MA/CS 21: Programming I **Spring 2020**

Syllabus

editec to correct the B+ error on Grading section

Instructor:	Dr. Jessica Kintner
Office:	Galileo 106 B
Office hours:	TBA and by appointment
E-mail:	`jkintner@stmarys-ca.edu`
Text:	**Python Programming: An Introduction to Computer Science**, (3rd Edition) by John Zelle.
Software:	**Python 3** There will be links on the course web page.
Meetings:	T Th 8-9:35, Gal 205
Web pages:	`http://physics.stmarys-ca.edu/courses/MACS021/` I greatly prefer to keep most course information on a regular html page (outside of moodle). One advantage of this is that you can search for things related to the course and they might show up! (Moodle pages do not.) I will keep grades on moodle for security There is a link on moodle to the html page

---

Course Description: Today many fields of study are critically dependent on the computational power provided by digital computers. This course will focus on learning to program a computer in the language Python 3.

Course Content and Goals: (*from Jim Sauerberg's 2017 syllabus*). The course will have three parts: What a program is, How to write programs, and How to think like a programmer. We will use Python, so this course is an introduction to writing Python programs that are easy to read, test, modify, and debug.

Learning Objectives: (*from Jim Sauerberg's 2017 syllabus*). Students will

- Appropriately use the vocabulary of computing and computer science.
- Demonstrate an understanding of the syntax and semantics of variables, assignments and expressions.
- Explain the purpose of existing code segments, modify existing programs and program segments to meet new requirements or to eliminate semantic and synax errors.
- Document a program with written comments using both plain English and programming terms so that a knowledgeable reviewer can immediately understand the purpose and logical structure of the program.
- Select and use appropriate types and built-in data structures (int, float, bool, string, list, tuple) and control structures (functions, loops, decisions, recursion) to perform given tasks.
- Perform basic algorithm design and analysis.
- Indicate an intuitive understanding of program flow, memory use, and information passing in Python. This includes storage, assignment, input/output and function parameters/arguments.
- Develop the logic of a program from problem specifications.

Homework: (*Back to my own words.*) Homework will be of a couple types. Programming is something you must *do* rather than read. As you read the text, you should have an open Python session, and you should enter every command in the text book into your own session.

In addition to the reading work, programming problems may be assigned each day. Be prepared to ask questions, present and/or discuss these problems the very next class period, It is very important that you attempt them in between each class meeting. I know this could be a challenge between Tuesday and Thursday. (See the next section on In Class Work.)

Since homework is a fairly large percentage of your grade in this class, it is important how you work. I encourage you to *discuss* problems with each other, but write up your own solutions individually. Direct copying is prohibited and will be considered a violation of the Honor Code! You should also cite any sources and give credit to anyone you worked with. (Citing someone you work with is clearly not a violation of the Honor Code–so do it!! I consider this sort of practice for co-authoring papers or journal articles. Notice how on this syllabus, I gave credit to Jim Sauerberg for his syllabus.) Exceptions to this are only work done by me, or perhaps in my office, or if we have all discussed the problem in class. Of course, if you use another source, you should credit that as well (just as if you were writing a paper!) You will not lose any points for this!!! You will, on the other hand, get in big trouble for plagiarism.

Chris Ray had this paragraph in his syllabus for the Physics version of this course, and I really like it: *Regarding the sharing of code: Don't do it. It is great to help each other figure out how to solve problems, but it is very hard to help someone understand by sharing code. So if you are not able to share code how can you help each other? The way to support each other in learning to code is to talk about it, talk about the ideas behind what you are doing. This helps everyone a lot, because there is a language used to talk about programming and computation, and learning to use that language is part of learning the ideas. If you find yourself saying something like "Type ex equals sign zee left parenthesis one colon five right parenthesis semicolon" then you are not doing the right thing. Say instead something like "you need to specify a subsection of the array".*

Doing the homework is the best way to learn the material. Please take it seriously and make every effort to do it on time.

In Class Work: When you come into class, you should be prepared to put homework problems on the projector or board as appropriate and/or ask questions about any problems you got stuck on or are confused about. You also might ask questions about the last lecture or sections of the book you've been reading.

Studies show that traditional lecture is the worst way for to learn. Believe it or not, even spending the hour doing homework problems is more effective

than pure lecture. (See any of tons of papers by Eric Mazur.) I suspect this is particularly true for programming. We will use a variety of activities in class such as board work, small group work, problem solving, and coding. **If you have your own laptop with your version of Python, I hope you will bring it to class each day!** In fact, much of what we will do needs only a browser, and it could be typed on a tablet if you don't have a laptop. It's easier for me to see if you work on the lab computers, so I like that too, but you will want to access all the things you've worked on, which means you won't want to leave them on the lab computers.

Some of the problems we do or discuss in class will be homework problems, and some will be "new" to the class that day. For those, I will typically have you work in small groups – but each of you entering things in your own Python or colab session. For new work in particular, I want to emphasize that I do not expect you to get it right the first time. We will be learning together. Active engagement with the material is the most important aspect of this kind of in class work. It will be a small class. I will notice if you are always checked out or doing something not related to the class. I will absolutely not mark down for getting things wrong in this situation! We have to make mistakes to learn. All of us. This is the time to make them and learn from them.

And some days, I will do lectures with every attempt to keep you involved.

If I find that it's necessary, I might add more traditional "quizzes" to the mix. They will count in this category.

Exams: There will be two one-hour exams (I might call them both midterms) and a final. The final exam will be two hours long, and it will be cumulative. The nature of the course is such that each exam will depend to some extent on the material before. See the attached schedule for planned dates. (Except for things like power outages or campus closures, exam dates will stay set even if material changes.)

Attendance: Although attendance will not necessarily be taken each day, it is strongly recommended that you attend lecture. The text for this class is challenging. And your In Class Work grade will be affected by more than two absences. You will be responsible for any material presented in class.

Exams cannot be made up without an approved excuse. Approved excuses are such things as illness and family or personal emergencies. If you must miss an exam, you must contact me prior to the exam. Each student is responsible for all assignments, etc, which are given during lecture.

Grading: The *approximate* weighting for the course is shown below.

Homework and In class work	40%
Midterm exams	30%
Final Exam	30%

Escape clause: You can replace 15% of the course grade, from any category, with your score on the final, if it is better. For example, it could look like replacing one midterm, if your score on a midterm were lower than other categories.

Letter grades: At the end of the course, I'll assign letter grades based on numerical scores calculated using the rubric above with:

A	> 93 %
A-	90-92.99%
B+	87-89.99%
B	83-86.99%
B-	80-82.99%

and so on.

College Policies: There have grown to be so many that are the same for all your SMC classes that I will refer you to the Student Handbook. We all should agree to abide by all of these. They include things like the Honor Code (already mentioned) and Student Disability Services.