BRIDGE TO TEXT

A 12-Week Framework & Curriculum Map · Ages 13–15

Awaken Innovation Lab · the step between Padma and iGCSE

Version 1.0 · 12 weeks × 2 sessions · ~48 contact hours **Entry:** completion of Padma Terms 1–3 · **Exit:** ready to *begin* iGCSE Computer Science

---

1. What this course is, in one paragraph

A student who finishes Padma can program. They cannot read code. This course fixes exactly that gap — it converts three terms of block-based competence into text-based competence, and gives them enough of the formal theory to recognise the iGCSE syllabus when they meet it.

It is not iGCSE. It is the twelve weeks that make iGCSE survivable.

---

2. The four gaps this exists to close

A parent will ask why they need another year. This table is the answer.

GAP	WHAT PADMA GAVE THEM	WHAT IGCSE ASKS	WEEKS HERE
Notation	Concepts, in blocks	The same concepts, in Python	1–6
Error messages	A block won't fit	`TypeError: can only concatenate str (not "int") to str`	8
Formal theory	Sensors, control loops	Binary, hex, networks, security	10–11
Exam technique	Building things	Tracing code on paper, writing pseudocode	9, 12

Notice the first row. **The concepts are not the gap.** Sequencing, selection, iteration, variables, functions — they have all of it. Only the notation is new.

That is the single most important thing to tell a nervous student on day one.

---

3. The design principle

> **"You already know how to program. You are learning a new language for something you already think in."**

Every lesson is built on a **port**: take a program they wrote in Padma, rewrite it in Python, run it, compare. Nothing is taught from scratch except lists, and lists exist because text has no sprites.

Why porting beats fresh teaching:

- The student's own code is already correct, so any failure is *notation*, not *thinking* — which is diagnosable.
- It makes the transfer visible. They can see that `repeat 4` and `for i in range(4):` are the same idea wearing different clothes.
- It gives the teacher a known-good reference. If the Python version misbehaves, the teacher knows the logic was right.

---

4. The transfer table

Print this. Put it on the wall for the first six weeks. It is the course.

PADMA	PYTHON	WHAT CHANGED
`when ⚑ clicked`	the first line of the file	Nothing runs until you run the file
`repeat 4`	`for i in range(4):`	A count becomes a range
`forever`	`while True:`	A loop with no exit — and no sprite to escape
`repeat until <condition>`	`while not condition:`	The condition is written, not clicked
`if <> then … else`	`if … : / elif … : / else … :`	Indentation replaces the C-shape
`set x to 5`	`x = 5`	Assignment is a single `=`
`change x by 1`	`x += 1`	Shorthand, same meaning
`and` / `or` / `not`	`and` / `or` / `not`	Identical
`define scan (angle)`	`def scan(angle):`	A definition becomes a function
calling a custom block	`scan(90)`	Parentheses instead of a snap
`broadcast [msg]`	a function call	Messages become calls

PADMA	PYTHON	WHAT CHANGED
`say [text]`	`print("text")`	Output goes to the console
`ask and wait`	`input()`	Input is a function that returns a string

The three things that are genuinely new:

1. **Indentation is meaning.** Python has no `end`. The whitespace *is* the structure.
2. **Types exist.** `"5" + 5` is an error, not a surprise.
3. **You run the whole file.** There is no "press the green flag and click a sprite."

Everything else is a rename.

---

5. Course structure

Twelve weeks, three phases. Two sessions per week, 40 minutes each.

PHASE	WEEKS	FOCUS	WHY
I. The language	1–6	Syntax, ported from Padma	Mechanical; must be automatic before anything else
II. The practice	7–9	Porting a system, debugging text, algorithms	Where competence becomes fluency
III. The theory	10–12	Data representation, networks, mock exam	The half of iGCSE that Padma never touched

Phase I — The language

WK	LESSON	PORTED FROM	NEW IN PYTHON
1	Blocks become text	First script	The editor, `print()`, running a file, indentation
2	Variables and input	Power grid	`=`, `int()`, `input()`, f-strings
3	Selection	Airlock	`if` / `elif` / `else`, comparison, boolean operators
4	Iteration	Loop ladder	`for`, `range()`, `while`, nested loops
5	Functions	The routine library	`def`, parameters, `return`
6	Lists	*(new — no Padma equivalent)*	`[]`, index, `append`, `len`, looping a list

Phase II — The practice

WK	LESSON	FOCUS
7	Port the follower	Take the four-state logic to Python as a simulation
8	Read the error	Tracebacks, exception types, the six-step protocol in text
9	Algorithms	Linear search, bubble sort, writing pseudocode

Phase III — The theory

WK	LESSON	FOCUS
10	Data representation	Binary, hex, denary conversion, characters, sound and images
11	Networks and security	Topologies, protocols, threats, and what a firewall actually does
12	Mock and practical	A timed theory paper and a timed programming task

---

6. Descriptor set

Eighteen competences, five strands. Same convention as the Padma map: **our codes, replaceable with official ones.**

Programming language (PL)

CODE	STATEMENT
PL1	Write and run a Python program that produces specified output
PL2	Declare, assign and update variables of different types
PL3	Write selection using `if`, `elif`, `else`, including compound conditions
PL4	Write iteration using `for`, `while` and `range`, including nesting
PL5	Define and call functions with parameters and return values
PL6	Create and manipulate a list

Transfer (TR)

CODE	STATEMENT
TR1	Explain how a block construct maps to its text equivalent
TR2	Port a working block program to Python so it behaves identically
TR3	Adapt a ported program to a changed requirement

Text debugging (DG)

CODE	STATEMENT
DG1	Read a traceback and identify the line, the type and the cause
DG2	Isolate a fault in text using one change at a time
DG3	Fix a fault and prove the fix by re-running

Theory (TH)

CODE	STATEMENT
TH1	Convert between binary, denary and hexadecimal
TH2	Explain how characters, sound and images are represented
TH3	Describe common network topologies and protocols
TH4	Identify common security threats and their mitigations
TH5	Describe and trace a standard algorithm

Exam technique (EX)

CODE	STATEMENT
EX1	Trace a program by hand and state the output
EX2	Write pseudocode for a described problem
EX3	Complete a timed task within the time

7. Coverage map

DESCRIPTOR	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12
PL1	●											
PL2		●										
PL3			●									
PL4				●								
PL5					●							
PL6						●						
TR1	○	○	○	○	○	○	●					
TR2							●					
TR3							○		●			
DG1								●				
DG2								●				
DG3								●				
TH1										●		
TH2										●		
TH3											●	
TH4											●	
TH5									●			
EX1												●
EX2							○		●			●
EX3												●

○ introduced · ● assessed

The rule holds from the Padma map: each descriptor is *assessed* once. If a week assesses something twice, another week is doing redundant work.

8. Assessment

COMPONENT	WEIGHT	EVIDENCE
Programming portfolio	40%	Six ported programs, each working and documented
Practical task	25%	A timed 45-minute problem, unseen
Theory test	25%	Binary, representation, networks, security
Trace-and-explain	10%	Given code on paper, state the output and one bug

The practical task is the one that matters. A student who can port known code is a translator; a student who can write new code under time is a programmer. Week 9 and Week 12 both rehearse it.

The exit standard

A student is ready to begin iGCSE when they can:

- write a 30-line Python program from a written specification, unaided, in 45 minutes
- read a traceback and say what is wrong without running it again
- convert 173 to binary and back
- explain the difference between a switch and a router

That is the bar. Not "has completed the course."

---

9. Entry requirements

Do not accept students who have not completed Padma Terms 1–3. The whole course depends on the concepts being *already present*.

The five-minute entry check:

1. "Write the four-state table from memory." *(selection)*
2. "What does `trim` do, and why does it drift?" *(variables, cause and effect)*
3. "Name the four fault categories." *(debugging)*
4. "What does your `followLine` routine take in?" *(functions, parameters)*
5. "Why did the robot stop when you released the button?" *(loops, conditions)*

Four out of five, or they repeat Term 3. A student who cannot answer these will spend the whole course learning two things at once, and will learn neither.

---

10. What this course is not

Say all four of these out loud to the client, in writing, before they buy.

It is not iGCSE. Twelve weeks of bridging does not cover a two-year specification. After this, a student is ready to **begin** iGCSE, not to sit it.

It is not a qualification. No exam board is involved. The certificate from the Padma programme does not extend to this course unless a separate record is issued.

It does not use the robot for programming. KanKaung appears twice, as a *conversation*, not a target. Python on an ESP32 is possible and is a different course.

It does not suit everyone. Students who struggled with abstraction in Term 3 will struggle more here, because text removes the visual scaffolding. The entry check in §9 exists to protect them.

---

11. Staffing — the honest constraint

This is the hardest staffing problem in the whole programme.

A teacher who has delivered Padma well may still not be able to teach Python. The skills overlap less than they appear to: Padma teaching is about sequencing and debugging *student thinking*; Python teaching requires reading code fluently and knowing why `"5" + 5` fails.

Three realistic options:

OPTION	WHAT IT NEEDS	RISK
Upskill the Padma teacher	40–60 hours of personal Python before term, plus a term of co-teaching	Slow, but keeps continuity
Hire a CS teacher	A computing graduate, part-time	Costs more than the Padma teacher; may not know the students
Partner with a university	Undergraduates as tutors	Unreliable, but cheap

Recommendation: upskill, but only a teacher who *wants* to. A reluctant teacher delivering Python is worse than no course, because the students will conclude that code is not for them.

---

12. Logistics

ITEM	REQUIREMENT
Devices	One computer per student, **not shared** — text programming needs a keyboard
Software	Python 3 (any version ≥3.8), any editor; offline installation
Robot	Not required. Two demonstration sessions only
Room	Desks in rows, not clusters — this is a different kind of lesson
Printing	Wall reference: the transfer table (§4) and the error-message card (Week 8)

The inversion worth noticing: the expensive part of Padma was the hardware. The expensive part here is *one device per student*. A school that bought four robots for a class needs thirty keyboards for this, and that is a different budget conversation.

---

13. What changes in the teaching method

Padma lessons are group work at one device per four. This course is **one student, one keyboard**, and that changes almost every routine.

PADMA	BRIDGE TO TEXT
Four roles (Pilot, Navigator, Recorder, Reporter)	One programmer, working alone
Pair debugging	Rubber-duck debugging and code review in pairs, not shared typing
"Ask the robot"	Read the traceback
Showcase defence	Timed practical
One device per table	One device per student

The hardest habit to break is students asking the teacher to look at their screen. In Padma that was efficient; here it prevents them learning to read an error message. Week 8 is entirely about breaking it.

---

14. What a week looks like

Two 40-minute sessions. Same shape both days, so students build a rhythm fast.

Session A — Port (40 min)

1. **(5 min)** Show the Padma program from the matching term. Read it aloud.
2. **(10 min)** Teacher writes the Python version live, narrating each translation.
3. **(20 min)** Students write it themselves from the transfer table, closed book on the Python.
4. **(5 min)** One student's version on the projector; class compares to the teacher's.

Session B — Extend (40 min)

1. **(5 min)** Recap: what did the port teach you about Python?
2. **(10 min)** One new element (types, `return`, `append`, whichever this week needs).
3. **(20 min)** Change the requirement. Make the ported program do something the Padma one could not.
4. **(5 min)** Exit ticket: one thing that surprised you.

The design logic: Session A proves equivalence. Session B proves the text version is *more capable* — which is the argument for learning it at all.

---

15. The iGCSE handover

At the end of Week 12, each student leaves with:

- a programming portfolio (six ported programs, one new one)
- a trace-and-explain sheet
- a mock theory paper with a mark
- a one-page **readiness statement**: what they can do, what they still need, and whether they should start iGCSE next year

That statement is the honest product. It is also what protects the centre: a student who is not ready is told so in writing, before they enrol in a two-year course they will fail.

---

16. Verification notes

Same discipline as the Padma map.

STATUS	ITEM
✅ Designed	Structure, transfer table, descriptor set, assessment
⬜ **Unverified**	The current iGCSE Computer Science specification — paper structure, programming language permitted, theory topic list

STATUS	ITEM
▫ **Unverified**	Whether the centre can enter candidates, or only prepare them
▫ **Unverified**	Which language the local exam board accepts (Python is common, not universal)

Before printing: obtain the current specification from the awarding body. Replace the `PL/TR/DG/TH/EX` codes with the official assessment objectives. The lesson structure will not change; only the labels on §6 and §7 will.

Bridge to Text framework v1.0 · Awaken Innovation Lab · Specification unverified pending confirmation