ECE-C301 Advanced Programming for Engineers
Instructor: Dr. Shackleford
Course Syllabus and Policies

Course Summary

This course will cover advanced usage and understanding of programming concepts using Python and C within the Linux environment. Students should already have a working knowledge of bash, python, pylint, tmux/GNU screen, X11 tunnelling, and at least one terminal based editor (nano, vim, etc) from ECE-203. I have created and posted video tutorials on the ECE-C301 website for those requiring a refresher on working with these tools. The thanos development server (thanos.ece.drexel.edu) will provide the development environment necessary to complete all course assignments. You will receive your SSH login credentials for this machine at the start of the term. Please note that your account will expire at the end of the term, preventing future logins.

The course consists of two distinct parts. In the first part, Object Oriented Programming (OOP) and simulation concepts will be covered using the high-level language Python. The second part of the course will introduce the low-level language C as it pertains to Computer Engineering majors and will serve as a foundation for future embedded firmware and system level software authorship as well as a means to better understand the underlying mechanisms implemented by the Python interpreter in terms of CPU control and memory organization.

Mandatory Textbooks

Think Python: An Introduction to Software Design
by Allen Downey
O'Reilly Media 2015
ISBN-13: 978-1449330729
(Freely Available) http://www.greenteapress.com/thinkpython

C Programming: A Modern Approach, 2nd Edition
by K. N. King
W. W. Norton & Company
ISBN-13: 978-0393979503

Optional Textbooks

Learning Python, 5th Edition
by Mark Lutz
O'Reilly Media 2013
ISBN-13: 978-1449355739

Coverage
Course coverage begins with a continuation of Python topics introduced in ECE-203. Advanced programming topics including inheritance, polymorphism, and iterators are covered within the Python environment. C coverage focuses on a low level understanding of the language as pertinent to Computer Engineering. Coverage of the C language is specifically constrained to the C89 language specification and Python to version 2.7 unless otherwise explicitly indicated.

- **Programming Simulations** *(Week 1)* – Introduction to cellular automaton simulations; Conway's Game of Life; maintaining world state; simulation time; update rules; working with command line parameters in Python using `argparse`; block image transfer (i.e. blitting); using off-screen frame buffers; imposing toroidal boundary conditions using modular arithmetic; animating plots using `matplotlib`.

- **Object Oriented Programming** *(Week 2)* – Class definitions as a blueprint, objects as instances of a class, the use of `self` to refer to the current instance, methods, instance attributes, class attributes, object instantiation, object initialization/construction using `__init__`, explicit instance passing to class methods, implicit instance passing to methods (traditional method calling), static method; inheritance: subclasses, superclasses, overriding methods, operator overloading.

- **Polymorphism** *(Week 3)* – Duck Typing; overview of Polymorphism; abstract base classes; the `NotImplementedError` exception; reading Python stack traces; using dictionaries of functions; lambda functions

- **Building Custom Iterators** *(Week 4)* – Importance of separating program logic from user interface (UI) logic; the iterator protocol; `__iter__()` for custom container/iterable objects; `__iter__()` and `next()` for custom iterator objects; the `IndexError` exception; the `StopIteration` exception; "proper" separation between custom iterable objects and their custom iterator objects; "lazy" fusion of custom iterable objects with their iterator implementations; using `yield` to fulfill the iterator protocol requirements via automatic generator construction

- **Introduction to C** *(Week 5)* – **Virtual Address Space Basics:** the `.text` segment, the stack, shared libraries; **Function Mechanics:** parameter passing via the stack; return addresses; the stack trace; function return values via CPU register `eax`; the program counter CPU register; **C's Build Trajectory:** pre-processor intro; compiling to assembly; assembling to object files; linking object files and shared libraries to form executables; basic `gcc` usage; **Overview of Program Loading and Execution**. *Reading from King: Chapters 2 and 3*

- **Control Flow in C** *(Week 6)* – Logical expressions; **Selection Statements:** the `if-elseif-else` control flow construct, the `switch-case` control flow construct; **Loops:** the `while` construct, the `do-while` construct, the `for` construct; **Control Flow Modifiers:** the `break` keyword, the `continue` keyword, the `goto` keyword; **Functions:** function definitions, function declarations.
  Reading from King: Chapters 5, 6, and 9

- **Memory, Types, and Pointers** *(Week 7)* – The `char` type; the `int` types; `limits.h`; the `float` and `double` types; arrays; pointers; memory organization of arrays.
  Reading from King: Chapters 7, 8, 11, and 12
- **Strings, Structures, and Enumerations** *(Week 8)* – Sequences of `char` elements; `null` termination (`\0`); introduction to `string.h`; introduction to `structs`; the `->` operator; enumerated types (i.e. `enums`).
  Reading from King: Chapters 13 and 16
- **Heap Memory Management** *(Week 9)* – Working with dynamic memory, `malloc()`, `realloc()`, `calloc()`, `free()`; linked lists
  Reading from King: Chapter 17
- **Advanced Memory Organization** *(Week 10)* – Using linked lists to build a simple hash table (i.e. dictionary or associative array).

Grading Policy
The grading in this course is based on lab assignments, homework assignments, and special projects. The cumulative grade is based on the following:

- 30% In-lab Programming Assignments
- 30% Take-Home Programming Assignments
- 40% Programming Projects

The class rank in both the exam scores and the overall scores will be considered for the final grade. Homework is always to be submitted at the time indicated on the assignment. Homework submitted after the due date will not be accepted and will be graded at 0 points. In the event a student is unable to attend class, he/she should make alternate arrangements to deliver the homework to the instructor *before* it is due. All assignments submitted for a grade must be able to both build and run on the provided development server (`thanos.ece.drexel.edu`). Submissions not meeting this requirement will receive a zero without further consideration.

Academic Honesty Policy
Each student is expected to complete all assignments independently unless otherwise explicitly instructed. It is unacceptable to copy another student's work or solutions from any other source. Violators of this policy will be reported to the Office of Student Conduct and Community Standards (SCCS). Academic integrity violations could result in failure for the course or the assignment among other sanctions determined by the instructor. A second violation of the academic integrity policy will likely result in suspension.

Instructor
Dr. James Shackleford
shack@drexel.edu
Bossone 211
Office Hours: Wed 3–4 pm