# CSI 11 Computer Science for Everyone

**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.

**Prerequisites:** MTH 5 or CUNY Elementary Algebra Proficiency, and ENG 2 and RDL 2, if required

**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.

**Course Credits**: 4 hours, 3 credits

**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).

**Textbook/Resources**:
1) How to think like a computer scientist (Python 3) (free online textbook)
http://interactivepython.org/runestone/static/thinkcspy/index.html

2) ZyBooks: Computer Science for Everyone (online book)
https://learn.zybooks.com/

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

| | | |
|---|---|---|
| | | Chapter 2 *Hardware and Software*<br>2.8 Programming: Machine language<br>2.9 Programming: Assembly language<br>2.10 Programming: High-level language |
| | Basic Input and Output | ZyBooks: *Computer Science for Everyone*<br>Chapter 4 *Introduction to Python 3*<br>4.1 Programming introduction<br>4.2 Computational thinking<br>4.3 The Python interactive interpreter<br>4.4 Programming in Python<br>4.5 Basic output<br>4.6 Basic input<br>4.7 Errors<br>4.8 Additional practice: Output art<br>4.9 Development environment |
| 3 | Variables and Expressions in Python | ZyBooks: *Computer Science for Everyone*<br>Chapter 5 *Variables and Expressions*<br>5.1 Objects and variables<br>5.2 Assignments<br>5.3 More on objects<br>5.4 Names<br>5.5 Numeric types: Floating-point<br>5.6 Expressions<br>5.7 Module basics<br>5.8 Math module<br>5.9 Additional practice: Number games<br>5.10 Representing text |
| 4 | Number Representation | ZyBooks: *Computer Science for Everyone*<br>Chapter 6 *Integer Properties*<br>6.1 Representing information as bits<br>6.2 Number representation |
| | Types | ZyBooks: *Computer Science for Everyone*<br>Chapter 7 *Types*<br>7.1 String basics<br>7.2 Lists basics<br>7.3 Dictionary basics<br>7.4 Common data types summary<br>7.5 Additional practice: Grade calculation<br>7.6 Type conversions<br>7.7 String formatting<br>7.8 Numbers in binary<br>7.9 Additional practice: Health data |
| 5 | Propositional Logic | ZyBooks: *Computer Science for Everyone*<br>Chapter 8 *Logic*<br>8.1 Propositions and logical operations<br>8.2 Evaluating compound propositions<br>8.3 Conditional statements<br>8.4 Logical equivalence |

| | | |
|---|---|---|
| | | 8.5 Laws of propositional logic<br>Logical Puzzles (Rosen, Discrete Mathematics) |
| | Branching | ZyBooks: *Computer Science for Everyone*<br>Chapter 9 *Branching*<br>9.1 If-else statement<br>9.2 Relational and equality operators<br>9.3 Multiple if-else<br>9.4 Boolean operators and expressions |
| 6 | Branching | Book: *How To Think Like a Computer Scientist*<br>7.1. Boolean Values and Boolean Expressions<br>7.2. Logical operators<br>7.3. Precedence of Operators<br>7.4. Conditional Execution: Binary Selection<br>7.5. Omitting the else Clause: Unary Selection<br>7.6. Nested conditionals<br>7.7. Chained conditionals<br>7.8. Boolean Functions<br><br>ZyBooks: *Computer Science for Everyone*<br>Chapter 9 *Branching*<br>9.5 Membership operators<br>9.6 Code blocks and indentation<br>9.7 Conditional expressions<br>9.8 Additional practice: Tweet decoderet decoder |
| 7 | Loops | ZyBooks: *Computer Science for Everyone*<br>Chapter 10 *Loops*<br>10.1 Loops<br>10.2 While loops<br>10.3 More while examples<br>10.4 Counting<br>10.5 For loops<br>10.6 Counting using the range() function<br>10.7 While vs. for loops<br>10.8 Nested loops |
| 8 | Loops<br><br><br><br><br>*Midterm Exam* | ZyBooks: *Computer Science for Everyone*<br>Chapter 10 *Loops*<br>10.9 Developing programs incrementally<br>10.10 Break and continue<br>10.11 Loop else<br>10.12 Getting both index and value when looping: enumerate()<br>10.13 Additional practice: Dice statistics |
| 9 | Functions | ZyBooks: *Computer Science for Everyone*<br>Chapter 11 *Functions*<br>11.1 User-defined function basics<br>11.2 Function parameters<br>11.3 Returning values from functions |

| | | |
|---|---|---|
| | | 11.4 Dynamic typing<br>11.5 Reasons for defining functions<br>11.6 Function with branches/loops<br>11.7 Function stubs<br>11.8 Functions are objects<br>11.9 Functions: Common errors<br>11.10 Scope of variables and functions |
| 10 | Functions | ZyBooks: *Computer Science for Everyone*<br>Chapter 11 *Functions*<br>11.11 Namespaces and scope resolution<br>11.12 Function arguments<br>11.13 Keyword arguments and default parameter values<br>11.14 Arbitrary argument lists<br>11.15 Multiple function outputs<br>11.16 Help! Using docstrings to document functions<br>11.17 Engineering examples |
| 11 | Strings | ZyBooks: *Computer Science for Everyone*<br>20.1 String slicing<br>20.2 Advanced string formatting<br>20.3 String methods<br>20.4 Splitting and joining strings<br>20.5 The string format method |
| 12 | Lists and Dictionaries | ZyBooks: *Computer Science for Everyone*<br>Chapter 12 *Lists and Dictionaries*<br>12.1 Lists<br>12.2 List methods<br>12.3 Iterating over a list<br>12.4 List games<br>12.5 List nesting<br>12.6 List slicing<br>12.7 Loops modifying lists<br>12.8 List comprehensions<br>12.9 Sorting lists<br>12.10 Command-line arguments<br>12.11 Additional practice: Engineering examples |
| 13 | Lists and Dictionaries | ZyBooks: *Computer Science for Everyone*<br>Chapter 12 *Lists and Dictionaries*<br>12.12 Dictionaries<br>12.13 Dictionary methods<br>12.14 Iterating over a dictionary |
| | Files | ZyBooks: *Computer Science for Everyone*<br>Chapter 13 *Files*<br>13.1 Reading files<br>13.2 Writing files |
| 14 | Plotting | ZyBooks: *Computer Science for Everyone*<br>Chapter 14 *Plotting*<br>14.1 Introduction to plotting and visualizing data |

| | | |
|---|---|---|
| | | 14.2 Styling plots<br>14.3 Text and annotations<br>14.4 Numpy<br>14.5 Multiple plots |