Syllabus

This is a single, concatenated file, suitable for printing or saving as a PDF for offline viewing. Please note that some animations or images may not work.

Description

This module is also available as a concatenated page, suitable for printing or saving as a PDF for offline viewing.

MET CS 300

Introduction to Software Development

This course introduces basic concepts in discrete mathematics, computer systems and programming that are necessary for modern computing systems. It also develops analytic and logical thinking and prepares students to take graduate-level courses in software development degree. This course first reviews the basic concepts in discrete mathematics including logic, sets, functions, relations and combinatorics. Then it discusses the fundamental concepts in computer systems such as computer organization, basic OS concepts, CPU scheduling, memory management, process management and synchronization. Concurrently with the above mathematics and systems studies, programming concepts are introduced and practiced throughout the whole course using Python.

Technical Notes

The table of contents expands and contracts (+/- sign) and may conceal some pages. To avoid missing content pages, you are advised to use the next/previous page icons in the top right corner of the learning modules.

This course requires you to access files such as word documents, PDFs, and/or media files. These files may open in your browser or be downloaded as files, depending on the settings of your browser.

Learning Objectives

The course is designed to prepare students without a technical background in computer science to succeed in graduate courses in the Master of Science in Software Development. Students often ask how completion of the course relates to acceptance into these graduate programs. The department policy is as follows:

> “In making the decision regarding matriculating a student, the Admissions Committee considers the student’s prior academic record and any relevant experience. The Admissions Committee may require some applicants to take CS 300 to better prepare for graduate study in Software Development before making a final matriculation decision. For students who complete CS 300 the Committee also considers each student’s performance in each of the areas of CS 300, such as discrete math, computer systems and programming. If a student has demonstrated that they are ready for graduate study in *each* of these areas, as demonstrated by a combination of prior coursework, professional experience, and their performance in CS 300, then the Admissions Committee will matriculate them into the MSSD program. Simply passing CS 300 does not assure matriculation, though excellent performance in all areas of CS 300 will earn an applicant matriculation into the program.”

For students coming from other programs, this course is a technically oriented introductory survey of modern computing system.

Course Objectives

Upon completion of this class, students are expected to:
For Discrete Math:

- Explain the basic concepts of logic, and solve mathematical problems that involve the laws of logic and the rules of inference.
- Explain the basic concepts of sets, and solve mathematical problems that involve sets and subsets, set operations and laws of set theory
- Explain the basic concepts of combinatorics, and solve mathematical problems that involve permutations and combinations.
- Explain the basic concepts of functions and relations, and solve mathematical problems that involve one-to-one, onto functions and function composition, the product sets and relations, equivalence relations and partitions.

For Computer Systems:

- Convert numbers between binary, decimal and hexadecimal formats, and explain how basic data types (such as integer, real number, character) are represented in the machine.
- Explain how source program are compiled/interpreted, interpreted executed, including the instruction execution cycle and the role of interrupts.
- Identify the he basic components in a computer system. Explain how multiprocessor/multicore CPU affect the system performance. Explain the storage hierarchy and how caching is applied between different layers in the hierarchy.

- Explain basic concepts of an OS. Explain OS kernel and basic protection mechanisms implemented in an OS.
- Describe the process model and basic operations.
- Explain basic CPU scheduling algorithms in an OS.
- Identify the synchronization issues and describe basic solutions.
- Explain virtual address space and the paging scheme.
- Describe basic components and features of distributed systems and networks.

For Programming:

- Describe the basic concepts of programming.
- Explain how algorithms are developed and implemented in high level languages.
- Design, write, and debug python programs that use sequence, selection and repetition statements, primitive data types, strings, lists, tuples and that do I/O and file manipulation.
- Decompose the program into functions and modules
- Explain and apply basic object oriented concepts including classes, objects and inheritance.
- Design, write and debug basic GUI programs.

Course Organization

This course is 7 weeks long: 6 weeks of content, and 1 week for the final exam. Each of 6 weeks includes:

- reading assignments
- online content
- review questions
- graded assignments
- self-test or quiz
- Possibly one discussion question as extra credit

Each assignment is due in 1 week.
The study guide, which precedes each module, lists specific due dates. Assignments and quizzes are due at 6am ET each Tuesday. Review questions and math assignments are optional, but strongly encouraged. The review questions are very similar to the quiz questions. Review questions may be answered as many times as you like, while quizzes are timed and may be taken only once. Math assignment solutions are provided in videos labeled “echo 360” Solutions to information technology assignments and quizzes will be provided after they are graded.
You will see “blocks” of content in the online material that are labeled “Advanced Content.” We have found that some students like additional material beyond what is formally part of the course. Hence, we are in the process of adding such content. You are not responsible for advanced content on the quizzes, assignments, or final exam.

Instructor

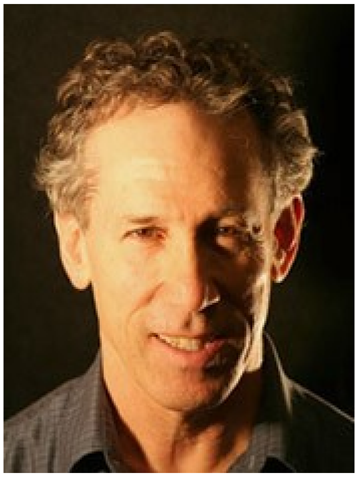
Kuang-Jung (Michael) Huang, Ph.D.

Ph.D, Computer Science, Boston University, USA
MS, Computer Science, Taiwan University, Taiwan
BS, Computer Engineering and Science, Fend-Chia University, Taiwan

Michael has been teaching and working in the computer science area for more than 20 years. He teaches at Boston University Metropolitan College part time while he works at the software industry since 1997. He is currently a senior software developer at SAS. He actively develops software on customer intelligent, inference engine, speech recognition, data mining, and big data. His areas of expertise include artificial intelligence, programming language, and software engineering.

Contact: ghuang@bu.edu

Additional Course Developers



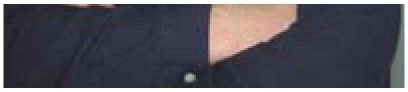
Eric Braude, Ph.D.

Eric Braude received his Ph.D. from Columbia University in mathematics and Master's in Computer Science from the University of Miami. He taught at CUNY and Penn State, followed by twelve years in government and industry as a software engineer, scientist, and manager. He is an Associate Professor of Computer Science at Boston University's Metropolitan College where he has at times held the chairmanship and the acting associate deanship. His research concerns reliable program construction. Eric has written, co-written, or edited six books, including “Software Engineering” and “Software Design.”

http://www.bu.edu/csmet/braude

Anatoly Temkin, Ph.D.

Dr. Anatoly Temkin has been a BU faculty member since 1989. He has taught numerous graduate and undergraduate courses from the math and computer science curricula. He is currently a professor and a graduate student advisor in the Boston University Metropolitan College.

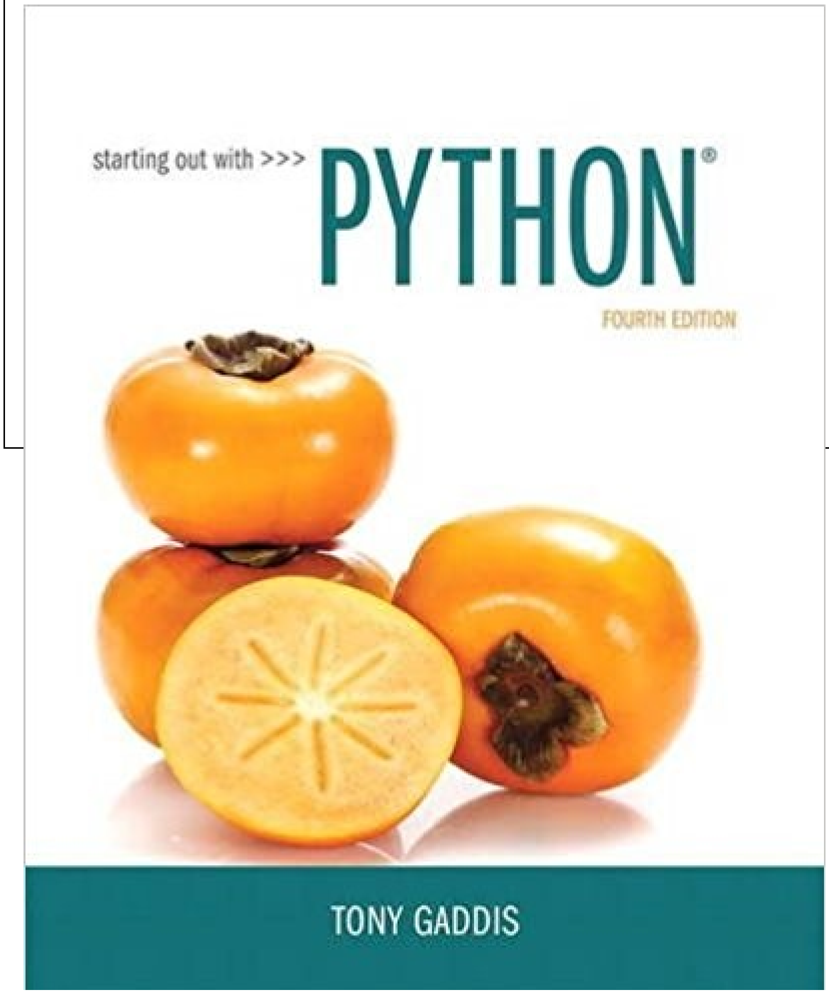


Yuting Zhang, Ph.D.

Dr. Zhang’s current research mainly focus on mobile security, particularly Android security. Her past research focus was in resource management in soft real-time systems, virtual machine systems, and Internet end-systems, though her interest spreads to all areas of computer systems and networks. Conducted through both theoretic analysis and empirical evaluation, her research has resulted in publication in a number of conference proceedings and journals. Zhang served as an assistant professor at Merrimack College, Wentworth Institute of Technology, Allegheny College, and University of Science and Technology Beijing. She has taught a variety of courses, including information technology, Java/C++/C programming, operating systems, computer networks, analysis of algorithms, software engineering, programming languages, and a research seminar.

Materials

Required Book

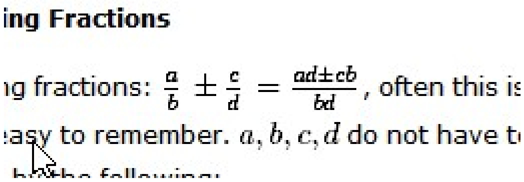


Starting Out with Python (4th Edition)
by Tony Gaddis
Publisher: Pearson
ISBN-10: 0134444329

This book can be purchased from Barnes and Noble at Boston University.

MathJax

Variables, formulae, and equations in this course are rendered using MathJax.



ing Fractions

ig fractions: $\frac{a}{b} \pm \frac{c}{d} = \frac{ad \pm cb}{bd}$, often this i

asy to remember. a, b, c, d do not have t

the following:

To enable its features in your browser, right-click (or ctrl-click on a single-mouse-button Mac) on a variable or equation to see your MathJax settings.

MathJax can be used with the MathPlayer plugin for Internet Explorer, which converts math to speech and highlights the math as it is spoken.

Boston University Library Information

Boston University has created a set of videos to help orient you to the online resources at your disposal. An introduction to the series is below:

met_ode_library_14_sp1_00_intro video cannot be displayed here

All of the videos in the series are available on the Online Library Resources page, which is also accessible from the Campus Bookmarks section of your Online Campus Dashboard. Please feel free to make use of them.

As Boston University students, you have full access to the BU Library. From any computer, you can gain access to anything at the library that is electronically formatted. To connect to the library, use the link http://www.bu.edu/library. You may use the library's content whether you are connected through your online course or not, by confirming your status as a BU community member using your Kerberos password.

Once in the library system, you can use the links under “Resources” and “Collections” to find databases, eJournals, and eBooks, as well as search the library by subject. Some other useful links follow:

Go to Collections to access eBooks and eJournals directly.

If you have questions about library resources, go to Ask a Librarian: Help & FAQs to email the library or use the live-chat feature.

To locate course eReserves, go to Reserves.

Please note that you are not to post attachments of the required or other readings in the water cooler or other areas of the course, as it is an infringement on copyright laws and department policy. All students have access to the library system and will need to develop research skills that include how to find articles through library systems and databases.

Free Tutoring Service

Free online tutoring with SMARTHINKING is available to BU online students for the duration of their courses. The tutors do not rewrite assignments, but instead teach students how to improve their skills in the following areas: writing, math, sciences, business, ESL, and Word/Excel/PowerPoint.

You can log in directly to SMARTHINKING from Online Campus by using the link in the left-hand navigation menu of your course.

YouTube

Please Note

SMARTHINKING may be used only for current Boston University online courses and career services. Use of this service for purposes other than current coursework or career services may result in deactivation of your SMARTHINKING account.

Study Guide

Module 1 Study Guide and Deliverables

Readings:	Module 1 online content Gaddis, Chapter 1: Section 2.3
Lab:	Programming Lab 1 due Saturday, January 26, at 6:00 AM ET
Assignments:	Programming Assignment 1 due Tuesday, January 29, at 6:00 AM ET

Assessments: Math Quiz 1 due Tuesday, January 29, at 6:00 AM ET

Live Classrooms:

- Wednesday,January 23 at 8:30-9:30 PM ET
- Sunday, January 27 at 8:30-9:30 PM ET

Module 2 Study Guide and Deliverables

Readings: Module 2 online content
Gaddis, Chapters 2 and 3

Lab: Programming Lab 2 due Saturday, February 2, at 6:00 AM ET

Assignments: Programming Assignment 2 due Tuesday, February 5, at 6:00 AM ET

Assessments: Math Quiz 2 due Tuesday, February 5, at 6:00 AM ET

Live Classrooms:

- Wednesday, January 30 at 8:30-9:30 PM ET
- Sunday, February 3 at 8:30-9:30 PM ET

Module 3 Study Guide and Deliverables

Readings: Module 3 online content
Gaddis, Chapter 4

Lab: Programming Lab 3 due Saturday, February 9, at 6:00 AM ET

Assignments:	Programming Assignment 3 due Tuesday, February 12, at 6:00 AM ET
Assessments:	Math Quiz 3 due Wednesday, Tuesday, February 12, at 6:00 AM ET
Live Classrooms:	• Wednesday, February 6 at 8:30-9:30 PM ET • Sunday, February 10 at 8:30-9:30 PM ET

Module 4 Study Guide and Deliverables

Readings:	Module 4 online content Gaddis, Chapter 5
Lab:	Programming Lab 4 due Saturday, February 16, at 6:00 AM ET
Assignments:	• Programming Assignment 4 due Tuesday, February 19 at 6:00 AM ET • Systems Assignment 1 due Tuesday, February 19 at 6:00 AM ET
Assessments:	Systems Quiz 1 due Tuesday, February 19 at 6:00 AM ET
Live Classrooms:	• Wednesday, February 13 at 8:30-9:30 PM ET • Sunday, February 17 at 8:30-9:30 PM ET

Module 5 Study Guide and Deliverables

Readings:	Module 5 online content Gaddis, Chapter 6
Lab:	Programming Lab 5 due Saturday, February 23 at 6:00 AM ET
Assignments:	• Programming Assignment 5 due Tuesday, February 26 at 6:00 AM ET • Systems Assignment 2 due Tuesday, February 26 at 6:00 AM ET
Assessments:	Systems Quiz 2 due Tuesday, February 26 at 6:00 AM ET
Live Classrooms:	• Wednesday, February 20 at 8:30-9:30 PM ET • Sunday, February 24 at 8:30-9:30 PM ET

Module 6 Study Guide and Deliverables

Readings:	Module 6 online content Gaddis, Chapters 7, 9, and 12
Lab:	Programming Lab 6 due Saturday, March 2 at 6:00 AM ET
Assignments:	• Programming Assignment 6 due Tuesday, March 5 at 6:00 AM ET • Systems Assignment 3 due Tuesday, March 5 at 6:00 AM

	ET
Assessments:	Systems Quiz 3 due Tuesday, March 5 at 6:00 AM ET
Live Classrooms:	• Wednesday, February 27 at 8:30-9:30 PM ET • Sunday, March 3 at 8:30-9:30 PM ET

Important: Final Exam

You will be responsible for setting up your own appointment with an approved proctoring option. This exam will be three hours in length and will cover material from the entire course. Further information about the testing centers will be forthcoming from the exam coordinator.

Final Exam Details

The Final Exam is a proctored exam available from **Wednesday, March 6, at 6:00 AM ET to Saturday, March 9, at 11:59 PM ET**. The Computer Science department requires that all final exams be proctored.

The exam will only be accessible during the final exam period. You can access it from the Assessments section of the course. Your proctor will enter the password to start the exam.

You will receive a technical support hotline number before the start of the exam. Please bring this number with you to the exam.

Grading Information

Each week will have a Part A--Python--and Part B--Math or Operating Systems. The assignments, quizzes, and labs each week total 12%: 6% for Python programming and 6% Math (for the first three weeks) or Systems (for the second three weeks).

The final is 28% of the total grade. Half of the final will concern Python, a quarter each will concern Math and Systems.

Weekly Python Assignments

The weekly Python assignments count for 95% of the Python programming grade for the week.

Weekly Python Labs

These questions are similar to what you will find on the same week's assignments, and are intended to help you with the associated subject matter so that you can get feedback before your assignment for that week is due. We encourage you to start the lab early each week. You can work on it through the entire week up-until the interim assessment deadline. All labs are weighted equally, and count for 5% of the Python programming grade for the week.

Course Grading

Weekly Python Labs:	0.3% x 6
Weekly Python programming assignments:	5.7% x 6
Weekly Math Quiz:	6% x 3
Weekly System Assignment:	4% x 3
Weekly System Quiz:	2% x 3
Python portion of Final Exam:	14%
Math portion of Final Exam:	7%
Systems Portion of Final Exam:	7%

Translation between letter grades and percentages.

A+ (We can't suggest improvements)	97-100
A (Excellent)	93-96.99
A- (Excellent; minor improvement evident)	90-92.99
B+ (Very good)	87-89.99
B (Good)	83–86.99
B- (Good mostly some significant improvements needed)	80-82.99

C+ (Satisfactory; some significant improvements needed)	77-79.99
C (Satisfactory; significant improvements needed)	73–76.99
C- (Satisfactory; significant improvements required)	70-72.99
D Many improvements required	65
Fail	0

Boston University Metropolitan College