This course will cover the fundamentals of programming using Python language. The process of software development will be discussed to include: designing, writing source code, executing, and testing and debugging. Data types, arithmetic and logical expressions, debugging, looping, branching, modularization, simple database access, simple database structures and simple Graphical User Interface will be discussed in lectures and practiced through lab projects. Business, scientific and mathematics applications will be designed and created.

JUSTIFICATION FOR COURSE:

PREREQUISITES:

COREQUISITES:

ADVISORIES:
  • CS G102: Computer Software Development, Introduction

ASSIGNED DISCIPLINES:
  Computer science

MATERIAL FEE: Yes [ ] No [X] Amount: $0.00

CREDIT STATUS: Noncredit [ ] Credit - Degree Applicable [X] Credit - Not Degree Applicable [ ]

GRADING POLICY: Pass/No Pass [X] Standard Letter [X] Not Graded [ ] Satisfactory Progress [ ]

OPEN ENTRY/OPEN EXIT: Yes [ ] No [X]

TRANSFER STATUS: CSU Transferable[ ] UC/CSU Transferable[X] Not Transferable[ ]

BASIC SKILLS STATUS: Yes [ ] No [X] LEVELS BELOW TRANSFER: Not Applicable

CALIFORNIA CLASSIFICATION CODES: Y - Not Applicable

NON CREDIT COURSE CATEGORY: Y - Not applicable, Credit Course

OCCUPATIONAL (SAM) CODE: C

REPEATABLE ACCORDING TO STATE GUIDELINES: No [X] Yes [ ] NUMBER REPEATS:

REQUIRED FOR DEGREE OR CERTIFICATE: No [X] Yes [ ]

GE AND TRANSFER REQUIREMENTS MET:

COURSE LEVEL STUDENT LEARNING OUTCOME(S) Supported by this course:
1. Create programs partitioned into appropriate functions and modules.
2. Design a simple graphical user interface to satisfy the user interactions.
3. Develop programming solutions using all the necessary expressions, branches, loops, functions, classes.
4. Implement the appropriate error handling routines.

**COURSE OBJECTIVES:**
1. Assess, analyze, and design software solutions for simple to moderately complex business and database problems.
2. Create properly documented programming solutions.
3. Construct the software code, mathematical formulas/expressions, and algorithms in the Python language.
4. Test for and eliminate coding and logic errors using debugging techniques.
5. Understand and apply basic optimization techniques.
6. Design and implement large software solutions using manageable modules.
7. Develop code to interact with input, output devices and files.

**COURSE CONTENT:**

**LECTURE CONTENT:**

A. Introduction to Computers and Programming
   1. Computer hardware
   2. Operating systems
   3. Networks and network protocols
   4. Programming languages
   5. Software libraries

B. Input, Processing, and Output
   1. The Python Programming Language
   2. Abstraction and modeling
   3. Imperative Programming
   4. Algorithms
   5. Data types
   6. Expressions, Variables and Assignments

C. Decision Structures and Boolean Logic
   1. The if Statement
   2. The if-else Statement
   3. Comparing Strings
   4. Nested Decision Structures and the if-elif-else Statement
   5. Logical Operators
   6. Boolean Variables

D. Repetition Structures
   1. The while Loop: a Condition-Controlled Loop
   2. The for Loop: a Count-Controlled Loop
   3. Calculating a Running Total
   4. Sentinels
   5. Input Validation Loops
   6. Nested Loops

E. Functions and Modules
   1. Defining and Calling a Void Function
   2. Designing a Program to Use Functions
   3. Local Variables
4. Passing Arguments to Functions
5. Global Variables and Global Constants

F. Files and Exceptions
   1. File Input and Output
   2. Using Loops to Process Files
   3. Processing Records
   4. Exceptions

G. Lists and Tuples
   1. Sequences
   2. Introduction to Lists
   3. List Slicing
   4. Finding Items in Lists with the in Operator
   5. List Methods and Useful Built-in Functions
   6. Two-dimensional lists

H. Text Data
   1. Basic String Operations
   2. String Slicing
   3. Testing, Searching, and Manipulating Strings
   4. Formatted Output

I. Containers and Randomness
   1. Dictionaries
   2. Sets
   3. Serializing Objects
   4. Class tuple
   5. Module random

J. Classes and Object-Oriented Programming
   1. Procedural and Object-Oriented Programming
   2. Classes
   3. Working with Instances
   4. Designing Classes
   5. Inheritance
   6. Class Relationships
   7. Polymorphism

K. Recursion
   1. Introduction to Recursion
   2. Problem Solving with Recursion
   3. Examples of Recursive Algorithms

L. GUI Programming
   1. Graphical User Interfaces
   2. Using the tkinter Module
   3. Display Text with Label Widgets
   4. Organizing Widgets with Frames
   5. Button Widgets and Info Dialog Boxes
   6. Getting Input with the Entry Widget
   7. Using Labels as Output Fields
   8. Radio Buttons and Check Buttons

LABORATORY CONTENT:
1) Given a set of requirements for a small business or scientific problem, prepare the software development specification
2) Design the software components and draw flow-charts for the complex code sections
3) Create a project with the correct logic implementation
4) Develop Python scripts to implement the solution
5) Design a simple user interface to satisfy the user interactions
6) Code all the needed expressions, branches, loops, functions and data structures
7) Add the appropriate error handling routines

METHODS OF INSTRUCTION:

A. Lecture:
B. Lab:
C. Online:
D. Independent Study:

INSTRUCTIONAL TECHNIQUES:

Lecture and/or discussion on specific topics with sample code demonstration using the projector and the whiteboard.

COURSE ASSIGNMENTS:

Reading Assignments

Text
Websites

Out-of-class Assignments

Writing Assignments

Students will be required to complete software development projects presented to them in the form of business automation problems requiring solution implementation. Students will be required to write documentation for their projects.

METHODS OF STUDENT EVALUATION:

Midterm Exam
Final Exam
Short Quizzes
Written Assignments
Report
Projects (ind/group)
Problem Solving Exercises
Oral Presentations
Skills Demonstration

Demonstration of Critical Thinking:

Students will be demonstrating their laboratory projects. Optional research papers and classroom presentations will further demonstrate their ability in critical thinking and problem solving.

Required Writing, Problem Solving, Skills Demonstration:

Students will be required to complete software development projects presented to them in the form of business automation problems requiring solution implementation. Students will be required to write documentation for their projects.

TEXTS, READINGS, AND RESOURCES:

TextBooks:


LIBRARY:

Adequate library resources include:

Comments: