New Credit Course: CS 139 - Advanced 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 - F,SP

## COURSE DESCRIPTION

This course continues with the Python programming language, using topics of personal and social relevance to investigate the impact of computing. This course includes data structures and libraries for modularization, data extraction and visualization, web APIs, server applications, and an introduction to machine learning.

## ENROLLMENT RESTRICTIONS

**Prerequisite**
CS 138

## OUTLINE OF COURSE LECTURE CONTENT

**The course lecture will address the following topics:**
I. API implementation
A. Working with data (JSON, YAML, CSV, or Excel)
1. Python libraries for decoding
2. Python libraries for data manipulation.
B. Web APIs
1. Using web APIs
2. Creating a simple web API
3. Security and authentication.
C. API design and creation
1. Architecture
2. Collections, resources, and URLs
3. Building sessions
4. Building APIs to accept a file upload.
D. Database integration

1. Connecting APIs to SQLite
2. Python and other databases
3. Reading and writing to databases with APIs.
E. Introduction to deploying APIs within Enterprise
1. Deploying a Flask app on serverless platform
2. Deploying a Flask app on containers
3. Automating build processes.

II. Building modules using classes
A. Definitions
1. Variables
2. Functions
3. Classes.
B. Import and implementation.

III. Server and serverless systems
A. API backends
B. Data processing
C. Stream processing workloads
D. Common applications.

IV. Data science and machine learning
A. Mathematical computing
B. Scientific computing
C. Data manipulation
D. Machine learning
E. Natural language processing
F. Data extraction and visualization.

V. Computational structures and techniques
A. Advanced searching and sorting
B. Maps and hashing
C. Trees.

## OUTLINE OF COURSE LAB CONTENT

**The course lab will address the following topics:**
Individual and group projects in the lab are designed to be hands-on activities that support, compliment, and extend the material and theory presented in the lectures. Students gain significant project experience.

## PERFORMANCE OBJECTIVES

**Upon successful completion of this course, students will be able to do the following:**
1). Demonstrate proficiency in implementing programs that draw on APIs of increasing size and complexity for a variety of problem domains.
2). Build simple modules using classes.
3). Compare and contrast server and serverless systems to identify the best solution for a given problem.
4). Analyze and solve problems by implementing simple data science and machine learning algorithms.
5). Implement data extraction and visualization methods.
6). Apply computational structures and techniques of abstraction to build increasingly complex programs.

## 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 that relates Python programming to the student's personal or professional interests.
3). Contribute to discussion board topics on computer science areas and the Python language. Examples include "Compare and contrast a server with a serverless platform" or "How is machine learning influenced by societal bias?"
4). After reading articles/material on the subject of emotional intelligence, students will be given prompts to respond to this topic: 1. Why is developing emotional intelligence important in the field of CS? 2. The article mentions, "when it comes to gauging important job candidates, many companies now rate emotional intelligence as important as technical ability and employ EQ testing before hiring." In what specific ways might emotional intelligence be crucial when working as part of a team?

## 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.
2). Complete individual and group programming projects including a requirements specification.
3). Post to class discussion boards.
4). Prepare for individual presentations on specific topics discussed in the text.

## STUDENT LEARNING OUTCOMES

1. The student will be able to solve culturally relevant computer science problems by applying critical thinking, object-oriented topics, and software engineering concepts.

2. The student will be able to work collaboratively and cooperatively in teams to develop software designs using algorithms and class diagrams as well as develop computer programs that implement these designs.

3. The student will be able to investigate and compare existing software libraries and incorporate elements to effectively implement a software solution.

## METHODS OF INSTRUCTION

**Instructional methodologies will be consistent with, but not limited by, the following types or examples:**
1). Instructor lecture and guided discussion about computer science and programming using an object-oriented language.
2). Group projects in class provide students with the opportunity to practice critical thinking skills in solving a problem as they evaluate their peers' code.
3). Use of peer learning on coding, practice quizzes, and tests.
4). Policies and practices that promote ownership of active learning, such as the following:

- Late policy that allows flexibility for various student circumstances.
- Opportunities to resubmit assignments to demonstrate continuous improvement.
- Personalizing choices of assessments to learn computer science and apply it to a student's life.
- Regular interactions with students to provide support and feedback.

## METHODS OF EVALUATION

**Evaluation methodologies will be consistent with, but not limited by, the following types or examples:**
1). Discussions evaluated for the student’s demonstrated ability to relate computer science to their lives.
2). In-class activities evaluated for the student’s ability to explain and use object-oriented and programming concepts.
3). Participation in campus events evaluated for the student’s ability to relate computer science to their lived experiences.
4). Programming labs and exercises evaluated for demonstrated improvement in the student’s ability to debug and resolve errors that occur while programming.
5). Unit deliverables evaluated for the student’s ability to design and implement user-centered solutions that relate to their lives and interests.
6). Exams and quizzes to align with industry certifications.

## REQUIRED TEXTBOOKS

Examples of typical textbooks for this course include the following:

1. **Author** Karamagi, Robert

   **Title** Advanced Python Programming

   **Edition** 1st ed.

   **Publisher** Independently Published

   **Year** 2020

   **ISBN** 979-8699711352

   **This is the most current, in-print edition.** Yes

2. **Author** Sharma, Pooja

   **Title** Programming in Python

   **Edition** 1st ed.

   **Publisher** BPB

   **Year** 2018

**ISBN** 978-9386551276

**This is the most current, in-print edition.** Yes

## OTHER REQUIRED INSTRUCTIONAL MATERIALS

1. Lubanovic, Bill. Introducing Python Modern Computing in Simple Packages. 2nd ed., O'Reilly, 2019. 978-1492051367
2. Sweigart, Al. Automate the Boring Stuff with Python: Practical Programming for Total Beginners. 2nd ed., No Starch, 2017. 978-1593279929

## 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.):**
None.

## ARTICULATION

**Transfer Status:** Acceptable for Credit: CSU -

**CSU/IGETC GE Area(s):** 102 - CSU

## THIS COURSE IS INCORPORATED INTO THE FOLLOWING PROGRAM(S)

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