THOMAS ADEWUMI UNIVERSITY, OKO-IRESE	
Faculty	Computing and Applied Sciences
Department	Mathematical and Computing Science
Program	Computer Science
Course Code	CSC 401
Course Title	**ORGANIZATION OF PROGRAMMING LANGUAGES**
Study Year	4
Credit Hours	3
Contact Hours	36
Pre-requisite	
Status	Compulsory
Semester	First
Mode of Assessment	Lecture, Assessment and Practical
Mode of Delivery	• Classroom Lectures • Laboratory Practical Sessions
Continuous Assessment **Examination** **Total**	30% 70% 100%
Course Lecturer and Instructor	
Course Description	The organization of programming languages refers to how programming languages can be classified and categorized based on their features and characteristics. There are several ways to organize programming languages, including: Programming Paradigms: Programming languages can be classified based on their programming paradigm, which refers to the way they support and facilitate the creation of software. The main programming paradigms are procedural, object-oriented, functional, logical, and concurrent. Type System: Programming languages can also be classified based on their type system, which refers to how they handle data types and their compatibility. The main type systems are static and dynamic typing. Syntax: Programming languages can be classified based on their syntax, which refers to the grammar and structure of the language. The main syntax types are imperative, declarative, and functional.

	Abstraction: Programming languages can also be classified based on their level of abstraction, which refers to how high or low-level the language is. The main levels of abstraction are low-level languages, like assembly and machine languages, and high-level languages, like Python and Java. Application Domain: Programming languages can also be classified based on their application domain, which refers to the types of applications they are best suited for. For example, languages like R and Matlab are suited for scientific and data analysis applications, while languages like JavaScript and PHP are suited for web development.
Course Objectives	1. Understand the features and characteristics of programming languages: By studying the organization of programming languages, programmers can gain a deeper understanding of the different features and characteristics of different programming languages. This knowledge can help them choose the best language for a particular task or project. 2. Compares programming languages: By organizing programming languages based on their features and characteristics, programmers can compare different programming languages and make informed decisions about which language to use for a particular task or project. 3. Understand the evolution of programming languages: By studying the organization of programming languages, programmers can gain an appreciation for how programming languages have evolved over time and how new features and characteristics have been added to them. 4. Improve programming skills: Studying the organization of programming languages can help programmers to improve their programming skills by giving them a deeper understanding of programming concepts and principles. 5. Prepare for learning new programming languages: By studying the organization of programming languages, programmers can prepare themselves for learning new programming languages more easily. They can use their knowledge of the features and characteristics of different languages to understand new languages more quickly and easily.
Learning Outcome	• By studying the organization of programming languages, students will gain knowledge of the different programming paradigms, including procedural, object-oriented, functional, logical, and concurrent. They will be able to identify the strengths and weaknesses of each paradigm and understand when to use each one. • Students will gain an understanding of the different type systems used in programming languages, including static and dynamic typing. They will learn how type systems affect the reliability and maintainability of code. • Students will be able to analyze the syntax of programming languages and understand how it affects the readability and maintainability of code.

	• Students will gain an understanding of the different levels of abstraction in programming languages and be able to identify when to use high-level or low-level languages for different tasks. • By studying the organization of programming languages, students will gain knowledge of the different application domains for programming languages. They will be able to identify which languages are best suited for different types of applications. • Students will be able to compare programming languages and evaluate their suitability for different tasks based on their features and characteristics. • By studying the organization of programming languages, students will be better prepared to learn new programming languages more easily and quickly.	
Detailed course contents	Language definition structure. Data types and structures, Review of basic data types, including lists and trees, control structure and data flow, Run-time consideration, interpretative languages, lexical analysis and parsing.	

Course Contents Sequencing

Weeks	**Detailed Course Outline**	**Allocated Time**
WEEK 1	• Overview of programming languages • History and evolution of programming languages • Language classification	3 Hours
WEEK 2	Syntax and Semantics • Syntax diagrams and regular expressions • Context-free grammars • Semantics of programming languages • Data types, expressions, control structures, functions, and parameter passing	3 Hours
WEEK 3, 4	Programming Paradigms • Procedural programming • Object-oriented programming • Functional programming • Logic programming • Concurrent programming **C.A Test**	6 Hours

WEEK 5, 6	Type Systems • Type systems and their role in programming languages • Static typing and dynamic typing • Type inference and type checking	6 Hours
WEEK 7,8	Language Implementation • Lexical analysis and scanning • Parsing and syntax analysis • Intermediate representations and code generation • Optimization techniques	6 Hours
WEEK 9, 10	Domain-Specific Languages • Embedded and domain-specific languages • Application of domain-specific languages • Creating a domain-specific language **C.A Test**	6 Hours
WEEK 11	• Language implementation: parsing • Syntax diagrams and regular expression • Language implementation: code generation • Intermediate representations and optimization	3 Hours
WEEK 12	• Domain-specific languages • Embedded and domain-specific language • Comparison of programming languages • Evaluation of programming languages for different application domains	3 Hours
	REVISION	

READING LIST:

- Programming Language Pragmatics by Michael L. Scott:
- Concepts of Programming Languages by Robert W. Sebesta:
- Types and Programming Languages by Benjamin C. Pierce:
- Programming Languages: Design and Implementation by Terrence W. Pratt and Marvin V. Zelkowitz:
- Essentials of Programming Languages by Daniel P. Friedman and Mitchell Wand:
- Programming Language Foundations by Benjamin C. Pierce, et al.:
- Language Implementation Patterns by Terence Parr:

- Programming Languages – An Introductory text on concepts and principles by E.K. Olatunji ,2014
-