CONVENTION TO WRITE PYTHON CODES

Prepared by Nupur Thakur(Jun-Nov 2019)

# STANDARDS TO MAKE THE CODE PATH INDEPENDENT

When you write a python code to be integrated with some other module or to be used by some other person on their system you should make the code as standard as possible so that other person need not to make changes to use your code or to test your code.For the same you can use the path variables which are mostly the same on the machines of different people.

To do so the function to be used is **os.getenv()**

E.g. import os

…

(Some code snippets)

…

temp\_path=os.getenv(“HOME\_anu\_tmp”)+”/tmp/”+”ai1E\_tmp” #Path to the temp Directory of AI Corpus

…

(Some code snippets)

…

# MAKING THE CODE AS MODULAR AS POSSIBLE

Always try to make your code divided into different modules for better understanding of other people and your future self.

E.g.

You want to do some operation on words of entire Corpus then,

1. Write a function to read the corpus and split into lines.
2. Write a function to remove punctuation from the lines.
3. Write a function to split lines into words.
4. Write a main function to do main operations.
5. You can also write functions for the sub operation you are going to use in the main function.

**def** Corpus\_into\_lines(corpus) :

…

**return** sentences

**def** Remove\_punctuations\_from\_sentence(sentence) :

…

**return** wo\_punct\_sentence

**def** Sentence\_to\_words(wo\_punct\_sentence) :

…

**return** words

**def** <name\_of\_the\_main\_operation>() :

...

**return** <final\_value>

# CONVENTION FOR GIVING NAMES TO THE MODULES AND VARIABLES

Try to keep the names of the functions and variables as explanatory as possible.

DO’s

1. Save the files with the name of their functionality e.g. if a shell script is extracting nouns from a Dictionary, then name it as: Extracting\_nouns\_from\_<dict\_name>.sh
2. Give the name to the functions with their functionality. E.g. if a function is splitting sentences into words then name should be Sentence\_to\_words()
3. Name of the variables should be self explanatory. E.g. a variable to be used for English words of the corpus can be named as eng\_words,e\_words,ewords\_from\_corpus etc.
4. Objects for file should also be used relevantly. E.g. file object reading english corpus should be named as e\_corpus\_file etc.

DON'Ts

1. Don’t use i,j etc as iterating variables in for loops.
2. Don’t use file objects as f,f1 etc.
3. Don’t use function names as function1,fun1,func1 etc.
4. Don’t use program names as program.py,shell.sh etc.

# STANDARD INDENTATION IN PYTHON CODE

There are three big Python code formatters :

1. Yapf

2. Autopep8

3. Black.

Black is better than other formatters because of the reasons mentioned in the article linked below.

<https://medium.com/@boxed/a-quick-performance-comparison-of-python-code-formatters-3a89478da8b8>

**To download black: *pip install black***

**To use black: *black <filename.py>*** *e.g. black test.py*

# ALWAYS RETURN THE VALUE FROM THE FUNCTIONS

Whenever you write a module and write its data in a file for further use in the pipeline. Never forget to return its value from the main function.

(**Reason:** There are multiple modules in the pipeline and you never know which one will run first and create it’s file so if some module depends on the file created by your code but have to run before your module, in that case when your module don’t return any value it will cause a deadlock like condition. So to handle that the returned value of your module will be used to generate the file of other module.)

# CONVENTION TO RUN A PYTHON MODULE ON ALL SENTENCES OF THE CORPUS

As of now, I assume you know about the format of temp Directories of the corpus which contains folders for each sentence.

So whenever you run a python code for any temp directory, There are two conventions to choose from :

1. Run the Python code with for loop for all sentences.
2. Create a shell script to run you python code(for a single sentence) to run it for all sentences.

# LOG CONVENTION

Always write a log file for your module.

E.g. The input file you are using if it is not present in some sentences, then write it in a log file.

Log file for the code which takes E\_sentence and H\_sentence as input and sentence 2.11,2.98,2.101,2.112,2.113 etc. doesn’t contain E\_Sentence so output of the code will obviously not get generated so to check why output of the code is not getting generated always mention such things in a log file.

***POINTS TO REMEMBER WHILE CREATING A LOG FILE***

1. Create a log file in the main temp directory of the corpus. Until or unless there is no specific need for the log creation in the sentence folder.
2. Always create a log file in the write mode so that it overwrites every time it get called for entire corpus. (**Don’t overwrite the file for every sentence otherwise it will contain the logs of last sentence only., Just remember log file should be refreshed every time you run the module)**

# CREATING VIRTUAL ENVIRONMENT FOR PYTHON