

CIS 530 Fall 2009 General Information

Mitch Marcus

September 25, 2009

1 Working on Assignments

How to contact us

The instructor for this class is Mitch Marcus (`firstname@cis.upenn.edu`, Levine 503). The TA is Constantine Lignos (`lastname@seas.upenn.edu`, IRCS 410). The course website is <http://www.cis.upenn.edu/~cis530>. Office hours will be announced soon. Rather than e-mailing us individually, it's best to send any mail to `cis530@seas.upenn.edu` and that way both the instructor and TA will be able to respond. Feel free to email for technical support or if you have any questions about the homework.

Policy on working in groups

This is an introduction to NLP, not a programming course, and we want you to be able to concentrate on NLP issues. Thus you are encouraged to help each other debug your programs and tackle the intricacies of Python, NLTK, and any other programming languages you might use. Apart from this specific case, we expect each of you to submit original work. The intellectual effort (designing and writing programs, working on essay questions, etc.) has to be done individually. Submitting homeworks as a group is not possible in this course. If you feel this policy is not optimal or not clear, please come and talk to us.

2 Using Python and NLTK

In this course we will use Python version 2.6¹ and NLTK version 0.9.9. They are already installed on Eniac, but if you want you can install them on your own machine using the instructions at <http://www.python.org/download/> and <http://www.nltk.org/download>. If you are installing NLTK, we recommend installing Numpy and Matplotlib as in the installation instructions but skipping installing Prover9.

Because NLTK provides so many tools for performing NLP tasks, you will typically be writing very short programs that use NLTK's modules to do most of the work. To make NLTK and its data accessible on Eniac, add the following lines to your `.bashrc` file (or the equivalent lines to your shell of choice), creating the file in your home directory if it doesn't exist):

```
export PYTHONPATH=/home1/c/cis530/Software/python2.6/site-packages
export NLTK_DATA=/home1/c/cis530/data
```

Once you've added these environment variables, log out and log in and you should be able to do the following to verify that things are working:

```
% python
Python 2.6 (r26:66714, Feb  3 2009, 20:49:49)
[GCC 4.3.2 [gcc-4_3-branch revision 141291]] on linux2
Type "help", "copyright", "credits" or "license" for more information.
```

¹If you are using your own installation, there is no problem using Python 2.4 or 2.5, but 3.0 will not work with NLTK.

```
>>> import nltk
>>> nltk.corpus.brown.words()[0]
'The'
```

This prints out the first word of the Brown corpus. If you encounter problems, e-mail the TA immediately.

3 Coding Standards

Style

We ask that you make your code consistent with the PEP 8 style guide at <http://python.org/dev/peps/pep-0008/>. We will not require that your code precisely follow the “Code lay-out” guidelines, but we do expect you to follow the “Naming Conventions” and “Programming Recommendations” because doing so makes your code much more readable and often prevents you from creating some types of difficult-to-identify bugs. If neglecting style conventions results in your code being unreadable, you will be penalized.

Documentation

Your code should contain clear, concise comments that explain what is going on at a higher level than the code itself expresses. For example, ‘‘Increment x each iteration’’ is not a particularly helpful comment; ‘‘Increment the word counter for each word’’ is much more insightful. Every function, no matter how trivial, should have at least a one-line docstring. Code that is missing docstrings or is poorly commented will be penalized.

Quality

You should focus on making your code match the specification of the assignment but need not concern yourself with handling error conditions or malformed input. Again, the purpose of the course is not to test your programming skills. While you are welcome to put in assertions for your own debugging purposes, your code will never be tested against empty or bad input or under stress conditions.

Organization

Unless otherwise noted, all code for an assignment should be turned in as a single file. All code should be within functions, with no code “loose” outside of functions other than import statements and testing code as described below.

While it will not impact your grade, you should feel free to include the test code you used to verify your work. If you have any test code in your file, it **must be inside a “main” block** like so:

```
# Test code
if __name__ == "__main__":
    # Test code goes here
    assert sum(1, 1) == 2
```

This allows you to run your tests by calling Python on the file, but allows us to skip your tests when we load your file as a module.