DRAFT/Version 1.0 May 2021

all you need to know.

NCFE Level 4 Award in Programming: Python (603/7501/2/PYT)

NCFE Level 4 Award in Programming: C# (603/7501/2/C#)

NCFE Level 4 Award in Programming: PHP (603/7501/2/PHP)

Qualification Specification

Contents

Section 1

About this qualification

Introduction

This qualification specification contains details of all the units and assessments required to complete this qualification.

To ensure that you are using the most up-to-date version of this qualification specification, please check the version number and date in the page footer against that of the qualification specification on QualHub.

If you advertise this qualification using a different or shortened name, you must ensure that learners are aware that their final certificate will state the full regulated qualification title.

Support handbook

This qualification specification must be used alongside the mandatory support handbook on the qualifications page on QualHub, which contains additional supporting information to help with the planning, delivery and assessment, such as:

- definition of total qualification time (TQT)
- quality assurance
- staffing requirements
- assessment
- qualification support
- diversity and equality

This qualification specification contains all of the qualification-specific information you will need that is not covered in the support handbook.

Qualification summary	
Qualification title	NCFE Level 4 Award in Programming
Qualification number (QN)	603/7501/2/PYT – Python pathway 603/7501/2/C# – C# pathway 603/7501/2/PHP – PHP pathway
Aim reference	(60375012)
Total qualification time (TQT)	105
Guided learning hours (GLH)	65
Minimum age	19
Qualification purpose	The purpose of the NCFE Level 4 Award in Programming is to give learners the knowledge and skills to support progression into employment in a programming role. Learners will gain the knowledge and skills by successfully completing the units required in their chosen programming language pathway: • Python • C# • PHP
Aims and objectives	This qualification aims to: • focus on the study of programming in the digital sector • offer breadth and depth of study, incorporating a key core of knowledge or programming fundamentals • provide opportunities to acquire a number of practical skills, including planning and creating a program, and debugging and testing code. The objectives of this qualification are to enable the learner to: • understand and use the fundamentals of programming, including: o how code and algorithms work o data types, statements and syntax o functions, objects, classes and libraries • understand and demonstrate how to debug and test code • understand the significance of emerging technology when working in programming • understand and use programming concepts that are specific to their chosen programming language • plan, create and review a program in their chosen programming language.
Work/industry placement experience	Work/industry placement experience is not required.

Real work environment (RWE) requirement/ recommendation	Experience in the real work environment is not required.
Rules of combination	Learners are required to complete the 2 mandatory units. See below pp.7-8 for further information on the units and pathways.
Grading	Achieved/not yet achieved
Assessment method	Internally assessed and externally quality assured portfolio of evidence
Staffing requirements	Staff delivering this qualification should have a good working knowledge of programming.
Progression	Learners who achieve this qualification could progress to: • Level 5 qualifications in computing Learners who achieve this qualification could also progress to employment in the following roles: • programmer • backend developer • web developer • app developer • data scientist • software engineer • software developer • game developer • computer systems analyst • network systems administrator
Regulation information	This is a regulated qualification. The regulated number for this qualification is 603/7501/2.
Funding	This qualification may be eligible for funding. For further guidance on funding, please contact your local funding provider.

Entry guidance

This qualification is designed for learners who want to increase their knowledge, skills and understanding of programming and progress into specialist study or employment in the sector.

Entry is at the discretion of the centre. However, learners should be aged 16 or above to undertake this qualification.

There are no specific prior skills/knowledge a learner must have for this qualification. However, learners may find it helpful if they have already achieved a level 3 qualification.

Centres are responsible for ensuring that this qualification is appropriate for the age and ability of learners. They need to make sure that learners can fulfil the requirements of the learning outcomes and comply with the relevant literacy, numeracy and health and safety aspects of this qualification.

Achieving this qualification

To be awarded the Level 4 Award in Programming: Python, learners are required to successfully achieve **2** mandatory units:

- Unit 1 – The fundamentals of programming
- Unit 2 – Developing a program in Python

To be awarded the Level 4 Award in Programming: C#, learners are required to successfully achieve **2** mandatory units:

- Unit 1 – The fundamentals of programming
- Unit 2 – Developing an app in C#

To be awarded the Level 4 Award in Programming: PHP, learners are required to successfully achieve **2** mandatory units:

- Unit 1 – The fundamentals of programming
- Unit 2 – Using PHP for web development

To achieve this qualification, learners must successfully demonstrate their achievement of all learning outcomes of the units as detailed in this qualification specification. A partial certificate may be requested for learners who do not achieve their full qualification but have achieved at least one whole unit.

Pathways within this qualification

When registering learners, centres should use the qualification number (QN) followed by the relevant pathway code:

- Python (PYT)
- C# (C#)
- PHP (PHP)

As the chosen pathway will appear on the certificate, it is important that tutors make clear to learners the specific option they will be registered against. Centres must carefully consider which option they want to register the learner onto, as the registration fee will be applied to each option.

Units

The units within this qualification cross over into the various pathways available and therefore do not follow the standard unit numbering of Unit 01, Unit 02, for example.

The regulated unit number is indicated in brackets for each unit (for example, M/100/7116) within section 2.

Knowledge only units are indicated by a lightbulb. If a unit is not marked with a lightbulb, it is a skills unit or contains a mix of knowledge and skills.

Mandatory units

Unit number	Regulated unit number	Unit title	Level	GLH	Notes
Unit 01	H/618/7116	The fundamentals of programming	4	30	Mandatory for all pathways
Unit 02	K/618/7117	Developing a program in Python	4	35	Mandatory for the Python pathway
Unit 03	M/618/7118	Developing an app in C#	4	35	Mandatory for the C# pathway
Unit 04	T/618/7119	Using PHP for web development	4	35	Mandatory for the PHP pathway

The units above may be available as stand-alone unit programmes. Please visit our website for further information.

How the qualification is assessed

Assessment is the process of measuring a learner's skill, knowledge and understanding against the standards set in a qualification.

This qualification is internally assessed and externally quality assured.

The assessment consists of **one** component:

- an internally assessed portfolio of evidence which is assessed by centre staff and externally quality assured by NCFE (IQA must still be completed by the centre as per usual)

Learners must be successful in this component to gain the Level 4 Award in Programming.

Learners who aren't successful can resubmit work within the registration period; however, a charge may apply.

All the evidence generated by the learner will be assessed against the standards expected of a level 4 learner for each learning outcome.

Internal assessment

We have created some sample tasks for the 2 internally assessed units, which can be found on Qualhub. You can contextualise these tasks to suit the needs of your learners to help them build up their portfolio of evidence. The tasks have been designed to cover some knowledge, skills and learning outcomes for the 2 units and provide opportunities for stretch and challenge. For further information about contextualising the tasks, please contact the curriculum team.

Each learner must create a portfolio of evidence generated from appropriate assessment tasks, which demonstrates achievement of all the learning outcomes associated with each unit. The assessment tasks should allow the learner to respond to a real life situation that they may face when in employment. On completion of each unit, learners must declare that the work produced is their own and the assessor must countersign this. Examples of suitable evidence for the portfolio for each unit are provided in section 2.

Internally assessed work should be completed by the learner in accordance with the qualification specification. A representative number of assessment hours should be timetabled into the scheme of work. Internal assessment hours must be administered outside of scheduled teaching and learning hours and should be supervised and assessed by the tutor. Assessment activities can be integrated throughout, although separate from the teaching of the unit, and do not have to take place directly at the end of the unit.

Any work submitted for internal assessment must be completed during scheduled assessment hours in accordance with the scheme of work, and must be authenticated and attributable to the learner. The tutor must be satisfied that the work produced is the learner's own.

In practice, this means that all of the portfolio of evidence will be completed in normal class time within scheduled assessment hours and kept separate from any teaching and learning hours.

A centre may choose to create their own internal assessment tasks. The tasks should:

- be accessible and lead to objective assessment judgements
- permit and encourage authentic activities where the learner's own work can be clearly judged
- refer to course file documents on QualHub

Supervision of learners and your role as an assessor

Guidance on how to administer the internal assessment and the support you provide to learners can be found on QualHub.

Section 2

Unit content and assessment guidance

Unit content and assessment guidance

This section provides details of the structure and content of this qualification.

The types of evidence listed are for guidance purposes only. Within learners' portfolios, other types of evidence are acceptable if all learning outcomes are covered and if the evidence generated can be internally and externally quality assured. For approval of methods of internal assessment other than portfolio building, please contact our quality assurance team.

The explanation of terms explains how the terms used in the unit content are applied to this qualification. This document can be found in section 3.

For further information or guidance about this qualification, please contact our customer support team.

Unit 01 The fundamentals of programming (H/618/7116)

Unit summary	**In this unit learners will understand fundamental programming concepts, This includes how code runs, understanding data types, statements and syntax, understanding functions, objects, classes and libraries, understanding the process of testing and debugging code, and understanding emerging technology in programming.**

Guided learning hours	30
Level	4
Mandatory/optional	Mandatory

Learning outcome 1	
The learner will:	
1	**Understand how code runs**
The learner can:	
1.1	**Analyse the relationship between algorithms and code**
1.2	**Explain how to design an algorithm using computational thinking and demonstrate the use of pseudocode**
1.3	**Compare and contrast programming paradigms:** • **procedural** • **object-orientated** • **event-driven programming**
1.4	**Explain and research IDEs**

Learning outcome 2	
The learner will:	
2	**Understand data types, statements and syntax**
The learner can:	
2.1	**Identify data types and demonstrate their use**
2.2	**Explain and demonstrate the application of complex data types**
2.3	**Explain the functions of variables and constants and demonstrate their use**
2.4	**Identify operators and demonstrate their use**
2.5	**Explain and demonstrate control structures and expressions**
2.6	**Explain the role of syntax**

Learning outcome 3	
The learner will:	
3	**Understand functions, objects, classes and libraries**
The learner can:	
3.1	**Explain and demonstrate the application of functions and methods**

3.2	**Explain objects and classes and demonstrate their use**
3.3	**Explain the application of third-party libraries**

Learning outcome 4	
The learner will:	
4	**Understand debugging and testing**
The learner can:	
4.1	**Explain ways to avoid bugs and errors**
4.2	**Identify and resolve syntax errors in code**
4.3	**Identify and resolve logical errors in code**
4.4	**Resolve compile/run time errors**
4.5	**Explain how to test a program**

Learning outcome 5	
The learner will:	
5	**Understand emerging technology in programming**
The learner can:	
5.1	**Discuss the role and the significance of emerging technology in programming**

Assessment guidance

Delivery and assessment

LO1 – Understand how code runs

1.1 Analyse the relationship between algorithms and code

The learner will explain the purpose and characteristics of algorithms and code and explain the relationship between them. The learner will also explain the functions of different algorithm types and code components where applicable to their chosen programming language:

- algorithms:
  - purpose
  - characteristics, for example input, output
  - types:
    - greedy
    - brute force
    - recursive
    - divide and conquer
    - dynamic programming
    - backtracking
    - sorting, for example merge sort/insertion sort/quick sort
- code:
  - purpose
  - characteristics, for example safety, security, reliability
  - components:
    - pre-processor

        - compiler
        - linker
        - interpreter
- relationship between algorithms and code, for example how algorithms are translated into code.

1.2 Explain how to design an algorithm using computational thinking and demonstrate the use of pseudocode

The learner will explain how algorithms are used to solve problems using the stages of computational thinking where applicable to their chosen programming language:

- computational thinking:
    - decomposition:
        - interpret specification/brief, for example, what problem the algorithm will solve
        - inputs
        - outputs
    - pattern recognition:
        - effectiveness
    - abstraction:
        - definiteness
        - finiteness
    - action/algorithm design:
        - pseudocode
        - flowchart

The learner will also demonstrate how to write pseudocode for a simple problem, for example:

- creating a quiz, allowing users to input answers
- the use of IF and ELSE.

1.3 Compare and contrast programming paradigms:

The learner will understand the features of programming paradigms, covering the following areas where applicable to their chosen programming language:

- procedural:
    - procedures
    - function-based
    - systematic execution of instructions
- object-orientated:
    - inheritance
    - encapsulation
    - polymorphism/overriding
    - abstraction/interfaces
    - singleton classes
- event-driven:
    - user events:
        - mouse clicks
        - key presses
    - system events:
        - lost internet connection
        - completed download
    - event handlers
    - the event loop
    - sleeps
    - callbacks

The learner will compare and contrast procedural, object-orientated and event-driven programming, where applicable to their chosen programming language, including:

- the advantages and disadvantages of each paradigm
- the best application of each paradigm
- programming paradigm characteristics:
    - predefined functions
    - local variable
    - global variable
    - modularity
    - parameter passing.

1.4 Explain and research IDEs

The learner will explain the role of an IDE and compare different types of IDEs, where applicable to their chosen programming language, for example:

- Visual Studio
- PyCharm
- IDLE

The learner could research into which IDEs are used by different companies and why, for example:

- social media platforms
- online marketplaces.

LO2 - Understand data types, statements and syntax

2.1 Identify data types and demonstrate their use

The learner will identify and apply different data types and explain when each data type would be used in programming, including:

- integer
- floating point/float
- character
- Boolean
- string.

The learner will also demonstrate a data type in use.

2.2 Explain and demonstrate the application of complex data types

The learner will explain the application of complex data types and demonstrate them in use, covering the following areas:

- lists and arrays, including how elements are referenced using indices
- dictionaries and maps, including pairs of ‘keys’ to ‘values’
- sets
- multi-dimensional arrays
- the use of operators when working with complex data types, for example loops.

2.3 Explain the functions of variables and constants and demonstrate their use

The learner will explain what variables and constants are and be able to write an example of a variable and constant.

The learner will also demonstrate the following steps:

- declare and assign a value to a variable and a constant

- display the result using appropriate syntax.

2.4 Identify operators and demonstrate their use

The learner will be able to identify and demonstrate how to use mathematical operators

The learner will interpret and evaluate logical expressions using Boolean operators.

The learner will demonstrate the use of basic operators on different data types, for example:
- basic maths operations on numerical values
- adding strings together.

The learner will explain how some programming languages will 'convert' data types and others will cause errors when completing certain actions, such as adding a number to a string.

2.5 Explain and demonstrate control structures and expressions

The learner will explain the role and function of sequence in programming.

The learner will explain the role of selection in programming and the function of the following statements:
- IF
- ELSE

The learner will explain the role of iteration and the function of the following concepts:
  loops:
  - while
  - for
  - stop
- conditions
- counters

The learner will also explain the purpose and function of control flow.

The learner will explain how expressions can be used, and demonstrate expressions in use in sequence, selection and iteration, taking into consideration the following:
- checking the value of a variable, for example TRUE or FALSE
- the use of logical operators, for example AND, OR, NOT.

2.5 Explain the role of syntax

The learner will explain the role of syntax in programming, covering the following areas:
- the definition of syntax
- how to execute syntax
- indentation
- comments
- quotation
- waiting for user input
- case sensitive.

LO3 - Understand functions, objects, classes and libraries

3.1 Explain and demonstrate the application of functions and methods

The learner will explain how functions can be applied, covering the following areas:

- the use of functions to avoid code repetition and simplify readability
- defining and calling functions
- arguments/parameters and return values
- recursive functions and exit conditions.

The learner will then explain the application of methods, including how they differ from functions.

The learner will also create a function and then re-create that function as a method within a class, for example formatting a date.

3.2 Explain objects and classes and demonstrate their use

The learner will explain the applications of objects and classes, the relationship between them, and be able demonstrate them in use, covering the following areas:

- constructors
- member variables
- static variables
- public versus private scope.

3.3 Explain the application of third-party libraries

The learner will explain the application of third-party libraries and where to find them, taking into consideration the following:

- third-party libraries for specific languages such as Python/C#/PHP
- licensing requirements for using third-party libraries:
    - intellectual property
    - open source.

LO4 - Understand debugging and testing

4.1 Explain ways to avoid bugs and errors

The learner will explain the ways in which bugs and errors in code can be avoided, where applicable to their chosen programming language, including:

- the role that IDEs play in detecting and fixing errors
- examples of coding standards, for example no more than 1 statement on a line of code
- good coding practice, for example fixing errors as they appear while coding
- using correct data structures.

4.2 Identify and resolve syntax errors in code

The learner will explain syntax errors, and identify and resolve the following where applicable to their chosen programming language:

- missing comma
- missing quotation mark
- missing brackets
- missing semicolon
- misuse of keywords
- undeclared variable
- misspelled variable
- missing operators
- misspelled instruction

- functions not called
- functions with no return value
- wrong number of arguments/wrong values passed to functions

4.3 Identify and resolve logical errors

The learner will explain logical errors and identify and resolve errors, demonstrating the use of error-resolving methods where applicable to their chosen programming language:

- errors:
  - using the incorrect operator, for example OR instead of AND
  - using the incorrect selection statement
  - using the incorrect iteration
  - incorrect Boolean operators
  - index out-of-bounds errors
  - wrong data types passed to functions
- methods:
  - debugging tools, for example within an IDE
  - break points
  - logging

4.4 Resolve compile/run time errors

The learner will explain the relationship between syntax errors and compile/run time errors, and resolve a compile/runtime error in the context of a specific programming language.

4.5 Explain how to test a program

The learner will compare testing methods, explain testing stages and identify the most appropriate testing method for a specific program.

- testing methods:
  - manual test, for example build test
  - automated test
  - integration test
  - unit test
  - data test
  - assertion test
  - regression test
- testing stages:
  - plan a test
  - implement a test
  - compare expected result with actual output
    - identify errors.

LO5 - Discuss the role and the significance of emerging technology in programming

5.1 Discuss the emerging technology of programming

The learner will provide a detailed account of emerging technologies in programming, including contrasting perspectives, for example:

- threading
- quantum programming
- machine learning

Types of evidence
Evidence could include: • research/reports • written answers • oral answers • discussion • assignment • presentation • observation • screenshots, for example of sample code • digital notebook such as Jupyter, OneNote.

Unit 02 Developing a program in Python (K/618/7117)

Unit summary	In this unit learners will understand the factors involved in program development in Python and apply these with the fundamentals of programming to plan, develop and review a program using Python.
Guided learning hours	35
Level	4
Mandatory/optional	Mandatory for the Python pathway

Learning outcome 1	
The learner will:	
1	**Understand the factors involved in data program development in Python**
The learner can:	
1.1	**Critically compare Python version numbers**
1.2	**Analyse and explain the Python shell and the IDLE**
1.3	**Explain the Python directory and management of files**
1.4	**Explain Python modules and demonstrate their use**
1.5	**Explain and demonstrate list comprehension and slicing**
1.6	**Explain the function of generators and demonstrate how to create them**
1.7	**Explain and demonstrate data handling in relation to Python**

Learning outcome 2	
The learner will:	
2	**Plan a data program using Python**
The learner can:	
2.1	**Interpret a brief to create a project plan**
2.2	**Plan the code for a program**
2.3	**Create a testing plan for a program**

Learning outcome 3	
The learner will:	
3	**Create and test a data program using Python**
The learner can:	
3.1	**Apply Python programming fundamentals to create a product**
3.2	**Implement a testing plan**

Learning outcome 4	
The learner will:	
4	**Review the data program**
The learner can:	
4.1	**Review the appropriateness of the project plan**
4.2	**Critically analyse the program, including the use of Python features**
4.3	**Evaluate their application of the debugging process**
4.4	**Gather user feedback and implement changes**

Assessment Guidance

Delivery and assessment

LO1 Understand the factors involved in a data program development in Python

1.1 Critically compare Python version numbers

The learner will critically compare python version numbers, such as Python 3.6, 3.7 and 3.8, and research their differences to discuss the following concepts:

- evolution
- deprecation
- sector advancement
- how Python interacts with a computing environment, such as automating tasks.

The learner will also identify the most appropriate Python version for a particular task and justify the reasons behind their choice, including the advantages and disadvantages of multiple version numbers.

1.2 Analyse and explain the Python shell and the IDLE

The learner will analyse and explain the function of the Python shell and the IDLE, including:

- the differences between the shell and the IDLE
- the relationship between the shell and the IDLE.

1.3 Explain the Python directory and management of files

The learner will explain the Python directories and file management within Python, for example:

- copying files
- moving files
- deleting files.

1.4 Explain Python modules and demonstrate their use

The learner will explain the functions of python modules and submodules, and how to use the import statement within Python. Python module and submodule examples include:

- module:
  - ‘os’
- submodules:
  - ‘sys’
  - ‘path’

The learner will demonstrate how to write a module using the shell and IDLE, for example:
- the greeting module

1.5 Explain and demonstrate list comprehension and slicing

The learner will explain the function of both list comprehension and slicing in Python, including:
- definitions
- usage (when to apply)
- components of list comprehension, for example output expression, input sequence
- parameter values of slicing, for example start, end/stop, step

The learner will also apply this knowledge to demonstrate an example of slicing, for example:
- slicing an object
- slicing a sequence
- slice deletion
- slicing a list
- modifying lists via slices
- multiple slicing
- slicing strings.

1.6 Explain the function of generators and demonstrate how to create them

The learner will explain what generators are, how to use them and be able to create generator functions, including:
- returning an object
- using generators in a loop
- using generators to read large files.

1.7 Explain and demonstrate data handling in relation to Python

The learner will analyse and explain Python data handling, including:
- how Python reads data, for example reading each line of code
- how Python interprets data, for example performs the task written in the line of code
- how Python processes data, for example what is the output of the code

The learner will also explain and demonstrate how to load different types of data using Python, for example from:
- text files
- CSV files
- JSON files.

LO2 Plan a data program using Python

2.1 Interpret a brief to create a project plan

The learner will interpret a brief, and explain the functional requirements, for example:
- which algorithms will be required
- security requirements

2.2 Plan the code for a program

The learner will break down the functional requirements of the brief into detailed areas in order to write the code, including:

- planning what data types to use
- identifying the appropriate paradigm
- planning best practice according to the selected paradigm
- writing the pseudocode
- designing a flowchart, including:
  - inputs
  - outputs
- writing the algorithm.

2.3 Create a test plan

The learner will create a test plan for their program, incorporating the following elements:

- identifying an appropriate testing method
- selecting a test runner
- areas to test:
  - if expected inputs can be accepted
  - if data is processed appropriately
  - if generated outputs are accurate as expected
  - ensuring all functional requirements are delivered.

LO3 Create and test a data program using Python

3.1 Apply Python programming fundamentals to create a program

The learner will demonstrate the application of Python programming fundamentals in creating a program, including the following steps where appropriate:

- declare and or use the appropriate data types
- declare and assign a value to a variable/constant
- identify appropriate operators
- use expressions within sequence, selection and iteration
- apply a method/function
- use appropriate data structures, for example:
  - lists (such as ordered, changeable):
    - list comprehension
  - tuples
  - Python dictionaries
  - Python sets
- use objects and classes
- integrate a third-party library, for example importing a module from a Python library
- proper use of Python syntax, for example correct use of whitespace
- run the code.

3.2 Implement the testing plan

The learner will implement the elements of the testing plan previously devised.

The learner will then be able to utilise Python debugging concepts and IDE-specific debugging tools (for example, inserting break points) to identify and resolve programmatic errors (where appropriate), for example:

- errors
  - run time
  - syntax

- import
- indentation.

LO4 Review the data program

4.1 Critically analyse the project plan and testing plan

The learner will critically analyse both the project plan and the testing plan and make recommendations for improvements, taking into account the following aspects:

- if the plans achieved the desired outcomes and if not, why not
- if any aspects were overlooked, not planned for or not taken into account and what the repercussions were
- strengths of planning: what went well
- weaknesses of planning: what did not go well
- future recommendations for the project and test plans: what could be done differently and what could be improved

4.2 Critically analyse the finished program

The learner will critically analyse the final program and make recommendations for improvements taking into account the following aspects:

- to what extent the program met the brief
- their utilisation of Python functions, for example lists, objects
- strengths of the program, for example user functionality, easily understandable
- weaknesses of the program, for example errors in code, missing elements of code
- new skills acquired, for example error resolution, use of new functions
- future recommendations, for example what could be done differently when creating a program in future, how the program could be improved

4.3 Gather and analyse user feedback

The learner will gather user feedback on the suitability of the program and analyse the feedback to re-evaluate the product:

- data collection methods for gathering user feedback:
  - questionnaire
  - survey
- areas to gather feedback on:
  - functionality
  - usability
  - what the user likes
  - what the user dislikes
  - what could be improved
- analysis of feedback:
  - what the feedback says about the program's strengths
  - what the feedback says about the program's weaknesses
  - using feedback to inform what improvements will be made

4.4 Implement changes based on reviewing process

The learner will implement changes to their program based on both their own critical analysis and the user feedback.

The learner will also explain the reasons for implementing some aspects of feedback received and not others, based on the following:

- priorities, for example is the change plausible in terms of timescale
- validity of feedback, for example does the feedback support relevant and achievable improvements to the program.
- programming technical abilities, for example is it technically achievable.

The explanation of terms (section 3) explains how the terms used in the unit content are applied to this qualification.

Types of evidence

Evidence could include:

- research/reports
- written answers
- oral answers
- discussions
- observations
- assignments
- presentations
- screenshots eg of sample code
- project diary/commentary
- digital notebook such as Jupyter, OneNote.

Unit 03 Developing an app in C# (M/618/7118)

Unit summary	In this unit learners will understand the factors involved in application development in C# and apply these with the fundamentals of programming to plan, develop and review an app using C#.
Guided learning hours	35
Level	4
Mandatory/optional	Mandatory for the C# pathway

Learning outcome 1	
The learner will:	
1	**Understand C# specifics as used in application development (mobile and desktop)**
The learner can:	
1.1	**Identify and explain different application user environments**
1.2	**Understand and mitigate issues with connectivity**
1.3	**Explain application lifecycles and user interface**
1.4	**Explain the use of models, views and viewmodels**
1.5	**Explain the concept of dependency injection**
1.6	**Explain Async/await and the UI thread**

Learning outcome 2	
The learner will:	
2	**Plan a project using C#**
The learner can:	
2.1	**Interpret a brief**
2.2	**Plan the code**
2.3	**Design the application**
2.4	**Create a test plan**

Learning outcome 3	
The learner will:	
3	**Create an application using C#**
The learner can:	

3.1	**Apply the elements of the project plan**
3.2	**Implement the testing plan**

Learning outcome 4	
The learner will:	
4	**Review the plans and application**
The learner can:	
4.1	**Review the appropriateness of the project plan and testing plan**
4.2	**Critically analyse the application and explain its strengths and weaknesses**
4.3	**Gather and analyse user feedback**
4.4	**Implement changes based on reviewing process**

Delivery and assessment LO1 Understand C# specifics as used in application development (mobile and desktop)

1.1 Identify and explain different application user environments

The learner must identify and explain the different environments in which the application may need to be used and be aware of the constraints in functionality that exist:

- mobile application:
  - range of devices:
    - Android
    - iPhone
  - screen size:
    - smartphone
    - tablet
  - input:
    - touchscreen
  - variable network
- desktop application:
  - controlled PC
  - input:
    - mouse/trackpad
  - database server
  - wired network.

1.2 Understand and mitigate issues with connectivity

The learner must understand and consider mechanisms to handle and communicate connectivity issues:

- connectivity:
  - reliable
  - unreliable
  - online versus offline mode

- mechanisms:
  - working offline
  - communicating connectivity status
  - restrict functionality.

1.3 Explain application lifecycles and user interface

The learner must explain platform-specific constraints and explain the considerations involved in application design depending on the choice of platform and application structure:

- platform constraints
- lifecycle events
- navigation.

1.4 Explain the use of models, views and viewmodels

The learner must explain the benefits and responsibilities of a model, view, viewmodel pattern (MVVM), for example:

- benefits, such allowing for separate work by developer and designer
- responsibilities, such as the role of the view, the model and the viewmodel.

1.5 Explain the concept of dependency injection

The learner must explain how dependency injection signposts all the dependencies in the application within a single container and demonstrate how this fits within a model, view, viewmodel pattern, including:

- how dependency injection centralises the dependencies between objects:
  - shared database connection
- code reuse
- simplification of testing
- substitution of dependencies, for example data sources, testing data.

1.6 Explain Async/await and the UI thread

The learner must explain the concept of threads, in particular the UI thread, and how to delegate tasks to prevent blocking.

The learner must provide examples of common asynchronous tasks and the UI callbacks that may be implemented.

LO2 Plan a project

2.1 Interpreting the brief

The learner must identify the functional requirements and technical challenges of the application as part of interpreting a brief:

- application goals:
  - data outputs/calculations
  - functional achievements
- application inputs
- connectivity challenges, such as poor or no internet connection
- user input requirements, such as keyboard, mouse, touchscreen, voice.

2.2 Code planning

The learner must break down the functional requirements of the brief into detailed areas in order to write the code, including:

- platform-specific components:
  - user interface design
  - user input
  - connectivity
- functional areas of the system
- inputs and outputs.

2.3 Design the application

The learner must understand the purpose of the following functional areas and use a selection of these areas in appropriate detail where relevant:

- models
- views
- viewmodels
- dependencies.

2.4 Create a test plan

The learner must create a test plan for the application, incorporating the following elements:

- identify an appropriate testing method
- areas to test:
  - test to see if expected inputs can be accepted
  - test to see if data is processed appropriately
  - test to see if generated outputs are accurate as expected
  - test to ensure all functional requirements are delivered.

LO3 Create an application using C#

3.1 Apply C# programming fundamentals to create an application

The learner must demonstrate the application of C# programming fundamentals in creating an application, including the following steps where appropriate:

- create the platform-specific user interface using the appropriate tools
- create the appropriate models for data objects used within the application
- construct an appropriate model, view, viewmodel pattern using dependency injection principles:
  - data types
  - validation
- utilise async/await methods where appropriate
- consider data storage solutions appropriate to any connectivity issues

3.2 Implement the testing plan

The learner can implement the elements of the testing plan previously devised.

The learner should then utilise C# debugging concepts and IDE-specific debugging tools (such as inserting break points) to identify and resolve programmatic errors (where appropriate), for example:

- errors
  - run time
  - syntax
  - import.

LO4 Review the project plan and application

4.1 Critically analyse the project plan and testing plan

The learner can critically analyse both the project plan and the testing plan and make recommendations for improvements, taking into account the following aspects:

- if the plans achieved the desired outcomes, and if not, why not
- if any aspects were overlooked, not planned for or not taken into account and what the repercussions were
- strengths of planning: what went well
- weaknesses of planning: what did not go well
- future recommendations for the project and test plans: what could be done differently, what could be improved.

4.2 Critically analyse the application and explain its strengths and weaknesses

The learner can critically analyse the application and make recommendations for improvements taking into account the following aspects:

- to what extent the application met the brief
- their utilisation of C# functions
- strengths of the program, for example the database provider can easily be substituted for another
- weaknesses of the program, for example connectivity still a problem
- new skills acquired, for example error resolution, use of new functions
- future recommendations, for example what could be done do differently when creating a program in future, how the program could be improved.

4.3 Gather and analyse user feedback

The learner can gather feedback on the suitability of the application and analyse the feedback to re-evaluate the product:

- data collection methods for gathering user feedback:
  - questionnaire
  - survey
  - in-app
- areas to gather feedback on:
  - functionality
  - usability
  - what the user likes
  - what the user dislikes
  - what could be improved
- analysis of feedback:
  - what the feedback says about the program's strengths
  - what the feedback says about the program's weaknesses
  - using feedback to inform what improvements will be made.

4.4 Implement changes based on reviewing process

The learner can implement changes to their application based on both their own critical analysis and the user feedback.

The learner can also explain the reasons for implementing some aspects of feedback received and not others, based on the following:

- priorities, for example is the change plausible in terms of timescale, system/software requirements
- validity of feedback, for example does the feedback support relevant and achievable improvements to the program

- programming technical abilities, for example is it technically achievable.

The explanation of terms (section 3) explains how the terms used in the unit content are applied to this qualification.

Types of evidence

Evidence could include:

- research/reports
- written answers
- oral answers
- discussions
- assignments
- presentations
- observations
- screenshots, for example of sample code
- project diary/commentary
- digital notebook such as OneNote.

Unit 04 Using PHP for web development (T/618/7119)

Unit summary	In this unit learners will understand the factors involved in web development in PHP and apply these with the fundamentals of programming to plan, develop and review a product using PHP.
Guided learning hours	35
Level	4
Mandatory/optional	Mandatory for the PHP pathway

Learning outcome 1	
The learner will:	
1	**Understand the factors involved in web development using PHP**
The learner can:	
1.1	**Explain the purpose and function of forms**
1.2	**Explain form validation**
1.3	**Explain and demonstrate sessions**
1.4	**Explain and demonstrate databases**

Learning outcome 2	
The learner will:	
2	**Plan a project using PHP**
The learner can:	
2.1	**Interpret a brief**
2.2	**Plan the code**
2.3	**Design the solution**
2.4	**Create a test plan**

Learning outcome 3	
The learner will:	
3	**Create a solution using PHP**
The learner can:	
3.1	**Apply the elements of the project plan**
3.3	**Implement the testing plan**

Learning outcome 4	
The learner will:	
4	**Review the plans and solution**
The learner can:	
4.1	**Review the appropriateness of the project plan and testing plan**
4.2	**Critically analyse the solution and explain its strengths and weaknesses**
4.3	**Gather and analyse user feedback**
4.4	**Implement changes based on reviewing process**

Delivery and assessment

LO1 Understand the factors involved in web development using PHP

1.1 Explain the purpose and function of forms

The learner must explain the purpose and functions of forms including:

- difference between 'get' and 'post' requests
- security considerations.

1.2 Explain form validation

The learner must explain reasons for data validation and provide examples of how to achieve it, for example:

- consistent data input, such as drop-down lists
- prevent SQL injection, such as data input rules.

1.3 Explain and demonstrate sessions

The learner must explain the requirement for sessions and give some examples of their use.

The learner must also be able to demonstrate how to implement the following functionality using PHP:

- start a session
- manipulate session variables
- destroy/end session.

1.4 Explain and demonstrate databases

The learner must explain the use of server-side databases to store and retrieve relational data utilising one of the following:

- MySQL
- PostgreSQL
- SQLite

The learner must demonstrate how to use database technologies using PHP to complete the following steps:

- connect
- create

- read
- update
- delete.

LO2 Plan a project using PHP

2.1 Interpreting the brief for PHP

The learner must identify the functional requirements of the system as part of interpreting a brief:

- system goals:
  - reporting
  - data outputs/calculations
  - functional achievements
- system inputs
- security:
  - identify user restrictions and permissions.

2.2 Code planning

The learner must break down the functional requirements of the brief into detailed areas in order to plan the code:

- inputs and outputs
- data use
- functional areas of the system
- particular areas of need as stated in the brief.

2.3 Design the solution

The learner should understand the purpose of following functional areas and be able to use a selection of these areas in appropriate detail where relevant:

- form design
- database design
- data types
- validation
- data processing.

2.4 Create a test plan

The learner will create a test plan for the solution, incorporating the following elements:

- identify an appropriate testing method
- areas to test:
  - test to see if expected inputs can be accepted
  - test to see if data is processed appropriately
  - test to see if generated outputs are accurate as expected
  - test to ensure all functional requirements are delivered.

LO3 Create a Solution using PHP

3.1 Apply PHP programming fundamentals to create the solution

The learner can apply the appropriate/ relevant programming fundamentals to create the solution, including:

- build the form using appropriate data types and input validation
- process the data

- design and store the data within an appropriate database structure
- generate the output.

3.2 Implement the testing plan

The learner can implement the elements of the testing plan previously devised.

The learner should then utilise PHP debugging concepts and IDE-specific debugging tools, such as inserting break points, server logs, browser development tools, to identify and resolve programmatic errors where appropriate, for example:

- errors:
  - run time
  - syntax
  - import.

LO4 Review the plans and solution

4.1 Critically analyse the project plan and testing plan

The learner can critically analyse both the project plan and the testing plan and make recommendations for improvements, taking into account the following aspects:

- if the plans achieved the desired outcomes, and if not, why not
- if any aspects were overlooked, not planned for or not taken into account and what the repercussions were
- strengths of planning: what went well
- weaknesses of planning: what did not go well
- future recommendations for the project and test plans: what could be done differently, what could be improved.

4.2 Critically analyse the solution and make recommendations for improvements

The learner can critically analyse the solution and make recommendations for improvements taking into account the following aspects:

- to what extent the solution met the brief
- their utilisation of PHP functions, for example objects, arrays, built-in methods
- strengths of the program, for example user functionality, easily understandable
- weaknesses of the program, for example poor performance, form requirements not obvious
- new skills acquired, for example error resolution, use of new functions
- future recommendations, for example what could be done differently when creating a program in future, how the program could be improved.

4.3 Gather and analyse user feedback

The learner can gather feedback on the suitability of the solution and analyse the feedback to re-evaluate the product:

- data collection methods for gathering user feedback:
  - questionnaire
  - survey
  - widget
- areas to gather feedback on:
  - functionality
  - usability
  - what the user likes
  - what the user dislikes

- what could be improved
- analysis of feedback:
  - what the feedback says about the program's strengths
  - what the feedback says about the program's weaknesses
  - using feedback to inform what improvements will be made.

4.4 Implement changes based on reviewing process

The learner must be able to implement changes to their solution based on both their own critical analysis and the user feedback.

The learner can also explain the reasons for implementing some aspects of feedback received and not others, based on the following:

- priorities, for example is the change plausible in terms of timescale, system/software requirements
- validity of feedback, for example does the feedback support relevant and achievable improvements to the program.
- programming technical abilities, for example is it technically achievable.

The explanation of terms (section 3) explains how the terms used in the unit content are applied to this qualification.

Types of evidence

Evidence could include:

- research/reports
- written answers
- oral answers
- discussions
- observations
- assignments
- presentations
- screenshots, for example of sample code
- project diary/commentary
- digital notebook such as OneNote.

Section 3

Explanation of terms

Explanation of terms

This table explains how the terms used at Level 4 in the unit content are applied to this qualification (not all verbs are used in this qualification).

Analyse	Break the subject or complex situations into separate parts and examine each part in detail. Identify the main issues and show how the main ideas are related to practice and why they are important. Reference to current research or theory may support the analysis.
Critically analyse	This is a development of 'analyse' which explores limitations as well as positive aspects of the main ideas in order to form a reasoned opinion.
Clarify	Explain the information in a clear, concise way showing depth of understanding.
Classify	Organise accurately according to specific criteria.
Collate	Collect and present information arranged in sequence or logical order which is suitable for purpose.
Compare	Examine the subjects in detail, consider and contrast similarities and differences.
Critically compare	This is a development of compare where the learner considers and contrasts the positive aspects and limitations of the subject.
Consider	Think carefully and write about a problem, action or decision showing how views and opinions have been developed.
Demonstrate	Show an in-depth understanding by describing, explaining or illustrating using examples.
Describe	Provide a broad range of detailed information about the subject or item in a logical way.
Discuss	Write a detailed account which includes contrasting perspectives.
Draw conclusions (which….)	Make a final decision or judgment based on reasons.
Evaluate	Examine strengths and weaknesses, arguments for and against and/or similarities and differences. Judge the evidence from the different perspectives and make a valid conclusion or reasoned judgment. Apply current research or theories to support the evaluation.
Critically evaluate	This is a development of 'evaluate' where the debates the validity of claims from the opposing views and produces a convincing argument to support the conclusion or judgement.

Explain	Apply reasoning to account for how something is or to show understanding of underpinning concepts. Responses could include examples to support these reasons.
Identify	Apply an in-depth knowledge to give the main points accurately. (A description may also be necessary to gain higher marks when using compensatory marking).
Justify	Give a detailed explanation of the reasons for actions or decisions.
Review and revise	Look back over the subject and make corrections or changes based on additional knowledge or experience.
Reflect	Learners should consider their actions, experiences or learning and the implications of these in order to suggest significant developments for practice and professional development.
Summarise	Give the main ideas or facts in a concise way to develop key issues.

Section 4

Additional information

Additional information

Resource requirements

There are no mandatory resource requirements for this qualification, but centres must ensure learners have access to suitable resources to enable them to cover all the appropriate learning outcomes.

Support for learners

Learner's evidence tracking log (LETL)

The LETL covers the mandatory units in this qualification and it can help learners keep track of their work. This document can be downloaded free of charge from the qualifications page on QualHub. You do not have to use the LETL, you can devise your own evidence tracking document instead.

Support for centres

Qualification factsheet

This document outlines the key information of this qualification for the centre, learner and employer.

Useful websites

Centres may find the following websites helpful for information, materials and resources to assist with the delivery of this qualification:

- https://www.w3schools.com/python/
- https://www.w3schools.com/cs/
- https://www.w3schools.com/php/
- https://realpython.com/
- https://docs.microsoft.com/en-us/dotnet/csharp/
- https://www.php.net/manual/en/

These links are provided as sources of potentially useful information for delivery/learning of this subject area. NCFE do not explicitly endorse any learning resources available on these websites. For official NCFE endorsed learning resources please see the additional and teaching materials sections on the qualification page on QualHub.

Learning resources

We offer a wide range of learning resources and materials to support the delivery of our qualifications. Please check the qualifications page on QualHub for more information and to see what is available for this qualification.

Contact us

NCFE
Q6
Quorum Park
Benton Lane
Newcastle upon Tyne
NE12 8BT

Tel: 0191 239 8000*
Fax: 0191 239 8001
Email: customersupport@ncfe.org.uk
Websites: www.qualhub.co.uk
www.ncfe.org.uk

DRAFT/Version 1.0 May 2021
Information in this qualification specification is correct at the time of publishing but may be subject to change.

To continue to improve our levels of customer service, telephone calls may be recorded for training and quality purposes.