Genesee Community College
Syllabus – CIS127 – Computing in Math, Science and Engineering
Fall 2018

Instructor: Ken Mead
Email: kjmead@genesee.edu
Office location: D395 Math Science Area, Building D, Main Campus
Office hours: MW 11:15am - 1:15pm, TR 12 -12:30pm, Friday by Appointment.
Phone: 585-343-0055 x 6381
Homepage: http://faculty.genesee.edu/kjmead

Catalog description: An introduction to computer programming and problem solving, with special emphasis on problems found in mathematics, science, and engineering. Students will use software such as Maple and Excel, and/or a programming environment such as MatLab or Python, to efficiently and effectively solve problems by developing a strategy, applying appropriate techniques, and validating results.

Prerequisite: MAT140 or higher.

Course Student Learning Outcomes (CSLOs):
Upon successful completion of this course, the student will be able to:

1. Apply problem-solving techniques to solve a mathematical, scientific, or engineering problem using a computer application.
2. Apply problem-solving techniques to solve a mathematical, scientific, or engineering problem using a programming language.*
3. Write at least one program that solves a problem through the use of decision-making blocks
4. Write at least one program that solves a problem through the use of looping.
5. Write at least one program that solves a problem through the use of user-defined methods or functions.
6. Write a least one program that solves a problem through the use of data structures such as lists and arrays.

Course Overview:

Part 1 of Course – Python

Python is an easy to learn programming language that is used extensively in mathematics, science and engineering. It is open-source and completely free to use. There are two versions of Python in heavy use today: python 2 and python 3. We'll focus on python 3.

Lesson 1: Installing Python. We'll be using the Anaconda Python 3 distribution on PC and the standard python3 distribution on the csnlinux.genesee.edu server.

Lesson 2: Why we Program – in this lesson, we'll look at the big picture as to why we write computer programs, and why python is a good tool to use.

Lesson 3: Variables and Expressions - in this lesson, we cover how a program uses the computer's memory to store, retrieve and calculate information.

Lesson 4: Conditionals – in this lesson, we move from sequential code that simply runs one line of code after another to conditional code where some steps are skipped.

Lesson 5: Functions – in this lesson, we will learn about what functions are and how we can use them to solve problems.

Lesson 6: Loops and Iteration – in this lesson, we examine loops and iteration. Loops are the way we tell Python to do something over and over. Loops are the way we build programs that stay with a problem until the problem is solved.

Lesson 7: Strings – in this lesson, we look at how Python stores and manipulates textual data using string variables and functions.

Lesson 8: Files – in this lesson, we learn how to open data files on your computer and read throught the files using Python.

Lesson 9: Lists – in this lesson, we look at Python's simplest data structure - the list. Lists can store more than one item in a variable.

Lesson 10: Dictionaries – in this lesson, we look at dictionaries. The dictionary data structure allows us to store multiple values in an object and look up the value given a specific key.

Lesson 11: (time permitting) Tuples – in this lesson, we look at tuples. The tuple is a Python data structure that is like a simple and efficient list.

Lesson 12 (time permitting): Object Oriented Python – in this lesson, we take a quick look at how Python supports the Object-Oriented programming paradigm.

Part 2 of Course – MATLAB / Octave

MATLAB is a popular programming language and compuational toolbox used by engineers and scientists. Octave is open source software that strives to match MATLAB's capabilities feature by feature. In this course, we'll use Octave, but students can optionally use MATLAB if they wish. In the following descriptions, we'll use MATLAB and Octave interchangeably.

Lesson 1: Introduction to MATLAB – in this lesson, we'll familiarize ourselves with the Matlab user interface.

Lesson 2: Matrices and Operators – in this lesson, we'll learn about the basic unit with which we work in Matlab – the matrix, along with many of the operators to perform arithmetic on these matrices.

Lesson 3: Functions – in this lesson, we'll learn how to create Matlab functions so we can break complex problems into manageable and reusable components.

Lesson 4: The MatLab Toolbox – in this lesson, we'll learn about a bunch of Matlab's built in functions. We'll also learn about polymorphism, random numbers, how to get input from the keyboard, and how to plot graphs.

Lesson 5: Conditionals – in this lesson, we will learn how to make Matlab make decisions using the if statement. We will also examine relational operators and logical operators.

Lesson 6 (time permitting): Loops – in this lesson, we will learn Matlab's loop constructs, the for loop and the while loop. We will also learn about nested loops.

Required materials:
Python for Everybody – Exploring Data in Python 3. Author: Charles Severance. Free downloadable PDF at http://www.py4e.org/book

Optional materials:
Computer Programming with MATLAB, first revised PDF edition, June 2015. Authors: Fitzpatrick and Ledeczi. Download for $10 at http://cs103.net/buy/

Introduction to Python for Computational Science and Engineering (A beginner's guide to Python 3), July 2017. Author: Hans Fangohr, Faculty of Engineering and the Environment, University of Southampton. United Kingdom. Free downloadble PDF at https://github.com/fangohr/introduction-to-python-for-computational-science-and-engineering/blob/master/Readme.md

Grading:
Final grades are assigned according to the following scheme, with the final average rounded to the nearest integer (in %): 92 or higher = A, 90-91 = A-, 88-89 = B+, 82-87 = B, 80-81 = B-, 78-79 = C+, 72-77 = C, 70-71 = C-, 68-69 = D+, 62-67 = D, 60-61 = D-, 59 or less = F.

Grades will be weighted as follows:

- 60% - three in-class exams on 9/25, 10/25 and 12/4. (300 total points).
- 40% - Programming projects and homework assignments. (200 total points).

To compute your final grade, divide your total number of points by 5, round to the nearest integer, and compare to the scale above.

Breakdown of Projects and Assignments:

There will be many short programming assignments that must be completed throughout the semester. These will be graded as either complete or not, all or nothing (approximately 50 - 80 total points).

There will also be between 3 and 5 larger programming projects (most likely written in Python) which must be completed to specifications provided by the instructor. These larger projects will be mathematical, scientific, or engineering focused in nature. (approximately 120 - 150 total points).

Student Responsibilities:

- Come to class, especially on exam day! If you must miss an exam due to extenuating and unavoidable reasons, you need to contact your instructor beforehand to re-schedule. Missing an exam without permission and good reason will result in a grade of zero.

- Complete projects on time! Late penalties will apply as follows: 10% within 24 hours of deadline, 25% within first week of deadline, 50% after the first week. If a project is not

completed at all, it will receive a grade of zero and 2 points will be deducted from your overall average.

- Stay engaged! Come up with ideas for problems to solve! Enjoy the material because it's really good stuff!!!

- Put the phone away. Respect your classmates. Don't be disruptive. Blah blah blah. You know the drill.

Plagiarism and Cheating:

Cheating is obtaining or intentionally giving unauthorized information to create an unfair advantage in an examination, assignment, or classroom situation. Plagiarism is the act of presenting and claiming words, ideas, data, programming code or creations of others as one's own. Plagiarism may be intentional – as in a false claim of authorship – or unintentional – as in a failure to document information sources using MLA (Modern Language Association), APA (American Psychological Association), Chicago or other style sheets or manuals adopted by faculty at the College. Presenting ideas in the exact or near exact wording as found in source material constitutes plagiarism, as does patching together paraphrased statements without in-text citation. The purchasing or sharing of papers or projects between students or the re-use of papers or projects submitted for more than one assignment or class also constitutes plagiarism.

In short - **do your own work!** This means no cutting and pasting code from the internet. Students are encouraged to discuss programming assignments and projects, and even develop overall strategies for solving problems together, but must ultimately write their own code. Any evidence of plagiarism, or of cheating or sharing answers on a test, will result in a grade of zero for the assignment, project or test. Multiple offenses will result in a grade of F for the course.

Accessibility Statement:

If you have a physical, psychological, medical or learning disability that may impact your coursework or participation in this class, please contact the Assistant Dean of Student Services/Disabilities Coordinator, Success Coach, or Academic Advisor who will arrange an intake meeting. The Assistant Dean/Coordinator will determine with you what accommodations are necessary, appropriate and reasonable. All information and documentation is confidential.

The instructor reserves the right to make any reasonable and necessary modifications to the statements above. This document is subject to change.