Basic Python

https://github.com/deeplearningzerotoall/PyTorch

Python

- List

```
list1 = [1, 2, 3]

print(type(list1))
print(list1[0], list1[-1]) #인덱스가 음수일 경우 리스트의 끝에서부터 셈

list1[1] = “str1”
print(list1) #리스트는 자료형이 다른 요소도 저장 가능

list2 = range(10)
print(list2)
print(type(list2)) #range Type출력

list2 = list(list2) #List Type으로 변환
print(type(list2))
print(list2[2:4]) # 인덱스 2에서 4(제외)까지 슬라이싱
print(list2[2:]) # 인덱스 2에서 끝까지 슬라이싱
print(list2[:2]) # 처음부터 인덱스 2(제외)까지 슬라이싱
print(list2[:]) # 전체 리스트 슬라이싱
print(list2[::2]) # 2씩 증가하면서 리스트 출력
print(list2[:-1]) # 슬라이싱 인덱스는 음수도 가능

list2[2:4] = [8, 9] # 슬라이스된 리스트에 새로운 리스트 할당
list2[2:5] = [8, 9] # ??
print(list2)
```

Python

For Loop

```
animals = ['cat', 'dog', 'monkey']
for animal in animals:
    print(animal)

nums = [0, 1, 2, 3, 4]
squares = []
for x in nums:
    squares.append(x ** 2)
print(squares)
```

Python

Function (1)

```
def sign(x):
    if x > 0:
        return 'positive'
    elif x < 0:
        return 'negative'
    else:
        return 'zero'

for x in [-1, 0, 1]:
    print(sign(x))
# 출력 "negative", "zero", "positive", 한 줄에 하나씩 출력.
```

Python

Function (2)

```
def hello(name, loud=False):
    if loud:
        print ('HELLO, %s!' % name.upper())
    else:
        print ('Hello, %s' % name)

hello('Bob') # 출력 "Hello, Bob"
hello('Fred', loud=True)  # 출력 "HELLO, FRED!"
```

Python

- zip

```
for x, y, z in zip([1, 2, 3], [4, 5, 6], [7, 8, 9]):
    print(x, y, z)


for x, y in zip([1, 2, 3], [4, 5, 6], [7, 8, 9]): #???
    print(x, y, z)
```

Python

__call__

```
class HelloWorld():
    def __init__(self):
        pass


helloworld = HelloWorld()

callable(helloworld) # False

# helloworld() #Error


class HelloWorld():
    def __init__(self):
        pass

    def __call__(self):
        print("Hello world")


helloworld = HelloWorld()

helloworld()  # Hello world
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-26-f7a7725d180d> in <module>
      7 helloworld = Helloworld()
      8
----> 9 helloworld()
     10

TypeError: 'Helloworld' object is not callable
```

Python

- __call__

```
class Module():
    def __init__(self):
        #Initialize

    def __call__(self, *input, **kwargs):
        ...
        result = self.forward(*input, **kwargs)
```

Python

- __call__ (Pytorch)

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()         #use parant's __init__ func.
        self.conv1 = nn.Conv2d(1, 6, 3)
        self.conv2 = nn.Conv2d(6, 16, 3)

    def forward(self, x):
        # Max pooling over a (2, 2) window
        x = F.max_pool2d(F.relu(self.conv1(x)), (2, 2))
        # If the size is a square you can only specify a single number
        x = F.max_pool2d(F.relu(self.conv2(x)), 2)
        return x

net = Net()
print(net)

#net(inputData)
```

Numpy

Basic Code (1)

```
import numpy as np

a = np.array([1, 2, 3])  # rank가 1인 배열 생성
print(type(a))            # 출력 "<type 'numpy.ndarray'>"
print(a.shape)            # 출력 "(3,)"
print(a.ndim)             # 출력 "1"
print(a[0], a[1], a[2])   # 출력 "1 2 3"
a[0] = 5                  # 요소를 변경
print(a)                  # 출력 "[5, 2, 3]"

b = np.array([[1,2,3],[4,5,6]])   # rank가 2인 배열 생성
print(b.shape)                    # 출력 "(2, 3)"
print(b.ndim)                     # 출력 "2"
print(b[0, 0], b[0, 1], b[1, 0])  # 출력 "1 2 4"
```

Numpy

Basic Code (2)

```
import numpy as np

a = np.zeros((2,2))  # 모든 값이 0인 배열 생성
print(a)              # 출력 "[[ 0.  0.]
                     #       [ 0.  0.]]"

b = np.ones((1,2))   # 모든 값이 1인 배열 생성
print(b)              # 출력 "[[ 1.  1.]]"

c = np.full((2,2), 7) # 모든 값이 특정 상수인 배열 생성
print(c)               # 출력 "[[ 7.  7.]
                      #       [ 7.  7.]]"

d = np.eye(2)        # 2x2 단위행렬 생성
print(d)              # 출력 "[[ 1.  0.]
                     #       [ 0.  1.]]"

e = np.random.random((2,2)) # 임의의 값으로 채워진 배열 생성
print(e)                     # 임의의 값 출력 "[[ 0.91940167  0.08143941]
                            #                 [ 0.68744134  0.87236687]]"
```

Numpy

Basic Code (3)

```
import numpy as np

# [[ 1  2  3  4]
#  [ 5  6  7  8]
#  [ 9 10 11 12]]
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])

# 슬라이싱을 이용하여 첫 두 행과 1열, 2열로 이루어진 부분배열 생성;
# b는 shape가 (2,2)인 배열이 됨:
# [[2 3]
#  [6 7]]
b = a[:2, 1:3]

# 슬라이싱된 배열은 원본 배열과 같은 데이터를 참조, 즉 슬라이싱된 배열을 수정하면
# 원본 배열 역시 수정됨.
print(a[0, 1])   # 출력 "2"
b[0, 0] = 77    # b[0, 0]은 a[0, 1]과 같은 데이터
print(a[0, 1])   # 출력 "77"
```

Numpy

Basic Code (4)

```
import numpy as np

# [[ 1  2  3  4]
#  [ 5  6  7  8]
#  [ 9 10 11 12]]
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])


b = np.reshape(a, (2, 6))
#b = a.reshape(2, 6) # Equivalent Function

# 슬라이싱된 배열은 원본 배열과 같은 데이터를 참조, 즉 슬라이싱된 배열을 수정하면
# 원본 배열 역시 수정됨.
print(b[0, 0])   # 출력 "2"
b[0, 0] = 77    # b[0, 0]은 a[0, 0]과 같은 데이터
print(a[0, 0])   # 출력 "77"
```

Numpy

Exercise 1

Q1. Create a vector with values ranging from 10 to 49 (ex. 10, 11, ... 49)
Q2. Create a vector with values ranging from 49 to 10 (ex. 49, 48, ... 10)
Q3. Get the positions where elements of 'a' and 'b' match? (Hint: np.where)
Input)
a = np.array([1,2,3,2,3,4,3,4,5,6])
b = np.array([7,2,10,2,7,4,9,4,9,8])
Output) => array([1, 3, 5, 7])

Q4. Add 1 to matrix ‘a’ (Hint: Broadcast)
Input: a = np.arange(9).reshape(3, 3)
Output:
[[1 2 3] [4 5 6] [7 8 9]]

Q5. Swap rows 1 and 2 in the array 'a’ at Q4

```
[[3 4 5]
 [0 1 2]
 [6 7 8]]
```

Q6. Remove Negative values in the array ‘b’ (Hint: np.where)
Input: a = np.array((3, -2, 2, -1, 3))
Output: [3, 2, 3]

Numpy

Exercise 1

Q1. Create a vector with values ranging from 10 to 49 (ex. 10, 11, ... 49)
Ans) print(np.arange(10, 50))

Q2. Create a vector with values ranging from 49 to 10 (ex. 49, 48, ... 10)
Ans) print(np.arange(10, 50)[::-1])

Q3. Get the positions where elements of 'a' and 'b' match? (Hint: np.where)
Input)
a = np.array([1,2,3,2,3,4,3,4,5,6])
b = np.array([7,2,10,2,7,4,9,4,9,8])

Output)
⇒ array([1, 3, 5, 7])

Ans) print(np.where(a == b))

Numpy

Exercise 1

Q4. Add 1 to matrix ‘a’ (Hint: Broadcast)
Input: a = np.arange(9).reshape(3, 3)
Output:
print(a+1)

Q5. Swap rows 1 and 2 in the array 'a' (HARD)

print(a[[1, 0, 2]])

Q6. Remove Negative values in the array ‘b’ (HARD, Hint: np.where)
Input: a = np.array((3, -2, 2, -1, 3))
Output: [3, 2, 3]

print(a[np.where(a>=0)])

Tensor

Tensor

```
import torch
import numpy as np

t1 = torch.FloatTensor([0, 1, 2, 3, 4, 5, 6])
t2 = torch.tensor(np.arange(7)) #Same

print(t1.shape, t2.shape)

print(t1.dim())
print(t1.shape)
print(t1.size())
print(t1[:2], t1[3:])
```

Numpy <-> Tensor

- **Numpy to Tensor , Tensor to Numpy**

```
import torch
import numpy as np

b = np.arange(7)
print( type(b) , type(t1))

tt = torch.tensor(b)
type(tt)

tt = torch.tensor(b)
t_from =torch.from_numpy(b)
print( type(tt), type(t_from ))
print( tt )
print( t_from )

b[0]= -10
print( tt )
print( t_from )

t_to_np = t_from.numpy()
print( type(t_to_np))
```

Tensor

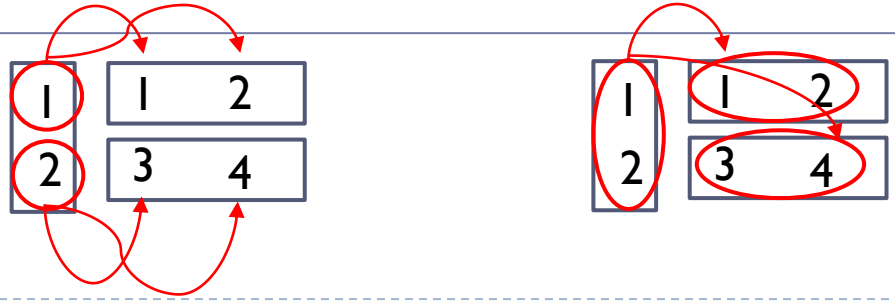
Broadcasting

```
import torch
import numpy as np

m1 = torch.FloatTensor([[3, 3]])
m2 = torch.FloatTensor([[2, 2]])
print(m1 + m2)

# Vector + scalar
m1 = torch.FloatTensor([[1, 2]])
m2 = torch.FloatTensor([3]) # 3 -> [[3, 3]]
#m2 = torch.FloatTensor(3) # 3 -> Error
print(m1 + m2)

# 2 x 1 Vector + 1 x 2 Vector
m1 = torch.FloatTensor([[1, 2]])
m2 = torch.FloatTensor([[3], [4]])
print(m1 + m2)
```

Tensor

- ## Mul vs MatMul

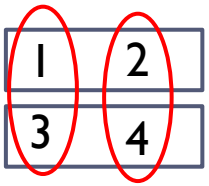
```
print('-------------')
print('Mul vs Matmul')
print('-------------')
m1 = torch.FloatTensor([[1, 2], [3, 4]])
m2 = torch.FloatTensor([[1], [2]])
print('Shape of Matrix 1: ', m1.shape) # 2 x 2
print('Shape of Matrix 2: ', m2.shape) # 2 x 1
print(m1.matmul(m2)) # 2 x 1

m1 = torch.FloatTensor([[1, 2], [3, 4]])
m2 = torch.FloatTensor([[1], [2]])
print('Shape of Matrix 1: ', m1.shape) # 2 x 2
print('Shape of Matrix 2: ', m2.shape) # 2 x 1
print(m1 * m2) # 2 x 2
print(m1.mul(m2))
```



Tensor

Mean

```
import torch
import numpy as np

t = torch.FloatTensor([[1, 2], [3, 4]]) #torch.tensor is not work. Long Tensor...
print(t.type())
print(t)

print(t.mean())
print(t.mean(dim=0)) #Remove Dim 0
print(t.mean(dim=1))
print(t.mean(dim=-1))
```



Tensor

- ## View (Numpy: Reshape)

```
import torch
import numpy as np

t = np.arange(12).reshape(-1, 2, 3)
print(t.shape)

floatT = torch.FloatTensor(t)
print(floatT.shape)

print(floatT.view([-1, 3]))
```

Tensor

Practice

```
import torch
import numpy as np

t = ??
print(t.shape)

floatT = torch.FloatTensor(t)

print(floatT.view([-1, 2, 3]))
```

Goal

```
(24,)

tensor([[13., 16.], [19., 22.]])
```

Tensor

- **Answer**

```
import torch
import numpy as np

t = np.arange(24)
print(t.shape)

floatT = torch.FloatTensor(t)

print(floatT.view([-1, 2, 3])[2:4,:,1])
```

Goal

```
(24,)

tensor([[13., 16.], [19., 22.]])
```

Exercise 2

Q 1. 아래 출력값을 갖도록 a 변수값을 설정 (Hint: np.arange(n) 활용)

```
#a = ??

print("Shape: ", a.shape)
print("Value: ", a)


#Output
Shape:  (2, 3, 4)
Value:  [[[ 0  1  2  3]
  [ 4  5  6  7]
  [ 8  9 10 11]]

 [[12 13 14 15]
  [16 17 18 19]
  [20 21 22 23]]]
```

Exercise 2

- ## Answer 1

```
import numpy as np

a = np.arange(24)
a = np.reshape(a, (2, 3, 4))
```

Exercise 2

Q 2. Exercise 1의 a값과 slicing을 통해, 다음 두 값을 출력

```
#print(a[???]) #Problem1
#print(a[???]) #Problem2

#Output 1
[[ 6  7]
 [10 11]]

#Output 2
[[[ 6  7]
  [10 11]]]
```

Exercise 2

- ## Answer 2

```
1.
print(a[0, 1:3, 2:4])
#print(a[0, 1:3, 2:]) 도 가능

2.
print(a[0:1, 1:3, 2:4])
```

Question and Answer