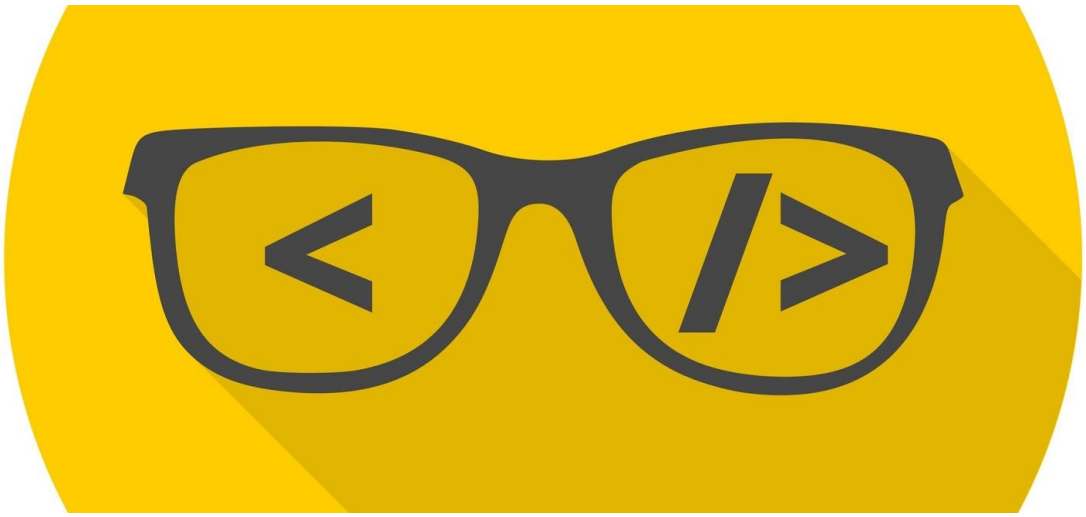
NYU Stern Graduate Summer Session #1
May 16th to June 30th, 2018
Monday and Wednesday nights from 6-9pm in KMC 460

Programming in Python & Fundamentals of Software Development (INFO-GB 2335-60)



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 2335-60)

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 in Kaufman Mgmt Center 460 (60)
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. Write, debug, and execute command-line applications in the Python programming language.
2. Create application software to perform 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 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!

Materials

Texts

Students are encouraged to leverage online resources and documentation such as:

- Python Documentation (Python.org)
- Python Tutorial (Python.org)
- Python Essential Training (Lynda.com)
- Git Documentation (Git-scm.com)
- Git and GitHub Learning Resources (GitHub.com)

Students may optionally reference technology books such as:

- Think Python (2e), by Allen B. Downey
- Pro Git, by Scott Chacon and Ben Straub

For additional context, students may optionally 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 should have access to a personal computer during class. Any student who doesn't own a personal computer may inquire about loaning a laptop from the library.

The computer should allow installation of Python, third-party packages and command-line utilities, and other software such as a text editor. Both Mac and Windows operating systems should provide a suitable development environment, but students with Windows computers may face additional and/or alternate instructions.

Students should NOT install Python or any other programs until instructed to do so by the professor.

Text Editor

Each student is expected to achieve proficiency using a text editor of choice. Ideally, the text editor should include syntax auto-completion functionality for the Python language. The official text editor for this course is Atom, however Sublime and Notepad++ are popular alternatives.

If using Atom, students should be able to download third-party packages which provide additional functionality from the "Settings > Install" menu, and view and configure installed packages from the "Settings > Packages" menu. One recommended package to install is called "Sublime-Style-Column-Selection", which enables vertical text selection.

Git Client

Each student is expected to achieve proficiency using a Git client of choice. By the end of the semester, students should strive to be interfacing with Git via the Git command-line utility, however students originally unfamiliar with Git may optionally start by using a Git GUI application like the GitHub online platform or the GitHub Desktop application before gradually transitioning to the Git command-line utility.

Python

Each student is expected to achieve proficiency executing Python scripts via the Python command-line utility and managing Python packages and versions using the Pipenv command-line utility.

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 of course materials, including programming language references, instructional exercises, and project descriptions.

Additionally, students will be expected to submit deliverables through GitHub, except when otherwise instructed.

Slack

Slack is a chat platform that will be used to share code snippets, links to helpful materials, and other incidental course communications. All students should join the course Slack organization at the beginning of the semester when invited by the professor.

Students are encouraged to post questions and answers in the discussion channel (**#2335**), to monitor the media channel (**#2335-media**) for links to class recordings, and to discuss upcoming events and opportunities for industry networking in the **#events** channel. Students may optionally join the **#2335-dev** channel to subscribe to a news feed of updates to the course repository. The professor 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.

Module I: Python Programming

Day	Date	Unit	Topic(s)
Wed	May 16	1	Intro to Information Systems and Application Software; Security, Privacy, and Ethics; Command-line Computing; Python Development Environment Setup
Mon	May 21	2	Intro to Object Oriented Programming; Basic Python (variables, functions, basic datatypes, etc.); Python Programming: processing user inputs and data in memory
Wed	May 23	3	Intermediate Python (lists, dictionaries, loops, classes, etc.); Python Programming: processing data from text files
Mon	May 28	N/A	Memorial Day (No Class)

Module II: Python Applications

Day	Date	Unit	Topic(s)
Wed	May 30	4	Version Control and Basic Git; Licensing, Intellectual Property, and Open Source; Survey of Third-party Python Packages
Mon	June 4	5	Networks, Internet Protocols (e.g. HTTP), and Application Programming Interfaces (APIs); Python Programming: processing data from the Internet
Wed	June 6	6	Databases and Datastores; Python Programming: processing data from databases; Project Support (Lab)

Module III: Management of Software Development

Day	Date	Unit	Topic(s)
Mon	June 11	7	Design Thinking, and the Systems Development Lifecycle (SDLC); Project Planning Support (Lab)
Wed	June 13	8	Quality Control and Automated Tests; Python Programming: testing
Mon	June 18	9	Hardware, Deployment Environments, and Server Management; Basic Heroku
Wed	June 20	10	Project Development Support (Lab)

Recap and Review

Day	Date	Unit	Topic(s)
Mon	June 25	11	Retrospective Exercise; Final Exam Preparation Session
Wed	June 27	12	Final Exam Period

Evaluation

Student learning will be evaluated primarily through three guided Python programming projects (30% total), a self-directed Python programming project (20%), an industry insights assignment (12%), and a final written exam (25%). Students will also receive credit for on-time submission of an onboarding survey and five weekly progress check-in forms (13% total). Students should consult the schedule and the calendar for due dates and weights of all items due for evaluation.

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.

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

Projects

Point-of-Sale App

The Point-of-Sale App provides a business with the capability to operate its checkout process. Students will write an interactive command-line application to “scan” grocery items, calculate a total invoice amount, and “print” an itemized receipt.

Inventory Management App

The Inventory Management App provides a business with the capability to manage its inventory of products. Students will create an interactive command-line application to create, read, update, and delete products. The system will read and write data in comma-separated values (CSV) format.

Stock Recommendation App

The Stock Recommendation App provides a business with the capability to automate its financial advisory processes. Students will create a command-line application to generate stock trading recommendations based on user risk preferences and real live historical stock market data from the Internet. The system will issue HTTP requests and parse JSON-formatted responses. It may also read and write data from a relational database.

Self-Directed Project

The Self-Directed 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 final class period. The exam will be administered in paper format, so students should remember to bring a pen or pencil. Details about the contents of the final exam will be announced during a final exam preparation session.

Policies

Attendance

All students are encouraged to attend class in-person. If not able to attend class in-person, students are encouraged to participate remotely via Slack, if possible.

Any student who is absent from class may risk missing in-class activities and deliverables. There will be no opportunities to make-up in-class assignments or exams due to absence, except with permission from the professor. To obtain permission for planned absences, a student must email the professor within the first week of class. The professor may grant make-up opportunities for unplanned absences only in the event of significant life events, in which case the professor reserves the right to request further documentation.

Instructional Continuity

If for any reason a class session is not able to be held in-person, the professor will implement a customized instructional continuity plan, which may include remote instruction via Slack.

Late Submissions and Extensions

Late submissions are generally not accepted. However, students may request a due date extension in response to extraordinary circumstances. Any student who requests a due date extension should email the professor days in advance of the original due date. Students should expect to submit deliverables on time unless the professor explicitly approves their extension request in writing, in which case late penalties may apply.

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.

Although students are encouraged to work with each other to discuss and solve problems, submission of identical or nearly identical work may be seen as an academic integrity infraction. And although students are encouraged to leverage Internet resources, submission of work product generated by any other person may also constitute an infraction. Furthermore, if one student violates academic integrity policies by submitting the work product of another student, both students may be considered in violation and subject to penalties.

As a rule of thumb, it is each student's responsibility to type and understand every line of code submitted for evaluation. In situations where lines of boilerplate code or shared code are included in a submission, it is the responsibility of the student to accompany such code with one or more lines of "comments" which include a source link or other manner of attribution (e.g. "adapted from source: https://stackoverflow.com/a/36156/670433" or "jjones@stern.nyu.edu suggested this looping strategy"). However, students should know that submissions comprised of significant portions of code obtained in this way, even if properly attributed, may still constitute an infraction.

Any questions about what constitutes an academic integrity infraction should be proactively directed to the professor; retroactive naivete is not acceptable. Violations of academic integrity

will be forwarded to the Academic Integrity Board, and may lead to consequences such as failure or dismissal.

Grading Guidelines

Min Score	Letter Grade	Interpretation
95	A	Excellent
90	A-	Excellent
87	B+	Good
83	B	Good
80	B-	Good
77	C+	Fair
73	C	Fair
70	C-	Fair
67	D+	Satisfies minimum requirements for credit
60	D	Satisfies minimum requirements for credit
0	F	Failing

Acknowledgement and Authorization

Class sessions will be recorded using university-administered systems and equipment. Links to these videos will be shared with students as soon as they become available, and may also be posted to the course repository on GitHub.

Students should be aware that class recordings may include their image, name, and voice. Any student who would like to opt out of class recordings should email the professor within a week of enrolling, and the professor will suggest some reasonable accommodations, which may include sitting in designated areas.