

Programming and Numerical Methods in Statistics Fall 2011

The instructor reserves the right to change the syllabus during the semester.

Instructor Information:

Kun Deng
Office: West Hall 431
E-Mail Address: kundeng@umich.edu
Web Homepage: http://www-personal.umich.edu/~kundeng/stats607
Office Hours: General: Friday 1-2 pm; other times by appointment

Course Description

This course is the Part I of an advanced introduction to modern python programming for use in statistics, modelling and data analysis. This part of the course will focus on programming fundamentals relevant for research on statistical methodology, and for working with large and complex data sets. We will cover topics such as basic data structures, iteration and recursion, functional programming, classes and object-oriented programming, memory management, strategies for documenting and debugging. Prerequisites: some knowledge of programming in another language is strongly recommended.

Course Objectives On successful completion of this course, students can expect to

1. Use data structures in python to write simple programs.
2. Master abilities to debug, test and document programs in a python development environment.
3. Explain basic principles of procedural programming, apply these principles to solve real-world problems.
4. Handle day to day programming tasks: string and file manipulation, interfacing with external tools and underlying operating systems.
5. Demonstrate some knowledge in algorithm design strategy: divide and conquer, greedy method and dynamic programming etc.
6. Understand basics of classes and object-oriented programming and apply them to design solutions to large scale problems.
7. Have some hands-on knowledge on advanced programming tasks: mathematical packages(numpy and scipy), XML processing and internet programming, random simulation and GUI programming.

Course Text and Materials

- Official Python tutorial. Online at: <http://docs.python.org/tutorial/index.html>
- *Dive into Python 3*. Online at: <http://diveintopython3.org/>
- Langtangen, H. P. (2009). *A Primer on Scientific Programming with Python (2nd ed.)*. Springer. Online Access from the library.

- Langtangen, H. P. (2005). *Python Scripting for Computational Science (2nd ed.)*. Springer. Online Access from the library.

note: The above material is suitable for students whom have had programming experience before. If you want a more “Computer Science” oriented introduction, I recommend

- *Think Python: How to Think Like a Computer Scientist*, at <http://www.greenteapress.com/thinkpython/html/index.html>
- John M. Zelle. *Python Programming: An Introduction to Computer Science*.

Grading Components

The instructor reserves the right to change grading structure as necessary. The total grade point is 50. Grades in this course will be based on the activities/components shown below.

Percent	Activity
30	Course Assignments
15	Project
5	Participation

The final letter grade will be given roughly based on your points. There will be four grades: A (41-50), B(30-40), B-(25-31) and C(rest) .

Class Attendance

The instructor will not check attendance in every session. There might be several random graded or ungraded quizzes for checking participation. The instructor assumes that you will be prepared and willing participate in discussions and course activities. You should also be respectful of the instructor and your classmates.

Academic Honesty Policy

Academic dishonesty of any kind will be dealt with in a manner consistent with the Rackham graduate school’s policy on Academic Integrity. You are expected to know and abide by this policy. For more information, see http://www.rackham.umich.edu/policies/academic_and_professional_integrity/statement_on_academic_integrity/

Project Information

Instead of final exams, students are required to conduct a project using python as the primary programming language, and give a presentation at the end of the course. The goal of this project is to prompt students to actively explore python language and tools for their research.

Requirement

0. You are free to use python to solve a data analysis problem from your area of study, demonstrate an important python programming technique under-explored in the class, or it could be just any programming task that might be useful for your future research.
1. This project should involve decent amount of coding (≥ 250 lines, counting code for test cases)
2. Code should be well tested and documented.
3. You are strongly encouraged to come up with an idea that is not completely boring.
4. Here a list of topics that might help you start. cc
 - Object-oriented design and design patterns applied to simplify a complex problem. Example: simulating network traffic or a blackjack game.
 - An important aspect of python programming features not covered in the class. Example: more extensive GUI programming using wxpython or QT, distributed and parallel programming to speed up your programs; interfacing python with C/Java/R code through ctypes, scipy, native interface or swig.
 - Introduction of an important python toolbox from your area of study, and demonstrate its use through examples. Example: Orange for data mining, Pybrain for machine learning, NLTK for natural language processing, Python imaging Library or ITK/VTK for computer vision, Biopython for genome sequencing etc. Please use real-world data if possible. You are encouraged to work on existing source code from a tutorial or a project, however you need to identify what contribution and improvement you will make. See <http://wiki.python.org/moin/NumericAndScientific> for other ideas.
 - You should focus on the scientific novelty, the computational challenge or the visualization of your data and results.
5. You can work individually or with 1-2 fellow students in the class.
6. Your presentation will be graded by your classmates and the instructor. The grading criteria will be posted later.

Components

0. A one-paragraph proposal of what you want to do with your project (due on 09/21) (1)
1. A 12 minute presentation that summarizes your work (5).
2. A 3-page summary (excluding references) (5).
3. Your python source code (4).