# MODULE SPECIFICATION

| Part 1: Information | | | |
|---|---|---|---|
| Module Title | Principles of Programming | | |
| Module Code | UFCFHS-30-1 | Level | Level 4 |
| For implementation from | 2020-21 | | |
| UWE Credit Rating | 30 | ECTS Credit Rating | 15 |
| Faculty | Faculty of Environment & Technology | Field | Computer Science and Creative Technologies |
| Department | FET Dept of Computer Sci & Creative Tech | | |
| Contributes towards | Computer Science BSc (Hons) 2020-21<br>Computing [Sep][FT][Frenchay][3yrs] BSc (Hons) 2020-21 | | |
| Module type: | Standard | | |
| Pre-requisites | None | | |
| Excluded Combinations | None | | |
| Co- requisites | None | | |
| Module Entry requirements | None | | |

| Part 2: Description |
|---|
| This module introduces the basic programming concepts, techniques, and fundamental problem solving and software development principles. Topics include variables and assignment, primitive and complex data types, control structures, iteration, functions, recursion, arrays, pointers, memory management. It also introduces programming paradigms, for instance, fundamental ideas and principles behind the object-oriented approach to programming, including classes, objects, encapsulation, abstraction, inheritance, and polymorphism. The module also teaches the basics of functional programming and introduces basic software development topics such as design, testing, and debugging. It also addresses aspects like vulnerabilities, defects and bugs in code so that students are aware of potential threats and attacks. This module suits for both the absolute beginners and those who have no prior programming experience. |

**Educational Aims:** Through a practice-led approach, this module aims to give students a solid foundational knowledge and understanding of how programming languages are used to develop software.

**Outline Syllabus:** Syllabus Outline:
Introduction to Imperative Programming
Core imperative programming concepts: sequences, selection, iteration, recursion, assignment, constants and variables. Data types, Functions, Parameters and scope of variables, Arrays, Pointers and memory management, File I/O

Program design
Problem solving and program design with pseudo code (thinking algorithmically). Developing small programs using functions. Designing and implementing simple algorithms and data structures, algorithm or program complexity (Big-O notations). Basic software development techniques: Implementation, debugging, testing, code review and version control using standard IDEs such as NetBeans or Eclipse

Functional programming
Programming with a functional notation, sessions and scripts, lambda functions, call-back functions, expressions and values. Evaluation of expressions, Lists and their operations, Recursion and induction.

Object-Oriented Principles
Fundamental principles and concepts of object-oriented programming; Abstract data types; Class Anatomy and its object creation; Encapsulation; Overloading principle; Object interaction; Abstract classes and Interfaces; Inheritance; Basic polymorphism; Overriding principle; Static and Dynamic Binding; Exception handling; Design, code and test a series of object-oriented programs.

**Teaching and Learning Methods:** This module will principally be delivered as combination of lectures and practical sessions with some occasional tutorials and seminars. Students are expected to attend all scheduled classes. We encourage students to be active in their learning. We provide a range of resources and activities to enable them to engage in achieving the learning outcomes.
The lectures will explain theoretical concepts. The theory will be underpinned by practical sessions during which the students will solve problems and write and experiment with programme code to implement those solutions.
The module will be supported by the University's VLE which will be used as a repository for course materials, a forum for discussion and, from time to time, tests and/or quizzes to enable the students to self-test their knowledge.

**Part 3: Assessment**

The assessment on this L4 module has been designed to build the students' confidence in writing software. Component A is a series of in-class tests of increasing difficulty. Taken together, the tests result in a portfolio of programming exercises to give the students a tangible sense of how they have developed.

The format of the controlled condition assessment is also designed to help the students build their confidence in the assessment process. For this early module, rather than running a formal exam the controlled condition assessment, Component A of both the main sit and the resit, takes place in the classroom setting.

Component B: A individual coursework assignment of problem solving, algorithm design and program implementation including debugging and testing. Assessment of this component will be an assignment portfolio submission including, design, coding and testing documents and an in-class software demonstration. The resit for this component will be design, coding and testing of a software system submitted as an individual portfolio with supporting software.

Students will have the opportunity for formative feedback during practical lab/tutorial sessions.

| First Sit Components | **Final Assessment** | **Element weighting** | **Description** |
|---|---|---|---|
| Portfolio - Component B | ✓ | 50 % | An individual coursework software development assignment – (submitted online). Assessment by an in-class demonstration and submitted portfolio. |
| In-class test - Component A | | 50 % | A portfolio of unseen, in-class programming exercises. |
| Resit Components | **Final Assessment** | **Element weighting** | **Description** |
| Portfolio - Component B | | 50 % | In this individual portfolio students will design and implement software. |
| In-class test - Component A | ✓ | 50 % | A portfolio of unseen, programming exercises within the computer labs. |

**Part 4: Teaching and Learning Methods**

Learning Outcomes

On successful completion of this module students will be able to:

| | **Module Learning Outcomes** |
|---|---|
| MO1 | Acquire and apply knowledge of a range of essential programming language concepts and assess their effectiveness for a given problem; (assessed in Component A+B) |
| MO2 | Develop and express solutions algorithmically and programmatically to solve simple problems; (assessed in Component A) |
| MO3 | Explain the underlying principles and concepts of Functional Programming and code an application; (assessed in Component A) |
| MO4 | Apply knowledge of basic concepts and principles of object-orientation such as objects and classes, encapsulation, object state and modularity; (assessed in Component B) |
| MO5 | Apply good programming style and interpret the impact of style on developing, testing, debugging and maintaining programs; (assessed in Component A+B) |
| MO6 | Evaluate the appropriateness of different programming paradigms for a given application (assessed in Component A+B) |

Contact Hours

| **Contact Hours** | |
|---|---|
| | |
| **Independent Study Hours:** | |
| Independent study/self-guided study | 228 |
| **Total Independent Study Hours:** | 228 |

| | | |
|---|---|---|
| | **Scheduled Learning and Teaching Hours:** | |
| | Face-to-face learning | 72 |
| | **Total Scheduled Learning and Teaching Hours:** | 72 |
| | **Hours to be allocated** | 300 |
| | **Allocated Hours** | 300 |
| Reading List | *The reading list for this module can be accessed via the following link:*<br><br>https://rl.talis.com/3/uwe/lists/37BC30DE-F9D6-2EF5-DD13-DC733C42F035.html?lang=en-GB&login=1 | |