1. **Course Number and Course Title:**
   CMP 321 – Programming Languages

2. **Credit Hours:**
   3 – 2 – 3

3. **Prerequisites and/or Co-Requisites:**
   Prerequisite: CMP 256 GUI Design and Programming or COE 312 Software Design for Engineers, and CMP 305 Data Structures and Algorithms
   Co-requisites: CMP 321L

4. **Name and Contact Information of Instructor:**
   Dr. Michel Pasquier

5. **Course Description (Catalog Description):**
   Introduces the fundamental principles and techniques in the design and implementation of modern programming languages. Covers key topics such as syntax and semantics, binding and scope, data types, control structures and expressions. Discusses different programming paradigms, such as imperative, functional, logic, and object-oriented.

6. **Textbook and other Supplemental Material:**
   Textbook:
   - R. Sebesta, *Concepts of Programming Languages*, 11th ed., Addison-Wesley, 2016.

   Supplemental material:
   - Development tools and docs for Python, Java, Scheme, Haskell, Prolog, etc.

7. **Course Learning Outcomes:**
   Upon completion of the course, students will be able to:
   1. Compare the readability, writability, reliability and cost factors of programming languages.
   2. Define language syntax using Backus–Naur Form (BNF) and Extended BNF.
   3. Specify language semantics using tools such as attribute grammar and axiomatic semantics.
   4. Evaluate different approaches to naming, storage binding, scopes, data types, control flow, and expressions.
   5. Implement a lexical analyzer and a recursive-descent parser for a simple language.
   6. Explain the characteristics of functions and lambda expressions and higher-order functions and apply them to solve sample problems using functional programming.
   7. Describe the components and mechanisms of predicate calculus and logical inference and apply them to solve sample problems using logic programming.

8. **Teaching and Learning Methodologies:**
   Methods include lectures, problem-based learning, class discussions, and group work, as well as laboratory sessions. Students learning is assessed via in-class quizzes, exams, homework, and programming assignments/projects.

9. Course Topics and Schedule:

Topic/Activity	**Weeks**
Introduction to programming languages	Week #1
Readability, writability, reliability, cost factors	Week #2
Data types, names, bindings, scopes, control flow	Week #3
Functions and lambda expressions	Week #4
Classes and inheritance, modules, advanced Python	Week #5
Expressions and assignments, subprograms	Week #6
Functional programming, higher-order functions	Week #7
Functional programming, design patterns – Midterm I	Week #8
Functional programming, iterators and generators	Week #9
Language syntax, grammars, BNF/EBNF	Week #10
Regular expressions, regex grammar	Week #11
Language semantics, attribute grammar	Week #12
Logic programming, predicate calculus – Midterm II	Week #13
Logic programming, logical inference, advanced Prolog	Week #14
Revision	Week #15
Final Exam	Week #16