Principles of Programming (COMP0002)

**Description**

**Aims:**
This module provides an introduction to computer programming using two different programming styles, imperative and functional programming. Its primary aim is to develop core design, programming, and problem-solving skills, with a secondary aim of building confidence in the ability to take on and learn new programming languages within a short space of time. In order to achieve these aims the module has a substantial practical element in the form of compulsory lab classes where students work through sets of programming exercises to apply the programming concepts introduced during the module lectures.

**Learning outcomes:**
On successful completion of the module, a student will be able to:

1. Design, implement, debug and test small programs using two different programming paradigms, given straightforward specifications;
2. Develop straightforward algorithms to solve a range of common programming problems;
3. Compare and contrast the different paradigms, understanding the relative advantages and disadvantages of each;
4. Use common programming tools such as compilers, editors and debuggers;
5. Gain a basic understanding of how programs use computer memory and of memory management, and the binary representation of data values.

**Content:**
Core Programming Concepts:
- What is programming?
- Programming languages

**Key information**

- **Year:** 2019/20
- **Credit value:** 15 (150 study hours)
- **Delivery:** UG L4, Campus-based
- **Reading List:** [View on UCL website](https://www.ucl.ac.uk)
- **Tutor:** Dr Graham Roberts
- **Term:** Term 1
- **Timetable:** [View on UCL website](https://www.ucl.ac.uk)

**Assessment**

- Written examination (main exam period): 80%
- Coursework: 10%
- Coursework: 10%

**Find out more**
For more information about the department, programmes, relevant open days and to browse other modules, visit [ucl.ac.uk](https://www.ucl.ac.uk)
• Programs and algorithms
• Compilers and tools
• Running and debugging programs

Introduction to Imperative Programming:
• Core imperative programming ideas: sequence, selection, iteration, recursion, assignment and variables
• Types and type checking
• Functions, parameters, scope and lifetime
• Arrays and files
• Pointers, memory allocation, and memory management

Introduction to Functional Programming:
• Core functional programming ideas
• Recursion and recursive data structures such as lists
• Lazy evaluation
• Role of types
• Higher order functions
• List comprehension
• Recursive functions

Program design in the small:
• Designing and implementing small programs
• Implementing and using basic algorithms and data structures
• Good programming and design practice

Requisites:
In order to be eligible to select this module, a student must be registered on a programme for which it is a formally-approved option or elective choice.