

NLP with Pytorch Tutorial 0

Python Basics

About this tutorial

- ❖ 11 parts (ideally)
 - Book
 - Additional Material (if necessary)
- ❖ Each time:
 - During the tutorial: learn something new (チュートリアルで: 新しい内容)
 - At home: Do a programming exercise (宿題: プログラミング演習)
- ❖ Working in pair is encouraged (ペアで話し合いをする)

- ❖ If English is difficult for you, you don't need to attend
 - However, I encourage to do the exercises at home if you have time

Plan for Today

❖ Python Basics

Slides and examples borrowed/adapted from:

Kevin Duh: <http://cs.jhu.edu/~kevinduh/index.html>

Graham Neubig: <http://www.phontron.com/index.php?lang=en>

(Both are great NLP researchers and I recommend everybody to read their papers)

What is NLP?

Natural Language Processing (NLP)
studies how computers can **interpret** and
manipulate text

Intro

- ❖ For writing programs for NLP, people often use **Python**

What is python?



- ❖ general-purpose, high-level scripting language
- ❖ Used in a variety of purposes: e.g. networking, web applications
- ❖ Most popular in the scientific community

Pros vs Cons

❖ Pros

Easy to understand (similar to English)

Works across systems (Windows, Mac, Linux)

Object Oriented

Great standard and additional libraries

❖ Cons

Python can be slow

Setting up your python environment

- ❖ If you are on Linux or Mac

Open the “terminal”

- ❖ If you are on Windows

Install cygwin

- ❖ Or use “ssh” to login to the one of the Linux servers

Install Software (if necessary)

- 1) Python: the programming language
- 2) A text editor (vim, emacs, etc.)
- 3) IDE: PyCharm, Sublime, VS Code

Download the Tutorial Files from Github

- ❖ Use the git “clone” command

```
$ git clone https://github.com/lis-kp/tutorial
```

1) Python on terminal

Go to terminal:

Type **python**

Type **print("hello!")**

Type **2 + 2**

```
>>> print("hello!")  
hello!  
>>> 2 + 2  
4
```

2) Using Vim

Open vim:

```
$ vi test.txt
```

Press **“i”** to start input

Write **“test”**

Press **“esc”**

Type **“:wq”** to save and quit (“:w” is save; “:q” is quit)

3) Using an IDE

- ❖ To write longer programs, people often use IDEs

Pycharm, Sublime, VSCode

- ❖ IDEs include features like as code completion and syntax checking

- ❖ If you choose to just use a text editor, you can run your program from terminal by using the command:

```
python <filename.py>
```

Hello World!

- 1) Open hello.py in an editor (e.g. vim)
- 2) Type the following program

```
print("Hello World!")
```

- 3) Run the program

```
lis@fermat:~$ python hello.py  
Hello World!
```

Main data types used

Strings: “hello”, “goodbye”

Integers: -1,0,1,3

Floats: -4.2, 0.0, 3.14

```
my_int = 4
my_float = 2.5
my_string = "hello"

print('string: {:s}\\tfloat: {:f}\\tint: {:d}'.format(my_string, my_float, my_int))
```


Main data types used

Strings: “hello”, “goodbye”

Integers: -1,0,1,3

Floats: -4.2, 0.0, 3.14

```
my_int = 4
my_float = 2.5
my_string = "hello"

print('string: {:s}\tfloat: {:f}\tint: {:d}'.format(my_string, my_float, my_int))
```

```
string: hello    float: 2.500000 int: 4
```

Loop (for) and Control Flow (if/else)

If this condition is true

Then do this

Otherwise

do this

```
my_variable = 5
```

```
if my_variable == 4:
```

```
    print("my_variable is 4")
```

```
else:
```

```
    print("my_variable is NOT 4")
```

```
for i in range(1, my_variable):  
    print("i == {:d}".format(i))
```

Loop (for) and Control Flow (if/else)

If this condition is true

Then do this

Otherwise

do this

For every element in this

Do this

```
my_variable = 5
```

```
if my_variable == 4:
```

```
    print("my_variable is 4")
```

```
else:
```

```
    print("my_variable is NOT 4")
```

```
for i in range(1, my_variable):
```

```
    print("i == {:d}".format(i))
```

Loop (for) and Control Flow (if/else)

If this condition is true

Then do this

Otherwise

do this

For every element in this

Do this

```
my_variable = 5
```

```
if my_variable == 4:
```

```
    print("my_variable is 4")
```

```
else:
```

```
    print("my_variable is NOT 4")
```

```
for i in range(1, my_variable):
```

```
    print("i == {:d}".format(i))
```

```
my_variable is NOT 4
```

```
i == 1
```

```
i == 2
```

```
i == 3
```

```
i == 4
```

Loop (for) and Control Flow (if/else)

If this condition is true

Then do this

Otherwise

do this

For every element in this

Do this

```
my_variable = 5
```

```
if my_variable == 4:
```

```
    print("my_variable is 4")
```

```
else:
```

```
    print("my_variable is NOT 4")
```

```
for i in range(1, my_variable):
```

```
    print("i == {:d}".format(i))
```

Be careful!

Range (1,5) == (1,2,3,4)

```
my_variable is NOT 4
```

```
i == 1
```

```
i == 2
```

```
i == 3
```

```
i == 4
```

Arrays (or “lists” in Python)

```
my_list = [1,2,4,8,16]
```

← Make a list with 5 elements

```
my_list.append(32)
```

← Add one or more elements to the end of the list

```
print(len(my_list))
```

← Print the length of the list

```
print(my_list[3])  
print("")
```

← Print the 4th element

```
for value in my_list:  
    print(value)
```

← Loop through and print every element of the list

Arrays (or “lists” in Python)

Index is an integer,
starting at 0

```
my_list = [1,2,4,8,16]
```

← Make a list with 5 elements

```
my_list.append(32)
```

← Add one or more elements to the end of the list

```
print(len(my_list))
```

← Print the length of the list

```
print(my_list[3])  
print("")
```

← Print the 4th element

```
for value in my_list:  
    print(value)
```

← Loop through and print every element of the list

Index	Value
0	1
1	2
2	4
3	8
4	16

Arrays (or “lists” in Python)

Index is an integer,
starting at 0

```
my_list = [1,2,4,8,16]
```

← Make a list with 5 elements

```
my_list.append(32)
```

← Add one or more elements to the end of the list

```
print(len(my_list))
```

← Print the length of the list

```
print(my_list[3])  
print("")
```

← Print the 4th element

```
for value in my_list:  
    print(value)
```

← Loop through and print every element of the list

Index	Value
0	1
1	2
2	4
3	8
4	16

6
8

1
2
4
8
16
32

Sorting Lists

```
random_list = [3,12,5,6]  
sorted_list = sorted(random_list)  
  
print(sorted_list)
```

← **sorted** function lets you sort a list

Sorting Lists

```
random_list = [3,12,5,6]  
sorted_list = sorted(random_list)  
  
print(sorted_list)
```

← **sorted** function lets you sort a list

```
[3, 5, 6, 12]
```

Maps (or “dictionaries” in Python)



```
my_dict = {"olivia": 22, "maria": 30, "sophia": 18, "emily": 45}
```

```
my_dict["rachel"] = 15
```

 ← Add a new entry

```
print(len(my_dict))
```

 ← Print size

```
print(my_dict["emily"])
```

 ← Print one entry

```
print("")
```

```
if "elizabeth" in my_dict:
```

 ← Check if a key exists

```
print("elizabeth exists in my_dict")
```

```
print("my_dict sorted by key")
```

```
for key, value in sorted(my_dict.items()):
```

 ← Print key/value

```
print('{:s} --> {:d}'.format(key, value))
```


pairs by key order

```
print("")
```

```
print("my_dict sorted by value")
```

```
for key, value in sorted(my_dict.items(), key=lambda x:x[1]):
```

 ← Print key/value

```
print('{:s} --> {:d}'.format(key, value))
```


pairs by value order

Key	Value
olivia	22
maria	30
sophia	18
emily	45

Maps (or “dictionaries” in Python)

```
my_dict = {"olivia": 22, "maria": 30, "sophia": 18, "emily": 45}

my_dict["rachel"] = 15

print(len(my_dict))
print(my_dict["emily"])
print("")

if "elizabeth" in my_dict:
    print("elizabeth exists in my_dict")

print("my_dict sorted by key")
for key, value in sorted(my_dict.items()):
    print('{:s} --> {:d}'.format(key, value))

print("")
print("my_dict sorted by value")
for key, value in sorted(my_dict.items(), key=lambda x:x[1]):
    print('{:s} --> {:d}'.format(key, value))

5
45

my_dict sorted by key
emily --> 45
maria --> 30
olivia --> 22
rachel --> 15
sophia --> 18

my_dict sorted by value
rachel --> 15
sophia --> 18
olivia --> 22
maria --> 30
emily --> 45
```

More on lists and dictionaries

```
my_list = [1,2,"3","four", "four"]
my_set = set(my_list)
my_dict = {"olivia": 22, "maria": 30, "sophia": 18, "emily": 45}
```

Lists and dictionaries do not have a fixed type: can contain anything

```
print(my_list)
print(my_set)
```

sets will remove duplicates: only one copy of "four"

```
print("")
```

```
for key, value in sorted(my_dict.items()):
    print('{:s} --> {:d}'.format(key, value))
```

```
[1, 2, '3', 'four', 'four']
{'3', 1, 2, 'four'}
```

```
emily --> 45
maria --> 30
olivia --> 22
sophia --> 18
```

Splitting and joining strings

In NLP: often split sentences into words

```
sentence = "this is a book"  
words = sentence.split(" ")
```

Split string at white space into an array of words

```
for word in words:  
    print(word)
```

```
print("")  
print(" ||| ".join(words))
```

Combine the array into a single string, separating with " ||| "

Splitting and joining strings

In NLP: often split sentences into words

```
sentence = "this is a book"
words = sentence.split(" ")
```

Split string at whitespace into an array of words

```
for word in words:
    print(word)
```

```
print("")
print(" ||| ".join(words))
```

Combine the array into a single string, separating with " ||| "

```
this
is
a
book
```

```
this ||| is ||| a ||| book
```

More string functions

```
my_string1 = "book"
```

```
print(my_string.upper())
```

 ← Convert string to uppercase

```
my_string2 = "JAPAN"
```

```
print(my_string.lower())
```

 ← Convert string to lowercase

```
print("")
```

```
if my_string1.startswith("b"):
```

 ← Checks if string starts with "b"

```
    print("my_string1 starts with \"b\"")
```

```
else:
```

```
    print("my_string1 does NOT starts with \"b\"")
```

```
if my_string2.endswith("S"):
```

 ← Checks if string ends with "S"

```
    print("my_string2 ends with \"S\"")
```

```
else:
```

```
    print("my_string2 does NOT end with \"S\"")
```

```
print("")
```

```
mystring3 = " desk "
```

```
print(mystring3.strip())
```

 ← Delete the whitespaces

More string functions

```
my_string1 = "book"

print(my_string.upper())

my_string2 = "JAPAN"

print(my_string.lower())

print("")

if my_string1.startswith("b"):
    print("my_string1 starts with \"b\"")
else:
    print("my_string1 does NOT starts with \"b\"")

if my_string2.endswith("S"):
    print("my_string2 ends with \"S\"")
else:
    print("my_string2 does NOT end with \"S\"")

print("")
mystring3 = " desk "
print(mystring3.strip())
```

HELLO
hello

my_string1 starts with "b"
my_string2 does NOT end with "S"

desk

Functions

Functions take an **input**, **transform** the input, and **return** and output

```
def mult_and_abs(x, y):
```

```
    z = x * y
```

```
    if z >= 0:
```

```
        return z
```

```
    else:
```

```
        return z * -1
```

```
print(mult_and_abs(-3, 2))
```

function mult_and_abs takes "x" and "y" as input

Multiplies x and y together and return the absolute value

Call mult_and_abs with x=-3 and y=2

Using command line arguments/ Reading files

```
import sys
my_file= open(sys.argv[1])
```



First argument

```
for line in my_file:
```

Read the file one line at a time

```
    line = line.strip()
```

Delete the symbol “\n”

```
    if len(line) != 0:
        print(line)
```

If the line is not empty, print

```
$ python my_program.py test.txt
```

Testing your code

Write code to test each function

Test several cases

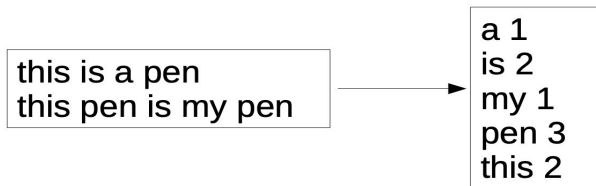
```
def test_mult_and_abs():  
    assert(mult_and_abs(-3,2) == 6), "Error"  
    return "Test passed!"  
  
print(test_mult_and_abs())
```

Test passed!

Practice Exercise

Practice Exercise

Make a program that counts the frequency of words in a file



Test it on test/00-input.txt, test/00-answer.txt

Run the program on the file data/

Report:

The number of unique words

The frequency of each word

Pseudo-code

create a dictionary *counts*

create a map to hold counts

open a file

for each *line* in the file

split *line* into *words*

for *w* in *words*

if *w* exists in *counts*, **add** 1 to *counts*[*w*]

else set *counts*[*w*] = 1

print key, value of *counts*

For next week:

Try to practice the examples

Try to solve the exercise

To learn more:

Python documentation: <https://www.python.org/doc/>