Next Generation Higher National Unit Specification

Programming for Data (SCQF level 8)

Unit code: J7EG 48

SCQF level: 8 (8 SCQF credit points)

Valid from: session 2023–2024

Prototype unit specification for use in pilot delivery only (version 1.0) May 2023

This unit specification provides detailed information about the unit to ensure consistent and transparent assessment year on year.

This unit specification is for teachers and lecturers and contains all the mandatory information required to deliver and assess the unit.

This edition: May 2023 (version 1.0)

Unit purpose

This unit enables learners to become familiar with the concepts, principles and practices around data programming. Programming elements include:

- integrated development environments (IDEs)
- programming languages for data analysis
- functional programming
- reproducibility
- version control
- use of data application programming interfaces (APIs)

The data elements of the unit include:

- data security
- data types and structures
- automation
- data manipulation
- statistical functions
- performing data analysis tests

This is a non-specialist unit, intended for a wide range of learners. It is particularly appropriate for learners with an interest in computing and its associated sub-disciplines, including data science. Learners should have previous experience of data analysis. It is also desirable for learners to have programming skills and an understanding of basic statistics.

On completion of the unit, learners may progress to further units in computer programming or data analysis, such as Computer Programming at SCQF level 8, Data Engineering at SCQF level 9 and/or Machine Learning at SCQF level 9.

Unit outcomes

Learners who complete this unit can:

1 describe the processes and technologies used in a data programming environment
2 explain the concepts, processes and practices used to create clean, efficient code for data analysis
3 write functions and programs to perform data analysis
4 test programs for data analysis

Evidence requirements

Learners must provide product evidence of at least two programs for data analysis. Knowledge is inferred from the product evidence. The product evidence relates to all outcomes.

The evidence must demonstrate that learners can:

- set up a development environment using appropriate processes and technologies
- select appropriate data structures and data models
- create quality, reusable code in line with good practice
- test and debug the program code
- produce technical and user documentation for the program

Each program must:

- read data from two or more external sources
- perform data cleaning using one or more algorithms
- perform data wrangling to produce usable data for analysis
- perform a statistical analysis of the data
- produce one or more visualisations from the analysis

The code must be error-free and produce correct results from its test data.

Learners can produce evidence over an extended period in lightly controlled conditions or generate it holistically in conjunction with other units in a group award. Evidence produced in lightly controlled conditions must be authenticated. The Guide to Assessment provides further advice on methods of authentication.

The standard of evidence should be consistent with the SCQF level of the unit.

Knowledge and skills

The following table shows the knowledge and skills covered by the unit outcomes:

Knowledge	Skills
Learners should understand: ♦ software types for data analysis ♦ high-level programming languages for data analysis, including functional languages ♦ database languages, including SQL ♦ low code software development tools, including notebooks ♦ IDEs ♦ command-line interfaces ♦ APIs ♦ software libraries ♦ programming process ♦ source and version control ♦ how to write documentation for a function of program ♦ programming techniques ♦ data sources ♦ data types and data structures ♦ library functions for data analysis ♦ data interfaces and data flow ♦ data modelling ♦ data security and security management ♦ code reuse ♦ algorithmic bias and data bias	Learners can: ♦ set up a project including software installation ♦ write a function or program to process data from an external file ♦ write a function or program to process data from a database ♦ write a function or program to clean data ♦ write a function or program to perform a statistical analysis ♦ write a function or program to create visualisations ♦ write documentation for a function/program ♦ create test plans ♦ test datasets ♦ train datasets ♦ review and debug programs ♦ perform tests and revise code accordingly

Meta-skills

Throughout this unit, learners develop meta-skills to enhance their employability in the software development sector.

Self-management

This meta-skill includes:

- focusing: sorting and maintaining documentation throughout development in a logical and efficient manner; attention to detail to ensure error-free, robust program code; considering all aspects of user requirements
- adapting: critically reflecting on personal skills development; self-learning to develop wider skills and extend development beyond requirements and taught content
- initiative: independent thinking to establish user requirements and design solutions; developing a data analysis application based on client information; self-motivation and taking ownership of personal development; time management

Social intelligence

This meta-skill includes:

- communicating: receiving information to establish the user requirements and ensure an understanding of the brief; giving information during application testing to confirm objectives and ensure that requirements are understood
- feeling: storytelling through the creation of technical documentation; providing a walk-through and detail of the program and data analysis outcomes

Innovation

This meta-skill includes:

- creativity: using imagination to provide a solution that automates data analysis; seeing the data analysis product through the eyes of the user; generating ideas and thinking about problem areas and how to provide a solution; visualising to create an overall impression of the completed solution throughout the process
- sense-making: analysis; seeing the bigger picture
- critical thinking: logical thinking to ensure a coherent approach and to meet requirements appropriately

Delivery of unit

If you deliver this unit as part of a group award, we recommend that you teach and assess it in the subject area of the group award to which it contributes.

You could deliver the unit following Working with Data at SCQF level 8. The knowledge and skills gained by completing that unit would provide a sound foundation for the automation of data analysis.

The unit focuses on data programming, rather than more general programming (or software development) skills. Units such as Computer Programming at SCQF level 7 and Computer Programming at SCQF level 8 should be used for the purpose of developing computer programming skills in general.

We suggest the following distribution of time:

Outcome 1 — Describe the processes and technologies used in a data programming environment
(5 hours)

Outcome 2 — Explain the concepts, processes and practices used to create clean, efficient code for data analysis
(5 hours)

Outcome 3 — Write functions and programs to perform data analysis
(20 hours)

Outcome 4 — Test programs for data analysis
(10 hours)

Additional guidance

The guidance in this section is not mandatory.

Content and context for this unit

This unit focuses on the importance of writing structured, clean, and logical code, following good programming practices. The unit is suitable for a wide range of learners with an interest in the practicalities of using code for data analysis.

In outcome 1, learners explore the programming languages used to create the programs for data analysis. They look at the range of languages used in data analysis, such as Python, R, MATLAB, and Scala. While learners should be initially exposed to a range of languages, they should focus on one programming language for programming tasks in the remainder of the unit.

Learners look at where and how datasets are stored (locations and formats) and how they are read into a program by utilising the programming language's read functions and libraries, such as Python's comma-separated values (CSV) module. They also explore data types; how data is stored in the language; and how it flows through the program. For example, in Python, learners can use the pandas library to replace an Excel table with data frames. You should place emphasis on the use of modularity, using functions, libraries and code reuse.

In outcome 2, learners focus on the importance of setting up a proper data programming environment. They look at options for setting up, installing and configuring different development environments. For example, to utilise Python programming, this could include setting up Anaconda Python distribution as a package and environment manager, and an IDE like Atom or Jupyter Notebook. In the R programming language, learners may use RStudio as an IDE and Git for source control. Learners also utilise command line, such as Unix Shell to automate tasks. For example, using grep, awk and sed commands, along with Git.

Alongside the advantages of good programming practice (such as standardised naming conventions and code documentation), learners should also discover the importance of securing data, including the tools and functions used for managing digital authentication (secret management).

Outcome 3 is practical in nature and involves learners writing a series of functions and programs. Learners should follow the processes and structures they learned in outcomes 1 and 2 when they write their programs. They should write functions and/or programs to include:

- reading data from external sources (at least two)
- data-cleaning algorithms
- data wrangling
- statistical analyses
- visualisations

Outcome 4 is also practical in nature and involves learners setting up tests for their programs, writing the code to run the test, and reviewing the test after completion. They should be able to describe the test plan and report its results.

Approaches to delivery

In outcome 1, you should teach learners a range of development environments, languages and types of developer interface. Give learners opportunities:

- for presentations
- for classroom discussions
- for group work
- to consolidate their learning with practical exercises. For example:
  - setting up IDEs
  - interacting with version control software (Git, Bitbucket)
  - working with command-line interfaces
  - installing software for data analysis
- to research advantages and disadvantages of each language and approaches to programming with data

In outcomes 2, 3 and 4, learners should focus on one programming language and one development environment. It is not a prerequisite for learners to necessarily have programming experience. Therefore, you should take time to cover the key concepts needed to create efficient code. Learners should be familiar with these concepts to assist in the writing of their programs and functions.

Approaches to assessment

Evidence can be generated using different types of assessment. The following are suggestions only. There may be other methods that would be more suitable to learners.

Where learners experience a range of assessment methods, this helps them to develop different skills that should be transferable to work or further and higher education.

There are plenty of opportunities to carry out formative assessment through every stage in the unit. This provides you with the opportunity to intervene if learners have misconceptions or lack clarity on topics. You can provide diagnostic feedback to learners on an ongoing basis.

We recommend that you assess all the outcomes using an e-portfolio. E-portfolios give learners the opportunity to take ownership of their own work and to assemble the e-portfolios themselves. Learners can update their e-portfolio regularly, as they progress their learning and skills. However, learners must understand that, to achieve the unit, the e-portfolio they submit for assessment must contain the items that clearly recognise and record their achievement.

Learners could use the following examples of evidence in their e-portfolios:

- screenshots of installing software, interacting with Git, setting up IDEs
- blog posts: reflecting on programming languages, different interfaces
- presentations: (audio and/or video)
- code listings, documentation, test plans, test results, created charts, interaction with source control software

The programs must include:

- reading data from external sources (at least two)
- data-cleaning algorithms
- data wrangling
- statistical analyses
- visualisations

E-portfolios allow learners to use different types of electronic evidence for assessment in its original format. They offer an assessment approach that is inherently 'learner-centred' and more flexible. Learners can assemble one portfolio and tailor it to specific audiences by tagging items for different purposes (including different assessments). E-portfolios allow evidence to be stored more securely so it is more accessible to learners and to you. They make it easier for you to give feedback, which strengthens the links between formative and summative assessment, and between learning and assessment generally.

Opportunities for e-assessment

Assessment that is supported by information and communication technology (ICT), such as e-testing or the use of e-portfolios or social software, may be appropriate for some assessments in this unit.

If you want to use e-assessment, you must ensure that you apply the national standard to all evidence and that they meet the conditions of assessment (as specified in the evidence requirements), regardless of the mode of gathering evidence. The most up-to-date guidance on using e-assessment to support SQA's qualifications is available at: www.sqa.org.uk/e-assessment.'

Equality and inclusion

This unit is designed to be as fair and as accessible as possible with no unnecessary barriers to learning or assessment.

You should take into account the needs of individual learners when planning learning experiences, selecting assessment methods or considering alternative evidence.

Guidance on assessment arrangements for disabled learners and/or those with additional support needs is available on the assessment arrangements web page: www.sqa.org.uk/assessmentarrangements.

Information for learners

Programming for Data (SCQF level 8)

This information explains:

- what the unit is about
- what you should know or be able to do before you start
- what you need to do during the unit
- opportunities for further learning and employment

Unit information

This unit helps you become familiar with the concepts, principles and purposes around programming for data. The primary focus is on writing the good quality, reusable, clean program code needed to process and analyse data. The unit is particularly appropriate if you have an interest in computing and its associated subdisciplines, including data science.

Some experience of performing data analysis is required for the unit, and it would be advantageous to be familiar with programming concepts and have an understanding of basic statistics.

The unit covers the following topics, among others:

- setting up a data analysis environment
- understanding data flow
- functional programming
- data types and structures
- programming concepts, structures and processes used to write quality, reusable code
- designing a test plan for a program
- testing and debugging code
- writing documentation

You write complex programs in a designated language common in data analysis, such as Python, R, Scala or MATLAB.

Throughout the unit, you develop meta-skills covering self-management, social intelligence and innovation.

Assessment is likely to be through a range of assessment methods, all of which are practical in nature.

On completion of the unit, you may progress to Computer Programming at SCQF level 8, Machine Learning at SCQF level 9, or Data Engineering at SCQF level 9.

Administrative information

Published: May 2023 (version 1.0)

Superclass: CB

History of changes

Version	Description of change	Date

Note: please check SQA's website to ensure you are using the most up-to-date version of this document.