Unit 5: Introduction to Programming

Delivery guidance

Approaching the unit

The purpose of this unit is for learners to understand the key concepts of event-driven, procedural, and object-oriented programming paradigms and to develop coding skills in these languages. Learners need to be taught about the different programming paradigms. Learners should be encouraged to learn the difference between the three programming paradigms including what features they have. Learners need to understand why there are different programming paradigms and the types of programs that each programming paradigm is suited to.

Learners then need to develop programming skills in one chosen language. Learners can choose any suitable programming language that allows them to demonstrate the constructs and techniques listed in the specification. Once students have developed their programming skills they then need to develop, test, review and refine their own software solution to fulfil a brief.

Learners will need to be given considerable practical workshop opportunities to enable them to grasp the programming concepts.

This unit allows for a range of different delivery methods, such as:

- Class and group discussions – For example learners can be given lots of different snippets of code and then be asked to discuss the purpose of each line of code.
- Individual or group presentations – For example learners can research different programming tools and techniques and share their findings with others.
- Demonstrations – Learners can be shown different programming techniques and then be asked to practice using these in different scenarios.
- Case studies – For example, learners can look at specific program requirements and then assess which programming paradigm and constructs can be used to solve these.

You can involve local employers in the delivery of this unit if there are local opportunities to do so.

Delivering the learning aims

When delivering this unit, learners should be given access to an appropriate integrated development environment that allows them to write programming code. Throughout the unit it is important that learners understand the different features, constructs and techniques used in programs. However, as well as learning definitions of these, it is important that learners can identify where these are being used in programming code. The student book contains lots of small snippets of code written in C#, Python and Java that can be referred to while teaching the different learning aims.

Introducing the unit

Some learners may have limited experience writing programming code. In particular, they may have limited knowledge about the different programming paradigms that are available. Therefore, before learners start learning aim A, it is important that learners have a basic idea of

what a programming language is and how this can be used to develop programming code. The student book contains an introduction to the unit section that can be used to achieve this.

It would be beneficial if learners develop some small programs within a particular integrated development environment to give them an idea of how to write code, where to write code and how to test code. It is extremely likely that learners will come across bugs and errors while testing their programming code. Learners will therefore benefit from learning the different types of errors and common techniques that can be used to debug cod

Learning aim A

Once learners have had a basic introduction to what a programming language is and how it can be used to write programming code, learners can then be taught the unit content listed in A1, A2 and A3. Each of these covers a different programming paradigm and these can be covered in any order. However, it would be beneficial to start with the programming paradigm that learners have most experience with before moving onto the one(s) they have less experience with.

They should examine the key features of event-driven, object-oriented, and procedural paradigms. The slide presentation contains an overview of the different features of these paradigms. The student book contains more detailed explanations together with diagrams and practical activities to help learners understand the differences between these.

The different paradigms will be interesting to many learners, especially the way they are used for different software solutions. The class can be divided into groups, each with a different programming language and sample programs. The groups can then discuss the features, advantages, and disadvantages of their languages for each program. The programs can be rotated between the groups, so everyone gets to see each language. The groups can then present their findings back to the class.

Learning aim B

This learning aim focuses on developing and testing software solutions. Learners will develop a range of programming skills in one programming language. The slide presentation contains a description of all the key programming techniques in this unit. The student book also contains lots of small snippets of programming code within three different programming languages C#, Python and Java.

The unit content in topic B1 is largely practical and learners would benefit from watching demonstrations of the different constructs and techniques being used. These can then be reinforced by giving learners additional challenges by allowing them to use those constructs and techniques to solve a different problem. Although the slide presentation and student book contain all the key techniques in the unit, students are not required to use them all. They will need to consider the purpose of different programming constructs and then select which ones they want to use to address client requirements. Learners will then need to consider different processes involved with testing programming code to ensure it is functional and robust.

While a lot of the lessons will be predominantly practical with learners working on their programs, it is important that you should be ready to give recap sessions on topics that are causing learners particular problems. The presentation slides can be used to help learners recap their knowledge.

Once topic B1 has been covered, learners can then cover the unit content listed in topic B2. Here learners will learn how to test their programming code. It would be beneficial if learners had

access to programming code containing errors. They can then be challenged to create a test plan using a range of test data to locate and fix the errors.

Learning aim C

In this learning aim learners will learn what should be considered during a software review. The student book contains a variety of different checklists for each point listed in topic C1. Learners can use these to review their programming code. It would be beneficial if learners could practice applying these checklists to programs that they have already written. If this is not possible then learners can be given programming code to practice reviewing. Learners will be able to use the checklists in the student book to be able to identify strengths and weaknesses in their solution.

It is important that learners refer to the original requirements and explain how effective their solution is and how it meets the user requirements and the purpose. They will also consider any constraints that they had while creating their solution and how these could be overcome. They will end the unit by reviewing the strengths of their solution and identifying possible future improvements.

Assessment model

Learning aim	Key content areas	Recommended assessment approach
A Understand the key features of different programming paradigms	**A1** Features of event-driven programming **A2** Features of object-oriented programming **A3** Features of procedural programming	An investigation into each programming paradigm and their suitability for a particular problem(s).
B Develop and test a software solution	**B1** Developing software **B2** Testing and refining a software solution	A working program in one programming language. Testing documentation that includes the outcomes of testing and any refinements made during development. Refine the solution based on testing.
C Review the software solution	**C1** Software solution review	A written report that reviews the suitability of the software solution.

Assessment guidance

This unit is internally assessed by two summative assignments. Teachers should refer to the assessment guidance in the specification for specific detail, particularly in relation to the requirements for the Pass, Merit and Distinction grades.

There are two Authorised Assignment Briefs (AABs) for this unit, which centres are advised to use. It is sensible to issue the first assignment brief to learners after they have covered all the

unit content in learning aim A. Learners should be given the second assignment brief after they have covered all the unit content in learning aim B and C.

Learning aim A is assessed using a written report. In their report, the learners must demonstrate an understanding of the different programming paradigms and their suitability for a particular problem.

To support learners in their report writing, they should be encouraged to use headings, sub-headings, and annotated illustrations, such as pictures and diagrams, to support explanations.

Learning aim B will initially focus on developing a software solution to meet user requirements. To access the higher assessment criteria, the solution needs to be refined and improved with justifications given.

B.P3, B.P4 and B.M2 all require some practical assessment. The learners will need to successfully develop a software solution against specified user requirements. The learners will also need to test the software solution and clearly show how issues were rectified.

Hard evidence of the practical assessment should be provided, for example learner observation records, or screen recordings of the software solution being run.

A written or verbal discussion of the interpretation of the software test results should then be held.

Learning aim C requires learners to consider the effectiveness of the software solution against the intended user requirements and purpose. This is assessed by a written report, where learners make judgements about the effectiveness of the software solution, giving examples to support the judgements.

Getting started

This gives you a starting place for one way of delivering the unit, based around the recommended assessment approach in the specification.

Introduction

Introduce the unit to your learners by giving them a brief outline of the content and how it links with other units in the qualification. Engage learners by giving them the knowledge and understanding to interpret a software solution and explain that by the end of this unit they will have covered the skills needed to create their own software solutions.

Learning aim A: Understand the key features of different programming paradigms.

Consider starting delivery with a discussion to establish what learners already know about the different programming paradigms.

Give learners access to examples of software developed in different languages that use a variety of code bases. Learners should identify the techniques and processes used.

Demonstrate each programming paradigm, explaining the controls and features.

- **A1 Features of event-driven programming**

  Provide learners with simple programs created in an event-driven programming language and ask them to feed back the code/runtime features as well as strengths and weaknesses of the programs. Ask learners to consider what sort of applications would benefit from an event-driven environment, for example a graphics-based software solution would be better developed in an event- driven environment.
- **For A2**, provide learners with simple programs created in an object-oriented programming language and ask them to feed back the code/runtime features as well as strengths and weaknesses of the programs. Ask learners to consider what sort of applications would benefit from an object-oriented environment, for example artificial intelligence and expert systems software solutions would be better developed in an object-oriented environment.
- **For A3**, provide learners with simple programs created in a procedural programming language and ask them to feed back the code/runtime features as well as strengths and weaknesses of the programs. Ask learners to consider what sort of applications would benefit from a procedural environment, for example a server-side software solution would be better developed in a procedural environment.

Case studies:

The student book contains a case study on a fictional company called Programming World. It contains information about a treasure-hunt game. The case study suggests that a procedural programming paradigm will be used with a top-down approach. Learners can use this case study to analyse if this is the correct paradigm to use in this context.

Discussion topics:

Ask learners to talk about what programming experiences they have had already. It is likely that some learners will have a keen interest in programming and may have self-taught

themselves how to program. Ask learners what programming languages they have used and if they have been self-taught, ask them why they have chosen to learn these programming languages.

Formative assessment activities:

The student book contains the requirements for a new pet game. Learners can read the requirements of this program or another program and then asked to evaluate the suitability of event-driven, object-oriented, and procedural programming paradigms for this program. They can evaluate the features of each paradigm that can be used together with the benefits and drawbacks of each approach.

Preparing for summative assessment:

The student book contains a practice assessment. This places learners into a junior software developer role and contains information about three programs that need to be developed. These are a carpet cleaning booking system, a program that will ask and mark maths questions and a racing car game. Learners are required to create a presentation that evaluates the suitability of different programming paradigms for each of these programs.

Learning aim B: Develop and test a software solution

This learning aim will give learners the tools they need to create a software program in a chosen language. Give learners plenty of tuition in the chosen language to address each part of the unit specification.

- **For B1,** start by presenting learners with the theory behind the way in which data should be handled within a program.

  Deliver a series of 'supervised' practical coding workshops that will enable all learners (individually, in pairs or in small groups) to code in a particular language. It is at this stage that learners will need access to an IDE (Integrated Development Environment) for one of the programming paradigms.

  When you are confident that learners understand the key concepts, give them several practical exercises to practise these concepts. For example, learners could be given a workbook to work through at their own pace as the topic progresses. This workbook should give them coding snippets to show how the concepts are applied to problems. This will also give you an opportunity to monitor learners' progress.

  As learners hone their programming skills and work through various tasks, it would be useful to bring learners together periodically to recap the skills that they have mastered. Also, when an exercise proves challenging for some learners, you could work through model answers on an interactive whiteboard, with the assistance of other learners in the class. This will also give you a chance to ask learners direct questions to check their understanding. You could ask stronger programmers in the class to formally buddy/work with weaker learners.

  You will need to provide specific sample programs that are appropriate for your learners to be able to demonstrate the use of data types, variables, operators, data structures and event handlers.

  Provide learners with a 'client brief.' Discuss the brief, ensuring that all learners understand the requirements and what is expected of them.

Explain at the outset that there is always more than one way of improving the performance of software solution and give examples.

Deliver a presentation on testing software solutions and explain (with examples) how test plans are produced. Ask learners to think about the range of tests they would carry out on a software solution.

Provide short activities that cover each of the above points, so that learners have everything they need to prepare a plan.

- **For B2,** demonstrate the processes involved in testing and refining software solutions for a given brief. Ask learners to observe and take notes. Learners could then go onto testing and refining their own programs that were created in previous sessions.

Case studies:

The student book contains a case study on a fictional company called CodingSolutions123. It contains information about a program called SafeMate. The case study gives information about how the organisation plans to test this before releasing it to the public. Learners can use this to determine the impacts of the suggested testing strategy.

Discussion topics:

When learners are writing programming code there will be lots of opportunities for learners to discuss their code. There will always be multiple different ways to solve a problem and therefore learners can be questioned about alternative ways that they would solve the problem.

Formative assessment activities:

The student book contains lots of different programming activities for learners to complete. These include programs that require learners to demonstrate how to use sequence, selection, iteration, and arrays. It also contains the requirements for a new program that is needed to allow people to monitor how many minutes they exercise each week. Learners can use this or be given the requirements of another program and then be asked to develop, test, and refine the program using a programming language of their choice.

Preparing for summative assessment:

The student book contains a practice assessment. This places learners into a junior software developer role and contains information about three programs that need to be developed. These include a program that will ask the user mathematical questions, a two-player guess the word game and an Os and Xs game. Learners can assess the requirements of each of these and then develop, test, and refine the programming code for one of these programs.

Learning aim C: Review the software solution

This learning aim will provide learners with the knowledge required to make decisions on whether software solutions are successful.

- **For C1**, learners could individually review their work. They should:
  - explain why their software solution is suitable for the intended purpose and the original requirements

- detail whether they experienced any problems and, if they had to change their plan to resolve them, justify the reasons why they had to change their original plan
- make at least **three** specific suggestions for improving the completed program to ensure it is fully functional, well coded and fit-for-purpose.

You could ask learners to arrange an interview with the 'client' (which could be the teacher or other learners) to discuss the refined/modified software solution in terms of whether the requirements have been fully met. Learners should record the feedback and include this as evidence towards partially satisfying the assessment criteria. Before the interview, get learners to prepare what they will discuss, giving them advice, if necessary.

Case studies:

Learners can research examples of famous programs that were released with errors and bugs due to inadequate reviews. Learners can discuss what impact they think these programs had on users.

Discussion topics:

Each programming language comes with a different set of constraints. Learners can discuss features that they wanted to embed into their programs that they were not able to. They can then discuss alternative tools that they used to overcome this or an alternative language that could be used.

Formative assessment activities:

The student book contains lots of different checklists that can be used to review a program. Learners can open some programming code that they have developed and then practice applying the checklists to this program. They will be able to use these to identify the strengths and weaknesses of their code.

Preparing for summative assessment:

The student book contains a practice assessment. The practice assessment asks learners to review the program that they have created in learning aim B.

Details of links to other BTEC units and qualifications, and to other relevant units/qualifications

This unit links to:

- Unit 8: Introduction to App Development.

Resources

For this unit, learners must have access to:

- computers and internet access to research different features, constructs, and techniques of different programming languages
- an integrated development environment that allows learners to write, debug and test their programming code
- programming code tutorials for one programming language
- lots of small snippets of code to show learners practical uses of the different features, constructs, and techniques of different programming languages
- suitable word processing software to write up their notes
- Student Book

Integrated Development Environment:

Learners need to be given access to an integrated development environment that allows them to use the unit content listed in the specification. Example integrated development environments include:

Visual Studio Code

Eclipse

IntelliJ IDEA (International Design Excellence Awards)

PyCharm

Pearson is not responsible for the content of any external internet sites. It is essential for teachers to preview each website before using it in class so as to ensure that the URL is still accurate, relevant and appropriate. We suggest that teachers bookmark useful websites and consider enabling learners to access them through the school/college intranet.