

CS A231: PYTHON PROGRAMMING II

Item	Value
Curriculum Committee Approval Date	09/23/2020
Top Code	070600 - Computer Science (Transfer)
Units	4 Total Units
Hours	90 Total Hours (Lecture Hours 63; Lab Hours 27)
Total Outside of Class Hours	0
Course Credit Status	Credit: Degree Applicable (D)
Material Fee	No
Basic Skills	Not Basic Skills (N)
Repeatable	No
Grading Policy	Standard Letter (S), • Pass/No Pass (B)

Course Description

Advanced Python programming. Covers classes, modules, using the Python standard library and using third-party libraries. PREREQUISITE: CS A131. Transfer Credit: CSU; UC.

Course Level Student Learning Outcome(s)

1. Students will create, execute, and test Python applications using Object Oriented programming.
2. Students will write programs that use the Python standard library to solve a variety of problems, including interacting with the operating system and interacting with Web page.
3. Students will write programs that use external libraries to solve a variety of problems, including creating graphs and analyzing data sets.

Course Objectives

- 1. Develop Python classes to create modular programs.
- 2. Create Python modules.
- 3. Solve advanced programming problems using Python's standard libraries.
- 4. Describe how to use and handle exceptions to write robust programs.
- 5. Interact with a variety of other computer or Web applications through Python scripts and programs.
- 6. Implement systematic testing techniques to test and debug large programs.
- 7. Create graphical user interfaces for Python programs to make them user friendly.
- 8. Maintain persistent data through the use of different file types, including text and CSV files.
- 9. Explain why libraries are needed to interact with the operating system instead of directly accessing the operating system.
- 10. Demonstrate the ability to find and choose an appropriate library to solve a particular problem.

Lecture Content

REVIEW PYTHON DATA TYPES Strings Floats Ints Booleans Tuples
 REVIEW PYTHON COLLECTION TYPES Lists Dictionaries Sets REVIEW
 LOOPING AND CONTROL STRUCTURES Using Python Control Structures
 Iteration Handling Exceptions Getting Data In and Out of Python REVIEW
 BASIC USER INPUT AND FILE I/O Interacting with Users Using Text Files
 PYTHON FUNCTIONS Defining and Using Functions Generator Functions
 Lambda Functions CREATING PYTHON CLASSES AND MODULES
 Defining and Using Classes and Objects Creating and Using Modules and
 Packages SCRIPTING WITH PYTHON - ACCESSING THE OPERATING
 SYSTEM Accessing the Operating System Obtaining Information About
 Users and Their Computer Obtaining Information About the Current
 Process Managing Other Programs Obtaining Information About Files
 (and Devices) Navigating and Manipulating the File system Accessing
 the Operating System Libraries PROCESSING AND PARSING FILES
 Working with Dates and Times Handling Common File Formats Working
 with XML and HTML Files Parsing HTML Files Accessing Native APIs
 with ctypes and pywin32 MANAGING DATA Storing Data Using Python
 Using Pickle to Store and Retrieve Objects Analyzing Data with Python
 Analyzing Data Using Built-In Features of Python Analyzing Data with
 itertools Utility Functions Data Processing Functions USING DATABASES
 Managing Data Using SQL Relational Database Concepts Accessing
 SQL from Python Using SQL Connections BUILDING CONSOLE-BASED
 DESKTOP APPLICATIONS Structuring Applications Building Command-
 Line Interfaces Using the cmd Module to Build a Command-Line Interface
 Reading Command-Line Arguments BUILDING GRAPHICAL DESKTOP
 APPLICATIONS Introducing Key GUI Principles Event Based Programming
 GUI Terminology Building a Simple GUI Playing Games with Python
 MAINTAINING PERSISTENT PROGRAM DATA Storing Local Data
 Storing Application-specific Data Storing User- Selected Preferences
 PYTHON ON THE WEB Python on the Web Parts of a Web Application
 The ClientServer Relationship Middleware and MVC HTTP Methods
 and Headers What Is an API Web Programming with Python Using the
 Python HTTP Modules Static Site Generators UNIT-TESTING PYTHON
 PROJECTS Testing with the Doctest Module Testing with the unittest
 Module TestDriven Development in Python Debugging Your Python Code
 MANAGING EXCEPTIONS AND DEPENDENCIES Handling Exceptions in
 Python Working on Larger Python Projects

Lab Content

Create and systematically test programs that: Access the Operating
 System Obtain Information About Users and Their Computer Obtain
 Information About the Current Process Manage Other Programs
 Process files Process web files and interact with a network Use Python
 efficiently and ensure robustness in large project by Testing with the
 Doctest Module Testing with the unittest Module Working on Larger
 Python Projects involving multiple user-defined modules, the Python
 standard library, and third party libraries Developing GUIs to enhance user
 interactivity with the Python programs

Method(s) of Instruction

- Lecture (02)
- DE Live Online Lecture (02S)
- DE Online Lecture (02X)
- Lab (04)
- DE Live Online Lab (04S)
- DE Online Lab (04X)

Instructional Techniques

Lecture, demonstration and programming exercises.

Reading Assignments

Students will spend a minimum of 3 hours per week reading the textbook and/or other reading material assigned. Students will be expected to follow along with the exercises in the reading material.

Writing Assignments

Students will spend a minimum of 4 hours per week writing code.

Out-of-class Assignments

Students will spend a minimum of 4 hours per week completing weekly programming assignments.

Demonstration of Critical Thinking

Students will demonstrate the ability to write programs that solve different kinds of problems.

Required Writing, Problem Solving, Skills Demonstration

Students will demonstrate proficiency writing computer programs.

Eligible Disciplines

Computer science: Masters degree in computer science or computer engineering OR bachelors degree in either of the above AND masters degree in mathematics, cybernetics, business administration, accounting or engineering OR bachelors degree in engineering AND masters degree in cybernetics, engineering mathematics, or business administration OR bachelors degree in mathematics AND masters degree in cybernetics, engineering mathematics, or business administration OR bachelors degree in any of the above AND a masters degree in information science, computer information systems, or information systems OR the equivalent. Note: Courses in the use of computer programs for application to a particular discipline may be classified, for the minimum qualification purposes, under the discipline of the application. Masters degree required.

Textbooks Resources

1. Required Lutz, Mark. Programming Python, 4th ed. OReilly: Sebastopol, 2016