National Workshop 1
Key Messages

• Computer Science is for all
• There are many ways to use the specification
• All Learning Outcomes are interwoven and can be learned in any order
• Centrality of ALTs / Computational Thinking
• Digital Technologies can enhance collaboration, learning and reflection
Session 2 – Specification & Learning Outcomes
LCCS Interwoven
‘Learning outcomes can best be defined as statements of what a learner knows, understands and is able to do after completion of learning.’

CEDEFOP (2009)
Learning outcomes have become ubiquitous within worldwide curriculum policy in recent years. This move comes with many potential benefits, as it shifts the focus from providers to users of education, and it introduces a common language, addressing issues of progression, transparency and equity (CEDEFOP, 2009).

(Mark Priestly, Univ of Stirling)

Caveat: Use / History / Skills
Teaching and Learning in Computing
Teaching and Learning in Computing - 2

- New Specification
- Learning Outcomes
- Think / Pair / Share
- Differentiation
- Group work
- Independent / student-centered
Learning Outcomes

Learning outcomes are direct statements that describe the knowledge, skills, and habits of mind that students are expected to reliably demonstrate after a learning experience.

Plan instructional activities
What learning experiences will I design to help them achieve the intended outcomes?

Verbs are important
Use action verbs like:
- design
- create
- diagnose
- evaluate
- extrapolate
- predict

Identify desired learning outcomes
What knowledge, skills, and habits of mind do I want students to demonstrate?

Design suitable assessment strategies
How will I know students have achieved those outcomes?

Get the students on board with learning outcomes
How will I share the learning outcomes with students?
Group Activity
Group Warmup Activity - Instructions

Log onto the shared document and examine the learning outcomes and how they are grouped by strands of the course and by lower and higher order thinking.

https://tinyurl.com/ybx9cjgw
<table>
<thead>
<tr>
<th>Computational Thinking</th>
<th>Computers and Society</th>
<th>Higher-Order Thinking</th>
<th>Evaluation and Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mathematical Thinking</td>
<td>Inferencing &amp; Decision Making</td>
<td>Lower-Order Thinking</td>
<td>Data Skills</td>
</tr>
<tr>
<td>Interactive Information Systems</td>
<td>Analytics</td>
<td>Model Building &amp; Simulation</td>
<td>Information Systems</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Lower-Order Thinking</th>
<th>Higher-Order Thinking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remember - recall facts and basic concepts</td>
<td>Apply - use information in new situations</td>
</tr>
<tr>
<td>Understand - explain ideas or concepts</td>
<td>Analyze - draw connections among ideas</td>
</tr>
<tr>
<td>Evaluate - justify a stand or decision</td>
<td>Create - produce new or original work</td>
</tr>
</tbody>
</table>

### 1.1. Describe a systematic process for solving problems and making decisions

- 1.1. Describe a systematic process for solving problems and making decisions. Develop solutions to computational problems using a computational approach or an iterative approach. Evaluate the efficiency and effectiveness of your solutions.

### 1.2. Explain the power of computing to make decisions

- 1.2. Explain how computing can make decisions. Use computers to analyze large amounts of data and make informed decisions.

### 1.3. Explain the operation of a variety of algorithms

- 1.3. Explain the operation of a variety of algorithms. Understand the steps involved in executing an algorithm.

### 1.4. Discuss when heuristics should and should not be used

- 1.4. Discuss when heuristics should and should not be used. Understand the limitations and advantages of using heuristics.

### 1.5. Discuss the complexity relationship between algorithmic cost and social issues involving issues of ethics

- 1.5. Discuss the complexity relationship between algorithmic cost and social issues involving issues of ethics. Understand the implications of algorithmic cost on social issues.

### 1.6. Identify important computing developments that have taken place in the last 50 years

- 1.6. Identify important computing developments that have taken place in the last 50 years. Understand the impact of these developments on society.

### 1.7. Explain when and what machine learning algorithms might be used in certain contexts

- 1.7. Explain when and what machine learning algorithms might be used in certain contexts. Understand the role of machine learning in specific contexts.

### 1.8. Discuss the ethical and social implications of machine learning

- 1.8. Discuss the ethical and social implications of machine learning. Understand the potential impacts on society.

### 1.9. Implement algorithms using a programming language

- 1.9. Implement algorithms using a programming language to solve a range of problems. Understand the syntax and semantics of programming languages.

### 1.10. Reflect and communicate on the design and development process

- 1.10. Reflect and communicate on the design and development process. Understand the role of communication in the software development lifecycle.

### 1.11. Assess the impact of computing on society and cultural norms

- 1.11. Assess the impact of computing on society and cultural norms. Understand the role of computing in shaping society.

### 1.12. Read, write, test, and modify computer programs

- 1.12. Read, write, test, and modify computer programs. Understand the role of programming in software development.

### 1.13. Debug programs to remove errors and improve efficiency

- 1.13. Debug programs to remove errors and improve efficiency. Understand the role of debugging in software development.

### 1.14. Explain the importance of documenting source code

- 1.14. Explain the importance of documenting source code. Understand the role of documentation in software maintenance.

### 1.15. Analyze the role of testing in ensuring the quality of software

- 1.15. Analyze the role of testing in ensuring the quality of software. Understand the role of testing in software quality assurance.

### 1.16. Implement and evaluate software applications using embedded systems

- 1.16. Implement and evaluate software applications using embedded systems. Understand the role of embedded systems in software development.

### 1.17. Design and implement software applications using object-oriented programming

- 1.17. Design and implement software applications using object-oriented programming. Understand the role of object-oriented programming in software development.

### 1.18. Select and use appropriate software tools and development environments

- 1.18. Select and use appropriate software tools and development environments. Understand the role of software tools in development.

### 1.19. Use abstraction to describe systems and explain the relationship between various parts

- 1.19. Use abstraction to describe systems and explain the relationship between various parts. Understand the role of abstraction in system design.

### 1.20. Use a range of methods for identifying patterns and abstract common features

- 1.20. Use a range of methods for identifying patterns and abstract common features. Understand the role of pattern recognition in software development.

### 1.21. Implement modular designs to develop applications and maintain them over time

- 1.21. Implement modular designs to develop applications and maintain them over time. Understand the role of modularity in software development.

### 1.22. Use algorithms to analyze and interpret data in a variety of contexts

- 1.22. Use algorithms to analyze and interpret data in a variety of contexts. Understand the role of algorithms in data analysis.

### 1.23. Use appropriate programming languages to develop applications that perform a variety of tasks

- 1.23. Use appropriate programming languages to develop applications that perform a variety of tasks. Understand the role of programming languages in software development.

### 1.24. Use appropriate programming languages to develop applications that perform a variety of tasks

- 1.24. Use appropriate programming languages to develop applications that perform a variety of tasks. Understand the role of programming languages in software development.

### 1.25. Use appropriate programming languages to develop applications that perform a variety of tasks

- 1.25. Use appropriate programming languages to develop applications that perform a variety of tasks. Understand the role of programming languages in software development.
Group Activity A - Instructions

Each group should pick 2 different learning outcomes (one from strand 3 and one from strand 1 or 2).

Discuss and write down:

• What would you teach to your class for this LO?
• What teaching & learning strategies could you use?
• How would you know this learning outcome has been achieved?
• Can it be linked to other parts of the course?
Benefits of Learning Outcomes for Teachers

**Effective course design**
- By keeping learning outcomes front and center, teachers can develop courses in which all aspects of the course, including learning activities and assessments, support what they want students to learn [a].

**Effective assessment of learning**
- Clear expectations make it easier to evaluate students’ progress and ensure that assessments are targeting the appropriate level of knowledge or skill [a, b].

**Better time management**
- Well-defined learning outcomes simplify difficult decisions about what content to include and what to omit when preparing lessons and assessments [b, c].

**Improved communication**
- Teachers can use learning outcomes to have explicit and constructive dialogues with students about the course and their learning, and with colleagues about the expectations of courses [b].

**Improved teaching experience**
- Teachers who use learning objectives report less anxiety, more confidence interacting with students, and use more diverse teaching and assessment approaches [b, c].

---


Created by Sara M. Fulmer
Strands and learning outcomes

Computer Science
- Computer Science: Home
- Introduction
- Senior Cycle
- Rationale
- Aim and objectives
- Related Learning
- Structure of Learning
- Certificate Computer Science
- Key Skills of Senior Cycle
- Teaching and learning
- Assessment

Appendix A: Glossary of Action Verbs used
Appendix B: Glossary of Core Concepts

Strand 1: Practices and principles

Strand 2: Core concepts

Strand 3: Computer science in practice

Computer science in practice provides multiple opportunities for students to use their conceptual understanding in practical applications. Over the two years of the course, students engage with four team-based applied learning tasks. Student groups plan, design and develop computational artefacts that are personally relevant or beneficial to their community and society in general. Examples of computational artefacts include programs, games, simulations, visualisations, digital animations, robotic systems, and apps. Students are expected to document, reflect and present on each applied learning task.

Key Concepts
- Teaching and Learning
- Add to clipboard
- Assessment
- Examples in context

Applied learning task 1: Interactive information systems

Applied learning task 2: Analytics

Applied learning task 3: Modelling and simulation

Applied learning task 4: Embedded systems

https://www.curriculumonline.ie
Computational Thinking
What is Computational Thinking?

- Problem Solving Methods: for a ‘Computer’
- Help
- Several Models: eg 4 Concepts / Pillars:
  - Decomposition
  - Pattern Recognition
  - Abstraction
  - Algorithm Formation
Computational Thinking

- Monty Hall Problem
- How to Subtract
- London Underground
- Money – Change (Activity): White board
- Sisters
Computational Thinking

234-159

1. Modern Primary School way.
2. Old Primary school way.
3. Shop Assistant.
4. Dart player.
The Tube
CT Activity 1

Euro coins are issued in the denominations shown

What is the minimum number of coins required to make up €27.93 cents?
CT – Activity 2

Who has more sisters, boys or girls?
Applied Learning Tasks
Students work in teams to carry out four applied learning tasks over the duration of the course. Each of which results in the creation of a real or virtual computational artefact.

These artefacts should relate to the students’ lives and interests.

Where possible, the artefacts should be beneficial to the community and society in general.

Examples of computational artefacts include programs, games, web pages, simulations, visualisations, digital animations, robotic systems, and apps.
The four applied learning tasks explore the following contexts:

- Interactive information systems
- Analytics
- Modelling and simulation
- Embedded systems.
INVESTIGATE  
define the problem

PLAN  
understand the problem

DESIGN  
create a representation, decide on tools

CREATE  
implement the plan

EVALUATE  
determine if the solution is appropriate

DOCUMENT  
report, present and reflect on the process

ITERATE
LCCS Interwoven
Group Activity
Activity: ALT

1. Each group will use the shared document choose a particular ALT –
   (ALT 1: Web design  ALT 2: Analytics  ALT3: Modelling, Simulation
   ALT4: Embedded Systems)
2. In groups start brainstorming as to possible project ideas for students. Aim for as many ideas as you can.
3. Write your ideas in the doc – can be text / images etc.
4. Present ideas to the wider group.
ALT Activity: Pick one Activity from Brainstorming and Expand

In groups teachers expand on one of their ideas from earlier.
• What teaching & learning strategies could you use?

• How would you assess?

• Can it be linked to other parts of the course?

• What theory could be taught at the same time?

• In terms of planning where in the course do you see this ALT fitting in?