# Code Management

by Laura Miller, UNC Math Dept.

## *1) Overview*

Take a look at Wilson et al. (2012) *http://arxiv.org/abs/1210.0530* for the big picture. Abstract: "Scientists spend an increasing amount of time building and using software. However, most scientists are never taught how to do this efficiently. As a result, many are unaware of tools and practices that would allow them to write more reliable and maintainable code with less effort. We describe a set of best practices for scientific software development that have solid foundations in research and experience, and that improve scientists' productivity and the reliability of their software."

## *2) How to Properly Comment Your Code*

6.189 A Gentle Introduction to Programming, January IAP, 2011. Retrieved from *http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-189-a-gentle-introduction-to-programming-using-python-january-iap-2011/*
MIT Open Courseware: *http://ocw.mit.edu/terms/*

Scientists writing software need to write code that both executes correctly and can be easily read and understood by other programmers (especially the author's future self). If software cannot be easily read and understood it is much more difficult to know that it is actually doing what it is intended to do. To be productive, software developers must therefore take several aspects of human cognition into account: in particular, that human working memory is limited, human pattern matching abilities are finely tuned, and human attention span is short.

Compare the differences between the uncommented code below and commented code on the next page.

**Uncommented Code**

```
city=raw_input("Enter a city: ")
while city[-1]==" ":
    city = city[:-1]
temp=raw_input("Enter a temperature in Farenheit: ")
temp = float(temp)
temp = (temp - 32.0)*(100.0/180.0)
temp = round(temp,3)
temp = str(temp)
print "In "+city+" it is "+temp+" degrees Celcius!"
```

There are many problems with this code. A quick scan reveals no info as to who wrote it. What is the name of the file? What does it do? At a glance, we are lost.

**Commented Code**

```
#Alyssa P. Hacker
#fah_to_celsius.py

#collect a city name from user
city=raw_input("Enter a city: ")

#trunacte whitespace
while city[-1]==" ":
    city = city[:-1]

#collect a temp from user
temp=raw_input("Enter a temperature in Farenheit: ")

#convert string to float
temp = float(temp)

#convert Farenheit temp to Celsius temp
temp = (temp - 32.0)*(100.0/180.0)

#truncate to 3 decimal places
temp = round(temp,3)

#recast as string so we can concatenate
temp = str(temp)

#print result!
print "In "+city+" it is "+temp+" degrees Celcius!"
```

Here we clearly see the author's name and the name of her file. Further she has fully commented what the lines of her program do. Your code should include comments that explain what you're doing, and if you're doing something tricky or unique be sure to explain that, as well. A good goal is to have 1 comment for every 1-4 lines of code. Be sure to document not only what your code does, but, as you begin writing more advanced code, also what was intentionally left out, optimized away, tried and discarded, etc. – basically, any design decision you make.

Some additional advice: *http://net.tutsplus.com/tutorials/html-css-techniques/top-15-best-practices-for-writing-super-readable-code/*

### *3) Keeping a Computational Notebook or Diary*

Microsoft OneNote is an increasingly popular solution for information management and collaboration, and it has garnered a large population of mostly PC-centric users since its first release in 2003. Of OneNote's competitors, the most prominent is Evernote, which addresses some of the same requirements, but from a different starting point (more Web-centric than PC-centric). In addition to general note taking, these software packages offer a great solution to diary your scientific computing efforts. For example, you can include screenshots of your simulation results with links to that version of the code.

Unfortunately, neither of these software packages is available for Linux. A viable alternative that is available on all platforms is Tomboy notes. There are many other note taking programs available for Linux, some of which are reviewed in the links below.

Comparisons: *http://www.quepublishing.com/articles/article.aspx?p=1684320*
*http://en.wikipedia.org/wiki/Comparison_of_notetaking_software*

Further Details:

*Microsoft OneNote* is a computer program for free-form information gathering and multi-user collaboration. It gathers users' notes (handwritten or typed), drawings, screen clippings, and audio commentaries, and shares them with other OneNote users over the Internet. OneNote is available as an application for Windows, iOS, Android, Windows Phone, and Symbian. Notes can also be edited from a web browser.

*Evernote* is a suite of software and services designed for notetaking and archiving. A "note" can be a piece of formatted text, a full webpage or webpage excerpt, a photograph, a voice memo, or a handwritten "ink" note. Notes can also have file attachments. Notes can be sorted into folders, then tagged, annotated, edited, given comments, searched and exported as part of a notebook. Evernote supports a number of operating system platforms (including OS X, iOS, Chrome OS, Android, Microsoft Windows, Windows Phone, BlackBerry, and webOS) and also offers online synchronisation and backup services. *http://evernote.com/*

*Tomboy* is a desktop note-taking application for Linux, Unix, Windows, and Mac OS X. Simple and easy to use, but with potential to help you organize the ideas and information you deal with every day. Tomboy is written in C# and utilizes the Mono runtime and Gtk#. Automatic spell-checking is provided by GtkSpell. *http://projects.gnome.org/tomboy/*

*Tomboy Tutorial:*
When you first open Tomboy, the main window and a "Start Here" note will pop up. That note is basically a quick tutorial to get you started with the basic functions. The main window consists of three sections: the top hosts the Search bar, the left pane holds your Notebooks, and the main section will display all the notes in the selected notebook. You will already have two notes created so you'll get a general feel of the interface right from the beginning. The notification icon in GNOME's panel also has some nice functionalities as left-clicking on it will reveal the most recent notes (older notes can be "pinned" to the menu) and options for creating new ones. If you are interested in learning more about Tomboy, please read through the rest of the tutorial here: http://www.softpedia.com/reviews/linux/Tomboy-Review-109670.shtml

You may also be interested in a youtube tutorial:
*http://www.youtube.com/watch?v=xCstHOCVExA*

### *4) Software versioning systems*

Apache Subversion (often abbreviated SVN, after the command name svn) is a software versioning and revision control system distributed under an open source license. Developers use

Subversion to maintain current and historical versions of files such as source code, web pages, and documentation. Its goal is to be a mostly compatible successor to the widely used Concurrent Versions System (CVS). *http://subversion.apache.org/*

A general and basic introductory tutorial to svn is available here:
*http://www.thegeekstuff.com/2011/04/svn-command-examples/*

Mercurial is a cross-platform, distributed revision control tool for software developers. It is mainly implemented using the Python programming language, but includes a binary diff implementation written in C. It is supported on Windows and Unix-like systems, such as FreeBSD, Mac OS X and Linux. Mercurial is primarily a command line program but graphical user interface extensions are available. All of Mercurial's operations are invoked as arguments to its driver program hg, a reference to the chemical symbol of the element mercury.
*http://mercurial.selenic.com/*

A general tutorial for mercurial is available here: *http://mercurial.selenic.com/wiki/Tutorial*

Git is a distributed revision control and source code management (SCM) system with an emphasis on speed. Initially designed and developed by Linus Torvalds for Linux kernel development, Git has since been adopted by many other projects. *http://git-scm.com/*
There are several good online tutorials that can be used to get one started with get. There are also several online books available that go into more advanced details. Please see the links below for these additional resources:
*http://www.ralfebert.de/blog/tools/git_screencast/*
*http://net.tutsplus.com/tutorials/other/easy-version-control-with-git/*
*http://ftp.newartisans.com/pub/git.from.bottom.up.pdf*

### *5) Development environments for open source projects*

Project Hosting on Google Code provides a free collaborative development environment for open source projects. Each project comes with its own member controls, Subversion/Mercurial/Git repository, issue tracker, wiki pages, and downloads section. This project hosting service is simple, fast, reliable, and scalable, so that you can focus on your own open source development. *https://code.google.com/p/support/wiki/GettingStarted*

There are also numerous online tutorials that may be useful as you get started:

General tutorial for googlecode (and everything you need beforehand):
*http://engtech.wordpress.com/2007/03/03/howto_google_code_hosting_subversion_tortoisesvn/*

Tutorial on how to create a Google Code project and synchronize files with SVN repository using Subclipse:
*http://www.gdgankara.org/2011/06/04/how-to-create-google-code-project-and-synchronize-files-with-svn-repository-using-subclipse/*

Actual working example of Boyce Griffith's Immersed Boundary Method with Adaptive Mesh Refinement (IBAMR):
*https://code.google.com/p/ibamr/*