Class Syllabus

Instructors

Dr. Mark Palmeri, M.D., Ph.D.
`mark.palmeri@duke.edu`
Office Hours: TBD (258 Hudson Hall Annex)

Suyash Kumar, CTO & Co-Founder of Gradient Health
`suyash.kumar@duke.edu`
Office Hours: Tuesday 16:20-17:20 (Teer 106 or at http://chat.suyashkumar.com)

Dr. David Ward, Ph.D.
`david.a.ward@duke.edu`
Office Hours: Thursday 21:00-22:00 (Google Hangouts)

Google Hangout office hours can be joined by a URL that will be sent to the class via email.

Teaching Assistants

Ismael Perez
`ismael.perez@duke.edu`
Office Hours: Tuesday 14:00-15:00 (Gross Hall 3rd Floor Cubicle 36)

Tanvi Kamat Tarcar
`tanvi.kamat.tarcar@duke.edu`
Office Hours: Wednesday 11:20-12:20 (Teer Basement)

Lecture

Tues/Thur 15:05-16:20
Teer 106

All lecture content will be outlined in Lectures.

Course Overview

Software plays a critical role in almost all medical devices, spanning device control, feedback and algorithmic processing. This course focuses on software design skills that are ubiquitous in the medical device industry, including software version control, unit testing, fault tolerance, continuous integration testing and documentation. Experience will be gained in Python and JavaScript (and potentially other languages).

The course will be structured around a project to build an Internet-connected medical device that measures and processes a biosignal, sends it to a web server, and makes those data accessible to a web client / mobile application. This project will be broken into several smaller projects to develop software design fundamentals. All project-related work will be done in groups of 3 students.

Prerequisites

Basic familiarity with programming concepts (e.g., variables, loops, conditional statements).

Course Objectives

- Software version control (`git`, GitHub)
- Device programming fundamentals
  - Review of data types, variables, loops, conditional statements
  - Python (v3.6): numpy, scipy, pandas, scikit
  - Virtual environments & dependency management (`pip`, `requirements.txt`)
  - Use of a programming IDE
  - Debugging (`pudb`)
- Testing
  - Unit testing
  - Functional / System testing
  - Continuous integration (Travis CI)
- Fault tolerance (raising exceptions)
- Logging
- Resource profiling (`cProfile`)
- Documentation
  - Docstrings
  - Markdown
  - Sphinx
  - ReadTheDocs
- Working with data
  - Data Storage (Text, Binary, HDF5, MongoDB)
  - Data Wrangling
- Data Processing & Display
  - Jupyter Notebooks
  - Matplotlib / Seaborn
  - Pandas (DataFrames)
  - scikit-image & scikit-learn
- Define software specifications and constraints (Requests for Comments, RFC)
- Servers
  - Design & Implementation of a biomedical web service (Python Flask)
  - HTTP & RESTful APIs
  - Docker and dependency management intro

Attendance

Lecture attendance and participation is important because you will be working in small groups most of the semester. Participation in these in-class activities will count for 15% of your class grade. It is very understandable that students will have to miss class for job interviews, personal reasons, illness, etc. Absences will be considered \emph{excused} if they are communicated to your instructors at least 48 hours in advance (subject to instructor discretion as an excused absence) or, for illness, through submission of a Short Term Illness Form (STIF) **before** class. Unexcused absences will count against the participation component of your class grade.

Textbooks & Resources

There are no required textbooks for this class. A variety of online resources will be referenced throughout the semester.

- Python Resources

Project Details

Project details will be discussed in lecture throughout the semester.

Grading

The course GitHub repository will host all Assignments. Due dates–including those that change–will be announced in lecture and by Sakai announcements that will be emailed to the class.

The following grading scheme is subject to change as the semester progresses:

Participation	15%
Assignments	35%
Final project	50%

Class Schedule

The course schedule is very likely to change depending on progress throughout the semester. The updated schedule will always be available in the GitHub course repository.

Date	**Lecture**	**Assignment**
Tues Aug 28	Class Introduction, Objectives and Logistics	Setup Course Tools & Git Fundamentals
Thurs Aug 30	Git: Repo Setup, Cloning/Forking, Issues, Branching, Pushing/Pulling	
Tues Sept 04	Git Workflow	Getting Started with git
Thurs Sept 06	Python Virtual Environments	
Tues Sept 11	Python Fundamentals	
Thurs Sept 13	Class Cancelled (Severe Weather)	
Tues Sept 18	NO LECTURE (NC State Career Fair)	

Thurs Sept 20	Unit Testing: (py.test) & Continuous Integration (Travis CI)	
Tues Sept 25	Unit Testing: Comprehensive unit tests	Unit Testing & Continuous Integration (Travis CI)
Thurs Sept 27	IEC 62304	
Tues Oct 02	Unit Testing: Approximations, fixtures & more; Docstrings	PythonFundamentals.ipynb (Sakai) & IEC 62304 Assessment (Sakai)
Thurs Oct 04	PyCharm; Debugging; Property Decorators	
Tues Oct 09	Fall Break	
Thurs Oct 11	Functional Decomposition; Python: Data Structures	Heart Rate Monitor
Tues Oct 16	Exceptions & Logging	
Thurs Oct 18	Dictionary Type, Classes, Property Decorators, Numpy Docs	
Tues Oct 23	HRM Assignment Work	
Thurs Oct 25	HRM Assignment Work	
Tues Oct 30	Intro to Web Services & Cloud-connected Devices	
Thurs Nov 01	Python Flask, API design, virtual machines (Duke OIT VMs)	Call web services (SendGrid, Twilio, etc)
Tues Nov 06	Flask continued, deployment, production considerations	
Thurs Nov 08	Introduction to Databases	Heart Rate Sentinel Server
Tues Nov 13	Introduction to Security + Assignment Work	TBD
Thurs Nov 15	Working Project Code	
Tues Nov 20	Final project assignment	
Thurs Nov 22	Thanksgiving	Refactor Project Code
Tues Nov 27	Final project work	
Thurs Nov 29	LDOC!	
Tues Dec 04	Final project work	
Thurs Dec 13	Final project due	

Distributed Version Control Software (git)

Software management is a ubiquitous tool in any engineering project, and this task becomes increasingly difficult during group development. Version control software has many benefits and uses in software development, including preservation of versions during the development process, the ability for multiple contributors and reviewers on a project, the ability to tag *Releases* of code, and the ability to branch code into different functional branches. We will be using GitHub to centrally host our git repositories. Specifically, we will be creating student teams in the Duke BME Design group. Some guidelines for using your git repositories:

- *All* software additions, modifications, bug-fixes, etc. need to be done in your repository.
- The *Issues* feature of your repository should be used as a ”to do” list of software-related items, including feature enhancements, and bugs that are discovered.

- There are several repository management models that we will review in class, including branch-development models that need to be used throughout the semester.
- Instructors and teaching assistants will only review code that is committed to your repository (no emailed code!).
- All of the commits associated with your repository are logged with your name and a timestamp, and these cannot be modified. Use descriptive commit messages so that your group members, instructors, and teaching assistants can figure out what you have done!! You should not need to email group members when you have performed a commit; your commit message(s) should speak for themselves.
- Code milestones should be properly tagged.
- Write software testing routines early in the development process so that anyone in your group or an outsider reviewing your code can be convinced that it is working as intended.
- Modular, modular, modular.
- Document!
- Make commits small and logical; do them often!

We will review working with git repositories in lecture, and feedback on your software repository will be provided on a regular basis.

Online Slack Channels

We have online help through the Duke Co-Lab Slack team. We have started three specific channels for this class: `#linux`, `#git` & `#python`. Please add yourselves to these channels to get help from your instructors, your TAs and the Duke community!

Duke Community Standard & Academic Honor

Engineering is inherently a collaborative field, and in this class, you are encouraged to work collaboratively on your projects. The work that you submit must be the product of your and your group's effort and understanding. All resources developed by another person or company, and used in your project, must be properly recognized.

All students are expected to adhere to all principles of the Duke Community Standard. Violations of the Duke Community Standard will be referred immediately to the Office of Student Conduct. Please do not hesitate to talk with your instructors about any situations involving academic honor, especially if it is ambiguous what should be done.