# Project 2 Rubrics

*Due: Nov. 15, 2011, 2:25 pm*

## What to Hand in

You should hand in a zip file of your project folder which must contain

- a `README.txt` file that **briefly** describes the project
- a `link.txt` that contains the URL of your website for the project, which is a detailed documentation of your work
- a folder named `assets` that contains all your python and Excel files and all your data files

Click here if you do not know how to zip a folder

## Coding Style

- **Choose meaningful function names**. Even though you can name your functions whatever you like, but it is impossible for others to understand you program if you name a function that encrypt a message `pizza`.
- **Choose meaningful variable names**. Similarly, your variable names should indicate to some extent what they mean. Using `n` for number is OK, but you want to say

  ```
  for book in bookList:
  ```

  instead of

  ```
  for n in bookList:
  ```

- **Document your functions**. This envolves two things: using triple quotes to provide a 'tooltip' for your function and using comments to tell people what are your functions' arguments and return values, and what it does (it may have some overlap with the tooltip. An example is as below:

```
1 def add(a, b):
2    ''' adds a and b and returns the sum '''
3    # input: number, number (a pair of numbers that we'll
         add)
4    # output: number (their sum)
5    # purpose: compute the sum of two numbers and return it
6    return a + b
```

A more interesting example is

```
1
2 def averageVocabSize(booknamelist)
3    ''' Compute the average vocabulary size for the books
         in the list'''
4    # input: a list of strings (each one the name of a
         file containing a book's text)
5    # output: number          (the average of the vocab
         sizes of those books)
6    # purpose: take a list of booknames like ['pride.txt',
         'moby.txt'] and foreach bookname, find the
         vocabulary size,
7    # and then compute and return the average of these
         sizes (i.e., their sum divided by the number of
         booknames).
8
9    # some actual code here ...
```

- **Comment your code**. Whenever you are doing something tricky or not immediately obvious, you should use comment to exlain it. Even if everything is clear, you should also use comment to explain the structure of your code to make it easier to read. For example, comment a for-loop and say what it does.
- **Always close a file whenever you open one**. Having an opened file dangling there is a bad habit.

## Website

- **Home page** should concisely present the problem(s) being solved, or calculation being carried out.

  If they created the XKCD "Fucking awesome/awesome as shit" graph, the first page could say "I used automated internet searches and some python code and some Excel to produce the graph below [show pic]; this website describes the process used"

  For more question-based projects, a clear statement of the question and the computational approach used should appear on the front page, as in "Do young writers have larger vocabularies? Using 22 texts from Project Gutenberg, I analyzed this question and concluded they did not; they in fact had SMALLER vocabularies than older authors. For each text, I determined (by hand, using Wikipedia) the author's age at publication, and then used a variation of the CS931 work (see ¡link here¿) on concordances to determine the vocabulary size, and computed a line-of-best-fit between age and vocabulary size, shown below. Here's ¡LINK¿ a description of the analysis process; here's ¡link¿ a discussion of the results and their significance..."

- There should be an **Approach** page that discuss in detail how you attacked the problem or how the calculation is carried out.

  For analysis projects, discussion should be coherent, well-organized, well-supported. For projects like the XKCD graph project, the discussion should be about the "process", as in "one difficulty was finding a good list of adjectives to use, because ..." and "I tried the program a second time, using "damned X" and "X as hell", and found similar results – for SOME adjectives there was a large difference in frequencies between the two uses, for others almost no difference. From the limited data I got, I looks as if the large-difference adjectives are more likely to be negative (ugly, fat, stupid), while the small-difference ones were neutral or positive (tiny, huge, awesome, ...)"

- There should also be a **Data** page that describes the data source and any by-hand work done on the data, like "I opened each file in Notepad and removed the Project Gutenburg boilerplate at the start and finish, and then saved the file" and "I named each file with a short

mnemonic like "pride.txt" for "Pride and Prejudice", rather than using Gutenberg's name, which was something like "11030JX.txt", which was hard to read/remember.

### Handin

Email your zip file to `cs0931tas@cs.brown.edu` and title the file `'YOURNAME'proj2.zip` — for example, `DylanFieldProj2.zip`.