Spring Semester 2019
March 11th to May 11th
Monday and Wednesday nights from 5:00pm - 6:15pm

OPIM 244:
Mgmt of Software Development in Python

Build your own apps!

Businesses can earn revenue by delivering software products and services to customers. In this course, students will explore business models and best practices involved in the production and management of application software. In addition to building upon previous Python programming projects, students will propose and manage their own application software projects from ideation to delivery. By the end of the semester, students should emerge with marketable technology and product management experience.

For more information, contact Prof. Rossetti at mjr300@georgetown.edu.

Mgmt of Business Application Development in Python

University:	Georgetown
School:	McDonough School of Business
Program:	Operations and Information Management (OPIM)
Course:	Management of Business Application Development in Python (244)
Credits:	1.5
Prerequisites:	N/A

Description

Businesses can earn revenue by delivering software products and services to customers. In this course, students will explore business models and best practices involved in the production and management of application software. In addition to building upon previous Python programming projects, students will propose and manage their own application software projects from ideation to delivery. By the end of the semester, students should emerge with marketable technology and product management experience.

Learning Objectives

1. Discuss revenue models and distribution models related to the production and consumption of application software.
2. Follow software development best practices like version control and automated testing, and discuss their business implications.
3. Participate in all phases of the system development lifecycle, including: planning, analysis, design, implementation, and maintenance.
4. Discuss the business impacts of software licensing, open source software, and crowdsourcing.
5. Discuss security, privacy, and ethical considerations relevant in designing and managing computer-based information systems.
6. Expand upon an online portfolio of programming projects.
7. Have fun!

Learning Community

Students

This undergraduate business school elective has a maximum enrollment of 45 students.

Professor

Adjunct Professor Michael Rossetti, a professional data scientist and software developer, will be teaching this course. Students should feel free to direct questions to the professor by sending a Slack direct message to @prof-rossetti or an email to mjr300@georgetown.edu. If emailing, all parties should use university-issued addresses. The professor aims to respond to messages within around one to three business days.

When sending announcements and replying to students, the professor may send messages outside of normal business hours. There is no expectation for students to keep the same schedule. Students should feel free to read and reply to messages at whatever time is most preferable for them!

The professor will hold office hours at least once per week (see Canvas Calendar), as well as by appointment.

Graduate Assistant

Math and statistics graduate student Hiep Nguyen will be providing educational and operational assistance, focusing on students with Windows computing environments. Students should feel free to direct questions to Hiep by sending a Slack direct message to @hiepnguyen034 or an email to hxn3@georgetown.edu. If emailing, all parties should use university-issued addresses.

Hiep will hold office hours by appointment.

Materials

Texts

There are no required texts, but students are encouraged to consult online resources such as:

- The Lean Startup, by Eric Ries
- Rework, by Jason Fried and David Heinemeier Hansson
- Pro Git, by Scott Chacon and Ben Straub
- Git Documentation (Git-scm.com)
- Git and GitHub Learning Resources (GitHub.com)

Computers

Each student is encouraged to configure their own personal computer with a Python development environment (see “Software” section below) at the beginning of the semester as instructed by the professor, and to bring this computer to each class. Mac and Windows operating systems should each provide a suitable development environment. However chromebooks, tablets, and netbooks may prove problematic and previous students have advised against them.

Any student who doesn't have access to a compatible personal computer during class may inquire about loaning a laptop from the technology center or otherwise consult with the professor for more guidance within the first week of enrolling. Lab computers in the library are equipped with many of the tools necessary to complete course deliverables, and further accommodations may be made as necessary.

Software

This course will introduce students to a standard set of software development tools, including: a development-class text editor like *VS Code*, *Atom*, *Sublime*, or *Notepad++*; and command-line utilities such as *Anaconda*, *Python*, *Pip*, and *Git*. Students should WAIT to install these programs until instructed by the professor, in the manner prescribed.

Students who wish to use their own preferred tools (like alternative text editors or IDEs) may do so as long as they are able to meet course expectations, but they should be aware the instructors may not be able to provide the same level of support for those tools.

Operations

Canvas

All registered students should have access to the Canvas learning management platform. The Canvas Calendar contains the most up-to-date information about the scheduling of class sessions, office hours, and deliverable due dates. And the Canvas Gradebook will contain all student grades.

GitHub

GitHub is the leading online platform for sharing software and code-related resources. The course GitHub repository is the primary source for course materials, including programming language references, instructional exercises, and project descriptions.

Students will also use GitHub to submit project deliverables. When instructed by the professor, and in the manner prescribed, students should create a GitHub account as necessary and share their GitHub username with the professor to associate themselves with their work product. Deliverables will be submitted in public repositories by default. Student GitHub usernames and profiles need not contain any personally-identifiable information, but any student who desires additional privacy should consult with the professor within a week of enrolling, and the professor will recommend certain alternatives such as submission via private repositories.

Slack

Slack is an integrated chat platform that will help facilitate course communications. When invited by the professor, students should join the course Slack organization.

The professor will share code snippets and helpful links in the #opim-244 channel on a regular basis, and students are encouraged to ask questions and help each other in that channel as well. The professor will share links to class recordings in the #opim-244-videos channel, and may create additional channels as applicable to serve assignment-specific purposes or facilitate group communications.

Reference: Emoji Cheat Sheet 😃

Evaluation

Student performance will be evaluated through submission of six weekly progress check-in surveys (12% total), updating and improving three previous projects (30% total) and proposing and implementing a self-directed project (58%). Students should consult the Canvas Assignments page for a comprehensive list of deliverables, weights, and due dates.

Extra Credit

Additionally, the professor may award extra credit “Engagement Points” to be applied towards the final exam. Activities eligible for engagement points include, but are not limited to:

- Contributing meaningfully to classroom discussions.
- Volunteering programming solutions and strategies during class.
- Displaying meaningful effort and progress in class and/or during office hours.
- Helping other classmates (in “reasonable” ways) in class, during group office hours, or in the course Slack channel.
- Contributing new and/or improved class materials by proposing changes to the course repository and getting those changes accepted. Includes everything from fixing typos, to clarifying language, to adding context, to creating new reference documents altogether.
- Delivering projects which go above and beyond the minimum requirements in impressive and inspiring ways, while still meeting all of the minimum requirements.

Self-Directed Project

The Self-Directed Project provides students with the flexibility to follow their own interests by proposing and implementing their own application software. First students will brainstorm and submit a proposal outlining their project’s scope, objectives, and requirements. Then students will implement the requirements by writing their own Python program. The final project deliverable will include not only the software itself, but also accompanying documentation and automated tests.

Proposal Phase

During the project proposal phase, students will define project scope and objectives, including information inputs and outputs, and submit this information to the professor for approval. After reviewing the proposals, the professor may offer suggestions to help refine project focus, share helpful resources, and/or provide other guidance to help students succeed.

Implementation Phase

During the project implementation phase, students will write application software in Python. The software must transform information inputs into information outputs to achieve stated objectives as outlined in the project proposal. The software should strive to demonstrate a unique set of functionality which differentiates it in some significant way from other examples and student submissions.

Schedule

The schedule is tentative and may change to reflect actual pace of instruction. In the event of a major schedule change, the professor will likely send an announcement.

Unit 8: Delivery and Distribution

Day	Date	No.	Topics / Focus	Activities / Deliverables
Mon	Mar-11	1	Software Products and Services; Business Models	OPIM 243 Exam Review; Development Environment Setup;
Wed	Mar-13	2	Delivery and Distribution Models; The *git* CLI, The *heroku* CLI	"Deploying to Production" Exercise

Unit 9: Maintenance and Quality Control

Day	Date	No.	Topics / Focus	Activities / Deliverables
Mon	Mar-18	3	Code Refactoring and Simplification	"Codebase Cleanup" Exercise
Wed	Mar-20	4	Automated Testing; The *pytest* Package	"Testing 1,2,3" Exercise
Mon	Mar-25	5	Continuous Integration (CI)	"Continuous Integration" Exercise

Unit 10: Planning and Design

Day	Date	No.	Topics / Focus	Activities / Deliverables
Wed	Mar-27	6	The Systems Development Lifecycle (SDLC); System Planning and Analysis	"Freestyle" Project Proposal
Mon	Apr-01	7	System and Product Design; Design Thinking; Requirements Gathering; User Stories	"Freestyle" Project Proposal
Wed	Apr-03	8	System and Process Diagramming	"Freestyle" Project Proposal
Mon	Apr-08	9	[Project Planning and Investigation]	"Freestyle" Project Proposal

Unit 11: Implementation Sprint

Day	Date	No.	Topics / Focus	Activities / Deliverables
Wed	Apr-10	10	[Project Development and Testing]	“Freestyle” Project Deliverables
Mon	Apr-15	11	[Project Development and Testing]	“Freestyle” Project Deliverables
Wed	Apr-17	12	[Project Development and Testing]	“Freestyle” Project Deliverables

Unit 12: Recap

Day	Date	No.	Topics / Focus	Activities / Deliverables
Mon	Apr-22	N/A	Easter Break (no class)	N/A
Wed	Apr-24	13	[Project Demonstrations]	“Freestyle” Project Demos
Mon	Apr-29	14	[Project Demonstrations]	“Freestyle” Project Demos

Policies

Attendance

All students are encouraged but not required to attend class in-person. If not able to attend class in-person, students are still expected to review the assigned course materials, view the audiovisual class recordings, stay apprised of the schedule of deliverables, and participate in remote communications in Slack as applicable.

Instructional Continuity

If for any reason a class session is not able to be held in-person (e.g. due to inclement weather) or the professor is not able to attend in-person (e.g. due to illness), the professor will announce a specific instructional continuity plan, which may involve remote instruction or Graduate Assistant-led instruction.

Late Submissions and Extensions

Late submissions are generally not accepted. However there are a few exceptions to this rule.

First, although deliverables are generally due by 11:59pm on their respective due date, if a deliverable is submitted after the deadline but before an instructor has begun the grading process (e.g. a project submitted at 2am before the instructor wakes up around 10am to start grading the next day), the instructor is encouraged to consider that submission as being "on-time" and to include it in the list of deliverables to evaluate.

Second, students may request an "Advanced Notice Due Date Extension" by emailing the professor at least 72 hours in advance of the original due date. In their request, students should describe the reason for their request and propose an alternative due date. Example reasons include work travel, interviews, and family obligations. Students should expect to submit deliverables on time unless the professor explicitly approves their extension request in writing, in which case an alternative due date will be agreed upon and late penalties will be waived.

Finally, students may request a "Short Notice Due Date Extension" by emailing the professor anytime within the 72 hours immediately preceding the due date. In their request, students should describe the reason for their request and propose an alternative due date as well as a proposed late penalty. Example reasons include time mismanagement and last-minute emergencies. Students can expect to have their due date extension approved, and depending on the specific timeline and circumstances of the request, a modest late penalty may be applied. This option is to be used as an academically honest alternative to what would otherwise be a temptation to commit an academically dishonest action (see academic integrity section below).

Code of Conduct

Students should abide by all policies set forth by the university's Office of Student Conduct.

Academic Integrity

Students are expected to follow the university's Honor System, as well as the policies set forth here. For progress check-ins and the exam, student collaboration is strictly prohibited. However for project deliverables, collaboration is natural and helpful. The following guidelines define the nature of acceptable collaboration on project deliverables. They have been adapted from Harvard CS50's Academic Honesty Policy, which centers around examples of "reasonable" vs. "unreasonable" activities.

To summarize:

1. You may help others, but you may not do their work for them.
2. You may ask others for help, but you may not ask them to do your work for you.
3. Your work product must materially originate from you and you alone (except in some "reasonable" exceptions noted below, which must in all cases be appropriately cited and attributed).

Below are non-comprehensive lists of example activities which can be considered "reasonable" or "not reasonable", respectively. Any further questions about what constitutes an academic integrity infraction should be proactively directed to the professor. Instructors are required to report any suspected violation of academic integrity, and violations may lead to serious consequences such as failure or dismissal.

Reasonable / Acceptable

- Accessing and/or sharing links to publicly-available online resources, including documentation, tutorials, blog posts, video walkthroughs, etc., as well as course materials, exams, and project descriptions, even from past semesters.
- Accessing and reviewing for the purpose of gaining general inspiration or strategic guidance any deliverables submitted publicly by students past or present.
- Discussing concepts, strategies, and techniques in a human language, in pseudocode, or via illustrations and diagrams.
- Asking someone else for help debugging your program, and sharing a few lines of your code with them to support this purpose.
- Helping someone else debug their code, either in person or remotely, by viewing and/or executing it; and sharing with them general guidance, links to public resources, or a snippet or few lines of code.
- Searching for or soliciting help with general programming techniques and approaches (e.g. Googling "*how to read a CSV file in Python*", reviewing publicly-available

deliverables from past semesters, working with a tutor, etc.), and incorporating a few lines of resulting code into your deliverable, as long as you accompany the code with specific attribution citations (a code comment consisting of either a URL or email address to identify the information source). NOTE: citations are encouraged, but not required, for information originating directly from class instruction, course materials, or the official Python language documentation.

Unreasonable / Violations

- Accessing deliverables submitted privately by other students, past or present.
- Basing one's own deliverable materially off of any other, including but not limited to publicly-available submissions from past semesters, even if attribution is present.
- Decompiling, deobfuscating, or disassembling the work product of any other student, past or present, even if attribution is present.
- Failing to properly attribute any line of code originating from a source which requires attribution (see "Reasonable" guidance above).
- Sharing more than a few lines of code from your solution with a classmate if they need help. This includes a prohibition on sending, receiving, downloading, or otherwise sharing entire files of code.

Learning Accommodations

Any student requiring learning accommodations, such as longer exam periods, should register and coordinate through the university's Academic Resource Center within a week of enrolling.

Acknowledgement and Authorization

Class sessions will be recorded via a university-issued platform called Panopto. Links to recordings will be shared with students and may also be shared more publicly (e.g. in the course repository). Students should be aware that audiovisual class recordings may include their likeness, name, and/or voice. Any student who would like to opt-out of class recordings may do so by emailing the professor within the first week of enrolling, and the professor will suggest some reasonable accommodations, which may include sitting in designated areas, adopting a nickname during class discussions, etc.