Course number and name:	**CS 04315: Programming Languages**
Credits and contact hours:	3 credits / 3 contact hours
Course Coordinator:	Nancy Tinkham
Instructional Materials:	Concepts of Programming Languages, Sebesta, 2015. An Introduction to Functional Programming with Scheme, Tinkham, 2008. Programming in Prolog, Clocksin & Mellish, 2003.

Specific course information

Catalog description: A study of the fundamental principles underlying the design of programming languages. Students will study two or more languages from contrasting programming paradigms such as Functional, Object-Oriented, Logical, or Concurrent.

Prerequisites: (CS 04222 Data Structures and Algorithms or
CS 04225 Principles of Data Structures)

and

(CS 06205 Computer Organization or
ECE 09241 Digital I)

Type of Course: ☒ Required ☐ Elective ☐ Selected Elective

Educational objectives for the course

1. **specifying syntax**: Students have demonstrated their ability to use BNF grammars to describe syntactic components of programming languages.
    - ABET (1) Analyze a complex computing problem and to apply principles of computing and other relevant disciplines to identify solutions.

2. **learning new programming languages**: Students have demonstrated their ability to learn new programming languages and to implement solutions to programming problems in two or more different programming languages.
    - ABET (1) Analyze a complex computing problem and to apply principles of computing and other relevant disciplines to identify solutions.

3. **general concepts of programming languages**: Students have demonstrated their understanding of abstract concepts of programming languages, including data types, parameter-passing methods, and control constructs.
    - ABET (1) Analyze a complex computing problem and to apply principles of computing and other relevant disciplines to identify solutions.

Required list of topics to be covered

1. General concepts of programming languages, including: Data types, scope and lifetime of variables, parameter-passing, and control constructs.
2. Formal methods for specifying syntax, including BNF grammars
3. Two or more programming languages illustrating different approaches to programming

Optional list of topics that could be covered

1. History of programming languages
2. Formal methods for specifying semantics
3. The course should include at least two, perhaps more, different programming languages that the students will learn during the semester. There are many possibilities; Scheme, Haskell, Clojure, Prolog, Perl, Erlang, and Ruby are all good choices, for example. Ideally, the languages will illustrate some unusual feature of programming languages that is different from the languages that students have worked with in the past.