Pearson BTEC

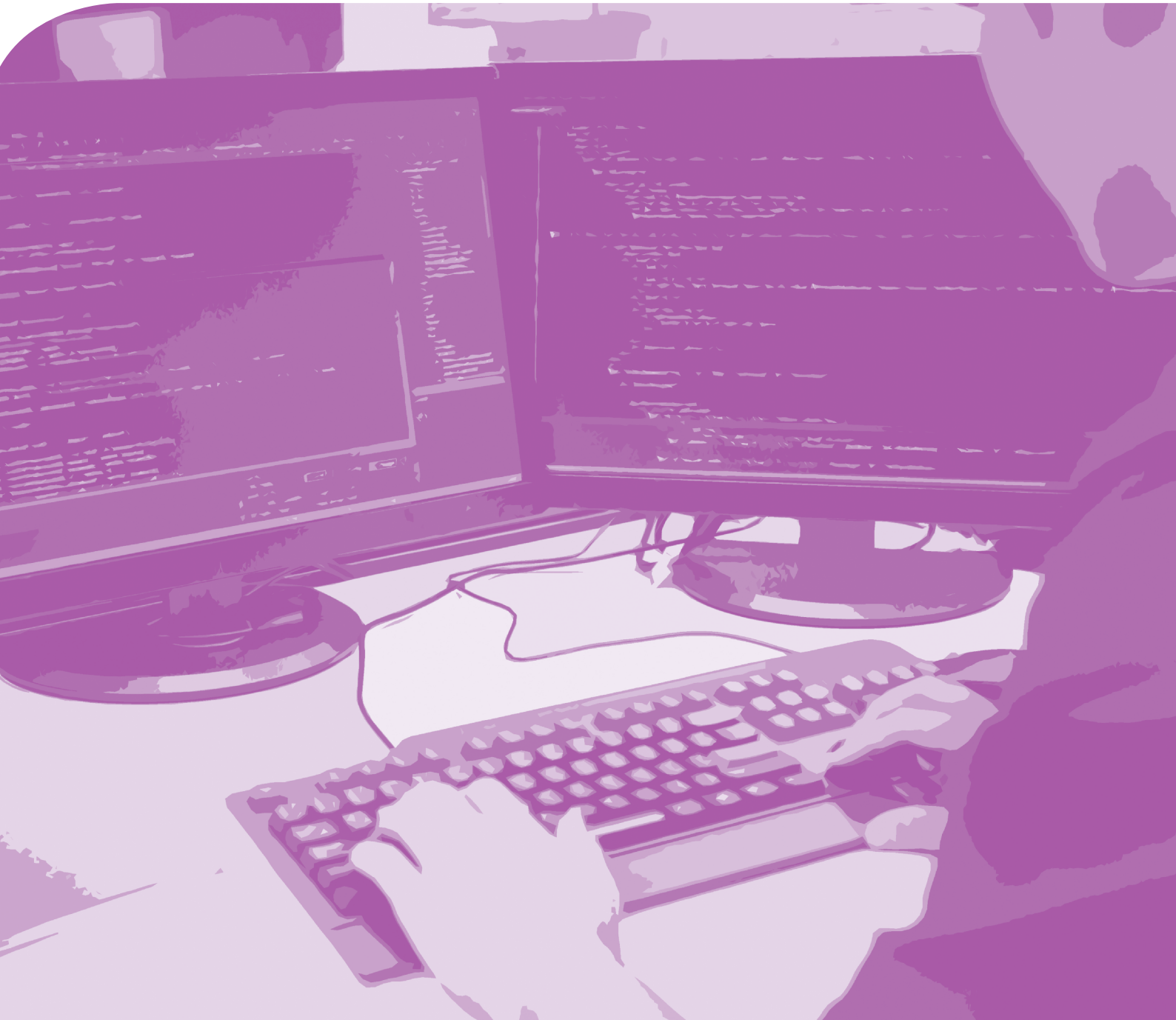
Pearson BTEC Uzbekistan Level 4 Qualifications in

Software Development

Unit 1: Introduction to Programming

Teacher Resources

Issue 1



Edexcel, BTEC and LCCI qualifications

Edexcel, BTEC and LCCI qualifications are awarded by Pearson, the UK's largest awarding body offering academic and vocational qualifications that are globally recognised and benchmarked. For further information, please visit our qualifications website at qualifications.pearson.com. Alternatively, you can get in touch with us using the details on our contact us page at qualifications.pearson.com/contactus

About Pearson

Pearson is the world's leading learning company, with 35,000 employees in more than 70 countries working to help people of all ages to make measurable progress in their lives through learning. We put the learner at the centre of everything we do, because wherever learning flourishes, so do people. Find out more about how we can help you and your learners at qualifications.pearson.com

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

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

ISBN 978 1 446 96133 9

Contents

Introduction

This resource booklet is a companion to the BTEC Uzbekistan Level 4 Qualifications in Software Development. The specification tells you what must be taught and what must be assessed. This resource booklet gives you suggestions and ideas as to how you can do this.

This booklet gives you ideas for teaching and learning, including practical activities, realistic scenarios, ways of involving employers in delivery and of managing independent learning, and how to approach assessments. The booklet also shows you how the specification content might work in practice and inspires you to start thinking about different ways of delivering your qualification.

This resource booklet gives you:

- guidance on how to deliver the units in the qualification
- recommended resources to support the delivery of the units in the qualification
- schemes of work that show the topics, activities and assessments covered in all units across the qualification
- lesson plans with detailed guidance on how to deliver the lessons in the units.

The information in this resource booklet has been put together by teachers who have been close to the development of the qualifications and so understand the challenges of finding new and engaging ways to deliver BTEC qualifications.

The delivery guidance in this booklet gives you information on what you need to consider as you plan the delivery of the qualification. There is information on:

- the structure of your qualification
- how you can build the qualification for your learners
- suggestions for how you might make contact with appropriate employers
- information on other support and resources available.

We have given you unit-by-unit guidance. This includes suggestions on how to approach the learning aims and unit content, as well as ideas for interesting and varied activities. You will also find tips and ideas on how to plan for and deliver your assignments.

We have included a list of carefully selected resources for each unit. These resource lists offer suggestions for books, websites and videos that you can direct your learners to use and/or that you can use to complement delivery.

Unit 1: Introduction to Programming

Delivery guidance

Approaching the unit

The purpose of this unit is to provide an introduction to the problem-solving skills that are essential for software developers. The unit assesses learners' ability to design, create and test a computer program to meet the requirements of a given brief. Explain to learners from the outset that programming in this unit focuses on the procedural programming paradigm. While centres are free to choose any procedural high-level language, as long as the content listed in the specification can all be covered, centres are encouraged to use Python 3 in the delivery and assessment of this unit.

For learning aim A, start by exploring the concepts of computational thinking and how they directly relate to problem-solving skills. Do this through whole-class discussions and small-group discussions about how these concepts are applied to the process of problem-solving and computer programming. Set small-group activities to examine how to apply the problem-solving skills identified in the specification to a given concept or problem.

For learning aim B, it is important to give learners practical activities that will develop their programming skills. Initially, activities should start as short, self-contained tasks that focus on single programming concepts or commands. Over time, you should increase the demand of activities in such a way that learners must select and combine a number of different skills.

Encourage your learners to take notes and create their own reference guides, which they can share with their peers. Ask learners to create a shared folder for their notes, either on a drive, on the centre's VLE or using a cloud-based service such as Google Drive™ or Microsoft OneDrive®. Where tasks allow, i.e. where they are non-assessment tasks, encourage learners to work collaboratively by giving them feedback on they work they have produced and when discussing solutions.

For learning aim C, you should ensure that learners are aware of the need for robust and reliable programs, and the role of testing. Encourage learners to carry out document testing as they go and explain that an iterative approach to programming leads to much higher-quality outcomes.

You must give learners appropriate feedback throughout the course, give them opportunities to read and understand feedback given, and encourage them to use their initiative to identify areas of development. Encourage learners to work on their programming skills as much as possible outside of the classroom. Programming is a skill that benefits from practice and repetition.

Getting started

This gives you a starting place for one way of delivering the unit. It is based on the recommended assessment approach given in the specification.

Unit 1: Introduction to Programming
Introduction Well-designed and well-developed software can be a crucial part of any business or organisation. Software that fails to work or that does not fully solve an identified problem will cost money and damage a business's reputation. It is important, therefore, to ensure that a problem is broken down into smaller, more manageable chunks and that a solution is clearly described in order for an effective program to be developed and fully tested.
Learning aim A – Plan a solution to an identified problem
• You could begin by introducing the aim of the unit and how problem-solving skills can be applied to practical problems. • Explain to learners that problem solving in software development makes use of 'computational thinking', which allows us to analyse a problem. This type of thinking can be considered to have four 'stages': decomposition, pattern recognition, pattern generalisation and abstraction, and algorithm design. • Learners work in small groups to look at an example of a program (such as a simple computer game) and identify how each of the four 'stages' of computational thinking may be applied. For example: o for 'decomposition', learners might be expected to explain the stages of the game, the win scenario, the role of a particular character in the game etc. o for 'pattern recognition', they may identify that certain non-playable characters move in the same way o for 'pattern generalisation and abstraction', they could identify the required variables or start defining the required inputs o for 'algorithm design', they could describe parts of the game in order to plan the logical steps needed in readiness for writing code, such as what happens when a playable character makes contact with a certain item. • Topic A2 defines the knowledge, skills, and understanding required for algorithm design in topic A1. For this topic, learners do not need to be introduced to programming code. At this stage, however, some concepts, such as 'sequence, selection and iterating' should be introduced, for example, identifying something that needs repeating, why this is the case and what form this repetition will take. • Give learners the opportunity to explore a range of problems to which they can apply their problem-solving skills.

Unit 1: Introduction to Programming
Learning aim B – Produce program code to solve an identified problem
• This learning aim will give learners the tools they need to create a software program. When covering this learning aim, ensure that you focus on the procedural programming paradigm. It is recommended that centres use Python 3 to deliver and assess this unit. • Teaching of this learning aim content should be as practical as possible. Start with short, self-contained tasks that focus on single programming concepts or commands. Over time, increase the demand of activities in such a way that learners must select and combine a number of different skills. • When you are confident that learners understand a concept, give them several practical exercises to practise working with the concept. For example, when introducing the concept of 'selection', give learners a number of different scenarios for which they have to develop small pieces of code. Learners could be given a workbook that contains a number of tasks and challenges for them to complete. • As learners develop their programming skills and work through various tasks, it would be useful to bring them together periodically to recap the skills they have mastered. Also, when an exercise proves challenging for some learners, you could work through model answers as a whole-class activity. This also gives you a chance to ask learners direct questions to check their understanding. You could ask stronger programmers in the class to formally buddy with or work with less able programmers. • At regular points throughout the delivery of the unit, you should set short assessment activities. These activities should be used to assess progress and identify areas that need to be improved. • Once all content has been covered, you should set learners a mock assessment. The mock assessment must reflect the external assessment they will complete, so it should supply an appropriate problem for which they can create solutions. This task should be completed independently under controlled conditions that replicate those of the final assessment.

Unit 1: Introduction to Programming
Learning aim C – Test and refine program code
• This learning aim is best taught in conjunction with learning aim B. Once learners have an understanding of some of the basic skills of programming, introduce the importance of ongoing testing. • Explore with learners the idea that, due to the size of most software development projects, programmers rarely work on their own and are often part of a much larger development team. Explain that, with large programs, errors are inevitable. Therefore, it is important to record the testing process so that issues can be traced and rectified more quickly. A test plan or test log is also an important way of communicating the requirements for testing to a larger team. • Explore with learners the importance of robust programs and the role that testing plays in ensuring programs are robust.

Details of links to other BTEC units and qualifications

This unit would benefit from being delivered at the same time as Unit 2: Software Analysis and Design.

Resources

Websites

https://docs.python.org/3/

The official site for Python programming language documentation.

www.learnpython.org/

Online Python programming tutorial. Useful for introducing individual commands or as reference guide. Learners can also use it to work on their programming skills outside the classroom.

www.w3schools.com/python/

Online Python programming tutorial. Useful for introducing individual commands or as reference guide, and learners can use it to work on their programming skills outside the classroom. The site also provides tutorials for other programming languages such as HTML and JavaScript, which may be useful for Unit 3.

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.

Scheme of work

Unit	Unit 1: Introduction to Programming
Guided Learning Hours	90
Number of lessons	45
Duration of lessons	2 hours
Links to other units	Unit 2: Software Analysis and Design

Key to learning opportunities			
AA	Assessment Activity	**RS**	Revision Session
GS	Guest Speaker	**V**	Visit
IS	Independent Study	**WE**	Work Experience

#	Topic	Lesson type	Suggested activities	Resources
1	A1 Problem-solving skills: • Decomposition.	IS	• **Starter activity:** introduce the unit and the main topics that will be covered. Explain the importance of problem-solving skills in programming. Explain that problem-solving will be applied throughout the qualification. • **Teacher-led demonstration:** explain decomposition. Demonstrate decomposition using an example of a computer problem. • **Small-group/paired activity:** learners practise decomposition using a given scenario. • **Teacher-led discussion:** learners feed their thoughts back to the whole group. • **Concluding activity:** give learners an opportunity to improve their decomposition, based on what has been discussed.	• Unit specification. • Flipchart or similar for learners to record discussions and ideas. • Presentation. • Example computer problem. • Example problem scenario for learners to examine.

#	Topic	Lesson type	Suggested activities	Resources
2	A1 Problem-solving skills: • pattern recognition • pattern generalisation and abstraction.	IS	• **Starter activity:** recap the concept of decomposition with a question and answer session. Explain that, in this lesson, learners will continue to explore problem-solving skills. • **Teacher-led demonstration:** explain pattern recognition, pattern generalisation and abstraction. Demonstrate using an example of a computer problem. • **Small-group/paired activity:** learners consider a given problem and analyse it using pattern recognition, pattern generalisation and abstraction. • **Teacher-led discussion:** ask learners to contribute their thoughts to the discussion. • **Concluding activity:** using what they have learned from the discussion, learners complete work on the scenario activity from the previous lesson.	• Unit specification. • Flipchart or similar for learners to record discussions and ideas. • Presentation. • Learners' work from previous lesson. • Example problem scenario from previous lesson.

#	Topic	Lesson type	Suggested activities	Resources
3	A1 Problem-solving skills: • algorithm design. A2 Techniques used to develop algorithms	IS	• **Starter activity:** introduce the lesson by explaining the terms *sequence*, *selection* and *iteration*. Explain that, over the course of the next two lessons, learners will be introduced to algorithm design. • **Teacher-led discussion:** use questioning to check learners' understanding of relational operators. • **Small-group activity:** give learners a problem to discuss. Learners should identify the order in which the tasks should be completed. • **Teacher-led demonstration:** introduce the idea of selection (branching) including If… Then… Else and Boolean operators. • **Small-group task:** give learners a problem to discuss. Learners identify the selection and iteration that would occur. • **Plenary activity:** recap main learning points from lesson.	• Unit specification. • Scenario and learner work from Lessons 1 and 2. • Presentation.

#	Topic	Lesson type	Suggested activities	Resources
4–5	A1 Problem-solving skills: • algorithm design. A2 Techniques used to develop algorithms	IS	• **Starter activity:** recap the concepts of decomposition, pattern recognition and pattern generalisation and abstraction through a question and answer session. • **Small-group/paired activity:** give learners a problem and ask them to consider 'algorithms' for different parts of the problem. • **Group activity (peer review and feedback):** at different stages during the two lessons, ask groups to discuss their work with other small groups. • **Concluding activity:** groups annotate their work during discussions to identify any changes to that will be made as a result of peer feedback.	• Unit specification. • Large sheets of paper. • Learner work from previous lesson. • Example problem scenario for learners to examine. • Different-coloured pens (for annotation).
6	A1 Problem-solving skills	AA	• **Starter activity:** organise learners so they can complete assessment tasks independently. • **Individual assessment activity:** give learners a scenario. They should produce a plan for how they could solve the problem. The plan produced should cover all main content points in learning aim A. **Note:** teachers should use the outcomes of the assessment activity to monitor learners' progress so far. This is not a final assessment.	• Unit specification. • Assessment activity sheet containing brief and scenario. • Copies of the content for learning aim A (problem-solving skills help sheet).

#	Topic	Lesson type	Suggested activities	Resources
7	Learning aim B • overview of procedural programming • B3 sequence	IS	• **Starter activity:** explain to learners that they will now look at a specific programming paradigm (procedural programming). • **Teacher presentation:** give an overview of the features and characteristics of the procedural programming paradigm. Give learners an introduction to the structure of procedural programming. • **Individual activity:** learners should research and make notes on the uses, benefits and drawbacks of procedural programming. • **Small-group activity:** organise learners into small groups to discuss their findings. Allow them to add to and expand their notes, based on these discussions. • **Plenary activity:** learners discuss what they have learned.	• Unit specification. • Presentation. • Computers with internet access. • Textbooks.

#	Topic	Lesson type	Suggested activities	Resources
8	B1 Handling data within a program: • constants and variables. B2 Arithmetic operations: • mathematical operators. B4 Built-in functions and libraries: • accept user input • output messages to the screen • convert numerical values to strings • convert strings to numerical values.	IS	• **Starter activity:** explain that, throughout the next phase of lessons, learners will be introduced to the basics of programming. • **Teacher presentation:** introduce some basic terminology that will be used in this session and give an introduction to the development environment that will be used. • **Individual activity:** learners work through a series of basic programming tasks (accepting user input, assigning a value to a variable, simple mathematical equations). • **Plenary activities:** learners discuss their experience of coding so far and complete a quiz on the topics covered in this lesson.	• Unit specification. • Presentation. • Programming task sheets. • Computers with Integrated Development Environment (IDE) installed. • Official Python documentation and online programming tutorials. [**Note:** these resources will be useful in all following lessons except during assessment activities.] • Quiz for plenary activity.

#	Topic	Lesson type	Suggested activities	Resources
9	B1 Handling data within a program: • list/array. B2 Arithmetic operations: • relational operators • Boolean operators. B3 Control structures: • selection (branching).	IS	• **Starter activity:** recap the work covered in the previous lesson and check learners' understanding with examples of code. • **Teacher-led discussion:** explore the concept of needing to store multiple values (lists of items) and needing to allow programs to branch (referring back to Lesson 3 as required). • **Teacher-led demonstration:** demonstrate how to create a list and how to create IF statements using Python. • **Individual activity:** learners work through a series of programming tasks on lists and IF statements. • **Concluding activity:** learners work in pairs to look at each other's code and suggest improvements.	• Unit specification. • Examples of code. • Presentation. • Computers with IDE installed. • Programming task sheets.

#	Topic	Lesson type	Suggested activities	Resources
10	B3 Control structures: • iteration. B4 Built-in functions and libraries: • define a range. B6 Principles of developing high-quality code: • maintainable code (code comments).	IS	• **Starter activity:** recap the work covered in the previous lesson and check learners' understanding with examples of code. • **Teacher-led discussion:** explore the concept of needing to repeat processes in a program (referring back to Lesson 3 as required). • **Teacher-led demonstration:** demonstrate how to create a 'For' loop and a 'While' loop using Python. • **Individual activity:** learners work through a series of programming tasks on iteration. • **Plenary activities:** demonstrate how to annotate code (code comments). Learners annotate their code to explain how it works.	• Unit specification. • Computers with IDE installed. • Examples of code. • Presentation. • Programming task sheets.
11	Learning aim B	AA	• **Starter activity:** organise learners so they can complete assessment tasks independently. • **Individual assessment activity:** give learners a series of programming challenges. The challenges should cover the content covered in Lessons 8–10. **Note:** teachers should use the outcomes of the assessment activity to monitor learners' progress so far. This is not a final assessment.	• Unit specification. • Assessment activity sheet. • Computers with IDE installed.

#	Topic	Lesson type	Suggested activities	Resources
12	B4 Built-in functions and libraries: • generate random numbers.	IS	• **Starter activity:** recap Lesson 10. Explain that this lesson will look at some of the built-in functions available in Python. • **Teacher-led discussion:** explore the concept of needing to make programs flexible and unpredictable. • **Teacher-led demonstration:** demonstrate how to generate random numbers using Python. • **Individual activity:** learners work through a series of tasks and a programming challenge on the use of random numbers. • **Concluding activity:** in pairs, learners discuss their work and provide peer feedback.	• Unit specification. • Presentation. • Computers with IDE installed. • Programming task sheets and challenge.

#	Topic	Lesson type	Suggested activities	Resources
13	B1 Handling data within a program: • using local and global variables • conventions B6 Principles of developing high-quality code: • produce readable code: variable names; naming conventions; spacing • produce maintainable code: built-in libraries; modular design – the use of self-contained functions.	IS	• **Starter activity:** recap the previous lesson and remind learners of the use of built-in functions. Explore the purpose of functions and the need to sometimes create your own functions. • **Teacher-led demonstration:** demonstrate how to generate, define and call a function and how to use local and global variables. • **Individual activity:** learners work through a series of tasks and a programming challenge on creating functions. • **Concluding activity:** in pairs, learners discuss their work and provide peer feedback.	• Unit specification. • Presentation. • Programming task sheets. • Computers with IDE installed.

#	Topic	Lesson type	Suggested activities	Resources
14	B4 Built-in functions and libraries: • round a number to a specific number of decimal places • truncate a number • locate the highest or lowest value in a group of numbers.	IS	• **Starter activity:** recap the previous lesson and explain that this lesson will explore other built-in functions. Recap use of lists. • **Teacher-led demonstration:** demonstrate the use of round, truncate, min and max functions in Python. • **Individual activity:** learners work through a series of programming tasks on creating functions. • **Plenary activity:** question and answer session on the topics covered in this lesson.	• Unit specification. • Presentation. • Computers with IDE installed. • Programming task sheets.

#	Topic	Lesson type	Suggested activities	Resources
15	B4 Built-in functions and libraries: • check if a string is written in upper case or lower case • check if a string contains numerical values • convert a string from upper case to lower case • convert a string from lower case to upper case.	IS	• **Starter activity:** recap Lesson 8 on converting from string to number. Also recap lesson 14. Explain that this lesson will continue to explore more built-in functions. • **Teacher-led discussion:** explore the need to make code robust, including checking user input. • **Teacher-led demonstration:** demonstrate the use of built-in functions in Python to check text/number and convert between lower case and upper case. • **Individual activity:** learners work through a series of programming tasks on creating functions. • **Plenary activity:** learners complete a quiz on the topics covered in this lesson.	• Unit specification. • Presentation. • Computers with IDE installed. • Programming task sheets. • Quiz for plenary activity.

#	Topic	Lesson type	Suggested activities	Resources
16	B1 Handling data within a program: • list/array • dictionary/map. B4 Built-in functions and libraries: • use functions for handling data in a list/array (append data; insert data at a specific location; remove data; pop data).	IS	• **Starter activity:** recap the previous lesson. Use a question and answer session to recap and to check learners' understanding of the basic use of lists/arrays (referring to Lesson 9 as needed). • **Teacher-led demonstration:** demonstrate the use of lists/arrays (including indexing). • **Individual activity:** learners work through a series of programming tasks using lists/arrays. • **Concluding activity:** learners work in pairs to discuss their code and provide feedback.	• Unit specification. • Presentation. • Computers with IDE installed. • Programming task sheets.

#	Topic	Lesson type	Suggested activities	Resources
17	B1 Handling data within a program: • list/array • dictionary/map. B4 Built-in functions and libraries: • use functions for handling data in a list/array (check if a specific character, word or value is in a list/array; return the location of a specific character, word or value; return the number of time a specific character, word or value is in a list/array; sort items in the list/array into ascending or descending order).	IS	• **Starter activity:** recap the previous lesson. Use a question and answer session to recap and check learners' understanding. • **Teacher-led discussion:** explore why programs need to store multiple values. Discuss some of the uses of lists. • **Teacher-led demonstration:** demonstrate the use of different functions to manipulate lists/arrays. • **Individual activity:** learners work through a series of programming tasks using the functions that have been demonstrated. • **Teacher-led demonstration:** demonstrate the use of a dictionary/map. Explore how it differs from a list/array. • **Individual activity:** learners work through a series of programming tasks using dictionaries/maps. • **Concluding activity:** learners work in pairs to discuss their code and provide feedback to one another.	• Unit specification. • Presentations. • Computers with IDE installed. • Programming task sheets.

#	Topic	Lesson type	Suggested activities	Resources
18	B4 Built-in functions and libraries: • slice a string in to individual characters • check if a specific letter or word is in a string • output the location of a specific character within a string • check the length of a string.	IS	• **Starter activity:** recap indexing (Lesson 16). • **Teacher-led demonstration:** demonstrate the use of indexing to locate specific letters in string and other string handling functions (including split(), len(), in). • **Individual activity:** learners work through a series of programming tasks using the functions that have been demonstrated. • **Concluding activities:** learners swap one of their pieces of code from today's lesson with another member of the class. Learners attempt to make the program crash and then make notes on what they did and what happened.	• Unit specification. • Presentation. • Computers with IDE installed. • Programming task sheets.

#	Topic	Lesson type	Suggested activities	Resources
19	B5 Validating data: • use built-in or custom functions to apply validation techniques to program code • write code that can handle incorrect input without causing errors • use iteration to ensure a program does not continue until acceptable input is received • generate program output that provides the user with helpful messages about any errors in data input and how to correct them.	IS	• **Starter activity:** recap concluding activity from Lesson 18. • **Teacher-led discussion:** explore the need for programs to be robust. Discuss specific scenarios in which programs would need to validate user input. • **Teacher-led demonstration:** demonstrate the use of programming conventions to create validation in programs. • **Individual activities:** learners work through a series of programming tasks and a programming challenge on validation. • **Concluding activity:** learners swap one of their pieces of code from today's lesson with a piece of code from another member of the class. Learners attempt to make the program crash and provide feedback.	• Unit specification. • Presentation. • Computers with IDE installed. • Programming task sheets. • Example of code requirements. • Code reference sheets (covering loops, string handling, indexing).

#	Topic	Lesson type	Suggested activities	Resources
20	B4 Built-in functions and libraries: • open external files • read from an external file • write to an external file.	IS	• **Starter activity:** recap the use of data structures (array/list, dictionary). Explain that sometimes it is more efficient to store data outside of the program code. • **Teacher-led demonstration:** demonstrate the use of built-in functions to manipulate an external .txt file. • **Individual activity:** learners work through a series of programming tasks on external .txt files. • **Plenary activity:** learners complete a quiz on the topics they have covered in the lesson.	• Unit specification. • Presentation. • Computers with IDE installed. • Programming task sheets. • Quiz for plenary.
21	Learning aim B	AA	• **Starter activity:** organise learners so they can complete assessment tasks independently. • **Individual assessment activity:** give learners a series of programming challenges. The challenges should cover the content covered in Lessons 8–19. **Note:** teachers should use the outcomes of the assessment activity to monitor learners' progress so far. This is not a final assessment.	• Unit specification. • Assessment activity sheet. • Computers with IDE installed. • Python coding help sheet or code and syntax reference guide.

#	Topic	Lesson type	Suggested activities	Resources
22	C1 Writing a test plan: • the purpose of the identified test • the test data to be used • the expected test results for the selected tests.	IS	• **Starter activity:** recap key points from Lessons 18 and 19 on robust programs. • **Teacher-led discussion:** explore the importance of robust programs and the role that testing plays in ensuring that programs are robust. • **Teacher-led demonstration:** explain the features of a test plan and the different types of test data. Explain why it is important for a software developer to document their testing. • **Individual activity:** learners create a detailed test plan for a given piece of code. • **Concluding activities:** in pairs, learners discuss their test plans. They should be prepared to explain or justify their choices of tests and test data. They can also add additional tests to their test plan.	• Unit specification. • Presentation. • Computers with IDE installed. • Partially completed program code and program requirements. • Test plan template.

#	Topic	Lesson type	Suggested activities	Resources
23	C2 Iterative code development: • systematically apply the test plan and associated test data.	IS	• **Starter activity:** recap key points from previous lesson. • **Teacher-led demonstration:** demonstrate the process of applying the planned tests, recording results, fixing code and retesting. • **Individual activity:** learners apply their test plan to the given code to identify errors and improve the code as required. • **Concluding activity:** learners annotate the code to identify how it works and the changes that they made.	• Unit specification. • Presentation. • Computers with IDE installed. • Partially completed program code and program requirements from previous lesson. • Learners' test plans from previous lesson.

#	Topic	Lesson type	Suggested activities	Resources
24	Learning aim A	IS	• **Starter activity:** explain to learners that in the next series of lessons they will complete a task that allows them to practise everything they have learned so far in this unit. • **Teacher-led discussion:** recap the key points from learning aim A (Lessons 1–6). Explain that learners will be given a problem that requires a computer program. Learners will plan, create and test the program. • **Individual activity:** learners create a detailed plan for the program (referring to learning aim A lessons as needed). • **Concluding activity:** in pairs, learners discuss their plans. They should be prepared to explain or justify their choices. Learners can then provide peer feedback.	• Unit specification. • Presentation. • Computers with IDE installed. • Problem-solving brief. • Copies of the content for learning aim A (problem-solving skills help sheet).

#	Topic	Lesson type	Suggested activities	Resources
25	Learning aims B and C	IS	• **Starter activity:** recap previous lesson. • **Teacher-led discussion:** explain the importance of iterative development and the need to test throughout the development of a program. Explain that some testing should be planned before development starts and then added to as the program develops. • **Individual activity:** learners create a test plan for the program they will create. • **Group/paired activity:** in pairs, learners discuss their plans. They should be prepared to explain or justify their choices and provide feedback. • **Individual activity:** learners start to create a program based on their plans from this lesson and the previous lesson.	• Unit specification. • Learners' work from previous lesson. • Presentation. • Problem-solving brief. • Test plan template. • Computers with IDE installed.

#	Topic	Lesson type	Suggested activities	Resources
26–27	Learning aims B and C	AA	• **Starter activity:** recap the focus for these lessons. • **Individual assessment activity:** learners develop and test their programs to meet the given brief. Use outcomes from Lessons 24–27 to identify learners' areas of strength and areas for improvement. **Note:** this is not a final assessment.	• Unit specification. • Learners' work from previous lesson. • Problem-solving brief. • Test plan template. • Language and syntax reference guide. • Computers with IDE installed.
28–29	Learning aims A, B and C	RS	• **Starter activity:** provide learners with your feedback for the work produced in Lessons 24–27. • **Individual activity:** learners read through the feedback they have been given and identify areas that require further study or improvement. • **Individual/paired activity:** learners complete individual or paired study on the areas that they need to improve.	• Unit specification. • Teacher feedback on work from Lessons 24–27. • Learners' work from Lessons 24–27. • Practice programming exercises. • Computers with IDE installed.

#	Topic	Lesson type	Suggested activities	Resources
30–32	Learning aims A, B and C	AA	• **Starter activity:** recap the process the learners followed in Lessons 24–27. Organise learners so they can complete assessment tasks independently. • **Individual assessment activity:** give learners a problem for which they must plan, create and test a solution. • **Independent study:** learners may use time away from the classroom to complete some tasks. **Note:** teachers should use the outcomes of the assessment activity to monitor learners' progress so far. This is not a final assessment.	• Unit specification. • Problem-solving brief. • Test plan template. • Language and syntax reference guide. • Computers with IDE installed.
33–34	Learning aims A, B and C	RS	• **Starter activity:** give learners copies of teacher feedback for the work produced in Lessons 30–32. • **Individual activity:** learners read through feedback and identify areas that require further study or improvement. • **Individual/paired activity:** learners complete individual or paired study on the areas they need to improve.	• Unit specification. • Teacher feedback on work from Lessons 30–32. • Practice programming exercises. • Computers with IDE installed.

#	Topic	Lesson type	Suggested activities	Resources
35–39	Learning aims A, B and C Practice assignment	AA	• **Starter activity:** explain that, in the next five lessons, learners will complete a practice assignment to prepare them for their final assessment. • **Individual assessment activity:** learners respond to a practice assignment brief that is similar to the final assessment. **Note:** the assessment activity used should follow a similar structure to the final assessment and should be graded against the same criteria. However, it **must not** be the same one that will be used to assess their final grade for the unit.	• Practice assessment activity. • Test plan template. • Language and syntax reference guide. • Computers with IDE installed.
40	Learning aims A, B and C	RS	• **Starter activity:** give learners their marked work and feedback on the outcomes from the practice assignment (Lessons 35–39). • **Individual activity:** learners read through their marked work and identify areas of improvement needed. • **Individual/paired task:** learners complete individual or paired study on areas which they need to improve.	• Unit specification. • Learners' work from Lessons 35–39. • Teacher feedback on practice assignment from Lessons 35–39. • Practice programming exercises. • Computers with IDE installed.

#	Topic	Lesson type	Suggested activities	Resources
41–45	Learning aims A, B and C Final assessment	AA	• **Starter activity:** introduce the assignment (see Assessment Workbook), detailing the main assessment criteria, deadline and submission criteria. • **Individual assessment activity:** learners use the Assessment Workbook complete Unit 1: Tasks 1, 2 and 3.	• Unit specification. • Assessment Workbook. • Computers with IDE installed. • Test plan template.

Lesson plan

Qualification	Pearson BTEC Uzbekistan Level 4 Qualifications in Software Development
Unit	Unit 1: Introduction to Programming
Lesson number	1 (2 hours)

Lesson objectives	At the end of the lesson, learners will: • understand the need to apply problem-solving skills to analyse problems • be able to identify and describe the problem • be able to break a problem down into smaller parts • be able to describe the main features of a problem or process.

Resources checklist	• Unit specification. • Flipchart or similar for learners to record discussions and ideas. • Presentation. • Example problem scenario for learners to examine.
Key: **AS**: Activity Sheet; **TF**: Template Form; **PS**: Presentation Slide	

Activities	Teaching notes
Starter activity (15 minutes)	• Introduce the unit and the main topics that will be covered. • Explain the importance of problem-solving skills in programming. • Explain that learners will need to apply problem-solving skills in all the units they study throughout this qualification.

Teacher-led demonstration
 	
(20 minutes)
	• Deliver a presentation to explain decomposition.
	• Demonstrate how decomposition is applied using an example computer problem. For example, this could be a computer program used by a hotel to manage room bookings.
	• Decomposition should focus on:
	  ○ identifying and describing the problem
	  ○ breaking a problem down into smaller parts
	  ○ describing the main features of a problem or process.
	• Use question and answer techniques throughout this demonstration to get involved in the learners' decomposition process as applied to the example.

Small-group/paired activity
 	
(45 minutes)
	• Learners practise decomposition using the following scenario.
	  ○ You have been asked to create a computer game similar to the classic game Tetris®.
	  ○ Decompose the problem. Make sure that you:
	  - identify and describe the problem
	  - break the problem down into smaller parts
	  - describe the main features of the problem or process.
	• Learners record their responses using a computer or on paper.

Teacher-led discussion (20 minutes)	• Give learners the opportunity to offer feedback on their work to the rest of the group. • Allow learners to discuss how their work differed from that of other learners.

Activities	Teaching notes
Concluding activity (20 minutes)	• Give learners an opportunity to improve their decomposition. Learners should take ideas that they developed during the discussion and use them to add to or refine their work.

Lesson plan

Qualification	Pearson BTEC Uzbekistan Level 4 Qualifications in Software Development
Unit	Unit 1: Introduction to Programming
Lesson number	2 (2 hours)

Lesson objectives	At the end of the lesson, learners will: • understand the need to apply problem-solving skills to analyse problems • be able to identify common elements or features of processes or problems • be able to identify differences between processes or problems • be able to identify individual elements within processes or problems • be able to describe patterns that have been identified • be able to make predictions based on identified patterns.

Resources checklist	• Unit specification. • Flipchart or similar for learners to record discussions and ideas. • Presentation. • Work from previous lesson. • Example problem scenario from previous lesson.
Key: **AS**: Activity Sheet; **TF**: Template Form; **PS**: Presentation Slide	

Activities	Teaching notes

Starter activity
 	
(10 minutes)
	• Recap the concept of decomposition with a question and answer session.
	• Explain that, in this lesson, learners will continue to explore problem-solving skills, particularly:
	  ○ pattern recognition
	  ○ pattern generalisation
	  ○ abstraction.

Activities	Teaching notes

Teacher-led demonstration
 	
(20 minutes)
	• Deliver a presentation to explain pattern recognition, pattern generalisation and abstraction.
	• Demonstrate how these skills are applied using an example computer problem. Use the same example as you used in the demonstration in Lesson 1, such as a computer program used by a hotel to manage room bookings.
	• The demonstration of pattern recognition should show learners how to:
	  ○ identify common elements or features of processes or problems
	  ○ identify differences between processes or problems
	  ○ identify individual elements within processes or problems
	  ○ describe patterns that have been identified
	  ○ make predictions based on identified patterns.
	• The demonstration of pattern generalisation and abstraction should show learners how to:
	  ○ identify information that is needed to solve an identified problem
	  ○ identify and remove information that is not needed to solve an identified problem
	  ○ describe parts of a problem or program in general terms by identifying:
	  - what needs to be input
	  - what the expected outputs are
	  - things that will vary
	  - things that will remain constant
	  - key actions that the program must perform
	  - repeated processes that the program will perform.
	• Use question and answer techniques throughout these demonstrations to get learners involved in the decomposition process of the example.

Activities	Teaching notes
Small-group/paired activity (50 minutes)	• Learners practise pattern recognition, pattern generalisation and abstraction, and expand on their work from Lesson 1, using the same given scenario as in Lesson 1. o You have been asked to create a computer game. o Decompose the problem. Make sure that you: - identify and describe the problem - break a problem down into smaller parts - describe the main features of a problem or process. • Learners can record their responses using a computer or on paper.
Teacher-led discussion (20 minutes)	• Give learners the opportunity to offer feedback on their work to the rest of the group. • Allow learners to discuss how their work differed from that of other learners.
Concluding activity (20 minutes)	• Give learners an opportunity to improve their decomposition. Learners should take ideas that they developed during the discussion and use them to add to or refine their work.

Lesson plan

Qualification	Pearson BTEC Uzbekistan Level 4 Qualifications in Software Development
Unit	Unit 1: Introduction to Programming
Lesson number	3 (2 hours)

Lesson objectives	At the end of the lesson, learners will: • understand the need to apply problem-solving skills to analyse problems • be able to describe processes required to solve a problem • be able to describe how sequence, selection and iteration are used in a given problem.

Resources checklist	• Unit specification. • Scenario and learner work from Lessons 1 and 2. • Presentation.
Key: **AS**: Activity Sheet; **TF**: Template Form; **PS**: Presentation Slide	

Activities	Teaching notes

Starter activity
(10 minutes)
	• Recap the problem-solving skills covered so far.
	• Explain that, in this lesson, learners will continue to explore problem-solving skills and focus on part of algorithm design, specifically:
	  ○ sequence
	  ○ selection
	  ○ iteration.
	• Use examples from the scenario used in the demonstrations in Lessons 1 and 2 to illustrate the points that you make.

Teacher-led discussion (10 minutes)	• Use a question and answer session to check learners' understanding of relational operators (=, <, >, <>, ≤, ≥). • Explain that learners will consider sequence first of all.

Small-group/paired activity
(30 minutes)
	• Using the same scenario as in Lessons 1 and 2, learners describe the sequence of key tasks and processes. They should identify:
	  ○ the most efficient and logical order for processes
	  ○ the order of operations required in calculations
	  ○ processes that are dependent on the output from other processes or calculations.

Teacher-led demonstration (15 minutes)	• Deliver a presentation that introduces the idea of selection (branching), If…Then…Else and Boolean operators (NOT, AND, OR). • Use a scenario to illustrate why selection (branching) is used (), such as different menu options or whether a player in a game has scored points or not. • examples of how branching and Boolean in computer programs can be represented using this method • The presentation should also cover the concept of iteration.

Activities	Teaching notes
Small-group/paired activity (45 minutes)	• Using the same scenario as used in Lessons 1 and 2, learners describe any branching and iteration that would be required in their program. They should cover: o the decisions that must need to be made by the user or program o the logical structure of a decision and alternative outcomes o processes or actions that will need to use iteration o the type of iteration that will be used (for example continuing to ask for a user input until the answer is correct).
Plenary activity (10 minutes)	• Confirm the main learning points identified in the lesson.

Lesson plan

Qualification	Pearson BTEC Uzbekistan Level 4 Qualifications in Software Development
Unit	Unit 1: Introduction to Programming
Lesson number	4–5 (4 hours)

Lesson objectives	At the end of the lesson, learners will: • understand the need to apply problem-solving skills to analyse problems • be able to identify and describe the inputs required • be able to describe the processes required to solve the problem • be able to identify and describe the outputs produced by a process or program.

Resources checklist	• Unit specification. • Large sheets of paper. • Work from previous lesson. • Example problem scenario for learners to examine. • Differently-coloured pens (for annotation).
Key: **AS**: Activity Sheet; **TF**: Template Form; **PS**: Presentation Slide	

Activities	Teaching notes
Starter activity (15 minutes)	• Recap the concepts of decomposition, pattern recognition and pattern generalisation and abstraction through a question and answer session. • Explain that, during the next two lessons, learners will explore algorithm design in further depth.
Small-group/paired activity (2.5 hours)	• Give learners a new scenario to examine, such as a computerised check-out in a supermarket or a Space Invaders-type game. • In the same groups or pairs as in previous lessons, learners consider the 'algorithms' for different parts of the game (e.g. controlling a spaceship, firing or invader movements). They write these down on large sheets of paper. • Explain that, in the early design stages of a software solution, the 'algorithms' need to be only step-by-step instructions using everyday language and do not need to be given as full programming code. However, learners should describe a solution in as much detail as possible.
Group activity (peer review and feedback) (45 minutes)	• At different stages in the two lessons, ask groups to share their work with other small groups. Each group should give feedback on the algorithms produced. Feedback should focus on the completeness and accuracy of each group's algorithms.
Concluding activity (30 minutes)	• Give the groups a selection of differently-coloured pens and ask them to annotate their work when receiving feedback, and any changes they make as a result of this feedback.

Lesson plan

Qualification	Pearson BTEC Uzbekistan Level 4 Qualifications in Software Development
Unit	Unit 1: Introduction to Programming
Lesson number	6 (2 hours)

Lesson objectives	At the end of the lesson, learners will be able to: • to apply problem-solving skills to analyse problems and to identify solutions that can be developed into computer programs • describe the computing tasks or processes needed to solve problems.

Resources checklist	• Unit specification. • Assessment activity sheet containing brief and scenario. • Copies of the content for learning aim A (problem-solving skills help sheet).
Key: **AS**: Activity Sheet; **TF**: Template Form; **PS**: Presentation Slide	

Activities	Teaching notes
Starter activity (5 minutes)	• Organise learners so that they can complete assessment tasks independently.
Individual assessment activity (1 hour 55 minutes)	• Give learners a brief that describes a scenario that requires a computer program. The scenario should not have been used in any of the previous lessons. • The brief should be of sufficient scope to allow learners to demonstrate understanding of all points in learning aim A. • Learners produce a plan for how the problem could be solved. • Give learners a copy of the content for learning aim A, to help them cover all points. **Note:** teachers should use the outcomes of the assessment activity to monitor learners' progress so far. This is a formative assessment and does not count towards their final unit grade.

Lesson plan

Qualification	Pearson BTEC Uzbekistan Level 4 Qualifications in Software Development
Unit	Unit 1: Introduction to Programming
Lesson number	7 (2 hours)

Lesson objectives	At the end of the lesson, learners will understand the: • concepts of the procedural programming paradigm • benefits and drawbacks of using the procedural programming paradigm.

Resources checklist	• Unit specification. • Presentation. • Computers with internet access. • Textbooks.
Key: **AS**: Activity Sheet; **TF**: Template Form; **PS**: Presentation Slide	

Activities	Teaching notes
Starter activity (10 minutes)	• Explain to learners that they will now look at a specific programming paradigm and investigate the procedural programming paradigm. • Explain that, as part of their final assessment for this unit, learners will need to plan, produce and test a program that uses this paradigm.
Teacher presentation (10 minutes)	• Deliver a presentation that gives an overview of the features and characteristics of the procedural programming paradigm. • Give learners a basic introduction to the structure of procedural programming. Highlight sequence, selection and iteration where appropriate.
Individual activity (40 minutes)	• Learners research and make notes on the uses, benefits and drawbacks of procedural programming. • If time permits, learners find examples of different procedural programming languages (e.g. Python) and look at the benefits and drawbacks of these languages.
Small group activity (40 minutes)	• Reorganise learners into small groups to discuss their findings. • Learners discuss the notes they made and what they have learned. During the discussion, learners should add to and expand on their notes, including any important points that other learners make.
Plenary activity (20 minutes)	• Give learners the opportunity to share what they have learned with the rest of the group. Allow discussion on how much of what some learners found out matches what other learners have said.

Lesson plan

Qualification	Pearson BTEC Uzbekistan Level 4 Qualifications in Software Development
Unit	Unit 1: Introduction to Programming
Lesson number	8 (2 hours)

Lesson objectives	At the end of the lesson, learners will be able to:
	• declare constants and variables that use appropriate data types
	• select and apply mathematical operators to simple calculations
	• use general functions to:
	  ○ accept user input
	  ○ output messages to the screen
	• use string handling functions to:
	  ○ convert numerical values to strings
	  ○ convert strings to numerical values.

Resources checklist	• Unit specification. • Presentation. • Programming task sheets. • Computers with Integrated Development Environment (IDE) installed. • Official Python documentation: https://docs.python.org/3/. • Online programming tutorials, such as www.learnpython.org/ or www.w3schools.com/python/. • Quiz for plenary activity.
Key: **AS**: Activity Sheet; **TF**: Template Form; **PS**: Presentation Slide	

Activities	Teaching notes
Starter activity (10 minutes)	• Explain that in the next phase of lessons, learners will be introduced to the basics of programming. • Explain that the lessons will focus on the procedural paradigm and, in particular, the programming language Python.
Teacher presentation (20 minutes)	• Deliver a presentation to introduce some key points that will be covered in this lesson, including: o the features of the development environment (editing code, running code and so on) o terminology (including constants, variables, input, output and string integer) o basic Python commands that will be used.
Individual activity (1 hour 10 minutes)	• Give learners a series of basic programming tasks, including: o accepting user input o assigning a value to a variable o simple mathematical equations o simple concatenation o outputting results to the screen o converting between string and integer. • Examples of Python commands/functions to be used include: o input() o print() o int() o str().
Plenary activities (20 minutes)	• Facilitate a brief discussion about what learners found difficult or surprising about coding. For example, the fact that all inputs are considered to be strings, so numbers entered must be converted before they can be used in calculations. • Hold a quiz to test learners' understanding of the topics covered in the lesson.

Lesson plan

Qualification	Pearson BTEC Uzbekistan Level 4 Qualifications in Software Development
Unit	Unit 1: Introduction to Programming
Lesson number	9 (2 hours)

Lesson objectives	At the end of the lesson, learners will be able to: • use appropriate data structures to store and handle data (lists) • select and apply mathematical operators (relational and Boolean operators) • apply an understanding of selection when writing code.

Resources checklist	• Unit specification. • Examples of code. • Presentation. • Computers with IDE installed. • Official Python documentation: https://docs.python.org/3/. • Online programming tutorials, such as www.learnpython.org/ or www.w3schools.com/python/. • Programming task sheets.
Key: **AS**: Activity Sheet; **TF**: Template Form; **PS**: Presentation Slide	

Activities	Teaching notes
Starter activity (10 minutes)	• Recap the work covered in the previous lesson and check learners' understanding with examples of code.
Teacher-led discussion (15 minutes)	• Explore the concept of needing to store multiple values (lists of items) and the need to allow programs to use selection (branching). Refer back to Lesson 3 as required.
Teacher-led demonstration (10 minutes)	• Deliver a presentation on the skills to be used in this lesson, including: o how to create a list o how to output the contents of a list o how to create IF statements o the difference in Python between = and ==.
Individual activity (1 hour 10 minutes)	• Give learners a series of basic programming tasks to complete, including: o creating and using lists o outputting the content of a list o providing different process/outputs based on user input. • Examples of Python commands or functions to be taught: o IF o ELSE o ELSEIF (ELIF).
Concluding activity (15 minutes)	• Organise learners into pairs. Learners share with their partner some of the code they have created in the individual task. Learners give their partner feedback on good features and areas for improvement.

Lesson plan

Qualification	Pearson BTEC Uzbekistan Level 4 Qualifications in Software Development
Unit	Unit 1: Introduction to Programming
Lesson number	10 (2 hours)

Lesson objectives	At the end of the lesson, learners will be able to: • apply an understanding of iteration when writing code • use built-in functions to define a range • add comments to code to aid understanding.

Resources checklist	• Unit specification. • Computers with IDE installed. • Examples of code. • Presentation. • Programming task sheets. • Official Python documentation: https://docs.python.org/3/. • Online programming tutorials, such as www.learnpython.org/ or www.w3schools.com/python/.
Key: **AS**: Activity Sheet; **TF**: Template Form; **PS**: Presentation Slide	

Activities	Teaching notes
Starter activity (10 minutes)	• Recap the work covered in the previous lesson and check learners' understanding with examples of code.
Teacher-led discussion (10 minutes)	• Explore the concept of needing to repeat processes in a program. • Use a question and answer session to check learners' understanding of iteration as covered in Lesson 3.
Teacher-led demonstration (20 minutes)	• Deliver a presentation on the skills to be used in this lesson, including: o how to create a 'for' loop o how to create a 'while' loop o how to define a range o the difference in Python between = and ==
Individual activity (1 hour)	• Give learners a series of basic programming tasks to complete, including: o creating and using different loops o deciding which loop to use in a specific situation (for example, a 'while' loop or a 'or' loop) o defining a numerical range as a condition for a loop o combining selection (see Lesson 9) and iteration o using a loop to output the contents of a list (see Lesson 9). • Examples of Python commands/functions to be taught: o while o for o range()
Plenary activities (20 minutes)	• Demonstrate how to annotate code (code comments). Highlight the use of # to inform the code to ignore the comment on that line. • Explain the importance of using comments when writing larger pieces of code and how they help other developers understand the logic of your code. • Learners annotate the code that they produced in the individual task to explain how the code works.

Lesson plan

Qualification	Pearson BTEC Uzbekistan Level 4 Qualifications in Software Development
Unit	Unit 1: Introduction to Programming
Lesson number	11 (2 hours)

Lesson objectives	At the end of the lesson, learners will: • be able to apply an understanding of programming code to solve simple problems.

Resources checklist	• Unit specification. • Assessment activity sheet. • Computers with IDE installed. • Official Python documentation: https://docs.python.org/3/. • Online programming tutorials, for example www.learnpython.org/ or www.w3schools.com/python/
Key: **AS**: Activity Sheet; **TF**: Template Form; **PS**: Presentation Slide	

Activities	Teaching notes
Starter activity (5 minutes)	• Organise learners so that they can complete the assessment tasks independently.
Individual assessment activity (1 hour, 55 minutes)	• Give learners a series of programming challenges. The challenges should cover the content covered in Lessons 8–10. • The challenges should be small coding problems that get gradually more difficult throughout the assessment. • The challenges do not need to produce full working programs. Instead, they should focus on testing the programming skills taught in Lessons 8–10. **Note:** teachers should use the outcomes of the assessment activity to monitor learners' progress so far. This is formative assessment and does not count towards their final unit grade.

Lesson plan

Qualification	Pearson BTEC Uzbekistan Level 4 Qualifications in Software Development
Unit	Unit 1: Introduction to Programming
Lesson number	12 (2 hours)

Lesson objectives	At the end of the lesson, learners will: • understand the purpose of built-in functions and libraries • understand the need to make programs flexible and unpredictable • be able to use built-in functions to define, generate and use random numbers.

Resources checklist	• Unit specification. • Presentation. • Programming task sheets and challenge. • Computers with IDE installed. • Official Python documentation: https://docs.python.org/3/. • Online programming tutorials, for example: www.learnpython.org/ or www.w3schools.com/python/.
Key: **AS**: Activity Sheet; **TF**: Template Form; **PS**: Presentation Slide	

Activities	Teaching notes
Starter activity (10 minutes)	• Recap the work covered in Lesson 10. Pay particular attention to learners' understanding of using the range function. Explain that this lesson will look at some other built-in functions available in Python.
Teacher-led discussion (10 minutes)	• Explore the reasons why you might need to make programs flexible and unpredictable. • Use a question and answer session to encourage learners to consider situations where processes or actions may need to be random.
Teacher-led demonstration (20 minutes)	• Deliver a presentation on the skills to be used this lesson, including: o a recap on the concept of a numerical range o how to import functions and libraries o the use of the random function. • Explore how the computer generates a random number and how to use ranges to add restrictions to what is generated.
Individual activity (1 hour)	• Give learners a series of basic programming tasks: o creating and using random numbers o defining the numerical range as parameters for the random number o combining other skills (e.g. input, output, lists and selection) to make more complex programs. • Examples of Python commands/functions to be used: o import o random() o randint() o uniform() o sample(). • Learners attempt a programming challenge to use random numbers and other programming concepts to solve a problem.
Concluding activity (20 minutes)	• In pairs, learners discuss their work and give peer feedback to each other on the quality and effectiveness of the code they have created.

Lesson plan

Qualification	Pearson BTEC Uzbekistan Level 4 Qualifications in Software Development
Unit	Unit 1: Introduction to Programming
Lesson number	13 (2 hours)

Lesson objectives	At the end of the lesson, learners will: • understand the need to produce high-quality code • understand how local and global variables are used • be able to create and use functions appropriately.

Resources checklist	• Unit specification. • Presentation. • Programming task sheets. • Computers with IDE installed. • Official Python documentation: https://docs.python.org/3/. • Online programming tutorials, such as www.learnpython.org/ or www.w3schools.com/python/.
Key: **AS**: Activity Sheet; **TF**: Template Form; **PS**: Presentation Slide	

Activities	Teaching notes
Starter activity (10 minutes)	• Recap the previous lesson and remind learners of the use of built-in functions. • Explain that, so far, learners have used built-in functions, but that today they will look into how they can create their own functions.
Teacher-led demonstration (20 minutes)	• Deliver a presentation on the skills and understanding required for this lesson, including: o the need to create functions (e.g. reusability, efficiency, reducing risk of error) o the difference between local and global variables o defining and calling functions o parameters and parameter passing o returning values.
Individual activity (1 hour 10 minutes)	• Give learners a series of tasks requiring them to define, use and call functions. • Examples of Python commands/functions to be taught: o def o global.
Concluding activity (20 minutes)	• In pairs, learners discuss their work and give each other peer feedback on the quality and effectiveness of the code they have created.

Lesson plan

Qualification	Pearson BTEC Uzbekistan Level 4 Qualifications in Software Development
Unit	Unit 1: Introduction to Programming
Lesson number	14 (2 hours)

Lesson objectives	At the end of the lesson, the learners will: • be able to round a number to a specific number of decimal places • be able to truncate numbers using built-in functions • locate the highest or lowest value in a group of numbers.

Resources checklist	• Unit specification. • Presentation. • Computers with IDE installed. • Programming task sheets. • Official Python documentation: https://docs.python.org/3/. • Online programming tutorials, such as www.learnpython.org/ or www.w3schools.com/python/.
Key: **AS**: Activity Sheet; **TF**: Template Form; **PS**: Presentation Slide	

Activities	Teaching notes
Starter activity (15 minutes)	• Recap use of lists and the previous lesson and explain that today's lesson will explore other built-in functions. • Discuss the need to specify the output format of numbers. Use a question and answer session to get learners to provide examples of when you may need to output numbers to a specific number of decimal places or to round up or down.
Teacher-led demonstration (20 minutes)	• Deliver a presentation on the skills and understanding required for this lesson, including: o the need to round or truncate outputs o the round function and how to specify the specific number of decimal places required o how to import the 'math' library o how to use, and the differences between, math.trunc(), math.floor() and math.ceil() o how to use the max and min function.
Individual activity (1 hour 10 minutes)	• Give learners a series of programming tasks to cover the skills taught in this lesson, including: o rounding and truncating positive and negative numbers o selecting and using appropriate truncation and rounding functions to solve given problems o outputting maximum or minimum number in a list. • Examples of Python commands/functions to be used include: o import math o round() o math.trunc() o math.floor() o math.ceil() o max() o min().
Plenary activity (15 minutes)	• Hold a question and answer session to confirm learners' understanding of the content of this lesson.

Lesson plan

Qualification	Pearson BTEC Uzbekistan Level 4 Qualifications in Software Development
Unit	Unit 1: Introduction to Programming
Lesson number	15 (2 hours)

Lesson objectives	At the end of the lesson, learners will: • understand the need to check the format or contents of a string • be able to check if a string is written in upper case or lower case • be able to check if a string contains numerical values • be able to convert a string from upper case to lower case • be able to convert a string from lower case to upper case.

Resources checklist	• Unit specification. • Presentation. • Computers with IDE installed. • Programming task sheets. • Quiz for plenary activity. • Official Python documentation: https://docs.python.org/3/. • Online programming tutorials, such as www.learnpython.org/ or www.w3schools.com/python/.
Key: **AS**: Activity Sheet; **TF**: Template Form; **PS**: Presentation Slide	

Activities	Teaching notes
Starter activity (10 minutes)	• Recap Lesson 8 on converting from string to number. Check learner understanding from previous lesson. • Explain that today's lesson will continue to explore more built-in functions.
Teacher-led discussion (10 minutes)	• Explore the need to make code robust, including checking user input. • Use a question and answer session to get learners to make suggestions on situations where they may need to check the input from a user. In particular, encourage them to consider when they might need to check case or check if a number has been included.
Teacher-led demonstration (20 minutes)	• Deliver a presentation on the skills and understanding required for this lesson, including: o using built-in Python functions to check text or number o converting between lower case and upper case.
Individual activity (1 hour 10 minutes)	• Give learners a series of programming tasks to cover the skills taught in this lesson, including: o taking user input and saving to a variable o checking if the stored string contains numerical values o converting the stored string from upper case to lower case o converting the stored string from lower case to upper case o using selection to check the input and provide feedback to the user. • Examples of Python commands/functions to be used: o isupper() o islower() o upper() o lower() o isalpha().
Plenary activity (10 minutes)	• Learners complete a quiz on the topics they have covered in the lesson.

Lesson plan

Qualification	Pearson BTEC Uzbekistan Level 4 Qualifications in Software Development
Unit	Unit 1: Introduction to Programming
Lesson number	16 (2 hours)

Lesson objectives	At the end of the lesson, learners will: • understand the need to use multiple items of data in a program • be able to use a list/array in a program • be able to use a dictionary/map in a program.

Resources checklist	• Unit specification. • Presentation. • Computers with IDE installed. • Programming task sheets. • Official Python documentation: https://docs.python.org/3/. • Online programming tutorials, such as www.learnpython.org/ or www.w3schools.com/python/.
Key: **AS**: Activity Sheet; **TF**: Template Form; **PS**: Presentation Slide	

Activities	Teaching notes
Starter activity (10 minutes)	• Use a question and answer session to recap the basic use of lists/arrays and check learners' understanding of this topic (referring to Lesson 9 as needed). • Explain that in this lesson, learners will explore more uses of lists and related data structures.
Teacher-led demonstration (20 minutes)	• Deliver a presentation on: o how indexing is used to refer to the location of an item in a list/array o how to set up a list/array o how to add data to a list/array during run time using results of calculations and user input o appending data to a list/array o inserting data at a specific location o how to 'pop' data.
Individual activity (1 hour 10 minutes)	• Give learners a series of programming tasks to cover the skills taught in this lesson, including: o taking user input and adding it to a list/array o handling data in a program using a list/array o appending data to a list o inserting data at specific locations o removing data o 'popping' data. • Examples of Python commands/functions to be used: o index() o append() o insert() o remove() o count() o pop() o sort().
Concluding activity (10 minutes)	• Learners work in pairs to give feedback to each other on the quality and effectiveness of each other's code.

Lesson plan

Qualification	Pearson BTEC Uzbekistan Level 4 Qualifications in Software Development
Unit	Unit 1: Introduction to Programming
Lesson number	17 (2 hours)

Lesson objectives	At the end of the lesson, learners will: • understand the need to use multiple items of data in a program • be able to use a list/array in a program • be able to use a dictionary/map in a program • be able to use functions for handling data in a list/array.

Resources checklist	• Unit specification. • Presentations. • Computers with IDE installed. • Programming task sheets. • Official Python documentation: https://docs.python.org/3/. • Online programming tutorials, such as www.learnpython.org/ or www.w3schools.com/python/.
Key: **AS**: Activity Sheet; **TF**: Template Form; **PS**: Presentation Slide	

Activities	Teaching notes
Starter activity (10 minutes)	• Recap the previous lesson. • Hold a question and answer session to recap and check learners' understanding (referring to Lesson 16 materials as needed). • Explain that, in this lesson, learners will explore more uses of lists and related data structures.
Teacher-led demonstration (15 minutes)	• Deliver a presentation on the skills and understanding required for this lesson, including: o a recap of how indexing is used to refer to the location of an item in a list/array o checking if a specific character, word or value is in a list/array o returning the location of a specific character, word or value o sorting the items in the list/array into ascending or descending order.

Activities	Teaching notes
Individual activity (35 minutes)	• Give learners a series of programming tasks to cover the skills taught in this lesson. These should build on those used in the previous lesson and introduce additional levels of challenge, such as: o taking user input and adding it to a list/array o checking if a specific character, word or value is in a list/array o returning the location of a specific character, word or value o sorting the items in the list/array into ascending or descending order. • Examples of Python commands/functions to be used include: o index() o append() o sort() o for loops o in o not in o count().
Teacher-led demonstration (10 minutes)	• Deliver a presentation on the differences between a dictionary/map and list/array in Python and how to use these in code.
Individual activity (40 minutes)	• Give learners a series of programming tasks to cover the use of dictionaries.
Concluding activity (10 minutes)	• Organise learners into pairs. Learners work together to discuss their code and give one another feedback on its quality and effectiveness.

Lesson plan

Qualification	Pearson BTEC Uzbekistan Level 4 Qualifications in Software Development
Unit	Unit 1: Introduction to Programming
Lesson number	18 (2 hours)

Lesson objectives	At the end of the lesson, learners will: • understand the need to manipulate items within a string • be able to use functions to manipulate data stored as a string.

Resources checklist	• Unit specification. • Presentation. • Computers with IDE installed. • Programming task sheets. • Official Python documentation: https://docs.python.org/3/. • Online programming tutorials, such as www.learnpython.org/ or www.w3schools.com/python/.
Key: **AS**: Activity Sheet; **TF**: Template Form; **PS**: Presentation Slide	

Activities	Teaching notes
Starter activity (10 minutes)	• Recap learners' understanding of indexing (referring to materials used in Lessons 16 and 17 as needed). • Explain that, in this lesson, learners will see how the skills they learned when using indexing can also be used to manipulate strings.
Teacher-led demonstration (15 minutes)	• Deliver a presentation on the skills and understanding required for this lesson, including: o how each character in a string can be located/referenced using an index in the same way as an item in a list o how to slice a string into individual characters o how to check if a specific letter or word is in a string o how to output the location of a specific character within a string o how to check the length of a string.
Individual activity (1 hour 10 minutes)	• Give learners a series of programming tasks to cover the skills taught this lesson, including: o slicing a string into individual characters o locating characters at specific points within the string and checking if a specific letter or word is in a string o checking that the length of a string meets pre-defined requirements and outputting appropriate messages to users. • Examples of Python commands/functions to be used include: o split() o len() o 'for' loops o in o not in o count().

Activities	Teaching notes
Concluding activities (25 minutes)	• Working in pairs, learners swap one of their pieces of code from today's lesson with one of their partner's pieces of code. Learners attempt to make their partner's program crash. • Learners make notes on what they did and what happened. Explain that they will explore the issue of a crashing program in the next lesson.

Lesson plan

Qualification	Pearson BTEC Uzbekistan Level 4 Qualifications in Software Development
Unit	Unit 1: Introduction to Programming
Lesson number	19 (2 hours)

Lesson objectives	At the end of the lesson, learners will: • understand the need to build robust program code • be able to apply validation techniques to program code.

Resources checklist	• Unit specification. • Presentation. • Computers with IDE installed. • Programming task sheets. • Example of code requirements. • Code reference sheets (covering loops, string handling, indexing). • Official Python documentation: https://docs.python.org/3/ • Online programming tutorials, for example www.learnpython.org/ or www.w3schools.com/python/.
Key: **AS**: Activity Sheet; **TF**: Template Form; **PS**: Presentation Slide	

Activities	Teaching notes
Starter activity (10 minutes)	• Recap the concluding activity from Lesson 18. Ask learners to describe examples of things that caused their code to crash. They may wish to demonstrate their code.
Teacher-led discussion (10 minutes)	• Explore the importance of robust programs. Ask learners to suggest specific scenarios in which programs would need to validate user input. Ask them to explore problems that may occur if an important program crashes.
Teacher-led demonstration (15 minutes)	• Deliver a presentation on the skills and understanding required for this lesson, including: o recapping the use of selection and iteration o limiting the number of options accepted o accepting only values that are within a given range o accepting only input that meets set length criteria o accepting only data of a specific type (e.g. text only, number only) o accepting only data that meets specific format requirements (e.g. must contain an @ symbol').
Individual activity (30 minutes)	• Give learners a series of programming tasks to cover the skills taught in this lesson, including: o using built-in or custom functions to apply validation techniques to program code o using iteration to ensure that a program does not continue until acceptable input is received. • Examples of Python commands/functions to be used include: o 'for' loops o 'while' loops o If…Else o in o not in o len().

Activities	Teaching notes
Individual activity (40 minutes)	• Give learners the requirements for a section of code (including its validation requirements). For example, the requirements could be: o Produce the programming code for a simple pizza-ordering system. The system should allow users to select small, medium or large pizzas and the quantity of pizzas required. The program should calculate the cost of the order and must not crash if an incorrect option is entered.
Concluding activity (15 minutes)	• Working in pairs, learners swap one of their pieces of code from today's lesson with one of their partner's pieces of code. Learners attempt to make the program crash and then give one another peer feedback.

Lesson plan

Qualification	Pearson BTEC Uzbekistan Level 4 Qualifications in Software Development
Unit	Unit 1: Introduction to Programming
Lesson number	20 (2 hours)

Lesson objectives	At the end of the lesson, learners will: • understand the need to use files that are external to the program to store data • be able to open, read from and write to external files.

Resources checklist	• Unit specification. • Presentation. • Computers with IDE installed. • Programming task sheets. • Quiz for plenary activity. • Official Python documentation: https://docs.python.org/3/. • Online programming tutorials, such as www.learnpython.org/ or www.w3schools.com/python/.
Key: **AS**: Activity Sheet; **TF**: Template Form; **PS**: Presentation Slide	

Activities	Teaching notes
Starter activity (15 minutes)	• Recap use of data structures (array/list, dictionary). • Explain that in some instances, such as when you have a large amount of data, it is more efficient to store data outside the program code.
Teacher-led demonstration (20 minutes)	• Deliver a presentation on the skills and understanding required for this lesson, including: o opening an external file in different modes (e.g. read, write) o creating an external file for use by the program o reading from, and writing to, an external file.
Individual activity (1hour 10 minutes)	• Give learners a series of programming tasks to cover the skills taught in this lesson, including: o using built-in or custom functions to apply validation techniques to program code o using iteration to ensure a program does not continue until acceptable input is received. • Examples of Python commands/functions to be used include: o open() o write() o close() o read() o readline() o readlines() o line.split().
Plenary activity (15 minutes)	• Give learners a quiz to complete that covers the topics they have covered in this lesson.

Lesson plan

Qualification	Pearson BTEC Uzbekistan Level 4 Qualifications in Software Development
Unit	Unit 1: Introduction to Programming
Lesson number	21 (2 hours)

Lesson objectives	At the end of the lesson, learners will be able to: • apply problem-solving skills to analyse problems and to identify solutions that can be developed into computer programs • describe the computing tasks or processes needed to solve problems.

Resources checklist	• Unit specification. • Assessment activity sheet. • Computers with IDE installed. • Python coding help sheet or code and syntax reference guide.
Key: **AS**: Activity Sheet; **TF**: Template Form; **PS**: Presentation Slide	

Activities	Teaching notes
Starter activity (5 minutes)	• Organise learners so they can complete assessment tasks independently.
Individual assessment activity (1 hour 55 minutes)	• Give learners a series of programming challenges. • They should produce code to solve each of the given problems. The challenges should be small problems rather than problems that require large complicated programs. • Provide a sufficient number of challenges to allow learners to use a wide a range of the skills they have learned in Lessons 8–19. • Examples of challenge topics include: o two six-sided die simulator o currency converter o username and email checker o test score to grade convertor. **Note:** teachers should use the outcomes of the assessment activity to monitor learners' progress so far. This is formative assessment and does not count towards their final unit grade.

Lesson plan

Qualification	Pearson BTEC Uzbekistan Level 4 Qualifications in Software Development
Unit	Unit 1: Introduction to Programming
Lesson number	22 (2 hours)

Lesson objectives	At the end of the lesson, learners will: • understand the need to test code during development • understand the importance of iterative development when writing code to produce solutions • be able to write a test plan to check a program against requirements.

Resources checklist	• Unit specification. • Presentation. • Computers with IDE installed. • Partially completed program code and program requirements. • Test plan template. • Official Python documentation: https://docs.python.org/3/. • Online programming tutorials, such as www.learnpython.org/ or www.w3schools.com/python/.
Key: **AS**: Activity Sheet; **TF**: Template Form; **PS**: Presentation Slide	

Activities	Teaching notes
Starter activity (10 minutes)	• Recap key points from Lessons 18 and 19 on the need to ensure that programs are robust.
Teacher-led discussion (10 minutes)	• Explain that the validation covered in previous lessons is just one part of establishing robust programs. • Explore the importance of robust programs and the role played by testing. Use a question and answer session to explore the fact that they are likely to have already experienced some 'testing' when creating their code in previous lessons, such as when they have written a new piece of code and entered data to see if it works. This is testing!
Teacher-led demonstration (20 minutes)	• Deliver a presentation on the skills and understanding required for this lesson, including: o the importance of formally planning and recording the testing process o the purpose of a test plan and how it is used o different types of test data including valid, valid extreme, invalid, invalid extreme and erroneous o the importance of checking functionality, robustness and the accuracy of outputs/calculations.
Individual activity (1 hour)	• Give learners a program requirements specification and a piece of code. The code should meet some but not all of the requirements. The program code, when run, should be functional (meaning that it will run), however, it may not be fully robust (for example not all inputs will be validated so it may produce incorrect outputs or crash if some unexpected data is input). • Give learners a test plan template. Learners create a detailed test plan for the code to check the extent to which it meets the given specification.
Concluding activities (20 minutes)	• In pairs, learners discuss their test plans. They should be prepared to explain and justify their choices of tests and test data to their partner. • Give learners time to add additional tests to their test plan, based on discussion with their partner.

Lesson plan

Qualification	Pearson BTEC Uzbekistan Level 4 Qualifications in Software Development
Unit	Unit 1: Introduction to Programming
Lesson number	23 (2 hours)

Lesson objectives	At the end of the lesson, learners will: • understand the need to test code during development • understand the importance of iterative development when writing code to produce solutions • be able to systematically apply a test plan and associated test data.

Resources checklist	• Unit specification. • Presentation. • Computers with IDE installed. • Partially completed program code and program requirements from previous lesson. • Learners' test plans from previous lesson. • Official Python documentation: https://docs.python.org/3/. • Online programming tutorials, such as www.learnpython.org/ or www.w3schools.com/python/.
Key: **AS**: Activity Sheet; **TF**: Template Form; **PS**: Presentation Slide	

Activities	Teaching notes
Starter activity (10 minutes)	• Recap key points from the previous lesson. Use a question and answer session to recap learners' understanding of different types of test data and the importance of testing.
Teacher-led demonstration (20 minutes)	• Deliver a presentation on the skills and understanding required for this lesson, including: o testing code and recording outcomes o identifying errors or unexpected behaviour in the program o fixing or improving code o recording improvements and retesting the code.
Individual activity (1 hour 10 minutes)	• Give learners the program requirements specification and a piece of code they used in the previous lesson. • Learners apply their test plan to the given code, to identify errors and improve the code as required.
Concluding activity (20 minutes)	• Learners annotate the code to identify how it works and the changes they have made.

Lesson plan

Qualification	Pearson BTEC Uzbekistan Level 4 Qualifications in Software Development
Unit	Unit 1: Introduction to Programming
Lesson number	24 (2 hours)

Lesson objectives	At the end of the lesson, learners will be able to: • apply problem-solving skills to analyse problems and to identify solutions that can be developed into computer programs • describe the computing tasks or processes needed to solve problems.

Resources checklist	• Unit specification. • Presentation. • Computers with IDE installed. • Problem-solving brief. • Copies of the content for learning aim A • Problem-solving skills help sheet.
Key: **AS**: Activity Sheet; **TF**: Template Form; **PS**: Presentation Slide	

Activities	Teaching notes
Starter activity (10 minutes)	• Explain to learners that, over the course of the next four lessons, they will complete a task that will allow them to practise all that they have learned so far in this unit.
Teacher-led discussion (20 minutes)	• Deliver a presentation and use a question and answer session to remind learners of the skills they developed in learning aim A (Lessons 1–6), including: o decomposition o pattern recognition o pattern generalisation and abstraction o algorithm design, including techniques used to develop algorithms.
Individual activity (1hour 10 minutes)	• Give learners a single, medium-sized computing problem. The brief should be of sufficient scope to allow learners to demonstrate understanding of all points in learning aim A. • Learners produce a plan for solving the problem. • Give learners a copy of the content for learning aim A, to help them cover all points of the learning aim.
Concluding activity (20 minutes)	• Working in pairs, learners discuss their plans. They should be prepared to explain and justify their choices to their partner. • If time allows, learners provide peer feedback to one another and use this feedback to make changes to their plan.

Lesson plan

Qualification	Pearson BTEC Uzbekistan Level 4 Qualifications in Software Development
Unit	Unit 1: Introduction to Programming
Lesson number	25 (2 hours)

Lesson objectives	At the end of the lesson, learners will: • be able to select appropriate tests and test data to test the functionality of a solution.

Resources checklist	• Unit specification. • Learners' work from previous lesson. • Presentation. • Problem-solving brief. • Test plan template. • Computers with IDE installed.
Key: **AS**: Activity Sheet; **TF**: Template Form; **PS**: Presentation Slide	

Activities	Teaching notes
Starter activity (15 minutes)	• Recap the previous lesson and the focus for this set of lessons. Refer learners to the notes they made on their plans at the end of the previous lesson. Give learners some time to make any additional changes based on their peer feedback.
Teacher-led discussion (15 minutes)	• Deliver a presentation and use a question and answer session to remind learners of the skills they will use in this lesson, including: o the importance of iterative development o the need to test throughout the development of a program o the reasons why some testing should be planned before development starts o the need to build on and adapt the initial test plan and test data as the program develops.
Individual activity (30 minutes)	• Learners create a test plan for the program they will create. • Encourage learners to use the plan they created in the last lesson, along with the problem brief, to guide them as to what is required in their test plan.
Small-group/paired activity (20 minutes)	• In pairs, learners discuss their plans. They should be prepared to explain and justify their choices, and give their partner constructive feedback.
Individual task (40 minutes)	• Learners use their plan from the previous lesson to start to create a program. • Ensure that learners test their code as they develop it. • Learners should aim to use the tests they designed in their test plan. In addition, any testing they do now that was not in their plan should be added to the plan as they carry it out. • Remind learners to record the outcomes of testing and any change they make as a result of the testing.

Lesson plan

Qualification	Pearson BTEC Uzbekistan Level 4 Qualifications in Software Development
Unit	Unit 1: Introduction to Programming
Lesson number	26–27 (4 hours)

Lesson objectives	At the end of the lesson, learners will be able to: • use standard programming structures and accepted programming conventions to produce code to solve identified problems • apply an understanding of iterative development when writing code to produce solutions.

Resources checklist	• Unit specification. • Learners' work from previous lesson. • Problem-solving brief. • Test plan template. • Language and syntax reference guide. • Computers with IDE installed. • Official Python documentation: https://docs.python.org/3/. • Online programming tutorials, such as www.learnpython.org/ or www.w3schools.com/python/.
Key: **AS**: Activity Sheet; **TF**: Template Form; **PS**: Presentation Slide	

Activities	Teaching notes
Starter activity (10 minutes)	• Recap the previous lessons and the focus for this set of lessons. • Explain to learners that, in this lesson and in the next lesson, they will continue to develop their solution to the given problem. • Remind learners to test their code as they develop it.

Individual assessment activity
 	
(3 hours 50 minutes)
	• Learners continue to develop their program to solve the given problem, using reference guides as required.
	• Encourage learners to seek feedback from other members of the group at different stages of development.
	• Learners should ensure that they record the tests they carry out, the results of the tests and any changes made to their code as a result of the test outcomes.
	• Use the outcomes from Lessons 24–27 to identify individual learners' areas of strength and areas for improvement.
	• At the end of these two lessons, learners submit their work (in any suitable format). Evidence submitted should include:
	  ○ program plan
	  ○ test plan/test log
	  ○ program file
	  ○ program code.
	• Mark learners work to give learners detailed feedback on the strengths and weaknesses of their work.
	
	Note: this is not a final assessment and should not be used as evidence towards their final grade. Teachers should use the outcomes of the of the assessment activity to monitor learners' progress so far. This is formative assessment and does not count towards their final unit grade.

Lesson plan

Qualification	Pearson BTEC Uzbekistan Level 4 Qualifications in Software Development
Unit	Unit 1: Introduction to Programming
Lesson number	28–29 (4 hours)

Lesson objectives	At the end of the lesson, learners will be able to: • identify areas of strength and weakness in their subject knowledge and their skills • demonstrate independence and self-development.

Resources checklist	• Unit specification. • Teacher feedback on work from Lessons 24–27. • Learners' work from Lessons 24–27. • Practice programming exercises. • Computers with IDE installed. • Official Python documentation: https://docs.python.org/3/. • Online programming tutorials, such as www.learnpython.org/ or www.w3schools.com/python/.
Key: **AS**: Activity Sheet; **TF**: Template Form; **PS**: Presentation Slide	

Activities	Teaching notes
Starter activity (10 minutes)	• Give learners copies of teacher feedback for the work produced in Lessons 24–27.
Individual activity (30 minutes)	• Learners read through the teacher feedback. They then identify the areas on which they need to improve, as well as topic areas they need further assistance with, and skills they feel they need to practise more.
Individual/paired activity (3 hours 20 minutes)	• Learners work on a variety of tasks or tutorials to practise and improve any areas of weakness. • Allow learners to self-direct the activities they use. Activities may include: o using online programming tutorials o further developing the work that was assessed o paired discussions and support o revisiting programming tasks from earlier lessons.

Lesson plan

Qualification	Pearson BTEC Uzbekistan Level 4 Qualifications in Software Development
Unit	Unit 1: Introduction to Programming
Lesson number	30–32 (6 hours)

Lesson objectives	At the end of the lesson, learners will be able to: • plan a solution to an identified problem • produce program code to solve an identified problem • produce program code to solve an identified problem.

Resources checklist	• Unit specification. • Problem-solving brief. • Test plan template. • Language and syntax reference guide. • Computers with IDE installed. • Official Python documentation: https://docs.python.org/3/. • Online programming tutorials for example www.learnpython.org/ or www.w3schools.com/python/.
Key: **AS**: Activity Sheet; **TF**: Template Form; **PS**: Presentation Slide	

Activities	Teaching notes
Starter activity (15 minutes)	• Recap the process that learners followed in Lessons 24–27. • Organise learners so they can complete assessment tasks independently. • Allow learners a short time to read through the program brief and clarify any issues.
Individual assessment activity (5 hours 45 minutes)	• Learners plan, develop and test a computer program to solve the problem detailed in the given brief. • The problem should be of sufficient scope to allow learners to practise and demonstrate a range of the programming skills listed in the specification. • For this activity, you may wish to provide a problem brief that focuses on particular areas where learners need to develop their skills. This will give learners the opportunity to improve before the final assessment. **Note:** teachers should use the outcomes of the assessment activity to monitor learners' progress so far. This is formative assessment and does not count towards their final unit grade.

Lesson plan

Qualification	Pearson BTEC Uzbekistan Level 4 Qualifications in Software Development
Unit	Unit 1: Introduction to Programming
Lesson number	33–34 (4 hours)

Lesson objectives	At the end of the lesson, learners will be able to: • identify areas of strength and weakness in their subject knowledge and skills • demonstrate independence and self-development.

Resources checklist	• Unit specification. • Teacher feedback on work from Lessons 30–32. • Practice programming exercises. • Computers with IDE installed. • Official Python documentation: https://docs.python.org/3/. • Online programming tutorials, for example www.learnpython.org/ or www.w3schools.com/python/.
Key: **AS**: Activity Sheet; **TF**: Template Form; **PS**: Presentation Slide	

Activities	Teaching notes
Starter activity (10 minutes)	• Give learners marked copies of their teacher feedback for the work produced in Lessons 30–32.
Individual activity (30 minutes)	• Learners read through the teacher feedback. They then identify areas on which they need to improve, as well as topic areas with which they need further assistance and the skills they feel they need to practise more.
Individual activity (3 hours 20 minutes)	• Learners work on a variety of tasks and tutorials to practise and improve any areas of weakness. • Allow learners to self-direct the activities they use. Activities may include: o using online programming tutorials o further developing the work that was assessed o paired discussions and support o revisiting programming tasks from earlier lessons.

Lesson plan

Qualification	Pearson BTEC Uzbekistan Level 4 Qualifications in Software Development
Unit	Unit 1: Introduction to Programming
Lesson number	35–39 (10 hours)

Lesson objectives	At the end of the lesson, learners will: • be able to complete a practice assignment.

Resources checklist	• Practice assessment activity. • Test plan template. • Language and syntax reference guide. • Computers with IDE installed.
Key: **AS**: Activity Sheet; **TF**: Template Form; **PS**: Presentation Slide	

Activities	**Teaching notes**
Starter activity (15 minutes)	• Explain to learners that, over the course of the next five lessons, they will complete a practice assignment to prepare them for their final assessment. • Explain that all work for this assignment must be their own and that they are not permitted to work in pairs.
Individual assessment activity (9 hours 45 minutes)	• Learners respond to a practice assignment brief similar to the final assessment. **Note:** the activity used should follow a similar structure to the final assessment and should be graded against the same criteria. However, the assessment activity used **must not** be the one used in the Assessment Workbook.

Lesson plan

Qualification	Pearson BTEC Uzbekistan Level 4 Qualifications in Software Development
Unit	Unit 1: Introduction to Programming
Lesson number	40 (2 hours)

Lesson objectives	At the end of the lesson, learners will be able to: • identify areas of strength and weakness in their subject knowledge and skills • demonstrate independence and self-development.

Resources checklist	• Unit specification. • Learners' work from Lessons 35–39. • Teacher feedback on practice assignment from Lessons 35–39. • Practice programming exercises. • Computers with IDE installed • Official Python documentation: https://docs.python.org/3/. • Online programming tutorials, such as www.learnpython.org/ or www.w3schools.com/python/.
Key: **AS**: Activity Sheet; **TF**: Template Form; **PS**: Presentation Slide	

Activities	Teaching notes
Starter activity (10 minutes)	• Give learners marked copies of their practice assignment.
Individual activity (30 minutes)	• Learners read through their teacher feedback. They then identify areas on which they need to improve, as well as topic areas with which they need further assistance, and the skills they feel they need to practise more.
Individual activity (1 hour 20 minutes)	• Learners work on a variety of tasks and tutorials to practise and improve any areas of weakness. • Allow learners to self-direct the activities they use. Activities may include: o using online programming tutorials o further developing the work that was assessed o paired discussions and support o revisiting programming tasks from earlier lessons.

Lesson plan

Qualification	Pearson BTEC Uzbekistan Level 4 Qualifications in Software Development
Unit	Unit 1: Introduction to Programming
Lesson number	41–45 (10 hours)

Lesson objectives	Assessment of learning aims A, B and C

Resources checklist	• Unit specification. • Assessment Workbook. • Computers with IDE installed. • Test plan template.
Key: **AS**: Activity Sheet; **TF**: Template Form; **PS**: Presentation Slide	

Activities	**Teaching notes**
Starter activity (15 minutes)	• Introduce the assignment (see Assessment Workbook), detailing the main assessment criteria, the deadline and the submission criteria.
Individual assessment activity (9 hours 45 minutes)	• Using the Assessment Workbook, learners complete Unit 1: Tasks 1, 2 and 3. • Learners must work independently to complete this task. • Learners must complete assessment activities only under supervision.

Rr301219 ENG_9781446961339_UZBEK_TRC1BK1_SOFTWARE_UNIT1_RR.DOCX.1-106/0

For information about Pearson Qualifications, including Pearson Edexcel, BTEC and LCCI qualifications visit qualifications.pearson.com

Pearson Education Limited. Registered in England and Wales No. 872828
Registered Office: 80 Strand, London WC2R 0RL.

VAT Reg No GB 278 537121

ISBN 978-1-4469-6133-9