1. **Course Number and Course Title:**

   CMP 321 – Programming Languages

2. **Credit Hours:**

   3 – 2 – 3

3. **Prerequisites and/or Co-Requisites:**

   Prerequisite: CMP 257 Web Application 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:**

   Name: Dr. Michel Pasquier
   Email: mpasquier@aus.edu
   Office: ESB 2083
   Phone: 06 515 2883
   Office Hours: Posted on office door and iLearn; also by appointment

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*, 12th ed., Addison-Wesley, 2019.

   Supplemental material:
   - Development tools, tutorials, and sample programs for Python, Scheme, Haskell, Prolog, etc.

7. **Course Learning Outcomes:**

   Upon completion of the course, students will be able to:
   1. Evaluate the readability, writability, reliability and cost factors of programming languages.
   2. Specify language syntax using Backus–Naur Form (BNF) and Extended BNF.
   3. Model language semantics using tools such as attribute grammar and denotational semantics.
   4. Implement a lexical analyzer and a recursive-descent parser for a simple language.
   5. Compare different approaches to naming, storage bindings, scopes, data types and structures, control flow, and expressions.
   6. Apply functions, lambda expressions, and higher-order functions, to solve sample problems using functional programming.
   7. Employ predicate calculus and logical inference to solve sample problems using declarative 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
Features, evaluation criteria, cost factors	Week #2
Language parsing: syntax, grammars, BNF/EBNF	Week #3
Language parsing: lexical and syntax analysis	Week #4
Language parsing: semantics, attribute grammar	Week #5
Programming: syntax, names, bindings, scope, expressions	Week #6
Programming: data types and structures, control flow	Week #7
Programming: functions and lambda expressions	Week #8
Object-oriented programming: classes, inheritance	Week #9
Object-oriented programming: abstraction, exceptions	Week #10
Functional programming: higher-order functions	Week #11
Functional programming: iterators and generators	Week #12
Language parsing: regular expressions, regex grammar	Week #13
Declarative programming: symbols and predicate calculus	Week #14
Declarative programming: logical inference and reasoning	Week #15
Final Exam	Week #16

10. Schedule of Laboratory and other Non-Lecture Sessions:

Assignment	Due Date (tentative)
Lab 0 – Revision: C++ and Java syntax (not graded)	Week #1
Lab 1 – Revision: C++ and Java programming	Week #2
Lab 2 – Language parsing: lexical analysis	Week #3
Lab 3 – Language parsing: syntax analysis	Week #4
Lab 4 – Language parsing: building parse trees	Week #5
Lab 5 – Introduction to interpreted programming	Week #6
Lab 6 – Programming: slicing and other new features	Week #7
Lab 7 – Programming: data types and operations	Week #8
Lab 8 – Programming: data structures and operations	Week #9
Lab 9 – Programming: functions and lambda expressions	Week #10
Lab 10 – Object-oriented programming: classes, inheritance	Week #11
Lab 11 – Object-oriented programming: abstraction, exceptions	Week #12
Lab 12 – Functional programming: higher-order functions	Week #13
Lab 13 – Functional programming: iterators and generators	Week #14

In the course project, students apply the principles of lexical analysis and syntax analysis studied in chapters 3-4 to implement a parser program for a simple programming language. The project is executed in teams of three students to enhance collaboration and coordination skills. Each team hands in a report comprising their source code, documentation, and test cases.

11. Out-of-Class Assignments with Due Dates:

Assignment	Due Date (tentative)
Homework 1 – Programming languages	Week #4
Homework 2 – Syntax and semantics	Week #7
Homework 3 – Data types, operations, tools	Week #9
Homework 4 – OOP, functions and classes	Week #11
Homework 5 – Functional programming	Week #13
Project – Language parser implementation	Week #14

12. Student Evaluation:

Assessment	Weight	Due Date (tentative)
In-class Quizzes (learning feedback)	(not graded)	(every class)
Homework Assignments	8 %	cf. section 11
Course Project	12 %	cf. section 11
Laboratory Assignments	10 %	cf. section 10
Midterm 1 Exam	18 %	Week #8
Midterm 2 Exam	18 %	Week #13
Final Exam	34 %	Week #16

13. Assessment Instruments:

Assessment	Course Learning Outcomes
In-class Quizzes	O1–O7
Homework and Project	O1–O7
Lab Assignments	O2, O5–O7
Midterm 1 Exam	O1, O5–O6
Midterm 2 Exam	O2, O5–O6
Final Exam	O1–O3, O5–O7

14. Contribution of Course to Program Outcomes:

BSCS Program Outcomes	Emphasis in this course	Course Learning Outcomes
(1) Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions.	◑	O1, O5–O7
(2) Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline.	●	O2–O7
(3) Communicate effectively in a variety of professional contexts.		
(4) Recognize professional responsibilities and make informed judgments in computing practice based on legal and ethical principles.		
(5) Function effectively as a member or leader of a team engaged in activities appropriate to the program's discipline.	○	O4–O7
(6) Apply computer science theory and software development fundamentals to produce computing-based solutions.	◑	O2–O7

Emphasis: ● High; ◑ Medium; ○ Low; Blank – Nothing Specific Expected

15. Letter Grade Policy:

Total (T)	Letter Grade
90 ≤ T	A
85 ≤ T < 90	A-
80 ≤ T < 85	B+
75 ≤ T < 80	B
70 ≤ T < 75	B-
65 ≤ T < 70	C+
60 ≤ T < 65	C
50 ≤ T < 60	C-
40 ≤ T < 50	D
T < 40	F

Refer to the **Course Information and Policies** (i.e., Part II of this Syllabus) for detailed information about Academic Integrity, Homework/Lab/Project Assignments and Exams, as well as the Grading Policy, Attendance Policy, Students Responsibilities, and Appendices.