CS 138 - Programming with Python

Units Lecture 2.00	**Units Lab** 1.00	**Units Total** 0.00 - 3.00
Lecture Weekly Contact Hours 2.00	**Lab Weekly Contact Hours** 3.00	**Total Weekly Contact Hours** 5.00
Lecture Weekly Out of Class Hours 4.00	**Lab Weekly Outside of Class Hours** 0.00	**Total Weekly Outside of Class Hours** 4.00
Total Contact Hours 80.00 - 90.00	**Total Outside of Class Hours** 64.00 - - 72.00	**Total Course Hours** 144.00 - 162.00

Typically Offered: Fall, Spring, and Summer - F,SP,SU

COURSE DESCRIPTION

This course introduces the Python programming language and its features. Python, a dynamic, object-oriented, extensible language, is perfect for the beginner and also meets industry needs. Python is well-suited for applications ranging from simple data manipulation to large, complex applications.

ENROLLMENT RESTRICTIONS

Advisory
CS 101

OUTLINE OF COURSE LECTURE CONTENT

The course lecture will address the following topics:
I. Introduction to object-oriented programming (OOP) design and analysis
A. Classes, objects, methods, or functions
B. Encapsulation, inheritance and class hierarchies, polymorphism
C. Constructors.

II. Basic language elements
A. Declarations, initializations, primitive data types, parameters
1. Syntax (programming grammar)
2. Integer arithmetic
3. And/or logic.
B. Decision structures
1. If/else
2. Switch.
C. Loop structures

1. While and do-while
2. For and for-each.

III. Software engineering design techniques and strategies
A. Algorithms and Unified Modeling Language (UML) diagrams
B. Testing, error handling, and debugging strategies
C. Multi-source files and reusable code.

IV. Introduction to data structures
A. Arrays
B. Searching and sorting.

OUTLINE OF COURSE LAB CONTENT

The course lab will address the following topics:
All lecture topics will be applied to lab activities.

PERFORMANCE OBJECTIVES

Upon successful completion of this course, students will be able to do the following:
1). Develop, edit, link, run, and debug programs written in Python.
2). Explain and use Python program syntax and semantics.
3). Analyze problems and develop object-oriented solutions.
4). Write simple programs in Python including games that use classes, functions, and libraries.
5). Create programs in Python using sound, graphics, and animation.
6). Develop graphical user interfaces.
7). Create and use exception handling routines.

READING ASSIGNMENTS

Reading assignments will be consistent with, but not limited by, the following types and examples:
1). Read about object-oriented programming using the Python language in the class text as well as using and understanding specialized terms, concepts, and theories.
2). Reinforce text topics by researching the material on the Web and other texts for further explanation.
3). Read instructor-distributed handouts on specialized topics relevant to successful programming in industry using Python and extending this knowledge to class projects.

WRITING ASSIGNMENTS

Writing assignments will be consistent with, but not limited by, the following types and examples:
1). Create and write a requirements specification that defines what the program will accomplish and how the user will interact with it for each project done as homework and in class.
2). Write an essay on a current topic assigned in class, such as "What's new in Computing," and what it means and provides for computer scientists.
3). Contribute to discussion board topics on computer science areas and the Python language. Examples include "Compare and contrast statically and dynamically linked languages" or "Pick an object-oriented principle and explain its use and the pros and cons of it."

OUTSIDE-OF-CLASS ASSIGNMENTS

Outside-of-class assignments will be consistent with, but not limited by, the following types and examples:
1). Complete weekly programming assignments including a requirements specification.
2). Complete individual and group programming projects.
3). Prepare for individual presentations on specific topics discussed in the text.

STUDENT LEARNING OUTCOMES

1. The student will be able to analyze designs incorporating basic control structures and object-oriented design techniques which will result in appropriate solutions to specified problems.
2. The student will be able to implement effective solutions to simple programming problems. These solutions may utilize primitive data types and objects of pre-defined classes and may employ basic control structures, class library objects, and functions to control program execution, event-handling and construct a simple graphical user interface.
3. The student will be able to convey the basic concepts of the object-oriented paradigm and to describe basic syntactic and program control structures and write simple program designs using algorithms and class diagrams as well as explain a program design verbally and in writing.
4. The student will be able to work collaboratively and cooperatively in teams to develop basic software designs using algorithms and class diagrams as well as develop simple computer programs that implement these designs.

METHODS OF INSTRUCTION

Instructional methodologies will be consistent with, but not limited by, the following types or examples:
1). Group projects in class give students the opportunity to practice critical thinking skills in solving a problem as they evaluate their peers' code. They also allow student to see how other students approach and solve a problem, which enhances their programming skills.
2). Peer learning on coding, practice quizzes, and tests helps students learn the material.
3). Instructor lecture and instructor-led discussion about the Python programming language and its features.

METHODS OF EVALUATION

Evaluation methodologies will be consistent with, but not limited by, the following types or examples:
1). Individual programming projects will assess the student's ability to understand the material presented in the text and to properly apply it to writing a program.
2). Individual and group programming projects will be evaluated on the following:
a). Use of requirement specifications, proper syntax, and the application of object-oriented concepts and techniques that result in appropriate solutions to a specified problem.
b). Ability to integrate student's understanding of programming with the application of industry-standard coding practices.
3). Group programming projects will be evaluated using a grading rubric to assess the students' ability to

contribute both verbally and with written code to a programming project within the group context.
4). Class presentations of coding assignments will be evaluated to assess the students' ability to understand and accurately communicate their knowledge to others of modern and object-oriented programming techniques.
5). Performance-based exams will be assessed to evaluate how well the student understands, uses, and retains object-oriented programming principles and techniques.

REQUIRED TEXTBOOKS

Examples of typical textbooks for this course include the following:

1. **Author** Zelle, John
   Title Python Programming: An Introduction to Computer Science
   Edition 3rd ed.
   Publisher Franklin, Beedle & Assoc.
   Year 2016
   ISBN 978-1590282755
   This is the most current, in-print edition. No

2. **Author** Gaddis, Tony
   Title Starting Out with Python
   Edition 4th ed.
   Publisher Pearson Education
   Year 2018
   ISBN 978-1292225753
   This is the most current, in-print edition. No

3. **Author** Guzdial, Mark J., and Barbara Ericson
   Title Introduction to Computing and Programming with Python
   Edition 4th ed.
   Publisher Pearson
   Year 2016
   ISBN 978-1292109862
   This is the most current, in-print edition. No

4. **Author** Liang, Y. Daniel
   Title Introduction to Programming Using Python
   Edition 1st ed.
   Publisher Pearson
   Year 2013
   ISBN 978-0132747189
   This is the most current, in-print edition. No

COURSE REPEATABILITY

Total Completions Allowed: 1

Rationale for multiple enrollments:

Courses Related in Content (CRC) in Physical Education, Visual Arts, and Performing Arts:

DISTANCE ED (FORM A)

Type of Approval: 100% Online or Hybrid

You may indicate here which component(s) of the course should never be conducted online (e.g. proctored exams, labs, in-person orientation, etc.):

ARTICULATION

Transfer Status: Acceptable for Credit: CSU, UC -

CSU/IGETC GE Area(s): 103 - CSU, UC

THIS COURSE IS INCORPORATED INTO THE FOLLOWING PROGRAM(S)

Software Development *CURRENT* AA Degree
Software Development *CURRENT* Certificate of Achievement
Software Development *FUTURE* Certificate of Achievement
Software Development *FUTURE* AA Degree