

## Liverpool John Moores University

Title: PROGRAMMING LANGUAGE THEORY  
Status: Definitive  
Code: **5129COMP** (121256)  
Version Start Date: 01-08-2021

Owning School/Faculty: Computer Science and Mathematics  
Teaching School/Faculty: Computer Science and Mathematics

Team	Leader
Paul Bell	Y

**Academic Level:** FHEQ5  
**Credit Value:** 20  
**Total Delivered Hours:** 57  
**Total Learning Hours:** 200  
**Private Study:** 143

### Delivery Options

Course typically offered: Semester 1

Component	Contact Hours
Lecture	33
Practical	22

**Grading Basis:** 40 %

### Assessment Details

Category	Short Description	Description	Weighting (%)	Exam Duration
Artefacts	AS1	Design and implementation of basic language, interpreter and associated report	60	
Exam	AS2	Exam	40	2

### Aims

*The module's aim is to provide an introduction to the concepts behind programming languages, along with an explanation of the underpinnings of programmable machines. It will also discuss and demonstrate a variety of programming languages across both Imperative and Declarative paradigms*

## Learning Outcomes

After completing the module the student should be able to:

- 1 Explain the key concepts in specifying and evaluating a programming language.
- 2 Apply appropriate formal methods to specify a programming language
- 3 Design and implement an interpreter/compiler for a simple imperative programming language
- 4 Appraise Imperative and Declarative programming paradigms as an appropriate mechanism for a variety of problem domains

## Learning Outcomes of Assessments

The assessment item list is assessed via the learning outcomes listed:

Basic language, interpreter Examination	1	2	3	4
	1	2		

## Outline Syllabus

*Abstract views of program execution:*

*FSMs, the interlock machine*

*Turing Machines; the computing machine; addition of configurable memory*

*Defining and processing a language:*

*Fundamental Language Theory: Grammars and Syntax*

*Fundamental Compiler Theory: Lexical, Syntactical and Semantic Analysis*

*Parse/syntax trees and binding*

*Language Paradigms:*

*Imperative vs Declarative Languages*

*State transformations vs. Referential transparency*

*How vs Why & why is this important?*

## Learning Activities

Lab-Lectures, Directed Study Tasks

## Notes

This module combines theory and practical work to familiarise a student with the fundamentals of programming languages and their compilation / interpretation for execution; culminating with the student specifying and designing their own basic imperative language.

It will make use of programming-related skills from previous modules; particularly functional decomposition and basic data design. It will, in parts, reinforce and

present alternative use cases for materials covered in Data Structures.

The module will assume that the student is already with some fundamentals of imperative programming, namely:

-variable declaration, state modification and scope.

-how to design and write software that correctly uses sequential execution, branching, iteration and function-calling.