AOIT Introduction to Programming

Course Scope and Sequence

July 2021

Introduction to Programming uses the Python programming language paired with a beginner friendly integrated development environment called Thonny to introduce students to basic programming skills. Students learn the principles of programming by comparing Python to other programming languages. The course begins with algorithms, and then it lays a foundation of mastering variables, operators, and control structures. Students use models as a way to quickly solve new problems using knowledge and techniques already learned. After this foundation is established, students learn to design programs and write functions. In addition, students learn program design, documentation, formal debugging, and testing.

Students complete numerous programs in the course, including both text and graphics/animation programs. In their culminating project, students create their own computer game of Tic-Tac-Toe, Pot-Shot, Blackjack, or Shoot-the-Ball, demonstrating all of the programming skills and knowledge they have acquired. Students hone the important skills of problem solving, thinking logically, looking at the big picture, and paying attention to detail. Students also examine career opportunities as system programmers, application programmers, and system engineers, and they consider the education, experience, and skills needed to enter and succeed in programming-related professions.

This course is expected to take a total of 86 50-minute class periods.

Unit 1: Getting Started

Lesson 1: Course Introduction

Estimated # of Class Periods: 2

Learning Objectives

- Infer the skills and knowledge about computer programming needed to be successful in an authentic project
- Evaluate the importance of learning about computer programming in terms of relevant professions
- Identify general computer programming terms with which to build a taxonomy

Lesson 2: Writing a Simple Program

Estimated # of Class Periods: 6

Learning Objectives

- Demonstrate the ability to enter and modify source code statements using the editing and interactive execution capabilities of Thonny
- Develop simple graphics programs using Thonny
- Demonstrate the ability to debug statements, distinguishing between syntactic and semantic errors
- Identify input, process, and output in a program

Lesson 3: Computers and Programming

Estimated # of Class Periods: 4

Learning Objectives

- Differentiate between hardware and software and explain how the two work together
- Explain how and why computer programs translate information into bits and bytes
- Differentiate between an interpreter and a compiler
- Compare and contrast different programming languages
- Describe the characteristics that set Python apart
- Explain the difference between knowing a programming language and knowing how to program

Lesson 4: Program Design and Problem Solving

Estimated # of Class Periods: 6

Learning Objectives

- Create a natural language procedure for solving a problem
- Create an algorithm to solve a programming problem
- Create a program using an algorithm as a planning guide
- Create a program that includes meaningful comments with correct syntax
- Demonstrate the ability to test a programming solution to make sure it meets all stated requirements

Unit 2: Manipulating Data

Lesson 5: Working with Variables

Estimated # of Class Periods: 5

Learning Objectives

- Define the term *variable* and demonstrate how operations and algorithms use variables to store and manipulate data
- Explain how Python handles different variable types
- Create assignment statements for variables using Python rules (for both programmer-assigned and user-assigned values)
- Create a simple Python program with two or more variables
- Demonstrate the ability to debug syntax errors, semantic errors, and runtime errors in statements that include variables
- Compare and contrast how Java, C++, and Python handle variables (use of types, declaring, etc.)

Lesson 6: Using Arithmetic Operators to Manipulate Data

Estimated # of Class Periods: 3

Learning Objectives

- Demonstrate the ability to order and calculate arithmetic operations correctly
- Create a program with statements that use arithmetic operators
- Demonstrate the ability to debug arithmetic operations

Lesson 7: Using String Operators and Methods to Manipulate Data

Estimated # of Class Periods: 5

Learning Objectives

- Create programs that process textual information using string operators and methods (functions)
- Demonstrate the ability to debug string operations

Unit 3: Control Structures

Lesson 8: Simple Conditional Branching and Looping

Estimated # of Class Periods: 11

Learning Objectives

- Explain the use of conditional branching and how it works
- Create if-statements and if/else-statements
- Develop programs using if-statements and if/else-statements
- Explain the use of looping and how it works
- Create for-loops and while-loops
- Develop programs using for-loops and while-loops
- Differentiate between for-loops and while-loops
- Classify statements with relational operators as true or false
- Demonstrate understanding of program flow in a decision-making model
- Create algorithms and code for decision structures that demonstrate understanding of initialization statements, control statements, Boolean expressions, and counter statements
- Demonstrate an ability to debug programs with branches and loops
- Develop test plans appropriate for programs with branches and loops

Lesson 9: User-Defined Functions and Sequences

Estimated # of Class Periods: 6

Learning Objectives

- Define programming functions
- Explain how a function call is executed
- Create programs that use string, list, and tuple methods and functions
- Create programs using indexing
- Create programs that use sequences of type string, list, and tuple

Lesson 10: Advanced Sequence Manipulation

Estimated # of Class Periods: 8

Learning Objectives

- Create programs that use advanced sequence manipulation techniques
- Design, code, test, and debug a complex software project
- Develop data for use in program testing
- Develop and implement a test plan for a complex software project
- Demonstrate understanding of how to work in a team to implement complex software projects

Unit 4: Designing a Program

Lesson 11: Advanced Debugging Techniques

Estimated # of Class Periods: 5

Learning Objectives

- Display understanding of various formal debugging techniques
- Locate errors in and fix graphics programs
- Demonstrate the ability to modify an existing program to add debugging code

Lesson 12: Animation Using Turtle Graphics

Estimated # of Class Periods: 4

Learning Objectives

- Display understanding of various animation techniques, including events and sound
- Demonstrate the ability to add animation to an existing program

Lesson 13: Designing and Implementing a Complex Software Project

Estimated # of Class Periods: 12

Note: This lesson includes the culminating project.

Learning Objectives

- Demonstrate the ability to use software best practices to complete a complex software project
- Demonstrate the ability to organize and successfully implement a complex software project
- Demonstrate the ability to turn a complex problem statement and set of requirements into working code that meets the project objectives
- Demonstrate the ability to plan, design, code, and test a complex software program
- Demonstrate the ability to divide a program into sets of cooperating functions and other blocks of related code
- Create a complex program in Python that contains branching, looping, sequences, and one or more user-defined functions
- Create a test plan for a complex software program and use it to test the program
- Locate and fix errors found during coding and testing using standard techniques

Unit 5: Careers in Programming and Course Closure

Lesson 14: Working in Programming and Learning from Industry Experts

Estimated # of Class Periods: 4

Learning Objectives

- List the types of jobs that are available in the programming industry
- Describe entry-level jobs in the programming industry and corresponding qualifications
- Evaluate which programming-related jobs are most suitable, based on personal interests and skills
- Write an effective letter of inquiry for entry-level jobs or internships in programming

Lesson 15: Project Presentation and Course Closure

Estimated # of Class Periods: 5

Learning Objectives

- Summarize key learning across the whole subject of programming
- Evaluate individual and group performance on a major programming project