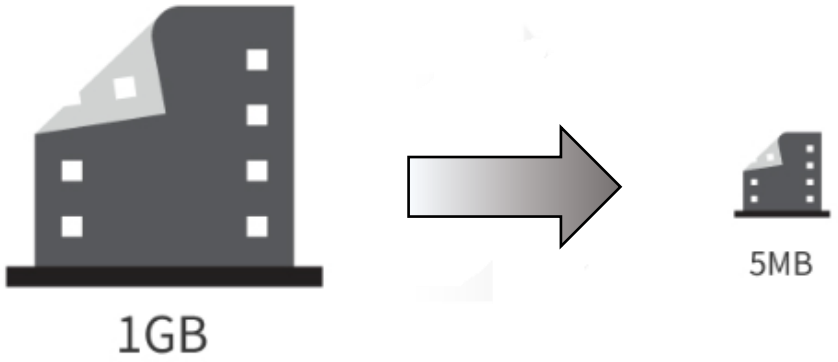
Pytorch를 이용한 딥러닝 모델 설계 및 구현

자기소개

Hardware-Friendly Algorithm



동영상 압축 / Computer Vision

Deep Learning

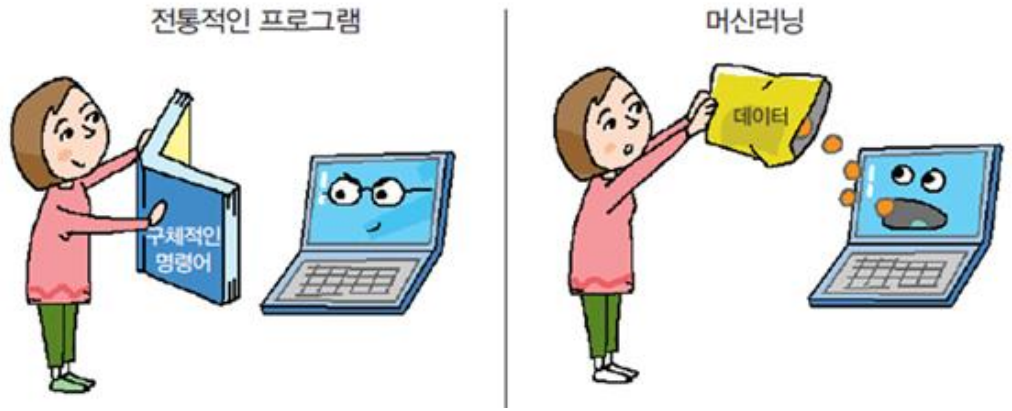
특강 계획

- 목표
  - 딥러닝?
  - 딥러닝 모델?
  - 딥러닝 모델 설계?
  - 딥러닝 모델 구현?
  - PyTorch?

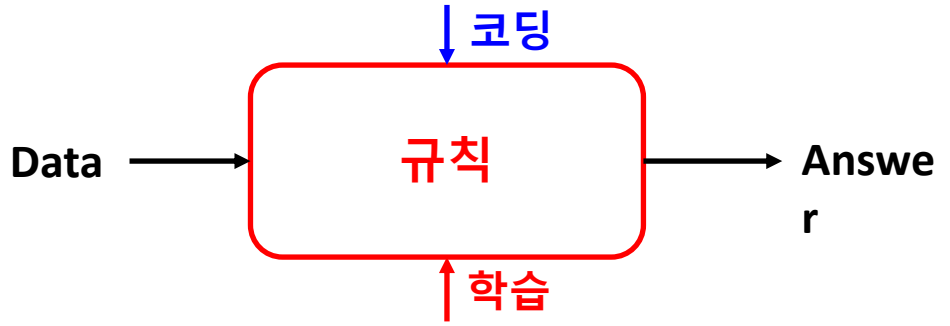
- 딥 러닝 파이토치 교과서 - 입문부터 LLM 파인튜닝까지 - WikiDocs

딥러닝(Deep Learning) 이란?

- 신경망을 깊게 쌓은 형태의 **기계학습(Machine Learning)** 방법
  - 컴퓨터가 명시적인 규칙을 프로그래머가 작성하지 않고, 데이터로부터 스스로 패턴과 규칙을 학습하여 작업을 수행하는 기술

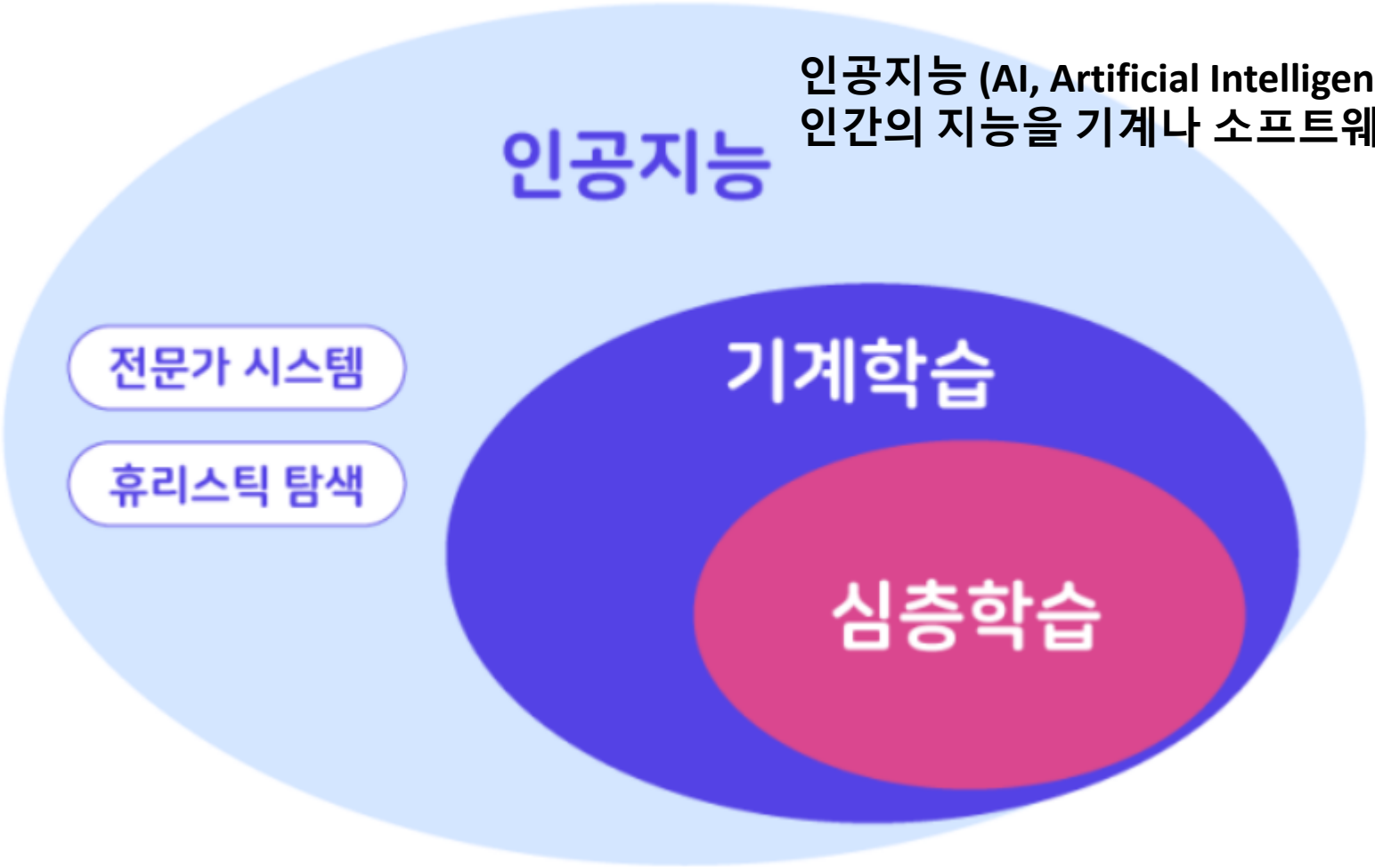


전통적인 프로그램:
직접 찾은 규칙, 논리, 지식 기반으로 사람이 Rule 생성



머신러닝:
주어진 데이터로 원하는 결과값을 얻도록 기계가 Rule 생성

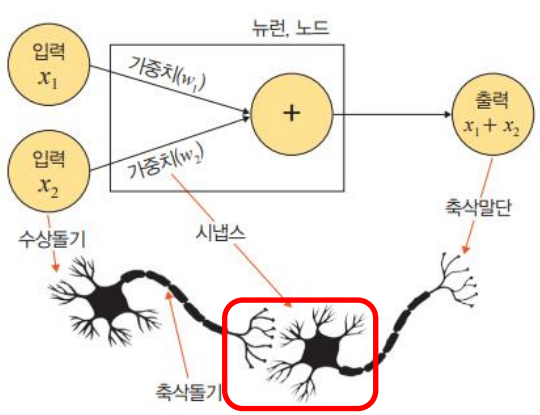
딥러닝(Deep Learning) 이란?

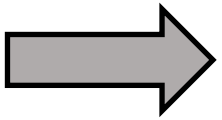


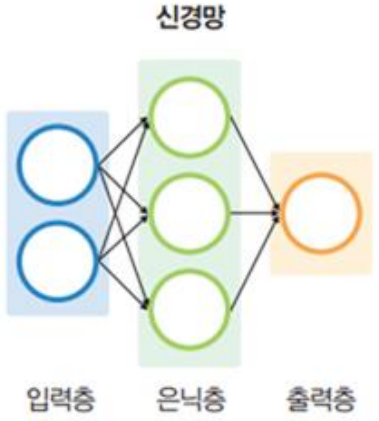
인공지능 (AI, Artificial Intelligence):
인간의 지능을 기계나 소프트웨어가 모방하거나 구현한 기술

딥러닝(Deep Learning) 이란?

- **신경망**을 깊게 쌓은 형태의 기계학습(Machine Learning) 방법
  - 뉴런: 수상돌기, 축삭 돌기, 축삭말단 등으로 구성
    - 수상돌기: 주변이나 다른 뉴런에서 자극을 받아들이고, 이 자극들을 전기적 신호 형태로 세포체와 축삭돌기로 보내는 역할
    - 축삭돌기: 다른 뉴런(수상돌기)에 신호를 전달하는 기능을 하는 뉴런의 한 부분
    - 축삭말단: 전달된 전기 신호를 받아 신경 전달 물질을 시냅스 틈새로 방출
    - 시냅스: 신경 세포들이 이루는 연결 부위로, 한 뉴런의 축삭돌기와 다음 뉴런의 수상돌기가 만나는 부분
  - 인공 신경망: 인간의 신경망 원리를(뉴런을) 모방한 수학적 모델
    - 퍼셉트론 (Perceptron)

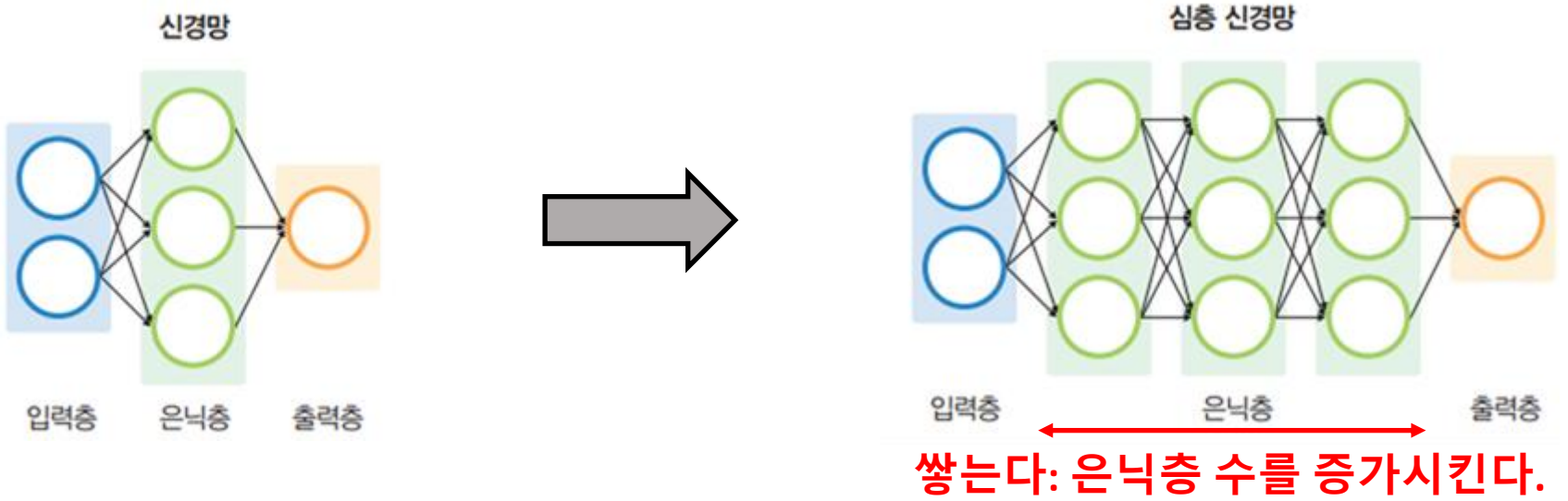






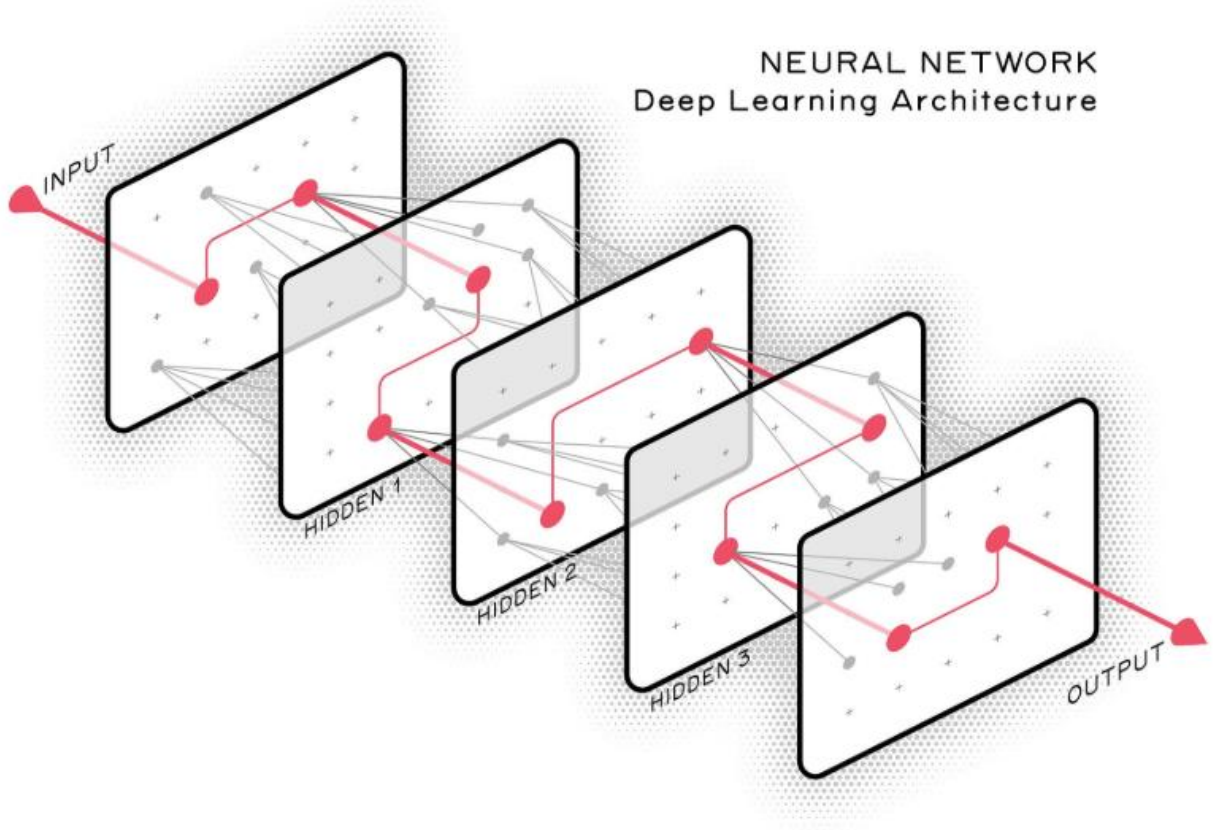
딥러닝(Deep Learning) 이란?

- 신경망을 **깊게 쌓은 형태**의 기계학습(Machine Learning) 방법
  - 은닉층의 수를 늘린다: 다양한 level (low to high)의 feature 생성, 복잡한 문제 해결
    - 다층 퍼셉트론 (Multi-Layer Perceptron)
    - 합성곱 신경망 (Convolutional Neural Network)
    - 순환 신경망 (Recurrent Neural Network)
    - Transformer



딥러닝(Deep Learning) 모델?

- 인간의 뇌 신경망 구조에서 영감을 받아 신경망을 기반으로, 컴퓨터가 스스로 데이터를 학습하고 판단할 수 있도록 컴퓨터 알고리즘 시스템



딥러닝(Deep Learning) 모델?

- 인간의 뇌 신경망 구조에서 영감을 받아 신경망을 기반으로, 컴퓨터가 스스로 데이터를 학습하고 판단할 수 있도록 컴퓨터 알고리즘 시스템

- 딥러닝 모델 설계
  - 시스템을 짜는 과정
  - 몇 개의 층, 어떻게 연결, 어떤 기준으로 평가할 것인가?
  - 기존에 잘 만들어진 모델을 활용하면 용이

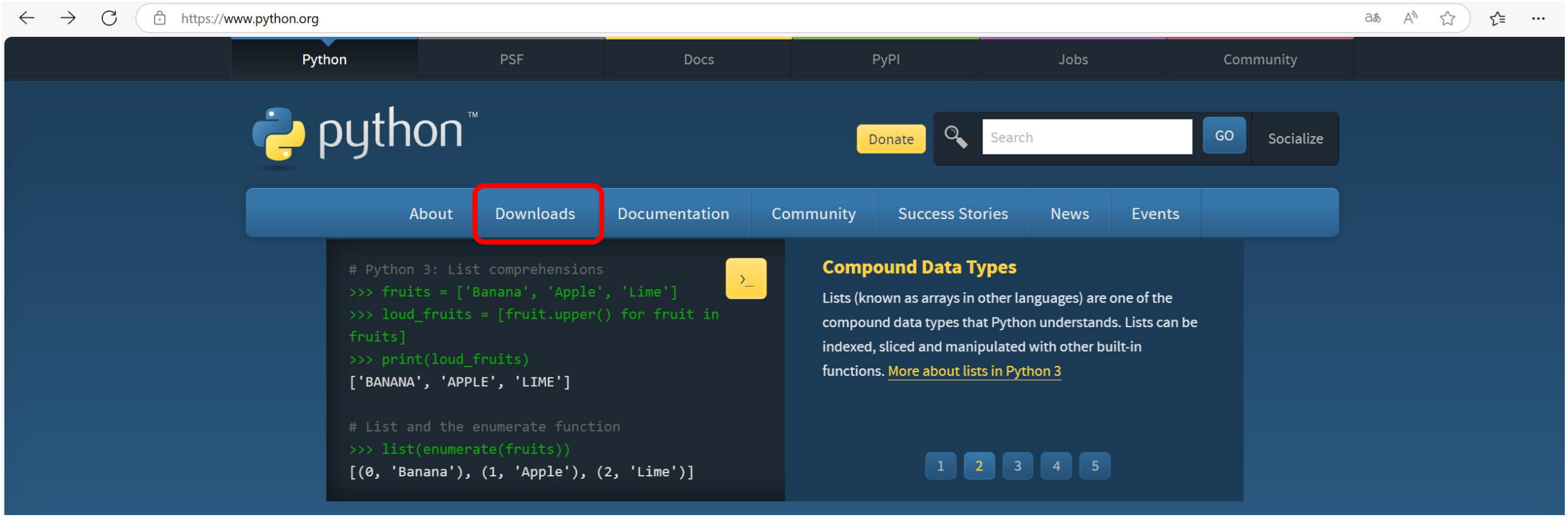
- 딥러닝 모델 구현
  - 설계한 모델을 실제 컴퓨터가 실행할 수 있도록 완성하는 과정
    - 데이터 준비 및 전처리
    - 모델 정의
    - 손실함수와 옵티마이저 설정
    - 학습 루프 구현
  - 프레임워크를 활용하면 용이해짐: PyTorch

파이토치 (Python + Torch)

- Facebook(Meta)에서 개발한 파이썬 기반 오픈소스 딥러닝 프레임워크
- Tensor 연산, 모델 정의, 학습, 추론 등 가능
- 간결하고 구현이 용이하여 개발용으로 주로 사용

파이썬 설치하기

- https://www.python.org/ 접속
  - Downloads → Windows



파이썬 설치하기

Python Releases for Windows

- Latest Python install manager - Python install manager 26.0
- Latest Python 3 Release - Python 3.14.3

Stable Releases

- Python install manager 26.0 - Feb. 23, 2026
  - Download Installer (MSIX)
  - Download MSI package
- Python 3.13.12 - Feb. 3, 2026

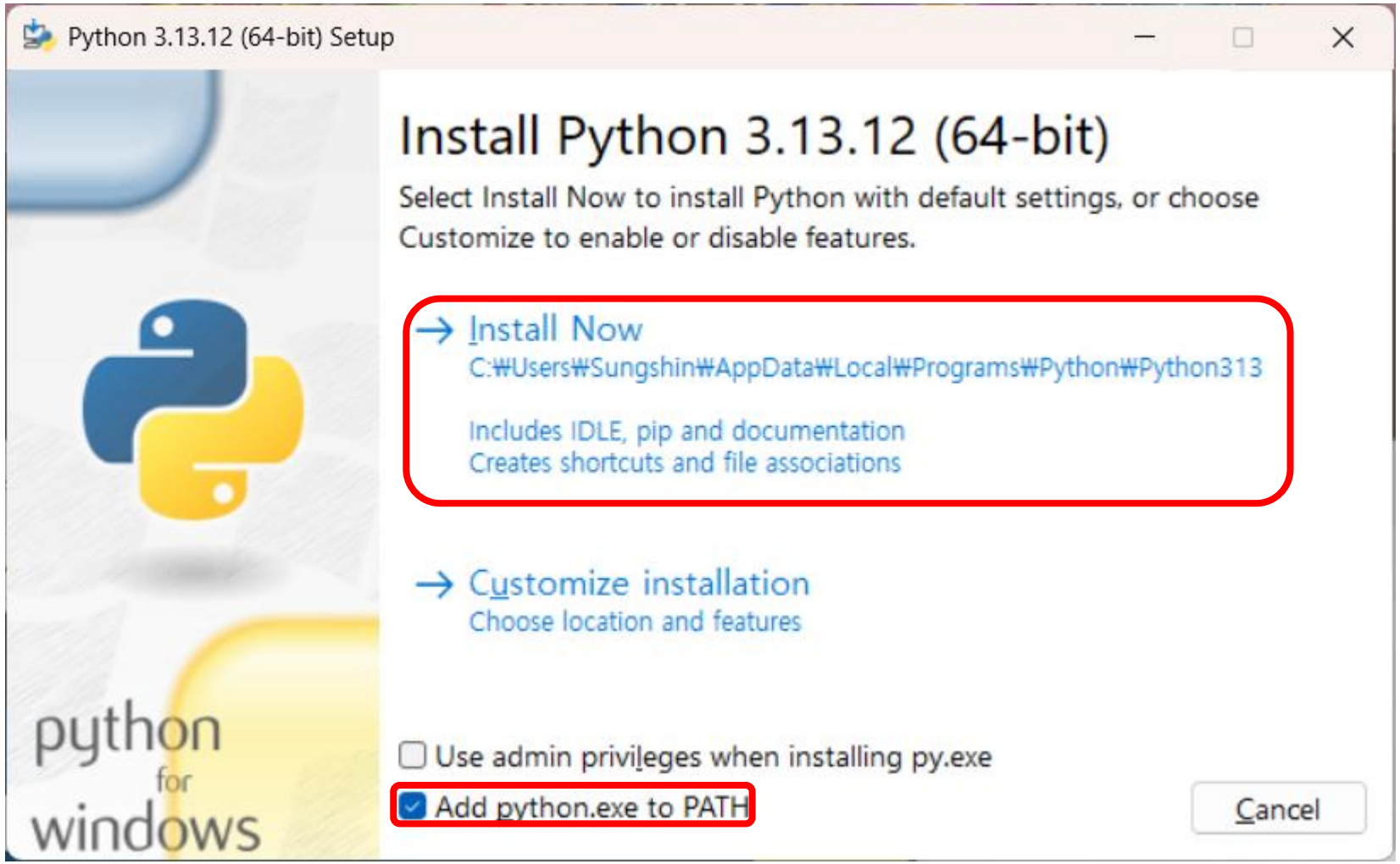
  Note that Python 3.13.12 *cannot* be used on Windows 7 or earlier.

  - Download Windows installer (64-bit)
  - Download Windows installer (32-bit)
  - Download Windows installer (ARM64)
  - Download Windows embeddable package (64-bit)
  - Download Windows embeddable package (32-bit)
  - Download Windows embeddable package (ARM64)
  - Download Windows release manifest

Pre-releases

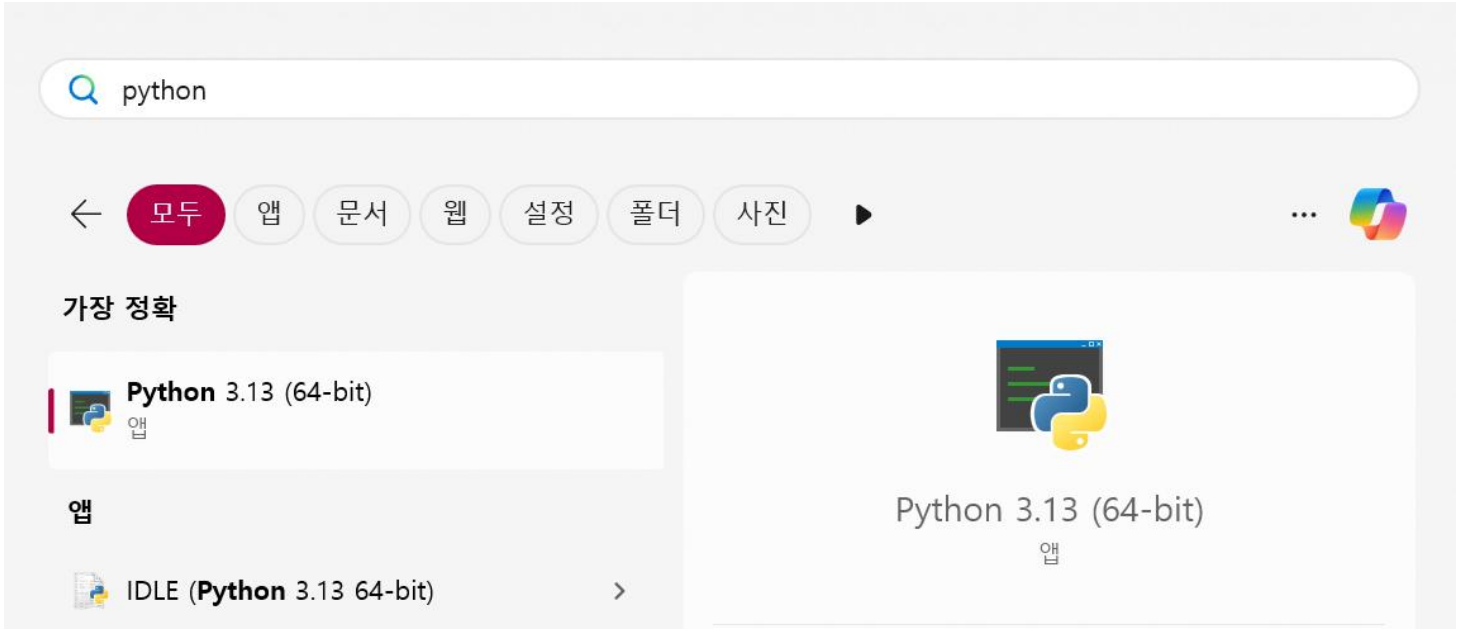
- Python 3.15.0a6 - Feb. 11, 2026
  - Download Windows installer (64-bit)
  - Download Windows installer (32-bit)
  - Download Windows installer (ARM64)
  - Download Windows embeddable package (64-bit)
  - Download Windows embeddable package (32-bit)
  - Download Windows embeddable package (ARM64)
  - Download Windows release manifest
- Python install manager 26.0 beta 2 - Feb. 6, 2026
  - Download Installer (MSIX)
  - Download MSI package
- Python install manager 26.0 beta 1 - Jan. 21, 2026

파이썬 설치하기

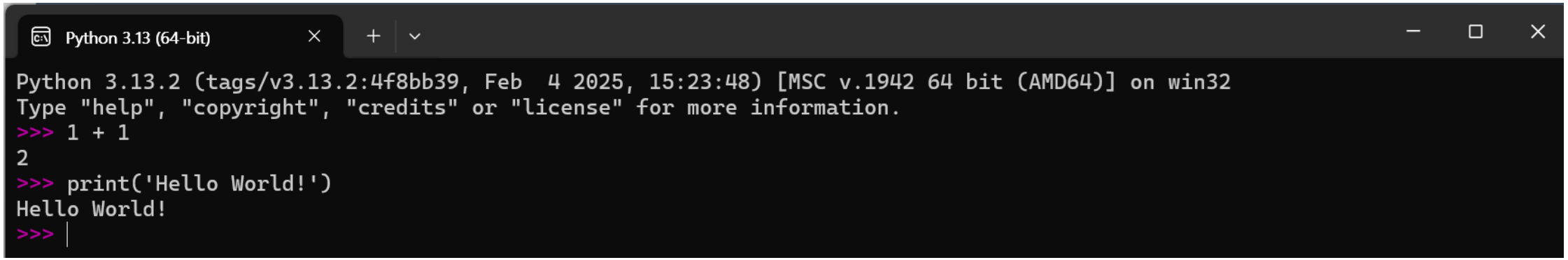

Python 3.13.12 (64-bit) Setup
Install Python 3.13.12 (64-bit)
Select Install Now to install Python with default settings, or choose
Customize to enable or disable features.
Install Now
C:₩Users₩Sungshin₩AppData₩Local₩Programs₩Python₩Python313
Includes IDLE, pip and documentation
Creates shortcuts and file associations
Customize installation
Choose location and features
python
for
windows
Use admin privileges when installing py.exe
Add python.exe to PATH
Cancel

파이썬 실행해보기

- [시작] → 'python'을 검색
  - Python 3.13 (64-bit) 실행

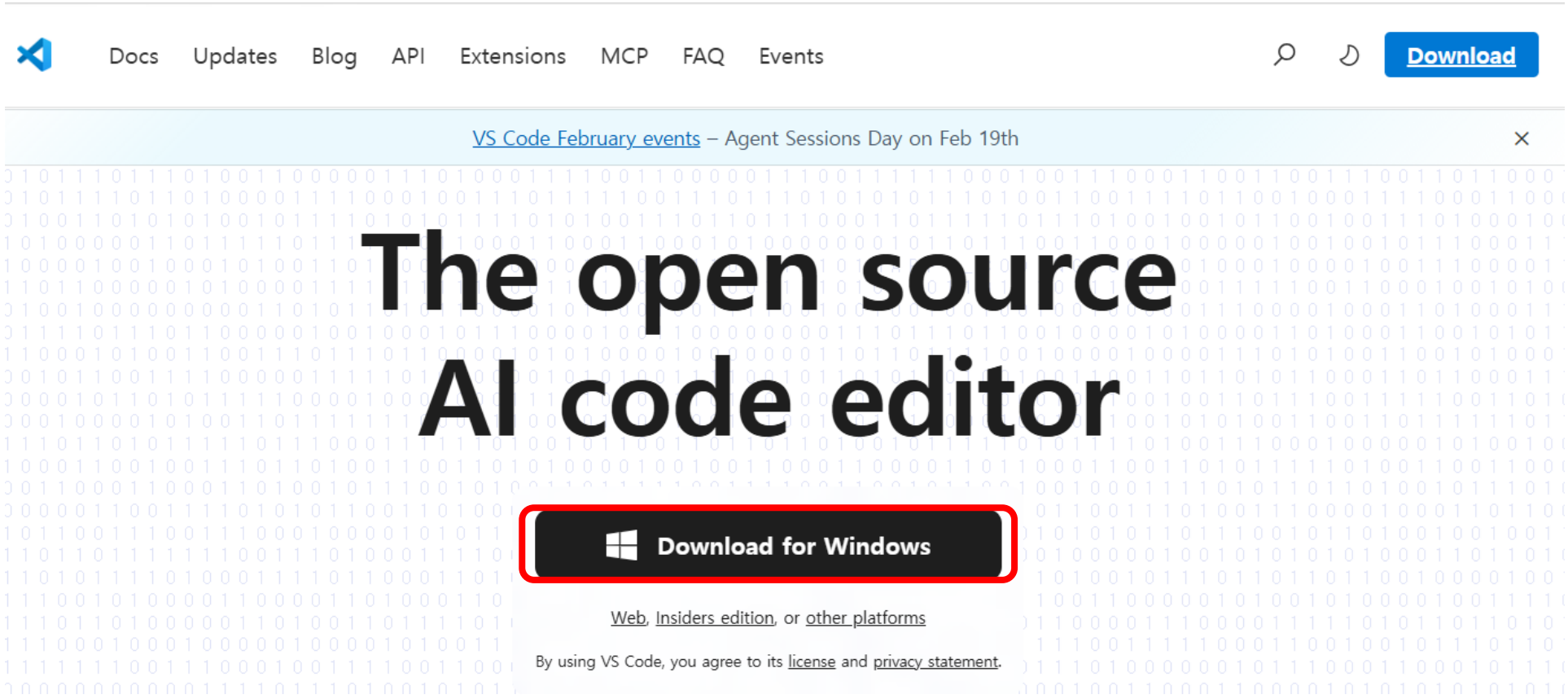


파이썬 실행해보기


Python 3.13 (64-bit)
Python 3.13.2 (tags/v3.13.2:4f8bb39, Feb 4 2025, 15:23:48) [MSC v.1942 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 1 + 1
2
>>> print('Hello World!')
Hello World!
>>>

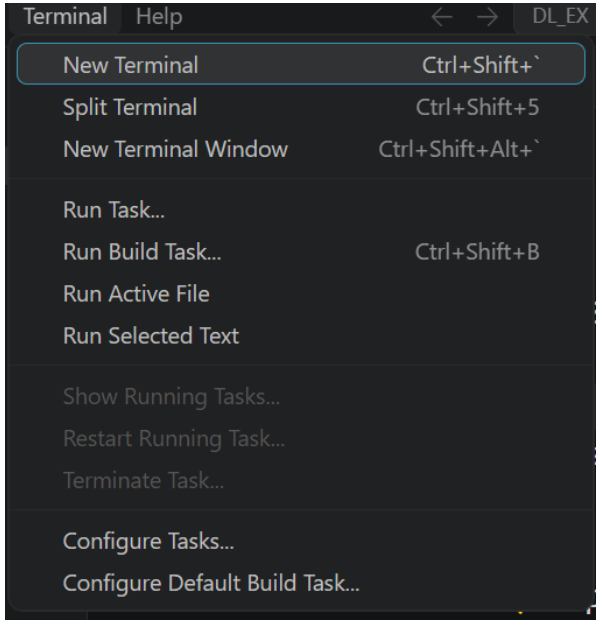
에디터 설치하기

- 코딩의 편의를 위하여 에디터 설치
  - https://code.visualstudio.com/
  - Download for Windows



파이토치 설치하기

- PyTorch 설치 (pip를 활용)
  - Terminal 메뉴 → New Terminal
  - pip install torch torchvision torchaudio



- 코랩 활용
  - https://colab.research.google.com

파이토치 설치확인

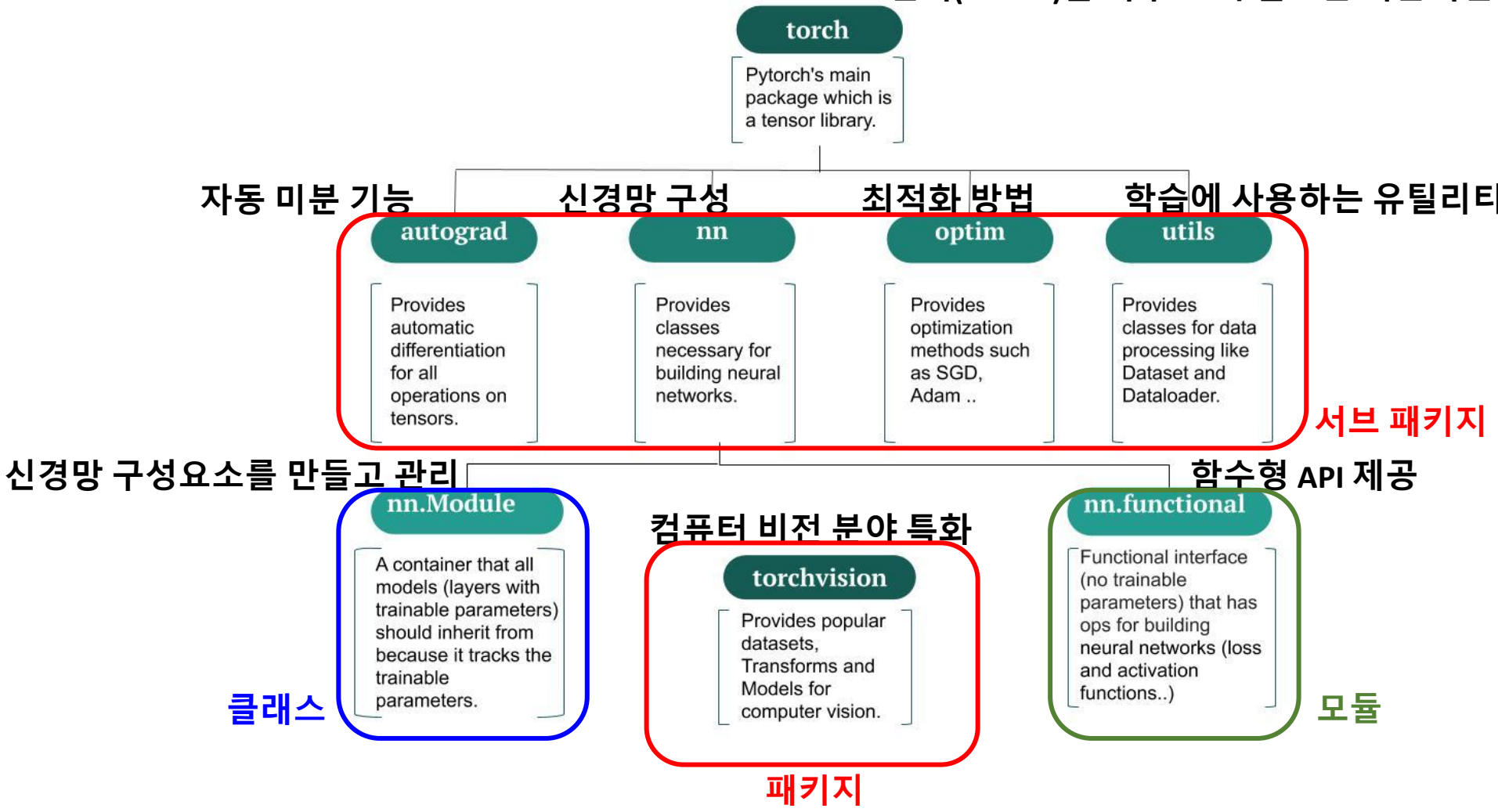
```
# Torch package import
import torch

# 텐서가 생성되는가?
x = torch.rand(5, 3)
print(x)

# GPU 사용 여부
print(torch.cuda.is_available())
```

파이토치 (PyTorch)

텐서(tensor)를 다루는 데 필요한 기본적인 기능 제공



파이토치 (PyTorch)

- torch: 텐서(tensor)를 다루는 데 필요한 기본적인 기능 제공

```
# Tensor 생성 관련
torch.tensor(data)   # 리스트나 NumPy 배열과 같은 데이터로부터 텐서를 생성
torch.zeros(shape)  # 모든 요소가 0인 텐서를 생성
torch.rand(shape)   # 0과 1 사이의 균일 분포 난수로 채워진 텐서를 생성합니다.

# Tensor 연산
torch.add(a, b)       # 두 텐서 a와 b를 더함
torch.matmul(a, b)  # 두 텐서 a와 b의 행렬 곱셈을 수행
torch.mean(tensor) # 텐서의 모든 요소의 평균을 계산

# Tensor 조작 함수
torch.reshape(tensor, new_shape)  # 텐서의 형태 변경
torch.cat(tensors, dim)                 # 텐서를 주어진 차원(dim)을 따라 합침

# Device 관련
torch.cuda.is_available()      # CUDA (GPU) 를 사용할 수 있는지 여부 확인
torch.device()                 # Tensor나 Model이 위치할 장치(device)를 나타내는 객체 생성
tensor.to(device)              # Tensor 장치 이동
model.to(device)               # Model 장치 이동
```

파이토치 (PyTorch)

- torch.autograd: 자동 미분 엔진, 손실 함수의 경사를 자동으로 계산 해 줌

```
for input, answer in 학습데이터:        # 학습 루프

  # 1. 경사 초기화: 이전 배치에서 계산된 경사 값을 초기화
  optimizer.zero_grad()

  # 2. 순전파(Forward Pass): 현재 파라미터로 예측값을 계산
  prediction = model(input)

  # 3. 손실 계산: 예측 값과 정답 사이의 오차 계산
  loss = loss_fn(prediction, input)

  # 4. 역전파(Backward Pass): 손실에 대한 각 파라미터의 경사를 계산
  # 이 때, torch.autograd를 활용하여 자동으로 계산
  loss.backward()

 # 5. 파라미터 업데이트: 계산된 경사를 이용해 파라미터를 업데이트합니다.
  optimizer.step()
```

파이토치 (PyTorch)

- torch.nn: 신경망 모델 구축에 필요한 구성 요소를 제공

```
import torch.nn as nn    # 관례적으로 별칭을 부여하여 간결하게 사용

# 1. Layer 구성
linear_layer = nn.Linear(in_features=10, out_features=5)                    # 완전 연결 계층
conv_layer = nn.Conv2d(in_channels=3, out_channels=16, kernel_size=3)       # 2차원 컨볼루션 계층
maxpool_layer = nn.MaxPool2d(kernel_size=2)                                 # 2차원 맥스 풀링 계층

# 2. 활성화 함수
relu_activation = nn.ReLU()                 # ReLU(Rectified Linear Unit)
sigmoid_activation = nn.Sigmoid()           # 시그모이드(Sigmoid) 함수
softmax_activation = nn.Softmax(dim=1)      # 소프트맥스(Softmax) 함수

# 3. 손실 함수
cross_entropy_loss = nn.CrossEntropyLoss()    # 교차 엔트로피 손실 함수
Mse_loss = nn.MSELoss()                       # 오차 제곱 평균(Mean Squared Error) 손실 함수
bce_loss = nn.BCELoss()                       # 이진 교차 엔트로피 손실 함수
```

파이토치 (PyTorch)

- torch.optim: 손실 값을 줄이기 위해 모델 매개변수를 갱신하는 방법

```
import torch.optim as optim    # 관례적으로 별칭을 부여하여 간결하게 사용

optimizer_sgd = optim.SGD(model.parameters(), lr=0.01, momentum=0.9)      # SGD (Stochastic Gradient Descent)
optimizer_adagrad = optim.Adagrad(model.parameters(), lr=0.01)            # AdaGrad (Adaptive Gradient)
optimizer_adam = optim.Adam(model.parameters(), lr=0.001)                 # ADAM (Adaptive Moment Estimation)


# 학습 루프안에서
    optimizer_adam.zero_grad()   # 1. 경사 초기화
    output = model(input_data)   # 2. 순전파
    loss = loss_fn(output, target)   # 3. 손실 계산
    loss.backward()              # 4. 역전파를 통해 경사 계산
    optimizer_adam.step()        # 5. 계산된 경사로 매개변수 업데이트
```

파이토치 (PyTorch)

- torch.utils: 모델 학습에 필수적인 유틸리티 기능 제공

```
from torch.utils.data import random_split, DataLoader     # 관례적으로 별칭을 부여하여 간결하게 사용
                                                          # random_split : 데이터셋을 무작위로 분할
                                                          # DataLoader: 효율적인 데이터셋 로드를 위한 클래스

# 임의로 training(dataset 의 80%), validation(dataset 의 20%) data 할당방법
train_size = int(len(full_dataset) * 0.8)
val_size = len(full_dataset) - train_size
train_dataset, val_dataset = random_split(full_dataset, [train_size, val_size]) # dataset 을 주어진 비율대로 랜덤하게 할당

train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True)    # 분할된 학습 dataset 을 배치단위로 묶어주기
Val_loader = DataLoader(val_dataset, batch_size=32, shuffle=False)       # 분할된 점검 dataset 을 배치단위로 묶어주기

for batch, (images, labels) in enumerate(train_loader):                  # 배치단위로 읽어 오면서 학습하기
    …
```

파이토치 (PyTorch)

- torch.nn.Module: 신경망 모델 만들때 사용

```
import torch
import torch.nn as nn

class MyModel(nn.Module):                    # nn.Module 상속: 레이어, 매개변수, 처리 방식들을 알아서 관리
    def __init__(self):                      # 구성에 필요한 레이어들 정의
        super(MyModel, self).__init__()
        self.linear1 = nn.Linear(10, 5)
        self.relu = nn.ReLU()
        self.linear2 = nn.Linear(5, 1)

    def forward(self, x):                    # __init__에서 정의한 계층들을 활용하여 원하는 모델 구성
        x = self.linear1(x)
        x = self.relu(x)
        x = self.linear2(x)
        return x

model = MyModel()
output = model(input_tensor)                 # __call__메소드를 활용하여 객체를 함수처럼 사용시 forward 메소드 수행
```

파이토치 (PyTorch)

- torch.nn.functional: 함수형 인터페이스, **학습할 매개변수가 없는 연산을 함수 형태로 제공**

```
import torch.nn.functional as F                    # 관례적으로 별칭을 부여하여 간결하게 사용

# 1. 활성화 함수
F.relu(input)              # ReLU(Rectified Linear Unit) 함수 적용
F.sigmoid(input)           # 시그모이드 함수 적용
F.softmax(input, dim)      #  소프트맥스 함수 적용

# 2. 풀링
F.max_pool2d(input, kernel_size, stride)     # 2차원 텐서에 대해 최대값 풀링 적용
F.avg_pool2d(input, kernel_size, stride)     # 2차원 텐서에 대해 평균값 풀링 적용

# 3. 손실함수
F.cross_entropy(input, target)               # 교차 엔트로피 손실 계산
F.mse_loss(input, target)                    # 오차 제곱의 평균 손실 계산
```

파이토치 (PyTorch)

- torchvision: 컴퓨터 비전 분야를 위한 다양한 기능을 지원

```
from torchvision import datasets, transforms, models          # 관례적으로 별칭을 부여하여 간결하게 사용

# 1`. transforms: 이미지 데이터에 대한 다양한 전처리 및 증강 변환 함수 제공
# 학습이나 추론시 필요한 데이터 형태로 변환하기 위한 함수들을 나열
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])

# 2. datasets: 컴퓨터 비전 분야에서 주로 사용하는 dataset 활용
train_data = datasets.MNIST(root="data", train=True, download=True, transform=transform)

# 3. models: 컴퓨터 비전 분야에서 주로 사용하는 사전 학습된 모델 활용
model = models.resnet18(weights=models.ResNet18_Weights.IMAGENET1K_V1)
```