Assessment Prep with Kognity
IBDP Computer Science

What is this guide for?

This guide helps teachers use Kognity to support formative assessment, targeted feedback, practical programming development and preparation for IBDP Computer Science assessment.

How does Kognity help with assessment preparation for IBDP Computer Science?

Formative assessment provides regular evidence of student understanding throughout the course. Kognity combines immediate student feedback with timely student data, making it easier to identify misconceptions, plan intervention and gradually increase the demands of assessment practice.

Feedback principle

"Think of feedback as received, not given."

Use student responses and progress data to decide what learners understand, what they misunderstand and what teaching action should follow.

For students:

Students can attempt worked examples, reveal solutions, complete section questions and use the Practice Centre to test their understanding. The notebook and strength indicators help them organise feedback and choose what to revise next.

For teachers:

The Assignments and Insights areas provide evidence from textbook questions, quizzes, strength tests and question assignments. Teachers can identify individual needs and class-wide patterns, then respond with targeted reteaching or extension.

Table of contents

How does Kognity help with formative assessment?

Revisions Quizzes

Use question assignments at the end of a subtopic or unit to check knowledge and application. Assign a focused set of questions so the results identify a specific area for action.

- Use auto-graded results for immediate feedback.
- Review common misconceptions before introducing new content.
- Reassign a similar question after reteaching to check whether understanding has improved.

Exit tickets

Set section questions at the end of a lesson, or ask students to answer one short conceptual, tracing or evaluation question before they leave.

- Use one question that matches the lesson objective.
- Scan responses for patterns rather than treating every error as an individual issue.
- Begin the next lesson by addressing the most significant misconception.

Notebook and self-assessment

The notebook allows students to collect definitions, code annotations, diagrams, feedback and personal examples in one place. Use it to turn feedback into a specific revision action.

- Write a one-sentence explanation for each learning objective.
- Record recurring syntax, tracing or command-term errors.
- Compare an initial answer with the model or markscheme and explain the difference.

How does Kognity help with formative assessments?

Exam-style practice tasks

Use exam-style questions, solutions and markschemes in a range of classroom formats.

- Plan an answer together before students write independently.
- Hide the markscheme for timed practice, then release it for self-marking and reflection.
- Compare two anonymised responses and identify which communicates Computer Science more precisely.
- Focus on command terms, technical vocabulary, code tracing and application to unfamiliar contexts.

How does Kognity help with IB Assessment Preparation?

Internal Assessment skills

Kognity activities can be used to rehearse the skills students need before and during the Internal Assessment. The goal is to build these habits through normal teaching rather than introducing them for the first time when the project begins.

From problem to measurable criteria

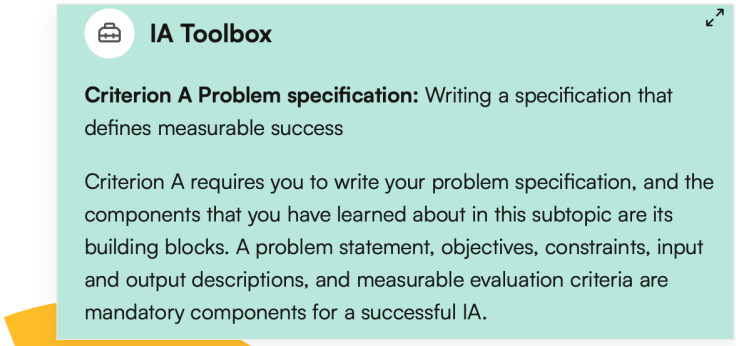
For example, use B1.1.1 content to practise translating a user need into a clear problem statement, objectives, constraints, inputs, outputs and objective evaluation criteria. Use the provided IA toolboxes to scaffold the long IA process.

- During the unit, require criteria that can be tested with observable evidence.
- Ask students to identify technical, legal, time, hardware and human constraints.
- Use peer review to identify missing requirements or vague wording.

Design, implementation and evaluation

Use programming and OOP activities to practise decomposition, modelling, implementation choices, testing and evaluation.

- Throughout the course, ask students to justify why a class, data structure or algorithm is appropriate.
- During the IA process, have students compare an initial design with the implemented solution and explain changes.
- After creating a solution, evaluate limitations and improvements using evidence from tests and user needs.



SCREENSHOT EXAMPLE
B1.1.1 problem specification activity

Practical programming and the assessment cycle

Practical programming preparation

Connect concepts to code

Assessment preparation should include regular opportunities to read, trace, debug, construct and evaluate code. The Java and Python examples in the book allow students to focus on the underlying concept and then recognise language-specific syntax.

- Trace variable values and method calls before running code.
- Predict errors caused by access modifiers, missing methods or inappropriate design choices.
- Compare an if/else solution with a polymorphic design and discuss maintainability.
- Write short code fragments under time limits, then test and annotate corrections.

A simple assessment cycle

1. Diagnose	2. Respond	3. Recheck	4. Reflect
Use a section question, strength test or short assignment.	Reteach, model or provide targeted practice based on the evidence.	Use a similar question to see whether the misconception has changed.	Students record the feedback and choose the next revision action.