Penn Engineering ONLINE LEARNING

Computational Thinking for Problem Solving

Instructors

- Susan Davidson, Weiss Professor, Dept. of Computer & Information Science, University of Pennsylvania
- Chris Murphy, Associate Professor of Practice, Dept. of Computer & Information Science, University of Pennsylvania

Course Description

Computational thinking is the process of approaching a problem in a systematic manner and creating and expressing a solution such that it can be carried out by a computer. But you don't need to be a computer scientist to think like a computer scientist. In fact, we encourage students from any field of study to take this course! Many quantitative and data-centric problems can be solved using computational thinking and an understanding of computational thinking will give you a foundation for solving problems that have real-world, social impact.

In this course, you will learn about the pillars of computational thinking, how computer scientists develop and analyze algorithms, and how solutions can be realized on a computer using the Python programming language. By the end of the course, you will be able to develop an algorithm and express it to the computer by writing a simple Python program.

This course will introduce you to people from diverse professions who use computational thinking to solve problems. You will engage with a unique community of analytical thinkers and be encouraged to consider how you can make a positive social impact through computational thinking.

Course Learning Outcomes

- Apply the pillars of computational thinking to develop a solution for solving a computational problem
- Choose the best solution to a computational problem
- Express an algorithm in a structured manner in terms of basic elements and operations of a modern computer
- Implement a solution to a computational problem using the Python programming language

Intended Audience
This course is intended for learners from all backgrounds who are interested in approaching problems more systematically, developing more efficient solutions, and understanding how computers can be used in the problem solving process.

Course Prerequisites
Familiarity with algebra, trigonometry, mathematical functions, exponents, and logarithms

Module 1: Pillars of Computational Thinking

- Learning Outcomes
  - Describe the four pillars of computational thinking and their benefits
  - Apply the pillars of computational thinking to solving a problem
  - Represent an algorithm using a flowchart
- Topics
  - What is computational thinking? Why is it important?
  - Pillars of computational thinking: decomposition; pattern recognition; data representation and abstraction; algorithms
  - Applying computational thinking to case studies

Module 2: Expressing and Analyzing Algorithms

- Learning Outcomes
  - Describe common algorithms that are used in computational thinking
  - Determine the complexity of an algorithm in terms of the number of operations as a function of the input size
  - Calculate the number of possible solutions to an optimization problem
  - Compare the relative benefits and limitations of common algorithmic approaches to problem solving
- Topics
  - Common algorithms: find-max, linear search, binary search
  - Algorithmic complexity: linear, logarithmic, quadratic, exponential, factorial
  - Approaches to solving optimization problems: brute force, greedy

Module 3: Operations of a Modern Computer

- Learning Outcomes
  - Describe the major components of modern computer hardware and their roles

- Determine the appropriate mechanism for storing data in a computer program
- Determine the appropriate control flow mechanism for a given algorithm
- Convert an algorithm from plain English or a flowchart to pseudocode

- Topics
    - History of the computer
    - Basics of the von Neumann Architecture
    - Data representation and control flow in the von Neumann Architecture
    - Expressing algorithms using pseudocode

Module 4: Applied Computational Thinking using Python

- Learning Outcomes
    - Develop simple Python programs using lists, conditional statements, and loops
    - Develop more complex Python programs using functions and objects
    - Convert an algorithm from pseudocode to Python
    - Debug and fix syntax and runtime errors in simple Python programs
- Topics
    - Data representation: variables, lists
    - Conditionals
    - Loops and Iteration
    - Functions
    - Classes and Objects

Assessment

This course will use a variety of assessments to determine whether learners understand and can apply the key concepts and skills that the course teaches. This includes:

- Knowledge Check quizzes at the end of each video lesson
- Peer graded assessments that relate to using computational thinking to solve problems
- Quantitative problem solving activities in which learners apply techniques and perform computations
- Programming assignments involving the development of short Python programs
- A course-long project that walks learners through the process of applying computational thinking to a problem and implementing the solution in code

To earn a certificate in this course, learners must earn a passing score (70% or above) on all assessments.