

Course Outline

COURSE: CSIS 42 **DIVISION:** 50 **ALSO LISTED AS:**

TERM EFFECTIVE: Fall 2014 **CURRICULUM APPROVAL DATE:** 02/24/2014

SHORT TITLE: PYTHON PROGRAMMING

LONG TITLE: Python Programming

<u>Units</u>	<u>Number of Weeks</u>	<u>Type</u>	<u>Contact Hours/Week</u>	<u>Total Contact Hours</u>
4	18	Lecture:	4	72
		Lab:	0	0
		Other:	0	0
		Total:	4	72

COURSE DESCRIPTION:

Introduction to computing using Python. Study and create programs that perform various tasks, including text and file manipulation, internet scripting, data structures, testing, and practical problem solving with examples. Covers object-oriented programming and the Python Standard Library. Introduces students to the fundamental concepts of programming. This course has the option of a letter grade or pass/no pass. (C-ID: COMP 112)

PREREQUISITES:

COREQUISITES:

CREDIT STATUS: D - Credit - Degree Applicable

GRADING MODES

- L - Standard Letter Grade
- P - Pass/No Pass

REPEATABILITY: N - Course may not be repeated

SCHEDULE TYPES:

- 02 - Lecture and/or discussion
- 05 - Hybrid
- 72 - Dist. Ed Internet Delayed

STUDENT LEARNING OUTCOMES:

1. Describe the software development life-cycle.

Measure: discussion, quiz

PLO:

ILO: 2,1,3,7

GE-LO: A3, A6

Year assessed or anticipated year of assessment: 2014-15

2. Describe the principles of structured programming and be able to describe, design, implement, and test structured programs using currently accepted methodology.

Measure: homework, quiz, case study, problem set

PLO:

ILO: 7,2,3

GE-LO:

Year assessed or anticipated year of assessment: 2014-15

3. Explain what an algorithm is and its importance in computer programming.

Measure: discussion, quiz, homework

PLO:

ILO: 7,3,2,1

GE-LO: A3, A6

Year assessed or anticipated year of assessment: 2014-15

4. Recognize and construct common programming idioms: variables, loop, branch, subroutine, and input/output.

Measure: quiz, case study, project, problem set, homework

PLO: 2

ILO: 2,7,3

GE-LO:

Year assessed or anticipated year of assessment: 2014-15

5. Define and demonstrate the use of the built-in data structures 'list' and 'dictionary'.

Measure: quiz, case study, project, problem set, homework

PLO: 1,2

ILO: 2,7,3

GE-LO:

Year assessed or anticipated year of assessment: 2014-15

6. Apply idioms to common problems such as text manipulation, web page building, and working with large sets of numbers.

Measure: case study, project, homework, problem set

PLO: 1,2

ILO: 2,7,3

GE-LO:

Year assessed or anticipated year of assessment: 2014-15

7. Design and implement a program to solve a real-world problem using the language idioms, data structures, and standard library.

Measure: project

PLO: 1,2

ILO: 2,7,3

GE-LO: B7

Year assessed or anticipated year of assessment: 2014-15

Program Learning Outcomes:

1. Student will create programs in three different languages that use control flow statements such as if and switch statements.

2. Student will create programs in three different languages that use loop statements

such as for and while statements.

CONTENT, STUDENT PERFORMANCE OBJECTIVES, OUT-OF-CLASS ASSIGNMENTS

Curriculum Approval Date: 02/24/2014

(8 Hours) Lectures: Introduction to: Computer Science, Computer Systems, Python Programming Language, and Computational Thinking. Python Data Types: Expressions, Variables, and Assignments; Strings; Lists; Objects and Classes; and Python Standard Library. Quiz and problem set.

Student Performance Objectives: Discuss the different components of a computer system including the hardware, the operating system, the network and the Internet, and the programming language used to write programs. Identify the Python programming language. Define the concept of a data type and that of an object that stores a value of a given type. Use the Python IDE interactive shell as a calculator to evaluate Python expressions, starting with simple algebraic expressions, and execute single Python statements. Utilize the turtle graphics module to visually illustrate the concepts: objects, types, and names; data abstraction and classes; and information hiding.

Out of Class Assignments: Read Introductory chapter. Read chapter on Python Data Types. HW: Explore Python.org website and the resources and documentation therein. Download a Python IDE on your computer, choose the appropriate installer for your system, and complete the installation.

(10 Hours) Lectures: Imperative Programming: Python Programs, Execution Control Structures, User-Defined Functions, Python Variables and Assignments, and Parameter Passing. Text Data, Files, and Exceptions: Strings, Revisited; Formatted Output; Files; and Errors and Exceptions.

Student Performance Objectives: Discuss how to develop Python programs. Discuss a few decision and iteration control flow structures that control whether and how many times particular statements are executed. Explain the benefits of functions. Explain the process of breaking down problems into steps that can be described computationally using Python statements. Utilize turtle graphics to illustrate code reuse, information hiding, and functional abstraction. Discuss the extensive set of string methods that give Python powerful text-processing capabilities. Describe how to read from and write to files from within a Python program. Define several patterns for reading files that prepare the file content for processing. List the common errors that can occur when working with data coming interactively from the user or from a file. Utilize the case study to showcase the text-processing and I/O concepts in the context of an application that logs accesses to files.

Out of Class Assignments: Read chapter on Imperative Programming. Read chapter on Text Data, Files, and Exceptions. HW: Write simple interactive programs that use built-in functions 'print ()', 'input ()', and 'eval ()'. Create programs that execute differently depending on the input entered by the user. Format output using features of the 'print ()' function and the string 'format ()' method.

(10 Hours) Lectures: Execution Control Structures: Decision Control and the 'if' Statement, 'for' Loop and Iteration Patterns, More on Lists: Two-Dimensional Lists, 'while' Loop, More Loop Patterns, and Additional Iteration Control Statements. Containers and Randomness: Dictionaries, Other Built-In Container Types, Character Encoding and Strings, and Module 'random'. Quiz and problem set.

Student Performance Objectives: Define the 'if' statement. Explain the purpose of the 'for' loop and the 'while' loop. Discuss the different iteration patterns and describe when and how they are used. Identify the Python statements and techniques that provide control over what code blocks will be executed when, and how often. Name the other built-in container classes available in Python and describe their uses. Explain when and how to use the 'tuple' and 'set' built-in container classes. Review strings and analyze them as containers of characters. Describe encoding and decoding characters from different writing systems, including the use of Unicode. Discuss how to generate "random" numbers.

Out of Class Assignments: Read chapter on Execution Control Structures. Read chapter on Containers and Randomness. HW: Programming problems utilizing the concepts covered in lecture.

(10 Hours) Lectures: Namespaces: Encapsulation in Functions, Global versus Local Namespaces, Exceptional Control Flow, Modules as Namespaces, and Classes as Namespaces. Object-Oriented Programming: Defining a New Python Class, Examples of User-Defined Classes, Designing New Container Classes, Overloaded Operations, Inheritance, and User-Defined Exceptions. Midterm exam and problem set.

Student Performance Objectives: Explain the purpose of namespaces. Discuss the concepts and techniques that deal with program design. Review the main benefits of functions. Describe how to develop new Python classes. Explain the benefits of the object-oriented programming (OOP) paradigm and discuss core OOP concepts. Utilize the case study to learn how to make a container class feel more like a built-in class. Use

the case study to enable indexing of items in the container and enabling iteration, using a 'for loop', over the items in the container.

Out of Class Assignments: Read chapter on Namespaces. Read chapter on Object-Oriented Programming. HW: Programming problems utilizing the concepts covered in lecture.

(12 Hours) Lectures: Graphical User Interfaces: Basics of 'tkinter' GUI Development, Event-Based 'tkinter' Widgets, Designing GUIs, and OOP for GUIs. Quiz and problem set.

Student Performance Objectives: List the reasons for using a GUI. Discuss the GUI API for Python - 'tkinter'. Illustrate how to facilitate the geometry specification of more complex GUIs by organizing the widgets in a hierarchical fashion. Describe how to define the handlers that are executed in response to user-generated events such as mouse button clicks, mouse motion, or keyboard key presses. Use the context of GUI development to showcase the benefits of OOP. Describe how to develop GUI applications as new widget classes that can be easily incorporated into larger GUIs. Apply OOP concepts such as class inheritance, modularity, abstraction, and encapsulation. Utilize the case study to implement a basic calculator GUI. Use OOP techniques to implement it as a user-defined widget class, from scratch. Explain how to write a single event-handling function that handles many different buttons.

Out of Class Assignments: Read chapter on Graphical User Interfaces. HW: Implement function 'cal ()' that takes as input a year and a month (a number between 1 and 12) and starts up a GUI that shows the corresponding calendar.

(10 Hours) Lectures: Recursion: Introduction and Examples, Run Time Analysis, and Searching. Quiz. Project proposal - describe the program you will design, implement and test, using Python, to solve a real-world problem.

Student Performance Objectives: Discuss when recursion should and should not be used. Analyze several fundamental search tasks. Describe how namespaces and the program stack support the execution of recursive functions. Develop recursive functions for a variety of problems, such as the visual display of fractals and the search for viruses in the files of a filesystem. Apply recursion to solve the Tower of Hanoi problem and develop a graphical tool, using OOP techniques and the 'turtle' graphics module, to visualize the solution.

Out of Class Assignments: Read chapter on Recursion. HW: Programming problems using recursion.

(10 Hours) Lectures: The Web and Search: The World Wide Web, Python WWW API, and String Pattern Matching.

Student Performance Objectives: Describe the three core WWW technologies, the Python Standard Library web APIs, and the algorithms that can be used to develop such applications. Apply the Python Standard Library APIs to download a web page HTML source file and parse it to obtain the web page content. Practice using the tools that recognize string patterns in texts: regular expressions and the Standard Library module 're'. Utilize the case study to develop a web crawler; which will access, search, and collect data hosted on the World Wide Web.

Out of Class Assignments: Read chapter on The Web and Search. HW: Do related programming problems.

(2 Hours) Written Final Exam.

METHODS OF INSTRUCTION:

Lectures, Computer Demonstrations, Case Studies.

METHODS OF EVALUATION:

No Change 0 Change 1

Category 1 - The types of writing assignments required:

Percent range of total grade: 20 % to 30 %

Written Homework

Other: Project

If this is a degree applicable course, but substantial writing assignments are not appropriate, indicate reason:

Course primarily involves skill demonstration or problem solving

Category 2 -The problem-solving assignments required:

Percent range of total grade: 30 % to 60 %

Homework Problems

Quizzes

Other: Case Studies

Category 3 -The types of skill demonstrations required:

Percent range of total grade: 10 % to 20 %

Performance Exams

Category 4 - The types of objective examinations used in the course:

Percent range of total grade: 10 % to 30 %

Multiple Choice

True/False

Matching Items

Completion

REPRESENTATIVE TEXTBOOKS:

Required:

Perkovic, Ljubomir. Introduction to Computing Using Python. John Wiley & Sons, Inc. 2012. Or other appropriate college level text.

ISBN: 978-0-470-61846-2

Reading level of text, Grade: 12 Verified by: MS Word

ARTICULATION and CERTIFICATE INFORMATION

Associate Degree:

CSU GE:

IGETC:

CSU TRANSFER:

Transferable CSU, effective 200870

UC TRANSFER:

Transferable UC, effective 200870

SUPPLEMENTAL DATA:

Basic Skills: N

Classification: Y

Noncredit Category: Y

Cooperative Education:

Program Status: 1 Program Applicable

Special Class Status: N

CAN:

CAN Sequence:

CSU Crosswalk Course Department: CSIS

CSU Crosswalk Course Number: 42

Prior to College Level: Y

Non Credit Enhanced Funding: N

Funding Agency Code: Y

In-Service: N

Occupational Course: C

Maximum Hours:

Minimum Hours:

Course Control Number: CCC000456077

Sports/Physical Education Course: N
Taxonomy of Program: 070710