Semester: 1				
Programme: Data Science				
Course: Introduction to Programming				
Paper code: M1DS250111P			**Credits: 3**	
Hours/week: 3 (3 Pr)				
Category: Core/MDC/SEC/VAC: Multi-Disciplinary				
Theory / Practical / Composite: Practical				
No of Module: 1				
Course Outcome:				
1. **Remember** fundamental C programming syntax, including data types, variables, constants, and operators, alongside basic algorithmic terminology such as flowcharts and pseudocode.				
2. **Understand** the principles of software design, the conceptual process of problem decomposition, and the structural role of programs and Integrated Development Environments (IDEs) in computer science.				
3. **Apply** sequential, conditional, and iterative statements, along with standard input/output functions, to implement logic and solve elementary programming problems.				
4. **Analyze** computational tasks by developing effective algorithms and flowcharts to determine optimal problem-solving strategies and logical flow.				
5. **Evaluate** program performance and code quality through the use of debugging tools and software design principles to identify and resolve logic and syntax errors.				
6. **Create** modular and well-structured software solutions by synthesizing functions with parameters, return values, and array concepts to address complex algorithmic challenges.				
Prerequisites: ***Basic knowledge about any prior course***				
SYLLABUS				
UNIT	**CONTENT**	**HOURS or NUMBER OF CLASSES**	**CO Mapping**	**COGNITIVE LEVEL**
1.	**Algorithmic Concepts** Defining programming and its role in Computer Science. Flowcharts, Algorithms and Pseudocode. Basic Structure of Programs. Introduction to Integrated Development Environments and their use. [4L] Problem-Solving and Algorithm Development. Problem Decomposition. [2L] Debugging Techniques: Using debugging tools and strategies to identify and fix errors in code. [2L] Software Design: Understanding basic software design principles.	**10**	**CO1, CO2, CO4, CO5**	**K1, K2, K4, K6**
2.	**Basic Programming**	**26**	**CO1, CO3, CO6**	**K1, K3, K6**

	Data Types, variables, constants, operators and expressions. [6L] Input/Output: Understanding how to get input from the user and display output. [2L] Conditional statements and loops. [8L] Function definition and function calling. Parameters and return values. Using parameters to pass data to functions and understanding return values. [6L] Arrays: Basic Concepts. [4L]			

Suggested readings
1. The C Programming Language, Kernighan and Ritchie, PHI Publications.
2. Programming with C, Gottfried, TMH Publications.
3. Programming in C, Dey and Ghosh, Oxford Publications.
4. Programming in ANSI C, Balaguruswamy, McGraw Hill.
5. Practical Julia: A Hands-On Introduction for Scientific Minds, Lee Phillips, No Starch Press.
6. Hands-On Julia Programming: An Authoritative Guide to the Production-Ready Systems in Julia, Sambit Kumar Dash, BPB Publications.
7. Computer Fundamentals, Sinha and Sinha, BPB Publications.
8. Fundamentals of Computers, Rajaraman and Adabala, PHI Publications
Web Resources
1. NPTEL course on Introduction to Programming in C by Dr. Satyadev Nandakumar, IIT Kanpur; course link: https://youtu.be/XTiliI-LOY8

Evaluation	**Continuous Assessment**	

Course outcomes (COs) and Cognitive Level Mapping

COs	**CO Description**	**Cognitive levels**
CO1	**Remember** fundamental C programming syntax, including data types, variables, constants, and operators, alongside basic algorithmic terminology such as flowcharts and pseudocode.	K1
CO2	**Understand** the principles of software design, the conceptual process of problem decomposition, and the structural role of programs and Integrated Development Environments (IDEs) in computer science.	K2
CO3	**Apply** sequential, conditional, and iterative statements, along with standard input/output functions, to implement logic and solve elementary programming problems.	K3
CO4	**Analyze** computational tasks by developing effective algorithms and flowcharts to determine optimal problem-solving strategies and logical flow.	K4
CO5	**Evaluate** program performance and code quality through the use of debugging tools and software design principles to identify and resolve logic and syntax errors.	K5
CO6	**Create** modular and well-structured software solutions by synthesizing functions with parameters, return values, and array concepts to address complex algorithmic challenges.	K6