# CSC 121: Python Programming

**COURSE DESCRIPTION:**

Prerequisites: None
Corequisites: None

This course introduces computer programming using the Python programming language. Emphasis is placed on common algorithms and programming principles utilizing the standard library distributed with Python. Upon completion, students should be able to design, code, test, and debug Python language programs. Course Hours per Week: Class, 2. Lab, 3. Semester Hours Credit, 3.

**LEARNING OUTCOMES:**

Upon completing requirements for this course, the student will be able to:

1. Create a software application using the Python programming language.
2. Debug a software application written in the Python programming language.
3. Test a software application written in the Python programming language.

**OUTLINE OF INSTRUCTION:**

I. General Introduction
   - A. Programs & Algorithms
   - B. Introduction to Python
   - C. Introduction to debugging

II. Simple Python Data
   - A. Variables, Expressions, and Statements
   - B. Values and Data Types
   - C. Statements and Expressions, Operators
   - D. Input and Output

III. Debugging Interlude
   - A. How to Avoid Debugging
   - B. Beginning Tips
   - C. Know your Error Messages

IV. Python Turtle Graphics
   - A. Instances
   - B. The *for* loop, Flow of Executions
   - C. Iteration and the Range Functions

V. Python Modules
   - A. Modules and Getting Help
   - B. The *math* module
   - C. The *random* module

VI. Functions
   A. Functions
   B. Unit Testing
   C. Local Variables and Parameters
   D. The Accumulator Pattern
   E. Nesting Functions, Flow of Execution
   F. Using the main function
   G. Program Development

VII. Selection
   A. Boolean Values and Expressions
   B. Operators and Precedence of Operations
   C. Conditional Execution
   D. Nesting and Chaining Conditionals

VIII. More about Iteration
   A. The *for* loop revisited
   B. The *while* statement
   C. Applications and Patterns
   D. Sentinels and Input Validation
   E. Algorithms Revisited

IX. Strings
   A. A Collection Data Type, Indexing
   B. String Methods and Slicing
   C. Traversal Patterns
   D. The *in* and *not in* Operators

X. Lists
   A. Another Collection Type
   B. Concatenation, Repetition, and Element Deletion
   C. Objects and References
   D. Lists and *for* loops
   E. Lists as Parameters and Return values from functions
   F. List Comprehensions
   G. Nested Lists

XI. Files
   A. Working with Data Files
   B. Reading and Writing Text Files
   C. *with* Statements

XII. Dictionaries
   A. Dictionaries and their Operations

B. Aliasing and Copying

XIII. Exceptions
- A. Exception Handling and Flow-of-control
- B. Principals of using Exceptions
- C. Catching Multiple Specific Exceptions
- D. Clean-up after Exceptions

XIV. Recursion
- A. What is Recursion?
- B. The Three Laws of Recursion
- C. Visualizing Recursion

XV. Classes and Objects
- A. Object Oriented Programming, a Change in Perspective
- B. User Defined Classes
- C. Constructors and other Methods
- D. Using Objects as Arguments and Parameters
- E. Instances as Return Values

XVI. Using Classes and Objects
- A. Fractions
- B. Mutating Objects
- C. Sameness
- D. Arithmetic Methods