T-LEVELS

HM Government

T LEVEL

Technical Qualification in Digital Production, Design and Development (Level 3)

Specification

First teaching September 2026

Version 1.5 – April 2026

Pearson

About Pearson

We are the world's leading learning company operating in countries all around the world. We provide content, assessment and digital services to learners, educational institutions, employers, governments and other partners globally. We are committed to helping equip learners with the skills they need to enhance their employability prospects and to succeed in the changing world of work. We believe that wherever learning flourishes so do people.

This specification is Version 1.5. We will inform providers of any changes to this version through bulletins and provider updates. The latest version can be found on our website.

References to third-party material made in this specification are made in good faith. Pearson does not endorse, approve or accept responsibility for the content of materials, which may be subject to change, or any opinions expressed therein. (Material may include textbooks, journals, magazines and other publications and websites.)

All information in this specification is correct at time of publication.

ISBN 978 1 446 96851 2

Copyright in this document belongs to, and is used under licence from the Department for Education, © 2026.

'T-LEVELS' and 'T Level' are registered trademarks of the Department for Education. Pearson is authorised by the Department for Education to develop and deliver this T Level Technical Qualification.

Pearson and logo are registered trademarks of Pearson

Summary of changes: Version 1.5		
Change	**Section**	**Page No.**
The summary tables have been updated	Qualification Summary and Structure	4, 5
The Administration section within the Scheme of Assessment has been updated, this signposts the links to assessment guidance for Providers	Scheme of Assessment (Core & Occupational Specialisms)	36, 63,
The administration information from sections '1, 5, 6, 7, 8' has been extracted from this version of the Specification. This information has been compiled to form an 'Administration Support Guide for the specific Technical Qualification'. The guide is located on the Training and Admin Support webpage	1,5,6,7,8	
The following sections have been updated/relabelled: • Section 5 has been updated to contain the Command Word Taxonomy • Section 6 contains the General Competency Frameworks for T Levels	 5 6	 66 67

Contents

1. Introducing the Qualification

T Level Programme

T Levels are two-year, Level 3 study programmes that follow the study of GCSEs and Technical Awards and offer an alternative to A Levels and Apprenticeships.

T Levels combine classroom theory, practical learning and a minimum 315 hours of industry placement with an employer. The work placement ensures students have real experience of the workplace.

T Level programmes are developed in collaboration with employers so that the content meets the needs of industry and prepares students for work. T Levels provide the knowledge and experience needed to progress to highly skilled employment, an Apprenticeship or higher-level study, including university.

Understanding the Specification and Administrative Guide

This specification should be read in conjunction with the Administrative Guide for Delivery and Assessment. The specification contains all the information you need to teach the technical qualification including content and assessment details. The Admin Guide contains the information and references you need to register as a provider, register students and administer their results. It also contains grading information and information on resources.

What is the Technical Qualification (TQ)?

The *T Level Technical Qualification in Digital Production, Design and Development* is the main classroom-based element of the T Level. Students will learn using a curriculum that has been shaped by industry experts.

During the two-year programme, students will acquire the core knowledge that underpins each industry. They will develop occupationally specific skills that will allow them to enter skilled employment within a specific occupation.

Technical Qualification and Outline Content

The Outline Content for the *T Level Technical Qualification in Digital Production, Design and Development* has been produced by T Level panels of employers, professional bodies and Providers. It is based on the Apprenticeship Standards.

Pearson has used the Outline Content to form the basis of the Technical Qualification specification. This includes:

- elaboration of the Outline Content to produce a specification that gives Providers an accurate interpretation of what needs to be taught and assessed
- enabling students to achieve threshold competence in relation to the Occupational Specialist component(s)
- the integration of English, maths and digital competencies.

Employer and Provider Panels

Pearson engaged with employer and Provider panels throughout the development of the Technical Qualification. This ensured:

- the content gives students quality preparation to help them progress
- assessments are realistic and assess the knowledge and skills that are important to employers
- the technical qualification meets the needs of Providers.

Pearson is grateful to all university and further education lecturers, teachers, employers, professional body representatives and other individuals who have generously shared their time and expertise to help us develop these new qualifications.

Employers, professional bodies and Providers who contributed to the development of the Technical Qualification include:

- ARM
- BBC
- BT
- Cisco
- CompTIA
- Nationwide
- Siemens.

Qualification Purpose

This Technical Qualification is for T Level students who are undertaking the *T Level in Digital Production, Design and Development*. It is intended for students who want to progress to a career in the Digital sector, with a focus on software design and development.

The purpose of the *T Level Technical Qualification in Digital Production, Design and Development* is to ensure students have the knowledge and skills needed to progress into highly skilled employment, an Apprenticeship or higher level study, including university, within the specialist area of software design and development.

At the end of the Technical Qualification, students are expected to demonstrate threshold competence, which means that they have gained the core knowledge and skills related to software design and development and are well placed to develop full occupational competence with additional development and support once in employment in the digital sector.

Student Profile and Progression

Students undertaking this Technical Qualification will be 16–19 years old and in full-time education.

The typical student has:

- a clear idea about the industry sector in which they wish to pursue a career
- an idea of the type of job role they would like to explore as a career.

This Technical Qualification aligns to the Software Development Technician Level 3 Apprenticeship and therefore supports progression to entry-level job opportunities in software design and development.

Job roles could include:

- Software Development Technician
- Junior Developer
- Junior Web Developer
- Junior Application Developer
- Junior Mobile App Developer
- Junior Games Developer
- Junior Software Developer
- Junior Application Support Analyst
- Junior Programmer
- Assistant Programmer
- Automated Test Developer.

The jobs available to the students will be based on their individual abilities in the digital sector and will be supported by their achievement of this qualification.

Alternatively, students could progress sideways to the Level 3 Software Technician Apprenticeship to develop and gain certification of their occupational competence, or they could progress to higher level Apprenticeships such as the Level 4 Software Developer, depending on their skills or experience.

Where students may not have access to an Apprenticeship or would prefer a more academic route, they could progress to relevant Higher National Certificate (HNC) or Higher National Diploma (HND) programmes or digital degree programmes such as Computer Games Programming BCs, Software Engineering BSc, Virtual Reality Design BA, Computing BSc, Digital Media Design and Development BSc or Computer Science BSc.

Students should always check the entry requirements for each degree programme with the relevant higher education provider

2. Qualification Summary and Structure

Summary

Qualification title	T Level Technical Qualification in Digital Production, Design and Development (Level 3)
Qualification number (QN)	603/5832/4
Total Guided Learning Hours (GLH)	1200 GLH (600 hours core)
Total Qualification Time (TQT)	1640 TQT (810 hours core)
Occupational Specialism	Digital Production, Design and Development (600 GLH, 830 TQT)
Components and weighting	Core Component = 50% of total qualification – made up of: ● Core Paper 1 =17% of total qualification (33.3% of core) ● Core Paper 2 =17% of total qualification (33.3% of core) ● Core ESP = 17% of total qualification (33.3% of core) Occupational specialism = 50% of total qualification
Grading information	Core and Employer Set Project (ESP) components are graded A*–E or unclassified. Occupational Specialism (OS) component is graded Pass, Merit, Distinction or unclassified. The overall grading is on a scale of Pass, Merit, Distinction, Distinction* or Unclassified.
Entry requirements	There are no formal prior learning requirements. It is the Provider's responsibility to ensure students recruited have a reasonable expectation of success. Students are most likely to succeed if they have qualifications at Level 2 (for example, five GCSEs at grade 4 and above including English and maths or a vocational Tech Award pass at Level 2). Students may demonstrate the ability to succeed in various ways. For example, they may have relevant work experience or may have shown specific aptitude through diagnostic tests or other non-educational experience.
Assessment	The Core and ESP components are externally set and marked by Pearson. The OS component is externally set and marked by Pearson.

Assessment Structure

The *T Level Technical Qualification in Digital Production, Design and Development* has two mandatory components:

1. Core Component

This component covers the underpinning knowledge, concepts and skills that support threshold competence in the digital industry. It is assessed by two externally set Core examinations and an Employer Set Project.

The content and details of each of these assessments is provided in *Section 3.*

Assessment component	Assessment method	Duration	Marks	Weighting	Availability
Core Paper 1: Digital Analysis, Legislation and Emerging Issues	Written examination	2.5 hours	100	33.3%	Summer/Autumn
Core Paper 2: The Business Environment	Written examination	2.5 hours	100	33.3%	Summer/Autumn
Employer Set Project	Externally set project	14.5 hours	100	33.3%	Summer/Autumn

2. Occupational Specialist Component

There is a single Occupational Specialist component in this Technical Qualification. Students undertaking the *T Level Technical Qualification in Digital Production, Design and Development* will complete this specialism.

This component covers the occupational specialist knowledge and skills required to demonstrate threshold competence for the specialism and it will be assessed by a skills-related project that synoptically assesses the Performance Outcome skills and associated underpinning knowledge.

The content and details of the assessment for the Occupational Specialist component is provided in Section 4.

Assessment component	Assessment method	Duration	Marks	Weighting	Availability
Digital Production, Design and Development	Externally set project	67 hours	145	100%	Summer (Spring/summer term)

What Does the Qualification Cover?

Students will learn about the following topics:

- problem solving
- programming
- emerging issues and impact of digital
- legislation and regulatory requirements
- business context
- data
- digital environments
- security.

3. Core Component

Content summary

The core content covers the knowledge, understanding and application of contexts, concepts, theories and principles relating to the following areas:

1. Problem solving

2. Introduction to programming

3. Emerging issues and impact of digital

4. Legislation and regulatory requirements

5. Business context

6. Data

7. Digital environments

8. Security

Paper 1: Digital Analysis, Legislation and Emerging Issues

Content area 1: Problem solving

Students must be able to apply problem-solving skills to analyse problems and to identify solutions that can be developed into computer programs. Students will be expected to solve realistic problems that may form a complete solution or a sub-part of a larger program. All pseudocode questions will be presented using the commands and structure listed in Appendix 1. Students will be expected to use the flowchart symbols listed in Appendix 2.

What students need to learn		
1.1 Computational thinking		
1.1.1	Be able to use top-down, bottom-up and modularisation approaches to solve problems.	**E1** **M7**
1.1.2	Be able to decompose problems by: • identifying and describing the main features of a problem or process • breaking a problem down into smaller, more manageable parts.	**E1 E2** **M7**
1.1.3	Be able to use pattern recognition to: • identify and describe trends and similarities within and between problems and processes • identify and describe common features between a given problem and existing solutions • make predictions and assumptions based on identified patterns.	**M2 M4 M8** **D4**
1.1.4	Be able to use abstraction to: • identify information that is needed to solve an identified problem • filter out unnecessary details at different stages of a problem • create a layer of abstraction appropriate to the stage in the problem-solving process, including: o what inputs are needed o what the expected outputs are o things that will vary o things that will remain constant o key actions the program must perform o repeated processes the program will perform.	**E1 E5** **M10**

1.2 Algorithms		
1.2.1	Understand what algorithms are and how they are expressed (flowcharts, written descriptions, pseudocode, program code).	
1.2.2	Be able to express an algorithm using flowcharts and pseudocode, and understand how to use these when planning a digital solution.	**M7 M10 D3 D6**
1.2.3	Be able to write algorithms that make use of programming constructs (sequence, selection, iteration).	**M7 M10 D3 D6**
1.2.4	Understand the purpose of a given algorithm (flowcharts, written descriptions, pseudocode, program code) and how the algorithm works.	**M4 M7**
1.2.5	Be able to determine the correct output of an algorithm.	**M2 M4 M5 D4**
1.2.6	Be able to identify and correct errors in an algorithm (flowcharts, written descriptions, pseudocode, program code).	**M2 M4 M5 M7 D4**

Content area 2: Introduction to programming

Students should be able to apply an understanding of computer programming to solve problems. Students should be able to design, read, write and debug program code. Students will be expected to solve realistic problems that may form a complete solution or a sub-part of a larger program.

When designing a program, students will be expected to use the flowchart symbols listed in Appendix 2. Students will be expected to write, interpret and debug code and algorithms written in the programming language Python 3.

Students will be expected to write, interpret and debug code and algorithms written in the programming language Python 3.

Students will be expected to create functions and procedures to structure and carry out programming requirements.

Students will be expected to use code development tools, including Integrated Development Environments (IDE).

When writing, interpreting and debugging code, students will be expected to understand and use the libraries, functions and methods listed in Appendix 3.

Where program concepts listed in this section are not available in the Python 3 programming language, students would be expected to demonstrate understanding through written responses and pseudocode-based algorithms.

What students need to learn		
2.1 Program data		
2.1.1	Understand the use of, and need for, data types: • string • character • integer • real/float • Boolean.	**M5** **D6**
2.1.2	Be able to declare and use constants and variables that use appropriate data types.	**M4 M5** **D4 D6**
2.1.3	Understand the use of, and need for, data structures: • list • array • dictionary.	**M4 M5 M6** **D4 D6**

2.1.4	Understand how to manage variables within a program, including:
	• use of local and global variables
	• when local and global variables should be used, and why
	• naming conventions:
	  ○ meaningful names
	  ○ case (camelCase, UPPER CASE, snake_case, PascalCase).
		M4 M5 M6
		D4 D6

2.2 Operators		
2.2.1	Understand the purpose of, and how to use, mathematical operators in program code and algorithms (add, subtract, divide, multiply, integer division, modulus).	**M3 M4**
2.2.2	Understand the purpose of, and how to use, relational operators (==, <, >, <>, <=, >=).	**M3 M4**
2.2.3	Understand the purpose of, and how to use, Boolean operators (NOT, AND, OR).	**M4** **D6**
2.3 File handling		
2.3.1	Understand how to use text files for input and output of data.	**M5** **D1 D4 D6**
2.4 Program structure		
2.4.1	Understand how sequence, selection (branching) and iteration are used within programs and algorithms.	
2.4.2	Be able to write, interpret and debug code that makes use of sequence: • Determine the most efficient and logical order for actions within a process. • Understand the correct order of operations in calculations and processes, to ensure outputs are accurate and errors are avoided.	**M2** **D6**
2.4.3	Be able to write, interpret and debug code that makes use of selection (branching): • IF, THEN, ELSE, ELSEIF (ELIF) • CASE.	**M2 M4 M10** **D6**
2.4.4	Be able to write, interpret and debug code that makes use of iteration: • Understand how 'For' loops are used to iterate code a set number of times. • Understand how 'While' loops are used to iterate code while a set criterion is met. • Understand how loops are used to iterate code until a set criterion is met.	**M4 M10** **D6**
2.4.5	Be able to declare and call functions and procedures.	**M4 M10** **D6**
2.4.6	Understand how standard searching and sorting algorithms work, and the benefits and drawbacks of each: • linear and binary search • bubble sort, insertion sort, merge sort.	**E1** **M4 M5** **D4 D6**

2.5 Built-in functions		
2.5.1	Understand the benefits and drawbacks of using pre-written code.	**E4** **D6**
2.5.2	Be able to select and justify the use of pre-written code provided by the Python programming language (e.g. built-in functions, standard libraries).	**E4 E5** **D6**
2.5.3	Be able to write code that makes use of user-written and pre-written code (e.g. built-in functions, standard libraries).	**E4 E5** **D6**
2.6 Validation and error handling		
2.6.1	Understand the need for different types of input validation to handle common and unexpected errors and be able to write, interpret and debug code that makes use of these validation techniques: • presence check • length check • type check • format check • range check • check digit.	**M2 M4 M5 M6** **D6**
2.6.2	Understand the need to develop reliable and robust code.	**M6** **D6**
2.7 Maintainable code		
2.7.1	Understand accepted style conventions (such as Python's PEP 8) and how these are implemented to create readable and maintainable code.	**M4** **D3 D6**
2.8 Testing		
2.8.1	Understand the fundamental importance of testing for all components: • software • hardware • data • interfaces • resulting service (final product).	**D1 D6**

2.8.2	Understand the use of testing and quality assurance methodologies to seek out problems and issues: • concept testing • unit testing • boundary testing • integration testing • performance testing • system testing • acceptance and usability testing • regression testing • load/stress testing.	**E5** **M6** **D1 D6**
2.8.3	Understand how automated and functional testing tools can be applied to test digital systems and code.	**E5** **D1 D6**
2.8.4	Understand how to apply root cause analysis to solve problems: • what it is (the five whys) • when to use it • how to use it • what next.	**E5** **D3**
2.8.5	Understand how to construct an effective test plan, including: • identifying tests to be carried out • describing the purpose of the identified test • identifying test data to be used (valid, valid extreme, invalid, invalid extreme, erroneous) • describing the expected results.	**E1 E5** **M2** **M10** **D2 D6**

Content area 3: Emerging issues and impact of digital

Students should be able to apply an understanding of ethical and moral issues in the digital sector in a range of business contexts. They should explore how developments in technology impact on organisations, individuals and society as a whole.

Students should be aware of the ever-developing nature of digital technologies, and keep up to date with knowledge of important and innovative developments in the sector.

What students need to learn		
3.1 Moral and ethical issues		
3.1.1	Understand the ethical and moral issues that an increasing reliance on technology raises, and how organisations and individuals can respond to these challenges: • acceptable use • autonomous operation • changes in societal norms and the behaviour of individuals • changes in the culture within an organisation • environmental issues • globalisation • inclusion and diversity • monitoring of employees • open source and Creative Commons • the collection and use of data • unequal access to technology and/or digital services.	**E2 E4 E5** **D5**
3.1.2	Understand how organisations and individuals respond to ethical and moral issues when designing and developing digital systems, including: • use of guidelines from professional organisations • strategic planning and decisions • the content of internal policy documents • company culture and how this is established, communicated and sustained • whistleblowing.	**E2 E4 E5** **D5**
3.1.3	Understand how individuals use a range of observational techniques to inform situational awareness: • observing normal behaviour • awareness of co-workers • recognising changing or abnormal behaviour.	**E2 E4 E5** **D5**
3.2 Emerging trends and technologies		
3.2.1	Understand how developments in digital technologies impact on organisations, individuals and society, including: • Internet of Things (IoT) • Artificial Intelligence (AI), machine learning and deep learning • Augmented Reality (AR) and Virtual Reality (VR).	**E2 E4 E5** **D1**

Content area 4: Legislation and regulatory requirements

Students should be able to apply an understanding of legal issues in the digital sector in a range of business contexts. Students should explore how compliance with legislation impacts on the way in which organisations and their stakeholders use and interact with digital technologies.

Students should be aware of the ever developing nature of digital technologies and keep up to date with changes in legislation in response to technological developments.

What students need to learn		
4.1 Legislation		
Understand the role of current legislation and its impact on the design, development and use of digital in relation to:		
4.1.1	Health and safety when working with computers: • display screen regulations • general working environment • possible risks and prevention.	**E1 E2 E4 E5 D1**

4.1.2	Data security and protection:
	• key guidance and legislation including:
	  ○ the principles of the Data Protection Act
	  ○ UK General Data Protection Regulations (GDPR).
		E1 E2 E4 E5 M6 D1 D5

4.1.3	Computer Misuse Act: • the principles of the act • consequences (company and employee) • employee awareness.	**E1, E2, E4, E5 D1, D5**
4.1.4	Equality Act: • types of discrimination (protected characteristics) • where individuals are protected • when to take action against discrimination. • how individuals can be discriminated against (direct, indirect, harassment and victimisation).	**E1 E2 E4 E5 D1 D5**
4.1.5	Intellectual Property Act: • unregistered designs • registered designs • patents.	**E1 E2 E4 E5 D1 D5**
4.1.6	Understand the use of digital technologies for monitoring the workplace: • monitoring electronic communications • use of secret monitoring • employers' monitoring policies • monitoring systems.	**E1 E2 E4 E5 D1 D5**

4.1.7	Understand the role of legislation relating to international law and its importance when designing, developing and using digital systems.	**E1 E2 E4 E5 D1 D5**
4.2 Guidelines and codes of conduct		
4.2.1	Understand the purpose and role of codes of conduct produced by professional bodies for the use of digital: • British Computer Society (BCS) Code of Conduct • The Institution of Analysts and Programmers Code of Conduct.	**E1 E2 E4 E5 D1 D5**
4.2.2	Understand the guidelines provided in professional codes of practice in terms of: • professional responsibilities (quality of work, meeting deadlines, communication, confidentiality, trust) • contribution to society • safety • security and privacy • innovation.	**E5 D5**
4.2.3	Understand the impact that implementing guidelines from professional codes of practice has on organisations and their stakeholders.	**E4 E5 D5**
4.2.4	Understand how guidelines and agreed standards ensure the accessibility and quality of IT systems, including: • ISO (international Standards Organisation) standards • Web Content Accessibility Guidelines (WCAG) 1.0 and 2.0 • World Wide Web Consortium (W3C®) • Internet Engineering Task Force (IETF).	**E4 E5 D1 D5**
4.2.5	Understand the role and implications of acceptable use policies within an organisation.	**E4 E5 D5**

Paper 2: The Business Environment

Content area 5: Business context

Students must apply an understanding of the business environment including the importance of serving customer and end user, business needs, stakeholders such as customers, competitors, suppliers and government and the social, political, legal and technological factors drive the need for and use of digital skills technologies.

What students need to learn		
5.1 The business environment		
5.1.1	Understand the purpose of different types of organisations in a range of sectors: • to provide a service • to provide a product.	**E2 E4**
5.1.2	Understand the key areas of organisations and how IT is used to support them: • Human Resources • Research, Design and Development • Logistics • Marketing • Finance • Management.	**E2 E4 E5**

5.1.3	Understand how digital supports the business needs of organisations.
	• the use of digital to enable automated stock/inventory control:
	  ○ how software is used
	  ○ how hardware is used
	  ○ the processes carried out
	  ○ how different parts of the system communicate with each other
	• the use of traditional and cloud-based technologies and services to communicate and collaborate with internal and external stakeholders and facilitate collaboration.
		E4 E5
		D1 D3

5.1.4	Understand the factors that determine the feasibility of a digital project: • benefits and drawbacks • risks, constraints and dependencies.	**E5** **M2 M8 M9** **D4**

5.1.5	Understand how digital is used to meet user needs and ensure quality of product/service: • appropriate and effective functionality • reduction of pain points • accessibility considerations • compatibility • availability • good user experience • cultural awareness and diversity.	**E5** **D1**
5.1.6	Understand how the characteristics of end users affect the use and characteristics of digital technologies to access a service or product: • age • skills • education level • internal/external audience • level of technical knowledge • additional needs (e.g. users with sight or hearing loss).	**E5** **D1 D2**
5.2 Digital value to business		
5.2.1	Understand the importance of digital within organisations, and the ways in which digital is used to add value to a company: • engagement of customers, users and other stakeholders • provision of products and services to customers • measurable value (reducing overheads, improving efficiency, facilitating growth, recruiting talent) • supporting processes and business models (product design, manufacturing control, data modelling, local and remote working) • context and market environment (stakeholders, user profiling, personalised/appropriate content, data).	**E5** **D1**
5.3 Technical change management		

5.3.1	Understand the factors that trigger change in organisations:
	• Planned for factors
	  ○ adding additional features and/or services
	  ○ diversification
	  ○ scaling
	  ○ rebranding
	  ○ adoption of new technologies
	  ○ changes in legislation
	  ○ response to competition.
	• unforeseen or previously unpreventable factors
	  ○ crisis (natural disasters, terrorism, cyber attacks)
	  ○ zero-day vulnerabilities
	  ○ data corruption
	  ○ system failures.
		E2 E4
		E5

5.3.2	Understand the technical change management process, including: • identifying the changes to be made • identifying and communicating potential risks and desired impact(s) to stakeholders • configuration of the new system or process • method of implementing change (parallel, phased, direct, pilot) • documenting the change process • importance of rollback planning • importance of ensuring reproducibility of performance and outcome • traceability of requirements throughout the development lifecycle.	**E1 E5 M6 M9 D3**
5.3.3	Understand how organisations respond to, prepare for, manage and reinforce change (relevant to digital) in a range of contexts: • economic, banking and financial • environmental • legal • people • political • regulatory • social • technological.	**E1 E5 M6 M9 D3**
5.3.4	Understand the benefits and drawbacks of technical change in organisations in relation to: • productivity • communication • security • replacing existing products • updating or changing processes • support for stakeholders • costs • stakeholder experience • company reputation.	**E1 E4 M2 M6 M9**
5.4 Risks in a business context		
5.4.1	Understand the potential risks to organisations of use of digital systems and technologies: • security breaches • privacy breaches • regulatory and legal non-compliance • system failure • audience exclusion • emerging rival services (mobile devices, digital download and cloud services) • rapid changes in technology and trends (education and transport sectors).	**E1 E5 M6 D5**

5.4.2	Understand the potential impact of identified risks on the organisation and its stakeholders.	**E1 E5 M6 D5**

Content area 6: Data

Students must apply an understanding of the use of data by organisations to support business needs. They should explore the benefits and challenges that digital technologies present in terms of the creation and use of data.

What students need to learn		
6.1 Data and information in organisations		
6.1.1	Understand the differences and links between data, information and knowledge.	**M6** **D4**
6.1.2	Understand why organisations need data and information and how they are used: • analysing market trends to identify patterns which inform decisions • system performance analysis • user monitoring • targeted marketing • informed decision making (strategic, tactical and operational) • threat/opportunity assessment (break-even, predictive models, cost analysis, market trends).	**E5** **M6** **D4**
6.1.3	Understand how data is generated: • human generated • artificial intelligence/machine learning • sensors • Internet of Things (IoT) • transactional data.	**E5** **M6** **D4**
6.2 Data formats		
6.2.1	Understand the forms that data can take and the implications this has on use and analysis: • data types (date, integer, real, character, string, Boolean) • common forms of data format (JSON, fixed-width text file, CSV, ASCII, XML).	**E5** **M5 M6** **D4 D6**
6.2.2	Understand the difference between file-based and directory-based structures, and how they are used in data analysis.	**E5** **M5 M6** **D4 D6**

6.3 Data systems		
6.3.1	Understand the features and functions of data systems and their importance to organisations: • data wrangling (structure, clean, enrich, validate, output) • core functions (input, search, save, integrate, organise (index), output, feedback loop) • data entry and maintenance (online data entry, risk of data entry errors, time to create the entry screen and enter data) • visualisation (graphs/charts, data tables, reports, infographics).	**E5** **M5 M6** **D4 D6**
6.3.2	Understand the purpose of business information tools and their use in business (e.g. business intelligence software, financial planning and analysis, Customer Relationship Management (CRM)).	**E5** **M5 M6** **D4 D6**
6.3.3	Understand the features of different data models and how organisations use them to organise data: • conceptual data model • logical data model • physical data model • hierarchical database model • relational model.	**E5** **M5 M6** **D4 D6**
6.4 Data management		
6.4.1	Understand factors that determine how data is gathered, entered and maintained: • the 'Six Vs' (volume, variety, variability, velocity, veracity, value) • data assurance/quality (validation, verification, reliability, redundancy) • types of data • research population • qualitative data and quantitative data • legislation and regulatory compliance • ethics • organisational factors (time, skills, cost).	**E5** **M5 M6** **D4 D6**
6.4.2	Understand the purpose of data analysis tools and their use in business (data warehousing, data lakes, data mining, reporting).	**E5** **M5 M6** **D4 D6**
6.4.3	Understand the role of metadata classification in defining the meanings of data.	**E5** **M5 M6** **D4**
6.4.4	Understand the use of data/access entitlements/permissions management, and its impact on organisations and stakeholders: • authorisation • privileges • access rights • rules.	**E5** **M5 M6** **D4 D6**

6.4.5	Understand how data can be accessed and managed across different platforms: • the role and use of Application Programming Interfaces (APIs) in managing, accessing and using data.	**E5** **M5 M6** **D4 D6**
6.4.6	Understand the concepts of data at rest, data in use and data in motion, and when each is used.	**E5** **M5 M6** **D4**

Content area 7: Digital environments

Students should be able to apply an understanding of the different platforms of delivery that enable access to digital tools and services. They should explore how different digital environments meet the needs of organisations and their stakeholders. They must apply an understanding of digital environments in a range of business contexts.

What students need to learn		
7.1 Physical environments		
7.1.1	Understand the features and characteristics of different types of physical computer system: • personal computers • mobile devices • servers • smart/internet-enabled devices.	**E1 E2 E4 E5** **D1**
7.1.2	Understand the features and characteristics of hardware and peripherals used in physical computer systems: • input devices • output devices • processors • memory • secondary storage devices (internal and external) • motherboard/mainboard • cooling • sensors.	**E1 E2 E4 E5** **D1**

7.1.3	Understand the purpose and functions of software used in computer systems:
	• operating systems
	  ○ batch operating system
	  ○ multitasking/time-sharing operating system
	  ○ real-time operating system
	  ○ network operating system
	  ○ mobile operating system
	• utility software
	• application software
	• code development tools (IDEs, debuggers).
		E1 E2 E4 E5
		M6
		D1

7.1.4	Understand the benefits and drawbacks of software, hardware and peripherals in different contexts.	**E1 E2 E4 E5** **M6** **D1**

7.1.5	Understand how physical data storage and recovery systems work, their features, benefits and drawbacks: ● redundant array of independent disks (RAID) ○ RAID 1 ○ RAID 5 ○ RAID 10 ● network attached storage (NAS) ● storage area network (SAN).	
7.2 Networks		
7.2.1	Understand the benefits and drawbacks of connecting devices to form networks.	**E1 E2 E4 E5 M6 D1**
7.2.2	Understand the features, characteristics, benefits and drawbacks of wireless connection methods.	**E1 E2 E4 E5 M6 D1**
7.2.3	Understand the features, characteristics, benefits and drawbacks of wired connection methods.	**E1 E2 E4 E5 M6 D1**
7.2.4	Understand different types of network: ● LAN ● WAN ● PAN.	**E1 E2 E4 E5 M6 D1**
7.2.5	Understand the concepts of bandwidth and latency, and their effect on the performance of networks and connected systems.	**E1 E2 E4 E5 D1**
7.2.6	Understand the concept of different network models: ● client-server ● thin client ● peer-to-peer.	**E1 E2 E4 E5 M6 D1**
7.2.7	Understand the characteristics of network topologies: ● logical vs physical ● star ● mesh ● tree ● VLAN.	**E1 E2 E4 E5 D1**

7.2.8	Understand the role and characteristics of common components of a network: • server • internet connection/internet backbone • router • network switch • client.	**E1 E2 E4 E5 D1**
7.2.9	Understand the seven-layer OSI model to describe how applications communicate over a network, including the function and related protocols of each layer: • application layer • presentation layer • session layer • transport layer • network layer • data link layer • physical layer.	**E1 E2 E4 E5 M4 M6**
7.2.10	Understand the four-layer TCP/IP model to describe how applications communicate over a network, including the function and related protocols of each layer: • application layer • transport layer • internet layer • network access layer.	**E1 E2 E4 E5 M4 M6**
7.2.11	Understand the role of data packets in transmitting over a network, including: • contents and structure of a data packet • role of the components of a data packet • packet switching • error handling – Cyclic Redundancy Check (CRC).	**E1 E2 E4 E5 M4 M6 M10**
7.2.12	Understand the role of common network protocols including: • DHCP • DNS • FTP • HTTP / HTTPS • NTP • POP3 / IMAP4 / SMTP • TCP/IP.	**E1 E2 E4 E5 M4 M6**
7.2.13	Understand how physical, virtual and cloud environments, along with networks, are used in combination, including the Internet of Things (IoT), to solve problems and meet the needs of organisations and their stakeholders.	**E1 E2 E4 E5 M10 D1 D3**

7.3 Virtual environments		
7.3.1	Understand the key features of virtual environments: • increased security • managed execution • sharing • aggregation • emulation • isolation • portability.	**E1 E2 E4 E5 M10 D1 D3**
7.3.2	Understand the benefits and drawbacks of the use of virtual environments for organisations in a range of contexts.	**E1 E2 E4 E5 M10 D1 D3**
7.4 Cloud environments		
Understand the ways in which organisations use cloud environments to provide access to digital tools, services, storage and systems.		
7.4.1	Understand the concepts of cloud computing deployment in terms of: • applications • data • runtime • middleware • operating system • virtualisation • servers • storage • networking.	**E2 E4 E5 M6 D1 D3 D4**

7.4.2	Understand common cloud delivery models and the way in which responsibility and ownership of resources are distributed between the subscriber and service provider:
	• IaaS (Infrastructure as a Service)
	  ○ subscriber (applications, data, runtime, middleware, operating system)
	  ○ service provider (virtualisation, servers, storage, networking).
	• PaaS (Platform as a Service)
	  ○ subscriber (applications, data)
	  ○ service provider (runtime, middleware, operating system, virtualisation, servers, storage, networking).
	• SaaS (Software as a Service)
	  ○ subscriber (user only)
	  ○ service provider (applications, data, runtime, middleware, operating system, virtualisation, servers, storage, networking).
		E2 E4 E5 M6 D1 D3 D4

7.4.3	Understand the concept of DaaS (Data as a Service) in terms of: • data science platforms • dashboards • business information tools • data lakes • databases • file systems.	**E2 E4 E5 M6 D1 D3 D4**
7.4.4	Understand how DaaS (Data as a Service) is used by organisations, and the benefits and drawbacks it provides for organisations and their stakeholders.	**E2 E4 E5 M6 D1 D3 D4**
7.4.5	Understand the concept of cloud sourcing and cloud portability, and the implications for service providers and organisations (subscribers).	**E2 E4 E5 M6 D1 D3 D4**
7.5 Resilience of environment		
7.5.1	Understand the need to ensure digital environments are resilient, and the impact on organisations and stakeholders if this is not achieved.	**E2 E4 E5 M6 D1 D4**
7.5.2	Understand methods used to improve the resilience of digital environments: • data and system redundancy • back-up systems • hot, cold and warm sites • data back-up and recovery procedures • device hardening	**E2 E4 E5 M6 D1 D4 D6**
7.5.3	Understand the benefits and drawbacks of methods used to improve the resilience of digital environments.	**E2 E4 E5 M6 D1 D4 D6**

Content area 8: Security

Students should be able to apply an understanding of the potential risks posed by the use of digital to an organisation and its stakeholders. Students should explore established and emerging risks, and understand ways in which risks can be mitigated. They should be able to demonstrate an understanding of risks and mitigation measures in a range of business contexts.

What students need to learn		
8.1 Security risks		
8.1.1	Understand the importance of maintaining privacy and confidentiality of an organisation's information, as well as that of stakeholders, including: • information about salaries • employee benefits/perks • client lists • trade secrets • sales numbers • customer information • news about pending restructuring.	**E2 E4 E5** **M6** **D1 D5**
8.1.2	Understand the potential impact on an organisation of failing to maintain privacy and confidentiality.	**E2 E4 E5** **M6** **D1 D5**
8.1.3	Understand potential technical threats and vulnerabilities to systems, data and information, including: • botnets • distributed denial-of-service (DDoS) • hacking • malware (including ransomware) • social engineering (pharming, phishing) • insecure Application Programming Interfaces (APIs) • use of ad hoc or open networks • eavesdropping/man-in-the-middle attacks.	**E2 E4 E5** **M6** **D1 D5**
8.1.4	Understand potential physical vulnerabilities to systems, data and information, including: • location of system or asset • circumstances of use • characteristics of users/community • system or asset layout • system or asset design/robustness.	**E2 E4 E5** **M6** **D1 D5**

8.1.5	Understand potential **human** threats and vulnerabilities to systems, data and information, including: • human error • malicious employees • disguised criminals • targeted attack.	**E2 E4 E5** **M6** **D1 D5**
8.2 Threat mitigation		
8.2.1	Understand the concept of the CIA (confidentiality, integrity, availability) and how it can be applied to define security aims.	**E2 E4 E5** **M6** **D1 D5**
8.2.2	Understand the interrelationship between security, identity, confidentiality, integrity, availability, threat, vulnerability and risk management within a business context.	**E2 E4 E5** **M6** **D1 D5**

8.2.3	Understand processes and procedures to mitigate threats and ensure security, including:
	• air gapping
	• anti-virus and anti-malware programs
	• certification of APIs
	• configuration and management of software-based access control
	• device hardening
	• encryption (hashing, asymmetric, symmetric)
	• user access restrictions
	  ○ usernames, passwords and passphrases
	  ○ data access levels/permissions
	  ○ physical access control/restrictions
	• multi-factor authentication (possession-based, biometric, knowledge, location-based)
	• firewalls
	• password managers
	• policy, policy enforcement and training
	• SYN cookies
	• use of Virtual Private Networks (VPNs)
	• security testing (penetration testing, ethical hacking).
		E2 E4 E5
		M6
		D1 D5

Core project

Task 1

Planning a project		
Project planning tools	Be able to use project planning tools to apply understanding of project planning in response to a scenario.	
Gantt chart	a) Assess the strengths and skills of people and assign appropriate tasks to them. b) Make scheduling decisions in response to a defined deadline. c) Prioritise activities or tasks based on analysis of requirements. d) Demonstrate how to correctly and appropriately assign resources to project tasks. e) Produce a Gantt chart to show project tasks and organise them efficiently, using an appropriate Software Development Lifecycle model.	**E5** **M1** **D1**
Resource and cost plan	a) Identify and calculate costs of a project, including: • materials • physical resources • personnel. b) Select and allocate resources to the resource list, and correctly attribute costs to provide an accurate estimate of the total project cost.	**M8**
Rationale	a) Consider the factors that are most relevant when planning projects. b) Justify project planning decisions made, with consideration given to: • cost, risk and benefits to identified stakeholders • order and timing of tasks • selection and allocation of resources, including personnel and physical resources c) dependencies and prerequisites.	**E5** **M9**

Task 2

Identifying and fixing defects in an existing code		
Use of testing to identify defects	a) Assess the given code against requirements. b) Carry out testing to identify issues in given code. c) Perform any remedial actions required, justifying any decision made when fixing the defect.	
Documenting the testing process	Provide annotated evidence of testing, including: • identifying tests to be carried out • describing the purpose of the identified test • identifying test data to be used (valid, valid extreme, invalid, invalid extreme, erroneous) • describing the expected results • describing the actual results of the tests performed • comparing the actual results of testing with the expected results • describing any further actions that are required • refining the system as required.	**E1 E2** **M10** **D3**
The solution	a) Correct errors in code add and/or remove code to ensure that the given code is functional and meets the given requirements. b) Follow appropriate programming conventions when fixing code to ensure that it makes use of precise logic and programming structures throughout, so that the program produces consistently correct outcomes.	**M4** **M5** **M7**

Task 3

Designing a solution		
Decomposition of the problem	a) Break down the problem into smaller parts suitable for computational solutions, justifying any decisions made. Make effective use of detailed abstraction and refinement. b) Use elements of reusable components. c) Use good decomposition to show all the necessary subsystems that make up the main solution. d) Visualise the decomposition. e) Use appropriate tools for communicating algorithms (flowcharts, pseudocode).	**D4**
Application of logical thinking	a) Describe the parts of the solution using algorithms, justifying how these algorithms form a complete solution to the problem. b) Clearly define the steps. c) Uniquely define each step. They should depend on the input and the result of the preceding steps. d) Ensure the algorithm makes use of key constructs (e.g. sequence, selection and iteration).	**M2** **D2**
Use of conventions	a) Demonstrate correct use of structure and convention for the chosen method of communication (flowcharts, pseudocode), such as correct use of symbols for flowcharts and key words used in pseudocode. b) Select and make consistent use of appropriate naming conventions throughout.	**E3**
Communication of the design	a) Ensure design documents are of sufficient detail to: • effectively communicate the intended solution • allow the client to make informed decisions • allow a third party to use design documents to create the proposed solution. b) Communicate intended solution effectively and clearly, with use of: • appropriate combination of written and diagrammatical presentation • appropriate use of technical vocabulary • consideration of audience • explanations of structures and process in the design.	**E4** **D3**

Task 4a

Developing a solution		
The solution	Apply an undertaking of programming to develop a solution that meets the requirements of a brief, including: • refining the system as required • demonstrating an appropriate level of technical skill, and understanding of programming techniques and problem solving • use of pre-written and user-written modules with appropriate interface • discussion of any issues that may have come about from the testing.	
Code organisation	Ensure code produced for the solution is appropriate to meet the demands of the brief, including: • trying to avoid multiple pages of nested if-clauses and for-loops with a lot of copy-pasted procedural code • clear meaningful indentation • precise use of logic functions, classes or objects, with proper structure • comments whenever possible to help explain the logic • good use of local variables and minimal use of global variables • use of constants • well-designed interface • consistent style throughout • defensive programming and handling data securely • good exception handling.	**M10**
User experience	Meet user needs, with consideration of: • consistency of the product • accessibility to all types of users • user interface design (visual design in line with brand goals) • meaningful feedback, changes made and problems overcome.	**E6**
Responsive-ness to the brief	a) Formal documentation of testing is not required in this activity. b) Testing by the student should still be carried out but its use should be implicit through: • a working product • a product that meets the requirements detailed in the given task brief.	**E6**

Task 4b

Reflective evaluation		
Programming outcomes	a) Be able to apply reflection and evaluation techniques. b) Provide evidence that the product meets brief requirements: • include measures against success criteria • provide evidence that the product meets user needs from testing • discuss how it could be improved if the problem was revisited and given detailed consideration.	
Comparison to designs	• Provide a well-developed line of reasoning which is clear and logically structured. The information presented is relevant and substantiated. • Discuss the maintainability of the solution. • Discuss potential further developments of the solution. • Discuss any changes and why they are different from the original designs.	**E3**

Scheme of Assessment – Core Component

There are three assessments in the Core Component of the *T Level Technical Qualification in Digital Production, Design and Development*:

1. Examination Paper 1: Digital Analysis, Legislation and Emerging Issues
2. Examination Paper 2: The Business Environment
3. Employer Set Project.

Core examinations

Paper 1: Digital Analysis, Legislation and Emerging Issues
Written examination: 2 hours 30 minutes **33.33% of the core assessments** **100 marks**
Content overview **1.** Problem solving **2.** Introduction to programming **3.** Emerging issues and impact of digital **4.** Legislation and regulatory requirements
Assessment overview • An externally-assessed written examination comprising two sections. Students answer all questions in Section A and Section B. • The examination will include short, medium and extended open-response questions, as well as labelling questions. • The examination will be set and marked by Pearson.
Administration The paper must be assessed under exam conditions, following JCQ's Instructions for Conducting Examinations (ICE).

Paper 2: The Business Environment
Written examination: 2 hours 30 minutes **33.33% of the core assessments** **100 marks**
Content overview **5.** Business context **6**. Data **7.** Digital environments **8.** Security
Assessment overview • An externally-assessed written examination comprising two sections. Students answer all questions in Section A and Section B. • The examination will include short, medium and extended open-response questions, as well as labelling questions. • The examination will be set and marked by Pearson.
Administration The paper must be assessed under exam conditions, following JCQ's Instructions for Conducting Examinations (ICE).

Both examinations will follow the same paper structure but they will assess different core content, and will be available paper-based. There are two sections in each paper:

- Section A is weighted 40%.
- Section B is weighted 60%.

Core examination Assessment Objectives

Assessment Objective			Range for Papers 1 and 2	Paper 1 proportion	Paper 2 proportion
AO1	**1a (i)**	**Knowledge** (isolated knowledge)	0%–3%	6%	6%
	1a (ii)	**Knowledge** (embedded knowledge)	3%–6%		
	1b	**Understanding**	N/A	25%	28%
AO2		**Application**	N/A	48%	45%
AO3		**Analyse and Evaluate**	N/A	21%	21%

Employer Set Project

Core project – Employer Set Project
Externally assessed project: 14.5 hours **33.33% of the core assessments** **100 marks**
Content overview When responding to the core project, students will need to draw on knowledge and understanding form across the core content in a synoptic manner, in order to effectively respond to a brief within a vocational context.
Assessment overview There are five parts to the assessment: Task 1: Planning a project Task 2: Identifying and fixing defects in existing code Task 3: Designing a solution Task 4a: Developing the solution Task 4b: Reflective evaluation • Students will undertake the assessed elements of the project tasks under supervised conditions. • The assessment will take place over multiple sessions up to a combined duration of 14.5 hours. • The project outcomes will consist of a portfolio of evidence submitted electronically. • Students will undertake a project in response to a realistic contextual challenge. • The project is validated by an employer panel, taking into account the client's requirements and the user experience. • The project will consist of planning documentation, an annotated digital portfolio, a prototype digital product, testing evidence and evaluation. • The project will be set and marked by Pearson.
Administration Providers must follow the guidance in the following: • General Administrative Support Guide • Administration Support Guide for the specific Technical Qualification Employer Set Project (if applicable) These are located on the Training and Admin Support webpage.

Employer Set Project Assessment Objectives

Assessment Objective			Proportion
AO1	**1. Planning**	Plan an approach to developing solutions to solve problems in response to a brief.	17%
AO2	**2. Application**	Apply knowledge and skills to develop software, create an artefact, fix defects and mitigate risks to security.	43%
AO3	**3. Selecting relevant techniques and resources**	Select relevant tools, techniques and resources to respond to a brief and work in a collaborative environment.	5%
AO4	**4a. Maths skills**	Use appropriate maths skills to realise a project outcome in response to a brief.	3%
	4b. English skills	Use appropriate English skills to communicate technical information to both technical and non-technical audiences.	
	4c. Digital skills	Use appropriate digital skills to realise a project outcome in response to a brief and communicate technical information to both technical and non-technical audiences.	
AO5	**5a. Project Outcome**	Realise a project outcome by producing software and artefacts in response to a brief.	23%
	5b. Review	Review how well digital solutions meet a brief, using reflective evaluation.	9%

Resources for the delivery of the Core Component content

The following resources will be required for the delivery of the Core Component for this technical qualification:

- Python 3 and the appropriate IDE
- Python Libraries, in addition to standard libraries or other libraries that provide the same/comparable functionality/capabilities:
  - pandas
  - Matplotlib.

4. Occupational Specialist content - Digital Production, Design and Development

Content summary

The Occupational Specialist content covers the knowledge and skills needed to achieve threshold competence across the following areas:

1. Be able to analyse a problem to define requirements and acceptance criteria aligned to user needs

What students need to learn	
1.1 Understand the stages of the software development life cycle and be able to apply them to digital projects	

Life cycle stage: Research and familiarisation
• Explore and understand the initial client request/project brief.
• Carry out research relating to the specific context and market environment, including:
  o common problems and risks
  o current uses of hardware and software within the identified context
  o newly emerging technologies
  o existing or potential solutions and how these meet different user needs
  o industry/situational-specific guidelines and regulations.
• Identify shortfalls in own skills and knowledge and plan learning opportunities to make up for these shortfalls.
Life cycle stage: Planning and requirement analysis
• Identify business requirements such as new software and amending/increasing security.
• Assess the measurable value of the proposed solution in relation to:
  o the user
  o the client/business.
• Apply computational thinking (decomposition, pattern recognition and abstraction) to split the problem into discrete objects.
• Define the functional and non-functional requirements of the solution.
• Define the key performance indicators (KPIs) of the solution.
• Identify the performance constraints in digital projects.
• Create user acceptance criteria.
• Schedule projects (tasks, subtasks, milestones).
• Allocate appropriate resources to digital projects.
• Estimate costs of digital projects.
	E1 E2 E3 E4 E5
	M2 M6 M7 M8 M9 M10
	D1 D2 D3 D4

• Choose programming language(s) for digital projects based on key criteria, including:
  o suitability for the proposed task
  o organisational policy
  o scalability
  o security
  o availability of trained staff
  o costs
  o reliability.
  o Identify risks and explore ways to mitigate these risks.
Life cycle stage: Performing a user analysis
• Select and use business analysis models to aid problem solving, including:
  o user stories
  o activity diagrams
  o mind maps
  o product road maps
  o process diagrams
  o entity relationship diagrams.
Life cycle stage: Designing the product
• Create interface designs.
• Plan how to solve key problems (design algorithms).
• Create data requirement designs.
• Create a proof of concept.
• Produce initial testing schedule.
Life cycle stage: Developing and testing the product
• Create a prototype.
• Plan and implement appropriate testing, including:
  o module testing
  o system integration testing
  o automated testing
  o user testing and feedback.
Life cycle stage: Deploying/implementing the product
• Install and configure the product.
• Update the product.
Life cycle stage: Maintenance
• Provide system support.
• Provide user support.
• Carry out bug fixing.
• Arrange/carry out user training.
• Release updates to enhance the product.
• Understand the value to the organisation of the digital product and the role the software development life cycle plays.	

1.2 Understand the roles and responsibilities of the digital team within the software development life cycle	

• Product owner/client – sets and communicates the requirements of the product.
• Scrum master – facilitates the team to maintain team cohesion, ensuring the team has access to resources they require.
• Technical lead – provides the technical guidance and support for the project team.
• Project manager – plans, organises and manages (budget, scope, schedule, risk and quality) on all phases of a project.
• Software development team:
  o systems analyst – analyses the current system and provides the requirements for the new system; uses the requirements to design the new software solution
  o UX/UI designer – interviews users, researches market data and gathers findings to design the user interface
  o software developer/engineer – uses the designs to create and maintain a working solution that is usable, secure and stable
  o operations engineer – ensures the stability of the product
  o security engineer – ensures the security of the product.
• Software testers – responsible for the quality assurance of the software and development.
	E4 E5

1.3 Understand the key features of, and be able to select, appropriate project methodologies when developing a software solution	

Key features of Agile:
• Incremental delivery model:
  o sprint
  o epic
  o story
  o spikes.
• Emphasis on producing high-quality products, with initially limited functionality.
• Each increment delivers additional functionality.
• Requirements can be continually altered throughout the project.
• Can lack formal documentation.
• Users/clients see working products at each iteration.
• Cost-effective method to get an initial product to market.
• Cancelled/partially completed projects will still result in some usable code/product(s).
Key features of Scaled Agile:
• Expanded incremental delivery model:
  o sprint
  o epic
  o story
  o spikes
  o product increments.
	E5
	D1 D3 D6

- Emphasis on producing high-quality products with initially limited functionality.
- Each increment delivers additional functionality.
- Requirements can be continually altered throughout the project.
- Provides cohesive reporting.
- Users/clients see working products at each iteration.
- Cost-effective method to get an initial product to market.
- Cancelled/partially completed projects will still result in some usable code/product(s).
- Final iteration focuses on stability.

Key features of Waterfall:

- Rigidly structured with systematic steps.
- Progress measured through the number of artefacts completed.
- High initial costs with slower return on investment.
- Products cancelled during early stages may result in no usable code/product(s).
- Limited user/client interaction.
- High importance placed on documentation.

Key features of Rapid Application Development (RAD)

- Focus on quick creation of prototypes, which are systematically improved.
- Emphasis on developing all features first and quality second.
- Users/clients may see only partially working products or mock-ups during early iterations.
- Relies heavily on reuse of previously developed code.
- Suitable for small- to medium-scale projects.

Key features of LEAN:

- Emphasis on reducing 'waste' in software (and projects), which may include:
  - unnecessary or incorrect features
  - repeated tasks
  - unnecessarily complex solutions
  - ineffective communication
  - unnecessary changes.
- Decisions are made last minute to reduce chance of waste.
- Short iteration cycles with fast delivery of working versions.
- Emphasis is placed on producing iterations that are fit for use, and not specifically in delivering all features or requirements.
- Suitable for projects with small teams and when resources are limited.
- Understand features and approaches of user-centred design (UCD) and how it is used within a software development life cycle:
  - considerations for UCD:
    - Who is the user?
    - What does the user want to achieve by using the product?
    - How does the user interact with the product?
    - When does the user interact with the product?
    - Why is the product being used?
    - What is the user experience?

○ characteristics of UCD (empathetic, iterative, interdisciplinary) ○ stages of the UCD iterative approach: – understand the context of use – specify the user requirements – design the solution – assess against requirements.	
1.4 Understand and define the functional and non-functional requirements of a software solution	
• Understand the concept of 'secure by design' and how this influences the decisions made regarding functional and non-functional requirements. • Understand and define functional requirements of a software solution: ○ the inputs required ○ the data needed ○ the data processing that must take place ○ the logic of the system ○ the deployment and usage platforms for the software. • Understand and define non-functional requirements of a software solution in terms of: ○ security considerations ○ required accessibility features ○ scalability requirements ○ key performance indicators and metrics in relation to: – responsiveness – load handling – reliability ○ user acceptance criteria. • Understand the use of 'spike testing' as an early product-testing method to establish requirements and determine the extent of the problem and the required scope of a software solution.	**E1 E5 M2 M6 M7 M8 D2 D3 D6**
1.5 Investigate the current and potential uses of emerging technologies and how they impact on industries	
Investigate the impact on software development of emerging technologies such as: • operational technology (OT) • artificial intelligence (AI) and virtual intelligence (VI) • conversational AI • the Internet of Things (IoT) • machine learning • object recognition • biometrics • computer vision • robotics • cloud services and platforms • blockchain • data lakes and data warehousing • drones	**E1 E2 E3 E4**

• 3D printing • 5G networks.	
1.6 Identify and be able to address personal training needs that can boost job performance during the software development process	

• Ensure the software developer is able to complete the required solution:
  o identify what further knowledge is needed
  o identify what new skills are needed
  o determine if the software developer has the ability needed to complete the required solution.
• Use different methods to address the personal training needs required to enable completion of the project:
  o undertake coaching from a professional or peer
  o learn new skills on the job
  o carry out self-study
  o use online professional forums
  o sign up to internet workshops for additional training.
	E5 E6
	D1 D3 D6

2 Apply ethical principles and manage risks in line with legal and regulatory requirements when developing software

What students need to learn	
2.1 Investigate the legal and regulatory requirements that apply to developing software	
• Investigate and apply legal and regulatory considerations appropriate to the context and market environment in which they are developing software: o intellectual property rights and licenses o consumer protection o age ratings and classifications o advertising laws o data protection and privacy o copyright and patent o gambling legislation o responsibilities concerning staff and employment practices o territorial restrictions o system security o equality and diversity. • Investigate and apply standards for software development: o ISO/IEC/IEEE 90003:2018 o W3C. • Investigate and consider ethical implications that apply to software development: o codes of conduct o professional practice o software licensing o inclusion and diversity.	**E5** **M4 M5 M6** **D1 D5 D6**
2.2 Identify and manage risks that apply to software development	
• Assess the potential risks associated with a developing a software product appropriate to the context and market environment in which it is being developed in terms of: o data and system security (malicious vs accidental damage) o compatibility with other systems o speed of development o meeting functional and non-functional requirements o meeting key performance indicators (KPIs) o legal and ethical considerations o user engagement o product reach o assessment of risk (likelihood vs seriousness) o potential impact of risk o potential ways to mitigate identified risks o contingency planning o ongoing monitoring.	**E5** **M4 M5 M6** **D1 D5 D6**

• Investigate policies and procedures that apply to software development to manage and mitigate risks:
  o backup
  o security
  o confidentiality, integrity and availability (CIA)
  o personnel, skills and training
  o business continuity planning
  o disaster recovery planning.
• Be able to make and justify software development decisions and recommendations based on effective assessment of risk vs reward in relation to the context and market environment in which they are developing software.	

3 Discover, evaluate and apply reliable sources of knowledge

What students need to learn	
3.1 Understand and evaluate the reliability of different sources of knowledge	

• Use different sources to find reliable information:
  o use search engines to find reliable websites
  o read wikis
  o read blogs
  o read academic papers
  o talk to peers
  o join forums
  o look at code comments
  o use code repositories.
• Evaluate the reliability of different sources:
  o reputation – find out who the author is and whether they are credible
  o understand bias – sources written by a particular individual/organisation
  o look at the evidence used to support the content of the digital source
  o cross-referencing/triangulation – compare to other sources
  o check how current the content is – note the date website was last updated.
	E1 E4 E5 E6 M2 M4 M6 M7 M8 D1 D3 D4 D5 D6

3.2 Select and use techniques to obtain qualitative and quantitative data to be able to evaluate software solutions	
• Use verbal feedback (formal and informal). • Create and deliver surveys/questionnaires. • Select and use performance and use data. • Conduct user observation and complete observation records. • Create a focus group that represents a cross-section of the target audience. • Conduct interviews to gather individual thoughts on a software solution. • Use peer mentoring. • Use formal line management and appraisal procedures.	**E1 E4 E5 E6 M2 M4 M6 M7 M8 D1 D3 D4 D5 D6**

4 Design

What students need to learn	
4.1 Understand the use of common design approaches	

• Be able to use critical thinking in order to select appropriate design approaches when developing a software product, and apply them within a larger project methodology.
• Understand features and approaches of function-orientated (top-down) design:
  o how data flows through the system
  o systems made up of many sub-systems (functions)
  o data flow diagrams to show how each function handles or changes the data
  o problems broken down logically (divide and conquer), based on what each function should do within the whole system.
• Understand features and approaches of object-oriented programming (OOP) design:
  o core concept of making code reusable through standard OOP structures (methods, classes, objects/instances)
  o characteristics of OOP (encapsulation, data abstraction, polymorphism and inheritance)
  o common OOP design patterns (creational, structural, behavioural).
• Understand features and approaches of data model design:
  o visualisation of the data needed and how it will be organised
  o common data models (conceptual, logical, physical, hierarchical relational)
  o common data modelling tools:
  – entity–relationship model (ER model)
  – Unified Modeling Language (UML).
• Understand features and approaches of test-driven development (TDD):
  o core concept that each new feature starts by writing a test that defines the improvements or function of that feature
  o common approaches to test-driven development:
  – add a test
  – run all tests until the new test fails
  – write some code
  – run tests and refactor code
  – repeat.
• Understand features and approaches of behaviour-driven development (BDD):
  o core concept that development is informed by the required behaviour of a software unit which is specified before coding starts
  o desired behaviours must have business value and relate to specified requirements
  o common behaviour specification structure (title, narrative, acceptance criteria).
	E1 E2 E3 E4 E5
	M1 M2 M3 M4 M5 M6 M7 M8 M10
	D1 D2 D4 D6

• Understand features and approaches of functional design: o core concept that a program is built and structured as a set series of modules that perform a single defined process/function o characteristics of functional programming (recursion, closures, first class functions, higher order functions, anonymous functions, currying) o common components of functional design (arguments, statements, blocks, procedures, functions/sub-routines).	
4.2 Understand and select platforms used for source code and content management	
• Understand the features of different software development platforms used at different stages of developing a software product: o coding o repositories o branching o building o testing o deployment. • Understand the differences between proprietary and open source platforms and how these may affect software design. • Understand the process required to manage the development workflow (Git flow, GitHub flow). • Understand the differences between different platforms and be able to make informed decisions about which to use and when, in relation to: o target audience o budget o technical features o staff and training o ease/speed of development o platform updates o security o reliability o performance o compatibility.	**E1 E3 E5 M10 D1 D2 D3 D4 D5 D6**
4.3 Understand and be able to design a software solution	
• Understand and apply user experience (UX) design principles: o consistency to ensure a product is intuitive for the user, aesthetically pleasing and promotes brand recognition o information hierarchy to navigate the product more easily o visual hierarchy to enable more important content to stand out o confirmation to ensure the user is aware of any actions performed and their impact o user control to allow navigation of the product, efficient workflow and error correction o accessibility to ensure the product is appropriate for a wide range of users.	**E1 E2 E3 E4 E5 M1 M2 M3 M4 M5 M6 M7 M8 M10 D1 D2 D3 D4 D6**

• Understand and apply user interface (UI) design principles:
  o wireframes
  o style guides
  o clickable prototype.
• Understand the features of content management systems and how they are used during design and development:
  o search engine optimisation (SEO)
  o page/screen management
  o social media integration
  o analytics tools
  o workflow management
  o publishing controls
  o security features and management
  o versioning and rollback
  o content repositories
  o open APIs
  o multilingual support
  o support.
• Create program designs using accepted conventions:
  o pre-defined code
  o flowcharts using standard BCS symbols
  o pseudocode
  o control structures (sequence, selection/branching, iteration)
  o data types
  o data validation
  o data structures.
• Understand considerations when selecting assets (graphics, audio, video, code) to be used in software design, including:
  o file types
  o file size
  o compression
  o streamed or encoded audio
  o streamed or embedded video
  o use of metadata
  o quality required
  o bandwidth and storage available
  o target platform.
• Understand the features of different target platforms and how they affect the design and development of a software solution:
  o operating systems
  o file systems
  o server and other infrastructure (physical, virtual)
  o programming language stack
  o mobile and web.	

• Understand how and when to use databases to support software solutions:
  o user management
  o e-commerce tasks (stock, order processing, page personalisation)
  o diagnostics
  o performance analysis.
• Create database designs to support software solutions using:
  o data dictionary/library
  o entity relationship diagram (ERD)
  o normalisation to third normal form.
• Identify and understand network integration points:
  o which data is processed locally, e.g. user input for a computer game
  o which data is processed remotely, e.g. actions of all players in multi-player environments
  o how data is transferred between local and remote sources, e.g. remote system calls
  o how local and remote systems will be connected, e.g. type of network, combination of networks
  o system boundaries to identify which tasks are carried out locally or remotely
  o which external systems to integrate with, e.g. OPTA for sports data.	

5 Create solutions in a social and collaborative environment

What students need to learn	
5.1 Understand the reasons for using collaborative techniques	

• Use team working on common projects that allows for:
  o a reduction in development time
  o better communication
  o sharing of knowledge
  o development of software development skills
  o code reviews
  – paired programming
  – informal walkthroughs
  – formal inspections.
• Understand when to collaborate and when working independently is more appropriate.
	E5
	M10
	D1 D3

5.2 Understand and be able to select appropriate technologies used in a social and collaborative environment	

• Collaborative technologies to aid working as part of a team:
  o communication (e.g. email, instant messaging)
  o resource management (e.g. cloud storage, back-up, synchronisation)
  o knowledge (collaboration hubs, wikis, community forums, news sites)
  o documentation for technical and non-technical audiences.
• Code collaboration technologies.
• Version control.
• Source control.
• Integrated Development Environments (IDEs).
	E5
	M10
	D1 D3
	D6

6 Implement a solution using at least two appropriate languages

What students need to learn	
6.1 Select and use languages to create a software solution for a software project appropriate to the context and market environment in which they are developing software	

• Select and use at least two appropriate languages to implement front-end and back-end solutions:
  o Python
  o C, C# and C++
  o Javascript
  o Java
  o Go
  o Kotlin
  o PHP
  o SQL
  o MongoDB/MQL
  o Node.Js.
• Be able to select tools and techniques to embed front-end and back-end solutions into a single usable artefact as appropriate to the chose solution such as:
  o frameworks and runtime environments (Node.JS Angular, Django, Bootstrap, Flask)
  o use of GUI components
  o embedding code/scripts within HTML5 and CSS
  o compiling/packaging.
• Be able to select and use appropriate application programming interfaces (APIs), packages, modules and libraries to add functionality and compatibility as appropriate to the chosen solution:
  o generating dynamic page content
  o containerisation
  o stateful vs stateless components
  o form handling
  o file and data handling:
  – handle local files
  – create, open, read, write, delete and close files on a server
  – send and receive cookies
  – add, delete and modify data in a database
  o interface components
  o media content
  o adaptive/responsive layout
  o working with existing applications, operating systems, cloud-based and traditional platforms
  o working with specific devices
  o communication over a network
  o infrastructure as code
	E5
	M1 M2 M3 M4 M5 M6 M7 M8 M10
	D1 D2 D3 D4 D6

○ security features: – controlling user-access – encrypting data. ● Understand the use of continuous integration pipelines and how they can be used to build and deliver software solutions: ○ the concept of 'continuous integration – continuous deployment' (CI/CD) ○ common stages of continuous integration pipelines: – source code control – build automation – unit test automation – deployment automation – monitoring. ● Use common coding conventions: ○ naming conventions ○ code annotations/commenting ○ modularisation ○ structure/indentation ○ version control. ● Use good practice when developing digital products (twelve factor principles): ○ One codebase tracked in revision control, many deploys. ○ Explicitly declare and isolate dependencies. ○ Store config in the environment. ○ Treat backing services as attached resources. ○ Strictly separate build and run stages. ○ Execute the app as one or more stateless processes. ○ Export services via port binding. ○ Scale out via the process model. ○ Maximise robustness with fast start-up and graceful shutdown. ○ Keep development, staging and production as similar as possible. ○ Treat logs as event streams. ○ Run admin/management tasks as one-off processes.	
6.2 Select and use appropriate tools and features to create user interfaces that apply user experience (UX) design principles	
● Be able to select and use appropriate user interface features: ○ images and animation ○ audio ○ effects ○ interactions: – user input – output/user feedback – textual – graphical – audio – haptic	**E5** **M1 M4 M7 M8 M10** **D1 D2 D6**

○ data visualization: – dashboard – graphing – data presentation. ● Be able to select and use appropriate user interface techniques: ○ layout grids ○ layout and use of space ○ font selection and typesetting ○ letter spacing ○ line spacing ○ justification ○ use of colour and contrast ○ input focus ○ hover controls. ● Be able to make appropriate design decisions, with consideration of: ○ browser support ○ target device/platform ○ user characteristics ○ available bandwidth ○ style and branding ○ accessibility ○ user input method, including: – voice – text – touch screen – mouse.	
6.3 Connect code to data sources as part of a software project	
● Create data sources to support software solutions using a database. ● Connect to data sources using different connections: ○ application programming interface (API): – types of requests and request methods – endpoints – retrieving and parsing data – displaying data – API keys ○ Java database connectivity (JDBC) – core API (application programming interface) – driver manager – connection statement – prepared statement – result set – SQL queries ○ open database connectivity (ODBC) – application – driver manager – driver – data source	**E5** **M4 M5 M6 M10** **D1 D3 D4 D5 D6**

○ connection method – database name or data source – credentials – username/password – optional parameters ○ extracting data ○ storing data ○ updating data ○ deleting data ○ connecting to network resources, using tools within the development environment. • Be able to select and use data sources and connection methods appropriate to the context and market environment in which they are developing software.	
6.4 Select and use deployment methods for a software project	
Use suitable deployment methods such as: • local installation • network/server installation • mobile platforms • web-based platforms • cloud-based platforms • containerisation • container scheduling platforms.	**E5** **M5 M6 M10** **D1 D4 D6**

7 Test a software solution

What students need to learn	
7.1 Understand, select and apply functional, non-functional and front-end testing	

Be able to select and carry out appropriate functional, non-functional and front-end testing relevant to the component or product being tested and the stage within the software development life cycle.
• Functional testing:
  o unit testing
  o smoke testing
  o integration testing
  o system testing.
• Non-functional testing:
  o availability testing
  o compatibility testing
  o configuration testing
  o load testing.
• Front-end testing to check:
  o code/script performance and functionality
  o browser compatibility
  o operating system compatibility
  o cross-browser performance
  o formatting and rendering
  o loading times
  o responsiveness.
• Security testing:
  o vulnerability scanning
  o static analysis
  o dynamic analysis
  o integration analysis.
	E1 E4 E5 E6 M2 M4 M5 M6 M8 M10 D1 D2 D4 D6

7.2 Understand, select and apply testing techniques	
• Acceptance testing. • Alpha testing. • Beta testing. • Black box testing. • White box testing/structural testing.	**E1 E4 E5 E6 M2 M4 M5 M6 M8 M10 D1 D2 D4 D6**

7.3 Select appropriate tests and test data to test the functionality of software	

Be able to devise and apply an appropriate test plan to ensure software is functional.
• Purpose of the identified test.
• Test data:
  ○ valid test data
  ○ invalid test data
  ○ valid extreme test data
  ○ invalid extreme test data
  ○ erroneous test data.
• Pre-requisite to each test.
• Expected test results.
• Update the plan to include:
  ○ actual results
  ○ changes made
  ○ retests/regression testing following changes.
	E1 E4 E5 E6
	M2 M4 M5 M6 M8 M10
	D1 D2 D4 D6

8 Change, maintain and support software

What students need to learn	
8.1 Understand the changing nature of digital products and the factors that drive change	

Understand how business-driven development affects the types of maintenance to be performed:
• To prevent identified and foreseeable issues such as:
  o changes to regulatory requirements
  o compatibility with new products or technology
  o changes in business process
  o release of a new product/service.
• To correct unforeseen or previously unpreventable errors such as:
  o new vulnerabilities due to changes in other products or systems (zero-day)
  o targeted attack
  o data corruption
  o system failures.
• Iterative development of digital products to maintain relevance:
  o review following user/client feedback
  o developments in technology
  o competition with other organisations
  o the need to improve efficiency
  o the need to future proof products.
	E1 E2 E3 E5
	D3 D6

8.2 Understand the stages involved in the software change management process	

• Identify issues/changes made during the feedback/review process.
• Document developments and changes in a software project.
• Communicate with different audiences:
  o technical
  o non-technical.
• Plan the changes required.
• Schedule the changes.
• Carry out regression testing.
• Control and release updated products:
  o planned
  o reactive.
	E1 E2 E3 E5
	M6 M10
	D1 D3 D6

8.3 Understand how to maintain code as part of a larger team.	
Understand and apply good coding principles when developing, adapting and maintaining code: • separating code and using modularity • readability • use of common/accepted coding conventions • informative commenting/annotation within the code • updating change logs and other documentation.	**E1 E4 E5 E6** **M2 M4 M5 M6 M8 M10** **D1 D2 D3 D4 D6**

8.4 Understand how to support software users	

• Be able to communicate with technical and non-technical audiences, using appropriate tone and levels of technical vocabulary through:
  o face-to-face communication
  o remote conferencing
  o written communication:
  – blogs
  – formal reports
  – technical documentation
  – release notes
  – user guides/help files
  – FAQs
  o visual and audio communication
  – user demonstrations
  – screencast videos
  – narration (recorded voice, text-to-speech)
  o machine-readable application programming interface (API) contact.
• Apply systematic processes to users and resolve issues:
  o identify or replicate the issue
  o investigate the possible cause of issues:
  – user error
  – system error
  – application error
  – security breach.
• Apply testing techniques to:
  o identify errors in the system/code
  o make changes to the system/code as required
  o ensure error does not return
  o ensure no additional issues have been caused as a result of the changes made.
• Communicate how and when the issue was resolved to appropriate stakeholders.
• Document lessons learned.
	E1 E4 E5 E6 M2 M4 M5 M6 M8 M10 D1 D2 D3 D4 D6

Scheme of Assessment – Occupational Specialist Component

The *Level 3 T Level Technical Qualification in Digital Production, Design and Development* includes a single Occupational Specialist Component. Therefore, there is a single synoptic assessment for the Occupational Specialist Component, which is an extended 'design, development and implementation' project. The synoptic element of the project is important in order to ensure that students are able to demonstrate threshold competence; this is the principal reason why the occupational specialism is assessed via a single extended project assessment to ensure that students are able to evidence all the skills required by the Performance Outcomes.

The Occupational Specialist Component consists of a number of activities grouped into four substantive tasks.

Each task will be completed during a window set by Pearson, during which you will schedule supervised assessment sessions. In some cases, tasks will also involve opportunities for unsupervised activities, where the requirements of the skills being assessed make this necessary.

Occupational Specialism Project – Digital Production, Design and Development
Externally assessed project: 67 hours **145 marks**
Performance Outcomes In this unit, students will: **PO1:** analyse a problem to define requirements and acceptance criteria, aligned to user needs **PO2:** design, implement and test software **PO3:** change, maintain and support software **PO4:** create solutions in a social and collaborative environment **PO5:** discover, evaluate and apply reliable sources of knowledge **PO6:** apply ethical principles and manage risks in line with legal and regulatory requirements when developing software.
Assessment overview There are four parts to the assessment: **Task 1:** Analysing the problem and designing a solution. **Task 2:** Developing the solution. **Task 3a:** Gathering feedback to inform future development. **Task 3b:** Evaluating feedback to inform further development. Students will respond to a given scenario to complete a substantial Digital Production, Design and development project. Students will be assessed on their application of the skills listed for the Performance Outcomes. The tables below show each assessment area of a typical Occupational Specialist project and the relevant skills that the project will target. Students will not be assessed against specific 'knowledge' outcomes but will be expected to draw on and apply related knowledge to ensure appropriate outcomes when applying the skills in response to an assessment scenario.

Occupational Specialism Project – Digital Production, Design and Development
• Students will undertake the project under a combination of supervised and non-supervised conditions. • The assessment will take place over multiple sessions, up to a combined duration of 67 hours. • The project outcomes will consist of a portfolio of evidence submitted electronically. • Students will respond to a scenario to design and develop a software-based solution. • This project will be set and marked by Pearson.
Administration Providers must follow the guidance in the following: • General Administrative Support Guide • Administration Support Guide for the specific Technical Qualification Occupational Specialism (if applicable) These are located on the Training and Admin Support webpage.

Performance Outcomes

Performance Outcome		Weighting	
		Raw marks	% of total marks
PO1	Analyse a problem to define requirements and acceptance criteria, aligned to user needs.	26	16.1%
PO2	Design, implement and test software.	69	42.9%
PO3	Change, maintain and support software.	34	21.1%
PO4	Create solutions in a social and collaborative environment.	9	5.6%
PO5	Discover, evaluate and apply reliable sources of knowledge.	13	8.1%
PO6	Apply ethical principles and manage risks in line with legal and regulatory requirements when developing software.	10	6.2%

Resources for the delivery of the Occupational Specialist Component content

For the Occupational Specialist Component, the following resources are required:

- IDE and debuggers appropriate to the chosen languages
- automated testing tools for:
  - UI testing
  - performance testing
  - load/stress testing
  - compatibility testing
- access to communication, collaboration and data collection tools, including:
  - online code repositories (such as GitHub)
  - forums (e.g. stack overflow)
  - questionnaire/survey tools (e.g. Google forms, SurveyMonkey)
  - email
- access to online third-party digital content (e.g. video and graphics)
- screencasting and video editing software
- microphones and headphones.

5. Command Word Taxonomy

Command word taxonomy list

The following table shows the command words that will be used consistently in our assessments to ensure students are rewarded for demonstrating the necessary skills. The list below will not necessarily be used in every paper and is provided for guidance only.

Term	Definition
Assess	Give careful consideration to all the factors or events that apply and identify which are the most important or relevant. Make a judgement on the importance of something, and come to a conclusion where needed.
Complete (diagram)	Complete a diagram or process flow that has already been started.
Complete (table)	Provide the missing information for a table/diagram so that it is complete (contains all the necessary information).
Describe	Present two (or more) linked descriptive points on characteristics, features, uses or processes. Do not need to include a justification or reason.
Develop (pseudocode)	Produce a section of code to provide a solution to a problem.
Discuss	Consider the different aspects in detail of an issue, situation, problem or argument, and how they interrelate.
Draw	Produce a diagram, either using a ruler or using freehand OR create a graphical or visual representation of information.
Evaluate	Consider various aspects of a subject's qualities in relation to its context, such as strengths and weaknesses, advantages and disadvantages, pros and cons. Come to a judgment supported by evidence, which will often be in the form of a conclusion.
Explain	Present one point that identifies a reason, way or importance and a second point that justifies/explains the first point. Where used, a third point is a further justification/explanation.
Give	Recall from memory a feature, characteristic or use.
Identify	Select the correct answer from the given context or stimulus.
Label	Correctly indicate parts of a diagram/image/graphical representation.
List	Recall from memory facts, dates, legal implications, etc. More than one.
State	Recall from memory a fact, date, legal implication, etc.

6. General Competency Frameworks for T Levels

English, maths and digital competencies

The General Competency Framework for T Levels articulates English, mathematical and digital competencies that students are required to develop over the course of the qualification. The tables below list the competencies from the framework that are relevant to the *T Level Technical Qualification in Digital Production, Design and Development*.

Competencies that can be developed in relation to a specification element of content are referenced in the column next to this content element. These competencies should be delivered through the content of this qualification and tutors should seek opportunities to allow students to develop the relevant skills to enable them to reach threshold competence in the specialism.

The English, Maths and Digital competencies are embedded in both the Core Component and the Occupational Specialist Components of the *T Level Technical Qualification in Digital Production, Design and Development*. This is so that students are able to demonstrate their knowledge and understanding of these skills over the course of the qualification.

General English competencies

E1	Convey technical information to different audiences
E2	Present information and ideas
E3	Create texts for different purposes and audiences
E4	Summarise information/ideas
E5	Synthesise information
E6	Take part in/leading discussions

General maths competencies

M1	Measure with precision
M2	Estimate, calculate and spot errors
M3	Work with proportion
M4	Use rules and formulae
M5	Process data
M6	Understand data and risk
M7	Interpret and represent with mathematical diagrams
M8	Communicate using mathematics
M9	Cost a project
M10	Optimise work processes

General digital competencies

Students should be supported to develop the digital knowledge and skills needed in order to:

D1	Use digital technology and media effectively
D2	Design, create and edit documents and digital media
D3	Communicate and collaborate
D4	Process and analyse numerical data
D5	Be safe and responsible online
D6	Code and program

Appendix 1: Pseudocode, commands and structure

This appendix provides additional information about the pseudocode commands and structure that will be used in the core examinations. Students should be provided with opportunities to explore and use pseudocode within the core and occupational specialism content.

This appendix does **not** replace the specification but should be used alongside the specification content to provide additional guidance and scope.

Pseudocode

Data types STRING CHARACTER INTEGER REAL

FLOAT

BOOLEAN

Type coercion

Type coercion is automatic if indicated by context. For example, 3 + 8.25 = 11.25 (integer + real = real).

Coercion can be made explicit. For example, RECEIVE age FROM (INTEGER) KEYBOARD assumes that the input from the keyboard is interpreted as an INTEGER, not a STRING.

Constants

The value of constants can only ever be set once. They are identified by the keyword CONST.

Two examples of using a constant are shown. CONST REAL PI

SET PI TO 3.14159

SET circumference TO radius * PI * 2

Data structures

ARRAY LIST

DICTIONARY

Indices start at zero (0) for all data structures.

When performing ‘slicing’ operations and other ‘sting handling’ operations, the data type STRING can be considered a data structure and should be indexed in the same way.

All data structures have an append operator, indicated by &.

Using & with a STRING and a non-STRING will coerce to STRING. For example, SEND ‘Fred’ & age.

TO DISPLAY, will display a single STRING of ‘Fred18’.

Identifiers

Identifiers are sequences of letters, digits and ‘_’, starting with a letter, for example MyValue, myValue, My_Value, Counter2.

Functions

LENGTH()

For data structures consisting of an array or string. RANDOM(n)

This generates a random number from 0 to n.

Comments

Comments are indicated by the # symbol, followed by any text. A comment can be on a line by itself or at the end of a line.

Devices

Use of KEYBOARD and DISPLAY are suitable for input and output. Additional devices may be required, but their function will be obvious from the context. For example, CARD_READER and MOTOR are two such devices.

Notes

In the pseudocode on the following pages, the < > symbols indicate where expressions or values need to be supplied. The < > symbols are not part of the pseudocode.

Variables and arrays		
Syntax	**Explanation of syntax**	**Example**
SET Variable TO <value>	Assigns a value to a variable.	SET Counter TO 0 SET MyString TO 'Hello world'
SET Variable TO <expression>	Computes the value of an expression and assigns to a variable.	SET Sum TO Score + 10 SET Size to LENGTH (Word)
SET Array [index] TO <value>	Assigns a value to an element of a one-dimensional array.	SET ArrayClass [1] TO 'Ann' SET ArrayMarks [3] TO 56
SET Array TO [<value>, ...]	Initialises a one-dimensional array with a set of values.	SET ArrayValues TO [1, 2, 3, 4, 5]
SET Array [RowIndex, ColumnIndex] TO <value>	Assigns a value to an element of a two-dimensional array.	SET ArrayClassMarks [2,4] TO 92

Note: the same methodology should be used when assigning values in all data structures.

Selection		
Syntax	**Explanation of syntax**	**Example**
IF <expression> THEN <command> END IF	If <expression> is true then command is executed.	IF Answer = 10 THEN SET Score TO Score + 1 END IF

IF <expression> THEN
<command> ELSE
<command> END IF
	If <expression> is true then first
	<command> is executed, otherwise second
	<command> is executed.
		IF Answer = 'correct' THEN SEND 'Well done' TO DISPLAY
		ELSE
		SEND 'Try again' TO DISPLAY
		END IF

Repetition		
Syntax	**Explanation of syntax**	**Example**

WHILE <condition> DO
<command> END WHILE
	Pre-conditioned loop. Executes
	<command> while
	<condition> is true.
		WHILE Flag = 0 DO
		SEND 'All well' TO DISPLAY
		END WHILE

REPEAT <command> UNTIL <expression>	Post-conditioned loop. Executes <command> until <condition> is true. The loop must execute at least once.	REPEAT SET Go TO Go + 1 UNTIL Go = 10
REPEAT <expression> TIMES <command> END REPEAT	Count controlled loop. The number of times <command> is executed is determined by the expression.	REPEAT 100-Number TIMES SEND '*' TO DISPLAY END REPEAT
FOR <id> FROM <expression> TO <expression> DO <command> END FOR	Count controlled loop. Executes <command> a fixed number of times.	FOR Index FROM 1 TO 10 DO SEND ArrayNumbers [Index] TO DISPLAY END FOR
FOR <id> FROM <expression> TO <expression> STEP <expression> DO <command> END FOR	Count controlled loop using a step.	FOR Index FROM 1 TO 500 STEP 25 DO SEND Index TO DISPLAY END FOR
FOR EACH <id> FROM <expression> DO <command> END FOREACH	Count controlled loop. Executes for each element of an array.	SET WordsArray TO ['The', 'Sky', 'is', 'grey'] SET Sentence to " FOR EACH Word FROM WordsArray DO SET Sentence TO Sentence & Word & " END FOREACH

Input/output		
Syntax	**Explanation of syntax**	**Example**
SEND <expression> TO DISPLAY	Sends output to the screen.	SEND 'Have a good day.' TO DISPLAY
RECEIVE <identifier> FROM (type) <device>	Reads input of specified type.	RECEIVE Name FROM (STRING) KEYBOARD RECEIVE LengthOfJourney FROM (INTEGER) CARD_READER RECEIVE YesNo FROM (CHARACTER) CARD_READER

File handling		
Syntax	**Explanation of syntax**	**Example**
READ <File> <record>	Reads in a record from a <file> and assigns to a <variable>. Each READ statement reads a record from the file.	READ MyFile.doc Record
WRITE <File> <record>	Writes a record to a file. Each WRITE statement writes a record to the file.	WRITE MyFile.doc Answer1, Answer2, 'xyz 01'

Subprograms		
Syntax	**Explanation of syntax**	**Example**
PROCEDURE <id> (<parameter>, ...) BEGIN PROCEDURE <command> END PROCEDURE	Defines a procedure.	PROCEDURE CalculateAverage (Mark1, Mark2, Mark3) BEGIN PROCEDURE SET Avg to (Mark1 + Mark2 + Mark3)/3 END PROCEDURE
FUNCTION <id> (<parameter>, ...) BEGIN FUNCTION <command> RETURN <expression> END FUNCTION	Defines a function.	FUNCTION AddMarks (Mark1, Mark2, Mark3) BEGIN FUNCTION SET Total to (Mark1 + Mark2 + Mark3)/3 RETURN Total END FUNCTION
<id> (<parameter>, ...)	Calls a procedure or a function.	Add (FirstMark, SecondMark)

Arithmetic operators	
Symbol	**Description**
+	Add
-	Subtract
/	Divide
*	Multiply
^	Exponent
MOD	Modulo
DIV	Integer division

Relational operators	
Symbol	**Description**
=	equal to
<>	not equal to
>	greater than
>=	greater than or equal to
<	less than
<=	less than or equal to

Logical operators	
Symbol	**Description**
AND	Returns true if both conditions are true.
OR	Returns true if any of the conditions are true.
NOT	Reverses the outcome of the expression; true becomes false, false becomes true.

Appendix 2: Flowchart symbols

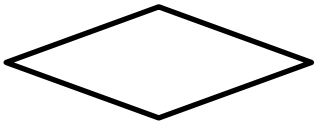
This appendix provides additional information about the flowchart symbols that will be used in the core examinations. Students are expected to respond to flowchart questions using these symbols. Students should be provided with opportunities to explore and use flowcharts within the core and occupational specialism content.

This appendix does **not** replace the specification but should be used alongside the specification content to provide additional guidance and scope.

Denotes the start and end of an algorithm

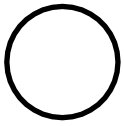
Denotes a process to be carried out

Denotes a sub-process



Denotes a decision to be made

Denotes input or output



Denotes a connection to part of a flowchart that cannot easily be linked using an unbroken flow arrow

Shows the logical flow of the program

Appendix 3: Python commands and libraries

This appendix provides additional information about the Python commands and libraries that will be used in the core examinations and Employer Set Project (ESP). Students should be provided with opportunities to explore and use the python within the core content to solve a range of programming problems.

Students may choose to explore Python in greater detail than outlined in this Appendix, as one of their chosen languages for the Occupational Specialism.

This appendix does **not** replace the specification but should be used alongside the specification content to provide additional guidance and scope.

Python commands

Handling basic input and output

- input()
- print()
- int()
- str()
- float()
- try:
- except:

Functions and variables

- def
- global
- return

Selection

- if
- else
- elif
- case (only available in Python 3.10)

Iteration

- while
- for

Built-in functions and standard library commands

- import
- from
- as
- *

Built-in functions and standard libraries

- math
- random
- datetime

Numerical

- random()
- randint()
- uniform()
- sample()
- range()
- round()
- math.trunc()
- math.floor()
- math.ceil()
- max()
- min()
- count()

String handling

- isupper()
- islower()
- upper()
- lower()
- isalpha()
- isdigit()
- split()
- len()
- format()
- join()

Using data structures (lists and arrays)

- index()
- append()
- insert()
- remove()
- count()
- pop()
- sort()
- in
- not in
- len()

Working with times and dates

- datetime.now()
- strftime()
- strptime()

Working with external text files

- open()
- write()
- close()
- read()
- readline()
- readlines()
- line.split()

Additional libraries and commands

For questions in Employer Set Project, students will be expected to have a working knowledge of these additional libraries:

pandas:

Students should be able to use the *pandas* library to perform data analysis on a given data file (.csv). They should be able to:

- import data from a provided .csv file.
- create and manipulate data frames which utilise all or part of the imported data (as required to meet given requirements)
- perform mathematical operations and statistical analysis on the contents of a data frame (or associated variable) such as:
  - identifying trends and patterns over time
  - calculating a total or average from a range of data
  - counting the number of occurrences of a specific item of data.

matplotlib.

Students should be able to use the *matplotlib* library to format and output, in conjunction with the *pandas* library data as part of analysis on a given data file (.csv). They should be able to:

- select data and output to appropriate graphs to meet the requirements of a brief
- format graph outputs to ensure they are meaningful and easy to use (e.g. axis labels, colour schemes, legends etc).

Explore Pearson's
T Levels offering at

quals.pearson.com/tlevels

Copyright in this document belongs to, and is used under licence from the Department for Education, © 2026.

'T-LEVELS' and 'T Level' are registered trademarks of the Department for Education. Pearson is authorised by the Department for Education to develop and deliver this T Level Technical Qualification.

Pearson and logo are registered trademarks of Pearson.

ELS_A1115 | V1