: Model – Convolution layer

- Convolution operator



- Example of a convolution layer

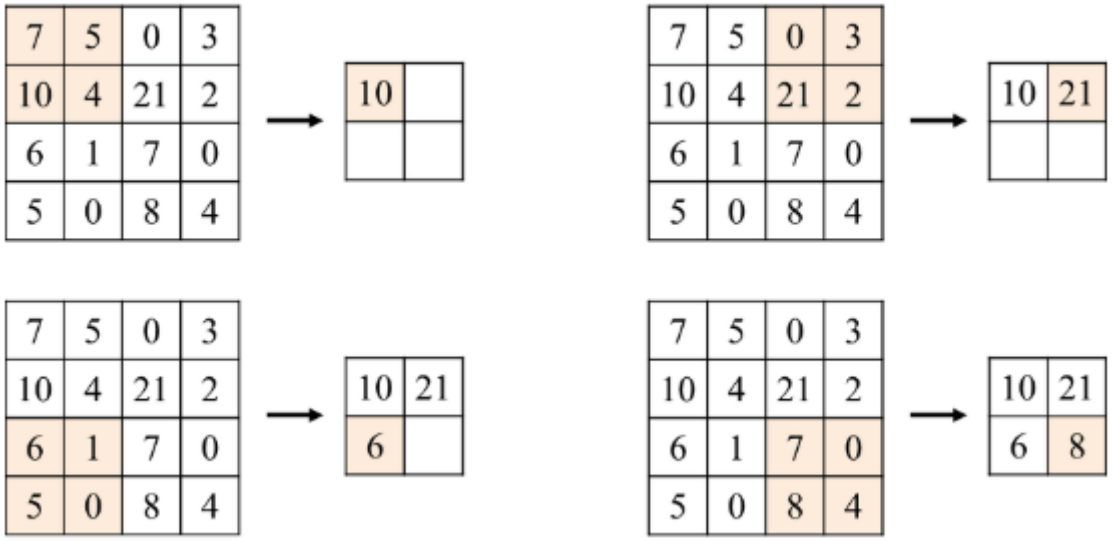
```
keras.layers.Conv2D(32, (3,3), activation = 'relu', input_shape = (28, 28,1))
```

  - Output channel size: 32
  - Filter size: 3 x 3
  - Output will go through the “ReLU” activation function

Project description

Note: Model – Maxpool layer

- Max pooling layer also has a filter
- In the example below, the size of filter is 2 x 2
- The layer extracts the maximum value of input for the given filter

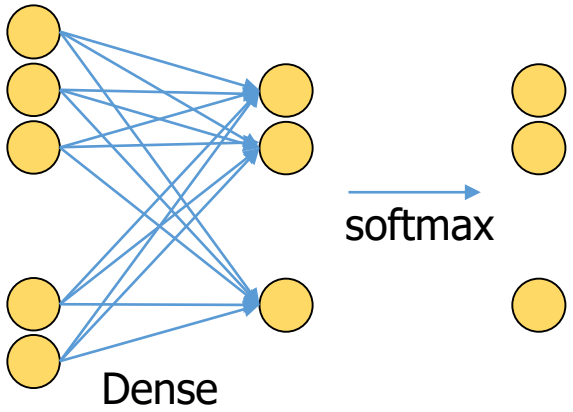


Project description

Note: Model – Dense (fully-connected) layer

- The flatten function should be performed before the dense layer
- Flatten : change the image to 1D counterpart
  - ex) 5 x 5 image => 1 x 25 or 25 x 1
- Dense : output of dense layer always has to be 10 (for MNIST, 0-9)
- Output of the dense layer will pass through the “Softmax” function

```
keras.layers.Flatten(),
keras.layers.Dense(10, activation = 'softmax')])
```



Project description

6. Train the model
   - Use the built-in function “compile”
   - Before training you choose an optimizer and a loss function
   - There are many optimizers and loss functions to choose

```
model.compile(
    optimizer = 'adam',
    loss = 'sparse_categorical_crossentropy',
    metrics = ['accuracy']
)
```

   - To train the model, use the built-in function “fit”
   - Epoch can be set differently according to problems

```
model.fit(train_images, train_labels,  epochs = 5)
```

```
Train on 60000 samples
Epoch 1/5
60000/60000 [==============================] - 31s 512us/sample - loss: 0.7183 - accuracy: 0.9410
Epoch 2/5
60000/60000 [==============================] - 29s 489us/sample - loss: 0.0890 - accuracy: 0.9742
Epoch 3/5
60000/60000 [==============================] - 29s 486us/sample - loss: 0.0739 - accuracy: 0.9782
Epoch 4/5
60000/60000 [==============================] - 30s 493us/sample - loss: 0.0664 - accuracy: 0.9798
```

Project description

7. Test the model for the test dataset
   - Use the built-in function “evaluate”

```
test_loss, accuracy = model.evaluate(test_images, test_labels, verbose = 2)
print('\nTest loss : ', test_loss)
print('Test accuracy :', accuracy)
```

```
10000/1 - 2s - loss: 0.0582 - accuracy: 0.9712

Test loss :  0.11443363445440191
Test accuracy : 0.9712
```

- For more details on optimizers and loss functions, see below:
  - https://towardsdatascience.com/common-loss-functions-in-machine-learning-46af0ffc4d23
  - https://ruder.io/optimizing-gradient-descent/

Project description

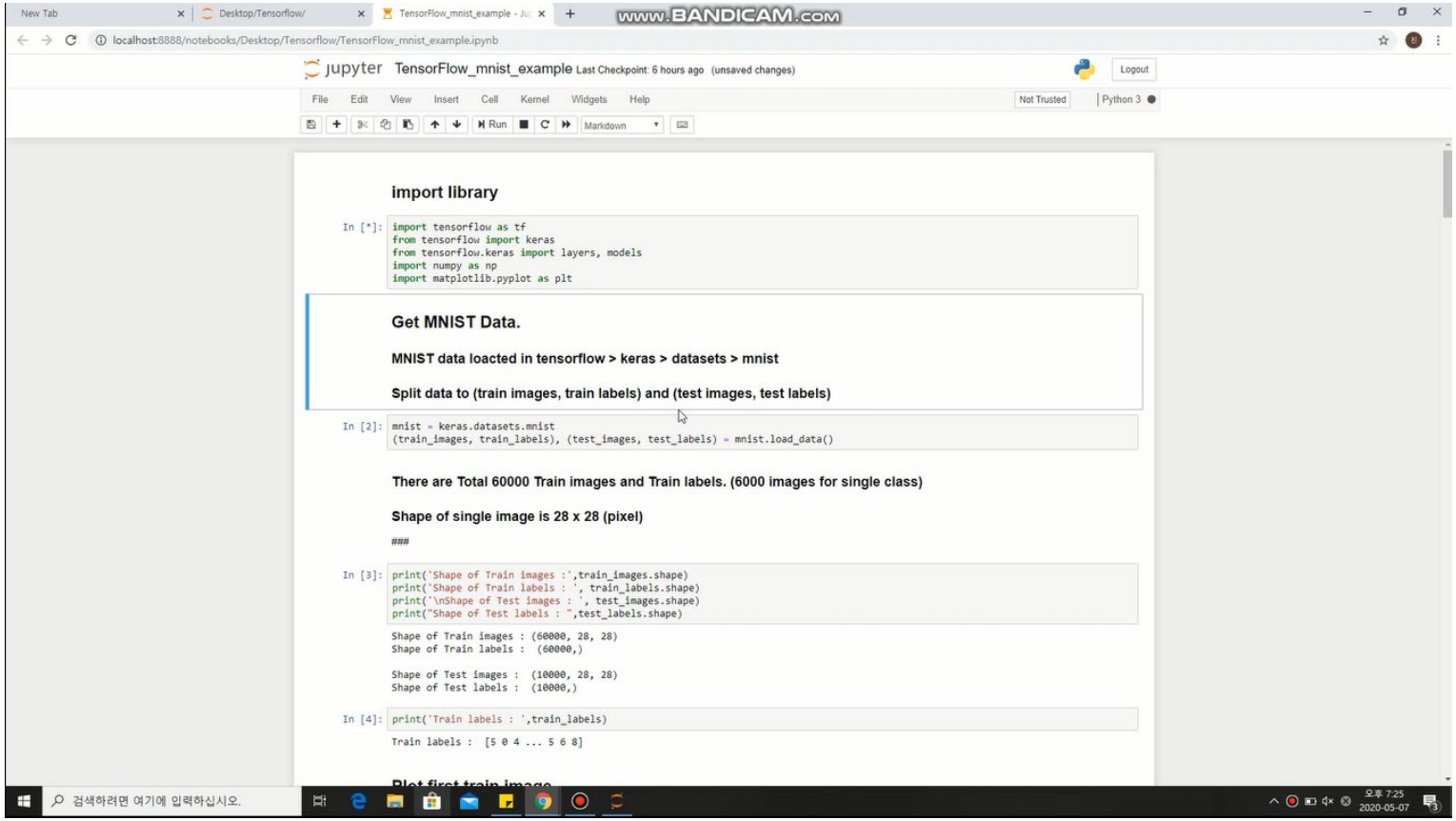
- Your task
  - Create your Github page.
  - Run the given code (three times) for **three** different CNN models .
    - Train/Test the CNN models for the MNIST dataset and get test accuracy.
  - After finishing the tasks, upload the codes you've performed and post your results on your Github page.
  - + Upload success/failure images of your results on the Github page.

- Evaluation is based on the codes and the results on your Github page.

- Check our Github out to get some ideas how to upload codes & results:
  - https://github.com/Jin0316/Algorithm

Demo video

- Demonstration for the project (the video will be uploaded in the e-class)



If you have any questions on the project, please email TA.

(03 class : cjpark137@cau.ac.kr, 04 class : popo1013@cau.ac.kr)