

NLP Programming Tutorial 0 - Programming Basics

Graham Neubig
Nara Institute of Science and Technology (NAIST)

About this Tutorial

- 14 parts, starting from easier topics
- Each time:
 - During the tutorial: Learn something new
 - At home: Do a programming exercise
 - Next week: Present 1 page report of results
- Programming language is your choice
 - Examples will be in Python, so it is recommended
 - I can help with Python, C++, Java, Perl
- Working in pairs is encouraged

Basic Programming

Open a Terminal

- If you are on **Mac or Linux**
 - From the program menu select “terminal”
- If you are on **Windows**
 - Use the windows terminal
 - or install cygwin
 - or use “ssh” to log in to a Linux machine

Hello World!

1) Open my-program.py in an editor (gvim, emacs, gedit)

```
$ gvim my-program.py
```

2) Type in the following program

```
#!/usr/bin/python
print "Hello World!"
```

3) Make the program executable

```
$ chmod 755 my-program.py
```

4) Run the program

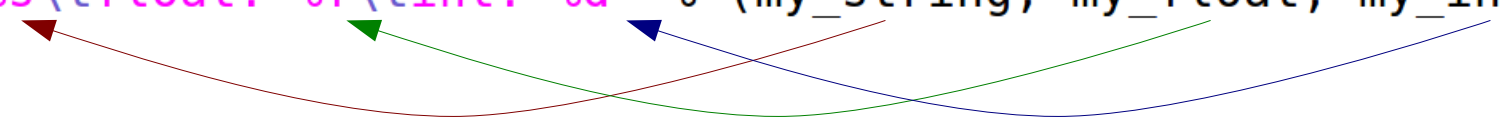
```
$ ./my-program.py
Hello World!
```

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" % (my_string, my_float, my_int)
```



```
$ ./my-program.py
string: hello    float: 2.500000 int: 4
```

if/else, for

```
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" % (i)
```

← if this condition is true
 ← then do this
 ← otherwise
 ← do this
 ← for every element in this
 ← do this

```
$ ./my-program.py
my_variable is not 4
i == 1
i == 2
i == 3
i == 4
```

← Be careful!
 range(1, 5) == (1, 2, 3, 4)⁷

Storing many pieces of data

Dense Storage

Index	Value
0	20
1	94
2	10
3	2
4	0
5	19
6	3

Sparse Storage

Index	Value
49	20
81	94
96	10
104	2

or

Index	Value
apple	20
banana	94
cherry	10
date	2

Arrays (or “lists” in Python)

- Good for dense storage
- 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 more element to the end of the list

```
print len(my_list)
```

Print the length of the list

```
print my_list[3]
```

Print the 4th element

```
print ""
```

```
for value in my_list:
    print value
```

Loop through and print every element of the list⁹

Maps (or “dictionaries” in Python)

- Good for sparse storage:

create pairs of key/value

```
my_dict = {"alan": 22, "bill": 45, "chris": 17, "dan": 27}
```

```
my_dict["eric"] = 33
```

add a new entry

```
print len(my_dict)
```

print size

```
print my_dict["chris"]
```

print one entry

```
if "dan" in my_dict:
```

```
    print "dan exists in my_dict"
```

check whether a key exists

```
for foo, bar in sorted(my_dict.items()):
```

```
    print "%s --> %r" % (foo, bar)
```

print key/value pairs in order

defaultdict

- A useful expansion on dictionary with a default value

```
from collections import defaultdict      import library
```

```
my_dict = defaultdict(lambda: 0)
```

```
my_dict["eric"] = 33
```

```
print my_dict["eric"]
```

```
print my_dict["fred"]
```

default value of zero

print existing key

print non-existent key
(causes error in dict)

Splitting and joining strings

- In NLP: often split sentences into words

```
import string
```

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

Split string at white space
into an array of words

```
for word in words:
    print word
```

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

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

```
$ ./my-program.py
```

```
...
```

```
this ||| is ||| a ||| pen
```

Functions

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

```
def add_and_abs(x, y):
    z = x + y
    if z >= 0:
        return z
    else:
        return z * -1

print add_and_abs(-4, 1)
```

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

add x and y together and return the absolute value

call add_and_abs with x=-4 and y=1

Using command line arguments/ Reading files

```
import sys
my_file = open(sys.argv[1], "r")
for line in my_file:
    line = line.strip()
    if len(line) != 0:
        print line
```

The first command line argument

Open a file for reading

Read the file one line at a time

Remove the line-ending character (“\n”)

If the line is not empty, print it

```
$ ./my-program.py test.txt
```

Testing Your Code

Simple Input/Output Tests

Example:

Program word-count.py should count the words in a file

1) Create a small input file

2) Count the words by hand, write them in an output file

test-word-count-in.txt

a	b	c
b	c	d

test-word-count-out.txt

a	1
b	2
c	2
d	1

3) Run the program

```
$ ./word-count.py test-word-count-in.txt > word-count-out.txt
```

4) Compare the results

```
$ diff test-word-count-out.txt word-count-out.txt
```


Unit Tests

- Write code to test each function
- Test several cases, and print an error if result is wrong
- Return 1 if all tests passed, 0 otherwise

```
def test_add_and_abs():
    if add_and_abs(3, 1) != 4:
        print "add_and_abs(3, 1) != 4 (== %d)" % add_and_abs(3, 1)
        return 0
    if add_and_abs(-4, 1) != 3:
        print "add_and_abs(-4, 1) != 3 (== %d)" % add_and_abs(-4, 1)
        return 0
    return 1
```

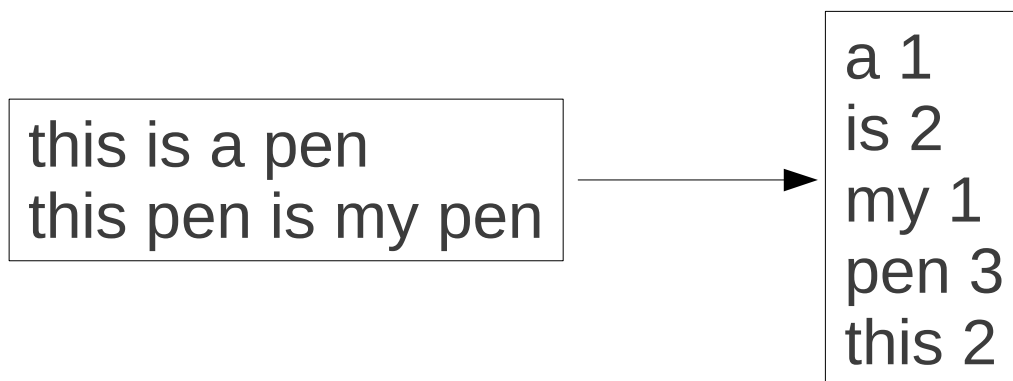
ALWAYS Test your Code

- Creating tests:
 - Makes you think about the problem before writing code
 - Will reduce your debugging time drastically
 - Will make your code easier to understand later

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/wiki-en-train.word
- Report:
 - The number of unique words
 - The frequencies of “in” “on” “with” “to” “the” and “a”

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 *counts*["in"], *counts*["the"] ... etc