Practical Computational Methods for Economists (Econ 895 005)
Tuesday, Thursday 3-4:15, Angel Cabrera Global Center 1306 B

Instructor: Professor Kevin McCabe, kmccabe@gmu.edu
All EMAIL correspondence should use the subject line ECON 895

Virtual Office Hours: W 4:00-5:00, Zoom

Course Description: The field of economics is becoming increasingly computational. If you want to be competitive in the job market, you must learn how to think in terms of algorithms and be able to program computations on computers. This class introduces you to computational methods using Python and the Python Scientific stack. This course teaches you basic Python using economics projects as examples. Each week you will complete Jupyter Notebooks that will introduce you to basic concepts. You will then work on several economics projects to gain confidence in your programming skills. This course is divided into four sections described below.

Section I: Python Programming

The first section (5 weeks) introduces you to Python. You will start programming using interactive Jupyter Notebooks. You will learn to design your programs using three programming paradigms: procedural programming, functional programming, and object-oriented programming.

Section II: Python Projects

The second section (3 weeks) introduces three computational projects. You will learn to use Visual Studio Code to edit, debug, and deploy your programs and Jupyter Notebooks to design, document, and use your programs. In the first project, you will design and build a module to construct and measure risk preferences. In the second project, you will design and build a module to construct and solve decision trees. In the third project, you will design and build a program to use decision trees to solve an optimal search problem.

Section III: Scientific Programming in Python

The third section (2 weeks) introduces you to the python libraries that make up the scientific programming stack. These include:

1. **NumPy**: a library that provides a multidimensional array object and an assortment of routines for fast operations on arrays, including statistical sampling.
2. **Matplotlib**: a library for creating visualizations in Python.
3. **SymPy**: a library for performing symbolic mathematics.
4. **Pandas**: a library that provides a general-purpose data frame for working with data.

Section IV: mTree Simulations and Experiments

The fourth section (4 weeks) introduces you to mTree, a concurrent actor-based system building computational microeconomic systems to be used in agent-based simulations and human subject experiments. This includes:

1. Building a Microeconomic System in mTree.
2. Building a VCM for agent-based simulations.
3. Building a mTree experiment.
4. Building a VCM human subject experiment.

Course Materials

A Laptop with Good Internet Connectivity: You will be asked to download and use open-source software throughout the semester.

Textbook: The following books are online and free to use: https://pythonbooks.org/free-books/

Blackboard: This is the central place to look for the notebooks, discussions, resources, and assignments used in this class.

How to Take This Class

Here are hints for taking this class:

Hint One: *Don't fall behind*. Learning to program is a cumulative process. You cannot cram. Section I is critical. Spend as much time as you can on this section. The projects will let you practice what you are learning. Feel free to experiment.

Hint Two: *Practice, practice, practice*. Writing and running small programs is the best way to learn to program. Don't wait to be asked to write code; instead, take a "what will happen if I do this?" attitude. Jupyter notebooks are great for this. Anytime you want, you can open a cell, write some code, and run it. This is the best way to learn.

Hint Three: *Break your code on purpose*. Much of the time spent programming is spent finding and fixing code. When you break your code and create an error, you know what the error is, which will provide more meaning to the error message you get. Later on, when you didn't mean to break the code, you will better understand the error message you get.

Hint Four: *Ask your 'google friends.'* If you get stuck in your program or want to learn how to do something, use the google search engine and ask, "In Python, how do I …?" or "In python, what is …?". You will notice stack overflow often appears. The site is https://stackoverflow.com/questions/tagged/python?sort=MostVotes&edited=true.

Hint Five: *Design first, then program and test incrementally*. Designing first is critical, but then as you implement, you want to write a line or two and then run it to check for errors. This way, you know the most recent error is due to what you did a few minutes ago. Of course, there may be a bigger problem, but you know, have a good starting point for figuring it out.

Hint Six: *Document and use/adhere to a style guide*. Using bad variable names, poor style, and little or no documentation will make your code hard for someone to understand in the future, and that someone will likely be you one week later. It is easy to slip into lousy programming habits and much harder to fix later.

Grading:

The class consists of three projects started in weeks 6, 7, and 8 and must be completed by Friday of week 10. After week 10, your project will be considered incomplete. Your class grade will depend on your class participation and the completion of class projects and exercises. A – regular participation and completes all three projects. B – regular participation and completes at least two projects. C – regular participation and completing at least one project.

Students with Disabilities: If you have a learning or physical difference that may affect your academic work, you must furnish appropriate documentation to the Office of Disability Services. If you qualify for accommodation, the ODS staff will give you a form detailing appropriate accommodations for your instructor. In addition to providing your professors with the appropriate form, please take the initiative to discuss the accommodation with them at the beginning of the semester and as needed during the term. Because of the range of learning differences, faculty members need to learn from you the most effective ways to assist you. If you have contacted the Office of Disability Services and are waiting to hear from a counselor, please tell me.

Honor Code: George Mason University is an Honor Code university; please see the Office for Academic Integrity for a complete description of the code and the honor committee process. What does academic integrity mean in this course? Essentially this:

(1) When you are responsible for a report, presentation, or case study, you will perform that task to the best of your ability. Case studies and quizzes are to be done entirely independently. Any interaction with others during these times violates the honor code.

(2) When you rely on someone else's work in your reports, presentations, or case studies, you will give full credit in the proper, accepted form.

(3) Another aspect of academic integrity is the free play of ideas. Vigorous discussion and debate are encouraged in this course, with the firm expectation that all aspects of the class will be conducted with civility and respect for differing ideas, perspectives, and traditions.

Course Schedule: This schedule might undergo further revision.

Section 1 – Week 1: Python data, expressions, and functions
Week 2: Booleans, conditionals, lists, and loops
Week 3: Dictionaries and file I/O
Week 4: Functional Programming
Week 5: Object Oriented Programming
Section 2 – Week 6: Risk Preference Project
Week 7: Decision Tree Project
Week 8: Optimal Search Project
Section 3 – Week 9: NumPy and Matplotlib
Week A: SymPy and Pandas
Section 4 – Week B: mTree Simulations
Week C: VCM Project
Week D: mTree Experiments
Week E: VCM Project