1. **COURSE ID:** CIS 117  
   **TITLE:** Python Programming  
   **Units:** 4.0 units  
   **Hours/Semester:** 48.0-54.0 Lecture hours; 48.0-54.0 Lab hours; and 96.0-108.0 Homework hours  
   **Method of Grading:** Grade Option (Letter Grade or P/NP)  
   **Recommended Preparation:**  
   Eligibility for ENGL 838 or ENGL 848  
   Eligibility for ESL 400, Completion of CIS 111 or CIS 254.

2. **COURSE DESIGNATION:**  
   Degree Credit  
   **Transfer credit:** CSU; UC

3. **COURSE DESCRIPTIONS:**  
   **Catalog Description:**  
   Comprehensive course in Python, an open-source dynamic object-oriented scripting language. Covers variables, arrays, lists, tuples, dictionaries, functions, methods, classes, objects, and writing server-side Python scripts for the Web. Also covered are exception handling, regular expressions, and modules. An introduction to SQL and the MySQL database, and advanced topics such as Model-View-Controller architecture and Web application development with the Django framework. Intended for students with previous programming experience.

4. **STUDENT LEARNING OUTCOME(S) (SLO'S):**  
   Upon successful completion of this course, a student will meet the following outcomes:  
   1. Develop server-side Python scripts for publishing on the Web.  
   2. Employ control structures, functions, and arrays to create Python programs.  
   3. Apply object-oriented programming concepts to develop dynamic interactive Python applications.  
   4. Employ Python sequences and mappings to store and manipulate data.  
   5. Use SQL commands and the MySQL database together with Python.  
   6. Create an advanced project using MySQL, Python and a Model-View-Controller framework.

5. **SPECIFIC INSTRUCTIONAL OBJECTIVES:**  
   Upon successful completion of this course, a student will be able to:  
   1. Develop server-side Python scripts for publishing on the Web.  
   2. Employ control structures, functions, and arrays to create Python programs.  
   3. Apply object-oriented programming concepts to develop dynamic interactive Python applications.  
   4. Employ Python sequences and mappings to store and manipulate data.  
   5. Use SQL commands and the MySQL database together with Python.  
   6. Create an advanced project using MySQL, Python and a Model-View-Controller framework.

6. **COURSE CONTENT:**  
   **Lecture Content:**  
   1. Introduction  
   2. History of Python  
   3. Dynamic Web Applications  
   4. Python Installation  
   5. Web Servers and Server-Side Programming  
   6. Python and Databases  
   7. UNIX/Linux Environment  
   8. Data and Operations  
   9. Data Types  
   10. Variables and Constants  
   11. Operators  
   12. Expressions  
   13. Operator Precedence  
   14. Scope

   1. Conditional Statements and Control Structures
3. Conditional Statements and Control Structures

- a. If Else
- b. Elif Clause
- c. While
- d. For
- e. Parallel Iteration
- f. Numbered Iteration

4. Functions
- a. Function Definition
- b. Blocks
- c. Calling a Function
- d. Function Parameters
- e. Function Return Values
- f. Lambda Expressions
- g. Modules
- h. Recursion
- i. Exception Handling

5. Collections
- a. Sequences
  - i. Lists
  - ii. Strings
  - iii. Tuples
- b. Mappings (Dictionaries)
- c. Array Module

6. Object-Oriented Python
- a. Defining a Class
- b. Constructors
- c. Encapsulation
- d. Inheritance
- e. Interfaces
- f. Iterators
- g. Generators
- h. Polymorphism
- i. Introspection
- j. Static and Class Methods
- k. Magic Methods

7. Web Interaction and File Access
- a. Python HTML Code Generation
- b. CGI Code Generation
- c. Templating Systems
- d. Form Input
- e. Validation and Regular Expressions
- f. Cookie Module
- g. Reading/Writing Files
- h. File Iterators

8. SQL and the MySQL Database
- a. Introduction to SQL Syntax
- b. Designing and Creating a Table in MySQL
- c. MySQL Data Types
- d. MySQLdb Module

9. Dynamic Python Applications with Django and MySQL
- a. Model-View-Controller Architecture
- b. Connecting to MySQL
- c. Object-Relational Mapping
d. Authentication  
e. Testing and Debugging  

**Lab Content:**
- Write Python applications using variables, data types, strings and functions.
- Write Python applications using loops, arrays, hashes and sorting.
- Write Python applications using dictionaries, lists and tuples.
- Write Python applications using matrices.
- Create graphical programs using objects.
- Create and manage a MySQL database and write data-driven Python applications.
- Create projects using MySQL, Python and the Django framework.
- Employ TDD (Test Driven Development) to write tests to validate an application's behavior.

7. **REPRESENTATIVE METHODS OF INSTRUCTION:**
   Typical methods of instruction may include:
   - A. Lecture  
   - B. Lab  
   - C. Activity  
   - D. Directed Study  
   - E. Discussion  
   - F. Observation and Demonstration  
   - G. Other (Specify): teacher will model problem solving techniques; teacher will create and manage an Internet conference for discussion of course topics; and students will work in small groups to solve programming assignments.

8. **REPRESENTATIVE ASSIGNMENTS**
   Representative assignments in this course may include, but are not limited to the following:
   - **Writing Assignments:** Weekly programming assignments.
   - **Reading Assignments:** Reading assignments accompanied by self-test questions and running code examples.
   - **Other Outside Assignments:** Studying posted lecture notes and relevant handouts. Computer programming assignments. The reading assignment frames the concepts covered and provides the basic knowledge necessary to do the self-test questions and understand the sample code. The lecture notes and handouts provide a more in-depth look at topics and distills the information down to what the faculty thinks is most important. The computer programming assignments provide hands on practice of the concepts covered in the readings.

9. **REPRESENTATIVE METHODS OF EVALUATION**
   Representative methods of evaluation may include:
   - A. Exams/Tests  
   - B. Group Projects  
   - C. Homework  
   - D. Lab Activities  
   - E. Projects  
   - F. Quizzes  
   - G. Written examination  
   - H. Bi-weekly quizzes (short answer--from textbook material) to provide feedback to students and teacher. Assessment of student contributions during class discussion and project time. Individual programming assignments. Midterm and Final exams (short answer from textbook material, general problem solving (similar to in-class work), short program segments (similar to programming assignments). Assessment of group participation on course projects, including peer-assessment of participation and contribution to the group effort.

10. **REPRESENTATIVE TEXT(S):**
    Possible textbooks include: