Topic 2.5 Programming languages and Integrated Development Environments – Lesson 1

FIRST ASSESSMENT SUMMER 2022

The big picture

Why is this relevant for the students?

- Students to produce a graphic organiser (as a circle) split down the middle. The left side to fill with how code needs to be written for a Computer to understand. The right side to fill with points on how code needs to be written to make it easier to write for the programmer.

This would set the big picture of the first lesson looking at the different generations of programming languages and the need for translators.

Notes: *Use Context Setting task to engage students and create discussion.*
May link to flipped resources if you use flipped learning.

Objectives

What should the students be confident/able to do at the end of the session?

- To be able to describe the different generations of programming language.
- To be able to describe the differences between low-level and high-level languages.
- To evaluate the benefits of programming in both low- and high-level languages.
- To state which translator is needed for each and why.

Notes: *These are the core learning that the students should develop during the lesson. This will link to the activities that provide ability to assess the Objectives.*

Engagement

What will make the students want to learn?

- Students given a section of Machine Code (pure Binary) – First Generation Programming. They have to spot the error in the code. What are the issues for the programmer?
- Thinking of a programming language that you are familiar with, what are all the pros of using this language as a programmer?

Notes: *A short activity that stimulates the students. Ideas taken from big picture activity could be used.*

Keywords

What exam/ specification specific words should the students be confident with and need to know?

- Low-level language
- High-level language
- First generation Language
- Language Translator
- Machine Code
- Assembly Language
- Second Generation Language
- Third generation Language
- Compiler
- Interpreter
- Device Driver
- Debug

Multiple Choice Questions will assess these keywords; use the MCQs supplied.

The sticking points

What do I want students to remember?

- First generation is Machine Code/Language – no translator is needed. It can be directly executed by the processor. It is however tedious to code and difficult to spot errors.
- Second generation is Assembly Language – a translator is needed to translate the language into Machine Code. Assembly language uses Mnemonics – it is easier to de-bug than first generation and faster to program than first.
- First and second generation Languages are low-level languages.
- Low-level languages are difficult to de-bug, difficult for anybody to understand the code, difficult to maintain, require lots of instructions to perform a specific task. They are useful for directly addressing hardware and for fast execution of a program. Suitable for Device Drivers.
- High-level languages are easier to understand (for the programmer), easier to de-bug, faster to write. They use English like key-words. One instruction in a high-level language would translate to many in a low-level language.
- Third generation languages include Pascal, VB, C#, Python.
- Third generation is high-level.
- Third generation requires Compiler to translate code.

Notes: *A list of concepts that you want the students to remember by the end of the lesson.*

Assessment for Learning

What am I looking for to show progress?

Expected progress

- Students can match sample pieces of code to low-/high-level languages.
- Translators can be matched to different generations of languages.

Good progress

- Students can accurately describe the differences between low-level and high-level languages.

Exceptional progress

- Advantages and disadvantages of programming in low-level and high-level languages can be evaluated.

Differentiation

How will I enable access to each area of learning?

- Some students will have more experience than others on using a variety of languages. In discussions on the pros and cons as a programmer in programming in different languages, students with wider experience should use examples beyond classroom taught programming.
- Link to tasks using the Little Man Computer (LMC): more able students will be able to give examples of code in Assembly Language and the equivalent code in a high-level language to model the idea of one instruction to many machine code instructions.
- First-hand experience at programming in both low-level and high-level languages will enable advantages and disadvantages to be seen more clearly.

Notes: *Use of stretch task ideas supplied may support high end differentiation.*

You will need to modify the resources to meet the needs of your students specifically. You may wish to refer to Departmental or School policies on differentiation methods used within your centre.

Activity 1

What tasks will I ask the students to complete to develop their understanding during the lesson?

- Match activity to link together examples of code from different types of languages along with the translators needed.

Notes: *Use the Activities given to develop the students' knowledge of the topic. Each activity may need further differentiation/adaptation for your needs.*

Reference the Common misconceptions/FAQ guide to support your delivery of the topic.

Activity 2

What tasks will I ask the students to complete to develop their understanding during the lesson?

- Use a graphics organiser to identify the advantages and disadvantages of programming using each generation of language, their uses and the translator required.
- Could use mandala Graphic Organiser (concentric circles split into thirds – inner circle to have generation, second circle Translator, third circle advantages/disadvantages).

Reference the Common misconceptions/FAQ guide to support your delivery of the topic.

Activity 3

What tasks will I ask the students to complete to develop their understanding during the lesson?

- Translation task from a pictorial language into English (literal translation in pictures) or a made up language into English.
- How is this like generations being translated from one language to another?
- Could give students one line of high-level language they are familiar with and translate to assembly language (LMC) and for the more able into machine code (Binary).

Summary/Plenary

How will I check that students have retained the knowledge?

- Pause-Pounce-Bounce method of questioning to ensure all students listen to each other's answers:
- Give an example of a generation of a language.
- Which translator is used?
- What would the code look like for that generation?
- Give an advantage? Why? Give a disadvantage.
- Why would a programmer prefer writing in a high-level language?
- What are the benefits of programming in a low-level language?
- All students to write three translators on a mini-whiteboard and link with generation – holding these up.

Notes: *Use the MCQs to check basic understanding of Keywords and Topics.*

Use the level of response (LOR) to develop deeper knowledge and allow Peer Assessment and Review. This can be developed to use the LOR ideas as homework etc.

Notes

Topic 2.5 Programming languages and Integrated Development Environments – Lesson 2

The big picture

Why is this relevant for the students?

Graphic Organiser for students to complete to weigh up the differences between high-level and low-level languages.

Notes: *Use Context Setting task to engage students and create discussion.*
May link to flipped resources if you use flipped learning.

Objectives

What should the students be confident/able to do at the end of the session?

- To be able to describe the differences in operation between a Compiler and Interpreter.

Notes: *These are the core learning that the students should develop during the lesson. This will link to the activities that provide ability to assess the Objectives.*

Engagement

What will make the students want to learn?

- Based on starter/flipped learning students to put one key point that they know about the topic on a post-it-note and one point that they feel they 'want to know'.

This can be used to check starting point and be used in plenary to check did students learn what they wanted to.

Notes: *A short activity that stimulates the students. Ideas taken from big picture activity could be used.*

Assessment for Learning

What am I looking for to show progress?

Expected progress

- To be able to identify which language translators are used for which generations of languages.
- To match up given features with either compiler or interpreter

Expected progress: This is likely to be activities and Learning tasks that meet your expectations for the class progress towards the objectives.

Good progress

- To be able to describe key features of Interpreters and Compilers.
- To be able to describe the importance of an IDE when producing programming source code.

Good progress: This would show a development from basic understanding and be indicative that some students use stretch and challenge material during the lesson.

Exceptional progress

- To be able to evaluate the reasons why a programmer would make use of all three language translators during the development of software.

Exceptional progress: This would indicate the level of progress if all extension activities have been completed and at 8/9 levels of understanding.

The sticking points

What do I want learners to remember?

- Compiler/Interpreter are used for third generation languages (high-level).
- Compiler converts high-level source code into Machine code.
- Compiler translates and converts entire source code all in one go.
- Compiler produces an error report and an executable file (Object code).
- Interpreter doesn't produce object code.
- Interpreter translates and executes lines of code statement by statement.
- Interpreter stops when it encounters an error.
- Interpreter is used during de-bugging stages and development.
- Compiler is used when program is ready for being shipped.
- Compiled code is optimised.
- Compiled code does not include the source code.

Notes: *A list of concepts that you want the students to remember by the end of the lesson.*

Keywords

What exam/specification specific words should the students be confident with and need to know?

- Compiler
- Interpreter
- Source Code
- Object Code (Executable Code)
- De-bug
- Translator
- IDE

Multiple Choice Questions will assess these keywords; use the MCQs supplied.

You may wish to customise these as needed.

Differentiation

How will I enable access to each area of learning?

- Producing a graphical representation of Compiler/Interpreter could be differentiated by having a different word limit for differing ability students.
- Differentiation by questioning.
- Differing ability students can be given a series of key points to include in their homework.

Notes*: Use of Stretch Task Ideas supplied may support high end differentiation.*

You will need to modify the resources to meet the needs of your students specifically. You may wish to refer to Departmental or School policies on differentiation methods used within your centre.

Activity 1

What tasks will I ask the students to complete to develop their understanding during the lesson?

- Similarities and differences between high- and low-level languages.
- Differences between the two types of translator.
- Could give students one line of high-level language they are familiar with and translate to assembly language (LMC) and for the more able into machine code (Binary).

Notes*: Use the Activities given to develop the students' knowledge of the topic. Each activity may need further differentiation/adaptation for your needs.*

Activity 2

What tasks will I ask the students to complete to develop their understanding during the lesson?

Students given a description of either an Interpreter or a Compiler. Using minimal amount of words, the students should draw a graphical representation of the text that they have been given.

Students then pair up and using their picture only – explain to their partner how Compiler/Interpreter works. Partner then relays back to the student the notes they have gathered, and this is compared to the original text.

Notes*: Use the Activities given to develop students' knowledge of the topic. Each activity may need further differentiation/adaptation for your needs.*

Activity 2

What tasks will I ask the students to complete to develop their understanding during the lesson?

Students given the table to complete.
Students write a summarised description for compiler and interpreter.
Differentiated students can have a table with hints.
Discuss students' findings and address any misconceptions that may have arisen.

Notes*: Use the Activities given to develop students' knowledge of the topic. Each activity may need further differentiation/adaptation for your needs.*

Summary/Plenary

How will I check that students have retained the knowledge?

A series of multiple choice questions given based on the MCQs – students use mini-whiteboards to answer/vote.

Homework to produce a short report describing the differences between the language translators, how these apply to generations of languages and where each would be used. (This could be marked as a LOR question.)

Notes: *Use the MCQs to check basic understanding of Keywords and Topics.*

Use the LOR to develop deeper knowledge and allow Peer Assessment and Review. This can be developed to use the LOR ideas as homework etc.

Homework/flipped learning

Topic 2.5 Programming languages and Integrated Development Environments – Lesson 3

The big picture

Why is this relevant for the students?

Graphic Organiser for students to complete to weigh up the features of IDEs and why programmers would want these features

Notes: *Use Context Setting task to engage students and create discussion.*

May link to flipped resources if you use flipped learning.

Objectives

What should the students be confident/able to do at the end of the session?

- To be able to describe the common tools and facilities in an Integrated Development Environment (IDE).

Notes: These are the core learning that the students should develop during the lesson. This will link to the activities that provide ability to assess the Objectives.

Engagement

What will make the students want to learn?

- Based on starter/flipped learning, students to put one key point that they know about the topic on a sticky note and one point that they feel they 'want to know'.
- This can be used to check starting point and be used in plenary to check did students learn what they wanted to.

Notes: *A short activity that stimulates the students. Ideas taken from big picture activity could be used.*

Assessment for Learning

What am I looking for to show progress?

Expected progress

- To be able to identify features of an IDE.

Expected progress: This is likely to be activities and Learning tasks that meet your expectations for the class progress towards the objectives.

Good progress

- To be able to describe the importance of an IDE when producing programming source code.

Good progress: This would show a development from basic understanding and be indicative that some students use stretch and challenge material during the lesson.

Exceptional progress

- To be able to evaluate the reasons why a programmer would make use of IDEs and have experience of more than one IDE.

Exceptional progress: This would indicate the level of progress if all extension activities have been completed and at 8/9 levels of understanding.

The sticking points

What do I want students to remember?

- IDE contains code editor, error diagnostics, run-time environment and has built in translators.
- Features available to programmers to help with human error and speed the programming process.

A list of concepts that you want students to remember by the end of the lesson.

Keywords

What exam/specification specific words should the students be confident with and need to know?

- Compiler
- Interpreter
- Source Code
- Object Code (Executable Code)
- De-bug
- Translator
- IDE

Multiple Choice Questions will assess these keywords; use the MCQs supplied.

You may wish to customise these as needed.

Differentiation

How will I enable access to each area of learning?

- Producing a graphical representation of the IDE could be differentiated by having a keywords available for differing ability students.
- Differentiation by questioning.
- Differing ability students can be given a series of key points to include in their homework.

Notes: *Use of stretch task ideas supplied may support high end differentiation.*

You will need to modify the resources to meet the needs of your students specifically. You may wish to refer to Departmental or School policies on differentiation methods used within your centre.

Activity 1

What tasks will I ask the students to complete to develop their understanding during the lesson?

Students are to find common features for an IDE one half of the circle for the feature on the other side why a programmer many want/need this to program.

Students then pair up and using their picture only – explain to their partner the features and reason why. Partner then re-lays back to the student the notes they have gathered, and this is compared to the rest of the class.

Notes*: Use the Activities given to develop the students' knowledge of the topic. Each activity may need further differentiation/adaptation for your needs.*

Reference the Common misconceptions/FAQ guide to support your delivery of the topic.

Activity 2

What tasks will I ask the students to complete to develop their understanding during the lesson?

Students are given a picture of an IDE and they have to identify key features.
To differentiate students could have key words at the bottom to assist them.
Higher ability students could have a different IDE similarly lower ability students could have a simpler IDE (Pythons default IDLE.
You could get students to compare VS IDE to Pythons default IDLE.

Notes*: Use the Activities given to develop the students' knowledge of the topic. Each activity may need further differentiation/adaptation for your needs.*
Reference the Common misconceptions/FAQ guide to support your delivery of the topic.

Activity 3

What tasks will I ask the students to complete to develop their understanding during the lesson?
To open up the programming language IDE that the students are familiar with and produce screen/shots set of notes that identify where:
- editor is located
- error diagnostics can be found and used
- run-time environment is found
- example of using Compiler/Interpreter.

Notes*: Use the Activities given to develop the students' knowledge of the topic. Each activity may need further differentiation/adaptation for your needs.*
Reference the Common misconceptions/FAQ guide to support your delivery of the topic.

Summary/Plenary

How will I check that students have retained the knowledge?

A series of multiple choice questions given based on the MCQs – students use mini-whiteboards to answer/vote.

Homework: Research as many different IDEs as you can find, compare their features, make a table to illustrate their differences. Could be used as a LOR.

Notes: *Use the MCQs to check basic understanding of Keywords and Topics.*

Use the LOR to develop deeper knowledge and allow Peer Assessment and Review. This can be developed to use the LOR ideas as homework etc.

Homework/flipped learning

We value your feedback

We'd like to know your view on the resources we produce. By clicking on the icon above you will help us to ensure that our resources work for you.

Whether you already offer OCR qualifications, are new to OCR, or are considering switching from your current provider/awarding organisation, you can request more information by completing the Expression of Interest form which can be found here: www.ocr.org.uk/expression-of-interest

Looking for a resource? There is now a quick and easy search tool to help find free resources for your qualification: www.ocr.org.uk/i-want-to/find-resources/