

텐서플로우 시작 1

텐서플로우란

텐서플로우란

- 텐서플로우 등장 배경
 - 2015년 11월 9일 구글이 만든 머신러닝 오픈 소스 라이브러리 텐서플로우를 공개
 - 구글의 딥러닝 연구팀인 구글 브레인팀은 2011년부터 심층 신경망(Deep Neural Network, DNN) 연구를 시작하였으며, DistBelief를 개발
 - DistBelief는 비지도 학습, 강화 학습, 언어 인식, 이미지 인식, 보행자 감지, 바둑 등 여러분야에 사용되었음
 - DistBelief를 활용하여 구글 검색, 광고, 음성 인식, 포토, 지도, 번역 등 다양한 제품에 적용 되었음

- **텐서플로우(TensorFlow)** : TensorFlow는 방향성이 있는 그래프 구조로 모델을 구성하는데 이때 이 그래프는 0 개 이상의 읍출력을 가지는 노드들의 연결체이며 노드는 operation의 instance라고 할 수 있음
- **심층신경망(Deep Neural Network, DNN)** : 심층 신경망은 입력층(Input Layer)과 출력층(Output Layer) 사이에 여러 개의 은닉층(Hidden Layer)들로 이루어진 인공 신경망(Artificial Neural Network, ANN)임

텐서플로우란

- 텐서플로우 특징
 - 차세대 대규모 머신러닝 시스템을 만들었으며, 기존 DistBelief를 개선하여 확장성과 유연성이 뛰어나도록 설계하였음
 - Android와 IOS 같은 모바일 환경 및 64비트 리눅스, 맥 OS의 데스크탑 혹은 서버 시스템 다수의 CPU와 GPU에서 구동할 수 있도록 지원하였고, 하드웨어에 대한 이해가 필요 없이 다중 병렬처리가 가능함
- 간단한 용어 알고 가기
 - 텐서(Tensor) : 2차원 이상의 배열을 의미하는 용어로서 임의의 차원을 가진 배열을 말 함
 - 오퍼레이션(Operation) : 임의의 계산을 수행하는 것으로 다양한 속성값(attribute)를 가질 수 있음
 - 노드(Node) : 그래프에서 Operation을 표현함
 - 엣지(Edge) : TensorFlow라는 이름은 edge를 통해 Tensor가 흐르면서 Node에서 연산이 일어남
 - 변수(Variable) : 학습을 통해 변화하는 배열값을 저장하기 위한 Operation 임. 메모리상에서 Tensor를 저장하는 버퍼 역할을 함
 - 세션(Session) : TensorFlow 그래프를 구성한 후 연산을 실행할 때 다양한 실행 환경(CPU, GPU, 분산처리)에서 Session을 만들어 전달함. 그래프 내에 Operation의 실행 환경을 캡슐화한 것임

텐서플로우 동작 확인

- 예제 - add 연산

- 간단한 TensorFlow 라이브러리를 이용하여, 입력한 데이터에 2를 더하는 예제
- 7번 라인 : 입력 데이터는 배열로 선언
- 9번 라인 : 변수 x는 계산을 위한 입력 데이터가 저장되는 공간으로 placeholder로 선언
- 11번 라인 : 변수 W는 값이 2인 상수형으로 선언
- 14번 라인 : 연산식 정의
- 17번 라인 : 변수 초기화
- 20~22번 라인 : 그래프 실행
- 24번 라인 : 연산 결과 출력

```
1 # add 연산
2
3 # Tensorflow 라이브러리 가져오기
4 import tensorflow as tf
5
6 # 입력 데이터 정의
7 inputData = [2,4,6,8]
8 # x : 입력데이터가 들어갈 데이터 자료형(Placeholder) 선언
9 x = tf.placeholder(dtype=tf.float32,name='x')
10 # W : 입력데이터와 연산을 할 상수형(Constant) 데이터형 선언
11 W = tf.constant([2],dtype=tf.float32, name='Weight')
12
13 # 연산식 정의
14 graph_function = tf.add(x,W)
15
16 # operation을 위해 변수 초기화
17 op = tf.global_variables_initializer()
18
19 # 정의된 연산식을 이용하여 그래프를 실행
20 with tf.Session() as sess :
21     # 초기화 실행
22     sess.run(op)
23     # 연산 결과 출력
24     print(sess.run(graph_function,feed_dict={x : inputData}))
```

결과

[4. 6. 8. 10.]

Data Flow Graph

- 예제 add 연산 Data Flow Graph



- Tensorflow 그래프는 엣지(edge)를 텐서(Tensor) 형태로 값이 다른 연산(operation)으로 이동
- w와 x는 텐서를 저장하기 위한 상수(constant)와 실행단계(session)에서 값을 전달하기 위한 노드
- tf.add() 연산(operation)은 입력 데이터 x(operation node : placeholder)와 w(operation node : constant) 두개의 노드값을 엣지를 통하여 입력 받음
- 입력 받은 후 add 연산을 수행하고 graph_function에 결과를 저장
- Data Flow Graph를 구성한 후 Session에서 이를 실행하여 결과를 출력

빌드구조와 실행구조

텐서플로우 빌드구조와 실행구조

- 빌드구조와 실행구조
 - Tensorflow는 텐서 형태의 데이터를 이용하여 노드와 엣지로 그래프를 표현
 - 표현된 그래프는 세션을 이용하여 계산
 - 세션은 CPU나 GPU와 같은 Device에 그래프 연산을 올린 뒤 연산을 실행할 수 있는 메소드 제공
 - 메소드를 통하여 반환되는 결과는 텐서 형태를 가짐
 - 빌드 단계 : 노드와 엣지로 구성된 그래프를 생성
 - 실행 단계 : 생성한 그래프를 연산하여 결과값 반환
- 예시
 - 빌드 단계에서 $a = 10$ 을 선언해도 실제로 a 변수에 10이 선언된 것이 아니고, 실행 단계에서 a 변수 값이 10으로 입력됨
 - 빌드 단계에서는 실행을 위한 그래프를 생성하고, 실행 단계에서는 그래프를 실행하게 되어 실제로 선언된 데이터값들이 입력되고 연산을 수행함

텐서플로우 빌드구조와 실행구조

- 예제 - 원의 넓이 및 총합 구하기
 - 6번 라인 : 입력 데이터
 - 8,10번 라인 : 입력 데이터 저장공간 선언
 - 13번 라인 : 연산식 정의
 - 14번 라인 : 결과값 저장
 - 17번 라인 : 변수 초기화
 - 19~21번 라인 : 그래프 실행
 - 25~26번 라인 : 연산 결과 출력
- 빌드 단계? or 실행 단계?
 - 1~15번 라인 : 빌드 단계
 - 17~26번 라인 : 실행 단계

```
# 반지름 입력 데이터 선언
inputData = [2., 3., 4., 5.]
# pi값 지정(소수점이하 생략)
ValuePI = tf.constant([3.], dtype=tf.float32)
# 반지름 데이터가 들어갈 데이터 자료형(Placeholder) 선언
radius = tf.placeholder(dtype=tf.float32)

# 원의 넓이, 넓이의 총합 공식 선언
area = tf.pow(radius, 2) * ValuePI
resultSum = tf.reduce_sum(area)

# operation을 위해 변수 초기화
op = tf.global_variables_initializer()
```

빌드 단계

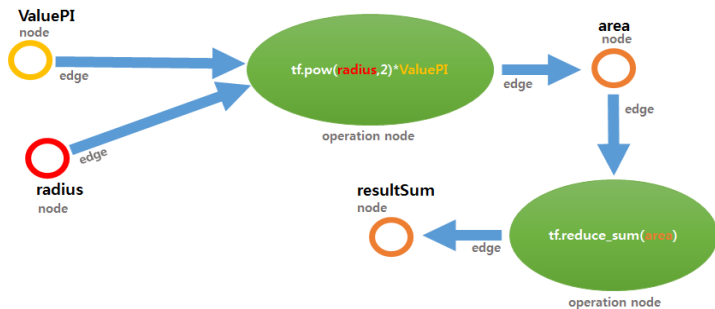
```
with tf.Session() as sess :
    # 초기화 실행
    sess.run(op)
    # fetch 방법으로 2개의 결과를 가져오고 feed 방법으로 실행시
    반지름 데이터 입력
    valueArea, valueSum = sess.run([area, resultSum],
    feed_dict={radius: inputData})
    # 결과 출력
    print ("Circle area : ", valueArea)
    print ("Total area sum : ", valueSum)
```

실행 단계

```
1 # 원의 넓이 및 총합 구하기
2 # Tensorflow 라이브러리 가져오기
3 import tensorflow as tf
4
5 # 반지름 입력 데이터 선언
6 inputData = [2., 3., 4., 5.]
7 # pi값 지정(소수점이하 생략)
8 ValuePI = tf.constant([3.], dtype=tf.float32)
9 # 반지름 데이터가 들어갈 데이터 자료형(Placeholder) 선언
10 radius = tf.placeholder(dtype=tf.float32)
11
12 # 원의 넓이, 넓이의 총합 공식 선언
13 area = tf.pow(radius, 2) * ValuePI
14 resultSum = tf.reduce_sum(area)
15
16 # operation을 위해 변수 초기화
17 op = tf.global_variables_initializer()
18
19 with tf.Session() as sess :
20     # 초기화 실행
21     sess.run(op)
22     # fetch 방법으로 2개의 결과를 가져오고 feed 방법으로 실행시 반지름 데이터 입력
23     valueArea, valueSum = sess.run([area, resultSum], feed_dict={radius: inputData})
24     # 결과 출력
25     print ("Circle area : ", valueArea)
26     print ("Total area sum : ", valueSum)
```

텐서플로우 빌드구조와 실행구조

- 빌드 단계
 - 빌드단계에서 가장 중요한 일은 그래프 생성
 - 그래프는 노드와 엣지로 구성
 - 그래프는 연산노드의 상태 유지, 갱신하는 노드나 분기, 루프를 구성하는 제어 노드 등의 확장된 노드타입
 - `tf.pow()`, `tf.reduce_sum()`노드는 연산노드
 - `area`, `resultSum`은 연산이 진행되면서 그 값이 갱신되는 노드
 - 연산노드는 입력(`ValuePI`, `radius`)값과 출력(`area`, `resultSum`)값을 가질 수 있음
 - 또한, 엣지를 따라 각 노드들 사이들을 텐서(다차원배열)가 이동



텐서플로우 빌드구조와 실행구조

- 학습모델
 - 최종목표는 Tensorflow를 이용하여 인공지능이 학습된 모델 만들기
 - 빌드단계에서 학습 모델을 만들기 위한 과정
 - 1단계 : 가설 수식 작성(Hypothesis)
 - 2단계 : 오차함수 작성(Cost Function)
 - 3단계 : 최적화 함수 작성(Optimizer Function)

- 1단계 : 가설 수식 작성(Hypothesis) : 준비한 학습 데이터를 기반으로 테스트 데이터를 예측하는 수식 작성
예) [예제2-2]에서 `tf.pow()`와 같이 연산 노드를 만들어 주는 것과 동일하게 연산을 위한 수식 작성
- 2단계 : 오차함수 작성(Cost Function) : 1단계 가설 수식 작성에서 만든 가설 수식을 이용하여 학습 데이터와 수식 간의 오차를 구하는 함수
- 3단계 : 최적화 함수 작성(Optimizer Function) : 최적화 함수를 작성할 때 오차 함수를 이용하여 얻은 오차를 최소화할 수 있도록 최적화 과정을 통해 파라미터값을 업데이트하고, 최적의 값을 결정
- 학습모델을 만들기 위한 빌드 과정에서는 위 3단계 과정으로 최종적인 그래프 작성

텐서플로우 빌드구조와 실행구조

- 실행 단계
 - 빌드단계가 완료되면 실행을 위한 그래프가 완성되었다는 의미
 - 그래프를 세션에 올려 이를 실행하여 변수들의 값을 업데이트하거나 결과 출력
 - 세션은 그래프를 실행하기 위한 클래스
 - `tf.Session()`을 이용하여 세션객체를 생성하고 사용
 - `tf.Session.run()`은 operation을 실행하거나 텐서 객체의 값을 구하기 위한 메소드
 - 세션의 실행이 완료되면 `tf.Session.close()`를 호출하여 자원을 해제 해야함

텐서플로우 빌드구조와 실행구조

- 실행 단계
 - 세션 객체를 반환하는 방법
 - `close()` 사용
 - `close()`를 이용하여 세션이 완료되면 직접 객체를 반환

```
sess = tf.Session()  
sess.run(...)  
sess.close()
```

- `with` 블록 사용
 - `with` 블록을 이용하여 세션 객체 생성
 - `with` 구문에서 `tf.Session()`으로 세션을 생성하고 `with` 구문이 끝나면 자동으로 세션 객체를 반환
 - `with` 구문 안에서는 `run()`을 사용하지 않고 `eval()`을 사용하여 기본 세션인 `sess`를 명시하지않고 사용 가능

```
with tf.Session() as sess:  
    sess.run(...)
```

```
a = tf.constant(1.0)  
  
with tf.Session() as sess:  
    print(a.eval())
```

텐서플로우 빌드구조와 실행구조

- 실행 단계
 - 세션 객체를 생성하는 다른 방법
 - `tf.InteractiveSession()` 사용
 - `InteractiveSession()`은 해당 메소드 자체가 세션을 기본 세션으로 만들어 명시적으로 세션을 호출하지 않고 `run()`, `eval()`을 사용할 수 있음

```
sess = tf.InteractiveSession()
a = tf.constant(5.0)
b = tf.constant(6.0)
c = tf.subtract(a, b)
# 'sess'의 전달없이도 'c.eval()'를 실행할 수 있습니다.
print(c.eval())
sess.close()
```

- `tf.Session()` vs `tf.InteractiveSession()` 차이점
 - 기능상 큰 차이점은 없음
 - `sess`(세션을 보유하는 변수)가 기본 세션으로 인지하고 실행을 할 수 있기 때문에 `sess` 생략 가능
 - Jupyter notebook과 같은 인터랙티브 파이썬 환경에서는 이용의 편의성 제공

텐서플로우 개발 환경

텐서플로우 설치

- 설치환경

- Tensorflow 공식홈페이지(<https://www.tensorflow.org/install>)에 운영체제 별 설치방법 확인 가능
- Tensorflow는 CPU버전과 GPU버전으로 구분되어 있음
- CPU버전은 설치가 쉬우며, 초보자가 학습하기에 알맞음
- GPU버전은 CPU보다 속도가 훨씬 빠름
- GPU버전을 사용하기 위해서는 CUDA, cuDNN 등의 NVIDIA 소프트웨어를 설치해야함
- 운영체제 별 최소 버전 및 CPU, GPU 지원 여부

운영 체제	최소 버전	CPU 지원	GPU 지원
Linux(Ubuntu)	16.04 or later	○	○
Windows	7 or later	○	○
Mac	10.12.6(Sierra) or later (no GPU support)	○	× (1.2버전부터 GPU 지원 하지 않음)

- Tensorflow는 64비트 운영체제 환경에서만 설치 가능

텐서플로우 설치

- 설치환경
 - 운영체제 별 텐서플로우 권장 설치방법

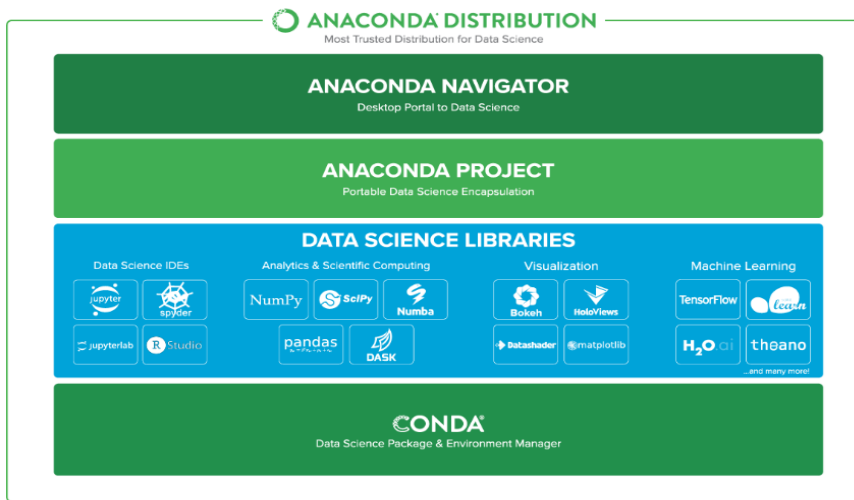
운영 체제 \ 설치 방법	Virtualenv	"native" pip	Docker	Anaconda	소스 코드
Linux(Ubuntu)	○ (권장)	○	○	○	○
Windows		○		○	
Mac	○ (권장)	○	○	○	○

- ☐ **virtualenv** : 한 컴퓨터에서 여러 프로젝트를 작업할 때 python 패키지의 의존성이 충돌하지 않도록 함
- ☐ **"native" pip** : 가상환경을 거치지 않고 시스템에 바로 Tensorflow를 설치
- ☐ **Docker** : Tensorflow를 docker 컨테이너에서 실행하므로 컴퓨터의 다른 프로그램과 분리되어 운영할 수 있음
- ☐ **Anaconda** : Tensorflow를 각 anaconda 가상 환경에 설치하기 때문에 다른 프로그램에 영향이 미치지 않음
- ☐ **소스코드** : pip wheel을 이용하여 빌드하고 설치

- Linux와 Mac에서는 모든 방법으로 설치가 가능함
- Windows는 일부 방법만 설치가 가능하다고 표시되어 있지만, 다른 방법으로도 설치가 가능함
- 다만, 공식적인 지원이 이루어지지 않으므로 권장하는 방법으로 설치하는것이 좋음

텐서플로우 설치

- Anaconda란
 - Anaconda는 세계에서 가장 유명한 python 데이터 과학 플랫폼
 - 한 번의 클릭으로 모든 데이터 과학 패키지를 쉽게 설치할 수 있음
 - 패키지, 종속성 및 환경을 관리할 수 있음
 - python 패키지들이 포함되어 있기 때문에, 별도의 python 설치가 필요하지 않음
 - Anaconda 구성요소는 네 부분으로 나뉨짐

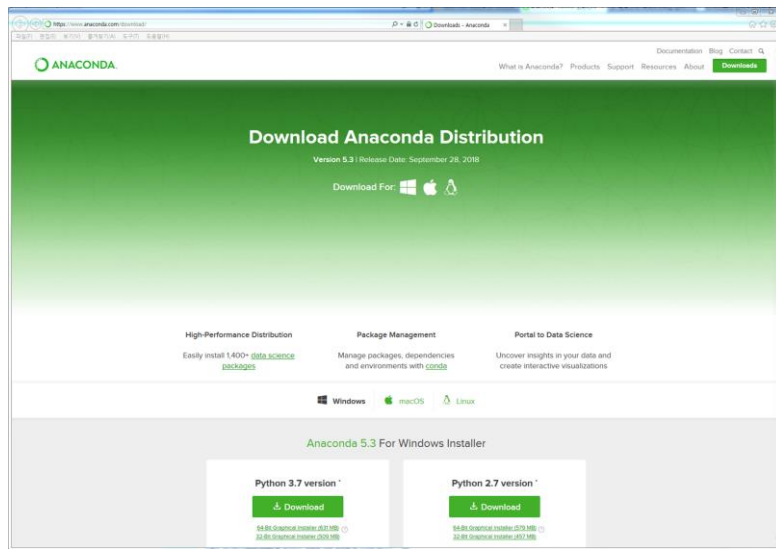


텐서플로우 설치

- Anaconda 구성요소
 - Anaconda Navigator
 - UI 클라이언트로 하부 컴포넌트를 쉽게 사용할 수 있음
 - Jupyter나 Spyder 같은 개발 도구를 사용할 수 있음
 - Anaconda Project
 - 올바른 패키지 설치 및 파일 다운로드, 환경 변수 등 설치 단계 자동화
 - 작업을 쉽게 재현하고 프로젝트를 다른 사람들과 공유하며 다른 플랫폼에서 실행 가능
 - Data Science Libraries
 - Jupyter와 같은 IDE개발도구, Numpy/SciPy 같은 과학 분석용 라이브러리
 - Matplotlib 같은 데이터 시각화(Data Visualization) 라이브러리
 - Tensorflow 같은 머신러닝(Machine Learning) 라이브러리 등을 포함하고 있음
 - Conda(Data Science Package & Environment Manager)
 - 모든 데이터 과학 패키지 포함
 - 패키지, 종속성 및 환경을 관리할 수 있음

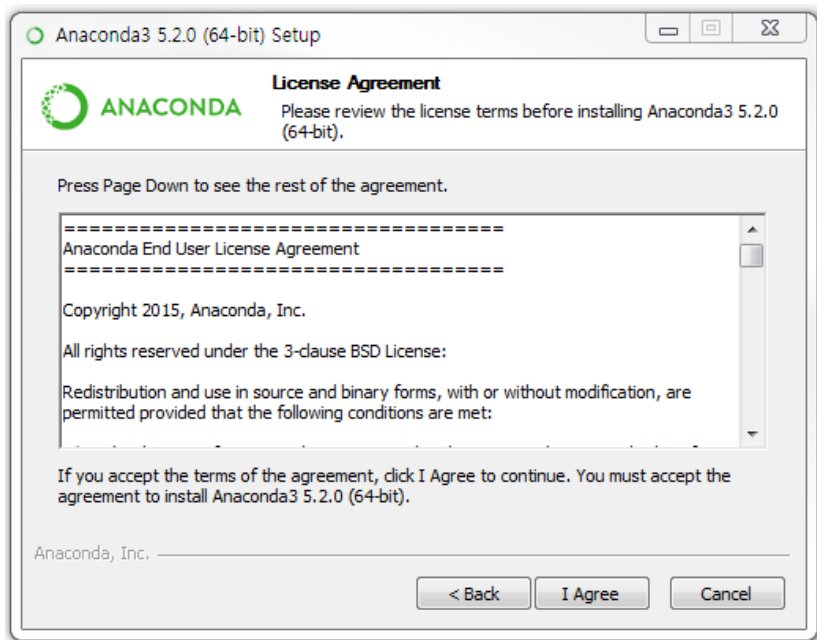
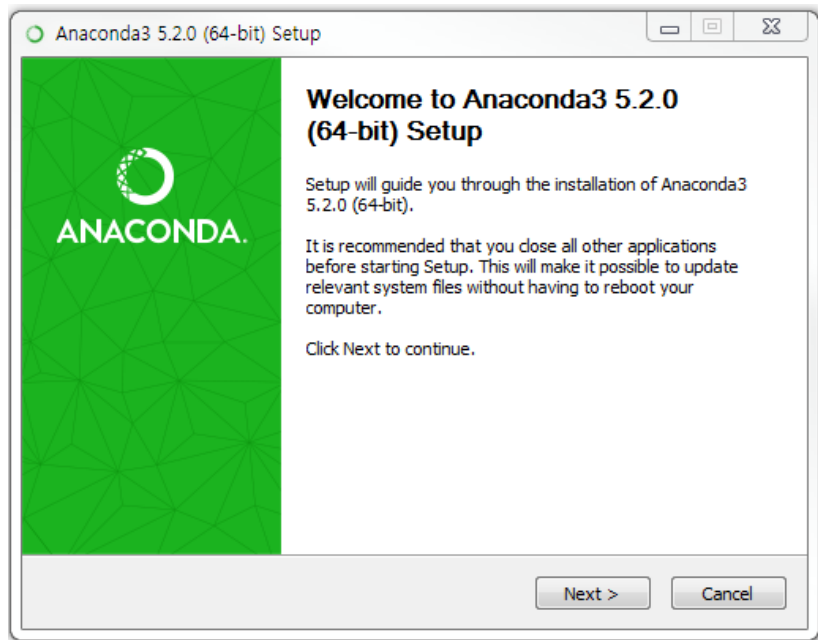
텐서플로우 설치

- Anaconda 설치
 - Tensorflow는 python, C++, Java, Go 언어를 지원하고 있음
 - 가장 쉽게 사용할 수 있는 python을 이용하여 환경 구축
 - Anaconda 공식 홈페이지센터(<https://www.anaconda.com/download/>)에서 파일을 다운 받아 설치
 - 운영체제에 맞춰 python 버전별로 제공
 - Tensorflow는 python3.5이상에서 동작
 - python3.5 이상 파일 다운 후 설치해야함



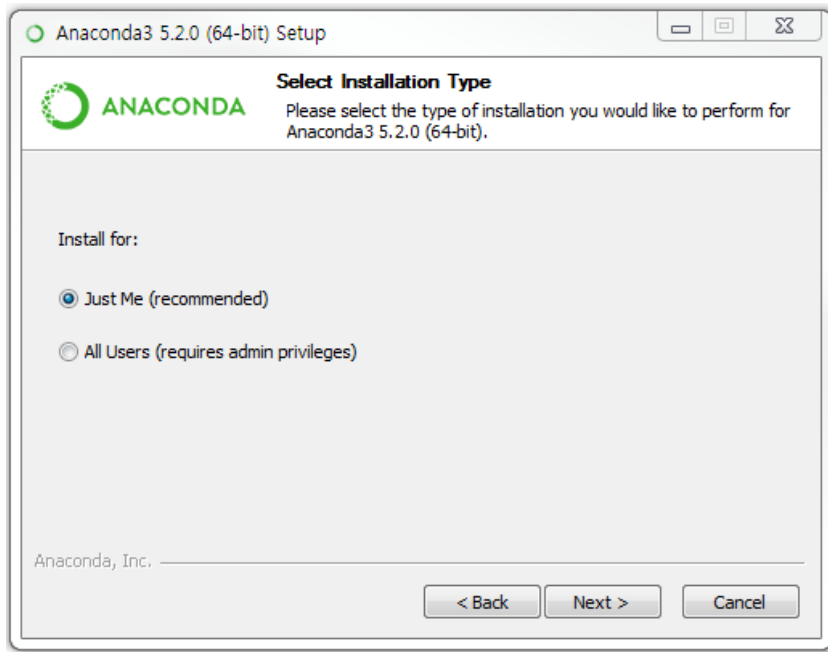
텐서플로우 설치

- Anaconda 설치 방법



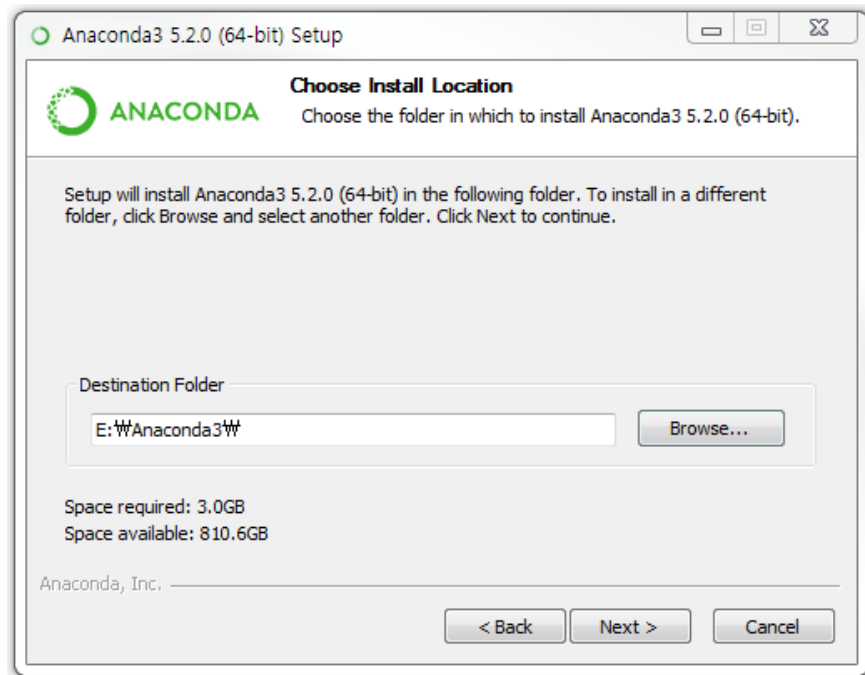
텐서플로우 설치

- Anaconda 설치 방법
 - 프로그램 접근 권한 설정
 - Just me : 현재 로그인 사용자에게만 접근 권한 허용
 - All Users : 전체 사용자에게 접근 권한 허용



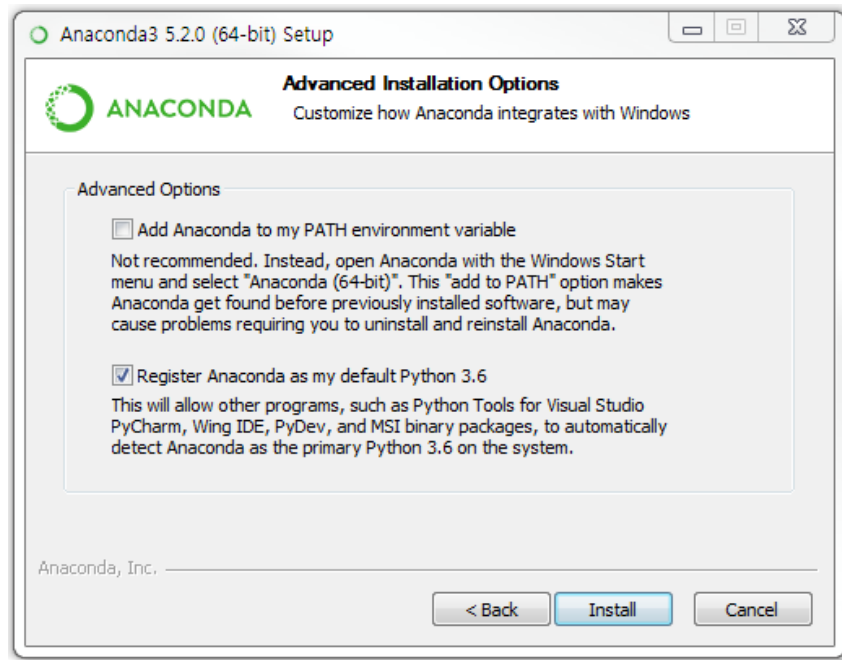
텐서플로우 설치

- Anaconda 설치 방법
 - Anaconda 설치 경로 설정
 - 디스크 용량 고려하여 사용자가 설치하고 싶은 곳에 경로 설정



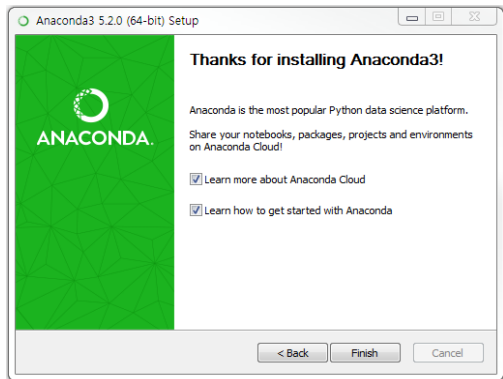
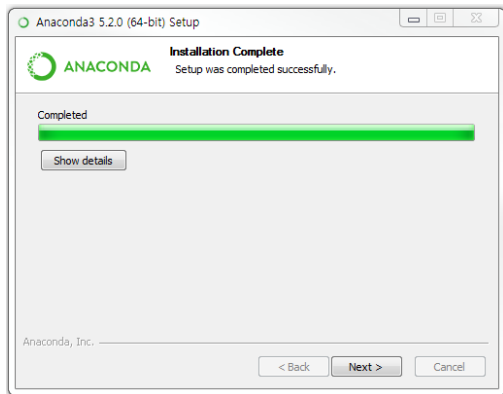
텐서플로우 설치

- Anaconda 설치 방법
 - 환경 변수 추가에 대한 고급옵션 선택
 - 기본적으로 두 번째 옵션이 선택되어 있음
 - Anaconda를 설치하면서 함께 설치되는 python 버전을 사용한다는 의미
 - 첫 번째 옵션은 다른 환경변수 경로를 설정한다는 의미
 - Anaconda를 삭제하거나 재설치하는 과정에서 환경 변수가 잘못되어 문제가 발생할 수 있기 때문에 추천하지 않음
 - 기본적으로 선택되어 있는 두 번째 옵션을 선택하고 다음 단계 진행
 - 주의 : 다른 버전의 python이 설치되어 있으면 개발 환경이 꼬일 수 있기 때문에 가급적 다른 버전의 python은 정리하는 것을 추천함

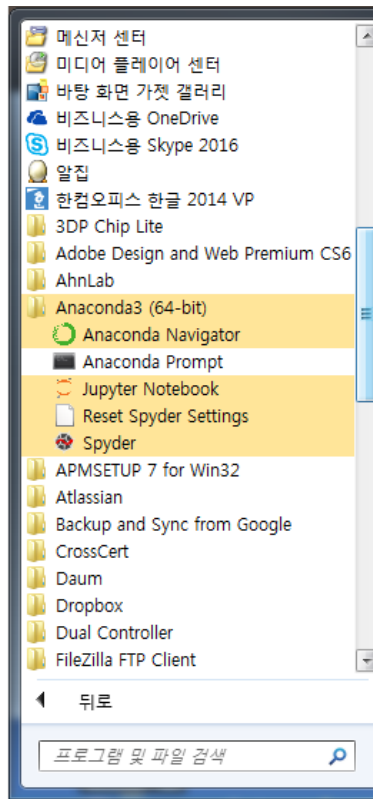
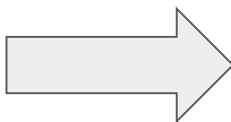


텐서플로우 설치

- Anaconda 설치 방법

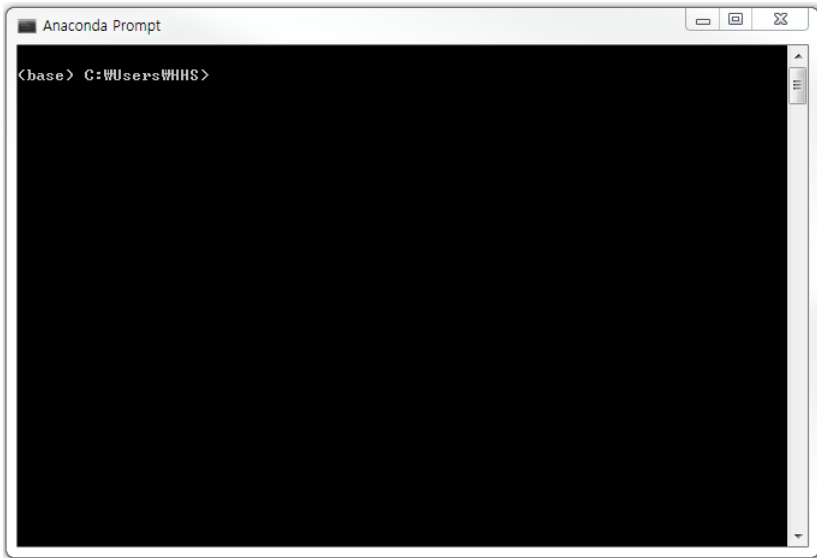


설치완료



텐서플로우 설치

- Anaconda에 텐서플로우 설치
 - Anaconda에 텐서플로우를 설치하기 위해서는 가상환경을 만들어야 함
 - 기존의 시스템에 영향을 받지 않고 독립적인 환경으로 구성
 - 가상환경에 설치하는 것이 안정적
 - 설치된 Anaconda Prompt 를 이용하여 가상환경 생성



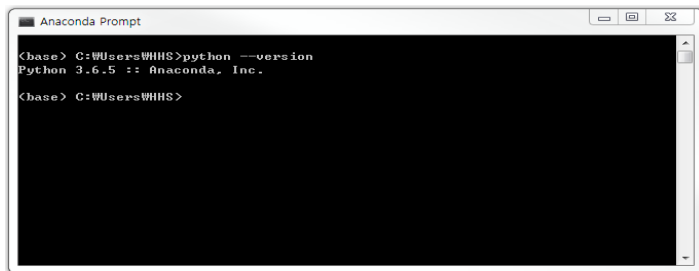
텐서플로우 설치

- Anaconda에 텐서플로우 설치
 - (base) C:\Users\Username > 형태로 기본 커서 표시(설치 환경에 따라 다르게 표시됨)
 - (base) : Anaconda 설치 경로
 - conda 명령어 사용
 - info -envs : 환경설정 리스트 확인

```
>conda info -envs

# conda environments :
#
base                E:\Anaconda3        // Anaconda 설치 경로
```

- python 버전 확인
 - Anaconda설치 시 python을 포함하고 있기 때문에 python 버전 확인
 - python -version 명령어

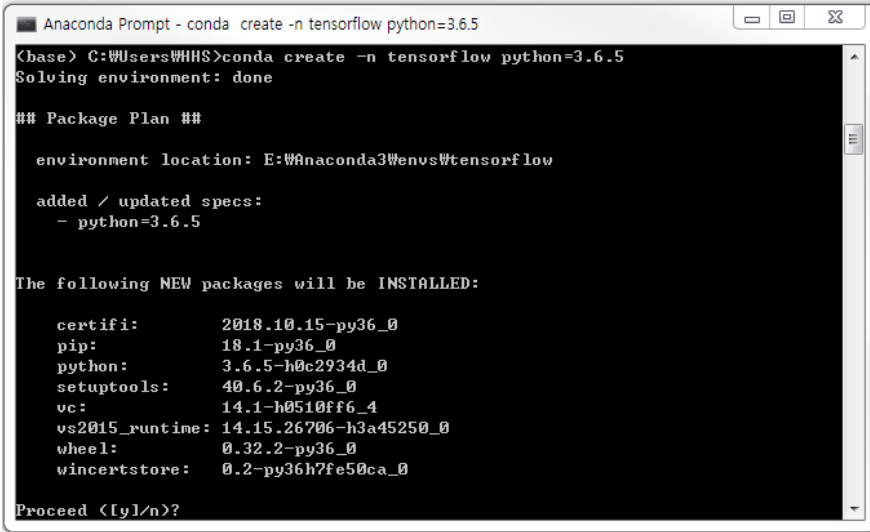


```

Anaconda Prompt
(base) C:\Users\WHHS>python --version
Python 3.6.5 :: Anaconda, Inc.
(base) C:\Users\WHHS>
```

텐서플로우 설치

- Anaconda에 텐서플로우 설치
 - 명령어를 사용하여 가상환경 생성
 - 명령어 : >conda create -n tensorflow python=3.6.5
 - 설명 : >conda create -n '가상환경이름' '설치할 패키지'
 - tensorflow라는 이름으로 python 3.6.5 버전을 사용할 수 있는 가상환경 생성



```
Anaconda Prompt - conda create -n tensorflow python=3.6.5
(base) C:\Users\WHHS>conda create -n tensorflow python=3.6.5
Solving environment: done

## Package Plan ##

  environment location: E:\Anaconda3\envs\tensorflow

added / updated specs:
- python=3.6.5

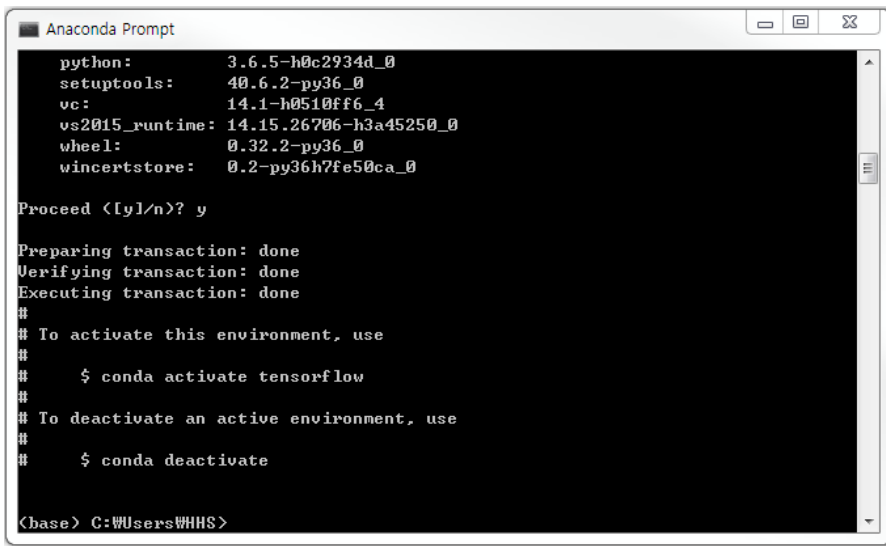
The following NEW packages will be INSTALLED:

certifi:         2018.10.15-py36_0
pip:             18.1-py36_0
python:          3.6.5-h0c2934d_0
setuptools:      40.6.2-py36_0
vc:              14.1-h0510ff6_4
vs2015_runtime: 14.15.26706-h3a45250_0
wheel:           0.32.2-py36_0
wincertstore:    0.2-py36h7fe50ca_0

Proceed [y]/n?
```

텐서플로우 설치

- Anaconda에 텐서플로우 설치
 - 가상환경 생성 완료
 - 가상 환경 실행 명령어
 - >conda activate tensorflow
 - 가상 환경 종료 명령어
 - >conda deactivate
 - 가상환경 실행
 - conda 명령어 생략 가능
 - 독립적인 환경 여러 개 생성 가능



```
Anaconda Prompt

python:      3.6.5-h0c2934d_0
setuptools:  40.6.2-py36_0
vc:          14.1-h0510ff6_4
vs2015_runtime: 14.15.26706-h3a45250_0
wheel:       0.32.2-py36_0
wincertstore: 0.2-py36h7fe50ca_0

Proceed <[y]/n>? y

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate tensorflow
#
# To deactivate an active environment, use
#
#     $ conda deactivate
#

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

텐서플로우 설치

- Anaconda에 텐서플로우 설치
 - Tensorflow 설치
 - 가상환경 실행 명령어 : >(conda) activate tensorflow
 - Tensorflow 설치 명령어 : pip install --upgrade tensorflow
 - CPU 버전은 간단하게 설치 가능
 - GPU 버전은 NVIDIA 그래픽 카드 필요
 - CUDA toolkit 및 cuDNN SDK 등 필요 소프트웨어 설치

```
Anaconda Prompt
(tensorflow) C:\Users\WHHS>pip install --upgrade tensorflow
Collecting tensorflow
  Downloading https://files.pythonhosted.org/packages/05/ed/c1/7142e33c8192b04560
ce864c26ebab3fed888fe5cd9ded61b2702f2ae/tensorflow-1.12.0-cp36-cp36m-win_and64.
whl (45.9MB)
100% |#####| 45.9MB 504kB/s
Collecting keras-preprocessing>=1.0.5 (from tensorflow)
  Downloading https://files.pythonhosted.org/packages/fc/94/74e0fa783d3fc07e4171
5773435dd051ca89c550881b3454233c39c73e69/Keras_Preprocessing-1.0.5-py2-py3-none-
any.whl
100% |#####| 45.9MB 504kB/s
Collecting six>=1.10.0 (from tensorflow)
  Using cached https://files.pythonhosted.org/packages/67/4b/141a581104b1f6397bf
a78ac9d43d8ad29a7ca43ea90a2d863fe3856e86a/six-1.11.0-py2-py3-none-any.whl
Collecting keras-applications>=1.0.6 (from tensorflow)
  Downloading https://files.pythonhosted.org/packages/3f/c4/2ff40221029f9098458f
8471b9b97a8100f3293f9856f8f5834bef100b/Keras_Applications-1.0.6-py2-py3-none-a
ny.whl (44kB)
100% |#####| 51kB 4.7MB/s
Collecting grpcio>=1.8.6 (from tensorflow)
  Downloading https://files.pythonhosted.org/packages/23/96/c98fb50c8c6b93e3846f
f4b9f39d09bc48b7382293c8ada3bf47c6467c58/grpcio-1.16.1-cp36-cp36m-win_and64.whl
(1.5MB)
100% |#####| 1.5MB 5.1MB/s
Collecting termcolor>=1.1.0 (from tensorflow)
  Using cached https://files.pythonhosted.org/packages/8a/a7/6be51647d0eb9f10e
2a4511bf3ff8b8c1e6b14e9e4fab46173aa79f981/termcolor-1.1.0.tar.gz
Collecting tensorboard>=1.13.0 (from tensorflow)
  Downloading https://files.pythonhosted.org/packages/a0/40/65fe48383146199f16db
45999f226b87bce63ad5cd73c848cf722637997/tensorboard-1.12.0-py3-none-any.whl (3.
0MB)
100% |#####| 3.1MB 4.7MB/s
Collecting absl-py>=0.1.6 (from tensorflow)
  Downloading https://files.pythonhosted.org/packages/0c/63/f505d2d4c21db849cf80
```

```
Anaconda Prompt
493217c44739646a106c61859622eccc297a5c05/h5py-2.8.0-cp36-cp36m-win_and64.whl (2.
3MB)
100% |#####| 2.3MB 4.5MB/s
Collecting markdown>=2.6.8 (from tensorboard<1.13.0>=1.12.0->tensorflow)
  Downloading https://files.pythonhosted.org/packages/7a/6b/5600647484ba15545ec3
7d2f7f58844d690baf2f81f3a60b862e48f29287/Markdown-3.0.1-py2-py3-none-any.whl (89
kB)
100% |#####| 92kB 5.8MB/s
Collecting werkzeug>=0.11.10 (from tensorboard<1.13.0>=1.12.0->tensorflow)
  Using cached https://files.pythonhosted.org/packages/2b/c4/12e3e56472e52375aa2
9c4764e79db8f3fa6682b8f8d0a04fe335243/Werkzeug-0.14.1-py2-py3-none-any.whl
Requirement already satisfied, skipping upgrade: setuptools in c:\waaconda3\envs\
tensorflow\lib\site-packages (from protobuf>=3.6.1->tensorflow) (40.6.2)
Building wheels for collected packages: termcolor, absl-py, gast
  Running setup.py bdist_wheel for termcolor ... done
  Stored in directory: C:\Users\WHHS\AppData\Local\Temp\WCache\Wheels\W7cW06W54Wbc84
598hdaif8f970247f550b175aaac85f68b40c5ab2c6
  Running setup.py bdist_wheel for absl-py ... done
  Stored in directory: C:\Users\WHHS\AppData\Local\Temp\WCache\Wheels\W18W0a5W5eW36e
1h8739c78cd2eba0a08fcd602c2h16a4b263912af8cb64
  Running setup.py bdist_wheel for gast ... done
  Stored in directory: C:\Users\WHHS\AppData\Local\Temp\WCache\Wheels\W9aW1fW0eW3cde
98113222h853e98fc0a0e9924480a3e25f1b408cedb4f
Successfully built termcolor absl-py gast
Installing collected packages: six, numpy, keras-preprocessing, h5py, keras-appl
ications, grpcio, termcolor, protobuf, markdown, werkzeug, tensorboard, absl-py,
gast, astor, tensorflow
Successfully installed absl-py-0.6.1 astor-0.7.1 gast-0.2.0 grpcio-1.16.1 h5py-2
.8.0 keras-applications-1.0.6 keras-preprocessing-1.0.5 markdown-3.0.1 numpy-1.1
6.4 protobuf-3.6.1 six-1.11.0 tensorboard-1.12.0 tensorflow-1.12.0 termcolor-1.1
.0 werkzeug-0.14.1
(tensorflow) C:\Users\WHHS>
```

텐서플로우 설치

- Anaconda에 텐서플로우 설치
 - 설치 확인 테스트
 - 간단한 예제로 Tensorflow 설치 확인 테스트

(예제) 간단한 문장 출력

Tensorflow 가상 환경 실행

(base) C:\Users\HHS>activate tensorflow

python 실행

(tensorflow) C:\Users\HHS>python

```
>>>import tensorflow as tf
>>>result = tf.constant("Tensorflow is easy!!")
>>>sess = tf.Session()
>>>print(sess.run(result))
```

결과

b'Tensorflow is easy!!'

```
<tensorflow> C:\Users\HHS>python
Python 3.6.5 |Anaconda, Inc.| (default, Mar 29 2018, 13:32:41) [MSC v.1900 64 bi
t (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
>>> result = tf.constant("Tensorflow is easy!!")
>>> sess = tf.Session()
2018-11-18 17:34:11.263200: I tensorflow/core/platform/cpu_feature_guard.cc:141
Your CPU supports instructions that this TensorFlow binary was not compiled to
use: AVX2
>>> print(sess.run(result))
b'Tensorflow is easy!!'
>>>
```

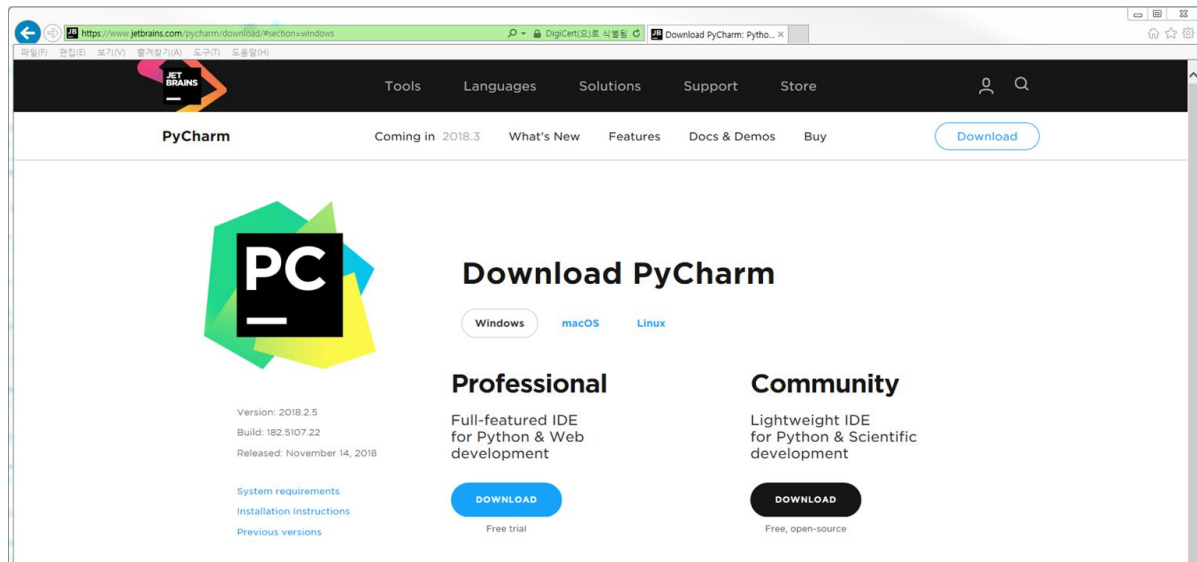
현재 사용하고 있는 시스템에서 AVX2 명령어를 지원하지만
Tensorflow에서는 해당 명령어를 사용하지 않도록 빌드가 된 버전이
설치되었다는 내용(해결방안 뒷 장에서 설명)

텐서플로우 개발 환경 구축

텐서플로우 개발 환경 구축

- PyCharm이란

- python 프로그램을 쉽게 개발할 수 있도록 도와주는 통합개발환경
- 통합개발환경은 개발자가 소프트웨어를 개발하는 과정에서 필요한 작업을 하나의 소프트웨어에서 처리할 수 있는 환경 제공
- PyCharm 공식 홈페이지(<https://www.jetbrains.com/pycharm>)에서 설치 파일 다운

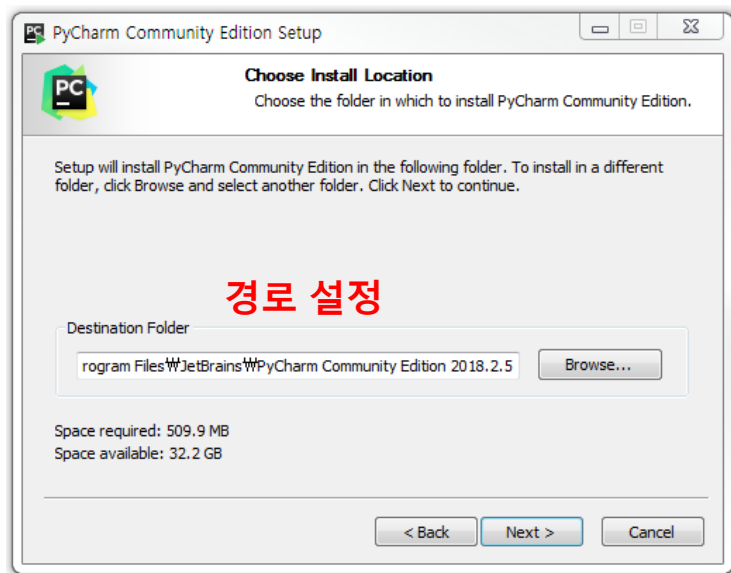


텐서플로우 개발 환경 구축

- PyCharm이란
 - Professional 버전
 - 유료버전
 - Free trial로 30일까지 사용가능
 - 학생인증 시 Student License를 통해 1년간 무료 사용 가능
 - Community 버전
 - 무료버전
 - Professional버전보다 제공하는 기능이 적음

텐서플로우 개발 환경 구축

- PyCharm 설치 방법

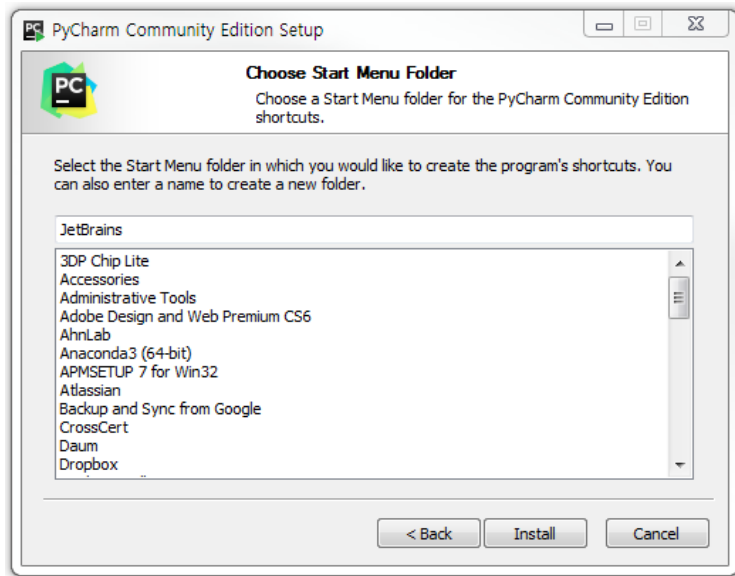
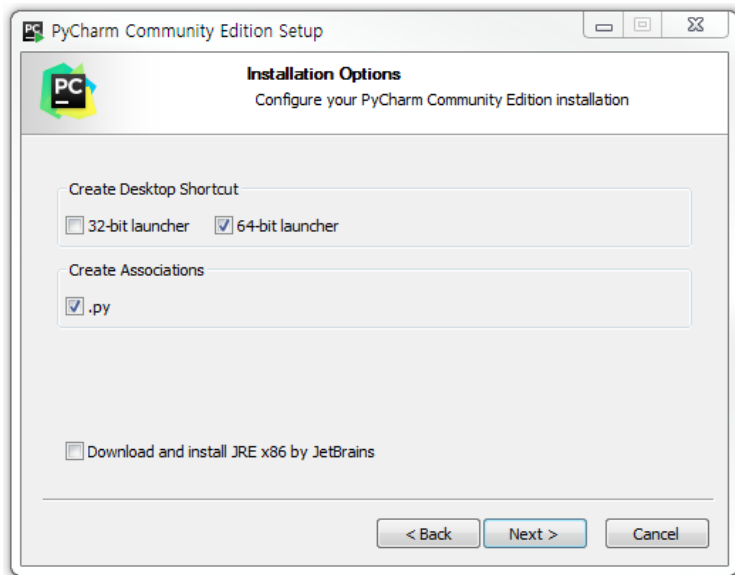


텐서플로우 개발 환경 구축

- PyCharm 설치 방법

Create Desktop Shortcut : 바탕화면 바로가기 선택

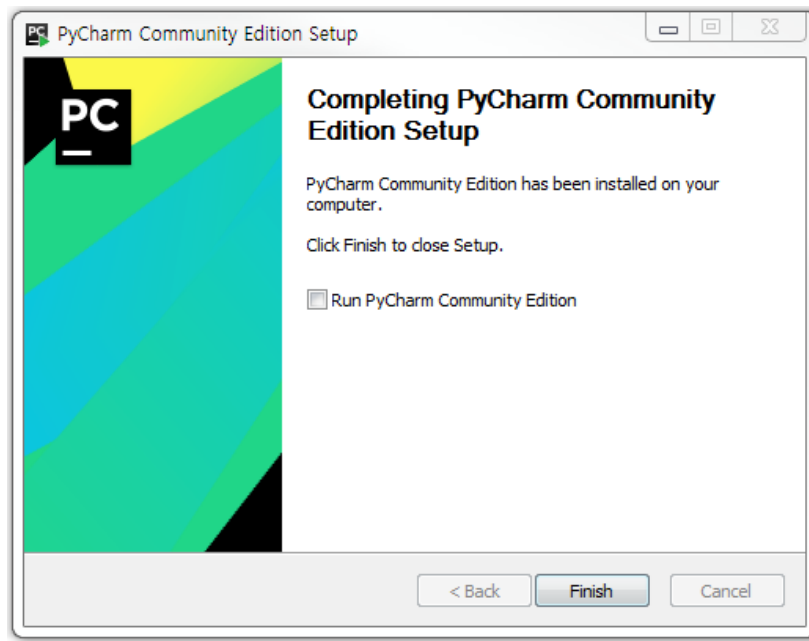
Create Associations : 모든 python 파일은 PyCharm로 연결



시작 메뉴에 폴더 설정

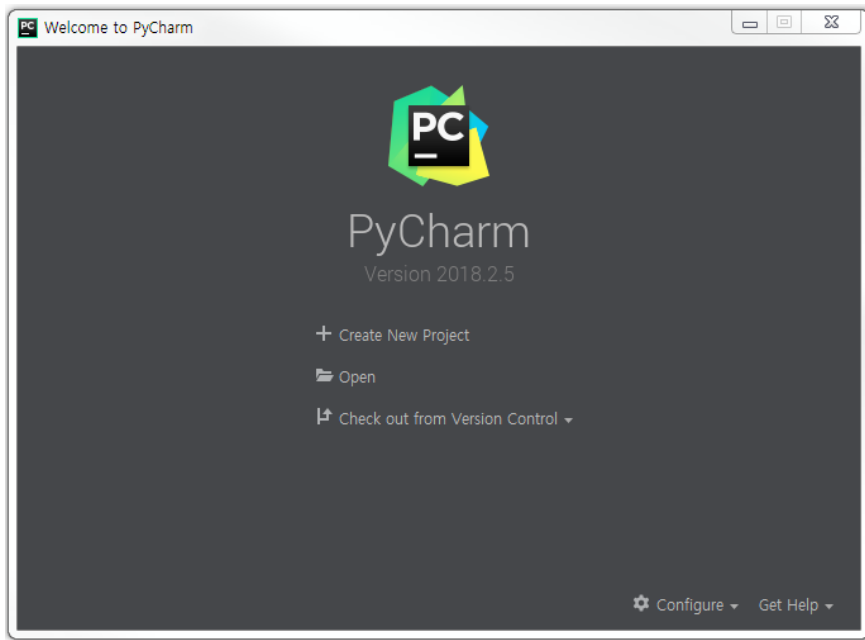
텐서플로우 개발 환경 구축

- PyCharm 설치 방법
 - 설치 완료 화면
 - 체크 박스를 선택하고 종료를 하면 PyCharm 프로그램 실행됨



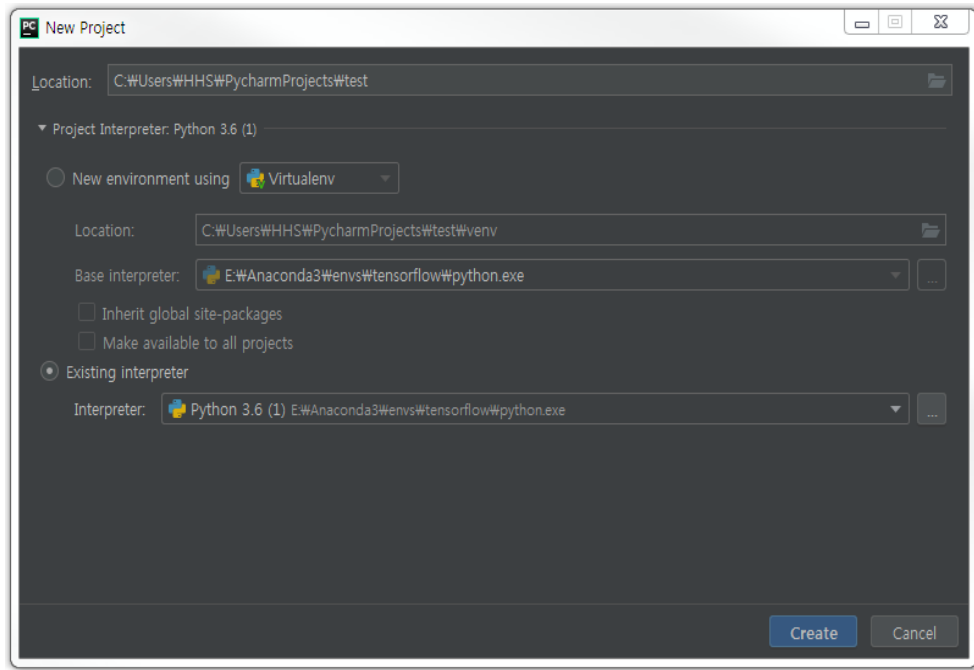
텐서플로우 개발 환경 구축

- PyCharm 실행
 - PyCharm 첫 실행 화면
 - Create New Project : 새 프로젝트 생성
 - Open : 저장되어 있는 파일 열기
 - Check out from Version Control :



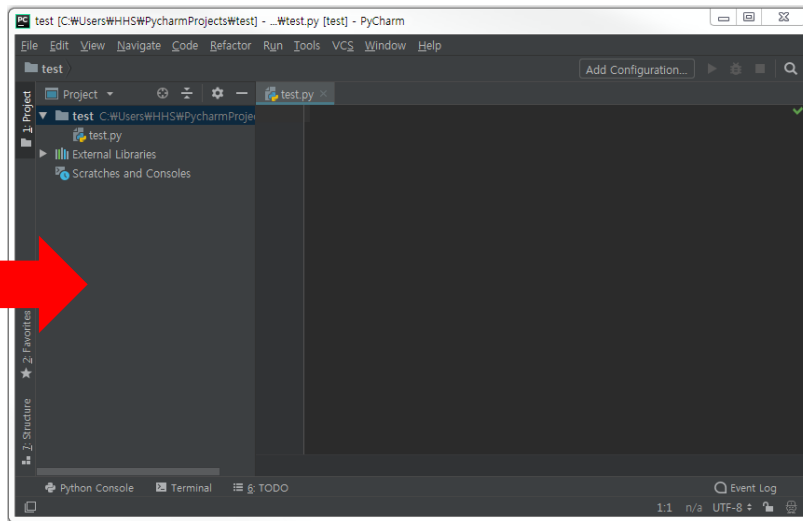
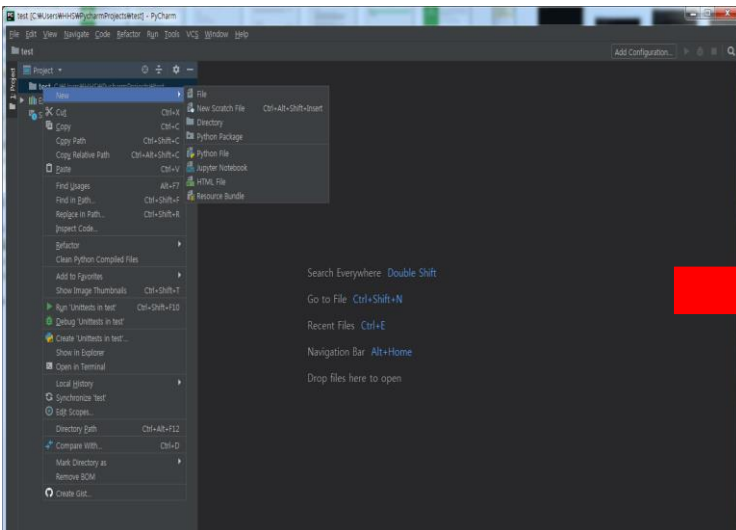
텐서플로우 개발 환경 구축

- 새 프로젝트 생성
 - Location : 프로젝트 저장 경로 및 프로젝트 명 설정
 - Project Interpreter
 - New environment Using
: 새로운 환경 사용
 - Existing Interpreter
: 기존 python 사용
 - Existing Interpreter 선택 후
기존에 설치된 python 경로 선택
 - 모든 작업 후 create 버튼을 클릭하여
새 프로젝트 생성



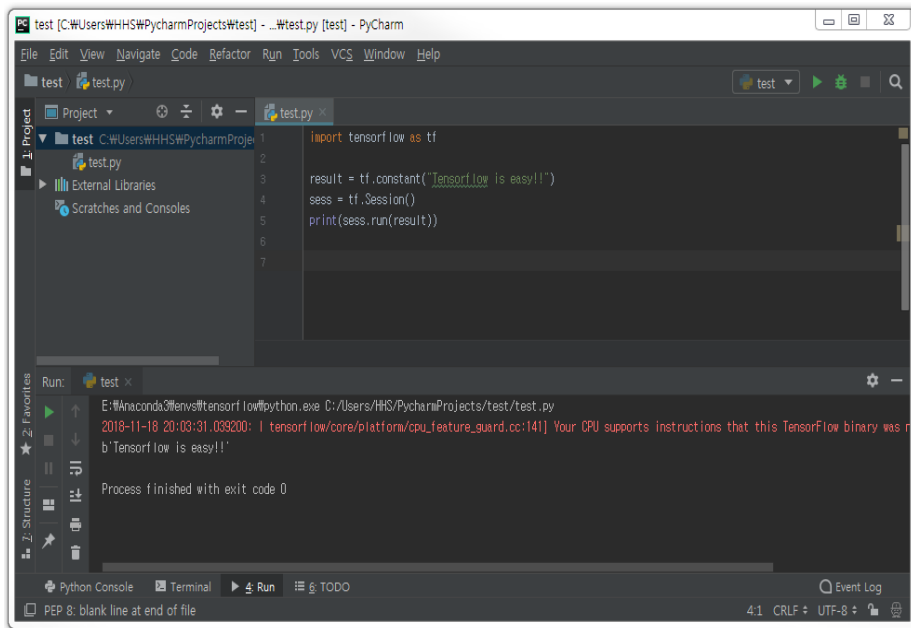
텐서플로우 개발 환경 구축

- python 파일 생성
 - 생성된 프로젝트 마우스 우 클릭
 - [New] - [python file] 선택
 - 파일명 설정 후 생성



텐서플로우 개발 환경 구축

- python 파일 생성
 - 가상환경에서 테스트한 소스코드 사용
 - 소스코드 입력 후 우측 상단 ▶ 버튼 클릭 또는, Ctrl + Shift + F10 단축키 사용
 - 가상환경 테스트 결과와 동일하게 결과값 앞에 'b' 출력 및 에러메시지 출력



텐서플로우 개발 환경 구축

- python 파일 생성

- 결과값 'b'가 출력되는 이유

- 인코딩 문제로 인하여 발생
 - 출력코드 작성 시
encoding='utf8' 코드 추가

- 예시

```
print(str(sess.run(result),  
encoding='utf8'))
```

- 예러메시지가 출력되는 이유

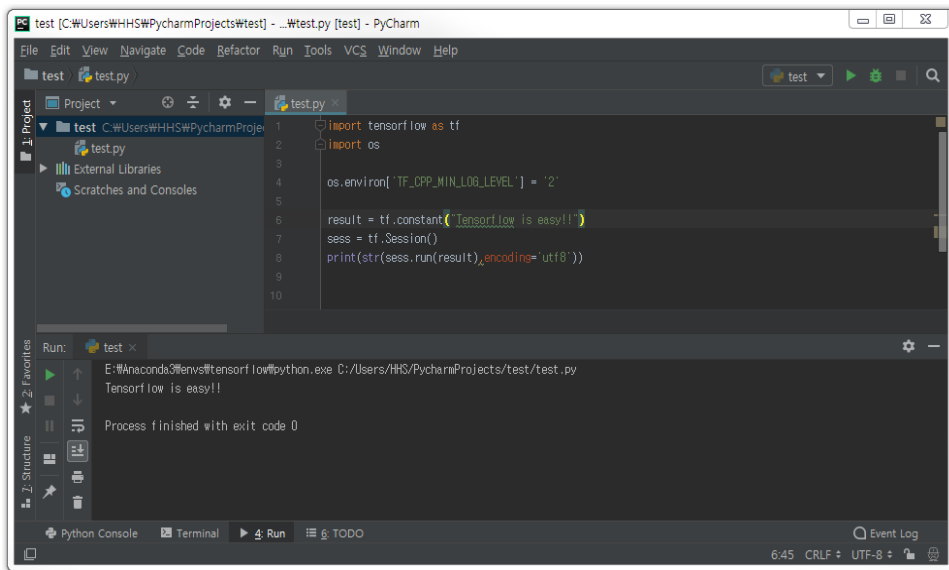
- 현재 사용하고 있는 시스템에서
AVX2 명령어를 지원하지만,
Tensorflow에서는 해당 명령어를
사용하지 않도록 빌드가 된 버전이
설치되어 있다는 내용

- import os

```
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2' 코드 추가
```

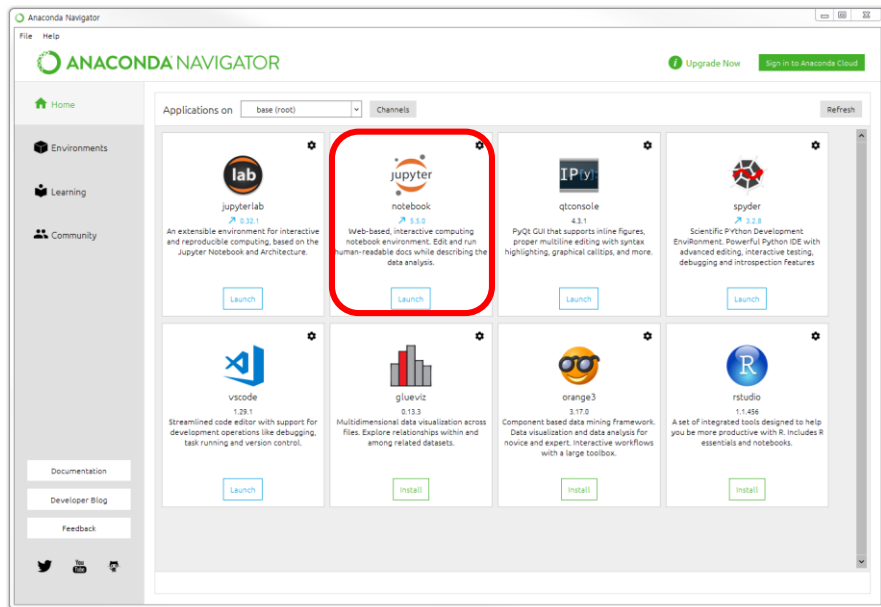
- TF_CPP_LOG_LEVEL : 로그를 담당하는 Tensorflow 환경 변수

- 값이 1은 INFO, 2는 INFO, WARNING, 3은 INFO, WARNING, ERROR 메시지가 출력되지 않음



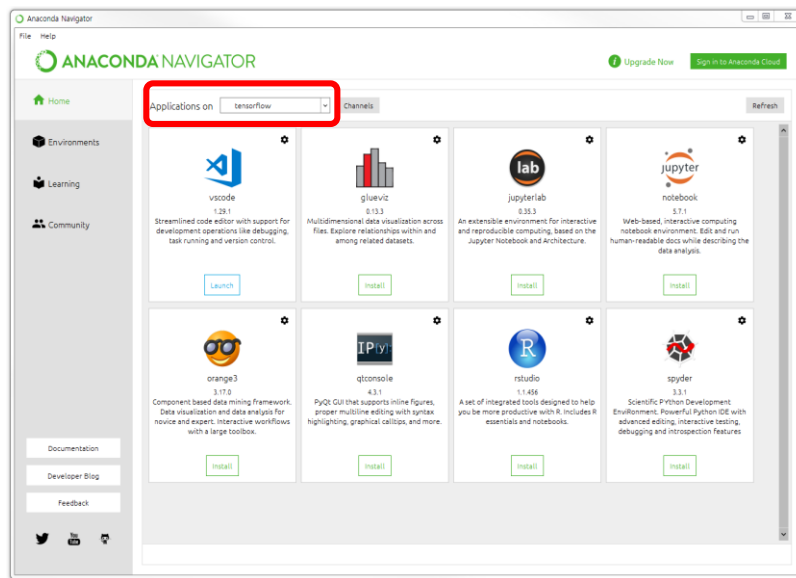
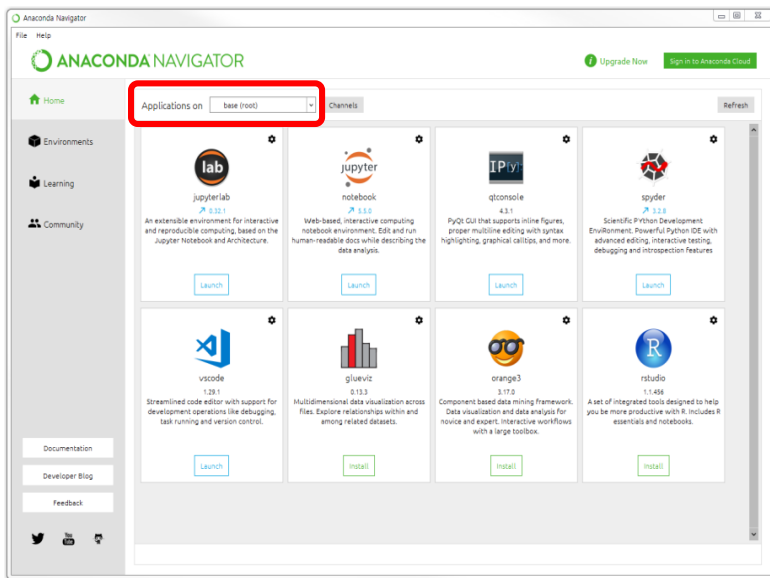
텐서플로우 개발 환경 구축

- Jupyter notebook이란
 - 크롬이나 익스플로러 같은 웹 브라우저에서 코드를 작성하고 실행이 가능한 에디터
 - Anaconda NAVIGATOR에서 쉽게 설치 가능
 - 별도로 설치할 경우 공식 홈페이지(<http://jupyter.org/>)에서 파일 다운 후 설치



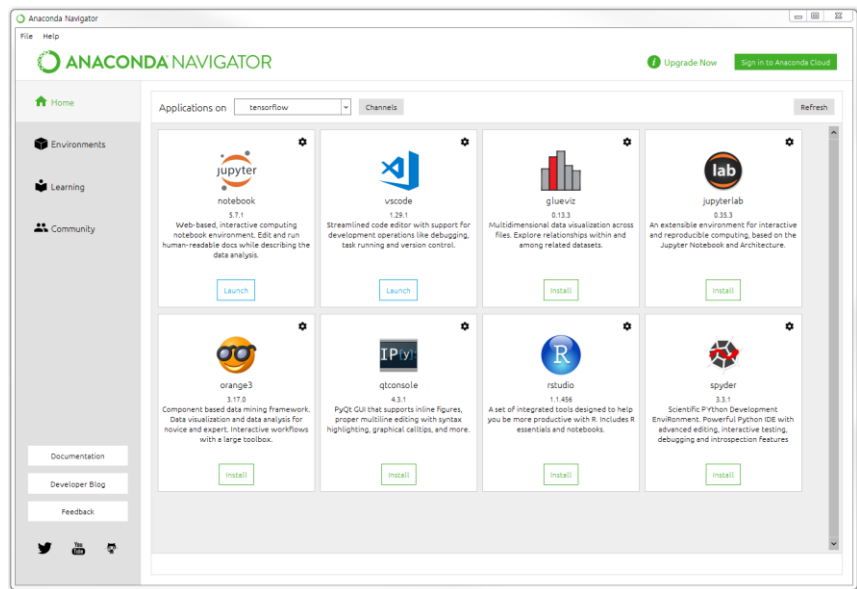
텐서플로우 개발 환경 구축

- Jupyter notebook 실행
 - NAVIGATOR 첫 실행 시 base(root)로 설정되어 있음
 - 가상환경에서 Tensorflow를 사용할 수 있도록 설정 하였음
 - Anaconda 환경을 생성했던 가상환경인 Tensorflow로 변경 후 Jupyter notebook 사용



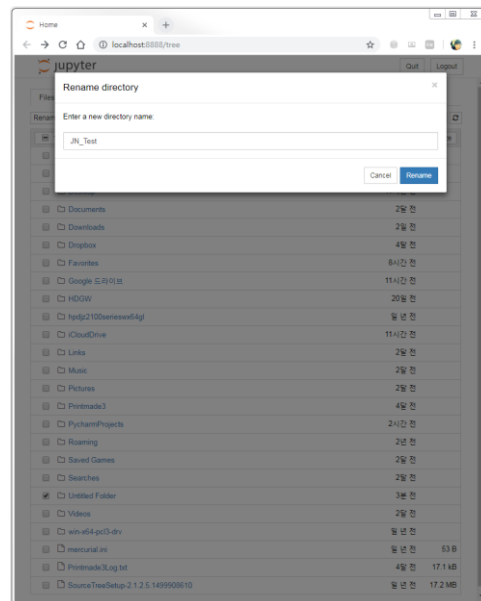
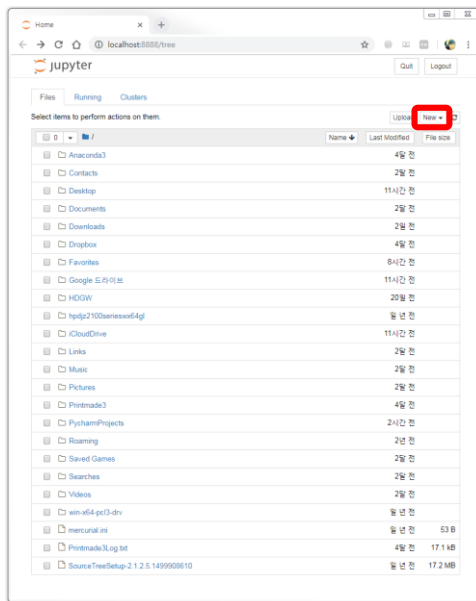
텐서플로우 개발 환경 구축

- Jupyter notebook 실행
 - Anaconda Navigator에는 여러 개의 프로그램이 제공됨
 - Launch로 표시된 것은 설치가 완료된 프로그램
 - Install로 표시된 것은 설치를 해야하는 프로그램
 - Jupyter notebook 설치 후 실행
 - Launch 버튼을 클릭하면 프로그램이 실행됨



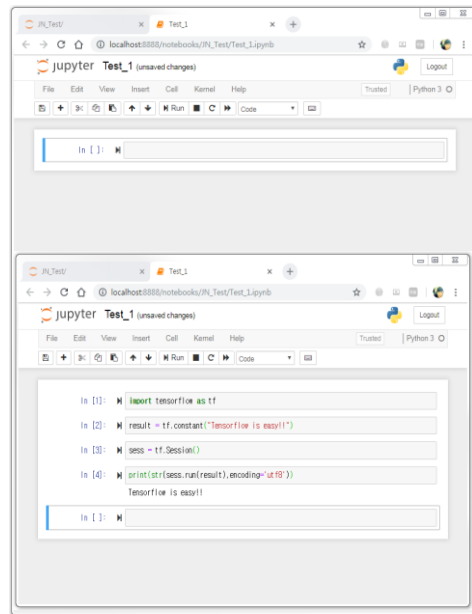
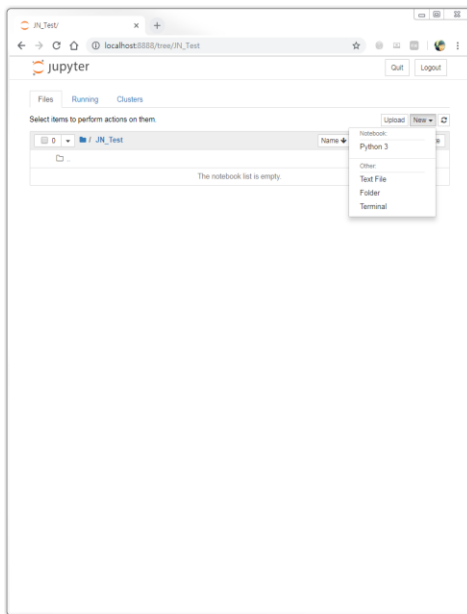
텐서플로우 개발 환경 구축

- Jupyter notebook 실행
 - Jupyter notebook 실행 후 우측 상단 [New] 클릭하여 새 폴더 생성
 - 자동으로 [Untitled Folder] 이름으로 폴더가 생성됨
 - rename 버튼을 이용하여 원하는 파일명으로 수정



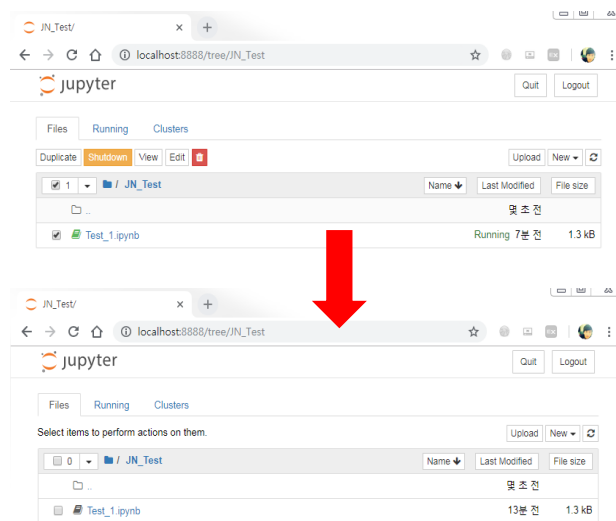
텐서플로우 개발 환경 구축

- Jupyter notebook 실행
 - 생성한 폴더로 이동하여 간단한 테스트 진행
 - 우측 상단 [New] 클릭하여 python3 실행
 - Jupyter notebook은 셀 단위로 실행됨
 - 한 줄 한 줄 소스코드를 입력하면서 해당 줄의 결과 및 에러 확인 가능
 - 화면 상단 [Run] 버튼 혹은 Shift+Enter를 이용하여 실행



텐서플로우 개발 환경 구축

- Jupyter notebook 실행
 - 작업 후에 파일을 종료하더라도 백그라운드로 실행되고 있음
 - 간단한 소스 경우 큰 문제가 발생하지 않음
 - 다수의 소스코드나 학습 데이터가 큰 경우 불필요한 자원 낭비
 - 작업 후 해당 파일은 반드시 종료해야함
 - 파일이 저장된 위치에서 보면 [Running] 상태 확인
 - 왼쪽 상단에 [Shutdown] 버튼을 이용하여 파일 종료

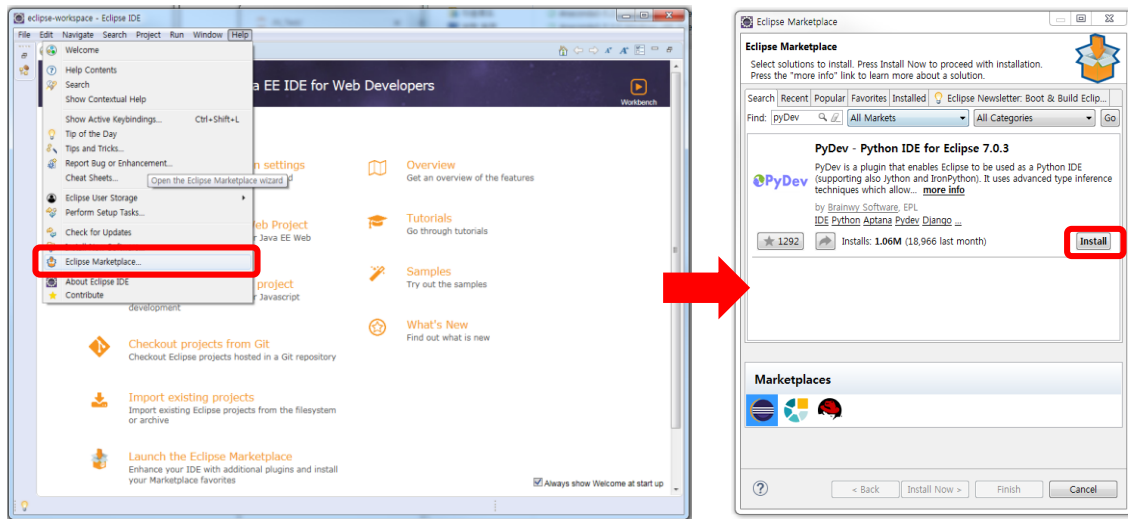


텐서플로우 개발 환경 구축

- Eclipse란
 - 다양한 언어를 지원하는 프로그래밍 통합 개발 환경
 - 다양한 플랫폼에서 사용 가능
 - 최근 OSGi를 도입하여 범용 응용 소프트웨어 플랫폼으로 진화
 - Java 언어를 사용할 때 대표적인 개발도구로서 전 세계 많은 개발자가 사용하는 개발 환경

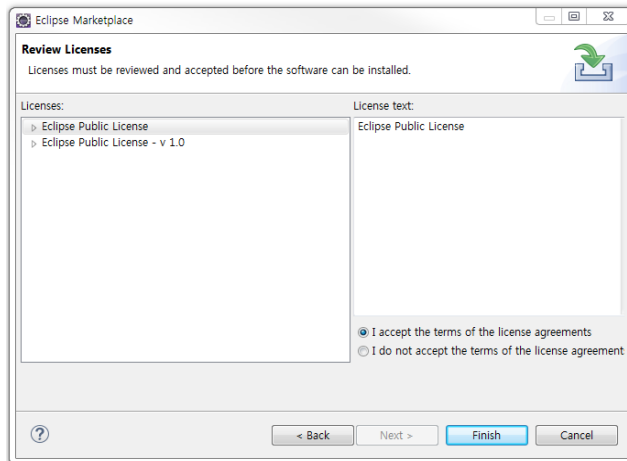
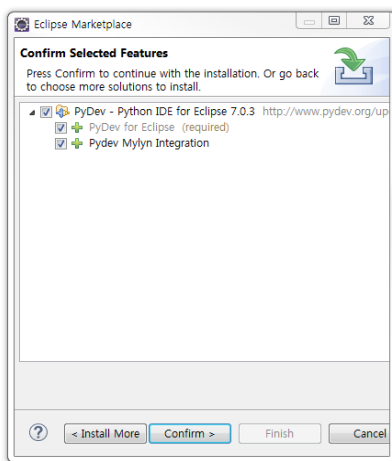
텐서플로우 개발 환경 구축

- Eclipse 실행
 - Eclipse에서 python 개발 환경을 구축하기 위해서 PyDev 플러그인 설치
 - PyDev는 python을 개발하기 위한 플러그인
 - Eclipse 실행 후 [Help] - [Eclipse Marketplace] 실행
 - PyDev 플러그인 검색 후 Install 진행



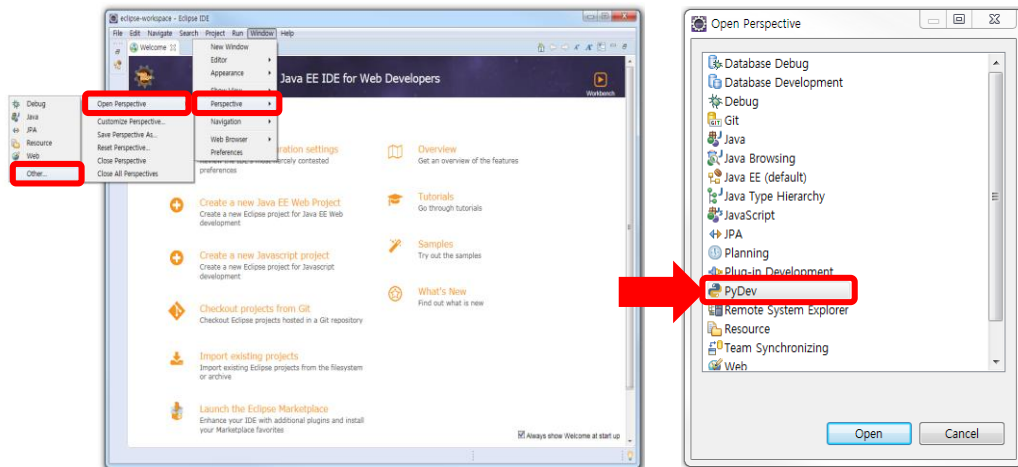
텐서플로우 개발 환경 구축

- Eclipse 실행
 - PyDev 특성 확인 후 모드 선택하여 설치 진행
 - 라이선스 확인 및 동의 후 플러그인 설치 완료



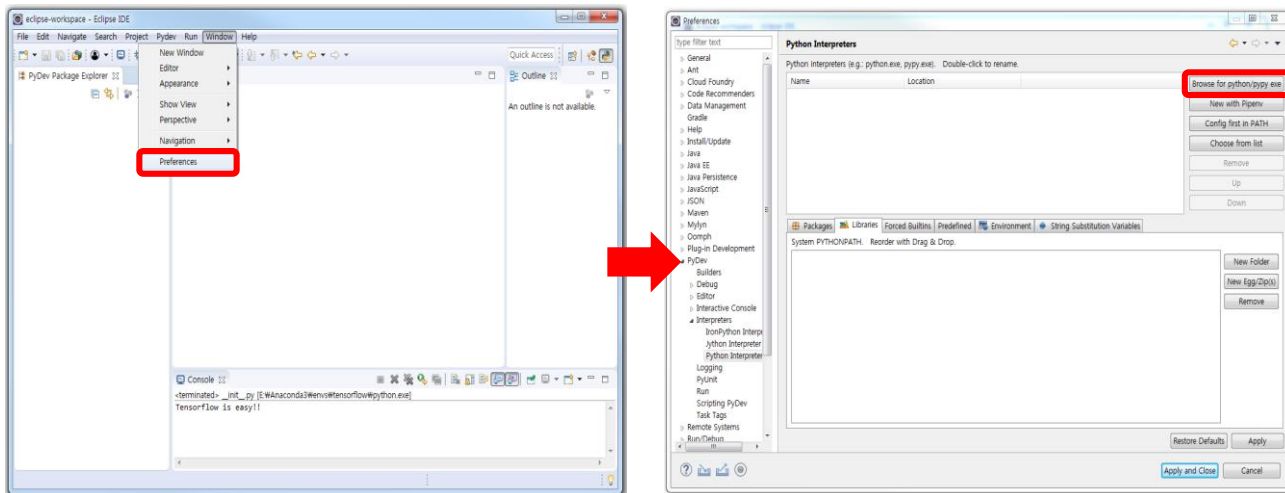
텐서플로우 개발 환경 구축

- Eclipse 실행
 - python 개발을 위한 환경설정
 - Eclipse 화면 - [Windows] - [Perspective] - [Open Perspective] - [Other] 선택
 - PyDev 선택 후 [Open] - 개발자 환경 구성 완료



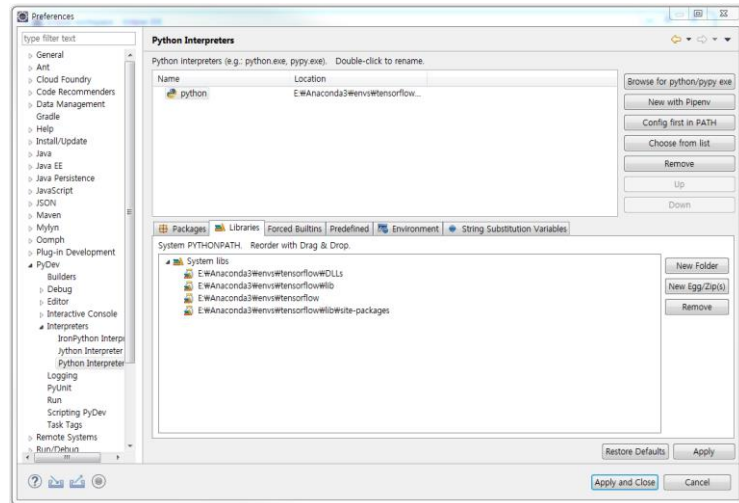
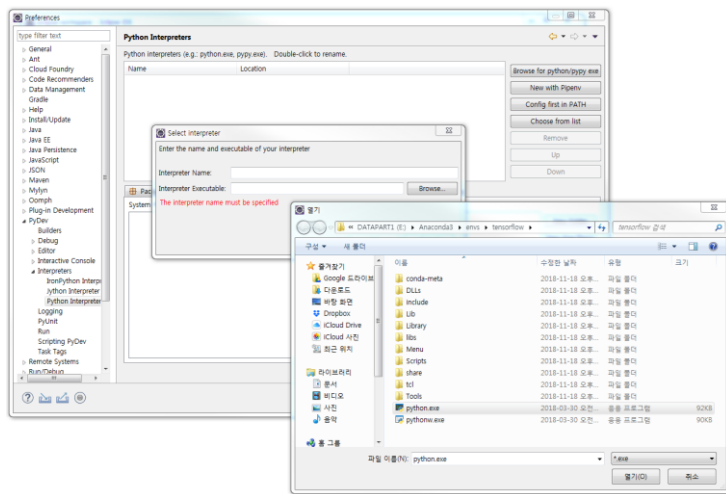
텐서플로우 개발 환경 구축

- Eclipse 실행
 - PyDev Interpreter 경로 설정
 - [Windows] - [Preferences] - [PyDev] - [Interpreters] - [Python Interpreter] - [Browse for python/pypy exe]
 - Anaconda와 함께 설치된 python 경로 설정



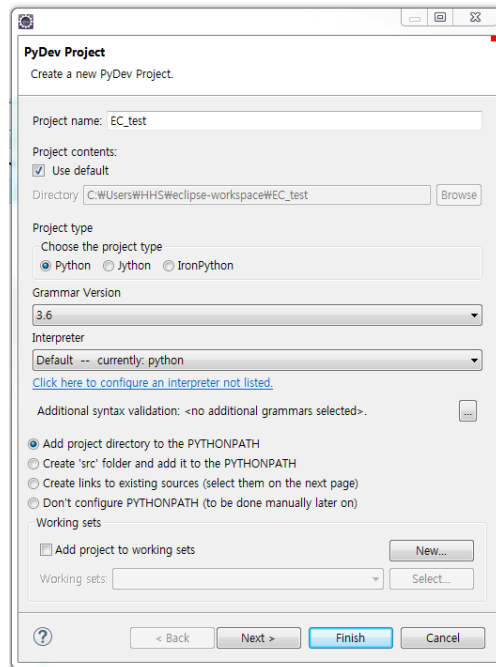
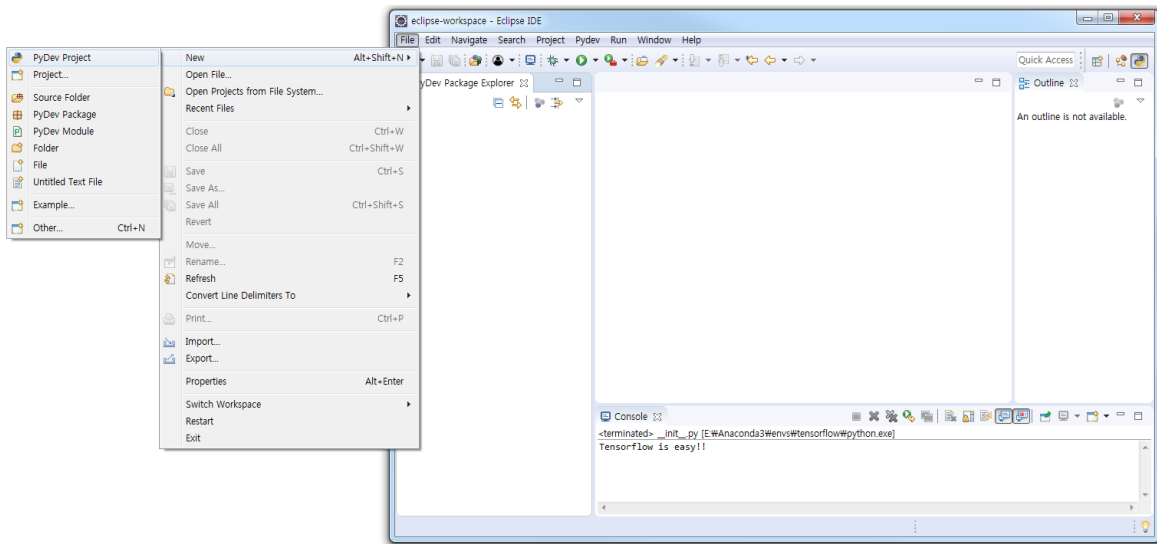
텐서플로우 개발 환경 구축

- Eclipse 실행
 - python 경로 및 라이브러리 확인 후 [Apply] - 설정완료



텐서플로우 개발 환경 구축

- Eclipse 실행
 - [File] - [New] - [PyDev Project] 새 프로젝트 생성
 - 프로젝트 명 및 옵션 설정
 - python 버전 및 기타 옵션 선택 후 [Finish] 클릭



텐서플로우 개발 환경 구축

- Eclipse 실행
 - 생성된 프로젝트 마우스 우클릭
 - [New] - [PyDev Package] 선택
 - Package명 입력 후 python 파일 생성 확인
 - python 파일 이름 변경 가능
 - 소스코드 입력 후 [Run] 버튼 혹은 Ctrl+F11을 이용하여 프로그램 실행

