# AOI02-Aidea AOI Project

Hsueh-Ting Chu

# 朱學亭老師

- 現職：亞洲大學資工系副教授(兼AI研究組長)


- EMAIL: htchu.taiwan@gmail.com
- FB: https://www.facebook.com/htchu.taiwan

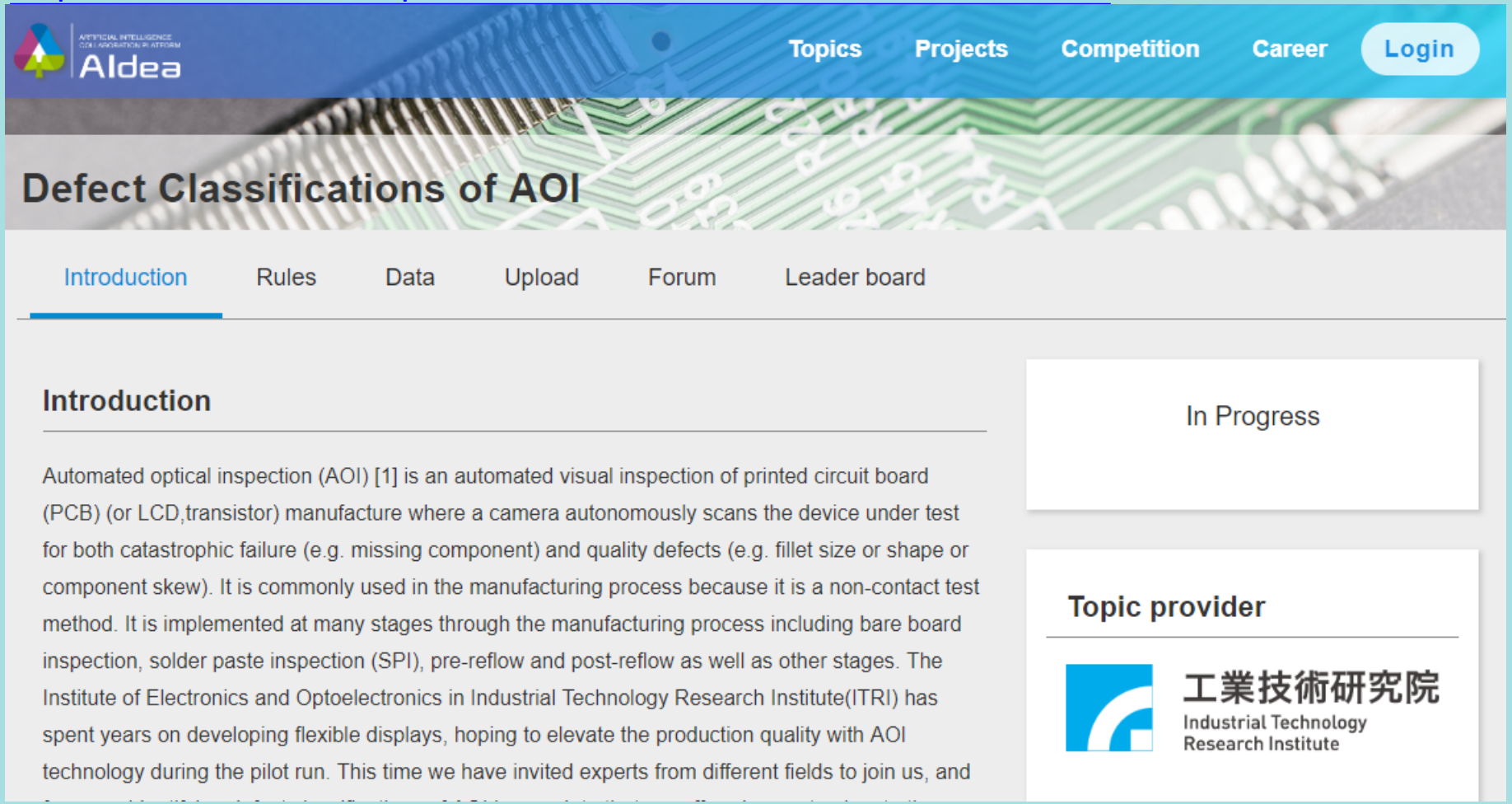

# Aidea platform

ARTIFICIAL INTELLIGENCE
COLLABORATION PLATFORM
Aidea

- Artificial Intelligence Collaboration Platform by ITRI(Industrial Technology Research Institute)
- Solving real AI industrial issues in Taiwan
- Building AI industrial datasets in Taiwan

# Open Topic: Defect Classifications of AOI

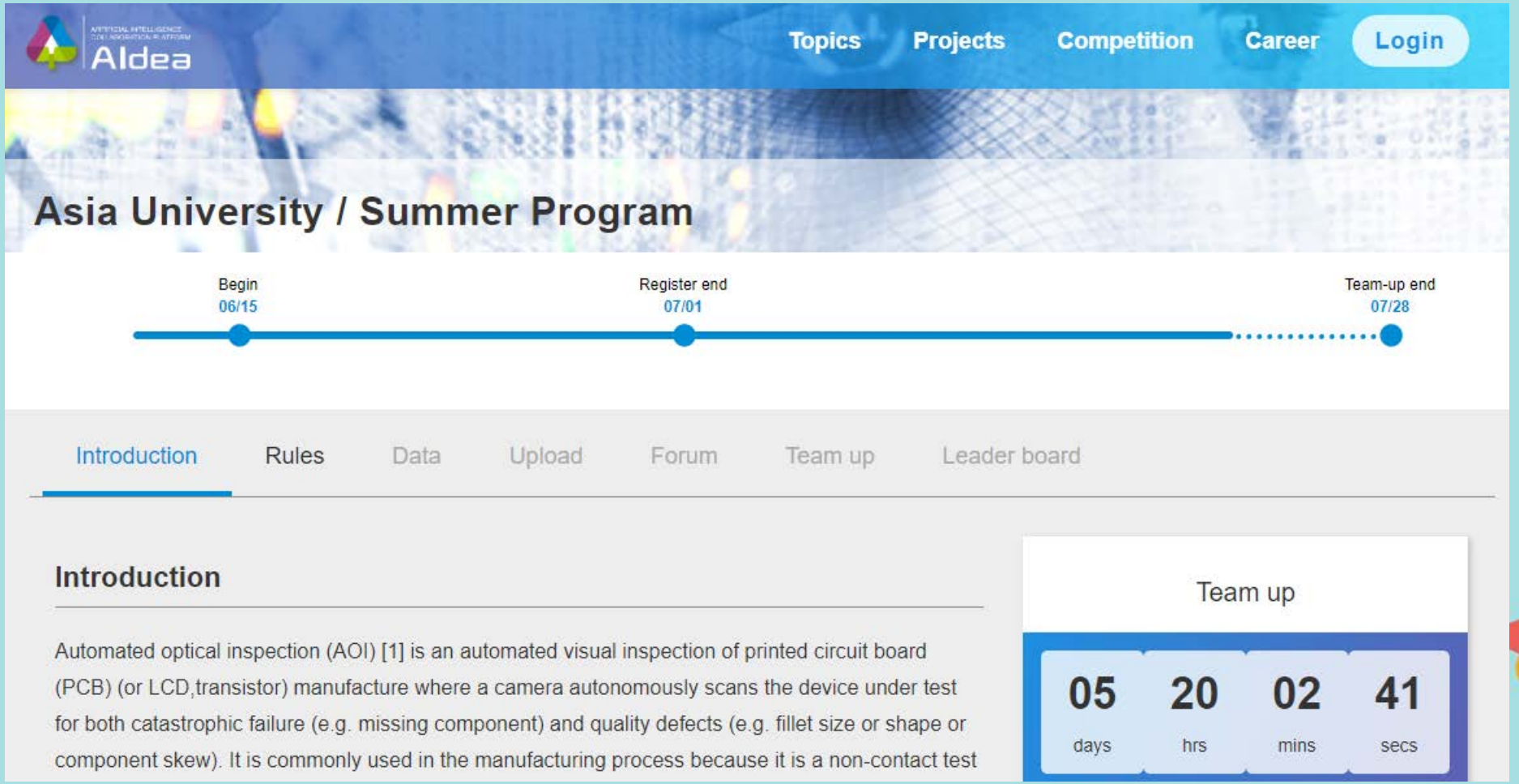
https://aidea-web.tw/topic/a49e3f76-69c9-4a4a-bcfc-c882840b3f27






The top three exceeding Baseline score (**0.998521**) on Private Leaderboard would be awarded

# Closed Project: Defect Classifications of AOI



# AOI Study

- Same dataset:
  - Open Topic: Defect Classifications of AOI
  - Closed Project: Defect Classifications of AOI

- Dataset is available from the Open Topic.
- Closed Project is for ranking.

# AOI data discription

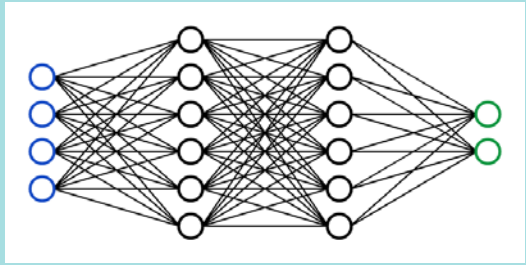
There are 6 categories included in image data that the issue offers
(1 normal category + 5 defect categories)

The download data file (aoi_data.zip) includes:

- train_images: image data for training (PNG format), 2,528 images in total.
- train.csv: includes 2 columns, ID and Label.
  - ID: the image filename
  - Label: defect classification category
    (0: normal, 1: void, 2: horizontal defect, 3: vertical defect, 4: edge defect, 5: particle)
- test_images: image data for testing (PNG format), 10,142 images in total.
- test.csv: includes 2 columns, ID and Label.
  - ID: the image filename
  - Label: Nan

# AOI Workflow





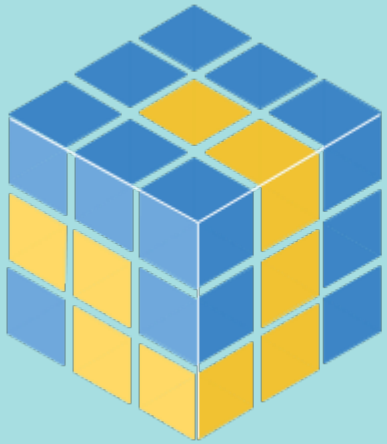

(A) Setup TF 2.0 (B) Mounting (optional) (C) Input training data (D) Model training and inference (E) Output test result

# Colab api

- **Magic commands**
  - %tensorflow_version 2.x
  - %matplotlib inline

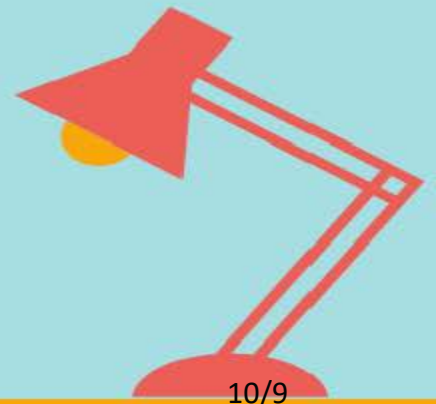
- **Google Colab API**
  - **from google.colab import drive**
  - **drive.mount('/content/drive')**

# NumPy



**NumPy**

- import numpy as np
- a = np.array([2,3,4])
- b = np.array([(1.5,2,3), (4,5,6)])
- c = np.zeros( (3,4) )
- d = np.ones(3, dtype=np.int32)

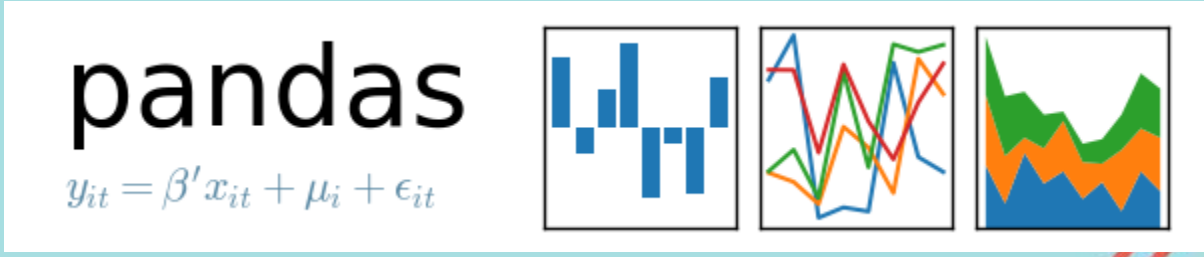


# numpy.argmax

- Returns the indices of the maximum values along an axis
- >>> a = np.arange(6).reshape(2,3) + 10
- >>> a
- array([[10, 11, 12],
- [13, 14, 15]])
- >>> np.argmax(a)
- 5
- >>> np.argmax(a, axis=0)
- array([1, 1, 1])
- >>> np.argmax(a, axis=1)
- array([2, 2])

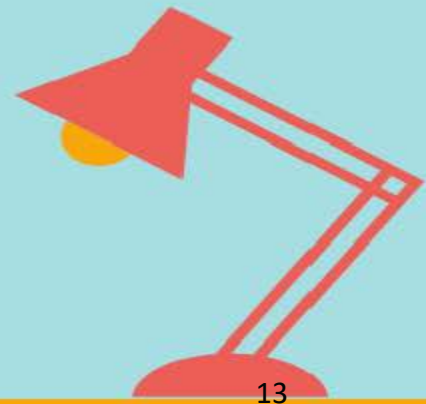
# Pandas

- Python Data Analysis Library
- import pandas as pd
- df_train = pd.read_csv("train.csv")
- df_out.to_csv( "A.csv", index=False)



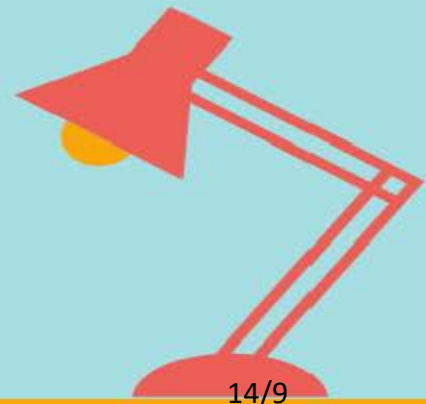
# Data Structures

- List ⇔ NumPy array
- List ⇔ Pandas DataFrame
- NumPy array ⇔ Pandas DataFrame



# Python Data Visualization Libraries

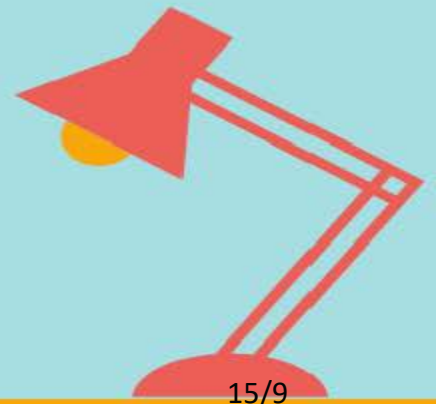
- Matplotlib
- Seaborn
- pandas.DataFrame.plot



# Matplotlib

- Matplotlib is a Python 2D plotting library

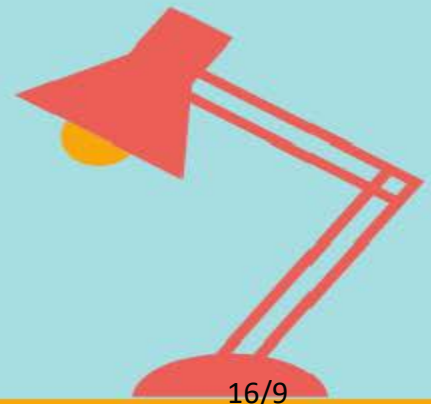
- import matplotlib.pyplot as plt
- fig, ax = plt.subplots()
- ax.plot(x, y)
- ax.set_xlim(0, 10)
- ax.set_ylim(-1, 1)
- plt.show()



# Seaborn heatmap

- seaborn: statistical data visualization

- import numpy as np; np.random.seed(0)
- import seaborn as sns;
- sns.set()
- uniform_data = np.random.rand(10, 12)
- ax = sns.heatmap(uniform_data)



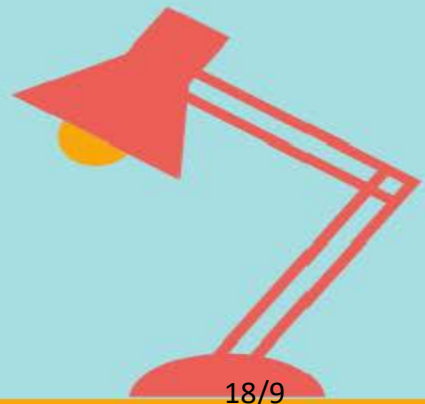
# Python  Image Libraries

- PIL: Python Imaging Library
- Pillow: the friendly PIL fork
- scikit-image: Image processing in Python
- OpenCV: Open source computer vision
- tf.keras.preprocessing.image

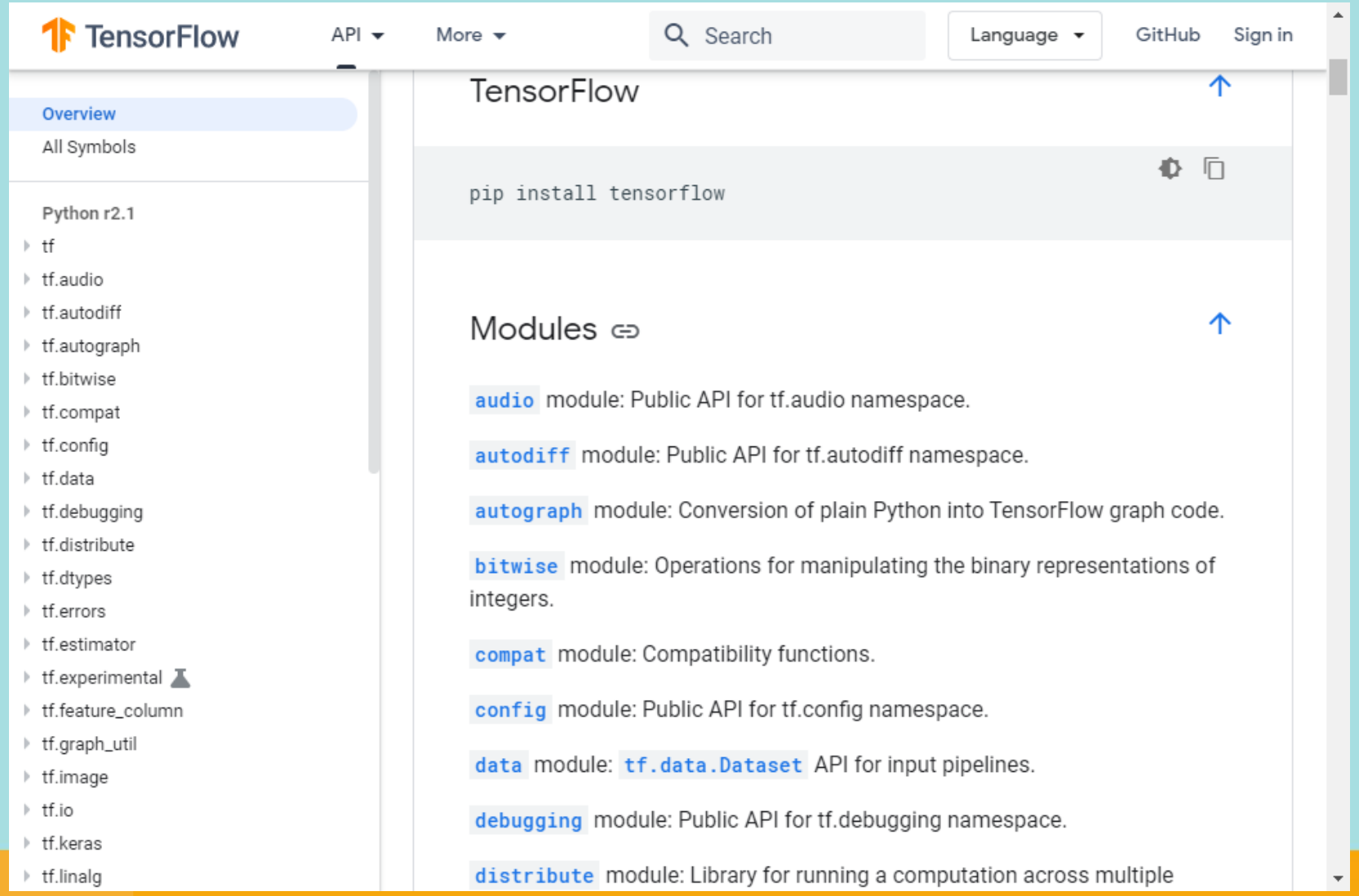
# scikit-learn

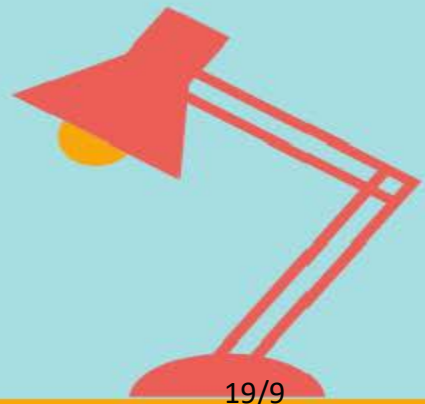

- scikit-learn: machine learning in Python

- from sklearn.metrics import confusion_matrix
- from sklearn.model_selection import train_test_split
- X_train, X_test, y_train, y_test = train_test_split(
- …  X, y, test_size=0.33, random_state=42)



# TensorFlow 2.x



# TensorFlow Tutorials

## For beginners

The best place to start is with the user-friendly Sequential API. You can create models by plugging together building blocks. Run the "Hello World" example below, then visit the **tutorials** to learn more.

To learn ML, check out our **education page**. Begin with curated curriculums to improve your skills in foundational ML areas.

```
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)
```

## For experts

The Subclassing API provides a define-by-run interface for advanced research. Create a class for your model, then write the forward pass imperatively. Easily author custom layers, activations, and training loops. Run the "Hello World" example below, then visit the **tutorials** to learn more.

```
class MyModel(tf.keras.Model):
  def __init__(self):
    super(MyModel, self).__init__()
    self.conv1 = Conv2D(32, 3, activation='relu')
    self.flatten = Flatten()
    self.d1 = Dense(128, activation='relu')
    self.d2 = Dense(10, activation='softmax')

  def call(self, x):
    x = self.conv1(x)
    x = self.flatten(x)
    x = self.d1(x)
    return self.d2(x)
model = MyModel()

with tf.GradientTape() as tape:
  logits = model(images)
  loss_value = loss(logits, labels)
grads = tape.gradient(loss_value, model.trainable_variable
optimizer.apply_gradients(zip(grads, model.trainable_varia
```

# Sample

```
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)
```

Thanks!

# Q&A