

WorkBooks

1

2020. 06. 27.

Mon.

**Prepared by DaeKyeong Kim
Ph.D.**

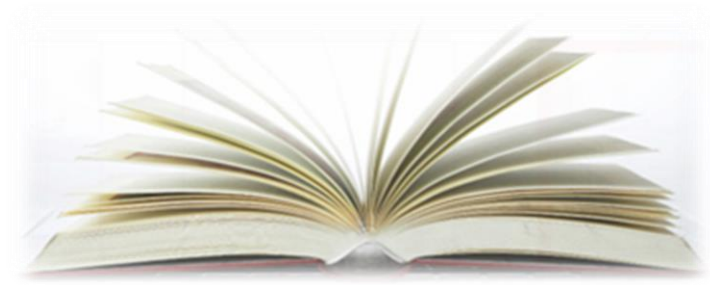


Contents



Section 1	Problem Definition , Data Collection	3
Section 2	Development environment Configuration	11
Section 3	Data Ingestion, Data Acquisition	79
Section 4	Data Preprocessing	79
Section 5	Data Preparation	79
Section 6	Learning	79
Section 7	Evaluation	79

Section 3



Data Ingestion, Data Acquisition

1. 패키지 로드
2. 데이터 셋 가져오기

학습목표



이 워크샵에서는 패키지 로드와 데이터 셋 가져오기에 대해 수행할 수 있습니다.

❖ 패키지 로드와 데이터 셋 가져오기

```

from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree

```

```

from IPython.display import Image

```

```

import pandas as pd
import numpy as np
import pydotplus
import os

```

```

tennis_data = pd.read_csv('playtennis.csv')
tennis_data

```

```

\
Outlook      Temperature  Humidity    Wind      PlayTennis
0      Sunny           Hot         High     Weak      No
1      Sunny           Hot         High     Strong    No
2      Overcast        Hot         High     Weak      Yes
3      Rain            Mild        High     Weak      Yes
4      Rain            Cool       Normal   Weak      Yes
5      Rain            Cool       Normal   Strong    No
...
13     Rain            Mild        High     Strong    No
\

```

파일



..



sample_data



playtennis.csv

Section 4



Data Preprocessing

- 1. Data Scaling**

- 2. Data Collection**

학습목표



이 워크샵에서는 Problem Definition과 Problem Definition , Problem Definition에 대해 수행할 수 있습니다.

❖ 데이터 전처리

```
tennis_data.Outlook = tennis_data.Outlook.replace('Sunny', 0)
tennis_data.Outlook = tennis_data.Outlook.replace('Overcast', 1)
tennis_data.Outlook = tennis_data.Outlook.replace('Rain', 2)
```

```
tennis_data.Temperature = tennis_data.Temperature.replace('Hot', 3)
tennis_data.Temperature = tennis_data.Temperature.replace('Mild', 4)
tennis_data.Temperature = tennis_data.Temperature.replace('Cool', 5)
```

```
tennis_data.Humidity = tennis_data.Humidity.replace('High', 6)
tennis_data.Humidity = tennis_data.Humidity.replace('Normal', 7)
```

```
tennis_data.Wind = tennis_data.Wind.replace('Weak', 8)
tennis_data.Wind = tennis_data.Wind.replace('Strong', 9)
```

```
tennis_data.PlayTennis = tennis_data.PlayTennis.replace('No', 10)
tennis_data.PlayTennis = tennis_data.PlayTennis.replace('Yes', 11)
```

tennis_data

、

Outlook	Temperature	Humidity	Wind	PlayTennis	
0	0	3	6	8	10
1	0	3	6	9	10
2	1	3	6	8	11
3	2	4	6	8	11
...					
13	2	4	6	9	10

、

- 데이터가 0과 1 사이에 위치하도록 스케일링 합니다. (default)
- 최소값 = 0, 최대값 = 1이 되도록 스케일링 합니다.
- $\frac{x - x_{min}}{x_{max} - x_{min}}$ 를 이용하여 변경.
- 데이터의 최소값과 최대값을 알 때 사용합니다.
- 이상치가 존재할 경우 스케일링 결과가 매우 좁은 범위로 압축될 수 있습니다.

```
from sklearn.preprocessing import MinMaxScaler
```

```
# minmax scaler 선언 및 학습
```

```
minmaxScaler = MinMaxScaler().fit(X_train)
```

```
# train셋 내 feature들에 대하여 minmax scaling 수행
```

```
X_train_minmax = minmaxScaler.transform(X_train)
```

```
# test셋 내 feature들에 대하여 minmax scaling 수행
```

```
X_test_minmax = minmaxScaler.transform(X_test)
```

- 데이터의 평균 = 0, 분산 = 1이 되도록, 즉 데이터가 표준 정규 분포(standard normal distribution)를 따르도록 스케일링 합니다.
 - 최솟값과 최댓값의 크기를 제한하지 않는다.
 - $\frac{x - \bar{x}}{\sigma}$
- 데이터의 최소값과 최대값을 모를 때 사용합니다.
- 평균(mean)과 분산(variance)을 사용합니다.
- 모든 feature들이 같은 스케일을 갖게 됩니다.
- 평균과 표준편차가 이상치로부터 영향을 많이 받는다는 점에서 이상치에 민감합니다.

```
from sklearn.preprocessing import StandardScaler
```

```
# standard scaler 선언 및 학습
```

```
standardScaler = StandardScaler().fit(X_train)
```

```
# train셋 내 feature들에 대하여 standard scaling 수행
```

```
X_train_standard = standardScaler.transform(X_train)
```

```
# test셋 내 feature들에 대하여 standard scaling 수행
```

```
X_test_standard = standardScaler.transform(X_test)
```

- 데이터의 중앙값 = 0, IQE = 1이 되도록 스케일링 합니다.
- 중앙값(median)과 IQR(interquartile range)을 사용합니다.
- RobustScaler를 사용할 경우 StandardScaler에 비해 스케일링 결과가 더 넓은 범위로 분포합니다.
- 모든 feature들이 같은 스케일을 갖게 됩니다.
- 이상치의 영향을 최소화 합니다. 중간 값은 정렬시 중간에 있는 값을 의미하고, 사분위값은 1/4, 3/4에 위치한 값을 의미.
 - 전체 데이터와 아주 동떨어진 데이터 포인트(이상치)에 영향을 받지 않는다.
 - $\frac{x - q_2}{q_3 - q_1}$ 각각이 사분위값과 중간 값.

```
from sklearn.preprocessing import RobustScaler
```

```
# RobustScaler 선언 및 학습
```

```
robustScaler = RobustScaler().fit(X_train)
```

```
# train셋 내 feature들에 대하여 robust scaling 수행
```

```
X_train_robust = robustScaler.transform(X_train)
```

```
# test셋 내 feature들에 대하여 robust scaling 수행
```

```
X_test_robust = robustScaler.transform(X_test)
```

- 데이터가 -1과 1 사이에 위치하도록 스케일링 합니다.
- 절대값의 최소값 = 0, 절대값의 최대값 = 1이 되도록 스케일링 합니다.
- 데이터의 값이 양수만 존재할 경우 MinMaxScaler와 유사하게 동작합니다.
- 이상치가 큰 쪽에 존재할 경우 이에 민감할 수 있습니다.

```
from sklearn.preprocessing import MaxAbsScaler
```

```
# MaxAbsScaler 선언 및 학습
```

```
maxabsScaler = MaxAbsScaler().fit(X_train)
```

```
# train셋 내 feature들에 대하여 maxabs scaling 수행
```

```
X_train_maxabs = maxabsScaler.transform(X_train)
```

```
# test셋 내 feature들에 대하여 maxabs scaling 수행
```

```
X_test_maxabs = maxabsScaler.transform(X_test)
```

- 앞의 4가지 스케일러는 각 특성(열)의 통계치를 이용하여 진행됩니다.
- 그러나 Normalizer 의 경우 각 샘플(행)마다 적용되는 방식입니다.
- 이는 한 행의 모든 특성들 사이의 유클리드 거리(L2 norm)가 1이 되도록 스케일링합니다.
 - 즉 길이가 1인 원 또는 구로 투영하는 것이고, 각도만이 중요할 때 적용.
 - l1, l2, max 옵션을 제공하며 유클리디안 거리인 l2가 기본값.
- 일반적인 데이터 전처리의 상황에서 사용되는 것이 아니라
- 모델(특히나 딥러닝) 내 학습 벡터에 적용하며,
- 특히나 피쳐들이 다른 단위(키, 나이, 소득 등)라면 더욱 사용하지 않습니다.

```
from sklearn.preprocessing import Normalizer
```

```
# 변형 객체 생성
normal_scaler = Normalizer()
# 훈련데이터의 모수 분포 저장
normal_scaler.fit(X_train)
# 훈련 데이터 스케일링
X_train_scaled = normal_scaler.transform(X_train)
# 테스트 데이터의 스케일링
X_test_scaled = normal_scaler.transform(X_test)
# 스케일링 된 결과 값으로 본래 값을 구할 수도 있다.
# X_origin = normal_scaler.inverse_transform(X_train_scaled)
```

```
import pandas as pd
df = pd.DataFrame({
    'category':
    ['fruits','fruits','fruits','fruits','fruits','vegetables','vegetables','vegetables','vegetables','vegetables'],
    'product' : ['apple','orange','durian','coconut','grape','cabbage','carrot','spinach','grass','potato'],
    'sales' : [10,20,30,40,100,10,30,50,60,100]
})
df.head()
```

```
from sklearn import preprocessing
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df_fruits = df[df['category'] == "fruits"]
df_fruits['sales'] = scaler.fit_transform(df_fruits[['sales']])
df_fruits.head()
```

```
\
category    product    sales
0          fruits    apple    0.000000
1          fruits    orange    0.111111
2          fruits    durian    0.222222
3          fruits    coconut    0.333333
fruits      grape    1.000000
\
```

Section 5



Data Preparation

- 1. Problem Definition**
- 2. Data Collection**

학습목표



이 워크샵에서는 Problem Definition과 Problem Definition , Problem Definition에 대해 수행할 수 있습니다.


```
# k-폴드 교차검증을 만듭니다.  
kf = KFold(n_splits=10, shuffle=True, random_state=1)  
  
# 라이브러리를 임포트합니다.  
from sklearn.model_selection import train_test_split  
  
# 훈련 세트와 테스트 세트를 만듭니다.  
features_train, features_test, target_train, target_test = train_test_split(  
    features, target, test_size=0.1, random_state=1)  
  
# 표준화 객체를 만듭니다.  
standardizer = StandardScaler()  
  
# 훈련 세트로 standardizer의 fit 메서드를 호출합니다.  
standardizer.fit(features_train)
```

```
from sklearn.model_selection import ShuffleSplit

# ShuffleSplit 분할기를 만듭니다.
ss = ShuffleSplit(n_splits=10, train_size=0.5, test_size=0.2, random_state=42)

# 교차검증을 수행합니다.
cv_results = cross_val_score(pipeline, # 파이프라인
                              features, # 특성 행렬
                              target, # 타깃 벡터
                              cv=ss, # 교차 검증 기법
                              scoring="accuracy", # 평가 지표
                              n_jobs=-1) # 모든 CPU 코어 사용
```

❖ RepeatedKFold를 사용해 교차검증을 반복 실행. n_splits 기본값 5, n_repeats 기본값은 10

```
from sklearn.model_selection import RepeatedKFold

# RepeatedKFold 분할기를 만듭니다.
rfk = RepeatedKFold(n_splits=10, n_repeats=5, random_state=42)

# 교차검증을 수행합니다.
cv_results = cross_val_score(pipeline, # 파이프라인
                              features, # 특성 행렬
                              target, # 타깃 벡터
                              cv=rfk, # 교차 검증 기법
                              scoring="accuracy", # 평가 지표
                              n_jobs=-1) # 모든 CPU 코어 사용

# 평균을 계산합니다.
print("평균 : {}".format(cv_results.mean()))

# 10개 폴드의 점수를 모두 확인하기
print("10개 검증 점수 개수 : {}".format(len(cv_results)))
\

평균 : 0.9695065176908755
10개 검증 점수 개수 : 50
```

```
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
```

```
import pandas as pd
import numpy as np
```

```
tennis_data = pd.read_csv('playtennis.csv')
tennis_data
```

```
,
```

Outlook	Temperature	Humidity	Wind	PlayTennis	
0	Sunny	Hot	High	Weak	No
1	Sunny	Hot	High	Strong	No
2	Overcast	Hot	High	Weak	Yes
3	Rain	Mild	High	Weak	Yes
4	Rain	Cool	Normal	Weak	Yes
5	Rain	Cool	Normal	Strong	No
6	Overcast	Cool	Normal	Strong	Yes
7	Sunny	Mild	High	Weak	No
8	Sunny	Cool	Normal	Weak	Yes
9	Rain	Mild	Normal	Weak	Yes
10	Sunny	Mild	Normal	Strong	Yes
11	Overcast	Mild	High	Strong	Yes
12	Overcast	Hot	Normal	Weak	Yes
13	Rain	Mild	High	Strong	No

```
,
```

파일



sample_data

playtennis.csv

```

tennis_data.Outlook = tennis_data.Outlook.replace('Sunny', 0)
tennis_data.Outlook = tennis_data.Outlook.replace('Overcast', 1)
tennis_data.Outlook = tennis_data.Outlook.replace('Rain', 2)

tennis_data.Temperature = tennis_data.Temperature.replace('Hot', 3)
tennis_data.Temperature = tennis_data.Temperature.replace('Mild', 4)
tennis_data.Temperature = tennis_data.Temperature.replace('Cool', 5)

tennis_data.Humidity = tennis_data.Humidity.replace('High', 6)
tennis_data.Humidity = tennis_data.Humidity.replace('Normal', 7)

tennis_data.Wind = tennis_data.Wind.replace('Weak', 8)
tennis_data.Wind = tennis_data.Wind.replace('Strong', 9)

tennis_data.PlayTennis = tennis_data.PlayTennis.replace('No', 10)
tennis_data.PlayTennis = tennis_data.PlayTennis.replace('Yes', 11)

```

tennis_data

、

Outlook	Temperature	Humidity	Wind	PlayTennis	
0	0	3	6	8	10
1	0	3	6	9	10
2	1	3	6	8	11
...12	1	3	7	8	11
13	2	4	6	9	10

、

```
X = np.array(pd.DataFrame(tennis_data, columns = ['Outlook', 'Temperature', 'Humidity', 'Wind']))
```

```
y = np.array(pd.DataFrame(tennis_data, columns = ['PlayTennis']))
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y)
```

```
print('X_train :', X_train)
```

```
print('X_test :', X_test)
```

```
print('y_train :', y_train)
```

```
print('y_test :', y_test)
```

```
,
```

```
X_train : [[1 4 6 9]
```

```
[2 4 6 8]
```

```
[0 3 6 9]
```

```
[0 4 7 9]
```

```
...
```

```
[11]
```

```
[11]
```

```
[10]
```

```
[10]]
```

```
y_test : [[11]
```

```
[10]
```

```
[11]
```

```
[11]]
```

```
,
```

```
# iris - train, validation, test dataset으로 분리
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier #모델-결정트리
from sklearn.metrics import accuracy_score # 모델 평가함수 -> 정확도
from sklearn.model_selection import train_test_split # 데이터셋을 나누는 함수.

iris_data = load_iris()
iris_data.keys()
X, y = iris_data.data, iris_data.target
X.shape, y.shape

import numpy as np
# train, test로 분리
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.2, #기본:0.25
                                                    stratify=y, #target의 class비율에 맞춰서 분리
                                                    random_state=1)

print(X_train.shape, y_train.shape, X_test.shape, y_test.shape)
print(np.unique(y_train, return_counts=True))

\
(120, 4) (120,) (30, 4) (30,)
(array([0, 1, 2]), array([40, 40, 40]))
\
```

```
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train,  
                                                test_size=0.2, stratify=y_train,  
                                                random_state=1)
```

```
X_train.shape, X_val.shape, X_test.shape
```

```
`
```

```
((96, 4), (24, 4), (30, 4))`
```



```
# 모델 생성
tree = DecisionTreeClassifier()
# 모델 학습 - train set
tree.fit(X_train, y_train)
# 예측 및 평가 - train/validation set
pred_train = tree.predict(X_train)
pred_val = tree.predict(X_val)

train_score = accuracy_score(y_train, pred_train)
val_score = accuracy_score(y_val, pred_val)
print("train set의 예측결과: {}, validation set의 예측결과: {}".format(train_score, val_score))
# ==> train set의 예측결과: 1.0, validation set의 예측결과: 0.9166666666666666

# 최종 평가 - test
pred_test = tree.predict(X_test)
test_score = accuracy_score(y_test, pred_test)
print("최종평가(test set): {}".format(test_score))
# ==> 최종평가(test set): 0.9666666666666667

\

train set의 예측결과: 1.0, validation set의 예측결과: 0.9166666666666666
최종평가(test set): 0.9666666666666667
```

Section 6

Learning



1. Problem Definition

2. Data Collection

학습목표



이 워크샵에서는 Problem Definition과 Problem Definition , Problem Definition에 대해 수행할 수 있습니다.

❖ 다음은 scikit-learn의 LinearRegression 클래스를 이용하여 보스턴 집값 데이터에 대해 회귀분석을 하는 예입니다

❖ 데이터 셋 로드

```
from sklearn import datasets
boston_house_prices = datasets.load_boston()
print(boston_house_prices.keys())
print(boston_house_prices.data.shape)
print(boston_house_prices.feature_names)
```

```
\ndict_keys(['data', 'target', 'feature_names', 'DESCR', 'filename', 'data_module'])
(506, 13)
['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRATIO'
 'B' 'LSTAT']
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function load_boston is deprecated; `load_boston` is deprecated in 1.0 and will be removed in 1.2.
```

```
The Boston housing prices dataset has an ethical problem. You can refer to
the documentation of this function for further details.
```

```
...
```

❖ 데이터 셋 정보 확인

```
print(boston_house_prices.DESCR)
```

```
.. _boston_dataset:
```

```
Boston house prices dataset
```

```
**Data Set Characteristics:**
```

```
:Number of Instances: 506
```

```
:Number of Attributes: 13 numeric/categorical predictive. Median Value (attribute 14) is usually the target.
```

```
:Attribute Information (in order):
```

- CRIM per capita crime rate by town
- ZN proportion of residential land zoned for lots over 25,000 sq.ft.
- INDUS proportion of non-retail business acres per town
- CHAS Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
- NOX nitric oxides concentration (parts per 10 million)
- RM average number of rooms per dwelling
- AGE proportion of owner-occupied units built prior to 1940
- DIS weighted distances to five Boston employment centres
- RAD index of accessibility to radial highways
- TAX full-value property-tax rate per \$10,000
- PTRATIO pupil-teacher ratio by town
- B $1000(B_k - 0.63)^2$ where B_k is the proportion of black people by town
- LSTAT % lower status of the population
- MEDV Median value of owner-occupied homes in \$1000's...

❖ 데이터셋을 데이터프레임으로 정제

```
data_frame = pd.DataFrame(boston_house_prices.data)
data_frame.tail()
```

```
\
0          1          2          3          4          5          6          7          8
501         9         10         11         12         0.573        6.593        69.1        2.4786
          1.0        273.0        21.0        391.99        9.67
502        0.04527        0.0        11.93        0.0        0.573        6.120        76.7        2.2875
          1.0        273.0        21.0        396.90        9.08
503        0.06076        0.0        11.93        0.0        0.573        6.976        91.0        2.1675
          1.0        273.0        21.0        396.90        5.64
504        0.10959        0.0        11.93        0.0        0.573        6.794        89.3        2.3889
          1.0        273.0        21.0        393.45        6.48
505        0.04741        0.0        11.93        0.0        0.573        6.030        80.8        2.5050
          1.0        273.0        21.0        396.90        7.88
\
```

❖ 숫자로 된 컬럼명을 feature_names으로 교체합니다.

```
data_frame.columns = boston_house_prices.feature_names
data_frame.tail()
```

```
\
CRIM      ZN      INDUS    CHAS      NOX      RM      AGE      DIS      RAD
501      0.06263    0.0      11.93    0.0      0.573    6.593    69.1    2.4786
      1.0      273.0      21.0    391.99    9.67
502      0.04527    0.0      11.93    0.0      0.573    6.120    76.7    2.2875
      1.0      273.0      21.0    396.90    9.08
503      0.06076    0.0      11.93    0.0      0.573    6.976    91.0    2.1675
      1.0      273.0      21.0    396.90    5.64
504      0.10959    0.0      11.93    0.0      0.573    6.794    89.3    2.3889
      1.0      273.0      21.0    393.45    6.48
505      0.04741    0.0      11.93    0.0      0.573    6.030    80.8    2.5050
      1.0      273.0      21.0    396.90    7.88
\
```

❖ Price에 대한 파생 변수를 추가합니다.

```
data_frame['Price'] = boston_house_prices.target
data_frame.tail()
```

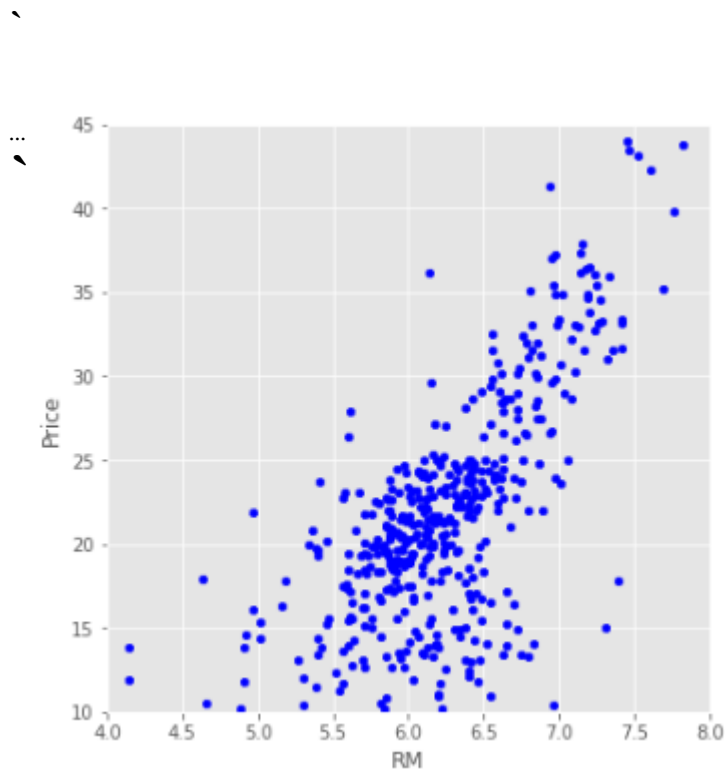
```

CRIM      ZN      INDUS  CHAS      NOX      RM      AGE      DIS      RAD
501      0.06263  0.0      11.93    0.0      0.573    6.593    69.1    2.4786
      1.0      273.0    21.0    391.99    9.67    22.4
502      0.04527  0.0      11.93    0.0      0.573    6.120    76.7    2.2875
      1.0      273.0    21.0    396.90    9.08    20.6
503      0.06076  0.0      11.93    0.0      0.573    6.976    91.0    2.1675
      1.0      273.0    21.0    396.90    5.64    23.9
504      0.10959  0.0      11.93    0.0      0.573    6.794    89.3    2.3889
      1.0      273.0    21.0    393.45    6.48    22.0
505      0.04741  0.0      11.93    0.0      0.573    6.030    80.8    2.5050
      1.0      273.0    21.0    396.90    7.88    11.9

```


❖ 산점도를 표현합니다. 특징 벡터의 이름과 비교하면 각각의 가중치가 가지는 의미를 알 수 있습니다.

```
data_frame.plot(kind="scatter", x="RM", y="Price", figsize=(6,6),  
                 color="blue", xlim = (4,8), ylim = (10,45))
```



❖ 데이터 학습

```
linear_regression = linear_model.LinearRegression()
linear_regression.fit(X = pd.DataFrame(data_frame["RM"]), y = data_frame["Price"])
prediction = linear_regression.predict(X = pd.DataFrame(data_frame["RM"]))
print('a value = ', linear_regression.intercept_)
print('b balue =', linear_regression.coef_)
```

、

```
a value = -34.67062077643857
b balue = [9.10210898]
```

、

❖ 적합도 검증

```
residuals = data_frame["Price"] - prediction
residuals.describe()
```

```
\n
count    5.060000e+02
mean      1.899227e-15
std       6.609606e+00
min      -2.334590e+01
25%      -2.547477e+00
50%       8.976267e-02
75%       2.985532e+00
max       3.943314e+01
Name: Price, dtype: float64
```

```
\n
SSE = (residuals**2).sum()
SST = ((data_frame["Price"]-data_frame["Price"].mean())**2).sum()
R_squared = 1 - (SSE/SST)
print('R_squared = ', R_squared)
```

```
\n
R_squared = 0.4835254559913341
```

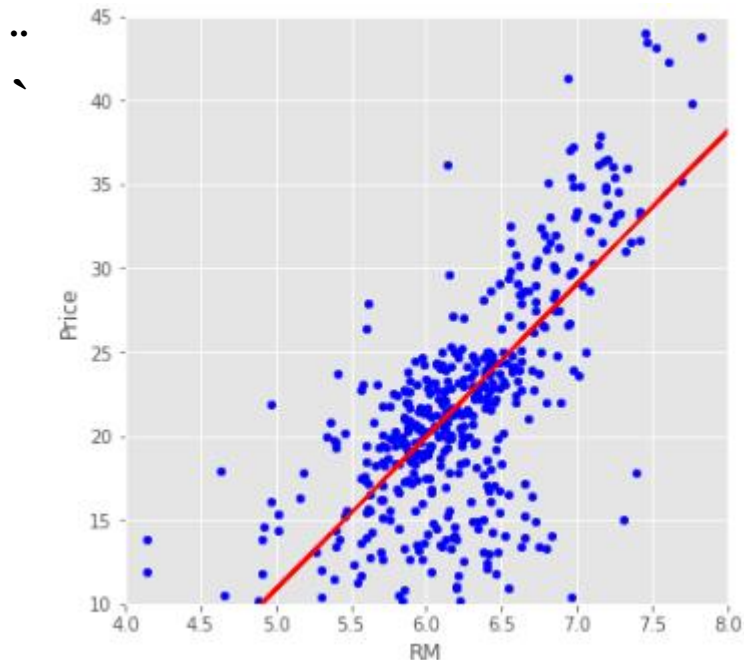
결정계수 48.35%로 결과를 통해 x값이 y값에 영향을 주는 것을 확인했습니다.

❖ 예측하여 플롯으로 표현

```
data_frame.plot(kind="scatter",x="RM",y="Price",figsize=(6,6),  
                  color="blue", xlim = (4,8), ylim = (10,45))
```

```
# Plot regression line
```

```
plt.plot(data_frame["RM"],prediction,color="red")
```



❖ 성능평가

```
print('score = ', linear_regression.score(X = pd.DataFrame(data_frame["RM"]), y =  
data_frame["Price"]))  
print('Mean_Squared_Error = ', mean_squared_error(prediction, data_frame["Price"]))  
print('RMSE = ', mean_squared_error(prediction, data_frame["Price"])**0.5)
```

```
`
```

```
score = 0.48352545599133423  
Mean_Squared_Error = 43.60055177116956  
RMSE = 6.603071389222561
```

```
`
```

Section 7

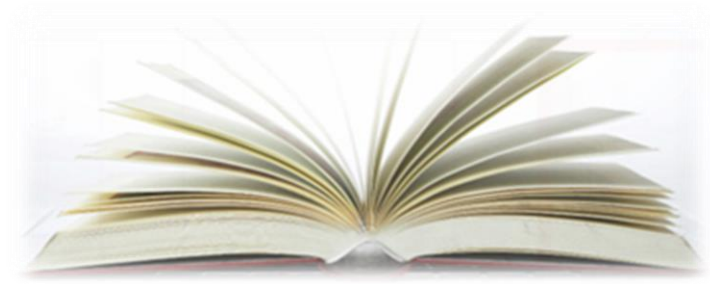


Evaluation

- 1. Problem Definition**

- 2. Data Collection**

학습목표



이 워크샵에서는 Problem Definition과 Problem Definition , Problem Definition에 대해 수행할 수 있습니다.

❖ 데이터 학습과 성능 평가

```
dt_clf = DecisionTreeClassifier()
dt_clf = dt_clf.fit(X_train, y_train)

dt_prediction = dt_clf.predict(X_test)

print(confusion_matrix(y_test, dt_prediction))

\
[[0 0]
 [2 2]]
\

print(classification_report(y_test, dt_prediction))

\
precision recall f1-score support

 10   0.00   0.00   0.00   0
 11   1.00   0.50   0.67   4

accuracy           0.50   4
macro avg   0.50   0.25   0.33   4
weighted avg   1.00   0.50   0.67   4

\
```


Section 1

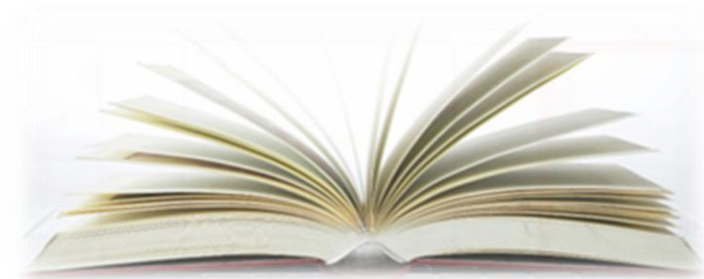


Problem Definition

- 1. Problem Definition**

- 2. Data Collection**

학습목표



이 워크샵에서는 Problem Definition과 Problem Definition , Problem Definition에 대해 수행할 수 있습니다.

Subsection 2



Problem Definition

분석 목표 정의서

분석 기본 정의	분석 명칭	OK/NG에 대한 추세 모니터링을 통해 장비 점검 시점 사전 예측	분석목표 확정일	2021-12-XX
	분석 목적	OK/NG에 대한 추세 모니터링을 통해 장비 점검 시점 사전 예측을 통해 설비가동률 향상 및 불량률 감소, 실패비용 감소	분석 목표 워크숍	2021-12-XX
	분석 우선순위	상	담당 조직명	
	분석 접근 방안	과거 품질불량 판정 데이터를 통해 시계열 시뮬레이션을 돌려보며 시간/주별 불량률 추세 분석과 불량률 사전 예측 이후 사전 예측 결과를 토대로 부품별 이슈 파악 후 사전 조치 가능할 수 있도록 하는 것이 목표		
성과 측정	정성적 기준	신규 기법/기술 : 다양한 EDA 및 예측 기법 및 머신러닝/ 딥러닝활용으로 Best Model 선정 기존 데이터: 공정에서 발생하는 불량 정보 포함 데이터 정보 신규 데이터 : ...		
	정량적 기준	기존 제조공정 대비설비 가동률 50% 이상 향상 제품 불량률 2% 이상 감소(2.5% → 0.05%) 목표 달성		
데이터 정보	내부 데이터	2020-01-21 부터 2020-01-25까지 A_Line 데이터 셋	데이터 입수 난이도	중
	외부 데이터		데이터 입수 난이도	

프로젝트 헌장(Project Charter)

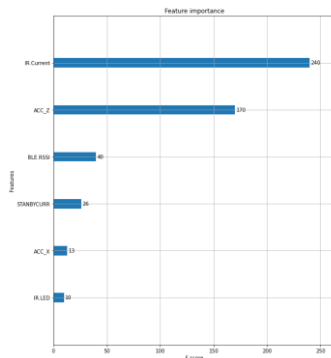
프로젝트 명 (Project Name)	OK/NG에 대한 추세 모니터링을 통해 장비 점검 시점 사전 예측을 위한 머신러닝·AI·딥러닝		
프로젝트 설명 (Project Description)			
프로젝트 매니저(Project Manager, PM)		승인 날짜(Date Approved)	
프로젝트 스폰서(Project Sponsor)		서명(Signature)	인용
비즈니스 케이스 (Business Case)	목표(Goals) / 산출물(Deliverables)		
팀 구성원(Team Member)			
이름(Name)	역할(Role)		
위험과 제약사항(Risk and Constraints)		주요 일정(Milestones)	

○○○ 품질 예측하기

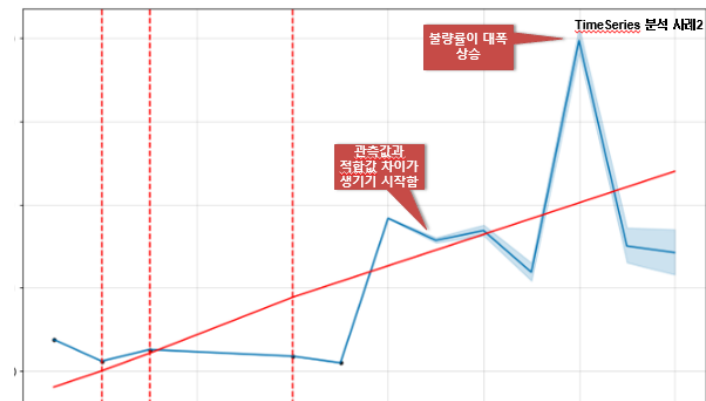
목표	OK/NG에 대한 추세 모니터링을 통해 장비 점검 시점 사전 예측을 통해 설비가동률 향상 및 불량률 감소, 실패비용 감소
핵심개념	ARIMA, Fbprophet
데이터 수집	○○○ 데이터 셋 : ○○○ 저장소에서 다운로드
데이터 준비	수집한 데이터 파일 병합
데이터 탐색	1. 정보 확인 : info() 2. 기술 통계 확인 : describe(), unique(), value_counts()

결과 시각화

xgb_model을 이용한 피쳐 엔지니어링



TimeSeries 분석을 이용한 시각화



Subsection 2



Data Collection

데이터 수집 세부 계획서

○○○ 데이터 수집 세부 계획서

OK/NG에 대한 추세 모니터링을 통해 장비 점검 시점 사전 예측을 위한

담당자

데이터 수집 세부 계획서

문서번호

작성자

작성일자

1. 분석 목적

2. 수집 데이터 상세 조사 내용

데이터 유형	통계 문자 텍스트 음성 이미지 동영상 GIS 기타				수집 주기	
위치	수요기업 공급기업 보유 또는 수집 허브 데이터셋 공공 데이터				확보 비용	
크기	레코드수	300	레 코 드 단위	장	데이터 이관 절차	
	크기	15	단위	MB		
보관 방식						

3. 적절성 검증 방식

데이터 누락/중복

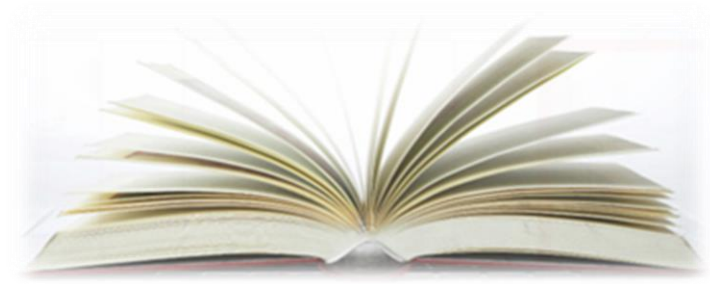
데이터 오류

개인정보 유무

포함 | 미포함

데이터 저작권

Section 2



Development environment Configuration

1. 윈도우10+Miniconda+텐서플로2-CPU
2. Flask 개발 환경 만들기

학습목표



❖ 이 워크샵에서는 환경구성에 대해 알 수 있습니다.

Subsection 1

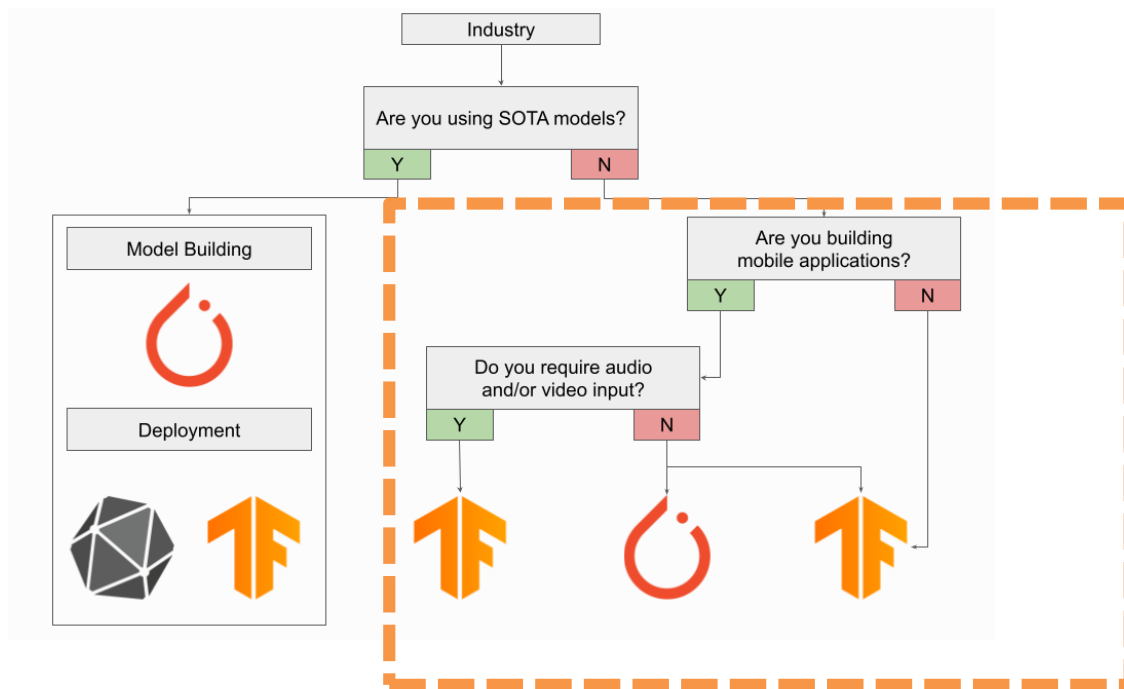


**원10+Miniconda+텐서플로
2-CPU**

Should I Use PyTorch or TensorFlow?

❖ 내가 산업체에 있다면?

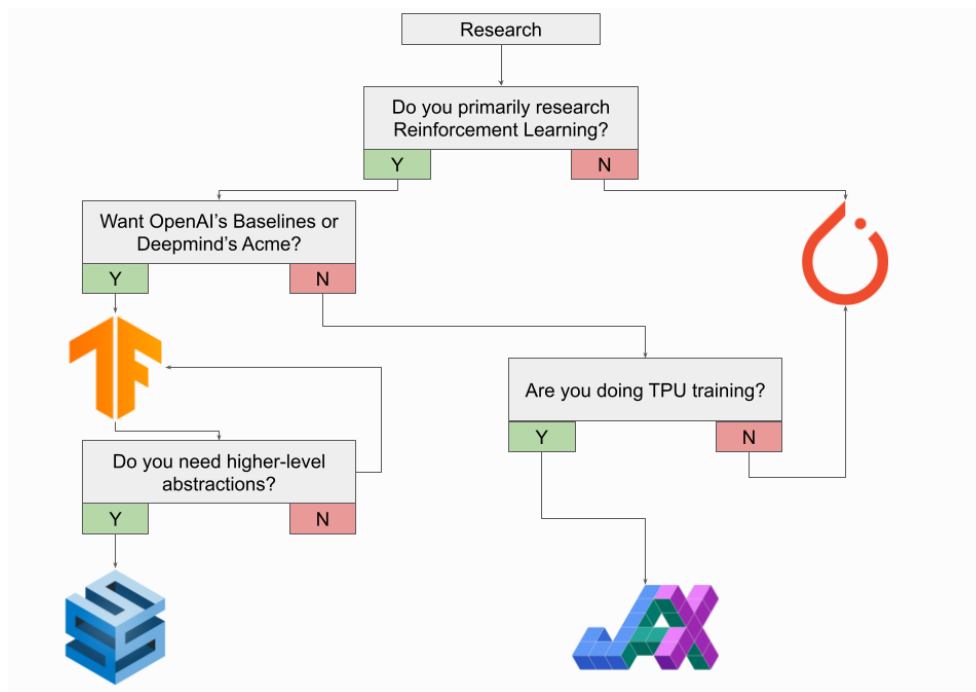
- 산업 환경에서 딥 러닝 엔지니어링을 수행하는 경우 TensorFlow를 사용하고 있을 가능성이 높으며 계속 사용해야 합니다. TensorFlow의 강력한 배포 프레임워크와 종단 간 TensorFlow Extended 플랫폼은 모델을 생산해야 하는 사람들에게 매우 중요합니다. 모델 모니터링 및 아티팩트 추적과 함께 gRPC 서버에 쉽게 배포하는 것은 업계에서 사용하기 위한 중요한 도구입니다.



Should I Use PyTorch or TensorFlow?

❖ 내가 연구원이라면?

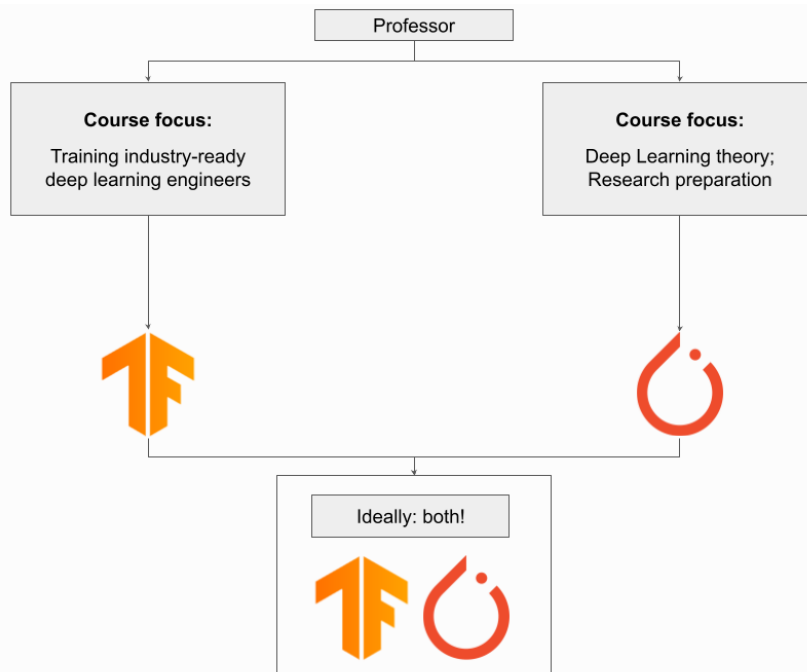
- 연구원이라면 거의 확실하게 PyTorch를 사용하고 있으며 현재로서는 계속 사용하고 있을 것입니다.



Should I Use PyTorch or TensorFlow?

❖ 내가 교수라면?

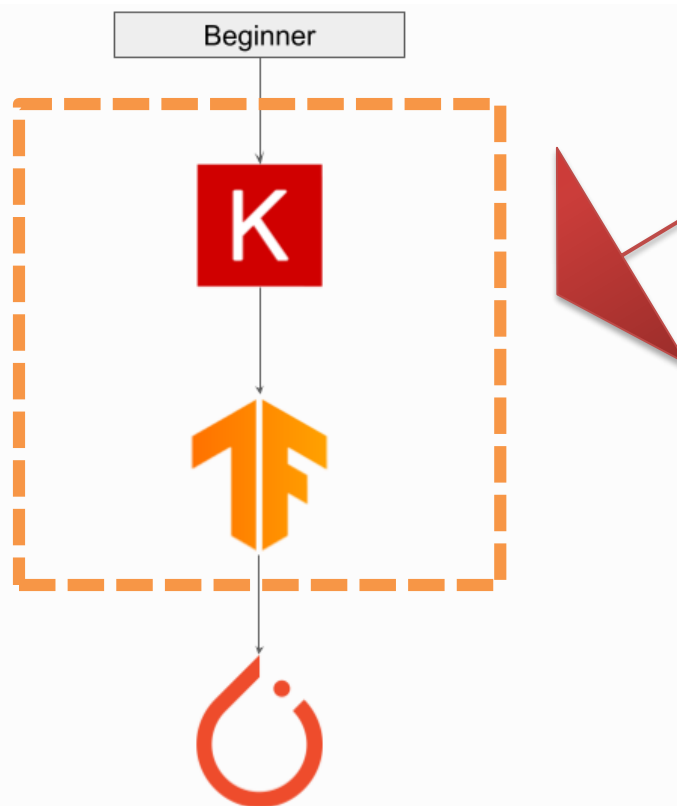
- 교육 과정의 초점이 딥 러닝 이론뿐만 아니라 전체 딥 러닝 프로세스에서 역량을 발휘하여 기초를 다질 수 있는 업계 준비 딥 러닝 엔지니어를 양성하는 것이라면 TensorFlow를 사용해야 합니다.
- 딥 러닝 연구를 수행할 수 있도록 학생들을 준비시키기 위한 고급 학부 과정이나 초기 대학원 수준 과정을 가르치는 경우 PyTorch를 사용해야 합니다.



Should I Use PyTorch or TensorFlow?

❖ 내가 완전 초보자라면?

- 딥 러닝에 관심이 있고 막 시작하려는 완전 초보자라면 Keras 를 사용하는 것이 좋습니다.



❖ Tensorflow2를 기준으로 설치

1. Miniconda 설치
2. 텐서플로2
3. JupyterNotebook 설치

● <https://www.tensorflow.org/install>

❖ Python 3.5 부터 3.8을 지원

TensorFlow

설치 학습 API 라이선스 커뮤니티 TensorFlow를 사용해야 하는 이유

설치

TensorFlow 설치

패키지
pip
Docker

추가 설정
GPU 지원
문제

소스에서 빌드
Linux/Mac OS
Windows
SIG 빌드

언어 바인딩
자바
자바(레거시)
C
Go

Help protect the Great Barrier Reef with TensorFlow on Kaggle [Join Challenge](#)

이 페이지의 내용
첫 ML 앱 빌드하기

TensorFlow 2 설치

TensorFlow는 다음 64비트 시스템에서 테스트 및 지원됩니다.

- Python 3.6~3.9
- macOS 10.12.6(Sierra) 이상(GPU 지원 없음)
- Ubuntu 16.04 이상
- Windows 7 이상(C++ 재배포 가능 패키지)

패키지 다운로드

Python의 `pip` 패키지 관리자를 사용해 TensorFlow를 설치하세요.

★ TensorFlow 2 패키지에는 `pip 19.0`가 넘는 버전(또는 macOS의 경우 20.3이 넘는 버전)가 필요합니다.

공식 패키지는 Ubuntu, Windows, macOS에서 사용할 수 있습니다.

CUDA® 지원 카드의 경우 [GPU 가이드](#)를 참고하시기 바랍니다.

```
# Requires the latest pip
$ pip install --upgrade pip

# Current stable release for CPU and GPU
$ pip install tensorflow

# Or try the preview build (unstable)
$ pip install tf-nightly
```

❖ 가상환경(Virtual Environment)을 이용하면 Python 버전 간의 의존성을 고려해서 가상의 격리된 환경을 만들어줌으로써 버전이 다름으로 인해 발생할 수 있는 호환이나 충돌 문제를 미연에 방지할 수 있음.

❖ Miniconda 설치

1. 윈도우 환경이라면 Miniconda3 Windows 64-bit를 다운로드합니다.

<https://docs.conda.io/en/latest/miniconda.html>

The screenshot shows the Miniconda documentation page. The left sidebar contains a search bar and a navigation menu with links to Conda, Conda-build, Miniconda, System requirements, Latest Miniconda Installer Links, Windows installers, macOS installers, Linux installers, Installing, Other resources, Help and support, Contributing, and Conda license. The main content area is titled 'System requirements' and lists the license, operating system, system architecture, and minimum disk space. Below this is a section for 'Latest Miniconda Installer Links' with a table of installers for Windows and macOS. The table has columns for Platform, Name, and SHA256 hash. The Windows 64-bit installer is highlighted with a red box.

System requirements

- License: Free use and redistribution under the terms of the [EULA for Miniconda](#).
- Operating system: Windows 8 or newer, 64-bit macOS 10.13+, or Linux, including Ubuntu, RedHat, CentOS 7+, and others.
- If your operating system is older than what is currently supported, you can find older versions of the Miniconda installers in our [archive](#) that might work for you.
- System architecture: Windows- 64-bit x86, 32-bit x86; macOS- 64-bit x86 & Apple M1 (ARM64); Linux- 64-bit x86, 64-bit aarch64 (AWS Graviton2), 64-bit IBM Power8/Power9, s390x (Linux on IBM Z & LinuxONE).
- The `linux-aarch64` Miniconda installer requires `glibc >=2.26` and thus will **not** work with CentOS 7, Ubuntu 16.04, or Debian 9 ("stretch").
- Minimum 400 MB disk space to download and install.

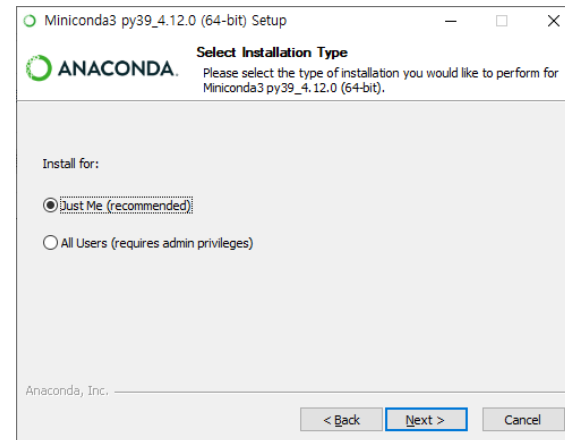
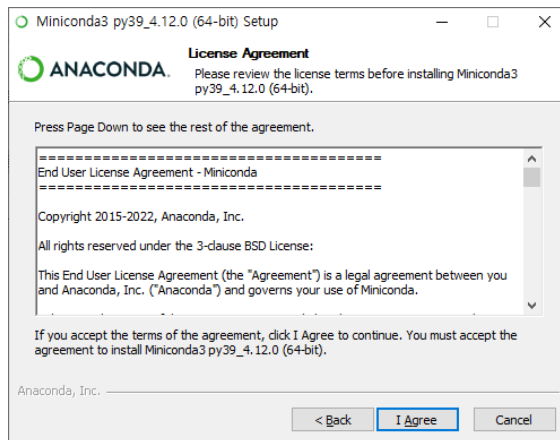
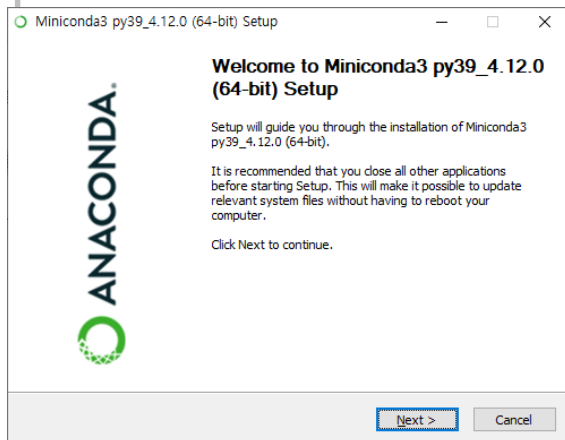
On Windows, macOS, and Linux, it is best to install Miniconda for the local user, which does not require administrator permissions and is the most robust type of installation. However, if you need to, you can install Miniconda system wide, which does require administrator permissions.

Latest Miniconda Installer Links

Latest - Conda 4.12.0 Python 3.9.7 released February 15, 2022

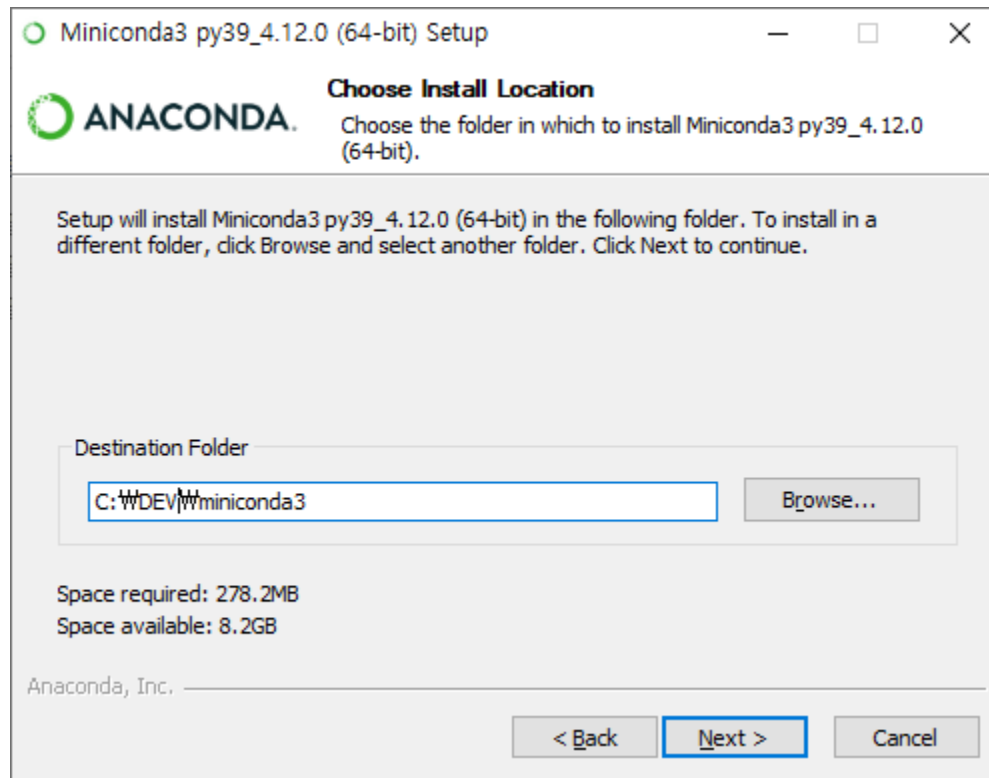
Platform	Name	SHA256 hash
Windows	Miniconda3 Windows 64-bit	1acbc2e8277ddd54a5f724896c7edee112d068529588d944782966c867e99cc
	Miniconda3 Windows 32-bit	4fb64e6c9c28b88beab16994bfba4829110ea3145baa60bda5344174ab65d462
macOS	Miniconda3 macOS Intel x86 64-bit bash	007bae6f18dc7b6f2ca6209b5a0c9bd2f283154152f82becf787aac709a51633

* Next 클릭

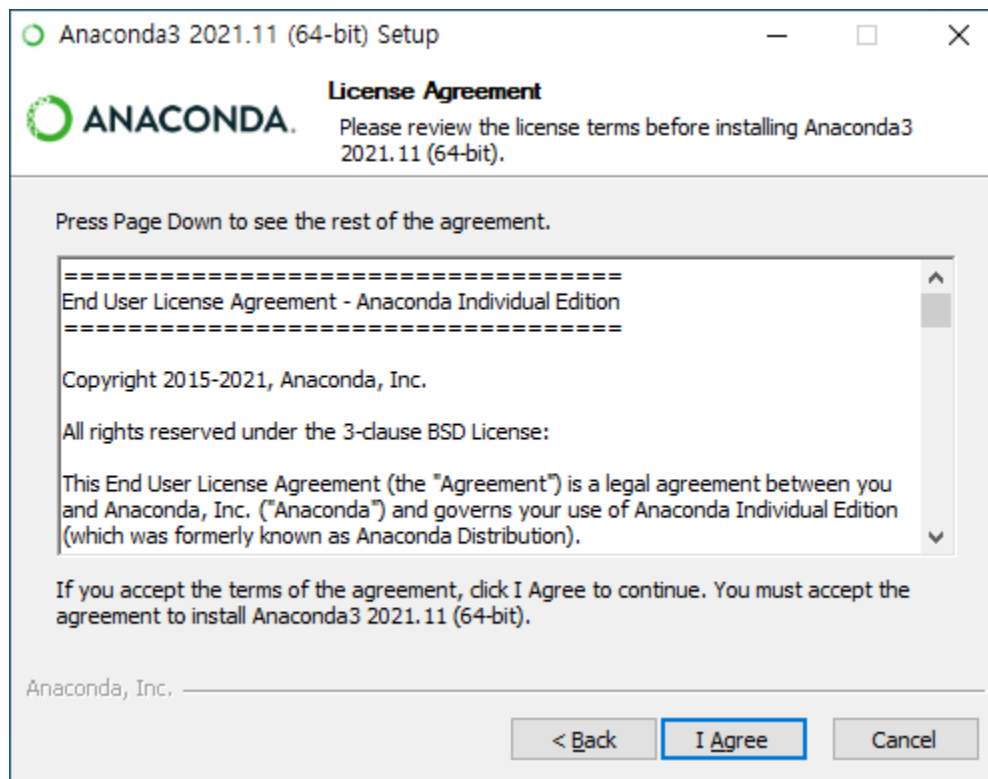


Miniconda 설치

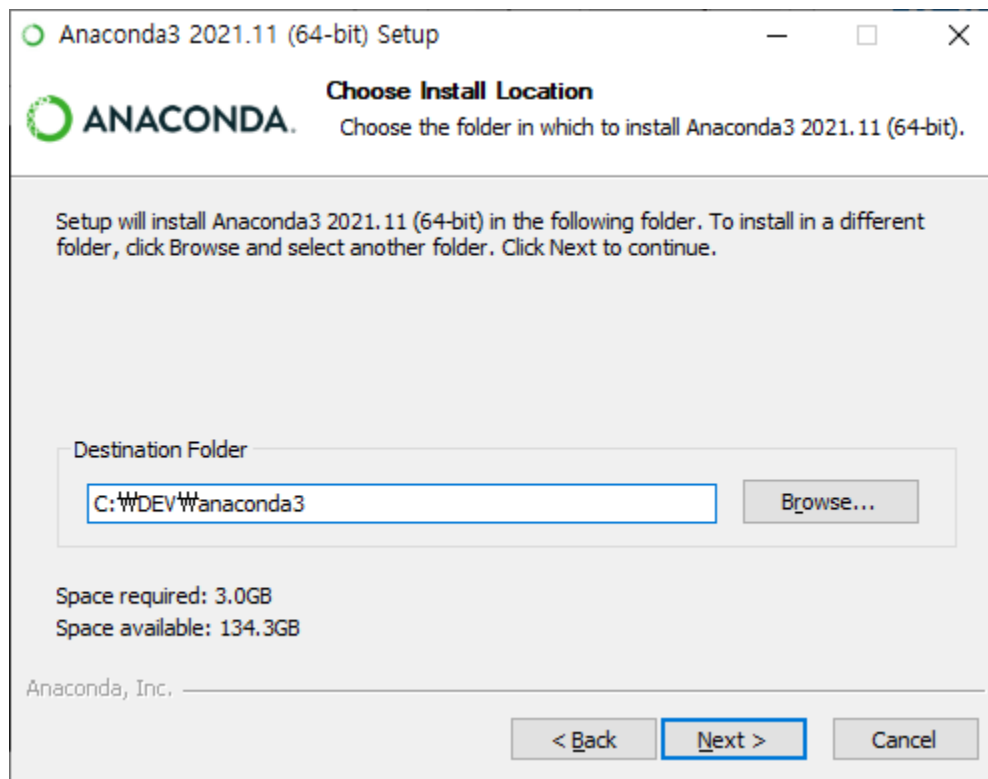
* Next 클릭



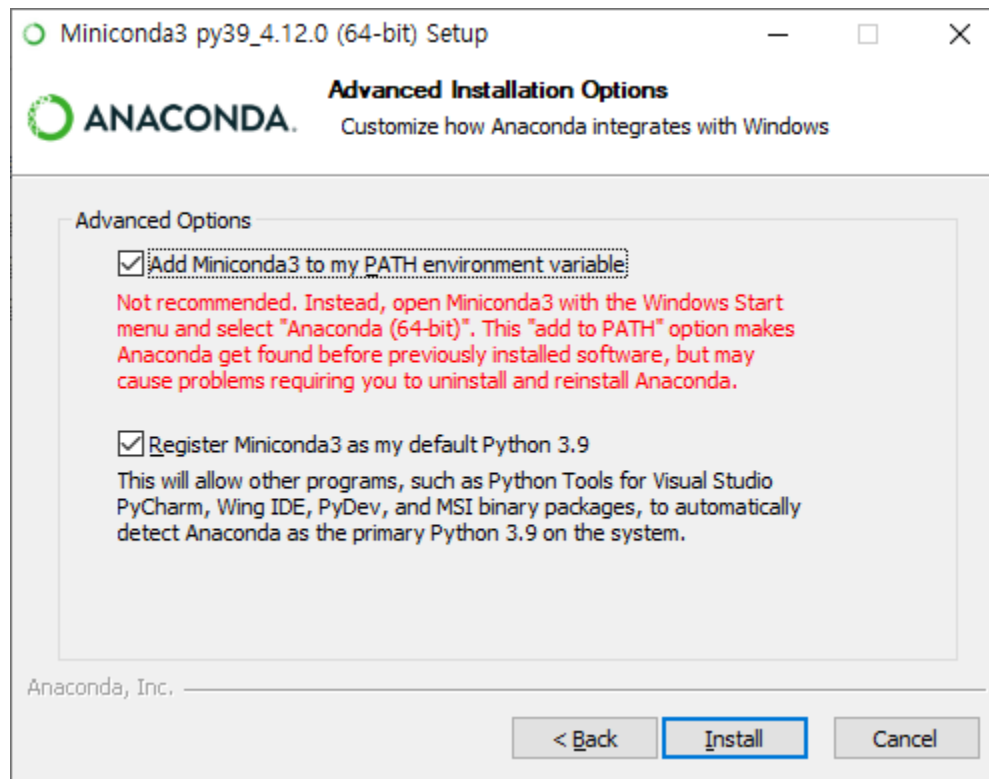
* I Agree 클릭



* C:\DEV\Miniconda3로 경로 설정하고 Next



* Register Miniconda3 as my default Python 3.8 '만 체크하고 Install 클릭



Miniconda 설치

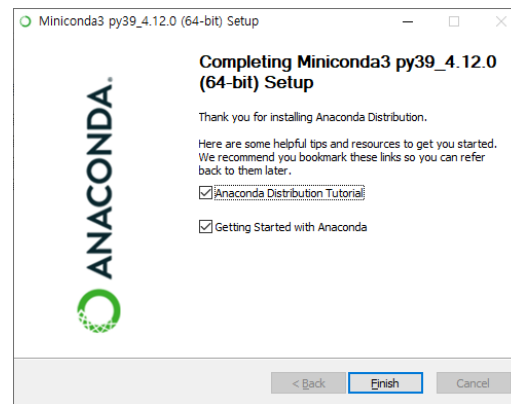
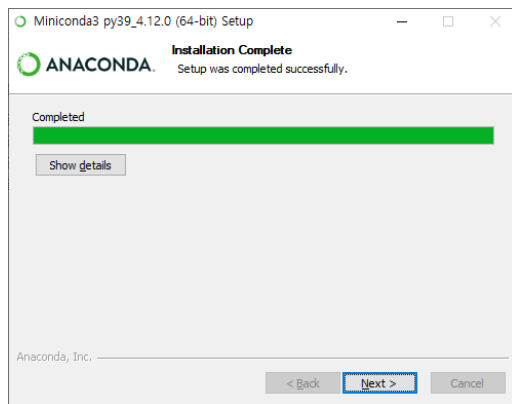
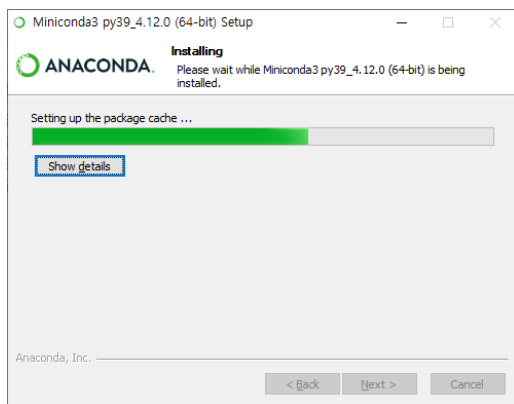
❖ Add Miniconda3 to my PATH environment variable:

- PATH 환경 변수에 Miniconda를 추가할지 선택
- Miniconda 외에 다른 파이썬 인터프리터를 환경변수에 등록해서 사용한다면 체크 해제
- Miniconda 만을 사용하는 경우, Miniconda가 주력일 경우로 윈도우 cmd 창에서 파이썬을 실행할 경우, 현재 PC에 파이썬을 설치한 적이 없는 경우에는 체크
- (체크할 경우 윈도우 cmd 창 경로와 상관없이 Miniconda를 파이썬으로 인식)

❖ Register Miniconda3 as my default Python 3.8:

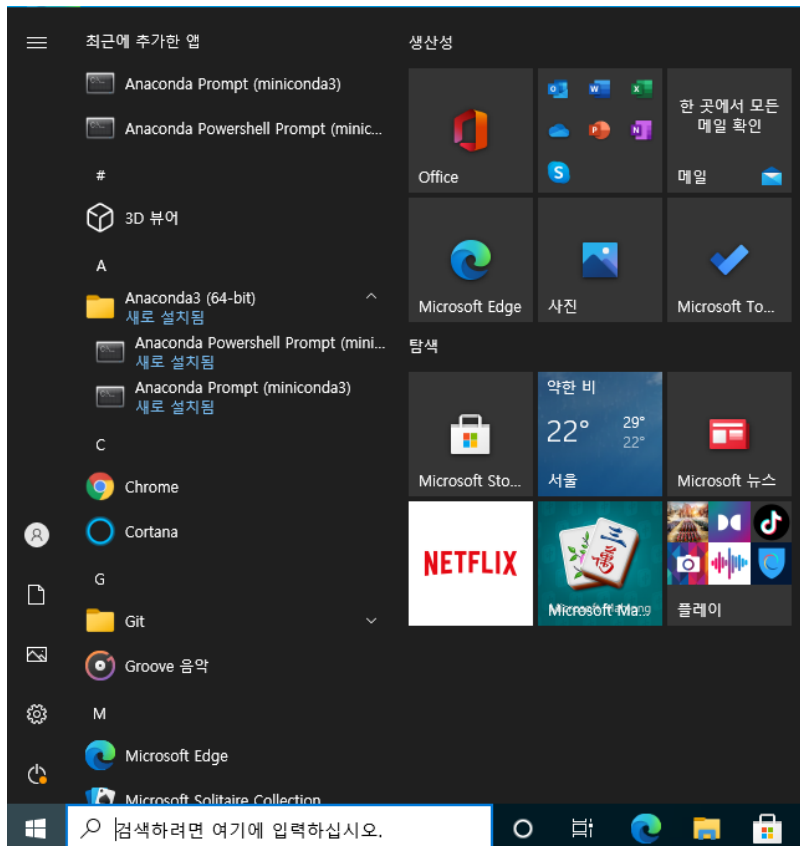
- Miniconda를 기본 파이썬으로 등록할지 여부를 선택
- (체크할 경우 개발 도구나 에디터에서 Miniconda를 파이썬으로 인식)

* Next 클릭



Miniconda 설치

2. 윈도우키를 누르고 **anaconda**를 입력시 보이는 **Anaconda Prompt**를 실행시킵니다. **conda** 위치를 시스템 변수 **PATH**에 추가하여 일반 명령프롬프트에서 사용가능하게 할 수 있지만 Miniconda 공식 문서에서 권장하지 않습니다.



```
(base) C:\Users\k8s>python
```

```
Python 3.9.12 (main, Apr 4 2022, 05:22:27) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32  
Type "help", "copyright", "credits" or "license" for more information.
```

```
>>> 3+5
```

```
8
```

```
>>> quit()
```

```
(base) C:\Users\k8s>
```

Jupyter notebook 설치

❖ pip을 업그레이드

```
(base) C:\Users\k8s>pip install --upgrade pip
```

```
Requirement already satisfied: pip in c:\dev\miniconda3\lib\site-packages (22.1.2)
```

```
(base) C:\Users\k8s>
```

❖ Jupyter notebook 설치

```
(base) C:\Users\k8s>pip install jupyter
```

```
Collecting jupyter
```

```
  Downloading jupyter-1.0.0-py2.py3-none-any.whl (2.7 kB)
```

```
Collecting jupyter-console
```

```
  Downloading jupyter_console-6.4.4-py3-none-any.whl (22 kB)
```

```
...
```

Jupyter notebook 설치

(base) C:\Users\k8s>jupyter notebook

```
[I 10:15:08.860 NotebookApp] Writing notebook server cookie secret to C:\Users\k8s\AppData\Roaming\jupyter\runtime\notebook_cookie_secret
[I 10:15:10.203 NotebookApp] Serving notebooks from local directory: C:\Users\k8s
[I 10:15:10.203 NotebookApp] Jupyter Notebook 6.4.12 is running at:
[I 10:15:10.203 NotebookApp] http://localhost:8888/?token=0633164bc72ce7ae73179310241e5c968e1576c718461681
[I 10:15:10.203 NotebookApp] or http://127.0.0.1:8888/?token=0633164bc72ce7ae73179310241e5c968e1576c718461681
[I 10:15:10.203 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 10:15:10.344 NotebookApp]
```

To access the notebook, open this file in a browser:

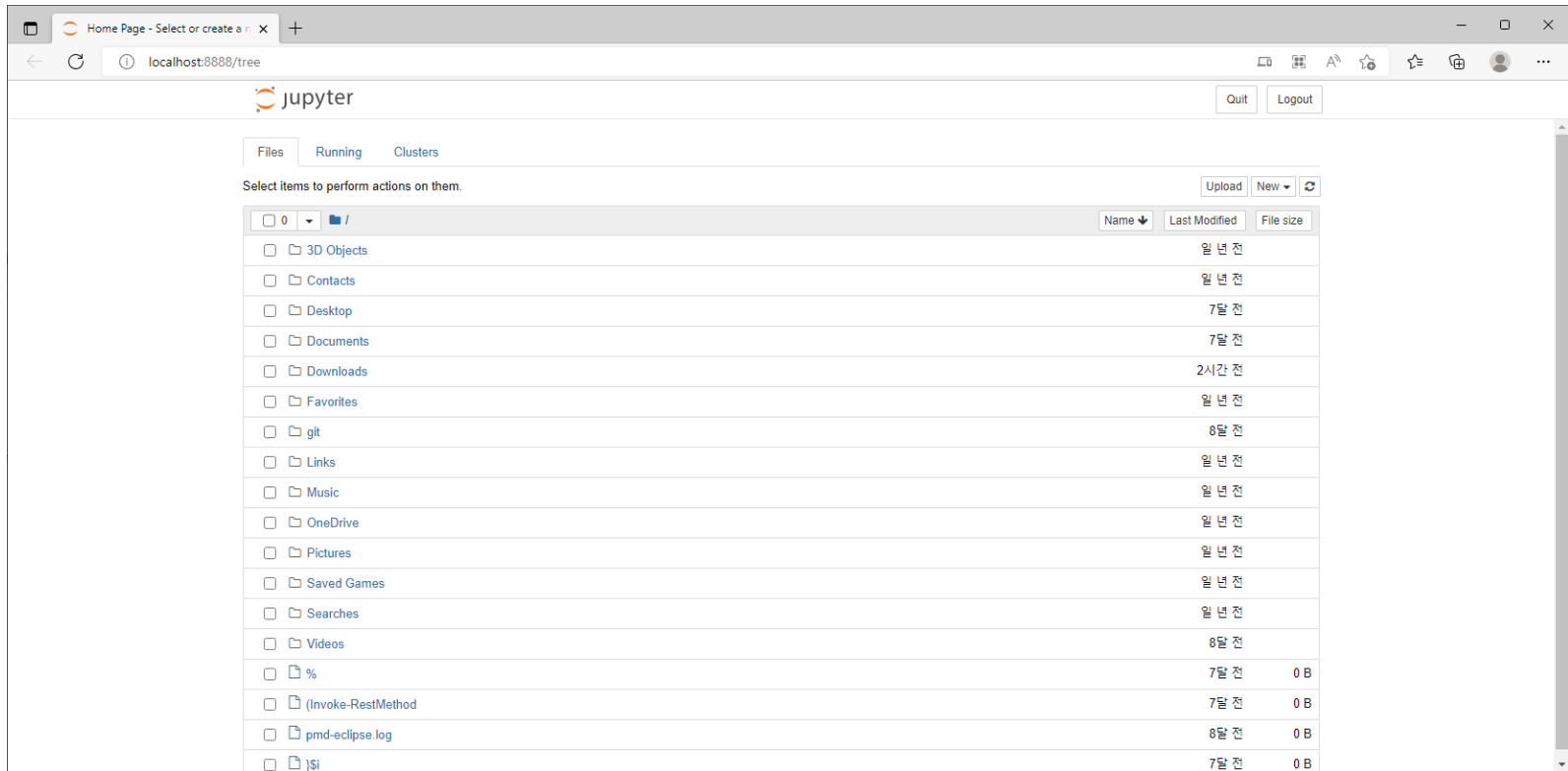
file:///C:/Users/k8s/AppData/Roaming/jupyter/runtime/nbserver-8528-open.html

Or copy and paste one of these URLs:

http://localhost:8888/?token=0633164bc72ce7ae73179310241e5c968e1576c718461681

or http://127.0.0.1:8888/?token=0633164bc72ce7ae73179310241e5c968e1576c718461681

<http://localhost:8888/>



(base) C:\Users\DaeKyeong>pip install jupyter_contrib_nbextensions

Collecting jupyter_contrib_nbextensions

Downloading jupyter_contrib_nbextensions-0.5.1-py2.py3-none-any.whl (20.9 MB)

██████████ | 3.2 MB 1.7 MB/s eta 0:00:11

...

(base) C:\Users\DaeKyeong>jupyter contrib nbextension install --user

...

[I 22:08:50 InstallContribNbextensionsApp] Enabling notebook extension contrib_nbextensions_help_item/main...

[I 22:08:50 InstallContribNbextensionsApp] - Validating: ok

[I 22:08:50 InstallContribNbextensionsApp] - Editing config: C:\Users\DaeKyeong\.jupyter\jupyter_nbconvert_config.json

[I 22:08:50 InstallContribNbextensionsApp] -- Configuring nbconvert template path

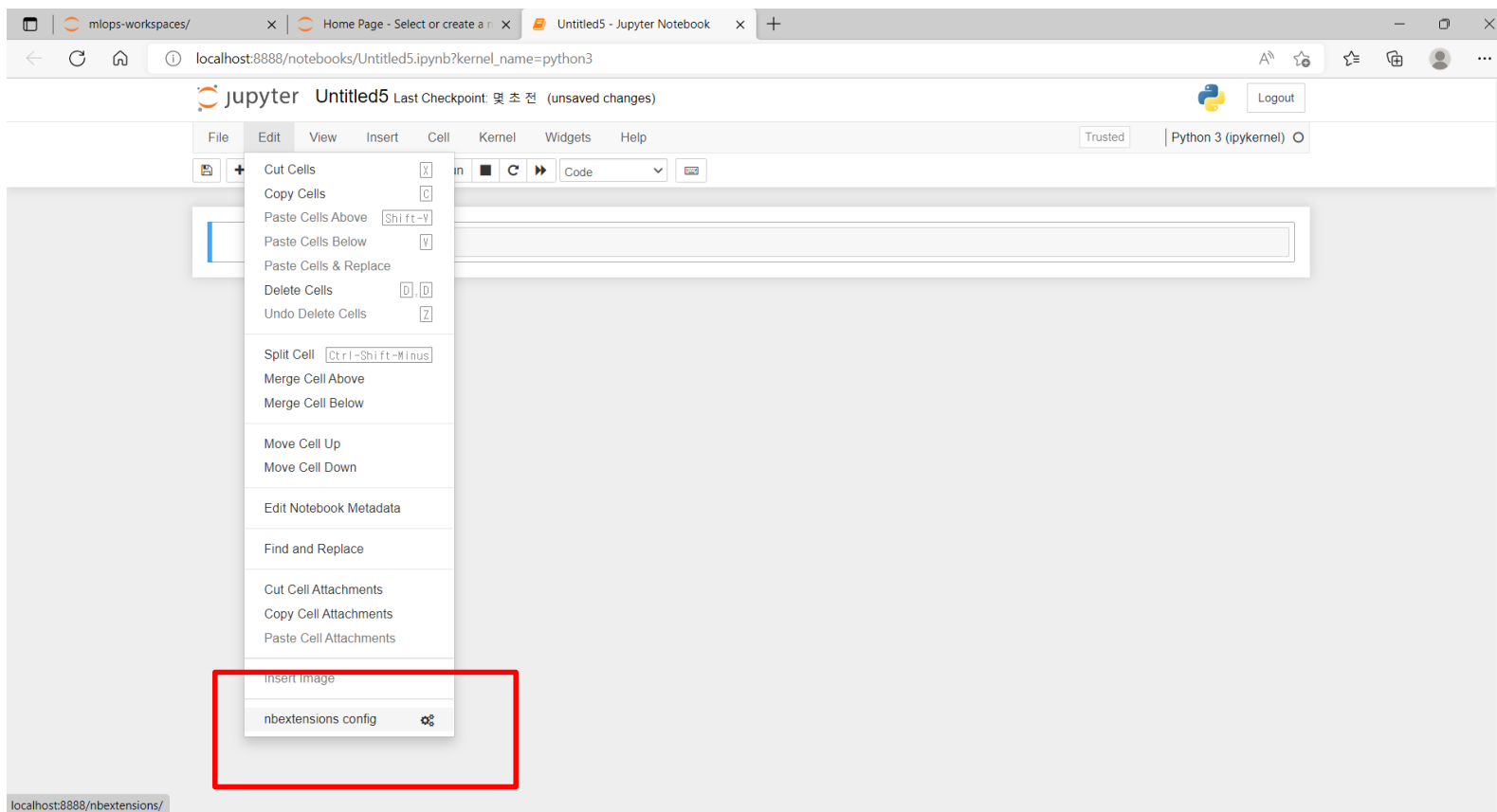
[I 22:08:50 InstallContribNbextensionsApp] -- Configuring nbconvert preprocessors

[I 22:08:50 InstallContribNbextensionsApp] - Writing config: C:\Users\DaeKyeong\.jupyter\jupyter_nbconvert_config.json

[I 22:08:50 InstallContribNbextensionsApp] -- Writing updated config file C:\Users\DaeKyeong\.jupyter\jupyter_nbconvert_config.json

(base) C:\Users\DaeKyeong>jupyter notebook

Edit > nbextensions config 클릭



항목 체크 해제 후 항목 체크

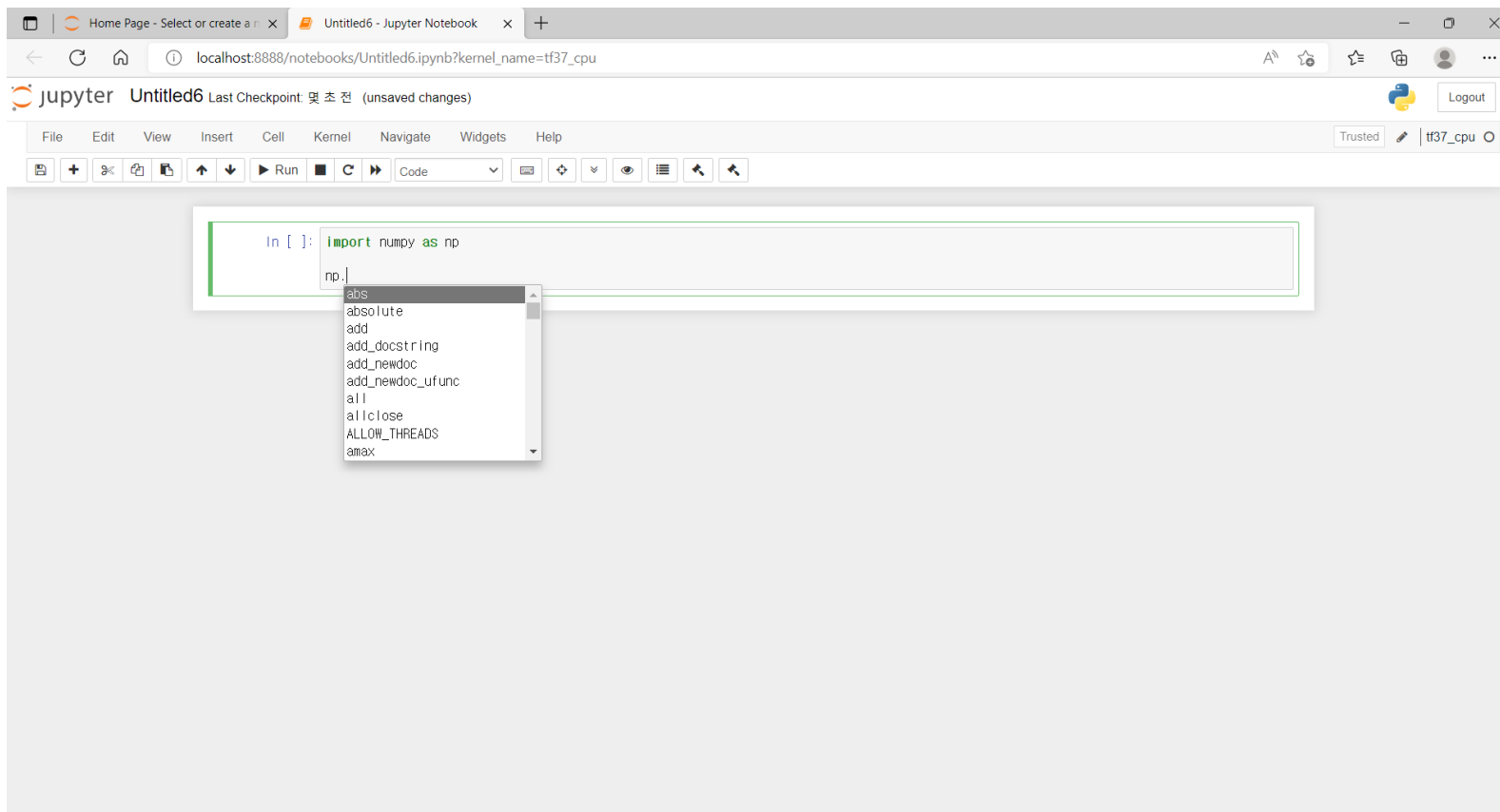
Configurable nbextensions

☒ disable configuration for nbextensions without explicit compatibility (they may break your notebook environment, but can be useful to show for nbextension development)

filter: by description, section, or tags

- | | | | |
|--|--|--|---|
| <input type="checkbox"/> (some) LaTeX environments for Jupyter | <input type="checkbox"/> 2to3 Converter | <input type="checkbox"/> AddBefore | <input checked="" type="checkbox"/> Autopep8 |
| <input type="checkbox"/> AutoSaveTime | <input type="checkbox"/> Autoscroll | <input checked="" type="checkbox"/> Cell Filter | <input type="checkbox"/> Code Font Size |
| <input checked="" type="checkbox"/> Code prettify | <input type="checkbox"/> Codefolding | <input type="checkbox"/> Codefolding in Editor | <input checked="" type="checkbox"/> CodeMirror mode extensions |
| <input type="checkbox"/> Collapsible Headings | <input type="checkbox"/> Comment/Uncomment Hotkey | <input checked="" type="checkbox"/> contrib_nbextensions_help_item | <input type="checkbox"/> datestamper |
| <input type="checkbox"/> Equation Auto Numbering | <input type="checkbox"/> ExecuteTime | <input type="checkbox"/> Execution Dependencies | <input type="checkbox"/> Exercise |
| <input type="checkbox"/> Exercise2 | <input type="checkbox"/> Export Embedded HTML | <input type="checkbox"/> Freeze | <input type="checkbox"/> Gist-it |
| <input type="checkbox"/> Help panel | <input type="checkbox"/> Hide Header | <input type="checkbox"/> Hide input | <input checked="" type="checkbox"/> Hide input all |
| <input type="checkbox"/> Highlight selected word | <input type="checkbox"/> highlighter | <input checked="" type="checkbox"/> Hinterland | <input type="checkbox"/> Initialization cells |
| <input type="checkbox"/> isort formatter | <input checked="" type="checkbox"/> jupyter-js-widgets/extension | <input checked="" type="checkbox"/> jupyter_tensorboard/tree | <input type="checkbox"/> Keyboard shortcut editor |
| <input type="checkbox"/> Launch QTConsole | <input type="checkbox"/> Limit Output | <input type="checkbox"/> Live Markdown Preview | <input type="checkbox"/> Load TeX macros |
| <input type="checkbox"/> Move selected cells | <input type="checkbox"/> Navigation-Hotkeys | <input checked="" type="checkbox"/> Nbextensions dashboard tab | <input checked="" type="checkbox"/> Nbextensions edit menu item |
| <input type="checkbox"/> nbTranslate | <input type="checkbox"/> Notify | <input type="checkbox"/> Printview | <input type="checkbox"/> Python Markdown |
| <input type="checkbox"/> Rubberband | <input type="checkbox"/> Ruler | <input type="checkbox"/> Ruler in Editor | <input type="checkbox"/> Runtools |
| <input type="checkbox"/> Scratchpad | <input checked="" type="checkbox"/> ScrollDown | <input type="checkbox"/> Select CodeMirror Keymap | <input type="checkbox"/> SKILL Syntax |
| <input type="checkbox"/> Skip-Traceback | <input type="checkbox"/> Snippets | <input type="checkbox"/> Snippets Menu | <input type="checkbox"/> spellchecker |
| <input type="checkbox"/> Split Cells Notebook | <input checked="" type="checkbox"/> Table of Contents (2) | <input type="checkbox"/> table_beautifier | <input type="checkbox"/> Toggle all line numbers |
| <input type="checkbox"/> Tree Filter | <input checked="" type="checkbox"/> Variable Inspector | <input type="checkbox"/> zenmode | |

쥬피터 노트북 재시작



Windows10에서 Miniconda Prompt를 이용해 가상환경 만들기

❖ Environment tf36_cpu

```
(base) C:\Users\Wk8s>conda create -n tf36_cpu python=3.6
```

```
...
```

```
done
```

```
#
```

```
# To activate this environment, use
```

```
#
```

```
# $ conda activate tf36_cpu
```

```
#
```

```
# To deactivate an active environment, use
```

```
#
```

```
# $ conda deactivate
```

```
(base) C:\Users\Wk8s>
```

```
(base) C:\Users\Wk8s>conda env list
```

#

```
base          * C:\WDEV\miniconda3
tf36_cpu      C:\WDEV\miniconda3\envs\tf36_cpu
```

```
(base) C:\Users\Wk8s>conda activate tf36 cpu
```

```
(tf36_cpu) C:\Users\Wk8s>python
```

```
Python 3.6.13 |Anaconda, Inc.| (default, Mar 16 2021, 11:37:27) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
```

```
>>> import tensorflow as tf
```

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

ModuleNotFoundError: No module named 'tensorflow'

```
>>> quit()
```

```
(tf36_cpu) C:\Users\Wk8s>pip install --upgrade tensorflow
```

Collecting tensorflow

Downloading tensorflow-2.6.2-cp36-cp36m-win_amd64.whl (423.3 MB)

423.3 MB 9.8 kB/s

Collecting opt-einsum~3.3.0

Downloading opt einsum-3.3.0-py3-none-any.whl (65 kB)

65 kB 4.5 MB/s

...

Windows10에서 Miniconda Prompt를 이용해 가상환경 만들기

❖ Environment tf36_cpu

(tf36_cpu) C:\Users\Wk8s>python

Python 3.6.13 [Anaconda, Inc.] (default, Mar 16 2021, 11:37:27) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.

>>> import tensorflow as tf

2022-06-25 09:28:41.566404: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlerror: cudart64_110.dll not found

2022-06-25 09:28:41.566922: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.

>>> helloworld = tf.constant("Hello World!")

2022-06-25 09:29:37.365717: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'nvcuda.dll'; dlerror: nvcuda.dll not found

2022-06-25 09:29:37.366046: W tensorflow/stream_executor/cuda/cuda_driver.cc:269] failed call to cuInit: UNKNOWN ERROR (303)

2022-06-25 09:29:37.505609: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:169] retrieving CUDA diagnostic information for host: DESKTOP-R0EQ2U6

2022-06-25 09:29:37.506340: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176] hostname: DESKTOP-R0EQ2U6

2022-06-25 09:29:37.509513: 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: AVX AVX2

To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.

>>> tf.print(helloworld)

Hello World!

>>> quit()

(tf36_cpu) C:\Users\Wk8s>

...

```
(tf36_cpu) C:\Users\Wk8s>pip install ipykernel
```

Downloading ipykernel-5.5.6-py3-none-any.whl (121 kB)

121 kB 3.3 MB/s

...

Successfully installed backcall-0.2.0 colorama-0.4.5 decorator-5.1.1 entrypoints-0.4 ipykernel-5.5.6 ipython-7.16.3 ipython-genutils-0.2.0 jedi-0.17.2 jupyter-client-7.1.2 jupyter-core-4.9.2 nest-asyncio-1.5.5 parso-0.7.1 pickleshare-0.7.5 prompt-toolkit-3.0.29 pygments-2.12.0 python-dateutil-2.8.2 pywin32-304 pyzmq-23.2.0 tornado-6.1 traitlets-4.3.3 wcwidth-0.2.5

```
(tf36_cpu) C:\Users\Wk8s>python -m ipykernel install --user --name tf36_cpu --display-name "tf36_cpu"
```

Installed kernelspec tf36_cpu in C:\Users\wk8s\AppData\Roaming\jupyter\kernels\tf36_cpu

```
(tf36_cpu) C:\Users\Wk8s>
```

Windows10에서 Miniconda Prompt를 이용해 가상환경 만들기

❖ Environment tf37_cpu

```
(tf36_cpu) C:\Users\Wk8s>conda deactivate
```

```
(base) C:\Users\Wk8s>conda create --name tf37_cpu python=3.7
```

```
...
```

```
done
```

```
#
```

```
# To activate this environment, use
```

```
#
```

```
# $ conda activate tf37_cpu
```

```
#
```

```
# To deactivate an active environment, use
```

```
#
```

```
# $ conda deactivate
```

```
(base) C:\Users\Wk8s>
```

Windows10에서 Miniconda Prompt를 이용해 가상환경 만들기

❖ Environment tf37_cpu

```
(base) C:\Users\Wk8s>conda activate tf37_cpu
```

```
(tf37_cpu) C:\Users\Wk8s>pip install --upgrade tensorflow
```

```
Collecting tensorflow
```

```
  Downloading tensorflow-2.9.1-cp37-cp37m-win_amd64.whl (444.0 MB)
```

```
    | 69.3 MB 3.3 MB/s eta 0:01:54
```

```
...
```

```
Successfully built termcolor
```

```
Installing collected packages: urllib3, pyasn1, idna, charset-normalizer, zipp, typing-extensions, six, rsa, requests, pyasn1-modules, oauthlib, cachetools, requests-oauthlib, importlib-metadata, google-auth, werkzeug, tensorboard-plugin-wit, tensorboard-data-server, pyparsing, protobuf, numpy, markdown, grpcio, google-auth-oauthlib, absl-py, wrapt, termcolor, tensorflow-io-gcs-filesystem, tensorflow-estimator, tensorboard, packaging, opt-einsum, libclang, keras-preprocessing, keras, h5py, google-pasta, gast, flatbuffers, astunparse, tensorflow
```

```
Successfully installed absl-py-1.1.0 astunparse-1.6.3 cachetools-5.2.0 charset-normalizer-2.0.12 flatbuffers-1.12 gast-0.4.0 google-auth-2.8.0 google-auth-oauthlib-0.4.6 google-pasta-0.2.0 grpcio-1.47.0 h5py-3.7.0 idna-3.3 importlib-metadata-4.11.4 keras-2.9.0 keras-preprocessing-1.1.2 libclang-14.0.1 markdown-3.3.7 numpy-1.21.6 oauthlib-3.2.0 opt-einsum-3.3.0 packaging-21.3 protobuf-3.19.4 pyasn1-0.4.8 pyasn1-modules-0.2.8 pyparsing-3.0.9 requests-2.28.0 requests-oauthlib-1.3.1 rsa-4.8 six-1.16.0 tensorboard-2.9.1 tensorboard-data-server-0.6.1 tensorboard-plugin-wit-1.8.1 tensorflow-2.9.1 tensorflow-estimator-2.9.0 tensorflow-io-gcs-filesystem-0.26.0 termcolor-1.1.0 typing-extensions-4.2.0 urllib3-1.26.9 werkzeug-2.1.2 wrapt-1.14.1 zipp-3.8.0
```

```
(tf37_cpu) C:\Users\Wk8s>
```



```
(tf37 cpu) C:\Users\Wk8s>python
```

```
>>> import tensorflow as tf
```

2022-06-25 09:49:19.036816: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.

```
>>> helloworld = tf.constant("Hello World!")
```

2022-06-25 09:50:06.185879: W tensorflow/stream_executor/cuda/cuda_driver.cc:269] failed call to cuInit: UNKNOWN ERROR (303)

2022-06-25 09:50:06.194635: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:169] retrieving CUDA diagnostic information for host: DESKTOP-R0EQ2U6

2022-06-25 09:50:06.195290: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176] hostname: DESKTOP-R0EQ2U6

2022-06-25 09:50:06.197010: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX AVX2

To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.

```
>>> tf.print(helloworld)
```

Hello World!

```
>>> quit()
```

```
(tf37_cpu) C:\Users\Wk8s>pip install ipykernel
```

Collecting ipykernel

Downloading ipykernel-6.15.0-py3-none-any.whl (133 kB)

133 kB 2.2 MB/s

```
(tf37_cpu) C:\Users\Wk8s>python -m ipykernel install --user --name tf37_cpu --display-name "tf37_cpu"
```

Installed kernelspec tf37_cpu in C:\Users\Wk8s\AppData\Roaming\jupyter\kernels\tf37_cpu

```
(tf37_cpu) C:\Users\Wk8s>
```

❖ Environment tf38_cpu

(tf37_cpu) C:\Users\k8s>conda deactivate

(base) C:\Users\k8s>conda create -n tf38_cpu pip python=3.8

...

(base) C:\Users\k8s>conda activate tf38_cpu

(tf38_cpu) C:\Users\k8s>pip install --ignore-installed --upgrade tensorflow-cpu

Collecting tensorflow-cpu

Downloading tensorflow-cpu-2.9.1-cp38-cp38-win_amd64.whl (256.3 MB)

...

Successfully built termcolor

Installing collected packages: urllib3, pyasn1, idna, charset-normalizer, certifi, zipp, six, rsa, requests, pyasn1-modules, oauthlib, cachetools, requests-oauthlib, importlib-metadata, google-auth, wheel, werkzeug, tensorboard-plugin-wit, tensorboard-data-server, setuptools, pyparsing, protobuf, numpy, markdown, grpcio, google-auth-oauthlib, absl-py, wrapt, typing-extensions, termcolor, tensorflow-io-gcs-filesystem, tensorflow-estimator, tensorboard, packaging, opt-einsum, libclang, keras-preprocessing, keras, h5py, google-pasta, gast, flatbuffers, astunparse, tensorflow-cpu

Successfully installed absl-py-1.1.0 astunparse-1.6.3 cachetools-5.2.0 certifi-2022.6.15 charset-normalizer-2.0.12 flatbuffers-1.12 gast-0.4.0 google-auth-2.8.0 google-auth-oauthlib-0.4.6 google-pasta-0.2.0 grpcio-1.47.0 h5py-3.7.0 idna-3.3 importlib-metadata-4.11.4 keras-2.9.0 keras-preprocessing-1.1.2 libclang-14.0.1 markdown-3.3.7 numpy-1.23.0 oauthlib-3.2.0 opt-einsum-3.3.0 packaging-21.3 protobuf-3.19.4 pyasn1-0.4.8 pyasn1-modules-0.2.8 pyparsing-3.0.9 requests-2.28.0 requests-oauthlib-1.3.1 rsa-4.8 setuptools-62.6.0 six-1.16.0 tensorboard-2.9.1 tensorboard-data-server-0.6.1 tensorboard-plugin-wit-1.8.1 tensorflow-cpu-2.9.1 tensorflow-estimator-2.9.0 tensorflow-io-gcs-filesystem-0.26.0 termcolor-1.1.0 typing-extensions-4.2.0 urllib3-1.26.9 werkzeug-2.1.2 wheel-0.37.1 wrapt-1.14.1 zipp-3.8.0

(tf38_cpu) C:\Users\k8s>

❖ Environment tf38_cpu

(tf38_cpu) C:\Users\k8s>python

Python 3.8.13 (default, Mar 28 2022, 06:59:08) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32

Type "help", "copyright", "credits" or "license" for more information.

>>> import tensorflow as tf

>>> helloworld = tf.constant("Hello World!")

2022-06-25 10:04:47.613593: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX AVX2

To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.

>>> tf.print(helloworld)

Hello World!

>>> quit()

(tf38_cpu) C:\Users\k8s>

❖ Environment tf38_cpu

(tf38_cpu) C:\Users\k8s>pip install ipykernel

Collecting ipykernel

Using cached ipykernel-6.15.0-py3-none-any.whl (133 kB)

Collecting ipython>=7.23.1

Downloading ipython-8.4.0-py3-none-any.whl (750 kB)

...

(tf38_cpu) C:\Users\k8s>python -m ipykernel install --user --name tf38_cpu --display-name "tf38_cpu"

Installed kernelspec tf38_cpu in C:\Users\k8s\AppData\Roaming\jupyter\kernels\tf38_cpu

(tf38_cpu) C:\Users\k8s>

Jupyter Notebook

(base) C:\Users\k8s>jupyter notebook

```
[I 10:15:08.860 NotebookApp] Writing notebook server cookie secret to C:\Users\k8s\AppData\Roaming\jupyter\runtime\notebook_cookie_secret
[I 10:15:10.203 NotebookApp] Serving notebooks from local directory: C:\Users\k8s
[I 10:15:10.203 NotebookApp] Jupyter Notebook 6.4.12 is running at:
[I 10:15:10.203 NotebookApp] http://localhost:8888/?token=0633164bc72ce7ae73179310241e5c968e1576c718461681
[I 10:15:10.203 NotebookApp] or http://127.0.0.1:8888/?token=0633164bc72ce7ae73179310241e5c968e1576c718461681
[I 10:15:10.203 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 10:15:10.344 NotebookApp]
```

To access the notebook, open this file in a browser:

file:///C:/Users/k8s/AppData/Roaming/jupyter/runtime/nbserver-8528-open.html

Or copy and paste one of these URLs:

http://localhost:8888/?token=0633164bc72ce7ae73179310241e5c968e1576c718461681

or http://127.0.0.1:8888/?token=0633164bc72ce7ae73179310241e5c968e1576c718461681

❖ 가상환경 커널 tf38_cpu이 잘 보임

The screenshot shows the Jupyter Notebook interface. At the top, there's a 'jupyter' logo and 'Quit' and 'Logout' buttons. Below the logo, there are tabs for 'Files', 'Running', and 'Clusters'. The 'Files' tab is active, showing a file browser. The file browser has a header with 'Select items to perform actions on them.' and buttons for 'Upload', 'New', and a refresh icon. A dropdown menu is open under the 'New' button, showing options for 'Notebook:', 'Python 3 (ipykernel)', 'tf2_38', 'tf38_cpu', 'Other:', 'Text File', 'Folder', and 'Terminal'. The 'tf38_cpu' option is highlighted with a red box. The file browser lists various folders and files, including '3D Objects', 'ansel', 'Contacts', 'Documents', 'Downloads', 'Favorites', 'Links', 'Music', 'nodeserver', 'OneDrive', 'OneDrive - (주)엘케이랩', 'Saved Games', 'Searches', 'Videos', 'Untitled.ipynb', and 'Untitled1.ipynb'. The 'Untitled.ipynb' file is selected, and its details are shown on the right: '6달 전' and '1.51 kB'.

Name	Modified	Size
3D Objects	일 년 전	
ansel	일 년 전	
Contacts	3달 전	
Documents	10달 전	
Downloads	10달 전	
Favorites	33분 전	
Links	일 년 전	
Music	일 년 전	
nodeserver	일 년 전	
OneDrive	6달 전	1.51 kB
OneDrive - (주)엘케이랩	4달 전	72 B
Saved Games		
Searches		
Videos		
Untitled.ipynb		
Untitled1.ipynb		

❖ 가상환경 커널 tf38_cpu에서 텐서플로 코딩

```
In [1]: import tensorflow as tf
```

```
In [2]: with tf.compat.v1.Session() as sess:  
         helloworld = tf.constant("Hello World!")  
         print(sess.run(helloworld))
```

```
b'Hello World!'
```

```
In [3]: helloworld = tf.constant("Hello World!")  
         tf.print(helloworld)
```

```
Hello World!
```

❖ 소스 코드

```
import tensorflow as tf
```

```
with tf.compat.v1.Session() as sess:
```

```
    helloworld = tf.constant("Hello World!")
```

```
    print(sess.run(helloworld))
```

```
+++++
```

```
helloworld = tf.constant("Hello World!")
```

```
tf.print(helloworld)
```

```
import sys
print("python 버전 : {}".format(sys.version))
import pandas as pd
print("pandas 버전 : {}".format(pd.__version__))
import matplotlib
print("matplotlib 버전 : {}".format(matplotlib.__version__))
import numpy as np
print("numpy 버전 : {}".format(np.__version__))
import scipy as sp
print("scipy 버전 : {}".format(sp.__version__))
import IPython
print("IPython 버전 : {}".format(IPython.__version__))
import sklearn
print("sklearn : {}".format(sklearn.__version__))
```



```
pip install numpy scipy sklearn pandas matplotlib
```

가상환경 삭제

```
conda remove --name 가상환경이름 -all
```

```
conda env remove -n 가상환경이름
```

Miniconda3

Miniconda3\Library\bin

Miniconda3\Scripts

Miniconda3\envs\tf37_cpu\Library\bin

Miniconda3\envs\tf38_cpu\Library\bin

Miniconda3\envs\tf38_gpu\Library\bin

환경 변수 편집

C:\DEV\miniconda3

C:\DEV\miniconda3\Library\mingw-w64\bin

C:\DEV\miniconda3\Library\usr\bin

C:\DEV\miniconda3\Library\bin

C:\DEV\miniconda3\Scripts

%USERPROFILE%\AppData\Local\Microsoft\WindowsApps

C:\Program Files\Bandizip\

%USERPROFILE%\go\bin

C:\DEV\miniconda3\envs\tf36_cpu\Library\bin

C:\DEV\miniconda3\envs\tf37_cpu\Library\bin

C:\DEV\miniconda3\envs\tf38_cpu\Library\bin

새로 만들기(N)

편집(E)

찾아보기(B)...

삭제(D)

위로 이동(U)

아래로 이동(O)

텍스트 편집(T)...

확인

취소

Subsection 2



Flask 개발 환경 만들기

(base) C:\Users\Wk8s>conda create -n flask_37 python=3.7

Collecting package metadata (current_repodata.json): done

Solving environment: done

...

done

#

To activate this environment, use

#

\$ conda activate flask_37

#

To deactivate an active environment, use

#

\$ conda deactivate

(base) C:\Users\Wk8s>conda activate flask_37

(flask_37) C:\Users\Wk8s>conda list

packages in environment at C:\DEV\miniconda3\envs\flask_37:

#

# Name	Version	Build	Channel
ca-certificates	2022.4.26	haa95532_0	
certifi	2022.6.15	py37haa95532_0	
openssl	1.1.1o	h2bbff1b_0	
pip	21.2.4	py37haa95532_0	
python	3.7.13	h6244533_0	
setuptools	61.2.0	py37haa95532_0	
sqlite	3.38.5	h2bbff1b_0	
vc	14.2	h21ff451_1	
vs2015_runtime	14.27.29016	h5e58377_2	
wheel	0.37.1	pyhd3eb1b0_0	
wincertstore	0.2	py37haa95532_2	

(flask_37) C:\Users\Wk8s>

<https://code.visualstudio.com/download>

Download Visual Studio Code

Free and built on open source. Integrated Git, debugging and extensions.

Windows
Windows 8, 10, 11

User Installer 64 bit 32 bit ARM
System Installer 64 bit 32 bit ARM
.zip 64 bit 32 bit ARM

Linux
Debian, Ubuntu Red Hat, Fedora, SUSE

.deb 64 bit ARM ARM 64
.rpm 64 bit ARM ARM 64
.tar.gz 64 bit ARM ARM 64
Snap Store

Mac
macOS 10.11+

.zip Universal Intel Chip Apple Silicon

Windows 정품 인증
[설정]으로 이동하여 Windows를 정품 인증합니다.
모두 표시

User Installer, System Installer, zip 압축파일이 배포되고 있습니다.

User Installer의 경우 다음 위치에 설치되며 유저 인터페이스의 디폴트 언어가 영어가 됩니다.

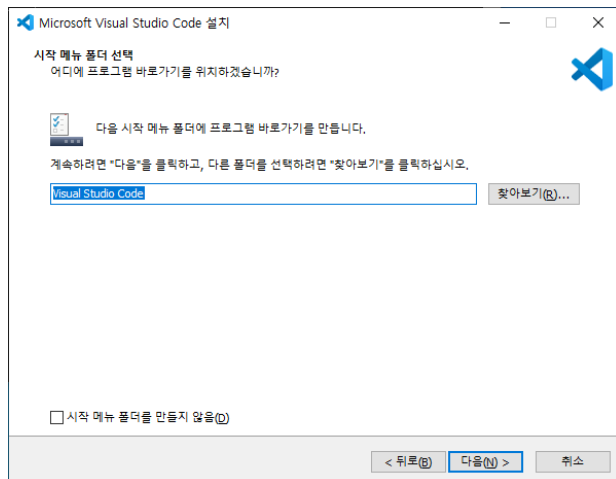
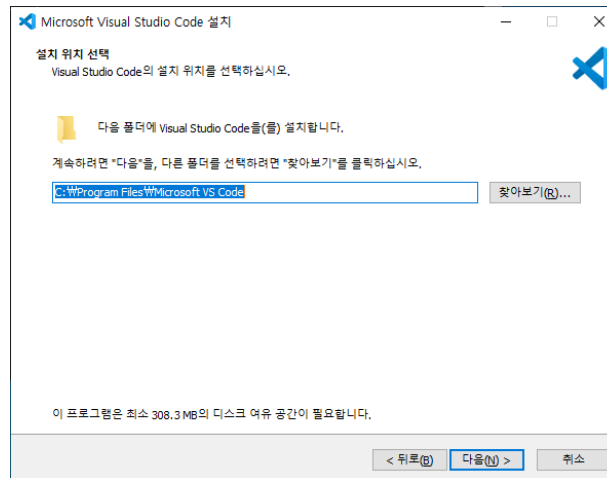
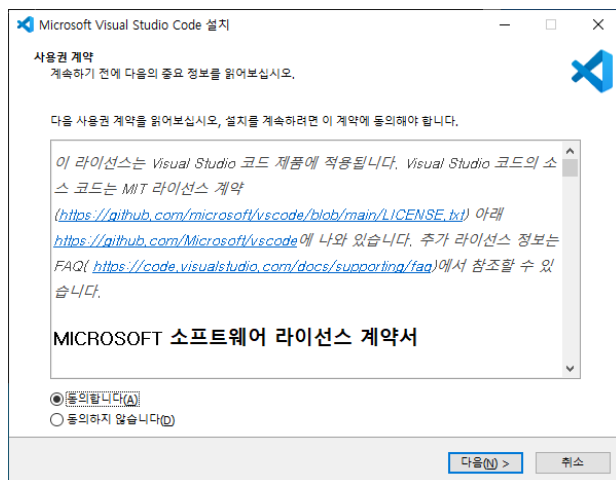
C:\Users\사용자이름\AppData\Local\Programs\Microsoft VS Code

System Installer의 경우에는 다음 위치에 설치되며 유저 인터페이스의 디폴트 언어가 영어가 됩니다.

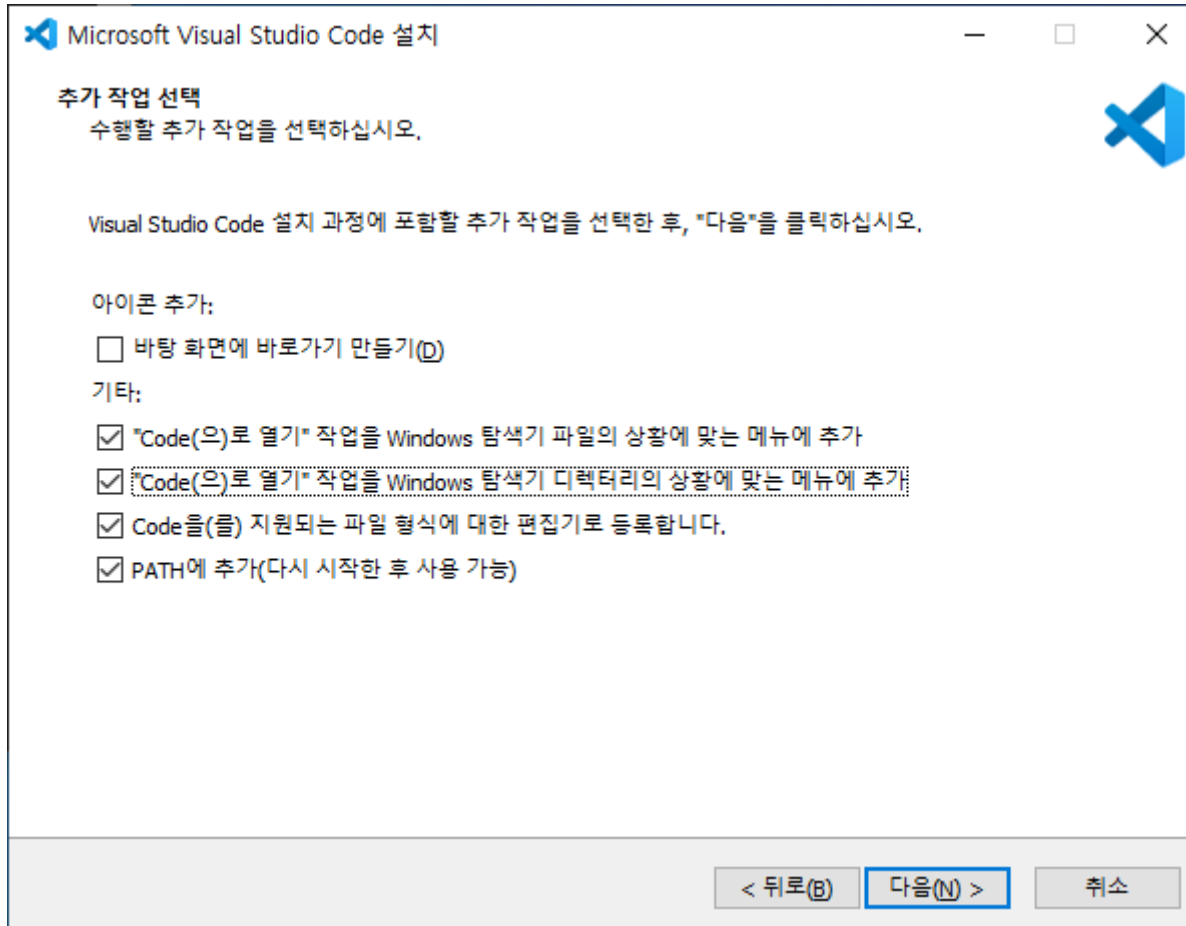
C:\Program Files\Microsoft VS Code

System Installer를 설치

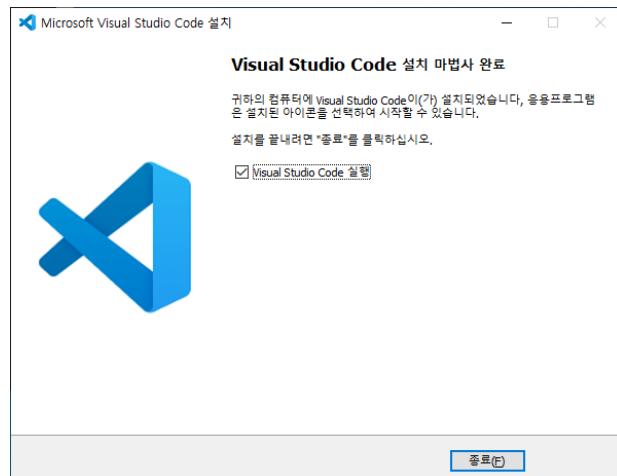
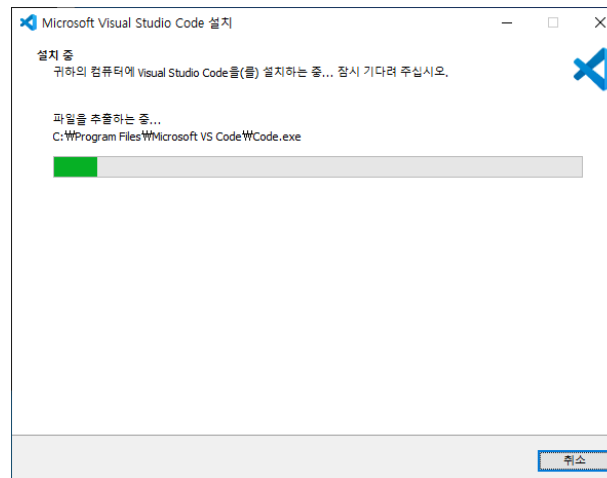
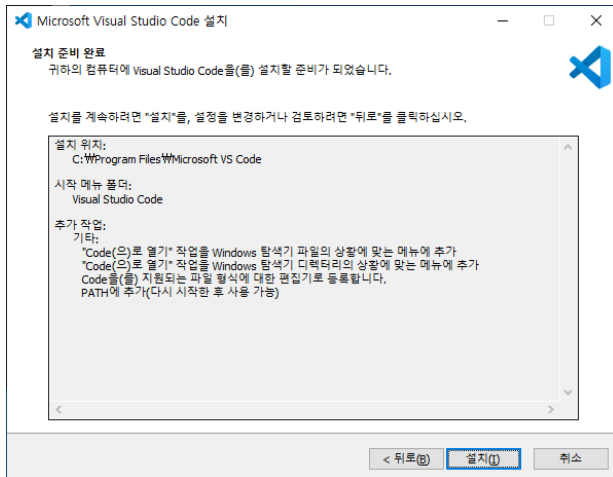
Visual Studio Code 설치



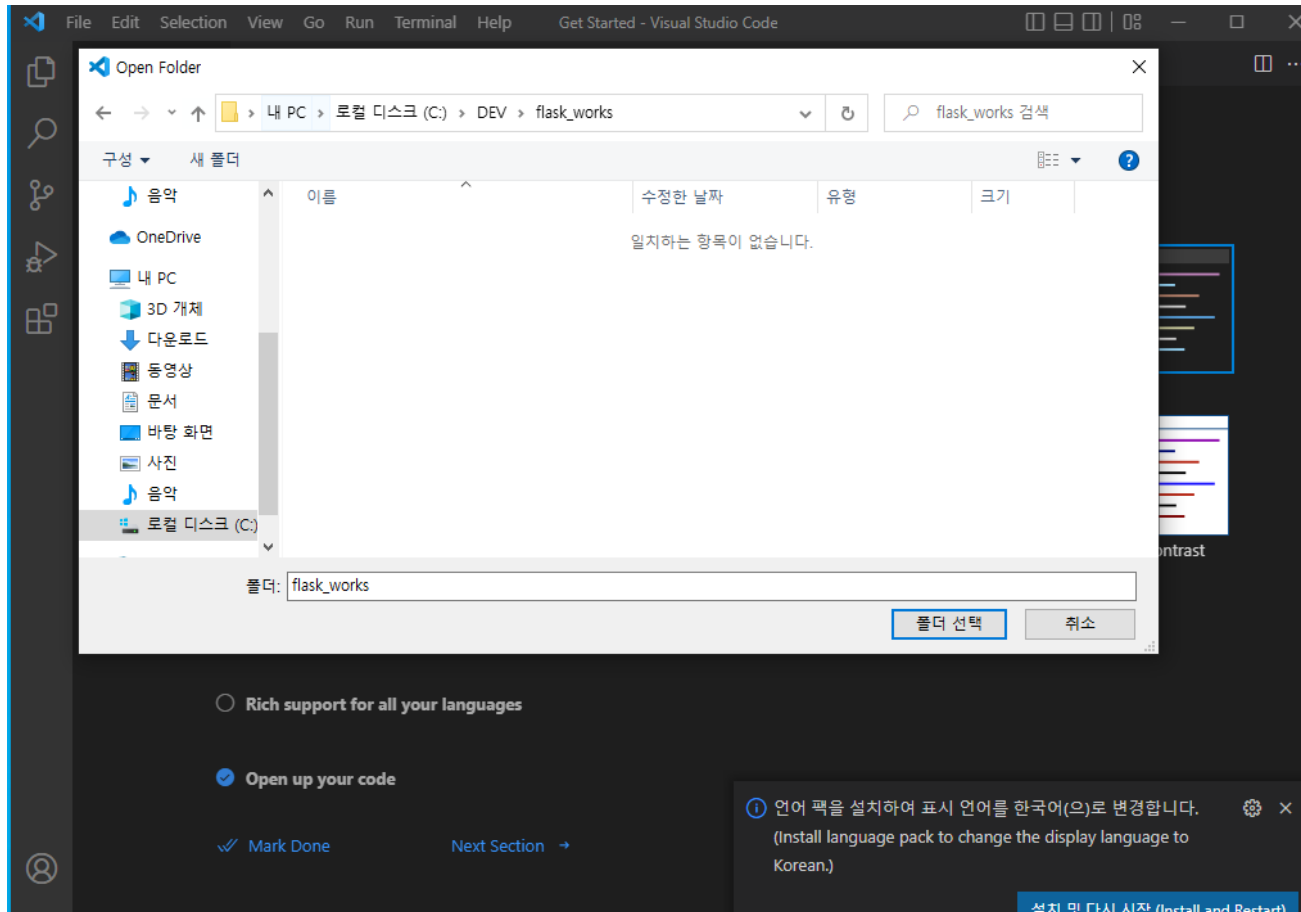
Visual Studio Code 설치



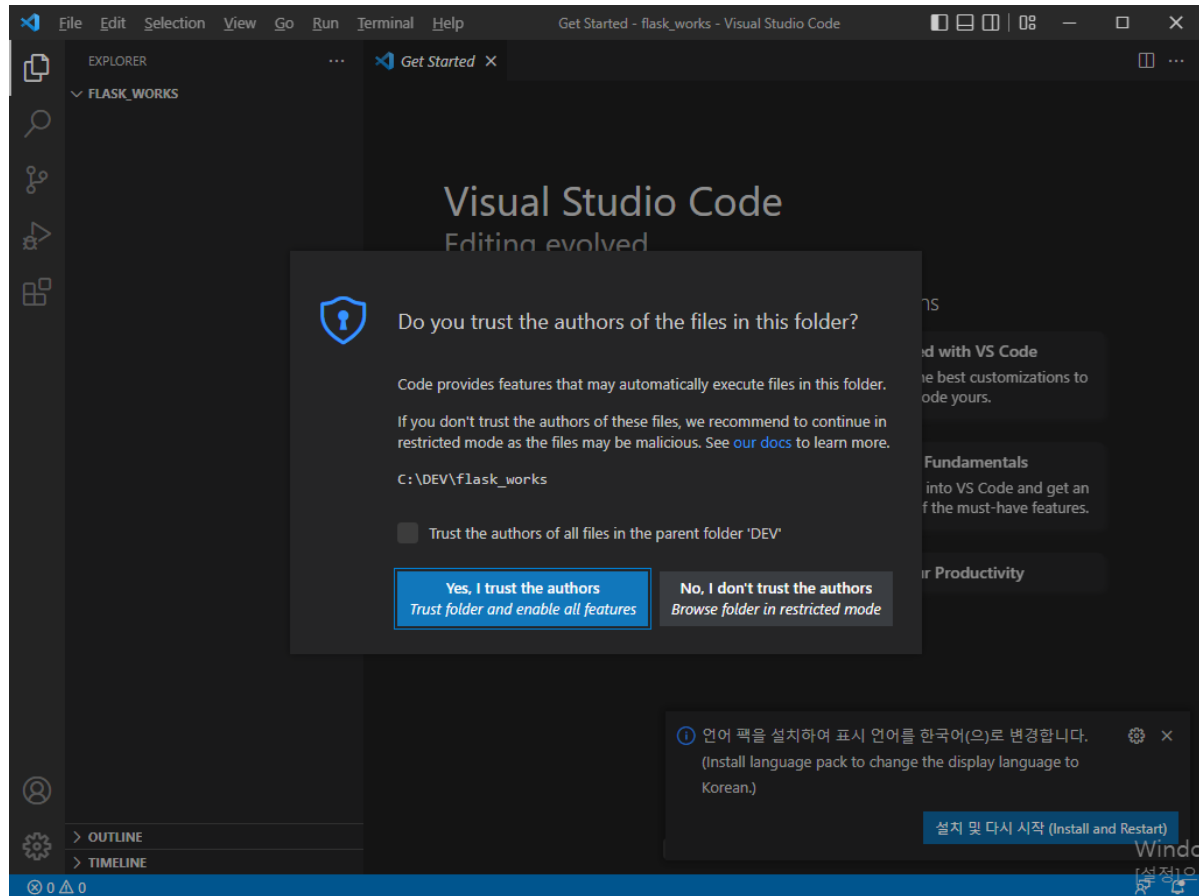
Visual Studio Code 설치



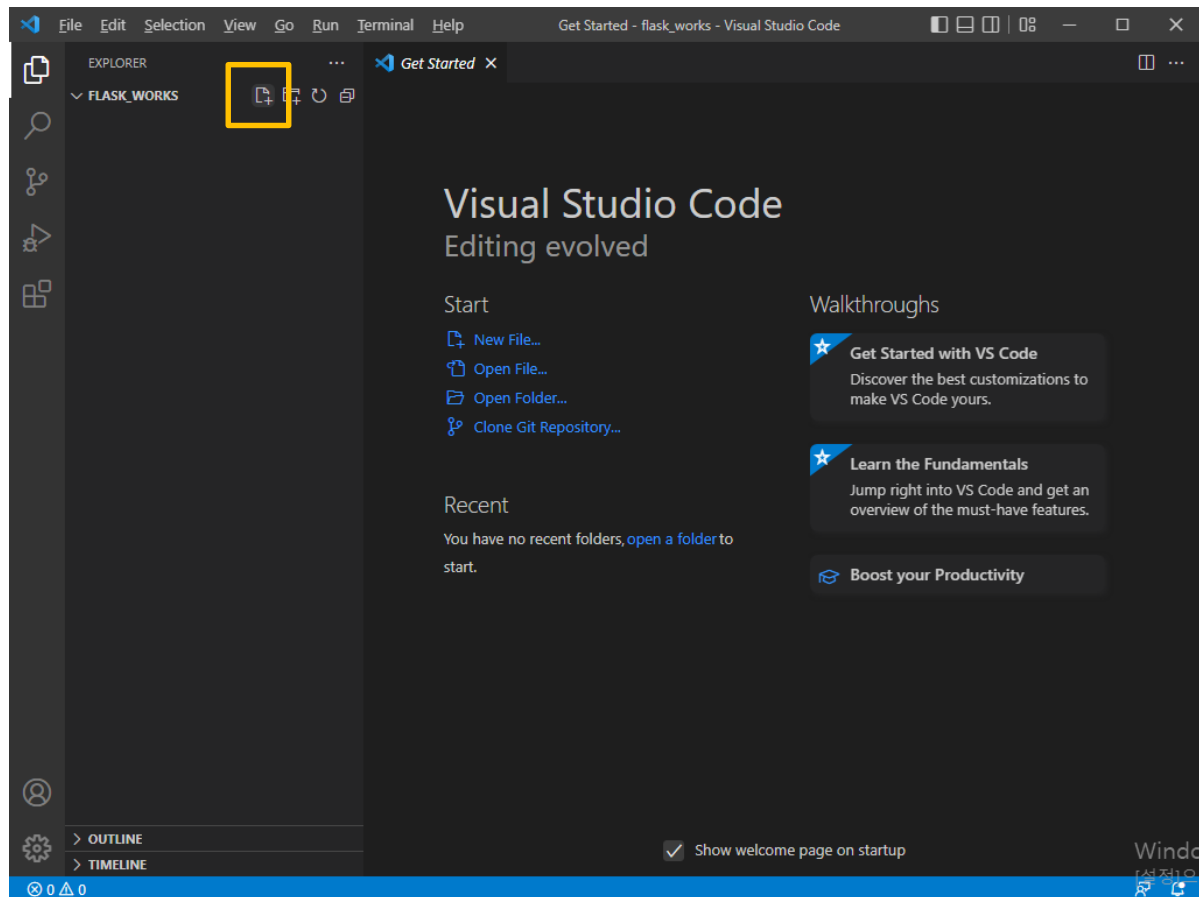
❖ Visual Studio Code를 실행하여 메뉴에서 File > Open Folder를 선택



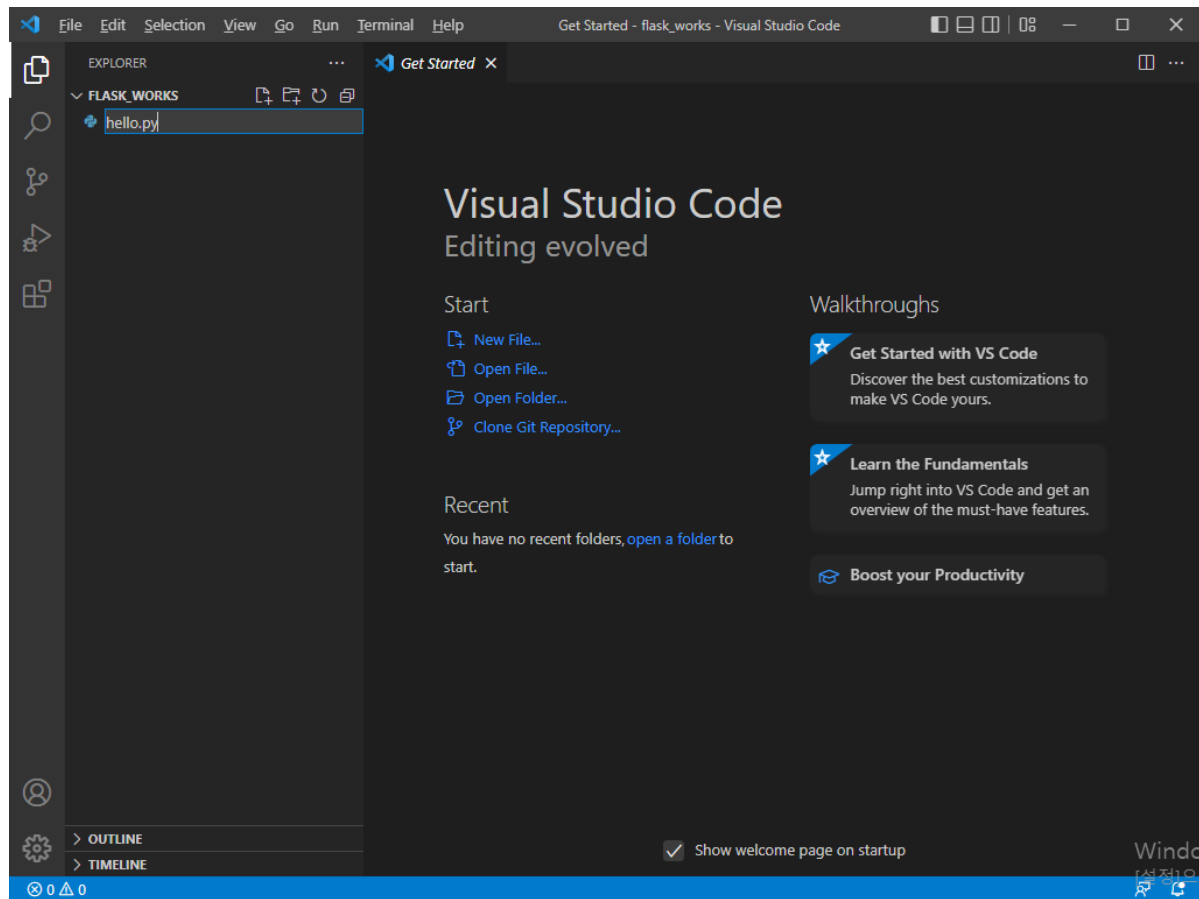
❖ Visual Studio Code를 실행하여 메뉴에서 File > Open Folder를 선택



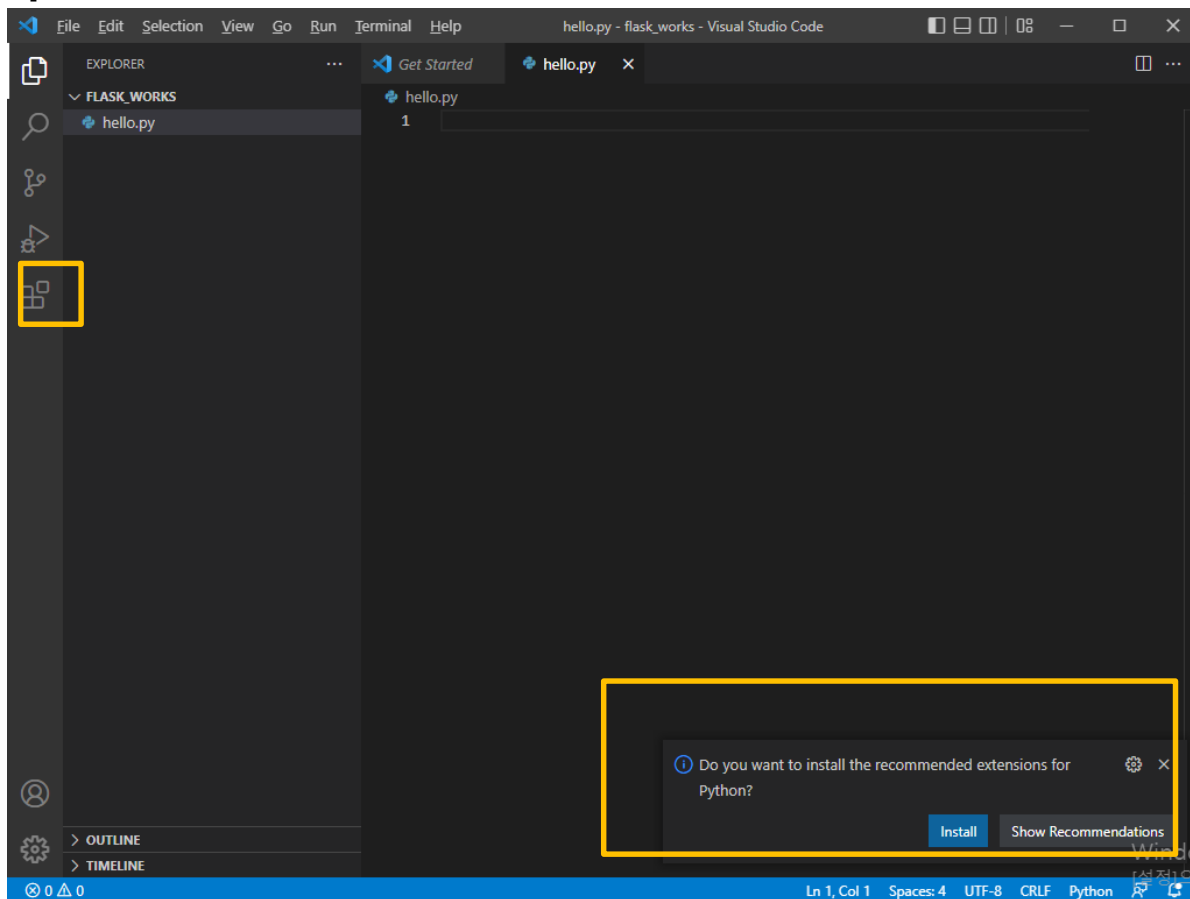
- ❖ 해당 폴더가 Visual Studio Code 왼쪽에 보이게 됩니다. New File 아이콘을 클릭



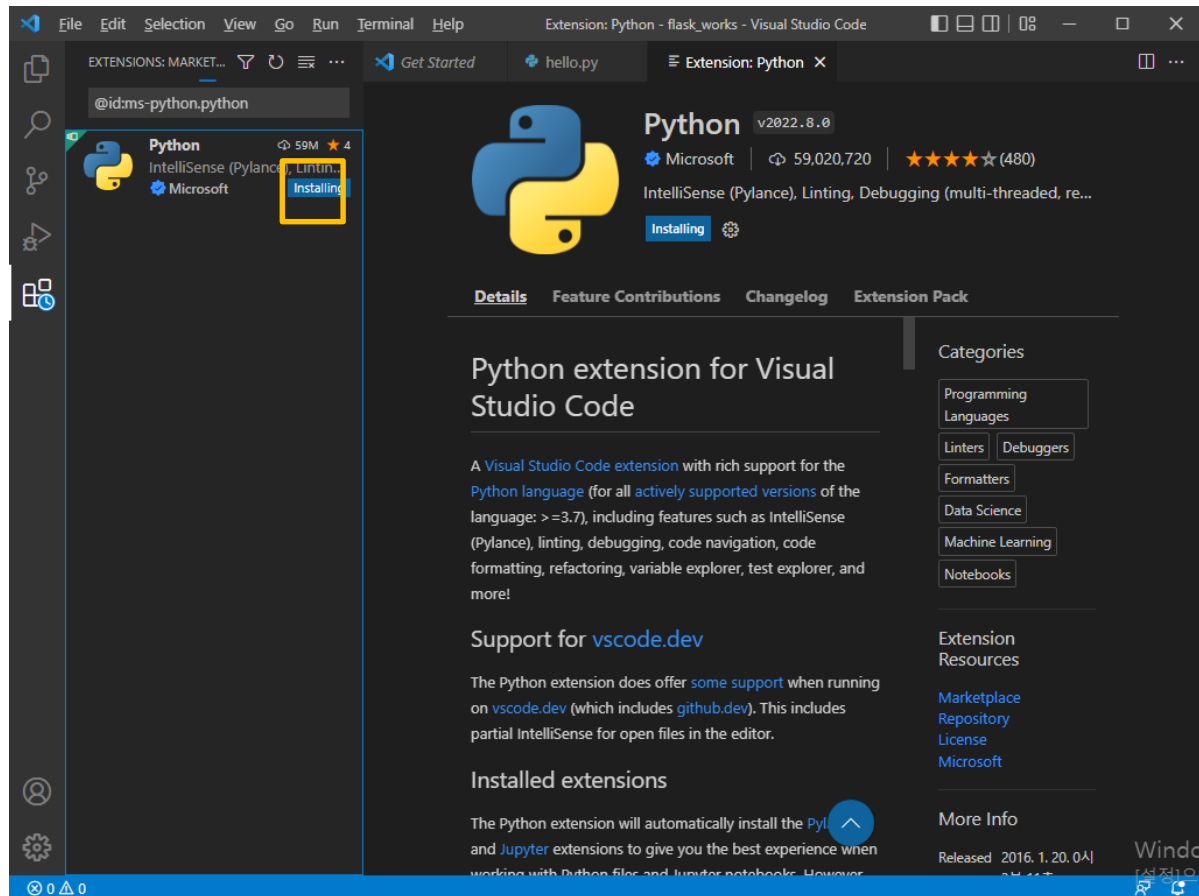
- ❖ **hello.py**를 입력 후, 엔터를 누르면 해당 파일이 추가되면서 오른쪽에 **hello.py** 파일이 열리게 됩니다.



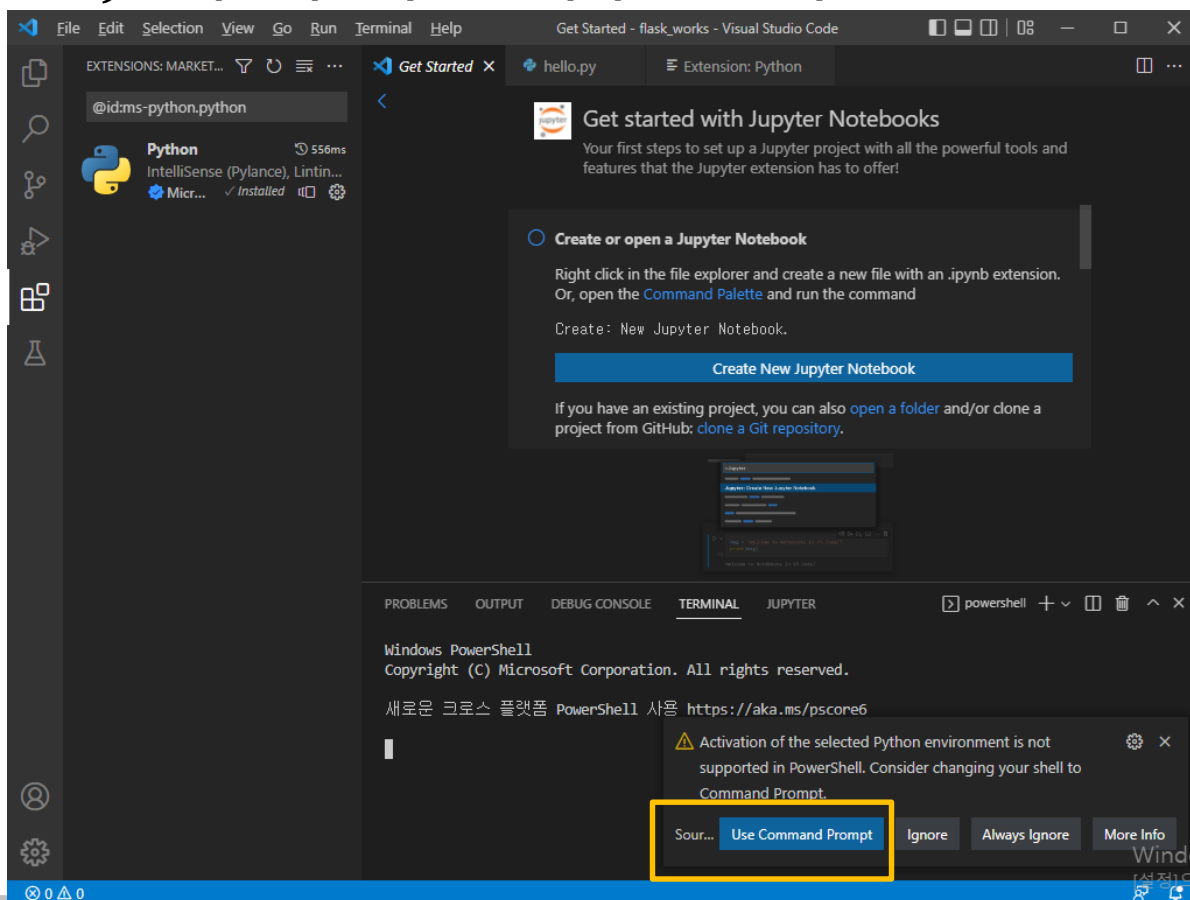
- ❖ 오른쪽 아래에 Python 확장을 설치하는지 물어보는 메시지 박스가 보이면 설치



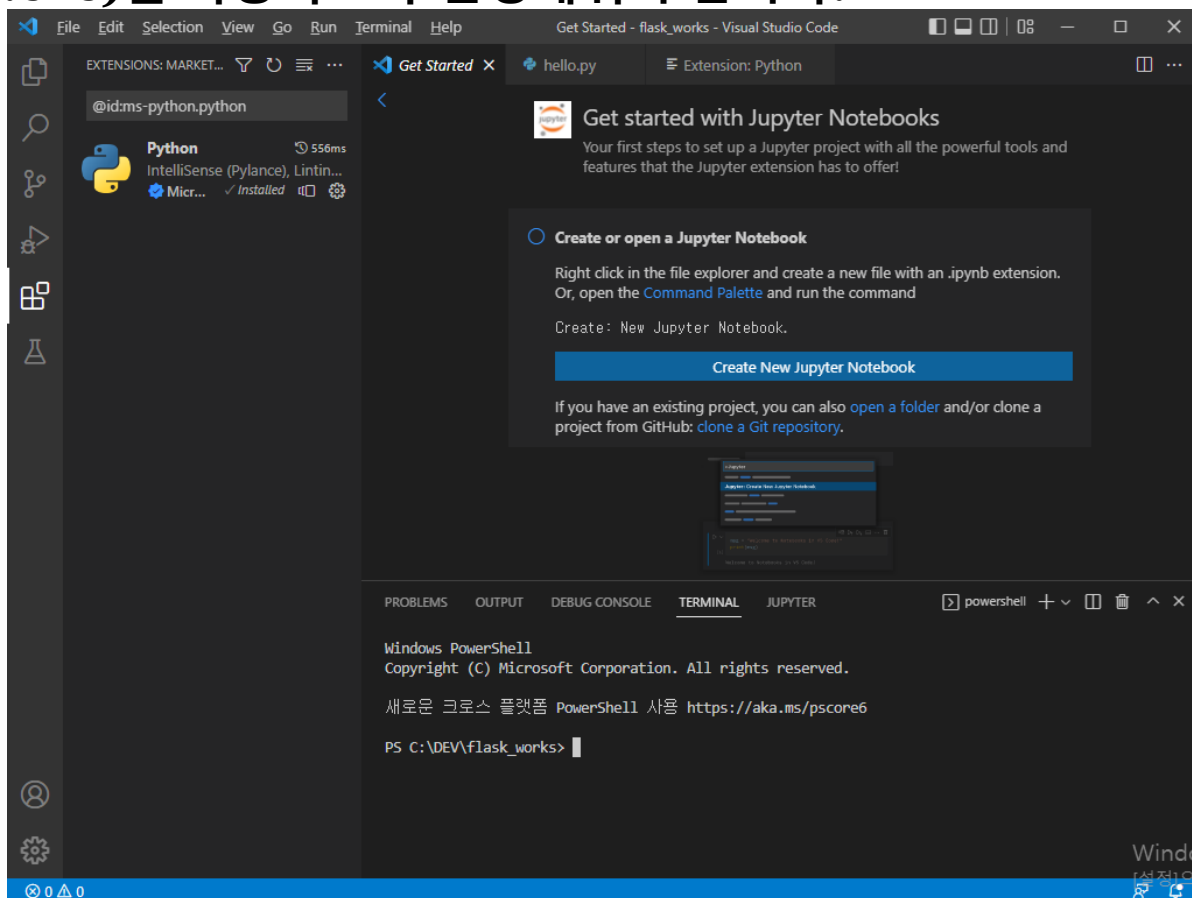
- ❖ 오른쪽 아래에 Python 확장을 설치하는지 물어보는 메시지 박스가 보이면 설치



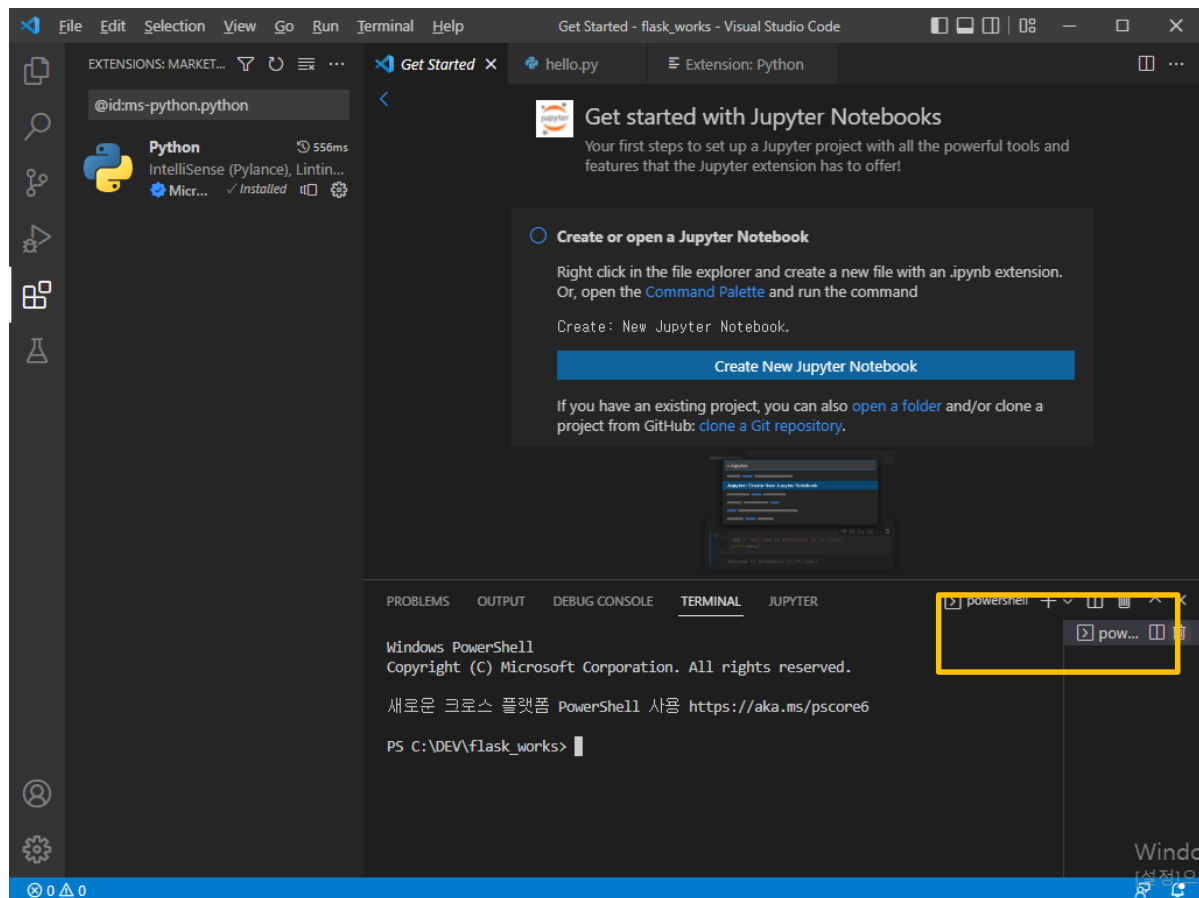
- ❖ 메뉴에서 View > Terminal을 선택합니다. 다음처럼 PowerShell이 실행되면 Conda의 가상환경이 적용되지 않기 때문에 명령 프롬프트(cmd.exe)를 사용하도록 변경해줘야 합니다.



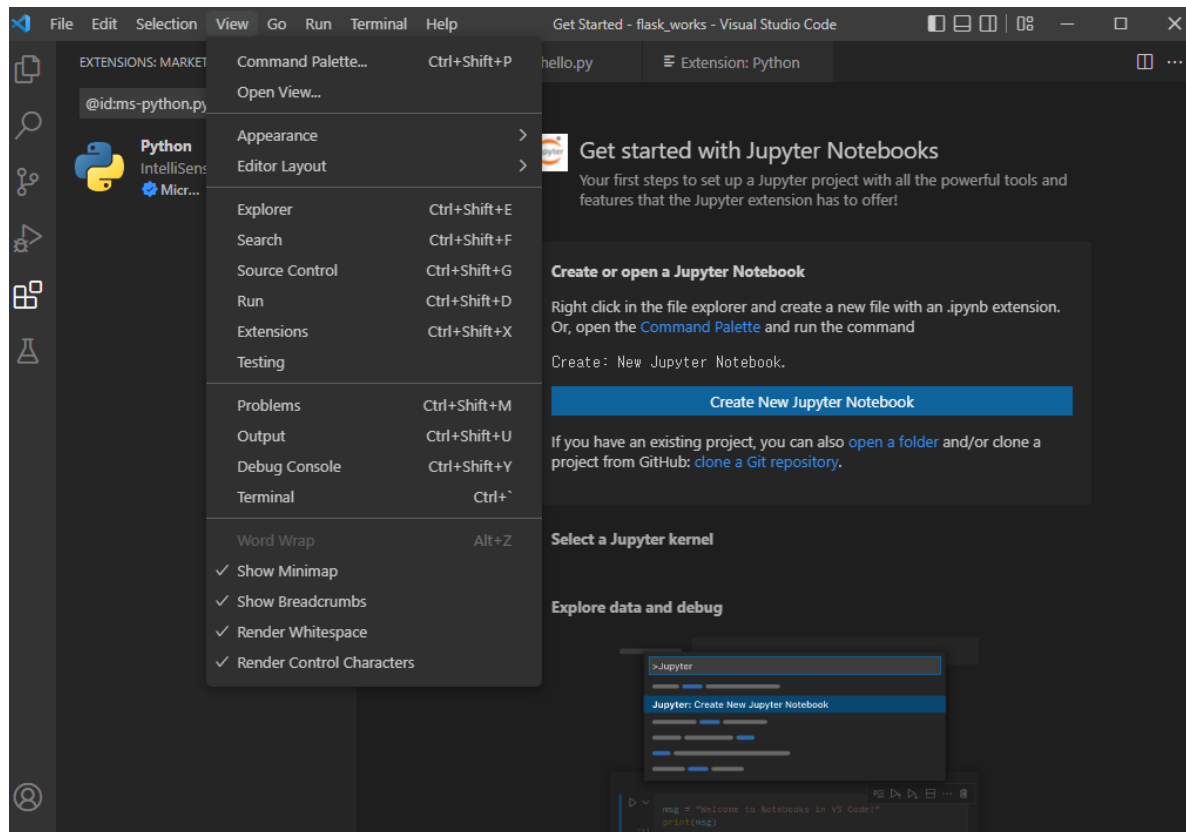
- ❖ 메뉴에서 View > Terminal을 선택합니다. 다음처럼 PowerShell이 실행되면 Conda의 가상환경이 적용되지 않기 때문에 명령 프롬프트 (cmd.exe)를 사용하도록 변경해줘야 합니다.



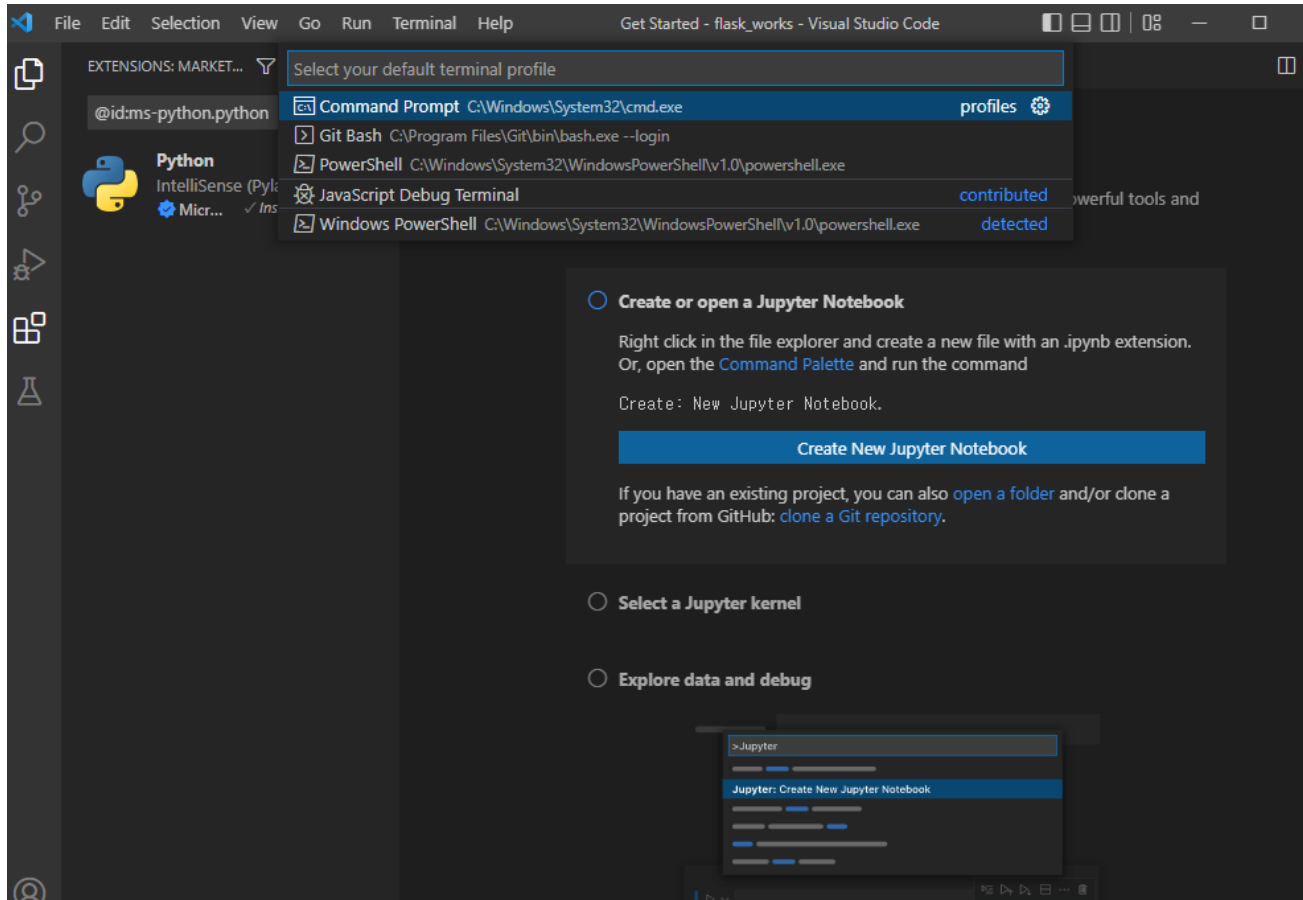
❖ 오른쪽에 보이는 쓰레기통 아이콘을 클릭



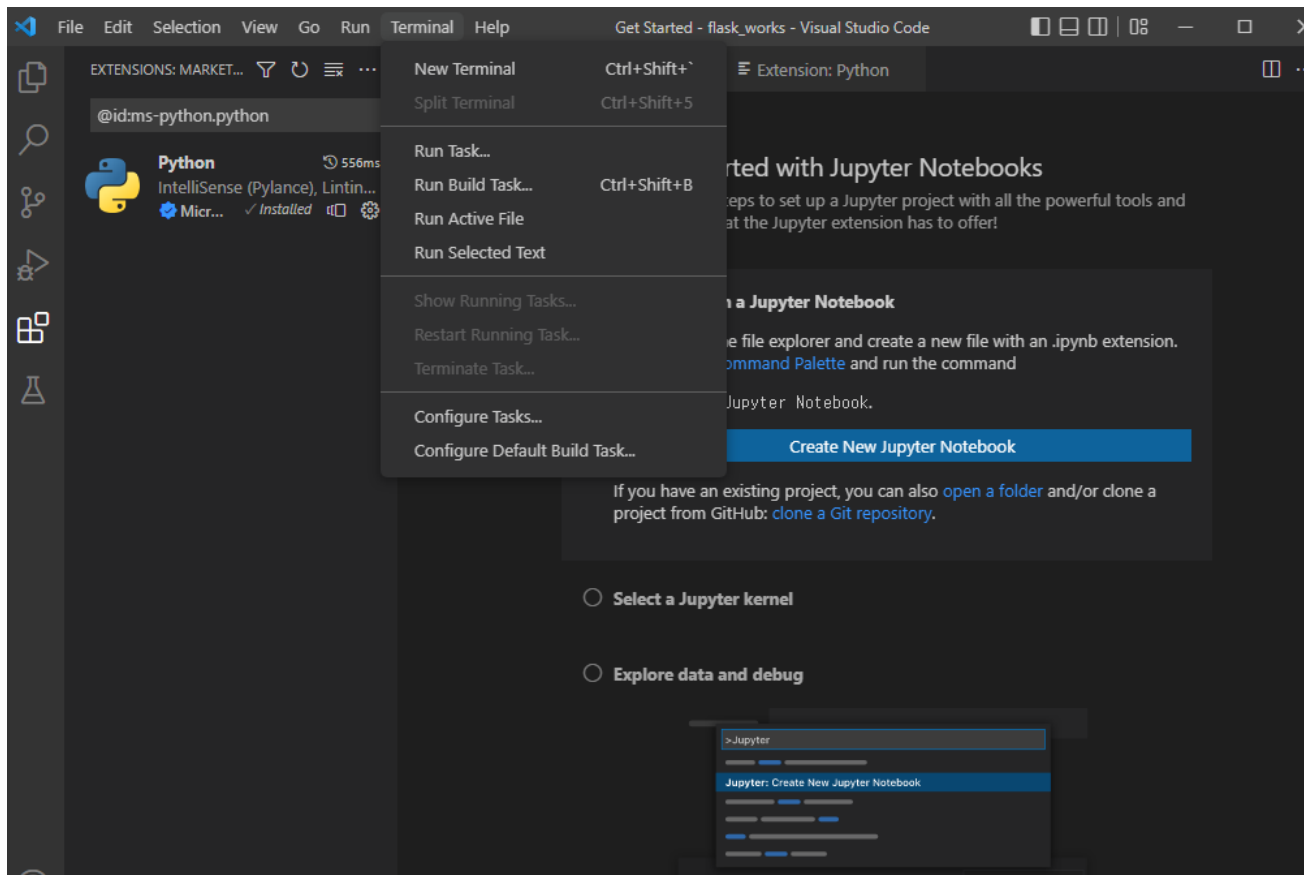
❖ **Ctrl + Shift + P**를 입력한 후, **default profile**을 입력하여 검색되는 항목을 선택



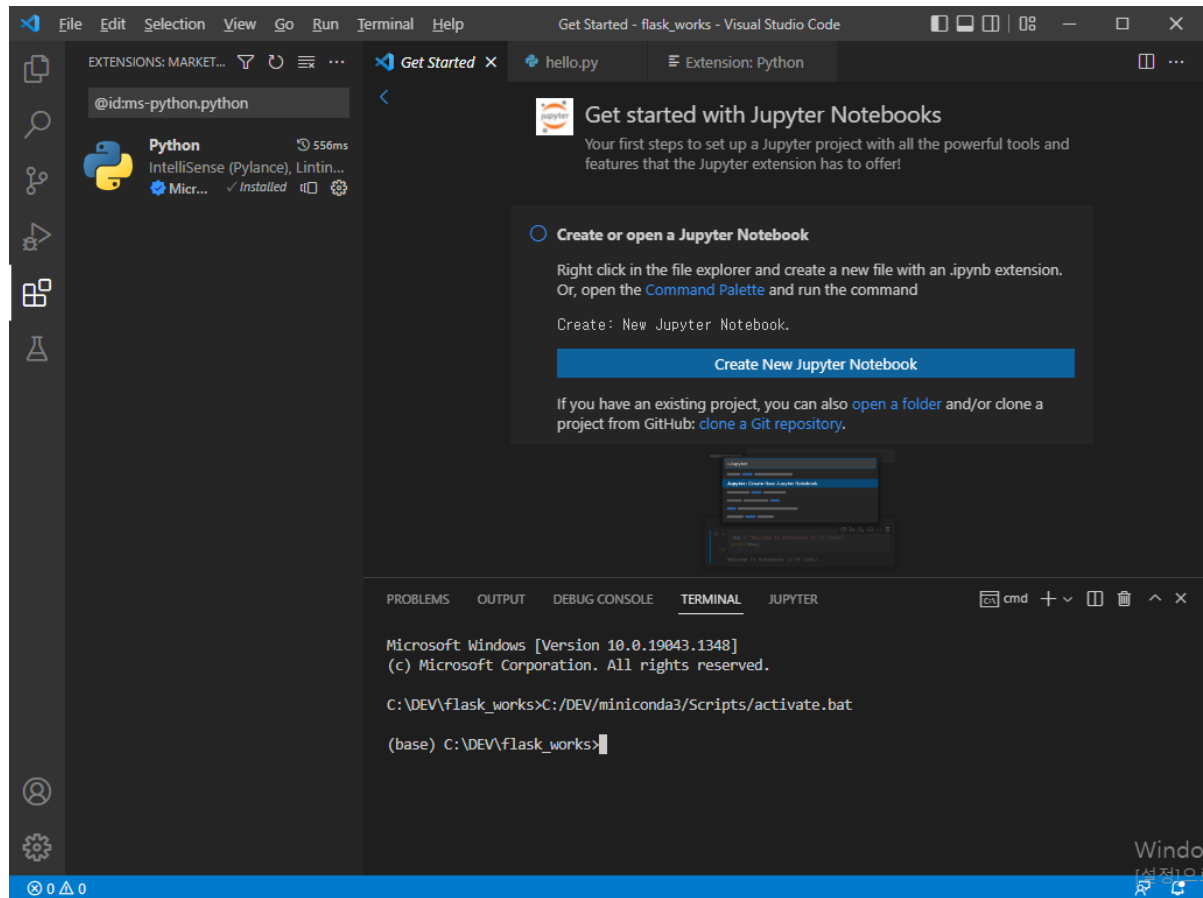
❖ Command Prompt를 선택



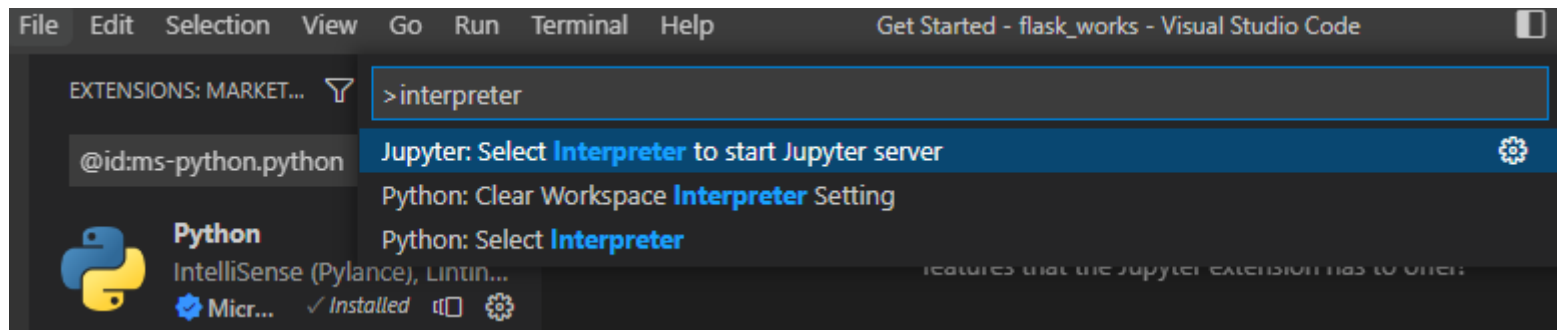
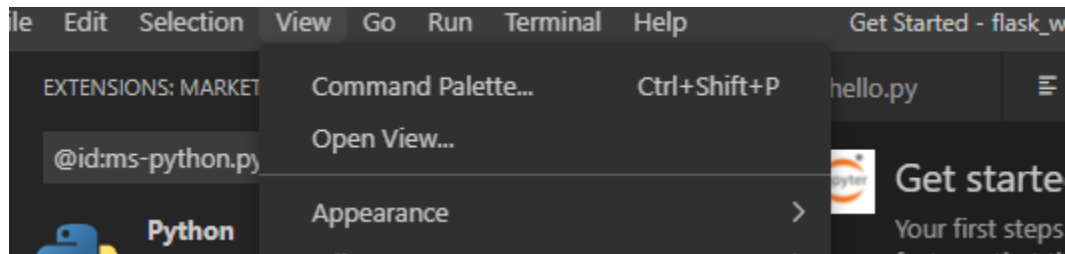
❖ 메뉴에서 Terminal을 선택하면 명령 프롬프트가 실행



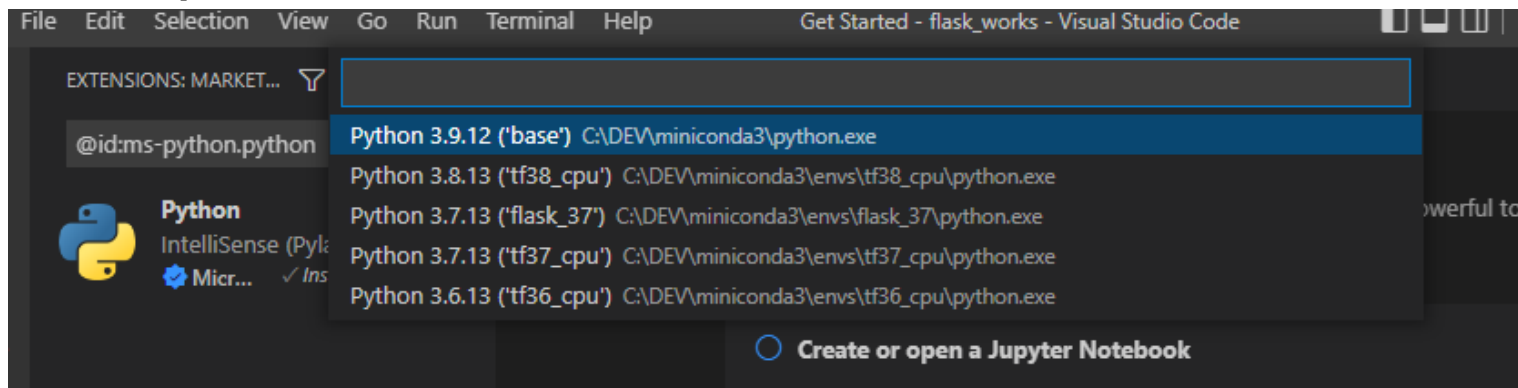
❖ 명령 프롬프트가 실행



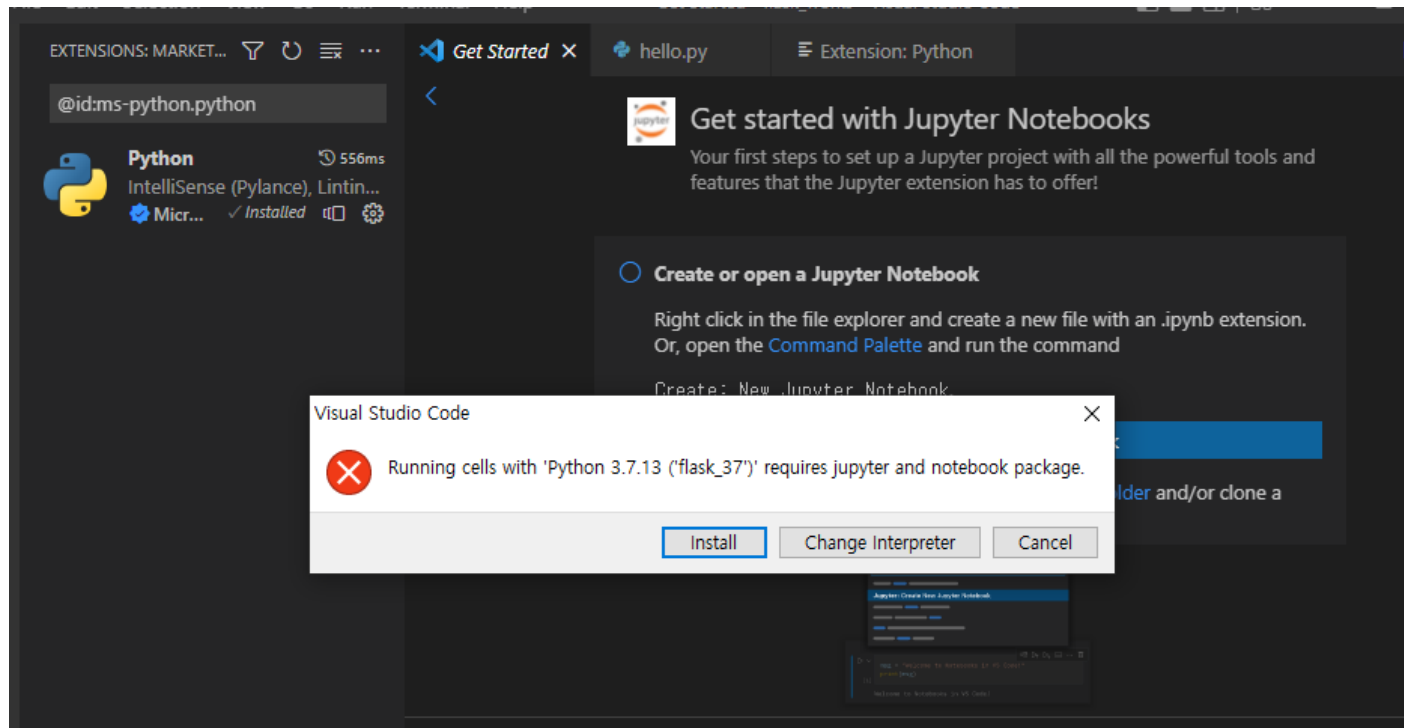
- ❖ Conda에서 생성한 가상환경의 인터프리터로 변경해야 합니다.
- ❖ **Ctrl + Shift + P**를 누른 후, **interpreter**를 입력하여 검색되는 다음 항목을 선택



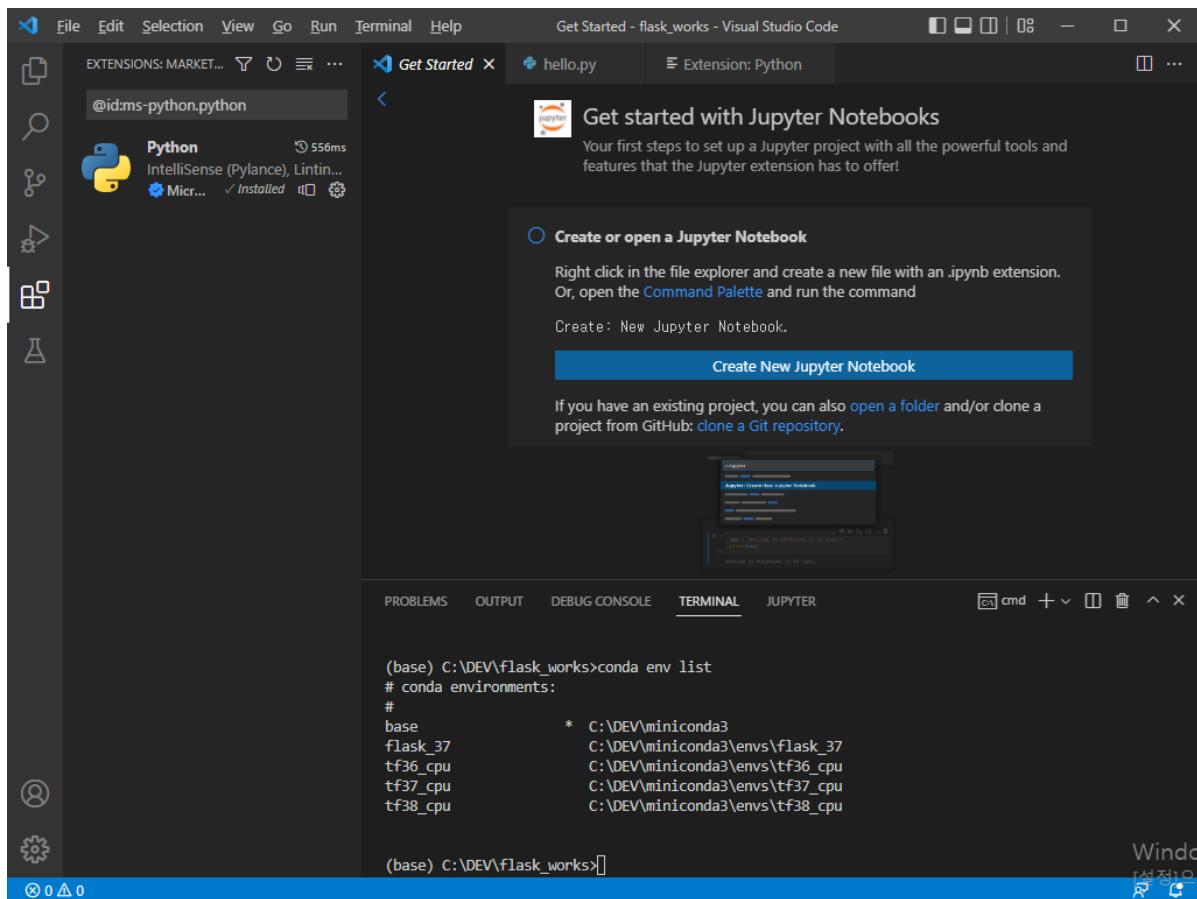
- ❖ Conda에서 생성한 가상환경의 인터프리터로 변경해야 합니다.
- ❖ **Ctrl + Shift + P**를 누른 후, **interpreter**를 입력하여 검색되는 다음 항목을 선택



- ❖ Conda에서 생성한 가상환경의 인터프리터로 변경해야 합니다.
- ❖ **Ctrl + Shift + P**를 누른 후, **interpreter**를 입력하여 검색되는 다음 항목을 선택

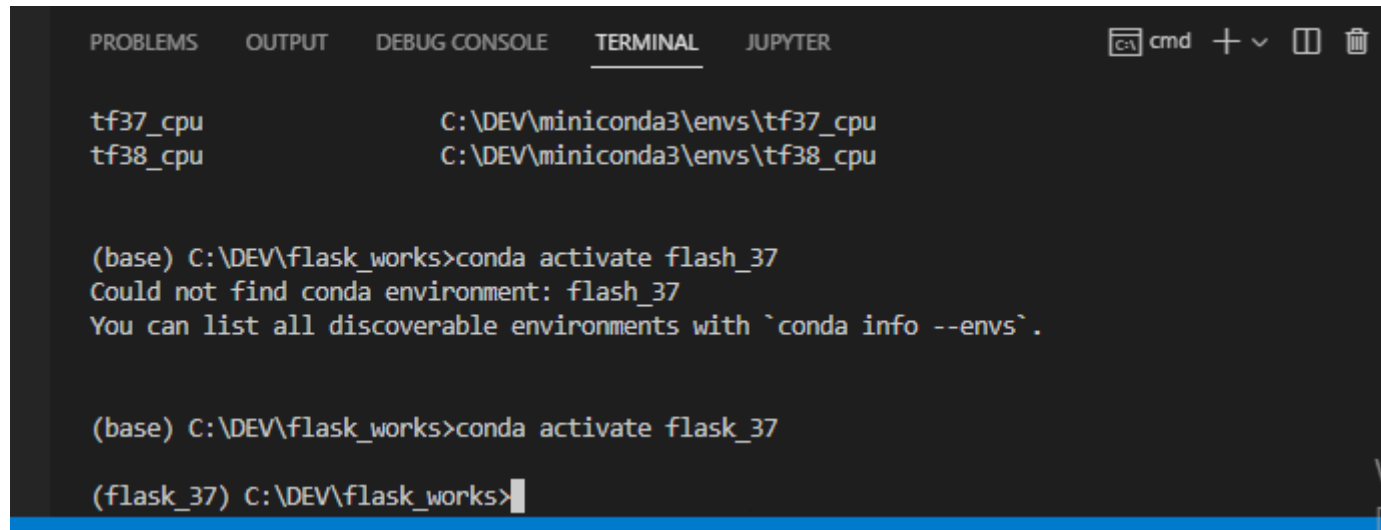


- ❖ 올바르게 인터프리터가 설정이 되었는지 확인해봅니다. 현재 터미널이 열려있다면 오른쪽에 보이는 쓰레기통 아이콘을 클릭하여 닫아줍니다.



```
(base) C:\DEV\flask_works>conda activate flask_37
```

```
(flask_37) C:\DEV\flask_works>
```



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER  cmd + v [ ] [X]

tf37_cpu      C:\DEV\miniconda3\envs\tf37_cpu
tf38_cpu      C:\DEV\miniconda3\envs\tf38_cpu

(base) C:\DEV\flask_works>conda activate flask_37
Could not find conda environment: flask_37
You can list all discoverable environments with `conda info --envs`.

(base) C:\DEV\flask_works>conda activate flask_37

(flask_37) C:\DEV\flask_works>
```

```
(flask_37) C:\DEV\flask_works>
```

❖hello.py 작성

```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello_world():
    return "<p>Hello, World!</p>"
```

❖터미널에서 다음 명령을 실행

(flask_37) C:\DEV\flask_works>dir

C 드라이브의 볼륨에는 이름이 없습니다.

볼륨 일련 번호: BEDo-C858

C:\DEV\flask_works 디렉터리

```
2022-06-25 오후 03:46 <DIR>      .
2022-06-25 오후 03:46 <DIR>      ..
2022-06-25 오후 03:16          o hello.py
2022-06-25 오후 03:46 <DIR>      __pycache__
          1개 파일          0 바이트
          3개 디렉터리 12,450,320,384 바이트 남음
```

(flask_37) C:\DEV\flask_works>set FLASK_APP=hello

(flask_37) C:\DEV\flask_works>flask run

* Serving Flask app 'hello' (lazy loading)

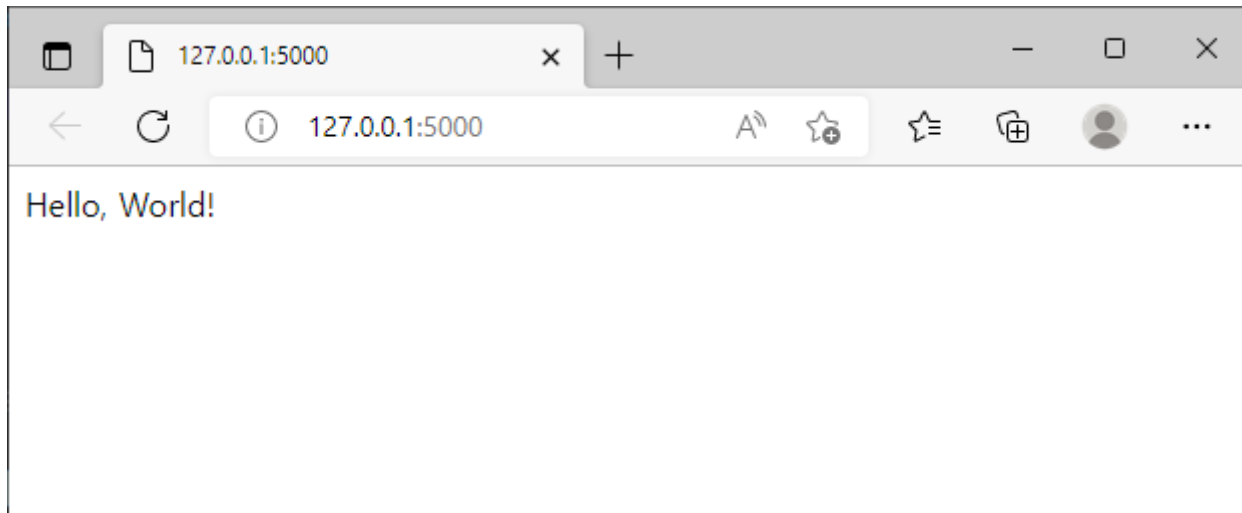
* Environment: production

WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.

* Debug mode: off

* Running on http://127.0.0.1:5000 (Press CTRL+C to quit)

❖ 127.0.0.1:5000



Section 3



Data Ingestion, Data Acquisition

1. **library**
2. **Data Ingestion, Data Acquisition**

학습목표



❖ 이 워크샵에서는 **library**와 데이터를 가져올 수 있습니다.

Subsection 1



library

```
pip install numpy scipy sklearn pandas matplotlib  
pip install xlrd=1.2.0  
pip install openpyxl
```

```
import sys  
print("python 버전 : {}".format(sys.version))  
import pandas as pd  
print("pandas 버전 : {}".format(pd.__version__))  
import matplotlib  
print("matplotlib 버전 : {}".format(matplotlib.__version__))  
import numpy as np  
print("numpy 버전 : {}".format(np.__version__))  
import scipy as sp  
print("scipy 버전 : {}".format(sp.__version__))  
import IPython  
print("IPython 버전 : {}".format(IPython.__version__))  
import sklearn  
print("sklearn : {}".format(sklearn.__version__))
```

```
cd C:\Users\k8s\mlops_workspaces\learning_project
!dir/w/p
```

```
`
```

C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: BEDo-C858

C:\Users\k8s\mlops_workspaces\learning_project 디렉터리

```
[.]          [..]
[.ipynb_checkpoints]  2020-01-21_A_Line.xlsx
2020-01-22_A_Line.xlsx  2020-01-23_A_Line.xlsx
2020-01-24_A_Line.xlsx  2020-01-25_A_Line.xlsx
learning_project_ex1.ipynb
        6개 파일      6,364,907 바이트
        3개 디렉터리 12,031,934,464 바이트 남음
```

```
`
```

Subsection 2



Data Ingestion, Data Acquisition

```
import pandas as pd
import glob
import os

all_workbooks = glob.glob('*.xlsx')
data_frames = []
for workbook in all_workbooks:
    all_worksheets = pd.read_excel(workbook, sheet_name=None, index_col=None)
    for worksheet_name, data in all_worksheets.items():
        data_frames.append(data)
all_data_concatenated = pd.concat(data_frames, axis=0, ignore_index=True)

writer = pd.ExcelWriter("A_Line_2020_01.xlsx")
all_data_concatenated.to_excel(writer, sheet_name='all_data_all_workbooks', index=False)
writer.save()
```

```
#불러올 파일의 경로를 filename 변수에 저장
filename = './A_Line_2020_01.xlsx'
```

```
import pandas as pd
```

```
#pandas read_excel로 불러오기
A_Line_2020_01 = pd.read_excel(filename)
A_Line_2020_01.head()
```

```
\
```

Date	Time FATPSERIAL ACC_X	Result DSNSERIAL ACC_Y	Periods ETC ACC_Z	WRITING BLE RSSI	BLE DEVICENAME ATTIVECURR	BLE MAC ADDRESS STANBYCURR	FCTVER IR/Current	MLBSERIAL IR LED
0	2020-01-21 GRS65006337020A24N00002 224.1	19:06:34 1628	OK WIP24211QUH100002 -8	19.4 24211QUH100002 -7	OK OK 1151	AbbeyFactoryTest -33.0	B010A059463E 2.1	Dec 15 2019 2.9
1	2020-01-21 GRS65006337020A24N00004 212.3	19:06:34 741	OK WIP24211QUH100004 9	19.8 24211QUH100004 -8	OK OK 1122	AbbeyFactoryTest -34.0	B010A0594640 2.1	Dec 15 2019 2.4
2	2020-01-21 GRS65006337020A24N00005 217.4	19:06:34 1133	OK WIP24211QUH100005 -21	19.6 24211QUH100005 -45	OK OK 968	AbbeyFactoryTest -42.0	B010A0594646 2.1	Dec 15 2019 2.5
3	2020-01-21 GRS65006337020A24N00003 217.1	19:06:34 1254	OK WIP24211QUH100003 -5	18.6 24211QUH100003 -40	OK OK 942	AbbeyFactoryTest -33.0	B010A0594642 2.0	Dec 15 2019 1.9
4	2020-01-21 GRS65006337020A24N00001 225.1	OK 2785	18.9 WIP24211QUH100001 -35	OK 24211QUH100001 -11	AbbeyFactoryTest OK 948	B010A0594648 -37.0	Dec 15 2019 2.0	2.5
5								


```
A_Line_2020_01.to_csv('./A_Line_2020_01.csv', index=False)  
import pandas as pd
```

```
A_Line_2020_01=pd.read_csv('./A_Line_2020_01.csv')
```

Unit A



참고자료

- ❖ <http://www.ncs.go.kr>
- ❖ NELLDAL/JOHN LEWIS 지음, 조영석/김대경/박찬영/송창근 역, 단계별로 배우는 컴퓨터과학, 홍릉과학출판사, 2018
- ❖ 혼자 공부하는 머신러닝+딥러닝 박해선 지음 | 한빛미디어 | 2020년 12월
- ❖ 머신러닝 실무 프로젝트, 아리가 미치아키, 나카야마 신타, 니시바야시 다카시 지음 | 심효섭 옮김 | 한빛미디어 | 2018년 06월
- ❖ 파이썬을 활용한 머신러닝 쿡북 크리스 알본 지음 | 박해선 옮김 | 한빛미디어 | 2019년 09월
- ❖ 처음 배우는 머신러닝 김의중 지음 | 위키북스 | 2016년 07월
- ❖ 파이썬으로 배우는 머신러닝의 교과서 : 이토 마코토 지음 | 박광수(아크몬드) 옮김 | 한빛미디어 | 2018년 11월
- ❖ <https://www.assemblyai.com/blog/pytorch-vs-tensorflow-in-2022/>
- ❖ 기타 서적 및 웹 사이트 자료 다수 참조

❖ 리눅스 다운로드

```
!rm -rf bigStudy
```

```
!git clone 'https://github.com/looker2zip/bigStudy.git'
```

❖ 윈도우 다운로드

<https://github.com/looker2zip/bigStudy>

The screenshot shows the GitHub repository page for 'looker2zip/bigStudy'. The repository is public and has 4 commits, 0 stars, and 0 forks. The commit history shows a commit '9c012f5' 7 days ago and a commit '12 days ago' for the 'README.md' file. The repository contains a 'dataset' folder and a 'README.md' file. The page also shows the 'Code' button and the 'About' section with the text 'No description, website, or topics provided.'

Commit Hash	Commit Message	Commit Date
9c012f5	Add files	7 days ago
	Initial commit	12 days ago

1. 404에러 발생 시 방문기록 삭제 후 재접속 한다. 재접 속시 암호 물어보면... 암호는 goorm
 2. !git clone <https://github.com/onlookertozip/bigStudy.git>
 3. apt-get update
 4. python3 -m pip install --upgrade pip
 5. pip install selenium
 6. apt-get update
 7. apt install chromium-chromedriver
 8. pip install beautifulsoup4
 9. pip install fancyimpute
 10. pip install mglearn
- apt-get install -y fonts-nanum
fc-cache -fv
rm ~/.cache/matplotlib -rf
- import matplotlib.pyplot as plt
plt.rc('font', family='NanumBarunGothic')

1. `vpip install lxml`
2. `pip3 install konlpy`
3. `pip install wordcloud`
4. `pip install nltk`
5. `apt-get update`
6. `apt-get install g++ openjdk-8-jdk`
7. `pip3 install konlpy JPytype1-py3`
8. `bash <(curl -s https://raw.githubusercontent.com/konlpy/konlpy/master/scripts/mecab.sh)`

```
import os
```

```
path = './xlsxdataset/'
```

```
file_list = os.listdir(path)
```

```
file_list_py = [file for file in file_list if file.endswith('.xlsx')]
```

```
for i in file_list_py:
```

```
    workbook = open_workbook(path + i)
```



감사합니다.

- ❖ Mobile: 010-9591-1401
- ❖ E-mail: onlooker2zip@naver.com