MODULE SPECIFICATION

Part 1: Information			
Module Title	Computational Thinking and Practice		
Module Code	UFCFQN-30-0	Level	Level 3
For implementation from	2019-20		
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		
Module type:	Standard		
Pre-requisites	None		
Excluded Combinations	None		
Co- requisites	None		
Module Entry requirements	None		

Part 2: Description

Educational Aims: In this module, you will acquire the skills necessary for identifying and modelling a problem into a computational solution. You will then practice on how to transform a computational solution into a computing program using a programming language.

Outline Syllabus: The module will cover:

What is computational thinking: abstract thinking, algorithmic thinking, logical thinking, scalable thinking

Computational thinking for everyone and everywhere

Process of problem solving:
Formulating a problem that could be solved through computation;
Logically organizing and analysing data;
Representing data through models and simulations;
Generalizing to a wide variety of problems;
Searching for the most efficient and effective solution.

Design and implement software through algorithmic thinking:

Approaches and methods for eliciting what is required for programming;
Design techniques;
Programming language constructs and data types;
Strategy, methods and techniques for developing testing.

Teaching and Learning Methods: The module will be delivered via a combination of lectorial/workshop and lab sessions, with face-to-face and online help provided by tutors. Online resources such as Lynda.com are also available.

The first half of the module will focus upon computational thinking and the second half will focus more on programming. The transition from computational thinking to programming will be interleaved and iterated.

For the computational thinking part, there is an emphasis on group discussions and collaborations to re-enforce each student's understanding. In the lectorial/workshop sessions, students will be working both individually and in groups on the tasks set by the tutors. Students from different groups will be given opportunities to share their insights with the rest of the class. The assessment for this part will be group-based and supported by work that is introduced during the module when the students will work on problem(s) taken from their own experience. For example, they might be asked to consider algorithmic aspects of the provision of catering at the University or transport services or allocating resources to student societies.

For the programming part, the learning is more focused on developing the individual student's ability to program. The assessment for the programming part will be individual in-class tests and an exam. The in-class test will act as a preparation for a more formal exam which will in turn, be a preparation for the intense summer exam period. In the workshop/lab sessions, students will use their computational thinking skills to derive solutions and to implement the solutions using a programming language.

Part 3: Assessment

Assessment will comprise two components: a portfolio (Component B); and a pair of programming tests under controlled conditions (Component A)

The assessment regime has been designed to ensure that students' understanding and skills are developed incrementally in a supported way, and that the students' experience is developed in a social and collaborative atmosphere, in accordance with the programme aims.

The portfolio in Component B will consist of outputs from group tasks. These tasks will be introduced to the students incrementally during the first half of the module in the workshops. Each group will produce an output for each task. The output must address how they have formulated and modelled problems, and what strategies were applied to derive the solution. Members from the same group will get the same mark for each output. However mark adjustment may take place where there is an evidence of significant unbalanced contributions from the group members.

Component A is used to assess the programming aspect. This is done as one in-class test and an exam.

This phased approach will enable both the tutors and the students to evaluate the students' understanding after each phase. It will also allow the students to progress from controlled conditions in the familiar environment of the classroom, taken with their immediate peer group to the more formal setting on an exam with the whole cohort. This form of assessment will provide opportunities for students to learn in an incremental way.

During the first phase students will be given formative feedback leading to the test at the end of that phase. Feedback from the summative assessment will help to inform students' learning in the build up to exam.

For resit, component A will be an individual 2 hour programming exam. Component B will be an individual task. The nature of the work will be same as in the main run. The size of the task will be scaled appropriately.

First Sit Components	Final Assessment	Element weighting	Description
Portfolio - Component B		50 %	Portfolio of group outputs
In-class test - Component A		13 %	Individual in-class programming test (1 hour)
Examination - Component A	✓	37 %	Examination (1 hour)
Resit Components	**Final Assessment**	**Element weighting**	**Description**
Written Assignment - Component B		50 %	Individual coursework
Examination - Component A	✓	50 %	Exam (2 hours)

Part 4: Teaching and Learning Methods

Learning Outcomes

On successful completion of this module students will achieve the following learning outcomes:

Module Learning Outcomes	Reference
Formulate and model problems in various task domains, identifying significant features and how to apply an appropriate strategy to finding solutions.	MO1
Translate solution to a problem into a computable form, and evaluate its effectiveness.	MO2
Explore methods to represent solutions as a formal process, such as instructions, algorithms or pseudocode.	MO3
Develop and evaluate ways of representing solutions as code, using basic programming constructs, data types and test methods.	MO4

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:

https://uwe.rl.talis.com/modules/ufcfqn-30-0.html

Part 5: Contributes Towards
This module contributes towards the following programmes of study: Software Engineering for Business {Foundation} [Sep][FT][Frenchay][4yrs] BSc (Hons) 2019-20 Software Engineering for Business {Foundation} [Sep][SW][Frenchay][5yrs] BSc (Hons) 2019-20 Computing {Foundation} [Sep][SW][Frenchay][5yrs] BSc (Hons) 2019-20 Computing {Foundation} [Sep][FT][Frenchay][4yrs] BSc (Hons) 2019-20