CS A131: PYTHON PROGRAMMING I

Item	Value
Curriculum Committee Approval Date	12/06/2023
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

Introduction to fundamental concepts and techniques for writing software in the Python programming language. Covers the syntax and semantics of data types, expressions, exceptions, control structures, input/output, methods, classes, and pragmatics of Python programming. ADVISORY: CIS A090; and CIS A100 or CIS A111. Transfer Credit: CSU; UC. C-ID: COMP 112.**C-ID:** COMP 112.

Course Level Student Learning Outcome(s)

1. Students will create, execute, and test Python applications using a development enviromnent.
2. Students will write programs that correctly use Python data types, including lists and dictionaries, to solve various kinds of problems.
3. Students will write programs that correctly use functions and control structures, including selection and iteration to solve different kinds of programming problems.

Course Objectives

- 1. Write, test, and debug code within a development environment
- 2. Combine data types and organization schemes to model real-world information
- 3. Implement algorithms within the constraints of Pythons syntax and semantics
- 4. Employ Pythons abstraction mechanisms to reduce complexity and increase reliability of software
- 5. Follow a systematic methodology to produce solutions expeditiously and reduce the introduction of bugs
- 6. Predict the results that source code will produce when executed
- 7. Describe the practical limitations of software systems
- 8. Describe how programming languages fit into the broader context of modern computing systems

Lecture Content

Introduction to Computer Science What is Computer Science. What do computing professionals do. Models, algorithms and programs Tools that are used to program Introducing Computer Systems Computer hardware Operating systems Networks and network protocols Programming languages Software libraries The Python Programming Language A short history of Python Setting up the Python development environment Computational Thinking Abstraction and modeling Algorithms Data types Assignment and control structures Python Data Types Expressions, Variables and Assignments Algebraic expressions and functions Boolean expressions and operators Variables and assignments Rules for variable names Strings String operators The indexing operator Lists List operators Mutablility of Lists (immutability of Strings) List methods Objects and Classes Object type Valid values for number types Operators for number types Creating objects Implicit and explicit type conversions Class methods and Object-Oriented programming Introducing the Python Standard Library Module math Module fractions Imperative Programming Python Programs (IPO) Python Modules Built-in function print() Interactive input with input() Using the eval() function Introducing Control Structures One-way and two-way decisions Iteration control structures Nesting of control flow Using the range() function User-defined Functions Basic function syntax Using return instead of print() Function definition and assignment Comments and docstrings Python Variables and Assignments Mutable and immutable types Assignments and mutability Swapping Parameter Passing Immutable parameter passing Mutable parameter passing Text Data, Files and Excpetions String Representation The indexing operator String methods Formatted Output The print() function The String method format() Lining up data in columns Files The file system Opening and closing a file Reading a text file Writing to a text file Errors and Exceptions Syntax errors Built-in runtime exceptions Execution Control Structures Decision Control and the if Statement Multi-way decisions Ordering of conditions Loops for loop while loop Iteration Patterns Iteration loop Counter loop Accumulator loop Sequence loop Infinite loop Loop and a half Lists Two-dimensional lists Nested loops Iteration Control Statements break statement continue statement pass statement Containers and Randomness Dictionaries Dictionary class properties Dictionary operators Dictionary methods Substituting a multi-way condition Collecting counters I> Class tuple tuple Objects as dictionary keys Dictionary method items() Class set set operators set methods Removing duplicates with the set constructor Module random Choosing a random integer Choosing random "real" Shuffling, choosing, and sampling

Lab Content

The following programming labs and exercises are designed to help students master the topics learned by providing hands-on practice for a comprehensive understanding of the material: Practice solving algebraic expressions and functions using Python. Manipulate strings and lists efficiently. Create simple classes with methods tailored for different number types. Identify and fix syntax errors in provided Python code. Implement user-defined functions for common computations. Read and write data to/from text files proficiently. Handle various types of errors and exceptions. Engage in programming exercises focused on string representation and formatted output. Develop programs that make decisions using if statements. Implement loops to address various iteration patterns. Solve problems using lists and nested loops for intricate scenarios. Create, modify, and access data in dictionaries efficiently. Use tuples and sets strategically based on specific scenarios. Apply the random module for generating random data in a controlled manner. Collaborative project where students apply multiple concepts

learned. Peer review: Analyze and enhance each others code for better practices.

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 in-class programming exercises.

Reading Assignments

Students will spend a minimum of 4 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 6 hours per week writing code.

Out-of-class Assignments

Students will spend a minimum of 6 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 Python 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 Gaddis, Tony. Starting Out with Python, 5th ed. New York: Pearson, 2020