

FVE3011

AI/ML/Pytorch

담당교수: 최 학남 xncui@inha.ac.kr



인하대학교

- <http://pytorch.org/>
 1. Deep learning framework that puts Python first
 2. Tensor Computation (like numpy) with strong GPU acceleration



A graph is created on the fly

```
from torch.autograd import Variable  
  
x = Variable(torch.randn(1, 10))  
prev_h = Variable(torch.randn(1, 20))  
W_h = Variable(torch.randn(20, 20))  
W_x = Variable(torch.randn(20, 10))
```



Installing PyTorch

- On the latest pip and numpy packages
- Anaconda is recommended package manager

START LOCALLY

Select your preferences and run the install command. Stable represents the most currently tested and supported version of PyTorch. This should be suitable for many users. Preview is available if you want the latest, not fully tested and supported, builds that are generated nightly. Please ensure that you have **met the prerequisites below (e.g., numpy)**, depending on your package manager. Anaconda is our recommended package manager since it installs all dependencies. You can also **install previous versions of PyTorch**. Note that LibTorch is only available for C++.

PyTorch Build	Stable (1.13.1)		Preview (Nightly)	
Your OS	Linux	Mac	Windows	
Package	Conda	Pip	LibTorch	Source
Language	Python		C++ / Java	
Compute Platform	CUDA 11.6	CUDA 11.7	ROCm 5.2	CPU
Run this Command:	conda install pytorch torchvision torchaudio pytorch-cuda=11.6 -c pytorch -c nvidia			

NOTE: PyTorch LTS has been deprecated. For more information, see [this blog](#).

CUDA-Enabled NVIDIA Quadro and NVIDIA RTX			
NVIDIA Quadro and NVIDIA RTX Desktop GPUs		NVIDIA Quadro and NVIDIA RTX Mobile GPUs	
GPU	Compute Capability	GPU	Compute Capability
RTX A6000	8.6	RTX A5000	8.6
RTX A5000	8.6	RTX A4000	8.6
RTX A4000	8.6	RTX A3000	8.6
T1000	7.5	RTX A2000	8.6
T600	7.5	RTX 5000	7.5
T400	7.5	RTX 4000	7.5
Quadro RTX 8000	7.5	RTX 3000	7.5
Quadro RTX 6000	7.5	T2000	7.5
Quadro RTX 5000	7.5	T1200	7.5
Quadro RTX 4000	7.5	T1000	7.5
Quadro GV100	7.0	T600	7.5
Quadro GP100	6.0	T500	7.5
Quadro P6000	6.1	P620	6.1
Quadro P5000	6.1	P520	6.1
Quadro P4000	6.1	Quadro P5200	6.1
Quadro P2200	6.1	Quadro P4200	6.1
Quadro P2000	6.1	Quadro P3200	6.1
Quadro P1000	6.1	Quadro P5000	6.1
Quadro P620	6.1	Quadro P4000	6.1
Quadro P600	6.1	Quadro P3000	6.1
Quadro P400	6.1	Quadro P2000	6.1
Quadro M6000 24GB	5.2	Quadro P1000	6.1
Quadro M6000	5.2	Quadro P600	6.1
Quadro K6000	3.5	Quadro P500	6.1
Quadro M5000	5.2	Quadro M5500M	5.2
Quadro K5200	3.5	Quadro M2200	5.2
Quadro K5000	3.0	Quadro M1200	5.0
Quadro M4000	5.2	Quadro M620	5.2
Quadro K4200	3.0	Quadro M520	5.0
Quadro K4000	3.0	Quadro K6000M	3.0

Installing PyTorch



Prerequisites

1. Install **Anaconda**
2. Install **CUDA**, if your machine has a **CUDA-enabled GPU**.
3. If you want to build on Windows, Visual Studio with MSVC toolset, and NVTX are also needed.
The exact requirements of those dependencies could be found out [here](#).
4. Follow the steps described here: <https://github.com/pytorch/pytorch#from-source>

Installing PyTorch



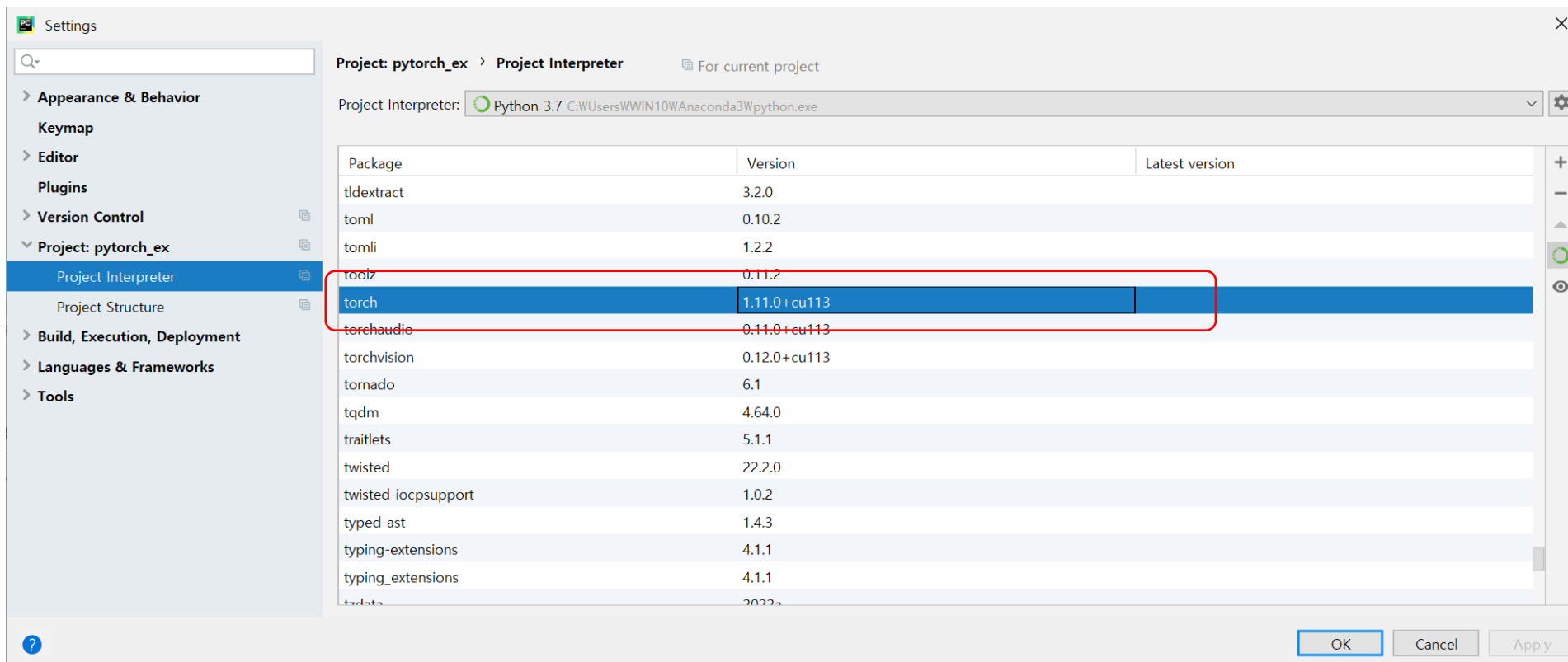
```
import torch  
x=torch.rand(5,3)  
print(x)
```



```
tensor( [[0.4033, 0.3859, 0.8813],  
         [0.8811, 0.7242, 0.5033],  
         [0.8249, 0.2634, 0.3112],  
         [0.5948, 0.1092, 0.6213],  
         [0.7350, 0.9898, 0.9165]])
```

Installing pytorch(PyCharm)

- File → settings → Project → Project interpreter

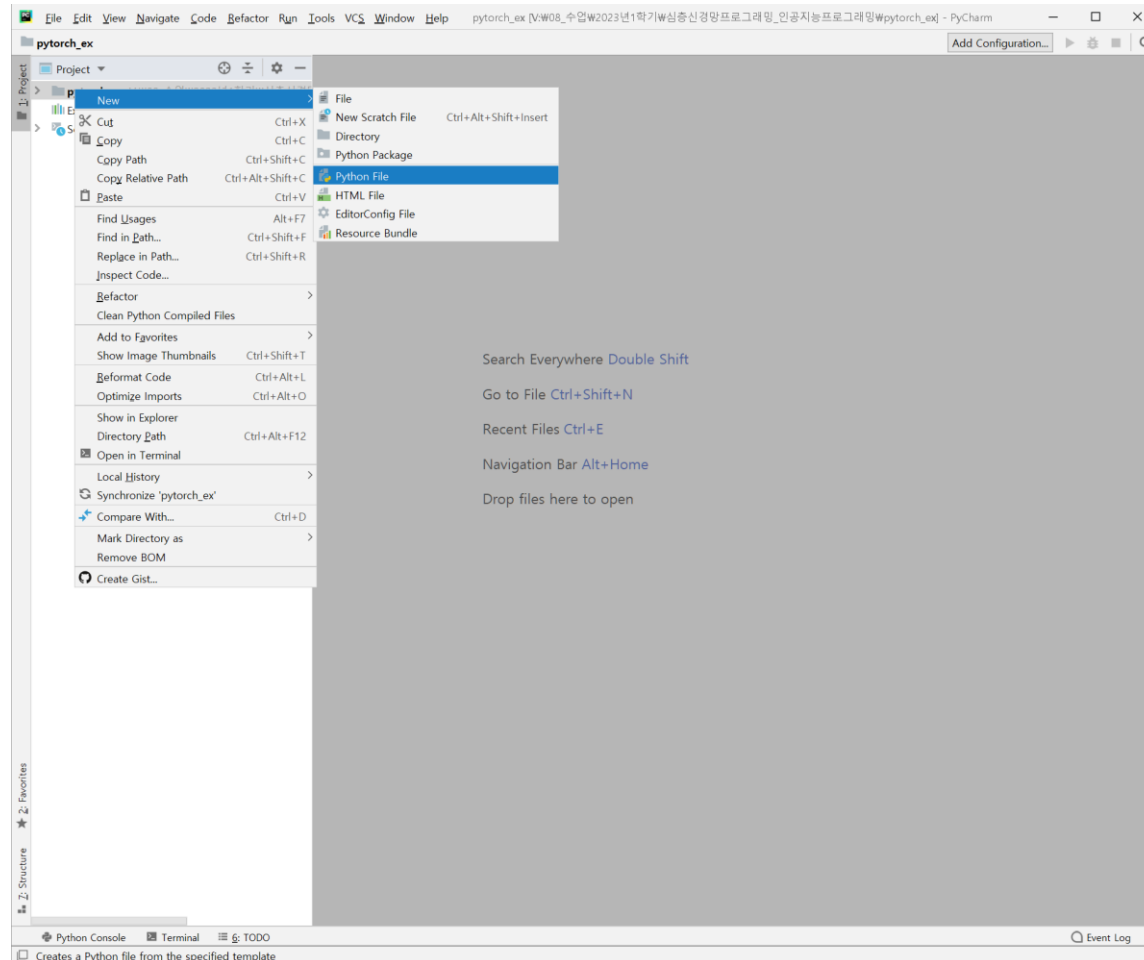


The image shows the PyCharm Settings dialog, specifically the 'Project Interpreter' tab for a project named 'pytorch_ex'. The interpreter is set to 'Python 3.7 C:\Users\WIN10\Anaconda3\python.exe'. A table lists installed packages and their versions, with 'torch' highlighted in blue and a red box around it. The table also shows the latest version for each package.

Package	Version	Latest version
tlextract	3.2.0	
toml	0.10.2	
tomli	1.2.2	
toolz	0.11.2	
torch	1.11.0+cu113	
torchaudio	0.11.0+cu113	
torchvision	0.12.0+cu113	
tornado	6.1	
tqdm	4.64.0	
traitlets	5.1.1	
twisted	22.2.0	
twisted-iocpsupport	1.0.2	
typed-ast	1.4.3	
typing-extensions	4.1.1	
typing_extensions	4.1.1	
tzdata	2022.1	

Installing pytorch(PyCharm)

- File → new project



Installing pytorch

A screenshot of an IDE window titled 'pytorch_ex [V:\W08_수업\2023년1학기\심층신경망프로그래밍_인공지능프로그래밍\pytorch_ex] - ...\ex_1.py'. The left sidebar shows a project tree with 'pytorch_ex' and 'ex_1.py'. The main editor displays a Python script in 'ex_1.py' with three lines: 'import torch', 'x = torch.rand(5, 3)', and 'print(x)'. A yellow banner above the code states 'No Python interpreter configured for the project' with a 'Configure Python interpreter' link. The bottom panel shows the 'Run' output for 'ex_1'. The output text is: 'C:\Users\WIN10\Anaconda3\python.exe V:/08_수업/2023년1학기/심층신경망프로그래밍_인공지능프로그래밍/pyt', followed by a 5x3 tensor of random values: 'tensor([[0.0502, 0.2461, 0.7299], [0.2371, 0.5035, 0.6506], [0.1398, 0.9607, 0.9621], [0.7621, 0.1391, 0.8091], [0.3967, 0.4189, 0.4299]])'. Below the tensor, it says 'Process finished with exit code 0'. The bottom status bar shows 'Python Console', 'Terminal', 'Run', and 'TODO' tabs, along with 'Event Log', '2:20', 'UTF-8', '4 spaces', and '<No interpreter>'.

Tensors

Tensors are similar to numpy's ndarrays, with the addition being that Tensors can also be used on a GPU to accelerate computing.

```
In [3]: from __future__ import print_function
import torch
```

Construct a 5x3 matrix, uninitialized:

```
In [5]: x = torch.Tensor(5,3)
print(x)
```

```
1.000000e-36 *
 0.0000  0.0000  0.0000
 0.0000  0.0000  0.0000
 0.4113  0.0000  0.0000
 0.0000  0.0001  0.0000
 1.8967  0.0000  0.0000
[torch.FloatTensor of size 5x3]
```

Tensors

Construct a randomly initialized matrix

```
In [9]: x = torch.rand(5,3)
        print(x)

0.4381  0.1222  0.1948
0.6345  0.0023  0.4593
0.2548  0.3231  0.5043
0.2990  0.9189  0.7335
0.0187  0.8618  0.4062
[torch.FloatTensor of size 5x3]
```

Get its size

```
In [11]: print(x.size())

torch.Size([5, 3])
```

- Note: torch Size is in fact a tuple, so it supports the same operations.

Operations

There are multiple syntaxes for operations..

Addition: syntax 1

```
In [13]: y = torch.rand(5,3)
         print(x + y)

0.8535  0.6359  1.0222
0.6420  0.0091  0.6165
0.9360  0.5439  0.9757
1.1209  1.2988  1.6813
0.4131  1.4377  0.6229
[torch.FloatTensor of size 5x3]
```

Addition: syntax2

```
In [14]: print(torch.add(x,y))

0.8535  0.6359  1.0222
0.6420  0.0091  0.6165
0.9360  0.5439  0.9757
1.1209  1.2988  1.6813
0.4131  1.4377  0.6229
[torch.FloatTensor of size 5x3]
```

Operations

Addition: giving an output tensor

```
In [16]: result = torch.Tensor(5, 3)
         torch.add(x, y, out=result)
         print(result)

0.8535  0.6359  1.0222
0.6420  0.0091  0.6165
0.9360  0.5439  0.9757
1.1209  1.2988  1.6813
0.4131  1.4377  0.6229
[torch.FloatTensor of size 5x3]
```

Addition: in-place

```
In [20]: # adds x to y
         y.add_(x)
         print(y)

1.2916  0.7581  1.2170
1.2765  0.0114  1.0757
1.1908  0.8670  1.4800
1.4199  2.2177  2.4148
0.4318  2.2995  1.0290
[torch.FloatTensor of size 5x3]
```

- Note: Any operation that mutates a tensor in-place is post-fixed with an `_`. For example, `x.copy(y)`, `x.t_()`, will change `x`.

Operations

You can use standard numpy-like indexing with all bells and whistles!

```
In [22]: print(x[:, 1])
```

```
0.1222  
0.0023  
0.3231  
0.9189  
0.8618  
[torch.FloatTensor of size 5]
```

- Read later: 100+ Tensor operations, including transposing, indexing, slicing, mathematical operations, linear algebra, random numbers, etc are described here <http://pytorch.org/docs/torch>

Numpy Bridge

- Converting a torch Tensor to a numpy array and vice versa is a breeze.
- The torch Tensor and numpy array will share their underlying memory locations, and changing one will change the other.

Converting torch Tensor to numpy Array

```
In [23]: a = torch.ones(5)  
print(a)
```

```
1  
1  
1  
1  
1  
[torch.FloatTensor of size 5]
```

```
In [25]: b = a.numpy()  
print(b)
```

```
[ 1.  1.  1.  1.  1.]
```

Numpy Bridge

See how the numpy array changed in value.

```
In [26]: a.add_(1)
          print(a)
          print(b)
```

```
2
2
2
2
2
[torch.FloatTensor of size 5]

[ 2.  2.  2.  2.  2.]
```


Numpy Bridge

Covering numpy Array to torch Tensor

See how changing the np array changed the torch Tensor automatically

```
In [27]: import numpy as np
a = np.ones(5)
b = torch.from_numpy(a)
np.add(a, 1, out=a)
print(a)
print(b)
```

```
[ 2.  2.  2.  2.  2.]
```

```
2
2
2
2
2
```

```
[torch.DoubleTensor of size 5]
```

All the Tensors on the CPU except a Char Tensor support converting to Numpy and back.

Tensors can be moved onto GPU using the `.cuda` function..

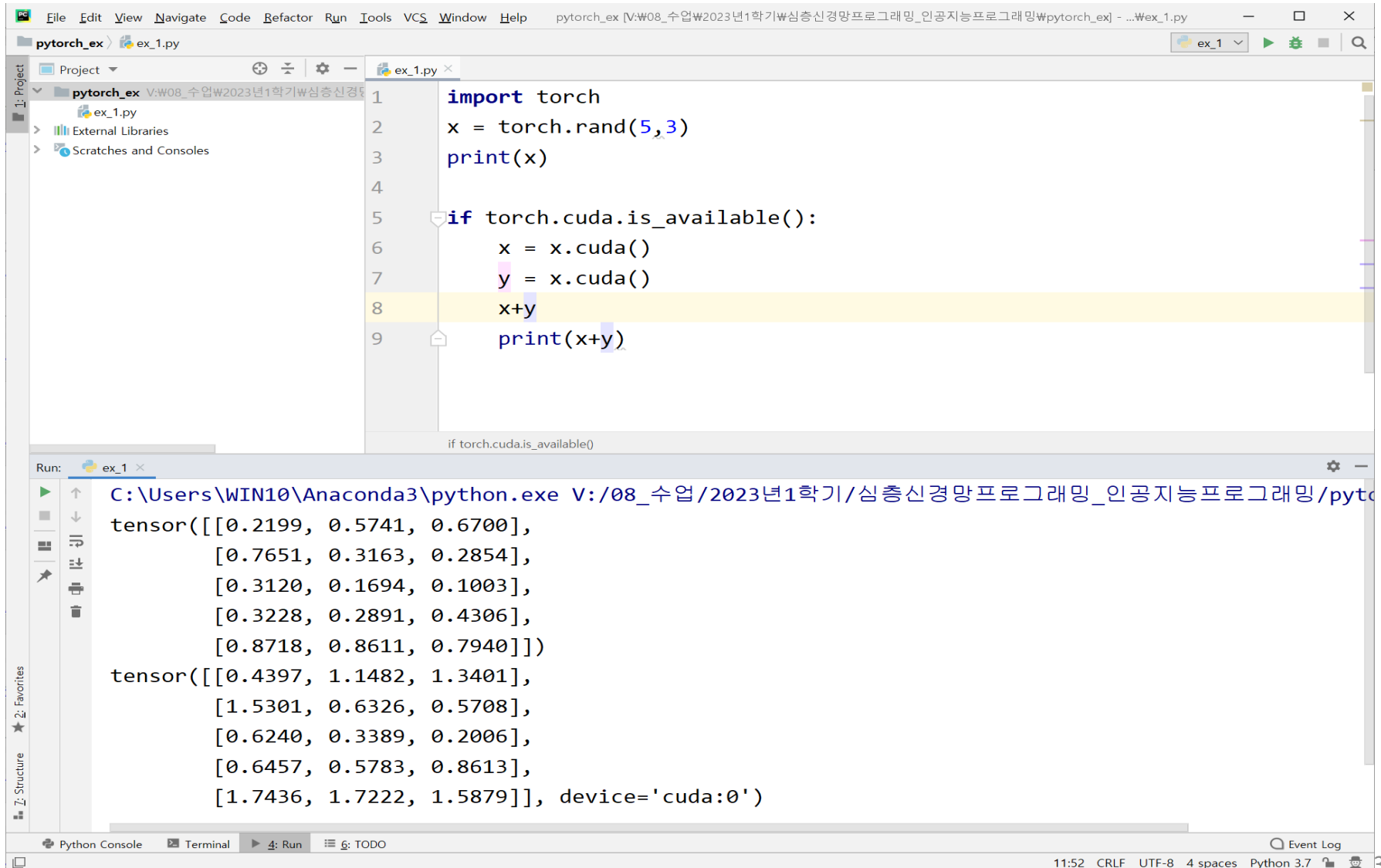
```
In [32]: # let us run this cell only if CUDA is available
if torch.cuda.is_available():
    x = x.cuda()
    y = y.cuda()
    x + y
    print(x + y)
```

```
1.7296  0.8803  1.4118
1.9110  0.0137  1.5350
1.4456  1.1901  1.9842
1.7189  3.1366  3.1483
0.4506  3.1613  1.4352
```

```
[torch.cuda.FloatTensor of size 5x3 (GPU 0)]
```

- References: http://pytorch.org/tutorials/beginner/blitz/tensor_tutorial.html#tensors

CUDA Tensors



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help pytorch_ex [V:\W08_수업W2023년1학기W심층신경망프로그래밍_인공지능프로그래밍Wpytorch_ex] - ...Wex_1.py
```

```
pytorch_ex > ex_1.py
```

```
Project
├── pytorch_ex V:\W08_수업W2023년1학기W심층신경망프로그래밍_인공지능프로그래밍Wpytorch_ex
│   ├── ex_1.py
│   ├── External Libraries
│   └── Scratches and Consoles
```

```
1 import torch
2 x = torch.rand(5,3)
3 print(x)
4
5 if torch.cuda.is_available():
6     x = x.cuda()
7     y = x.cuda()
8     x+y
9     print(x+y)
```

```
Run: ex_1
```

```
C:\Users\WIN10\Anaconda3\python.exe V:/08_수업/2023년1학기/심층신경망프로그래밍_인공지능프로그래밍/pyt
tensor([[0.2199, 0.5741, 0.6700],
        [0.7651, 0.3163, 0.2854],
        [0.3120, 0.1694, 0.1003],
        [0.3228, 0.2891, 0.4306],
        [0.8718, 0.8611, 0.7940]])
tensor([[0.4397, 1.1482, 1.3401],
        [1.5301, 0.6326, 0.5708],
        [0.6240, 0.3389, 0.2006],
        [0.6457, 0.5783, 0.8613],
        [1.7436, 1.7222, 1.5879]], device='cuda:0')
```

```
Python Console Terminal 4: Run 6: TODO
```

11:52 CRLF UTF-8 4 spaces Python 3.7