

**BRONX COMMUNITY COLLEGE**  
**Of the City University of New York**  
**DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE**

**SYLLABUS:** CSI 11 Computer Science for Everyone 3 credits / 4 hours  
**PREREQUISITE:** MTH 5 or CUNY Elementary Algebra Proficiency, and ENG 2 and RDL 2, if required  
**TEXT:** How to think like a computer scientist (Python 3) (free online textbook)  
<http://interactivepython.org/runestone/static/thinkcspy/index.html>  
ZyBooks: Computer Science for Everyone (online book)  
<https://learn.zybooks.com/>

**Course Description:** This course presents an introduction to computer science (CS) with an emphasis on problem-solving and computational and algorithmic thinking through “coding”. It aims to present computers as a tool for modeling and solving real-world problems. It will offer an introduction to programming, and students will be exposed to more advanced topics selected from the following partial list: artificial intelligence, robotics, cybersecurity, data science, networking, and neuroscience, conferring an advantage for students considering a CS major. Students majoring in any other discipline will learn how computers can be used to help solve problems in one’s area of expertise.

**Course Goals/Objectives:**

CSI 11 helps to answer questions like “What computer science is?”, “What is computational thinking?”, and “How they can be used to help solve problems?” by introducing students to basic principles of computational thinking.

In this course students will see the basic design of a computer system, how the information is represented and processed. Students will learn to analyze a given problem, design clear, step-by-step solution to the problem; translate this solution into a program; then test and debug it. By the end of the course students will understand the difference between an algorithm and a computer program, and will be able to use, where appropriate, functions; data structures; file and user input/output; decision structures; and loops.

**Student Learning Outcomes (SLO’s):**

- Students will be required to gather information from a variety of sources: the textbook, the internet, and discussion group. Through class discussions students will learn to interpret the collected data as it pertains to presented topic and will be guided to assess the applicability and quality of the data being acquired.
- Students will analyze problems, design an algorithmic solution, and implement that solution into a functioning program via the assigned coding exercises. Throughout the process, students will need to analyze and critique different proposed solutions to the given problems, and they will analyze anomalies (“bugs”) in the process of generating correct code.
- Students will be required to design algorithms and write the programs that implement them. A well written program contains detailed comments to document and justify the choices that students made in their algorithm. In addition, group projects are structured specifically so that students state and justify why a chosen algorithm solves the stated problem via a report or an in-class discussion.
- Students will explore and apply the fundamental concepts in computer science (principles of coding, information theory, artificial intelligence, robotics, data science, and cybersecurity), via coding exercises, group projects, and class discussions. Students will gather information, represent it meaningfully, and use it to solve a posed problem.
- Students will complete weekly group projects in which they will model and solve real-world problems from a variety of fields. They will analyze them using mathematical and formal techniques in order to find an algorithmic solution that can be implemented into a program.

- Students will participate in class discussions on topics from cybersecurity and cryptography, including the impact of digital technologies in issues of privacy, security, and the nature of social structures.

### Grading Policy and Assessment:

Students will be given in-class quizzes or will be asked to submit in-class work (once a week); homework assignments and group project assignments will be given once a week each. All group projects are programming assignments with grading rubric provided for each of them and will be submitted as a program along with the accompanying documentation answering the posed question. All homeworks and group projects have a due date and must be submitted by the due date. In addition, there will be a Midterm Exam and a Final Exam.

Grading for the course will be based on:

- In-class work or in-class quizzes: 10%
- Homeworks: 20%
- Group projects: 20%
- Midterm Exam: 25%
- Final Exam: 25%

**Attendance Policy:** Attendance in class is essential to success in this course. If a student misses a class, it is the student's responsibility to get the material covered in class and all the assignments. There are no make-ups for in-class work nor for in-class quizzes. A student may receive a failing grade for the course if absent more than 6 times (6 times are equivalent to 12 hours).

Week	Topic	Reading
1	History and Basics	<i>ZyBooks: Computer Science for Everyone</i> Chapter 1 <i>History and Basics</i> 1.1 Brief history 1.2 Historical figures in computing 1.3 Computer programs 1.4 Computers all around us 1.5 Computing and careers
	Hardware and Software	<i>ZyBooks: Computer Science for Everyone</i> Chapter 2 <i>Hardware and Software</i> 2.1 Basic hardware 2.2 Cache, memory, drive 2.3 Types of computers 2.4 Common input devices 2.5 Common output devices
	Operating Systems	<i>ZyBooks: Computer Science for Everyone</i> Chapter 3 <i>Operating Systems</i> 3.1 OS basics 3.2 Common operating systems
2	Programming languages	<i>ZyBooks: Computer Science for Everyone</i> Chapter 2 <i>Hardware and Software</i> 2.8 Programming: Machine language 2.9 Programming: Assembly language 2.10 Programming: High-level language

	Basic Input and Output	ZyBooks: <i>Computer Science for Everyone</i> Chapter 4 <i>Introduction to Python 3</i> 4.1 Programming introduction 4.2 Computational thinking 4.3 The Python interactive interpreter 4.4 Programming in Python 4.5 Basic output 4.6 Basic input 4.7 Errors 4.8 Additional practice: Output art 4.9 Development environment
3	Variables and Expressions in Python	ZyBooks: <i>Computer Science for Everyone</i> Chapter 5 <i>Variables and Expressions</i> 5.1 Objects and variables 5.2 Assignments 5.3 More on objects 5.4 Names 5.5 Numeric types: Floating-point 5.6 Expressions 5.7 Module basics 5.8 Math module 5.9 Additional practice: Number games 5.10 Representing text
4	Number Representation  Types	ZyBooks: <i>Computer Science for Everyone</i> Chapter 6 <i>Integer Properties</i> 6.1 Representing information as bits 6.2 Number representation  ZyBooks: <i>Computer Science for Everyone</i> Chapter 7 <i>Types</i> 7.1 String basics 7.2 Lists basics 7.3 Dictionary basics 7.4 Common data types summary 7.5 Additional practice: Grade calculation 7.6 Type conversions 7.7 String formatting 7.8 Numbers in binary 7.9 Additional practice: Health data
5	Propositional Logic  Branching	ZyBooks: <i>Computer Science for Everyone</i> Chapter 8 <i>Logic</i> 8.1 Propositions and logical operations 8.2 Evaluating compound propositions 8.3 Conditional statements 8.4 Logical equivalence 8.5 Laws of propositional logic Logical Puzzles (Rosen, Discrete Mathematics)  ZyBooks: <i>Computer Science for Everyone</i> Chapter 9 <i>Branching</i>

6	Branching	<p>9.1 If-else statement 9.2 Relational and equality operators 9.3 Multiple if-else 9.4 Boolean operators and expressions</p> <p>Book: <i>How To Think Like a Computer Scientist</i> 7.1. Boolean Values and Boolean Expressions 7.2. Logical operators 7.3. Precedence of Operators 7.4. Conditional Execution: Binary Selection 7.5. Omitting the else Clause: Unary Selection 7.6. Nested conditionals 7.7. Chained conditionals 7.8. Boolean Functions</p> <p>ZyBooks: <i>Computer Science for Everyone</i> Chapter 9 <i>Branching</i> 9.5 Membership operators 9.6 Code blocks and indentation 9.7 Conditional expressions 9.8 Additional practice: Tweet decoderet decoder</p>
7	Loops	<p>ZyBooks: <i>Computer Science for Everyone</i> Chapter 10 <i>Loops</i> 10.1 Loops 10.2 While loops 10.3 More while examples 10.4 Counting 10.5 For loops 10.6 Counting using the range() function 10.7 While vs. for loops 10.8 Nested loops</p>
8	Loops	<p>ZyBooks: <i>Computer Science for Everyone</i> Chapter 10 <i>Loops</i> 10.9 Developing programs incrementally 10.10 Break and continue 10.11 Loop else 10.12 Getting both index and value when looping: enumerate() 10.13 Additional practice: Dice statistics</p>
	<i>Midterm Exam</i>	
9	Functions	<p>ZyBooks: <i>Computer Science for Everyone</i> Chapter 11 <i>Functions</i> 11.1 User-defined function basics 11.2 Function parameters 11.3 Returning values from functions 11.4 Dynamic typing 11.5 Reasons for defining functions 11.6 Function with branches/loops 11.7 Function stubs 11.8 Functions are objects</p>

10	Functions	11.9 Functions: Common errors 11.10 Scope of variables and functions ZyBooks: <i>Computer Science for Everyone</i> Chapter 11 <i>Functions</i> 11.11 Namespaces and scope resolution 11.12 Function arguments 11.13 Keyword arguments and default parameter values 11.14 Arbitrary argument lists 11.15 Multiple function outputs 11.16 Help! Using docstrings to document functions 11.17 Engineering examples
11	Strings	ZyBooks: <i>Computer Science for Everyone</i> 20.1 String slicing 20.2 Advanced string formatting 20.3 String methods 20.4 Splitting and joining strings 20.5 The string format method
12	Lists and Dictionaries	ZyBooks: <i>Computer Science for Everyone</i> Chapter 12 <i>Lists and Dictionaries</i> 12.1 Lists 12.2 List methods 12.3 Iterating over a list 12.4 List games 12.5 List nesting 12.6 List slicing 12.7 Loops modifying lists 12.8 List comprehensions 12.9 Sorting lists 12.10 Command-line arguments 12.11 Additional practice: Engineering examples
13	Lists and Dictionaries	ZyBooks: <i>Computer Science for Everyone</i> Chapter 12 <i>Lists and Dictionaries</i> 12.12 Dictionaries 12.13 Dictionary methods 12.14 Iterating over a dictionary
	Files	ZyBooks: <i>Computer Science for Everyone</i> Chapter 13 <i>Files</i> 13.1 Reading files 13.2 Writing files
14	Plotting	ZyBooks: <i>Computer Science for Everyone</i> Chapter 14 <i>Plotting</i> 14.1 Introduction to plotting and visualizing data 14.2 Styling plots 14.3 Text and annotations 14.4 Numpy 14.5 Multiple plots

**Academic Integrity**

Academic dishonesty (such as plagiarism and cheating) is prohibited at Bronx Community College and is punishable by penalties, including failing grades, dismissal and expulsion. For additional information and the full policy on Academic Integrity, please consult the BCC College Catalog.

**Accommodations/Disabilities**

Bronx Community College respects and welcomes students of all backgrounds and abilities. In the event you encounter any barrier(s) to full participation in this course due to the impact of a disability, please contact the disAbility Services Office as soon as possible this semester. The disAbility Services specialists will meet with you to discuss the barriers you are experiencing and explain the eligibility process for establishing academic accommodations for this course. You can reach the disAbility Services Office at: [disability.services@bcc.cuny.edu](mailto:disability.services@bcc.cuny.edu), Loew Hall, Room 211, (718) 289-5874.

Last updated 03/06/2019