Sophia

Sophia Learning

CS1100: Introduction to Python Programming (ACE recommends 3 credits)

COURSE DESCRIPTION

In this course, you will learn the basics of computer programming from data types, to creating classes, to algorithms and testing. You will learn these concepts while diving deep into the syntax of Python as your core programming language. The course culminates in the chance to design and build a project that answers a need or goal that you determine.

COURSE EFFECTIVE DATES: May 2022 - Present

PREREQUISITES: No prerequisites

LENGTH OF COURSE: This is a self-paced course. Students may use as much or as little time as needed to complete the course.

ACE LEARNING EVALUATIONS RECOMMENDATION: In the lower-division baccalaureate category, 3 semester hours in introduction to programming (ACE ID: SOPH-0058).

GRADING: This is a pass/fail course. Students must complete 11 Challenges (formative assessments), 3 Milestones (summative assessments), and 1 Touchstone (project-based or written assessments) with an overall score of 70% or better.

Challenges	Points Possible
Challenge 1.1: Learning to Code	5
Challenge 1.2: Data Types	7
Challenge 1.3: Functions and Conditions	7
Challenge 2.1: Lists	6
Challenge 2.2: Loops	6

Challenge 2.3: Complex Functions	5
Challenge 3.1: Class Basics	5
Challenge 3.2: Inheritance and Scope	5
Challenge 3.3: Modules and Files	7
Challenge 4.1: Planning the Algorithm	5
Challenge 4.2: Coding the Algorithm	5
Total	**63**

Milestones	Points Possible
Milestone 1	54
Milestone 2	51
Milestone 3	51
Total	**156**

Touchstones	Points Possible
Touchstone 4: Python Journal Project	100
Total	**100**

Grand Total	**319**

Touchstones are projects that illustrate comprehension of the course material, help refine skills, and demonstrate application of knowledge. Read further for information on the touchstones in this course:

- **Touchstone 4: Python Journal Project (100 points):** Students will learn how to effectively plan, design, develop, and test an original program of their choosing. This program is their choice and it can be as complex as they wish. This includes planning out the algorithm using pseudocode, coding their program using everything they learned from Units 1-3, and finally testing and debugging their program to make sure it fulfills their intended purpose. Students will fill out a journal template which has five sections that correspond to the five steps they will complete for their final project.

For more general information on assessments, please visit the Student Guide located on your course dashboard.

LEARNING OUTCOMES

Upon completion of the course, the student will be able to:

1. Demonstrate the use of basic data types, conditional statements, and functions in programming.
2. Use advanced data structures, iteration and complex functions in programs.
3. Write classes with attributes and methods while using external modules and files for added functionality.
4. Produce an original program from a planned algorithm to a coded, tested, and commented final project.

COURSE COMPETENCIES AND TOPICS

Unit	Competencies	Major Topics
1. Program Basics	• Describe the steps a programmer goes through to tackle a new problem. • Recall the most common data types and how to create and use them in a program. • Insert conditional statements and functions into a program.	• Programming Mindset • Thinking Through Examples • Forming an Algorithm • Guide to Using the IDE • Testing • Data Types Introduction • Using Integers and Floats • Operators and Operands • Using Strings • String Operations • Debugging Operations • Coding Your First Program • Functions and Methods Introduction • Conditional Statements • Boolean Operators • Multiple Conditions • Exceptions • Debugging Conditional Statements • Drink Order Program
2. Lists and Loops	• Create and manipulate lists and other common data collection types. • Use loops to repeat steps either a fixed number of times or dynamically based on conditions. • Develop and test complex functions including special Python functions.	• Introduction to Lists • Manipulate Lists • Sets, Tuples, and Dictionaries • Multiple Dimensions • Debugging Lists

		• Tic-Tac-Toe Program • Introduction to Loops • Loops Using while • Loops Using for • Nested Loops • Debugging Loops • Revisiting the Tic-Tac-Toe Program • Function Arguments • The Return Statement • Nested Functions • Debugging Functions • Finishing the Tic-Tac-Toe Program
3. Classes	• Write the code for a basic class. • Remember the situations where inheritance and scope will affect a program's operations. • Use modules and files to add functionality and data to a program.	• Introduction to Classes • The init Method and del Function • Object Attributes • Object Methods • The Employee Class Program • Introduction to Inheritance • Subclasses • Scope • Class Troubleshooting • Revisiting the Employee Class Program • Introduction to Modules • Using Modules • Creating Modules • Introduction to File I/O • Reading and Writing to a File • Common Issues with Files • Finishing the Employee Class Program

4. Project	• Plan an algorithm for an original program. • Code and test an original program based on a prepared algorithm.	• Python Touchstone Overview • Identifying a Problem to Solve • Working An Example • Identifying the Patterns • Forming an Algorithm • Translating to Code • Writing the Program • Testing As You Go • Commenting Your Code • Course Wrap-Up