NYU Stern Graduate Summer Session #1
May 22 to July 3, 2019
Monday and Wednesday nights from 6-9pm

Programming in Python & Fundamentals of Software Development (INFO-GB 2335)

Learn how to code!

Today's businesses rely on application software to perform operations, aid decision-making, and drive competitive advantage. In this course, students will learn how to write practical business applications in the Python programming language. No prior programming experience is required. Students will also explore business models and best practices involved in the production and management of application software. Throughout the semester, students will be immersed in hands-on Python programming projects and should emerge with marketable technology skills.

For more information, contact Professor Rossetti at mrossett@stern.nyu.edu.

Programming in Python & Fundamentals of Software Development (INFO 2335)

University:	New York University
School:	Stern School of Business
Department:	Information, Operations, and Management Sciences (IOMS) / Info Systems (INFO)
Course:	Programming in Python & Fundamentals of Software Development (2335)
Section:	Mondays and Wednesdays from 6pm to 9pm
Credits:	3
Prerequisites:	N/A

Description

Today's businesses rely on application software to perform operations, aid decision-making, and drive competitive advantage. In this course, students will learn how to write practical business applications in the Python programming language. No prior programming experience is required. Students will also explore business models and best practices involved in the production and management of application software. Throughout the semester, students will be immersed in hands-on Python programming projects and should emerge with marketable technology skills.

Learning Objectives

1. Learn how to write, debug, execute, and test command-line applications in the Python programming language.
2. Create application software to serve customers and automate business processes.
3. Identify and discuss revenue models and distribution models related to the production and consumption of application software.
4. Understand 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. Follow software development best practices like version control and automated testing, and discuss their business implications.
7. Gain marketable programming skills and build an online portfolio of projects.
8. Have fun!

Community

Students

This course has a maximum enrollment of 50 students. Most are graduate business school (MBA) students, but enrollment is open to students from other NYU graduate schools and programs as well.

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 mrossett@stern.nyu.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!

Teaching Fellow

Supriya Jha will be providing instructional assistance as the course's Teaching Fellow (TF). Students should feel free to direct questions to the TF by sending a Slack direct message to @Supriya or an email to sj2685@nyu.edu.

Materials

Texts

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

- Python Documentation (Python.org)
- Python Tutorial (Python.org)
- Pro Git, by Scott Chacon and Ben Straub
- Git Documentation (Git-scm.com)
- Git and GitHub Learning Resources (GitHub.com)

For additional context, students are encouraged to reference best-selling books such as:

- The Lean Startup, by Eric Ries
- Rework, by Jason Fried and David Heinemeier Hansson
- The Design of Everyday Things, by Don Norman

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 library or otherwise consult with the professor for more guidance within the first week of enrolling.

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

NYU Classes

All enrolled students should have access to the course site in NYU Classes. The course calendar in NYU Classes is the most up-to-date source of information about the scheduling of class sessions and deliverables. The professor will also send announcements through NYU Classes and distribute all grades through the NYU Classes gradebook.

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 **#2335** 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 **#2335-media** channel, and may create additional channels as applicable to serve assignment-specific purposes or facilitate group communications.

Reference: Emoji Cheat Sheet 😃

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.

Day	Date	Unit	Topic(s)
Wed	May 22	1	● Information Systems and Application Software ● Software Products and Services ○ Business and Revenue Models ○ Delivery and Distribution Models ○ Licensing Models, Intellectual Property, Crowdsourcing and Open Source ● Version Control: ○ *GitHub* and *GitHub Desktop* ○ Forking and Cloning Repositories ● Command-line Interfaces / Utilities (CLIs): ○ Navigating and Managing the Filesystem ○ Creating and Activating *Anaconda* Virtual Environments ○ Installing Python Packages with *Pip* ○ Executing *Python* Scripts ○ Running Tests with *Pytest*
Mon	May 27	N/A	Memorial Day (No Class)
Wed	May 29	2	● User Experience and Interfaces (UI/UX) ● Python Interface Capabilities: ○ Graphical User Interfaces (GUIs) with the *Tkinter* and/or *PySimpleGUI* Packages ○ Web Interfaces with the *Flask* Package ○ Voice Interfaces with the *SpeechRecognition* Package ○ Background Services: ■ Sending Emails with the *SendGrid* Package ■ Sending Text Messages (SMS) with the *Twilio*

			Package ■ Sending Tweets with the *Tweepy* Package
Mon	June 3	3	● Anatomy of a Python Code Repository ● Privacy and Security: Managing Environment Variables ● Object-oriented Programming (OOP) ● Python Programming: ○ Printing, Commenting, Debugging ○ Datatypes and Classes ○ Variables and Functions ○ Conditional Logic ○ Modules and Packages
Wed	June 5	4	● Python Programming: ○ Capturing and Validating User Inputs ○ Looping, Sorting, Mapping, and Filtering Lists ○ Processing Data In-memory ● Version Control: ○ Creating and Managing Repositories ○ Creating and Viewing Versions (a.k.a. “Commits”) ○ Syncing Local and Remote Repositories (“Pulling”, “Pushing”)
Mon	June 10	5	● Datastores (with a Focus on CSV) and Program-Data Independence ● Python Programming: ○ Reading and Writing Text Files ○ Processing CSV Data with the *CSV* Module ○ Processing CSV Data with the *Pandas* Package ○ Processing SQL Data with the *PyMySQL* and/or *PsycoPG* Packages ○ Processing Google Sheets Data with the *GSpread* Package
Wed	June 12	6	● Networks, the Internet, HTTP, and Application Programming Interfaces (APIs) ● Python Programming: ○ Issuing HTTP requests with the *Requests* Package ○ Processing JSON data with the *JSON* Module ○ Processing HTML pages with the *BeautifulSoup* Package ○ Automated Web Browsing with the *Selenium* Package
Mon	June 17	7	● The Systems Development Lifecycle (SDLC) ● System Analysis and Design: ○ Data Flow Diagramming (DFDs) ○ Business Process Analysis and Diagramming ○ Design Thinking, User Needs, Product-Market Fit, and Minimum Viable Products (MVPs) ● [Design Thinking Workshop]
Wed	June 19	8	● System Maintenance and Quality Control:

			○ Code Refactoring and Simplification ○ Automated Tests ○ Continuous Integration with *Travis CI* ● Python Programming: ○ Testing with the *Pytest* Package ● [Project Support]
Mon	June 24	9	● Software Deployment Environments: ○ Managing Servers and Deployments with *Heroku* ○ Managing Serverless Deployments with *Google Cloud Functions* ● [Project Support]
Wed	June 26	10	● [Final Exam Period] ● [Project Support]
Mon	July 1	11	● [Project Demonstrations] ● [Project Support]
Wed	July 3	12	● [Project Demonstrations] ● [Project Support]

Evaluation

The NYU Classes Calendar is the authoritative source of information about due dates and weights of all items due for evaluation. However the table below provides a tentative outline.

Deliverable	**Weight**	**Due**
Onboarding Survey	2.5%	May 22 by 5:00pm (before the first class starts)
Weekly Check-ins	(5 * 1.5%) = 7.5%	Once per week: sometime between when the last class of the week is dismissed, and Friday 11:59pm
“Shopping Cart” Project	12.5%	June 16 by 11:59pm
“Robo Advisor” Project	12.5%	June 23 by 11:59pm
“Freestyle” Project Proposal	2.5%	June 23 by 11:59pm
Final Exam	30%	During class on June 26
“Freestyle” Project Demonstration	7.5%	During class as scheduled on July 1 or July 3
“Freestyle” Project Implementation and Documentation	25%	July 3 by 11:59pm

The professor aims to evaluate submissions and return grades within around seven days from the due date, and may utilize graduate assistants during the grading process. Any student who has a question or concern about a grade should ask the professor in writing within seven days of receiving the grade, and the professor will look into the matter in a timely manner.

The professor reserves the right to award extra credit in recognition of valuable student participation and deliverables which exceed expectations.

Projects

Transaction Processing System

The Transaction Processing System (a.k.a. “Shopping Cart” Project) will facilitate a local corner grocery store’s customer checkout process. Students will write an interactive application which prompts an employee to scan grocery items, then calculates the total amount due and either prints or sends an email with an itemized receipt. The software may interface with a real-life laser barcode scanner.

Decision Support System

The Decision Support System (a.k.a. “Robo Advisor”) will provide an investment firm with the capability to automate its financial advisory processes. Students will create an application to generate stock market trading recommendations based on real-time historical stock market data from the Internet, and user inputs such as risk preferences. The system will issue HTTP requests and parse JSON-formatted responses, and may send price movement alerts via SMS.

Self-Directed Project

The Self-Directed (a.k.a. “Freestyle”) Project provides students with the flexibility to follow their own interests by proposing and ultimately 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 version history, documentation, and automated tests.

Proposal Phase

During the project proposal phase, students will define project scope and objectives 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 should 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 potential student submissions. If building upon an example project, the software should strive to differentiate itself from the example and/or add upon the example in a significant way.

Final Exam

The Final Exam is designed to evaluate student knowledge of Python programming concepts, software best practices, and technology management concepts. The exam will be administered during a 90-minute portion of the designated final exam period. The exam will be administered in paper format, so students should remember to bring a pen or pencil.

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).

Learning Accommodations

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

Code of Conduct

All members of the learning community should at all times abide by the university's Code of Ethical Conduct and the school's Code of Conduct.

Academic Integrity

Students are expected to follow the university's Academic Integrity Policy as well as those 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 advised 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.

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.