
HFOSS

Release

Ralph Bean

Sep 27, 2017

Contents

1	Syllabus	1
1.1	Projects Seminar in FLOSS Game Development	1
1.2	Text Books	1
1.3	What You'll Do	2
1.4	The spirit of the course	2
1.5	Licensing	2
1.6	Schedule	3
1.7	Grading	3
1.8	Lightning Talks - Extra Credit	4
2	Text Books	5
3	How to NOT make Arguments	7
4	Notes for Class Sessions	9
4.1	Week 02, Day 1: First Flight	9
4.2	Week 02, Day 2: Introduction to Python	9
4.3	Week 03, Day 1: Intermediate Python	10
4.4	Week 03, Day 2: So-called "Advanced" Python	11
4.5	Week 04, Day 1: OLPCs!!!	11
4.6	Week 04, Day 2: Sugar	12
4.7	Week 05, Day 1: Projects	12
5	Helpful Hints – A list of external resources	13
5.1	git	13
5.2	vim	13
6	README.rst – Humanitarian Free/Open Source Software Course @ RIT	15
6.1	Setting up your environment	15
6.2	Building the "Documentation"	16
6.3	Validating the <code>data/students.yaml</code> file	16
7	Homework - First Flight	19
7.1	IRC	19
7.2	Mailman	20
7.3	Blogging	20
7.4	github	20

7.5	Patch the Course Project	21
8	Homework - Bugfix	23
8.1	Find a bug	23
8.2	Use the Source, Luke	24
8.3	The Deliverable	24
8.4	An Afterthought (not required)	24

Projects Seminar in FLOSS Game Development

- Syllabus - <http://hfoss.rtfld.org/> – (subject to change)
- Course Number - 4085.582.01
- Room - Bldg 87, Room 1600, RIT MAGIC Center
- Monday, Wednesday – 5:00pm-6:15pm
- Instructor - Remy DeCausemaker <remydcsi@rit.edu>
 - Office: The MAGIC Center
 - Office Hours: Monday, Wednesday, 4:00-5:00pm
- Teacher's Assistant - TBA <HFOSSTA@gmail.com>. HFOSSTA in IRC.
- IRC - <irc.freenode.net>, #rit-foss
- Email list - floss-seminar@lists.rit.edu
- Blog Planet - <http://foss.rit.edu/planet>
- The source for this syllabus can be found at <http://github.com/FOSSRIT/hfoss>

Text Books

There are a number of textbooks we'll be referencing throughout the quarter. You can find these books at <http://hfoss.rtfld.org/books>

What You'll Do

This course will introduce students to the Free and Open Source Software (FOSS) and Open Content movements, to the open source development process, and to the open questions of the efficacy of technology in the classroom.

Students will learn FOSS process and Tools with class projects that support the One Laptop Per Child community by creating content and software for free distribution to students and teachers around the world. The OLPC project is driven by a world-wide community.

For this course students will be expected to attend and make final presentations to the RIT and Rochester FOSS communities via the irregular Rochester Pythonistas meet-ups and FOSSBox hack-a-thons when possible. Students will also become members of the Sugar and OLPC international communities. Local FOSS community members may join us in class sessions as well. Treat them as you would another instructor, but they're also your peers in moving this innovative project forward.

The spirit of the course

While still a course where you will receive a letter grade, the spirit of the course is intended to be both *open* and *fun*.

An *open* course – students will have access to the ‘document source’ for the syllabus. While you are reading *the syllabus* right now, as a student of the class you have a right to [fork the upstream repository](#), make modifications, and submit patches for review. Barring a troll festival, this can create a fun, dynamic environment in which the course curriculum can develop by the very same mechanism being taught during the quarter (community-driven).

Licensing

All code developed by students in the course must be licensed (by the student) under any one of the [licenses approved by the open source initiative](#).

Your code that you write is your code, with which you can do what you will; true. However, if you're unwilling to license code you write for an open source course with an open source license, you're in the wrong course.

Schedule

Week	Day	Topic	Assigned	Due
1	1	Meet online. Introductions		
	2	Doh!		
2	1	Go over the syllabus. Discuss open-advice and PyCon videos. Introduction to git.	<i>Homework - First Flight</i>	
	2	Lightning Talks. Introduction to Python		
3	1	Intermediate Python	<i>Homework - Bugfix</i>	<i>Homework - First Flight</i>
	2	Lightning Talks. "Advanced" Python		
4	1	Git Seminar. OLPC Distribution. OLPC Smoke Test.	hw/stest	<i>Homework - Bugfix</i>
	2	Lightning Talks. Introduction to Sugar		hw/stest
5	1	Project Choices and Teams http://bit.ly/AeDmaK	fnl/project	
	2	Lightning Talks. In class development.		
6	1	User Testing		
	2	Lightning Talks. In class development.		
7	1	User Testing		
	2	Lightning Talks. In class development.		
8	1	User Testing		
	2	Lightning Talks. In class development.		
9	1	User Testing		
	2	Lightning Talks. Crunch Time.	fnl/present	
10	1	Crunch Time.		
	2	Final Presentations	fnl/assmnt	fnl/present fnl/project
11	?	Return the OLPCs		fnl/assmnt

Grading

Assignments are due at midnight of the day they are marked as due.

Late submissions will be deducted 10% per day they are late.

Your final grade for the quarter will be derived from the following weights.

Component	Weight
In-Class Participation	15%
FLOSS Dev Practices (Blogging, patching, writing, IRC)	25%
Team Peer Assessment	20%
Completed Project	20%
Final Presentation	20%

Blog updates – students are required to keep a blog to which they post updates about their investigations, progress, success, and pitfalls. This blog can be hosted anywhere, but must be added to the course [planet](#) (there are instructions on how to do this in *Homework - First Flight*).

- You must make at least one blog post per week to receive full credit.
- You must participate regularly in the course’s IRC channel: asking and answering questions.

- You must participate in the course's mailman list, floss-seminar@lists.rit.edu.
- Contributions to the course curriculum, syllabus, and rubric are factored in here as well.

Blogging is good for you and good for the [FLOSS community at large](#).

The details for the final can be found at final.

Lightning Talks - Extra Credit

Every Wednesday for the first portion of class, any student has the opportunity to give a [lightning talk](#) on a topic of their choosing. Your lightning talk must be less than 5 minutes in length and must be at least remotely related to the course material.

You will receive +1 extra credit points towards your final grade for every lightning talk you give. Only the first three lightning talks offered will be allowed during a given class. Talks will be chosen from among those offered by students on a FIFO basis.

CHAPTER 2

Text Books

- 10 RULES FOR RADICALS by CARL MALAMUD
- CODING FREEDOM: THE ETHICS AND AESTHETICS OF HACKING by E. GABRIELLA COLEMAN
- RAPTURE OF THE NERDS by CORY DOCTOROW and CHARLES STROSS
- HACKING POLITICS edited by DAVID MOON, PATRICK RUFFINI, AND DAVID SEGAL
- OPEN ADVICE: WHAT WE WISHED WE HAD KNOWN WHEN WE HAD STARTED edited by LYDIA PINTSCHER
- OPEN GOVERNMENT edited by DANIEL LATHROP and LAUREL RUMA
- AN OPEN WORLD compiled by the contributing authors of <http://opensource.com>
- OPEN VOICES compiled by the contributing authors of <http://opensource.com>
- OPEN MISED CEOs compiled by the contributing authors of <http://opensource.com>
- HOW TO THINK LIKE A COMPUTER SCIENTIST: LEARNING WITH PYTHON by Allen Downey Jeffrey Elkner Chris Meyer
- THINK PYTHON: HOW TO THINK LIKE A COMPUTER SCIENTIST by ALLEN DOWNEY
- HOW TO TELL IF YOUR FLOSS PROJECT IS DOOMED TO FAIL by TOM 'SPOT' CALLAWAY
- THE SUCCESS OF OPEN SOURCE by STEVE WEBER

CHAPTER 3

How to NOT make Arguments

Rhetological Fallacies

Errors and manipulations of rhetoric and logical thinking

Appeal to the Mind



Appeal to Anonymous Authority

Using evidence from an unnamed 'expert' or 'study' or generalized group (like 'scientists') to claim something is true.

"They say that it takes 7 years to digest chewing gum."



Appeal to Authority

Claiming something is true because an unqualified or untrustworthy 'expert' says it is.

"Over 400 prominent scientists and engineers dispute global warming."



Appeal to Common Practice

Claiming something is true because it's commonly practiced.

"This bank has some problems with corruption. But there's nothing going on here that doesn't go on in all the other banks."



Appeal to Ignorance

A claim is true simply because it has not been proven false (or false because it has not been proven true).

"Nobody has proved to me there is a God. So there is no God."

Appeal to Emotions



Appeal to Consequences of a Belief

Arguing a belief is false because it implies something you rather not believe.

"That can't be the Senator on that sextape. If it were, he'd be about not knowing her. And he's not the kind of man who..."



Appeal to Fear

An argument is made by increasing fear and pressure towards the opposing side

"Before you know it there will be more mosques than..."



Appeal to Flattery

Using an irrelevant compliment to slip in an unfair claim which is accepted along with the compliment

"Intelligent and sophisticated readers will of course accept my fallacy like this when they read one."



Appeal to Nature

Making your claim seem more true by drawing an analogy with the "good" natural world.

"Of course homosexuality is unnatural. You don't see animals copulating."

Appeal to Incredulity



Appeal to Pity

Week 02, Day 1: First Flight

- Introductions
- Covering the Syllabus
- Discussing Open-Advice on Community Building
- Discussing PyCon Videos
- Introduction to `git` by ryansb from <http://ryansb.com/seminars/git>
- *Homework - First Flight*

Week 02, Day 2: Introduction to Python

- Lightning Talks!!!
- More `git` with ryansb and <http://ryansb.com/seminars/git>
- Introduction to Python (check out <http://learnpythonthehardway.org/book/>)
 - basic operators
 - strings
 - formatting
 - multiplying strings
 - lists
 - dicts
 - conditionals
 - boolean trickery, not and in

- any and all
- while loops
- continue
- break
- for loops
- functions
- args and kwargs
- whitespace

Week 03, Day 1: Intermediate Python

- *Homework - First Flight* is due. How'd it go?
- Other business
- Intermediate Python
 - stdlib
 - * argparse
 - * urllib2
 - * itertools
 - virtualenv
 - setup.py
 - sweet modules on pypi
 - * shelve
 - * fabulous
 - * nose
 - * sqlalchemy
 - zip
 - map
 - filter
 - list comprehensions (!)
 - generators
 - decorators
 - classes
 - dunder methods (reference, <http://www.siafoo.net/article/57>)
 - context managers
 - multiple inheritance

Week 03, Day 2: So-called “Advanced” Python

- Announcements
 - The [planet](#) is up. Subscribe to it with your RSS reader.
 - Homework 2 is due on Monday. Good to go?
 - Special guest [Luke Macken](#)!
- Lightning Talks!!!
- Advanced Python
 - context managers revisited
 - * <https://github.com/ralphbean/pyrasite/commit/cdca3dfc4b757249d50fcc2ab6fc7de6d40dc0f5>
 - locals() and globals()
 - the inspect module
 - * docstrings and ‘help’
 - * inspect.stack
 - * inspect.getsource
 - the abstract syntax tree
 - * desmaj’s tool
 - * macchiato
 - synthesizing stuff
 - * getattr and __getattr__, two sides of the coin
 - metaclasses

Week 04, Day 1: OLPCs!!!

- Homework 2 is due. How did it go?
- There is a TA for the class; Nate Case. [qalthos](#) in IRC or [qalthos ~@~ gmail.com](#)
- OLPC Distribution
 - I need your DCE name.
 - These must be returned at the end of the quarter under penalty of death.
- OLPC Smoke Test
 - http://wiki.laptop.org/go/Smoke_test/10.1.x/1_hour_smoke_test
 - The one exception to the smoke test is that you need to use `connect-rit` in the terminal activity to connect to `ritwpa2`. Just open the terminal activity and run `./connect-rit`. Each XO should have it.
 - **Oh no!** the `connect-rit` script is busted!
 - * If this is the case for you, you can get a new copy of it from <https://github.com/Qalthos/connect-rit> – Use a USB key to transfer it to the OLPC.

Week 04, Day 2: Sugar

- Lightning Talks
- Announcements
 - Project pitches are due on Monday
- Introduction to Sugar
 - Reading you can do later if you want more detail:
 - * <http://en.flossmanuals.net/make-your-own-sugar-activities>
 - * http://wiki.laptop.org/go/Understanding_Sugar_code
 - Sugar concepts
 - * Journal
 - * Different Views
 - * Sandboxing, not signing.
 - You can bust out `/usr/bin/sugar-session` (<http://git.sugarlabs.org/sugar/mainline/blobs/master/bin/sugar-session> or <http://gist.github.com/2297065> for a syntax-highlighted version)
 - * `ps -ef | grep sugar`
 - * `sudo yum -y install vim`
 - * `vim /usr/bin/sugar-session` and you'll see:
 - A lot of from `sugar import env, logger`
 - And some from `jarabe import model, view, keyhandler`
 - Check out <http://git.sugarlabs.org> for the big *tamale*.
 - What is an activity?
 - * A MANIFEST file.
 - * `activity.info` with metadata
 - * A .svg icon
 - * Translation files
 - * The source code (you'll need a class that extends `Activity`)
 - For an example, let's take a look at Fortune Hunter, <http://git.sugarlabs.org/project-xavier/mainline/trees/master/MAFH.activity>
 - What modules to use when writing your code? You can use either of the following. They will both be installed on whatever XO your activity makes its way to.
 - * PyGTK - <http://www.pygtk.org/tutorial.html>
 - * pygame - <http://www.pygame.org/wiki/tutorials>

Week 05, Day 1: Projects

- Lightning Talks
- Introduction to Sugar

Helpful Hints – A list of external resources

git

- [git cheat sheet](#)

vim

- [vim cheat sheet](#)

README.rst – Humanitarian Free/Open Source Software Course @ RIT

This is an all-purpose repository for storing some content, but mostly tools for teaching the open source projects seminar @ RIT.

Future tools could include things like scripts to produce blog/commit/unittest statistics. This is also a place the syllabus could live, where students could fork and produce pull requests.

Setting up your environment

Before you can do anything with this (build the documentation or run any of the scripts) you'll need to setup and activate a python `virtualenv`. Run the following at the command prompt...

On Linux/Mac OS X

If you don't have `virtualenv` installed yet, try:

```
$ sudo easy_install virtualenv virtualenvwrapper
```

If you're using a distro like Fedora or Ubuntu, you should try this instead:

```
$ sudo yum install python-virtualenv
```

Once you have `virtualenv` installed, you should be able to run:

```
$ virtualenv --no-site-packages -p python2 sphinxenv
$ source sphinxenv/bin/activate
$ git clone git@github.com:YOUR_USERNAME/hfoss.git
$ cd hfoss
$ python setup.py develop
```

On Windows

At the windows command prompt:

```
$ virtualenv --no-site-packages -p python2 sphinxenv
$ sphinxenv/Scripts/activate.bat
```

In msysGit or git-bash:

```
$ git clone git@github.com:YOUR_USERNAME/hfoss.git
```

Back in the windows command prompt:

```
$ cd hfoss
$ python setup.py develop
```

Building the “Documentation”

The “documentation” for the course (the syllabus, all the homework assignments, notes on the lectures) are all kept in the `doc/` directory of this repository. The files all end with the extension `.rst` which is the file extension for the `reStructuredText` markup language. They are all furthermore tied together the the `sphinx` framework for building integrated docs.

You might notice that the syllabus, et. al. is hosted on <http://readthedocs.org/>. The upstream github repository has a hook installed that automatically triggers a `git pull` at <http://readthedocs.org> from <http://github.com>. Thus, every time we change the docs here, they are automatically re-built into HTML for us and posted online. Awesome!

This however means that we should be careful before we push anything to github, or it will ‘go live’. To be careful, you should rebuild the documentation locally (on your machine) to check that whatever modifications you made to the `.rst` files actually renders into the HTML that you want.

In order to do that, first make sure you have your virtualenv activated.

Being certain of that, in the root directory, simply run:

```
$ sphinx-build -b html doc html-output
```

The html documentation will be generated in `html-output/`. Check `html-output/html/index.html` to see if it exists.

Note: If your machine complains that ‘sphinx-build’ is a command that could not be found, try running “`$ python setup.py develop`” in the root of the hfoss repository first. That `setup.py` file contains information about all *other* open source projects that are *required* for this project, and will automatically install them from <http://pypi.python.org/>

Validating the `data/students.yaml` file

The `data/students.yaml` file is a structured data file that keeps track of all the students in the class and metadata about them. Using this file and the bindings in `lib/hfoss/model/students.py` we can build scripts that count how many lines of code each student modifies each week, or how many words/blogpost, or whatever we like.

The data format (YAML) can be a little prickly though. It is *whitespace-sensitive*, meaning that how many spaces you put before an entry on each line has an impact on how the data is interpreted. It also means that tabs and spaces are distinctly different in their meaning. It also means that editing such a file is easy to mess up.

In order to ensure that you don't introduce any unparseable errors into the file, there is a script in `lib/hfoss/model/validate.py` that reads in the file and checks each entry. You should run it after every time you edit `data/students.yaml`.

In order to run the `validate.py` script, make sure you have your `virtualenv` activated.

In the root of the cloned source directory, run:

```
$ python lib/hfoss/model/validate.py
```

Homework - First Flight

The purpose of this homework assignment is to introduce students to their first FLOSS practices. Read it in full, there are a number of graded deliverables.

The due-date is listed in the *Syllabus*.

IRC

IRC is one of the primary means of communication for a FLOSS community, particularly for *informal* communication.

There is a course IRC channel on `irc.freenode.net`. The channel is `#floss-seminar`. Communicating regularly in IRC factors into the *FLOSS Dev Practices* component of your final grade.

Tasks:

- Download and install an IRC client on your development machine.
 - Windows: [mIRC](#)
 - Mac OS X: [Colloquy](#)
 - Linux: [irssi](#)
- Choose a nick and [register yourself with the NickServ](#).
- Connect to `#floss-seminar` on `irc.freenode.net` and introduce yourself.
 - The instructor's nick is `threebean`.

It is a good practice to “hang out” in IRC channels of projects that you use and especially of projects that you contribute to. Here you can find early alerts regarding any upcoming major changes or security vulnerabilities. It is also the easiest (lowest overhead) method for getting your questions answered.

Note: Only for the brave – if you want to be completely awesome, you can setup a proxy node so you are always logged in. People can leave you messages this way.

If you want to be completely completely awesome, you can setup [BitlBee](#) so you can tweet from your IRC client.

Mailman

Discussion mailing lists are a more formal mechanism of communication for FLOSS projects. More formal than *IRC*, less formal than bug trackers. Discussion mailing lists are often used to ask questions, announce upcoming releases and beta tests, and to debate redesigns and refactors. The advantage here is that mailing lists are typically archived and indexed by Google; discussions that should be preserved for posterity should occur here.

There is a [GNU Mailman](#) discussion list for the course hosted by RIT.

Tasks:

- Subscribe to it at <https://lists.rit.edu/mailman/listinfo.cgi/floss-seminar>
- Write your first email to floss-seminar@lists.rit.edu, introducing yourself. Include your name, major, hometown, and favorite color.

Communicating regularly over the course *mailman* list (asking and/or answering questions) factors into the *FLOSS Dev Practices* component of your final grade.

Blogging

Setup a blog if you don't have one. Much like mailing lists, blogs are archived, indexed by Google, and therefore preserved for posterity. When you encounter a technical challenge, typically you google for a solution and you typically find that solution in a blog post of some developer who has run into a similar situation. Blogging about your attempts, successes and failures (and writing tutorials!) is a best practice for increasing the general body of searchable knowledge available, for increasing the *Wisdom of the Ancients*.

Blogs around a topic are also typically aggregated by a [planet](#) (an RSS feed aggregator). This way, all developers blogging about *Project X* can have their blog posts fast-tracked to a readership subscribed to *Planet X*. For instance, here's a link to [Planet Python](#).

The Planet for the course is hosted at <http://threebean.org/floss-planet/>. There are instructions for how to subscribe your blog to it in the *Patch the Course Project* section below.

You must create a blog (if you don't have one already) and write at least one post per week about your progress, attempts, successes, failures, reflections, and/or all of the above.

Tasks:

- Create a blog if you don't already have one. There are lots of free services available. You might try <http://wordpress.com> or <http://blogspot.com>.
- Write an introductory post relevant to the course. The topic is your choice!

github

Code *forges* are service sites around which FLOSS development orbits, some of the more popular sites are [github](#), [bitbucket](#), [sourceforge](#), and [launchpad](#).

For your own enlightenment, review the following comparisons of the different forges:

- [Timeline](#)

- Metadata
- Artifacts
- Features
- Revision control
- Policies

You'll need to create your own account on github.com. All development for this course should be tracked on that forge. Github is, after all, [the most popular forge](#).

Tasks:

- Create a [github](#) account if you don't already have one.

Patch the Course Project

Check out the source repository for this course; it's hosted at <https://github.com/ralphbean/hfoss>.

Inside the repository, we'll keep an index of all the students in the course and metadata about them (you!).

Tasks:

- Load up the git cheatsheet listed at *Helpful Hints – A list of external resources* and keep it nearby.
- Work through this [git tutorial](#) if you don't have any experience with git.
- Fork [the repository](#) (link to [github help](#) on this).
- Clone a local copy.
- Follow the instructions in `README.rst` to setup your environment.
- Edit the file `data/students.yaml`. Perhaps obviously, it is a [YAML](#) file. Add yourself to the file with the necessary keywords.
- Verify that you added yourself correctly by running the script located at `lib/hfoss/model/validate.py`
- Edit the file `planet/config.ini`. Look at the very bottom of the file and there will be the beginnings of a list of subscribed blogs. Add your blog's RSS feed (or a topical sub-feed) to this list. Make sure its a working RSS URL! (Once the patch is accepted upstream and pushed to production, this should add your blog feed to the [course planet](#).)
- If everything checks out, then
 - Commit your change
 - Push to your github repository
 - Issue a pull request through the web interface.

Homework - Bugfix

Real learning in computing comes more from doing and less from studying. Debugging, figuring out how some software works and how it doesn't, is an interactive process that develops basic engineering practices and, in the open source context, community engagement and collaboration.

Find a bug

A bug can be anything: an unintended side-effect in a low-level routine, a user-interface cleanup, a feature enhancement, grammatical errors or lack of clarity in the project's documentation.

Broadly, you have two different options here. You can

- Find a known bug (or feature enhancement) listed in the project's bug tracker.
- Find a bug yourself by using the software.

In the event of the second case, make sure you file the bug in the project's tracker before proceeding.

You can fix a bug in any project you like. The best to pick is something you *already use*, something with which you're already familiar. If you can't think of any projects to investigate off the top of your head, here's a list of suggestions.

- Scope the OpenHatch Volunteer Opportunity Finder for [Bite-sized Bugs](#)
- Use the search function at <http://github.com/> and filter by language (to a language that you know).
- Look up some of the bounties at Gun.io <http://gun.io/>

Really, the sky is the limit here.

Note: For background, you might want to also check out the project on <http://ohloh.net/>. It can help you characterize what kind of community orbits around your choice.

Use the Source, Luke

Once you've identified a bug that needs fixing, you'll need to get ahold of the source. In most cases, the code for a project will be hosted on a forge and the process of forking and cloning the source will be straightforward. If you forget how to do this for [github](#), you can refer to *Homework - First Flight*.

For whatever project you've chosen, you should ask that project's community for help. Look for *IRC* channels and project mailing lists. You'll be communicating with developers who have a lot on their plate so make sure to **be polite and leave your ego at the door**.

Find the code related to the bug; use whatever code navigation tools you're more familiar with. The instructor's favorite method is: `grep -inr "some string" project_src/`.

Fix the bug, this may require some thinking, and some more asking around.

Test your fix! Project maintainers hate nothing more than receiving a patch that breaks every other function of the project. Often, projects have built-in test suites. If yours does, run it!

Prepare your patch with descriptive commit messages. Follow the method for submitting patches recommended by your project and submit!

Make sure the project community can easily understand what you did and why you did it.

Make sure there is a reference in the tracked bug ticket to your patch (that is, if the project maintains a bug tracker).

The Deliverable

Write a blog post about this process and provide relevant links where possible to documentation.

- A link to the patch(es) hosted somewhere on the web, usually forges provide the ability to link to changesets.
- A link to any mailing list discussions archived on the web
- Snippets of any relevant IRC conversations.

You will be graded on your post and the explanation of your process. Extra kudos (but not extra credit) for super epic patches.

An Afterthought (not required)

Once your patch has been accepted, mosey on over to <http://ohloh.net>.

- Create an account
- Find the project you patched
 - If it doesn't exist, you can add it yourself
- "Claim your position" as the author of the commit(s) you sent in to increase your rank among open source developers of the world!