MODULE 16	PROGRAMMING PARADIGMS	
CREDIT POINTS	7.5	
STATUS	Core	
ASSESSMENT	Continuous Assessment	70%
	Examination	30%
TOTAL CONTACT HOURS: 72		
Lecture: 36	Practical: 36	
Tutorial:	Other:	
TOTAL STUDENT EFFORT: 150		

Aims

Functional programming is a style of programming in which expressions and functions are the major elements. This contrasts with imperative and, to a certain extent, object-oriented programming in which commands play a major role. The notation of functional languages tends to be very simple and very powerful—functional programs are typically shorter and simpler than imperative programs for the same task, and can often be developed more quickly. This module is an introduction to programming and problem-solving in a functional style. We start from scratch but we move at a brisker pace than an introductory course. We cover all the fundamentals and see how the language can be put to work in the design of challenging and fun applications, including music, graphics, and web-based systems. We will see how imperative and object-oriented aspects such as commands, encapsulation, and exceptions are addressed in a modern functional language. Functional programming is finding more and more application in industry but, even if you don't program in a functional language when you finish this module, your general programming and problem-solving skills will be greatly enhanced by it. This module will change the way you think about computing science and software development.

Learning Outcomes

Upon successful completion of this module, you will have demonstrated the ability to:

1   design functional programs in a systematic, well-structured manner

2   explain the elements of functional programming clearly, exemplifying them with simple examples

3   implement imperative and object-oriented notions in a functional style

4   read, understand, and modify existing functional programs

5   develop non-trivial applications as systems of functional programs

6   implement and test your programs using a functional programming system.

Indicative Content

Topic	Description
Introduction and motivation	What is functional programming? Relationship with imperative programming. Benefits of programming in a functional style. Brief history and case studies of successful industrial applications. How to learn and do functional programming.
Elements of functional programming	Expressions. Evaluation (lazy versus eager). Definitions. Cases. Basic datatypes. Patterns. Recursion. Inductive proof. Lists. Designing functional programs. Examples, case studies, and exercises. Using a functional programming system. Interpretation. Compilation.
Modular programming	Modules. Abstract datatypes. Examples: sets, dictionaries, binary search trees. Generic programming. Commands. Exceptions.
Applications	A selection of non-trivial case studies and mini-projects, e.g. from some of the following areas: graphics, multimedia, games, symbolic computation, simulation, web applications.