Short Course
Coding

Specification for Junior Cycle

June 2025

Contents

Introduction to junior cycle

Junior cycle education places students at the centre of the educational experience, enabling them to actively participate in their communities and in society, and to be resourceful and confident learners in all aspects and stages of their lives. Junior cycle is inclusive of all students and contributes to equality of opportunity, participation and outcome for all.

Junior cycle allows students to make a strong connection with learning by focusing on the quality of learning that takes place and by offering experiences that are engaging and enjoyable for them, and relevant to their lives. These experiences are of a high quality, contribute to the physical, mental and social wellbeing of learners, and where possible, provide opportunities for them to develop their abilities and talents in the areas of creativity and enterprise. The student's junior cycle programme builds on their learning in primary school. It supports their further progress in learning. It helps students to develop the learning skills that can assist them in meeting the challenges of life beyond school.

Rationale

Computer science is present in every aspect of modern society. Correctly-functioning software systems allow airplanes to fly from one city to another; give out money at the ATM and diagnose the level of glucose in your blood. Fundamental understanding of how computer hardware and software operate and relate to everyday life is an increasingly important area of learning for students. Problem solving and computational thinking skills are developed in this course as students build and create software projects using their own ideas and imagination. The course looks to build on any coding skills that primary students might have acquired, including the exploration of a programming language, while offering insight into possible future studies in computer science and software engineering. In having a focus on the future, the course also requires students to critically review the use of software systems, taking into account ethics, data privacy and handling and other opportunities and challenges that may arise over time.[1]

1 Teachers and students may wish to refer to the junior cycle Digital Media Literacy short course at www.curriculumonline.ie/junior-cycle/short-courses/digital-media-literacy/

Aim

The short course aims to develop the student's ability to formulate problems logically; to design, write and test code through the development of programs, apps, games, animations or websites; and, through their chosen learning activities, to learn about ethical issues, computational thinking and computer science.

Overview: Links

Tables 1 and 2 on the following pages show how coding may be linked to central features of learning and teaching in junior cycle.

Coding and statements of learning

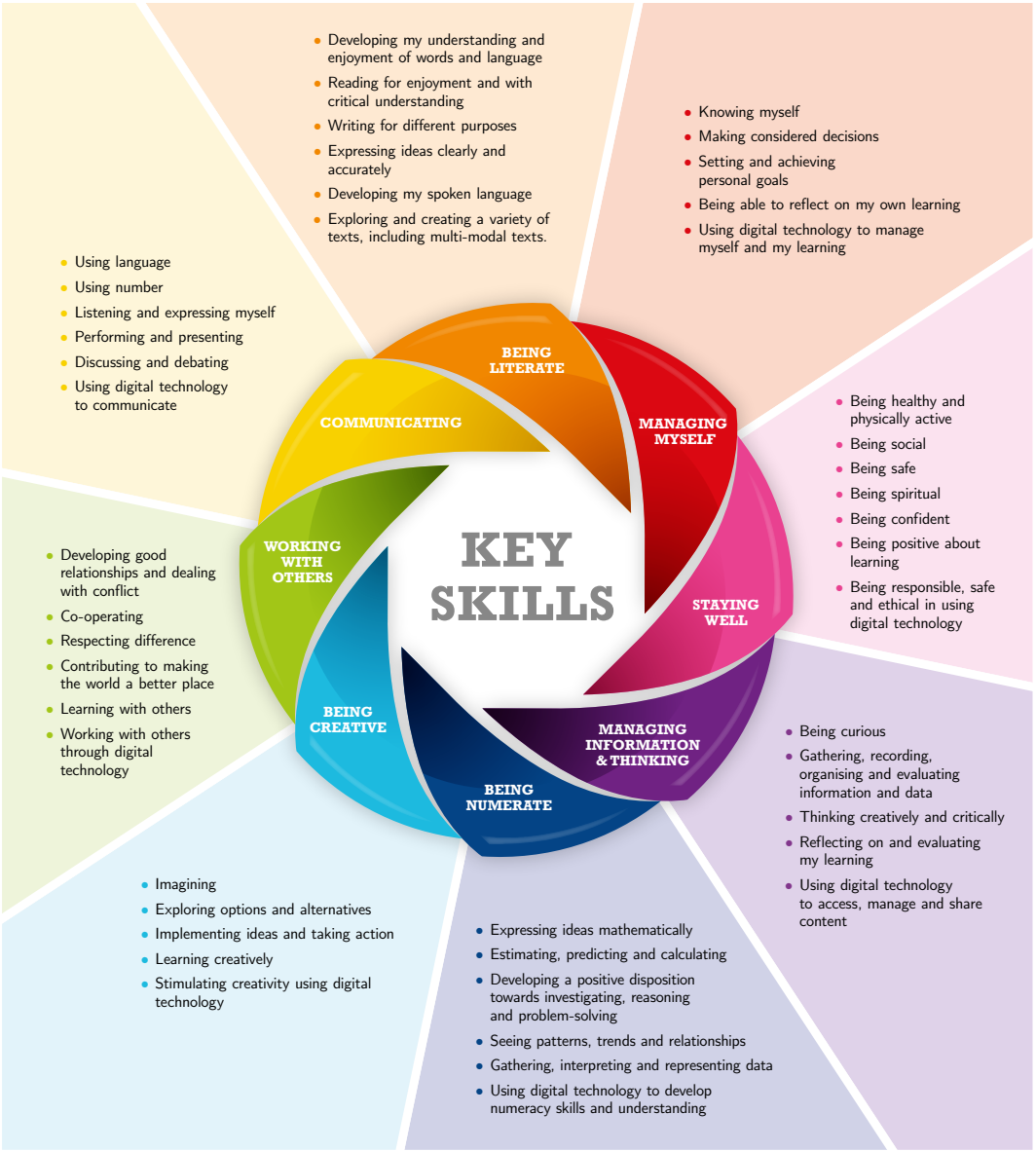
Table 1: Links between junior cycle Coding and the statements of learning

Statement	Examples of related learning in the course
SOL 17: The student devises and evaluates strategies for investigating and solving problems using mathematical knowledge, reasoning and skills.	Problem solving and computational thinking are central to this course. Students use their general and mathematical knowledge, skills and understanding when figuring out, evaluating and implementing solutions to particular problems.
SOL 16: The student describes, illustrates, interprets, predicts and explains patterns and relationships.	Students interpret and describe patterns and relationships as they solve problems and create projects using algorithms and programming languages.
SOL 20: The student uses appropriate technologies in meeting a design challenge.	Students, either individually or as part of a team, research and discuss the most appropriate technologies to use, to solve problems and deliver solutions.
SOL 23: The student brings an idea from conception to realisation.	Students engage in brainstorming and planning activities, move on to the design, development and test phases, culminating in the creation of a project solution to a particular problem.
SOL 24: Uses technology and digital media tools to learn, communicate, work and think collaboratively and creatively in a responsible and ethical manner.	Students review their own work and have an opportunity to explore current issues in computer science, and research and discuss their understanding of the ethical and legal aspects pertaining to these.

Coding and key skills

In addition to their specific content and knowledge, the subjects and short courses of junior cycle provide students with opportunities to develop a range of key skills. The junior cycle curriculum focuses on eight key skills.

Figure 1: Key skills of junior cycle



This course offers opportunities to support all key skills, but some are particularly significant. The examples below identify some of the elements that are related to learning activities in Coding.

Teachers can also build many of the other elements of particular key skills into their classroom planning.

Table 2: Links between junior cycle coding and key skills

Key skill	Key skill element	Student learning activity
Being Creative	Implementing ideas and taking action	Students brainstorm and generate ideas for design and implementation of solutions and projects.
Being Literate	Expressing ideas clearly and accurately	Students create a website to display the results of a topic they have researched. Students use computational thinking to address a problem in which they are interested.
Being Numerate	Developing a positive disposition towards investigating, reasoning and problem-solving	Students create short programs which demonstrate their use and understanding of mathematical and computational ideas.
Communicating	Discussing and debating	Students discuss ideas, evaluate the pros and cons of different approaches, review potential ethical and legal issues and propose possible solutions. They report on projects and provide feedback to others.
Managing Information And Thinking	Thinking creatively and critically	Students explore new and different ways of answering questions and solving problems. They use a variety of tools to access, manage and share information such as flow-charts, design documents, code documentation and bug lists.
Managing Myself	Knowing myself	Students take responsibility for personal learning by setting goals and seeking help when necessary, from classmates, the teacher or other appropriate sources, and by reflecting on the feedback they receive.
Staying Well	Being safe	Students learn to be responsible, safe and ethical in using digital technology. Students become aware of the wellness, health and safety issues associated with working with computers, and of practical ergonomic issues regarding the use of computers.
Working With Others	Cooperating	Students develop good working relationships with others and appreciate the value of respect and cooperation in reaching both collective and personal goals. They learn to appreciate diverse talents and how to engage in collaborative work.

Overview: Course

The specification for this junior cycle short course in coding focuses on developing students' problem-solving skills through three inter-connected strands: Introducing computer science, Let's get connected and Coding at the next level.

Strand 1: Introducing computer science

In this strand, students explore the range of uses computers have in today's world and learn to understand the hardware and basic software which operates them. This includes learning to write, test and evaluate code as well as taking account of and discussing potential ethical issues around the creation and use of digital technologies, including Artificial Intelligence.

Strand 2: Let's get connected

This strand deepens the student's understanding of the computer as a communications tool through the storage and manipulation of data. Students have the opportunity to identify, research, present and receive feedback on a topic or challenge in computer science that inspires them. Students also get to consider the ethical and legal and technical issues around the topic or challenge they have identified.

Strand 3: Coding at the next level

In this strand, students are introduced to more complex levels of coding where they can demonstrate their understanding through documentation and discussion of their work, and any potential ethical implications, and through engaging with feedback. Students may also decide to use a programming language of their choice during learning in this strand.

Figure 2: Course overview

Teamwork is encouraged throughout all three strands. Students collaborate, peer-explain, seek feedback, provide feedback and reflect on their work. Practical, hands-on and problem-solving learning activities should be in evidence across all strands of the course. Theoretical concepts can be reinforced through practical work and projects.

Free and open-source software should be used where practical, both to make software tools as widely available to students as possible and so that students have the opportunity to examine the source code of the tools they use.

In the context of their health and wellbeing, student safety in the use of computers is emphasised in the Statements of learning and should be a feature of the experience of students taking the course. There is a further NCCA short course available on Digital Media Literacy which addresses internet safety concerns and ethical, legal and responsible use in more detail. Students should also be made aware of the acceptable use policy of their own school

The Classroom-Based Assessment outlined below reflects the learning students undertake in this NCCA short course. Schools have the flexibility to adapt any NCCA short course to suit their particular needs and school context. If adapting the course, schools may also need to adapt the Classroom-Based Assessment, so that it reflects the learning their students undertook. Schools may also develop their own short course(s) and related Classroom-Based Assessment.

The learning outcomes in this short course are aligned with the level indicators for Level 3 of the National Framework of Qualifications (Appendix 1).

The course has been designed for approximately 100 hours of student engagement.

Teaching and learning

While the learning outcomes associated with each strand are set out separately in this specification, this should not be taken to imply that the strands are to be studied in isolation. This course is designed to be followed sequentially through strands 1–3, enabling students to build on the skills they have previously learned. Strand 1 is an introductory strand and should be undertaken first. Schools, however, may wish to approach the learning outcomes across the other two strands in the order they deem best for students.

The Classroom-Based Assessment outlined below reflects the learning students undertake in this NCCA short course. Schools have the flexibility to adapt any NCCA short course to suit their particular needs and school context. If adapting the course, schools may also need to adapt the classroom-based assessment, so that it reflects the learning their students undertook. Schools may also develop their own short course(s) and related classroom-based assessment. Guidelines for schools who wish to develop their own short course(s) are available at ncca.ie/en/junior-cycle/subjects-and-short-courses/develop-your-own-short-course/

Progression

Primary curriculum

One of the five curriculum areas in the Primary curriculum is STEM Education. Sitting alongside the Primary Mathematics Curriculum (Department of Education, 2023), the Science, Technology and Engineering curriculum specification, due to be published in 2025, focuses on computational thinking, which draws on the principles of computing to think about and solve problems. Students learn to develop their ability in using computational thinking so it can be applied not just to science, technology and engineering that they learn about in school but across other subjects as well. By applying previous learning and knowledge, students can use logical thinking and reasoning to break down problems into manageable pieces where they can focus on determining the key information relevant to the whole problem-solving process. Students learn this through un-plugged activities and, later, plugged (digital and online activities) where they may also incorporate programming languages.

Being a digital learner is a key competency in the Primary Curriculum Framework (Department of Education, 2023) and it aims to support students in becoming, "curious, creative, confident and critical users of digital technology". Students also learn to be responsible, respectful, safe and ethical users of technology. This knowledge and the skills they have developed can be transferred into this Coding short course and built upon by students as they progress through junior cycle.

Senior cycle

Students may be able to continue with their study of coding and computer science through relevant TY modules or, if on the school's curriculum, through Leaving Certificate Computer Science. Students can continue to develop their learning through these avenues and so further develop their skills and knowledge in computational thinking and computer science. This involves problem-solving, abstraction and logical reasoning and their implementation using automation, programming and computer systems.

As in primary Science, Technology and Engineering Education, and junior cycle Coding, Leaving Certificate Computer Science continues to foster student creativity while helping them to work both independently and collaboratively. They learn to apply the fundamental practices and concepts of Computer Science and to appreciate the diverse role that computing technology has within the society and environment in which they live.

Expectations for students

Learning outcomes are statements that describe what knowledge, understanding, skills and values students should be able to demonstrate having completed this junior cycle short course in Coding. The learning outcomes set out in the following tables apply to all students and represent outcomes for students at the end of their period of study (approximately 100 hours).

As set out here, they represent outcomes for students at the end of junior cycle. The specification stresses that the learning outcomes are for students across junior cycle and therefore the learning outcomes focused on at a point in time will not have been 'completed' but will continue to support the students' learning up to the end of junior cycle.

The outcomes are numbered within each strand. The numbering is intended to support teacher planning in the first instance and does not imply any hierarchy of importance across the outcomes themselves.

A glossary of action verbs in addition to a glossary of specific terms used throughout the specification is provided in the appendices included with this specification. These are intended to support teachers in planning for teaching, learning and assessment. Specific terms included within the Glossary have been asterisked.

Learning outcomes

Learning outcomes are statements that describe what knowledge, understanding, skills and values students should be able to demonstrate having completed this junior cycle short course in Coding. The learning outcomes set out in the following tables apply to all students and represent outcomes for students at the end of their period of study (approximately 100 hours).

The outcomes are numbered within each strand. The numbering is intended to support teacher planning in the first instance and does not imply any hierarchy of importance across the outcomes themselves.

Strand 1: Introducing computer science

Students learn about	Students should be able to
My digital world: The importance of computers in modern society and their lives, including the ethical uses of computers, software, and Artificial Intelligence (AI)	1.1 present and share examples of what computers are used for and discuss their importance in modern society and in their lives 1.2 describe the main components of a computer system (Central Processing Unit (CPU), memory, main storage, Input/Output (I/O) devices, buses) 1.3 explain how computers are devices for executing programs via the use of programming languages 1.4 discuss the ethical and legal issues that could be encountered when using computers, applications, or digital devices, or when engaging online
Being a coder–step by step: How to start programming and use computational thinking* to develop basic algorithms*	1.5 develop appropriate algorithms using pseudo-code* and/or flow charts 1.6 construct code to implement algorithms 1.7 discuss and implement core features of structured programming languages, including variables*, operators*, loops*, decisions*, assignment* and modules* 1.8 test the code to determine its functionality and to identify and manage errors 1.9 evaluate the results in groups of two or three and discuss how the code worked and potential modifications to help improve it

Strand 2: Let's get connected

Students learn about	Students should be able to
Making connections: The importance of computers as communication devices and how they provide access to data	2.1 discuss the basic concepts underpinning the internet 2.2 investigate and describe how data is transported on the internet and how computers communicate and cooperate through protocols 2.3 explain the difference between search engines and web browsers 2.4 investigate how search engines deliver results 2.5 create a small website using HTML and CSS to showcase their learning
Bits and bytes: How computers store data and how data can be stored in different formats	2.6 explain how computers represent data using 0s and 1s 2.7 investigate how drawings and photos are represented in computing devices
Artificial Intelligence: Learning about Artificial Intelligence (AI) and its ethical and safe use in coding	2.8 identify different AI models and how they can be applied in coding 2.9 discuss the use of AI models when applied in coding, their potential advantages and limitations and any ethical and safety issues which may arise
Real world problems: How computer science* can inspire critical and creative thinking	2.10 identify a topic or a challenge in computer science that inspires them 2.11 investigate the chosen topic/challenge including the related ethical, legal and technical issues 2.12 present the proposed topic or challenge to be addressed for discussion and reflect on feedback received 2.13 justify continuing with the proposed topic or challenge to be addressed outlining why it is worthwhile and any modifications made

Strand 3: Coding at the next level

Students learn about	Students should be able to
Being a coder: More advanced concepts in programming and computational thinking and how these support creativity	3.1 creatively design and write code for short programming tasks to demonstrate the use of operators for assignment*, arithmetic, comparison*, and Boolean combinations* 3.2 realise short programming tasks using basic linear data structures (such as array* or list*) 3.3 demonstrate how functions* and/or procedures* (definition and call) capture abstractions 3.4 describe program flow control, e.g. parallel* or sequential* flow of control (language dependent) 3.5 investigate any potential ethical or legal considerations which might be required by their chosen programming task(s)
Documenting the code: The importance of documenting actions in coding and code analysis, including reflecting on how their learning has developed	3.6 document programs to explain how they work 3.7 present the documented code to each other in small groups 3.8 test their work with external users to better understand the user experience of the code/ program they have developed 3.9 analyse code to determine its function and identify errors or potential errors 3.10 meaningfully reflect on how their learning has developed over time as they review and document the development of their code

Assessment and reporting

Essentially, the purpose of assessment and reporting at this stage of education is to support learning. This short course supports a wide variety of approaches to assessment. Some learning outcomes lend themselves to once-off assessment, others to assessment on an ongoing basis as students engage in different learning activities such as discussing, explaining, researching, presenting, planning and taking action. In these contexts, students with their teachers and peers reflect upon and make judgements about their own and others' learning by looking at the quality of particular pieces of work. They plan the next steps in their learning, based on feedback they give and receive. Ongoing assessment can support the student in their learning journey and in preparing for the Classroom-Based Assessment related to this short course.

It is envisaged that students will provide evidence of their learning in a variety of ways, including digital media, audio recordings and written pieces.

Assessment is most effective when it moves beyond marks and grades and reporting focuses not only on how the student has done in the past but on the next steps for further learning. Student progress and achievement in short courses, both in ongoing assessments and in the specific Classroom-Based Assessment relating to this short course will be communicated to parents in interim reporting and in the Junior Cycle Profile of Achievement (JCPA). The Focus on Learning Toolkit is available online to support ongoing reporting and formative assessment in the classroom.

Classroom-Based Assessment

Classroom-Based Assessments are the occasions when the teacher assesses the students in the specific assessment(s) that are set out in the subject or short course specification. Junior cycle short courses will have one Classroom-Based Assessment. Where feasible, teachers of short courses will participate in learning and assessment review meetings.

Classroom-Based Assessment: Putting the pieces together

This Classroom-Based Assessment is the culmination of the work undertaken in the three strands of the coding short course. The Classroom-Based Assessment should begin after work in the three strands has been completed.

Students will have a choice of two approaches:

1. **Develop a final software project of their choice in teams of two or three.**
   Students will research and establish requirements; design, implement and test the software. They will document their work and their code and present the project to their peers for review. They will reflect on feedback and also provide feedback on other students' projects

2. **Create a portfolio collection of examples of their work across the three strands.**
   Students will evaluate the work they have completed to date and choose examples that demonstrate how their learning has developed over time. They will reflect on each example; on the learning they took from it and how it helped them improve their ability to problem solve and use computational thinking.

Features of quality

The features of quality support student and teacher judgement of the Classroom-Based Assessments and are the criteria that will be used by teachers to assess students' final software projects or portfolios.

More detailed material on assessment and reporting in this junior cycle coding short course, including features of quality and details of the practical arrangements related to assessment of this Classroom-Based Assessment, will be available in separate assessment guidelines for Coding. The guidelines will include, for example, the suggested length and formats for student pieces of work, and support in using 'on balance' judgement in relation to the features of quality.

Inclusive assessment

Inclusive assessment practices, whether as part of ongoing assessment or the Classroom-Based Assessment, are a key feature of teaching and learning in schools. Accommodations, e.g. the support provided by a special needs assistant or the support of assistive technologies, should be in line with the arrangements the school has put in place to support the student's learning throughout the year.

Where a school judges that a student has a specific physical or learning difficulty, reasonable accommodations may be put in place to remove, as far as possible, the impact of the disability on the student's performance in the Classroom-Based Assessment.

Appendix 1:

Level indicators for Level 3 of the National Framework of Qualifications

This short course has been developed in alignment with the level indicators for Level 3 of the National Framework of Qualifications. Usually, for Level 3 certification and awards, the knowledge, skill and competence acquired are relevant to personal development, participation in society and community, employment, and access to additional education and training.

NFQ Level 3

Knowledge *Breadth*	Knowledge moderately broad in range
Knowledge *Kind*	Mainly concrete in reference and with some comprehension of relationship between knowledge elements
Know-how and skill *Range*	Demonstrate a limited range of practical and cognitive skills and tools
Know-how and skill *Selectivity*	Select from a limited range of varied procedures and apply known solutions to a limited range of predictable problems
Competence *Context*	Act within a limited range of contexts
Competence *Role*	Act under direction with limited autonomy; function within familiar, homogeneous groups
Competence *Learning to*	Learn to learn within a managed environment
Competence *Insight*	Assume limited responsibility for consistency of self-understanding and behaviour

Appendix 2:

Glossary of terms and concepts

This glossary is intended to clarify concepts and terms used in this specification for the teacher and student.

Algorithm	An algorithm is a sequence of steps designed to accomplish a specific task. Algorithms are translated into programs, or code, to provide instructions for computing devices.
Arithmetic combination	In programming, an arithmetic combination is an operator where it combines arithmetic terms (x^2, x $3xy^2$) along with arithmetic operators (+, -, X and ÷).
Array	In programming, an array is a list of related values, where the values are usually of the same type.
Assignment	In programming, an assignment, or assignment statement, sets and/or resets the value stored in a variable.
Boolean combination	In programming, a Boolean value is either true or false. Boolean combinations are logical expressions that combine values using logical operators such as AND, OR, and NOT.
Computational thinking	Computational thinking allows us to take a complex problem, understand what it is and develop potential solutions that can be presented in a way that a computer, a human, or both, can understand. Computational thinking involves four main steps: 1. **Decomposition** - breaking down a complex problem or system into smaller, more manageable parts 2. **Pattern recognition** – looking for similarities among and within problems 3. **Abstraction** – focusing on the important information only and ignoring irrelevant detail 4. **Algorithms** - developing a step-by-step solution to the problem; the rules to follow to solve the problem
Computer science	Computer science is the study of computing and algorithmic processes
Decisions	In programming, decisions are also known as conditional statements, and they direct the flow of the program.
Function	In programming, a function is a sequence of commands that can be re-used together in a program.
List	In programming, a list holds a collection of data or items.
Loop	In programming, a loop is a sequence of instructions that are repeated until a specific condition has been met.
Module	In programming, a module is an independent block of components, which cannot work on their own, but are inserted into a larger program much like a building block. They usually execute one particular aspect of functionality for a larger program.

Operator	In programming, an operator is a symbol or character that represents a specific mathematical or logical action/process.
Parallel flow	In programming, parallel flow is when a computer executes several operations at the same time.
Procedure	In programming, a procedure is a small section of a program that performs a specific task. It can be used and re-used repeatedly.
Pseudo-code	Pseudo-code is not a programming language that could be compiled as an executable program but rather is a detailed and readable description of what the proposed code should do.
Sequential flow	In programming, sequential flow is where a program is executed in the linear order it has been written.
Variable	In programming, a variable is a value that can be changed and is not fixed.

Appendix 3:

Glossary of Action Verbs

This glossary for Coding is designed to help to clarify the expectations for students within the learning outcomes. Each action verb is described in terms of what the learner should be able to do once they have achieved the learning outcome.

Action verbs	Students should be able to
Analyse	study or examine something in detail, break down in order to bring out the essential elements or structure; identify parts and relationships, and to interpret information to reach conclusions
Argue	challenge or debate an issue or idea with the purpose of persuading or committing someone else to a particular stance or action
Construct	develop information in a diagrammatic or logical form; not by factual recall but by analogy or by using and putting together information or to build or form from different elements
Create	use their knowledge, understanding, skills and values make something new
Debate	engage in structured argument where both sides of the issue are discussed
Demonstrate	prove or make clear by reasoning or evidence, illustrating with examples or practical application
Describe	develop and provide a detailed picture
Design	a plan, scheme or project idea or a combination of these, which is executed in accordance with appropriate functional and/or aesthetic criteria
Develop	advance a piece of work or an idea from an initial state to a more advanced state
Discuss	offer considered informed argument or opinion
Document	record a process in detail using examples where appropriate
Explain	give a detailed account including reasons or causes
Evaluate (data)	collect and examine data to make judgments and appraisals; describe how evidence supports or does not support a conclusion in an inquiry or investigation; identify the limitations of data in conclusions; make judgments about the ideas, solutions or methods
Evaluate (ethical judgement)	collect and examine evidence to make judgments and appraisals; describe how evidence supports or does not support a judgement; identify the limitations of evidence in conclusions; make judgments about the ideas, solutions or methods
Identify	recognise patterns, facts, or details; provide an answer from a number of possibilities; recognise and state briefly a distinguishing fact or feature
Investigate	observe, study or examine in detail in order to establish facts, and reach new insights and/or conclusions
Present	A specific form of communication which makes images/objects perceivable for others
Realise	Implement, execute or put into practice an idea or a product or a draft
Test	Critically examine or determine how genuine anything is or compare it to a theory or plan as to how it should behave or appear

An Roinn Oideachais agus Óige
Department of Education and Youth

NCCA
An Chomhairle Náisiúnta Curaclaim agus Measúnachta
National Council for Curriculum and Assessment

Specification for **Junior Cycle**