

0205506 PROGRAMING LANGUAGES

Normal Education

Evening Education

Spring 2017-2018

**Course Format:** face-to-face

**INSTRUCTOR INFORMATION**

**Instructor:**

**Title:**

**Office:**

**Phone:**

**Office Hours:**

**E-mail:**

**COURSE DESCRIPTION**

**Credit hours:** *3 credits (3+0)*

**ECTS**: *5*

**Required or elective:** *Required for Computer Engineering Students*

**Catalog Description:**  *This course provides an opportunity to discuss the concept of the extensive and in-depth programming language. By presenting the design issues of various language constructs, some of the most common languages are intended to provide students with the actual basis for understanding the concept of programming languages and the comparison of design alternatives and alternatives for the constructs.*

**Prerequisites:** *-*

**Textbook(s) and/or required materials:** *Kenneth C. Louden and Kenneth A. Lambert “Programming Languages: Principles and Practice, Third Edition”*

*R. Toal, R. Rivera, A. Schneider, and E. Choe, Programming Language Explorations, CRC Press, 2017.*

*Robert W Sebesta, Concepts of Programming Languages, 11th Edition, Addison-Wesley, 2016.*

**Course Objectives**

|  |
| --- |
| ***The objectives of this course are to:*** |
| *1* | *Teach the student the many of the fundamental concepts that underlie programming language syntax and semantics through a comparative study of several languages and their features* |
| *2* | *Teach the student learn several new programming language features and paradigms* |
| *3* | *Teach the student identify the basic objects and constructs in Object-Oriented Programming.* |
| *4* | *Analyze and evaluate new programming languages and new language features* |

**Course Topics**

|  |  |
| --- | --- |
| ***No*** | ***Topics*** |
| *1* | *Introduction: Why and how to study programming languages; Examples of languages with brief case studies; History and evolution; Programming paradigms; Good, bad, and successful languages.* |
| *2* | *Overview of Selected Languages: Tours of JavaScript, [Ruby | Python], Java, [Clojure | Haskell | Scala | ML], C and C++, [Go | Rust | Swift], with emphasis on new features and means of comparison.* |
| *3* | *Language Specification: Mathematical definition of language; Syntax, semantics and pragmatics; Forms of syntax specification: CFG, BNF, EBNF, other notations; A look at semantic specification; Differences between syntax errors and static semantic errors.* |
| *4* | *Names and Bindings: The meaning of and importance of name, binding, scope (static and dynamic) and extent (static, stack, and heap); Application to constants, variables, types, subroutines, and modules; Shallow vs. deep binding; Closures; Aliasing, overloading, and polymorphism.* |
| *5* | *Types: Type systems; Static vs. dynamic typing, strong vs. weak typing, and manifest vs. implicit typing; Type checking, type equivalence, type coercion, and type inference; Primitive types, numbers, text, enumerations, and pointers; Tuples (products), unions (sums), arrays, sets, streams, regular expressions; Abstract and generic types; Covariance, Contravariance, and Invariance; Construction, assignment, equality-testing, and destruction; Dependent types.* |
| *6* | *Expressions and Statements (Control Flow): Operator precedence, associativity, arity and fixity; Evaluation order, short-circuiting; Structured and unstructured control flow; Sequencing, selection, iteration, recursion and non-determinacy.* |
| *7* | *Subroutines: The runtime stack and activation records; Calling conventions; Passing arguments and returning values;*  |
| *8* | *Midterm Exam* |
| *9* | *Subroutines: Higher-order functions and functional programming; Closures revisited; Pattern matching for function arguments; Exceptions; Coroutines; Generic subroutines.* |
| *10* | *Abstraction, Encapsulation, And Object-Orientation: Modules, abstract data types; Tenets of object-orientation: encapsulation, inheritance and dynamic method binding;*  |
| *11* | *Abstraction, Encapsulation, And Object-Orientation: Modules, abstract data types; Tenets of object-orientation: encapsulation, inheritance and dynamic method binding;* |
| *12* | *Abstraction, Encapsulation, And Object-Orientation: Issues with multiple inheritance; Implementation issues; Class-based vs. prototype-based systems; Pure vs. hybrid object systems.* |
| *13* | *Concurrency: Motivation; Threads vs. events; Asynchronous programming: callbacks, promises, and async/await.* |
| *14* | *Concurrency: Multiprocessing vs. multithreading; Communication and synchronization issues; Shared memory vs. message passing; Language-intrinsic concurrency vs. library managed concurrency; Implementation* |

**Course Learning Outcomes**

*At the end of the course, students;*

* *Master many of the fundamental concepts that underlie programming language syntax and semantics through a comparative study of several languages and their features*
* *Gain the ability to study conceptual linguistic issues without being blinded by a language’s implementation.*
* *Identify the common concepts used to create programming languages.*
* *Compare factors and commands that affect the programming state.*
* *Explain the evolution and key features of the major programming languages.*
* *Analyze and evaluate new programming languages and new language features.*
* *Identify the basic objects and constructs in Object-Oriented Programming.*
* *Explain the characteristics of pure functional functions in functional programming.*
* *Describe the structures and components utilized in logical programming.*

**Evaluation methods**

|  |  |
| --- | --- |
| *1. Midterm Exam* | 40% |
| *2. Final Exam* | 60% |

**Professional component**

|  |  |
| --- | --- |
| *Engineering topics* | 100% |
| *General education* | 0% |
| *Mathematics and basic sciences* | 0% |

**Person(s) who prepared this description and date of preparation**

*Enes Ayan, April 2018*

**Date of last revision**

*April 2018*