education sciences

MDPI

Article

The Hidden Burden of Keywords: Cognitive Load and Language Differences in Novice Python Programming

Raina Mason * and Carolyn Seton

Faculty of Science and Engineering, Southern Cross University, Gold Coast Campus, Bilinga, QLD 4225, Australia; carolyn.seton@scu.edu.au

* Correspondence: raina.mason@scu.edu.au or dr.raina.mason@gmail.com

Abstract

Keyword recognition represents a fundamental skill in programming, yet little research has examined how novices develop this ability or how language background affects keyword learning. This study investigated cognitive load and keyword recognition accuracy amongst 27 novice programming students (15 English as an additional language [EAL] and 12 English as a native language [ENL]) during an intensive six-week Python course. Students completed a keyword recognition task at Weeks 1 and 6, identifying and classifying 23 Python keywords while reporting cognitive load using the Klepsch instrument. The results revealed no significant improvement in identification accuracy (Week 1: 39.80%; Week 6: 48.16%) or classification accuracy (40% at both time points) despite intensive instruction. The reported extraneous cognitive load significantly increased from Week 1 to Week 6 ($p = 0.039$, $d = 0.99$), contradicting Cognitive Load Theory predictions that schema automation reduces extraneous load with experience. EAL students reported a significantly higher intrinsic cognitive load ($p = 0.030$, $d = 0.91$) and a marginally lower keyword identification accuracy ($p = 0.058$, $d = -0.54$) than ENL students. All students (100%) who identified keywords also missed duplicate instances, indicating universal incomplete processing. These findings challenge assumptions about schema development timelines in programming education and document measurable linguistic barriers that persist even after substantial instruction, with implications for inclusive computing pedagogy.

Keywords: EAL; ESL; programming; CS1; international students; cognitive load

1. Introduction

The ability to recognise programming keywords is a foundational skill for novice programmers, yet the cognitive effort required to identify and classify these terms may vary based on students' linguistic backgrounds and the nature of the keywords themselves. This study explores the cognitive load experienced by students when identifying keywords in Python v3.13 code (Sweller, 1988, 2010), with particular attention to whether keywords that re-purpose common English words (such as class, if, and else) present different challenges than keywords that use truncated or constructed terms (such as def, elif, and __init__) (B. Becker, 2019; Munasinghe et al., 2021). By investigating the role of cognitive load during programming instruction, this research seeks to inform pedagogical strategies that better support diverse learners in introductory programming courses.

Programming relies on precise syntax and keywords that dictate the behaviour of code, making accurate identification crucial for comprehension and problem-solving. For students learning to program, this task involves not only recognising which words function as

Check for updates

Academic Editor: João Piedade

Received: 23 December 2025
Revised: 4 March 2026
Accepted: 14 April 2026
Published: 20 April 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

keywords but also understanding their syntactic role—a dual cognitive demand. Students with English as an Additional Language (EAL) may face an additional layer of complexity, as they must simultaneously navigate language processing and conceptual understanding. However, given the limited research on how linguistic background intersects with keyword recognition, this study takes an exploratory approach to examining potential differences between EAL and English as a native language (ENL) students.

This research was conducted over a six-week intensive programming course, equivalent to CS0-CS1 level instruction, where students progressed from basic sequence and selection structures through to functions, classes, and object-oriented programming concepts. Students completed keyword identification tasks at Week 1 (early exposure to programming concepts) and Week 6 (after sustained practice and instruction), allowing an examination of how cognitive load changes with increased familiarity and experience (Kalyuga et al., 2003; Klepsch et al., 2017). The tasks required students to identify keywords within Python code samples and classify each keyword as either a re-purposed English word or a new/constructed programming term, while timing data captured the speed and efficiency of their responses.

By examining how cognitive load evolves across the learning period, and preliminarily exploring whether linguistic background influences this progression, this study provides educators with insights to guide curriculum design and instructional approaches tailored to diverse student populations in introductory programming contexts.

The remainder of this paper is structured as follows. Section 2 reviews Cognitive Load Theory, the linguistic characteristics of programming keywords, and documented challenges for EAL students in computing education. Section 3 outlines the research design, participants, instruments, and analysis approach. Section 4 presents the results, organised by research question, and Section 5 discusses the findings in relation to theory and practice before presenting concluding remarks.

2. Literature Review

2.1. Cognitive Load Theory

Cognitive Load Theory (CLT) provides a framework for understanding how instructional design affects learning by considering the limitations of human working memory (Sweller, 1988, 2010). At its core, CLT posits that working memory has a severely limited capacity for recalling and manipulating information. Working memory is typically able to handle only a few elements simultaneously (Cowan, 2001; Miller, 1956). This constraint becomes particularly relevant when learners encounter complex material, such as programming syntax and semantics, for which multiple elements must be processed and integrated concurrently.

However, working memory limitations can be substantially overcome through the use of schemas stored in long-term memory (Cooper & Sweller, 1987). Schemas are cognitive structures that allow multiple elements of information to be treated as a single unit, effectively expanding working memory capacity for familiar domains. For example, an experienced programmer can perceive an entire code structure (such as a for-loop) as a single meaningful chunk, whilst a novice must process each keyword, variable, and operator as separate elements. This schema-based expertise develops through sustained exposure and practice, gradually reducing the cognitive burden of tasks that initially overwhelmed working memory (Kalyuga et al., 2003).

Sweller and colleagues distinguished three types of cognitive load that together comprise the total mental effort experienced during learning (Paas et al., 2003; Sweller, 2010):

Intrinsic cognitive load (IL) represents the inherent difficulty of the material being learned, determined by the number of elements that must be processed simultaneously

and the degree of interactivity between those elements (Sweller, 2010). Element interactivity cannot be altered without changing the nature of the material itself. For instance, understanding how a Python class instantiates objects requires learners to simultaneously consider the class definition, the `__init__` method, attribute assignment, and object creation—all of which interact with one another. This level of element interactivity is intrinsic to object-oriented programming concepts and cannot be reduced through instructional design.

Extraneous cognitive load (EL) refers to the cognitive effort imposed by a poor instructional design or presentation format (Chandler & Sweller, 1992; Sweller, 1988). Unlike intrinsic load, extraneous load can and should be minimised through careful design decisions. Examples in programming education include unnecessarily complex code formatting, using confusing variable names in learning examples, or requiring students to mentally integrate information presented in separate locations. When students must expend working memory resources on navigating poorly designed materials, fewer resources remain available for processing the actual learning content.

The germane cognitive load (GL) represents the productive cognitive effort directed toward schema construction and automation—the very processes that facilitate learning (Paas et al., 2003; Sweller, 2010). This includes activities such as comparing examples, identifying patterns, and integrating new information with existing knowledge. Effective instruction aims to minimise the extraneous load whilst managing the intrinsic load appropriately, thereby maximising the available working memory capacity for germane processing.

The three-factor model of cognitive load has important implications for measuring and understanding learners' experiences. Klepsch et al. (2017) developed and validated a subjective rating scale that separately assesses the intrinsic, extraneous, and germane cognitive load through learner self-reports. This instrument allows researchers to distinguish between whether students are struggling due to inherent task difficulty (high intrinsic load) or a poor instructional design (high extraneous load) or whether they are productively engaged in learning processes (high germane load). The scale has demonstrated reliability and validity across multiple studies and contexts, making it particularly suitable for investigating cognitive demands in educational settings (Klepsch et al., 2017).

2.1.1. Cognitive Load Theory in Programming Education

Programming education has been recognised as a domain where cognitive load considerations are particularly critical (Mason, 2012; Morrison et al., 2016). Novice programmers must simultaneously manage multiple sources of cognitive demand: understanding abstract concepts (such as variables, data types, and control structures), learning precise syntax rules, developing problem-solving strategies, and debugging code when errors occur. Each of these demands draws upon limited working memory resources (Mason et al., 2016).

Research in computing education has demonstrated that instructional approaches informed by CLT can significantly improve learning outcomes. For instance, worked examples and subgoal labelling reduce the extraneous load by providing a clear structure and reducing the search through problem spaces (Morrison et al., 2016). Similarly, attention to element interactivity when sequencing programming concepts helps manage the intrinsic load by ensuring that students are not overwhelmed by attempting to process too many novel elements simultaneously (Mason et al., 2016).

However, an often-overlooked source of cognitive load in programming education concerns the linguistic demands of technical terminology. Programming languages rely heavily on keywords—reserved words with specific syntactic and semantic functions. Many of these keywords are English words (or English-derived constructions), creating an

assumption that English proficiency provides an advantage in programming comprehension. This assumption has rarely been examined through the lens of cognitive load theory, despite its potential implications for linguistically diverse student populations.

2.1.2. Linguistic Background and Schema Availability

Schema theory suggests that learners' prior knowledge substantially influences the cognitive load they experience when encountering new material (Cooper & Sweller, 1987). For programming keywords that repurpose common English words (such as `class`, `if`, `while`, and `return`), students with native or advanced English proficiency may benefit from pre-existing linguistic schemas. These schemas, stored in long-term memory, reduce the working memory burden of processing and remembering such terms. When a native English speaker encounters the keyword `class` in programming, they may automatically activate semantic associations related to categorisation and grouping—associations that align with the technical meaning in object-oriented programming.

Conversely, students with English as an Additional Language (EAL) may possess weaker or qualitatively different schemas for English terminology (Roussel et al., 2017). This schema deficit could manifest in two ways. First, EAL students may lack the automatic activation of relevant semantic associations, requiring deliberate, working-memory-intensive processing to understand keyword meanings. Second, they may activate inappropriate schemas (such as associating `class` primarily with "classroom", rather than "category"), leading to confusion and an increased cognitive load.

The distinction between repurposed English keywords and constructed or truncated programming terms (such as `def`, `elif`, `__init__`) further complicates this picture. Constructed terms offer no schema-based advantage to any learner because neither English native speakers nor EAL students have pre-existing long-term memory structures for these novel constructions. Thus, cognitive load theory predicts that language backgrounds should differentially affect performance with repurposed keywords (with which schema availability varies) but not necessarily with constructed keywords (with which all learners face equivalent novelty).

Despite the theoretical basis for expecting language-related differences in the cognitive load during programming instruction, the empirical investigation of this phenomenon has been limited. The current study addresses this gap by examining whether EAL students experience measurably different patterns of cognitive load (using the validated Klepsch et al. instrument) when identifying and classifying programming keywords and whether any such differences increase or decrease following sustained exposure to programming instruction.

The following sections review the existing research on the linguistic characteristics of programming languages, document the challenges faced by EAL students in computing education, and examine what is currently known about keyword recognition as a foundational programming skill.

2.2. Linguistic Background and Programming Education

Technical units, such as programming units, include a number of technical and jargon terms that are integral to understanding the learning content of the unit. These include the necessary "keywords" that are essential to the programming language being learned. Programming keywords are reserved words with specific syntactic and semantic functions within a language. In Python, as in most widely used programming languages, these keywords are predominantly English words or English-derived constructions (Munasinghe et al., 2021). However, Python keywords can be broadly categorised into two types: those

that re-purpose common English words with established meanings and those that use constructed, truncated, or novel terms created specifically for programming purposes.

For example, the Cambridge Dictionary (2025) lists multiple English meanings for "class", including "a group into which goods, services, or people are put according to their standard" and "a group of plants or animals with similar biological structure." In Python, `class` defines the structure and behaviour of objects—a meaning most closely related to the latter definition. Students with strong English proficiency may activate this semantic connection automatically, whilst EAL students may encounter competing meanings that add a cognitive burden without supporting comprehension of the programming concept.

This dual nature of Python keywords, with some re-purposing familiar English words and others introducing novel or truncated constructions, has potential implications for cognitive load. Table 1 presents examples of both categories from the Python language used in this study. Note that, in this study, "keyword" is used in a broad pedagogical sense to include both Python's strictly reserved words (e.g., `if`, `def`) and commonly used built-in functions and types (e.g., `print`, `int`) that students are instructed to treat as reserved—that is, words to avoid reusing as variable names. Whilst these built-ins are not syntactically reserved in Python 3, shadowing them produces confusing runtime behaviour; students in this course encountered this distinction explicitly during instruction.

Table 1. Examples of keyword types in Python (using "keyword" in the broad pedagogical sense; see text).

Re-Purposed English Words	Constructed/Truncated Terms
`class`	`def` (define)
`if`, `else`	`elif` (else-if)
`for`, `while`	`__init__` (initialisation)
`return`	`bool` (boolean)
`break`, `pass`	`str` (string)
`in`	`len` (length)
`print`	`int` (integer)

For re-purposed English keywords, students with native or advanced English proficiency may benefit from their automatic activation of semantic associations (schema) stored in long-term memory. The keyword `if`, for instance, maps directly onto the conditional logic familiar from natural language ("if it rains, I will bring an umbrella"). Similarly, `while` leverages temporal/conditional understanding ("while waiting, I read a book"), and `return` draws on the concept of giving something back. These semantic connections may reduce cognitive load by providing meaningful anchors for understanding keyword function, particularly when combined with formal instruction about their technical usage.

In contrast, constructed or truncated terms such as `def` or `elif` offer no such schema-based advantage because these must be learned as a novel technical vocabulary by all students, regardless of their linguistic background. The keyword `def` requires explicit instruction concerning the fact that it introduces a function definition; there is no intuitive connection to guide understanding, and recognising that `def` is a truncated form of "definition" takes additional cognitive resources. Similarly, `__init__` combines a truncated English root word with programming conventions (double underscores) that signal special meaning in Python, creating a term that even proficient English speakers must learn as domain-specific jargon.

The predominance of English-based terminology in programming languages reflects historical and pragmatic factors. Python, like most widely used programming languages, was developed in English-speaking contexts and designed with English-speaking users as the primary audience (B. A. Becker, 2021). Early computing pioneers and language

designers predominantly worked in English, embedding English vocabulary and linguistic structures into the syntax and semantics of programming languages that subsequently achieved global adoption. Whilst this design choice facilitates adoption among English-proficient programmers and enables a degree of code readability through natural language parallels, it potentially creates systematic barriers for the global majority of programmers for whom English is an additional language (B. Becker, 2019; Diethelm & Goschler, 2014). A recent study on non-English-speaking students learning Python programming, with the assistance of GenAI tools, showed that English-based keywords, terminology and concepts frustrated their efforts to use the GenAI tool with their language effectively (Prather et al., 2025).

This linguistic asymmetry in programming language design raises questions about equity in computing education. If keyword recognition and classification depend partly on English language schemas, then linguistically diverse learners may face additional cognitive demands that are orthogonal to their capacity for computational thinking or problem-solving. Understanding whether and how these linguistic demands manifest as measurable differences in cognitive load is essential for developing inclusive pedagogical approaches in programming instruction. A study with Japanese students learning Python programming indicated that prior expertise had a direct effect on some aspects of learning—consistent with cognitive load theory—but that, even for advanced students, English-based programming languages and environments represented challenges linguistically (Martono & Ohwada, 2025).

Learning Challenges for EAL Students in Computing

Students with English as an Additional Language (EAL) face well-documented challenges when studying in English-medium instruction contexts (Guo, 2018; Power, 2023). These challenges encompass multiple dimensions: general language proficiency barriers, cultural adjustment, reduced confidence in classroom participation, concerns about academic integrity, and ultimately, performance gaps relative to English as a Native Language (ENL) peers (Raj et al., 2019). Research across diverse disciplines has consistently demonstrated that EAL students achieve lower grades and experience higher attrition rates, even when they possess an equivalent intellectual capacity and prior subject knowledge to ENL students (Cho & Bridgeman, 2012; Moore & Waters, 2020; Schudde et al., 2023). However, the majority of this research has focused on general language ability and psychosocial factors, without examining the specific cognitive demands associated with processing discipline-specific technical terminology and jargon.

This oversight is particularly significant in Computer Science and Information Technology education, where disciplines are notably jargon-heavy (B. A. Becker, 2021). Programming courses introduce students to extensive technical vocabulary—not only keywords within programming languages but also concepts such as algorithms, data structures, object-orientation, polymorphism, and inheritance (Scherer et al., 2020). For EAL students, the necessity of learning this technical terminology adds an additional layer of cognitive demand beyond the conceptual understanding required of all novice programmers (Guo, 2018; Mason et al., 2016). These students must simultaneously navigate language processing (decoding unfamiliar English terms and their technical meanings) and conceptual understanding (grasping abstract computational concepts), effectively managing a dual cognitive burden that may not be experienced to the same degree by ENL students. Despite the potential significance of this linguistic–cognitive intersection, relatively little research has systematically investigated how technical jargon affects EAL students' cognitive load in computing contexts.

Exploratory studies have begun to examine linguistic challenges faced by EAL students in programming education, though none have employed validated cognitive load measures. Notably, B. Becker (2019) and B. A. Becker et al. (2021) documented how compiler error messages—written in idiomatic English—pose comprehension challenges for EAL students, suggesting that language background influences not only code comprehension but also debugging processes.

Our own prior research provided documentation revealing that international students (the majority of whom are EAL learners) perform at lower levels in programming units compared to domestic students, even when prior academic achievement is controlled for (Mason & Seton, 2020, 2021). These performance gaps raised questions about the cognitive mechanisms underlying them—questions that performance data alone could not answer. The present study was designed to investigate those mechanisms directly, using validated cognitive load measures to examine whether the linguistic background creates measurable differences in the cognitive effort required for keyword recognition, rather than simply assuming that it does.

2.3. Keyword Recognition and Programming Comprehension

The ability to accurately identify programming keywords represents a foundational skill for novice programmers, serving as a prerequisite for code comprehension, production, and debugging (Izu et al., 2019). Keywords constitute the syntactic building blocks of programming languages, defining control structures, data operations, and program organisation. Without reliable keyword recognition, students struggle to parse code correctly, misinterpret program logic, and generate syntax errors that impede progress (B. A. Becker et al., 2021; Izu et al., 2019). Yet keyword recognition is not an innate skill. Novice programmers must learn to distinguish keywords from identifiers (variable and function names), operators (mathematical and logical symbols), and general syntax elements (brackets, colons, indentation). This discrimination requires sustained exposure to code and explicit instruction about language-specific conventions (Xie et al., 2019).

For beginning programmers, code appears as an undifferentiated stream of symbols and words, making it difficult to identify which elements carry syntactic significance. Research on novice programming has documented the fact that students often confuse keywords with variable names, fail to recognise the syntactic role of reserved words, or overlook keywords entirely when reading code (B. Becker, 2019; Xie et al., 2019). These recognition failures lead directly to comprehension errors. Students who cannot identify `if` and `else` as keywords controlling conditional execution will misunderstand program flow. Similarly, failing to recognise `def` as the keyword introducing function definitions prevents the understanding of program structure and modularity. Keyword recognition thus gives access to higher-level programming comprehension, making it a critical target for instructional support in introductory courses.

The development of keyword recognition skill follows a trajectory from effortful, deliberate processing to increasingly automatic pattern recognition. This is consistent with the information processing model of cognition, as used by Sweller (1994). Initially, novice programmers must consciously attend to each word in a code sample, checking whether it matches the set of keywords they have memorised. This process imposes a substantial cognitive load, particularly when keywords are embedded in complex code structures with multiple interacting elements (Sweller, 1994). With repeated exposure and practice, however, keyword recognition becomes more automatic. Experienced programmers can scan code and immediately perceive keywords as distinct from other elements. This is similar to fluent readers perceiving word boundaries without conscious effort (Kalyuga et al., 2003) and expert chess players recognising game boards from any point in the

game (Simon & Gilmartin, 1973). The six-week timeframe of an intensive introductory programming course provides an opportunity to observe this developmental trajectory, examining whether students demonstrate improved keyword recognition accuracy and reduced cognitive load as they gain experience with Python code.

Beyond simply identifying which words are keywords, a deeper understanding requires recognising the nature and function of different keyword types (Izu et al., 2019). As established in previous sections, Python keywords vary in their relationship to natural language: some re-purpose familiar English words with semantic connections to their technical function, whilst others employ constructed or truncated terms that must be learned as novel vocabulary. The ability to classify keywords into these categories may indicate a deeper conceptual understanding not just of what keywords do but also of how they relate to broader linguistic and conceptual systems. Students who can correctly identify `class` as a repurposed English word and `def` as a constructed term demonstrate an awareness of the linguistic foundations of programming syntax, potentially supporting a more flexible and transferable understanding. This classification ability is relevant to programming learning specifically because understanding *why* a keyword looks the way it does supports generalisation: a student who recognises `def` as a constructed abbreviation is better positioned to infer the function of similarly constructed terms in new contexts, whilst a student who understands `while` as a repurposed conditional word can leverage natural language intuition to anticipate its behaviour. Keyword classification accuracy, therefore, serves as an indicator of conceptual depth in programming comprehension, not merely of English vocabulary knowledge.

However, it remains unclear whether the keyword classification ability develops in parallel with keyword identification or whether it requires additional cognitive resources and meta-linguistic skills. It is also uncertain whether the cognitive load associated with keyword recognition and classification differs between repurposed and constructed keyword types and whether any such differences are moderated by students' English language proficiency. The current study addresses these questions by employing a two-level task design: students must first identify which words in a code sample are keywords (recognition) and then classify each identified keyword as either a repurposed English word or a constructed programming term (classification). By measuring both accuracy and timing alongside self-reported cognitive load, the study examines how these skills develop across an intensive six-week course and explores whether linguistic background influences the cognitive demands of these tasks.

3. Methodology

The preceding literature review established a theoretical and empirical foundation for the current study. Cognitive Load Theory and schema theory together predict that linguistic background should influence the cognitive demands of keyword recognition, yet no prior study has empirically tested this using validated instruments distinguishing between intrinsic, extraneous, and germane load. The current study addresses this gap through a systematic empirical investigation. Using the validated Klepsch et al. (2017) cognitive load instrument, this research examined students' self-reported cognitive load when completing keyword recognition and classification tasks at two time points—Week 1 (early exposure) and Week 6 (following sustained instruction)—of an intensive introductory programming course. Multiple measures (identification accuracy, classification accuracy, response timing, and cognitive load sub-scales) enabled a comprehensive examination of performance and cognitive effort, with an exploratory comparison between EAL and ENL students where sample sizes permitted.

3.1. Research Design Overview

Figure 1 presents an overview of the study design, illustrating the two-time-point structure and the measures collected at each point.

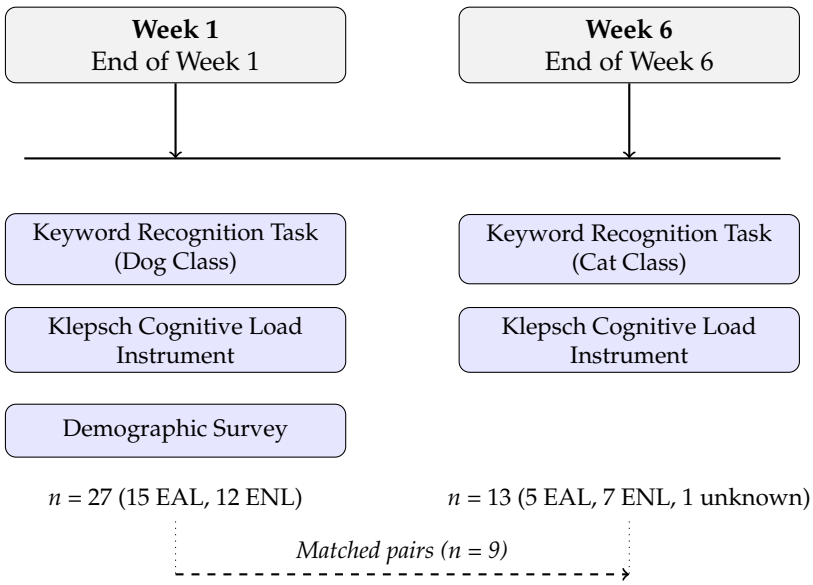


Figure 1. Research design overview showing measures administered at each time point and the matched pairs structure (dotted line).

3.2. Research Questions

This study investigated the cognitive load experienced by students when identifying programming keywords in Python code, with an exploratory examination of whether linguistic background (English as an Additional Language versus English as a Native Language) influences this cognitive effort. Specifically, the research addressed:

RQ1 How does cognitive load (intrinsic, germane, and extraneous) change from Week 1 to Week 6 of an intensive introductory programming course?

RQ2 Does keyword type (repurposed English words versus constructed/truncated programming terms) differentially affect identification accuracy and classification performance?

RQ3 Do timing measures (speed of keyword identification) correlate with reported cognitive load?

RQ4 Preliminary exploration: Do EAL and ENL students differ in reported cognitive load (intrinsic, extraneous, and germane) and keyword identification accuracy?

3.3. Research Design and Participants

All students enrolled in an undergraduate intensive six-week introductory programming course (equivalent to CS0–CS1 level) at an Australian regional university in the second half of 2025 were invited to participate in this research. The course covered fundamental programming concepts, including sequence, selection, loops, functions, methods, classes, and object-oriented programming principles. Students were introduced to the concept of programming keywords during their first week of instruction.

Participants included both domestic Australian students and students from international locations, representing diverse linguistic backgrounds (EAL and ENL). Of 77 students in the course, a total of 42 students initiated the first survey; however, only 27 students completed sufficient portions to be included in the analysis (15 EAL students, 12 ENL students). For the second survey administered at the end of Week 6, 15 students completed the questionnaire (5 EAL, 7 ENL, 3 with unknown language background). Due to the small matched sample size, particularly for EAL students in the Week 6 survey,

primary analyses focus on the full cohort with an exploratory examination of language background effects where sample sizes permit.

Participation was voluntary and anonymous, with students using a self-generated identification code to enable matched-pair comparisons across time points while maintaining anonymity.

3.4. Instruments and Measures

3.4.1. Validity and Reliability

The cognitive load instrument used in this study is the validated scale developed by Klepsch et al. (2017), which has demonstrated reliability and validity across multiple educational contexts. Rather than recalculating psychometric properties for the current small sample, we rely on the established validity and reliability of this instrument, as reported in the original development and validation studies.

The keyword recognition task was developed specifically for this study and thus required its own validity evidence. Face validity was established through a structured practice phase in which participants completed an identical task on a simpler code sample with immediate corrective feedback, ensuring that all participants understood the task requirements before the timed assessment began. Task equivalence across time points was addressed through the use of structurally identical code samples (a Dog class at Week 1 and a Cat class at Week 6), controlling for code familiarity whilst maintaining identical keyword distributions and task demands.

Construct validity was examined through pilot testing with four computing academics, all experienced Python programmers, prior to deployment. This pilot revealed a theoretically meaningful pattern: the one staff member who spoke English as an additional language failed to recognise some keywords and made errors in keyword type classification, whilst the three ENL staff members did not exhibit these errors. This result, occurring with expert programmers whose programming knowledge was equivalent, suggested that the task was sensitive to language-background effects independent of programming expertise, providing preliminary evidence that the task measured what it was designed to measure.

3.4.2. Participant Background Questions

The Week 1 survey collected demographic information and English language proficiency data. The participants reported their age group, gender, and whether English was the language they spoke most as a child. The students who indicated that English was not their native language provided additional information, including: the languages spoken at home, their years speaking English, and their methods of English language acquisition (e.g., living in an English-speaking country, compulsory school learning, parental influence, self-study, preparatory programmes).

Participants generated a unique identification code that used elements of their personal information (middle name initials, birth month, street name prefix) to enable matching between Week 1 and Week 6 responses whilst preserving anonymity.

3.4.3. The Keyword Recognition Task

The keyword recognition task required participants to identify Python keywords within code samples and classify each keyword as either:

- Re-purposed English words: keywords that use common English vocabulary (e.g., class, if, else, for, in, return, self).
- Constructed/truncated programming terms: keywords created specifically for programming purposes, including truncated English words and novel constructions (e.g., def, elif, __init__).

Practice component: Before the timed assessment, the participants completed a practice activity using a simple comparison function (def number_compare) that demonstrated the keyword identification and classification process. This practice included feedback showing correct answers, ensuring students understood the task requirements.

Timed assessment: Following the practice, the participants completed the keyword recognition task on a more complex code sample featuring object-oriented programming. Week 1 used a Dog class implementation; Week 6 used an identical structure with a Cat class to control for code familiarity whilst maintaining task equivalence.

The code samples contained multiple instances of the same keywords (e.g., def appeared three times, and self appeared multiple times) and included both keyword types. The interactive interface recorded the following.

- **Accuracy Measures:** Which words were clicked as keywords (correct identifications, missed keywords, false identifications of non-keywords). Identification accuracy was calculated as the proportion of all 23 keyword instances correctly identified, where each instance (including duplicate occurrences of the same keyword) was treated as a separate item in the denominator. Missing a duplicate instance of an already-identified keyword was, therefore, penalised equally to missing a unique keyword, reflecting the expectation that systematic processing would extend to all occurrences;
- **Classification Accuracy:** Whether identified keywords were correctly classified as repurposed or constructed;
- **Timing Data:** Time to first click, time to last click, time to page submission, and total number of clicks.

This design allowed for a nuanced analysis of performance errors, distinguishing between: (a) the failure to identify keywords, (b) the incorrect classification of correctly identified keywords, and (c) the misidentification of non-keywords as keywords.

3.4.4. Cognitive Load Measures

Immediately following the keyword recognition task, participants completed eight items from the validated Cognitive Load Scale developed by Klepsch et al. (2017). This instrument measures three dimensions of cognitive load using 7-point Likert scales (1 = Strongly Disagree; 7 = Strongly Agree):

- **Intrinsic Load (IL):** cognitive demand inherent to the task (2 items):
  - "For this task, many things had to be kept in mind simultaneously"
  - "This task was very complex"
- **Germane Load (GL):** productive cognitive effort directed toward learning (3 items):
  - "I made an effort, not only to understand several details but to understand the code"
  - "My point while doing the task was to understand everything correctly"
  - "The task consisted of elements supporting my understanding of the task"
- **Extraneous Load (EL):** cognitive effort from task design or presentation (3 items):
  - "During this task it was exhausting to find the important information"
  - "The design of this task was very inconvenient for answering the questions"
  - "During this task it was difficult to recognise and link the crucial information"

Sub-scale scores were calculated by averaging the relevant items, with higher scores indicating a greater cognitive load in each dimension.

3.5. Data Analysis

3.5.1. Data Preparation

Participant responses from both surveys were downloaded from Qualtrics (2025) into Excel spreadsheets and combined using the identification codes to create matched pairs for students who completed both surveys. Data were cleaned to remove incomplete responses and verify identification code consistency.

3.5.2. Statistical Approach

Given the exploratory nature of the study and small sample sizes, particularly for between-group comparisons, the analysis focused on descriptive patterns and effect sizes. Paired-sample analyses examined changes in cognitive load, keyword identification accuracy, and timing measures from Week 1 to Week 6 for matched students. Exploratory comparisons between EAL and ENL students were conducted when sample sizes permitted, with an emphasis on effect size estimation, rather than significance testing, due to power limitations. Appropriate parametric (paired *t*-tests) and non-parametric tests (Wilcoxon signed-rank test, Mann–Whitney *U* test) were selected based on distributional assumptions and sample characteristics, with alpha set to 0.05. Cohen's *d* is reported throughout as a standardised effect size estimate to aid practical interpretation and comparison across measures. Whilst Cohen's *d* formally assumes normally distributed data, its use alongside non-parametric tests is an accepted practice in small-sample educational research where effect size communication is prioritised alongside inferential caution (Lakens, 2013). All analyses were conducted in Microsoft Excel, with non-parametric tests (Wilcoxon signed-rank and Mann–Whitney *U*) performed using the Real Statistics Using Excel add-in (Zaiontz, 2024), a freely available and widely used resource for non-parametric analysis that extends Excel's native statistical functions.

3.6. Ethical Considerations

This research received ethics approval from the <Removed for Review> Human Research Ethics Committee (approval number 2025/108). All participation was voluntary, and students could withdraw at any time without consequence. Anonymity was preserved through the use of self-generated identification codes, and no personally identifying information was collected. Students who experienced distress were directed to university counselling services.

3.7. GenAI Use

The authors used Claude.ai (Anthropic) to assist with drafting and editing portions of the manuscript text. In particular, Claude.ai was used for improving language, grammar, and style. The authors did not use Claude.ai as a co-author, to generate original content, or to interpret research findings.

3.8. Expected Results

This study was designed to test the hypothesis that EAL students will have reduced performance (accuracy and time taken) and increased mental effort when reading and processing Python keywords to determine if they are re-purposed English words, compared to ENL students. It was further expected that EAL students would not increase in performance of identification of repurposed English words over the course of a Term, compared to the ENL students.

4. Results

4.1. Participants

Twenty-seven students completed the Week 1 survey and keyword recognition task. The sample was predominantly aged 18 to 24 years (77.8%, $n = 21$) and male (70.4%, $n = 19$). Participants were relatively evenly distributed across language backgrounds, with 12 ENL students (44.4%) and 15 EAL students (55.6%).

Among EAL students, the most common home language was Nepali ($n = 8$), and experience with English ranged from less than a year ($n = 4$) to over 5 years ($n = 7$). It should be noted that two EAL students reported English as their home language and had spoken English for over five years, despite indicating that English was not the language they spoke most frequently as children. These students were retained in the EAL group based on their self-identification but may function more similarly to ENL learners; a sensitivity analysis excluding these cases is reported in the Limitations section.

Thirteen students completed the Week 6 task, including nine who had also completed Week 1 (6 ENL, 3 EAL). Four students who only participated in Week 6 did not complete demographic questions, so their language background is unknown.

4.2. Cognitive Load

4.2.1. Research Question 1: Changes in Cognitive Load from Week 1 to Week 6

Descriptive statistics for cognitive load across time points and language groups are presented in Table 2 and Figure 2. At Week 1, students overall reported a moderate intrinsic load ($M = 3.78$, $SD = 1.40$), a high germane load ($M = 5.17$, $SD = 1.31$), and a low extraneous load ($M = 2.60$, $SD = 1.22$) on the 7-point scale. This pattern suggests that, while the keyword recognition task was moderately complex, students reported directing cognitive effort toward understanding, and the task design imposed a relatively low unnecessary cognitive burden.

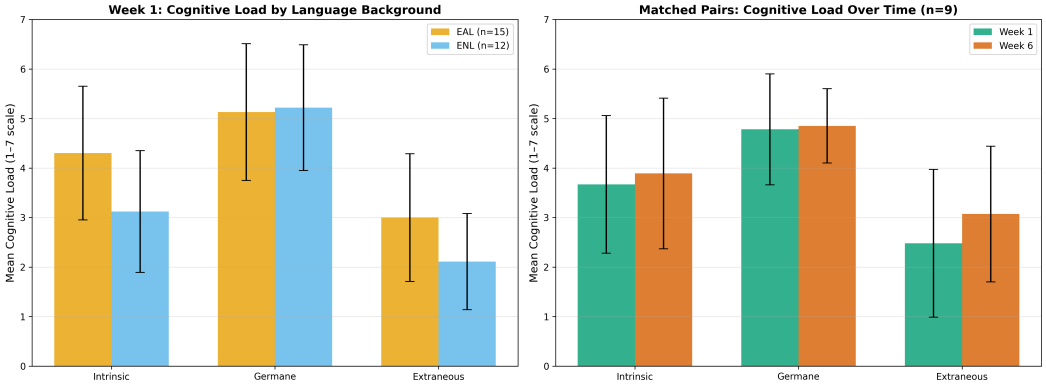


Figure 2. Cognitive load by time point and language background. Left panel: EAL students reported a significantly higher intrinsic load than ENL students at Week 1 ($p = 0.030$, $d = 0.91$), indicating greater perceived task complexity, whilst the germane load was equivalent across groups. Right panel: for matched pairs ($n = 9$), extraneous load increased significantly from Week 1 to Week 6 ($p = 0.039$, $d = 0.99$), contrary to CLT predictions of decreasing load with experience. The error bars represent standard deviations.

To examine changes in cognitive load over time, we analysed the nine students who completed both Week 1 and Week 6 assessments using Wilcoxon signed-rank tests. Contrary to expectations that cognitive load would decrease as students developed expertise, intrinsic and germane load remained relatively stable from Week 1 to Week 6, with small non-significant increases (intrinsic: +0.22 points, $W = 10.50$, $p = 0.656$, $d = 0.17$; germane: +0.07 points, $W = 21.00$, $p = 0.891$, $d = 0.08$). However, extraneous load showed a significant

increase of 0.59 points from Week 1 (M = 2.48, SD = 1.49) to Week 6 (M = 3.07, SD = 1.37), W = 2.50, p = 0.039, d = 0.99, representing a large effect. This unexpected increase in extraneous load may reflect students' heightened meta-cognitive awareness of task demands or fatigue effects in the intensive course format.

Table 2. Cognitive load across time points and language groups. Key finding: EAL students reported substantially higher intrinsic load than ENL students at Week 1 (M = 4.30 vs. 3.12); extraneous load increased across time for the matched sample despite intensive instruction.

Sample	Time	*n*	Intrinsic Load M (SD)	Germane Load M (SD)	Extraneous Load M (SD)
Cross-sectional samples					
All Students	Week 1	27	3.78 (1.40)	5.17 (1.31)	2.60 (1.22)
EAL students	Week 1	15	4.30 (1.35)	5.13 (1.38)	3.00 (1.29)
ENL students	Week 1	12	3.12 (1.23)	5.22 (1.27)	2.11 (1.97)
All Students	Week 6	12	4.08 (1.56)	5.03 (0.74)	3.19 (1.40)
EAL students	Week 6	3	4.83 (1.53)	5.33 (0.33)	3.56 (2.14)
ENL students	Week 6	6	3.42 (1.39)	4.61 (0.80)	2.83 (0.98)
Longitudinal sample (matched pairs)					
All students	Week 1	9	3.67 (1.39)	4.78 (1.12)	2.48 (1.49)
All students	Week 6	9	3.89 (1.52)	4.85 (0.75)	3.07 (1.37)
EAL students	Week 1	3	4.00 (1.80)	5.22 (0.51)	2.67 (2.33)
EAL students	Week 6	3	4.83 (1.53)	5.33 (0.33)	3.56 (2.14)
ENL students	Week 1	6	3.50 (1.30)	4.56 (1.31)	2.39 (1.16)
ENL students	Week 6	6	3.42 (1.39)	4.61 (0.80)	2.83 (0.98)

Note. EAL = English as an additional language; ENL = English as a native language. Cognitive load measured on 7-point Likert scale (1 = very low, 7 = very high) using Klepsch et al. (2017) instrument. Week 6 cross-sectional sample excludes 4 students with unknown language background who only completed Week 6.

4.2.2. Research Question 4: Exploratory Examination of Language Background Differences

At Week 1, EAL students reported significantly higher intrinsic load (M = 4.30, SD = 1.35) compared to ENL students (M = 3.12, SD = 1.23), U = 134.50, p = 0.030, d = 0.91, representing a large effect (see Figure 2). This finding is consistent with schema theory predictions: ENL students likely had pre-existing linguistic schemas for repurposed English keywords (e.g., `class`, `if`, `while`), reducing the inherent complexity of the task, whereas EAL students needed to build these associations alongside learning the programming concepts. EAL students also reported a higher extraneous load (M = 3.00, SD = 1.29) than ENL students (M = 2.11, SD = 0.97), a large effect (d = 0.77) that did not reach statistical significance at the conventional threshold (U = 130.00, p = 0.052), likely reflecting the limited statistical power of the small sample, rather than the absence of a meaningful difference. In contrast, the germane load was virtually identical between the language groups (EAL: M = 5.13, SD = 1.38; ENL: M = 5.22, SD = 1.27), U = 85.50, p = 0.844, d = −0.07, indicating that both groups reported equivalent effortful processing toward understanding despite the differential cognitive burden.

It should be noted that there was substantial variability within both language groups, as evidenced by the large standard deviations across all measures of cognitive load. This variability likely reflects individual differences in factors such as prior programming exposure, English language proficiency among EAL students (which ranged from less than one year to over five years of English language experience), and metacognitive awareness. The diversity within language groups suggests that language background alone does not fully account for the cognitive load experience but, rather, interacts with other learner characteristics.

Given the critically small sample sizes for language-specific longitudinal analyses (n = 3 EAL, n = 6 ENL in matched pairs), sizes that preclude any meaningful inferential test-

ing and fall well below thresholds for reliable effect size estimation, the following patterns are reported purely as descriptive observations. No conclusions should be drawn from these comparisons; they are included only to illustrate the range of individual variation and to inform the design of future, adequately powered studies. The three EAL students showed substantial increases in intrinsic load (+0.83 points) and extraneous load (+0.89 points) from Week 1 to Week 6, whilst the six ENL students showed minimal changes (intrinsic: −0.08; germane: +0.06; extraneous: +0.44). However, none of these language-specific changes reached statistical significance (all $p > 0.25$), likely due to insufficient statistical power. These exploratory patterns suggest the possibility of differential cognitive load trajectories by language background, but this interpretation must be made with considerable caution, given the sample limitations.

4.3. Performance

4.3.1. Research Question 2: Keyword Identification and Classification Accuracy

Descriptive statistics for keyword performance across time points and language groups are presented in Table 3 and Figure 3. At Week 1, students overall demonstrated modest performance on the keyword recognition task, correctly identifying a mean of 9.15 keywords out of 23 (M = 39.80%, SD = 26.85%). Of the keywords successfully identified, students correctly classified approximately 39% as either repurposed or constructed terms (M = 39.35%, SD = 19.10%). False positive rates were generally low, with students incorrectly identifying non-keywords as keywords at a mean rate of 3.54% (SD = 9.35%). The large standard deviations across all measures indicate substantial individual variability in keyword recognition performance.

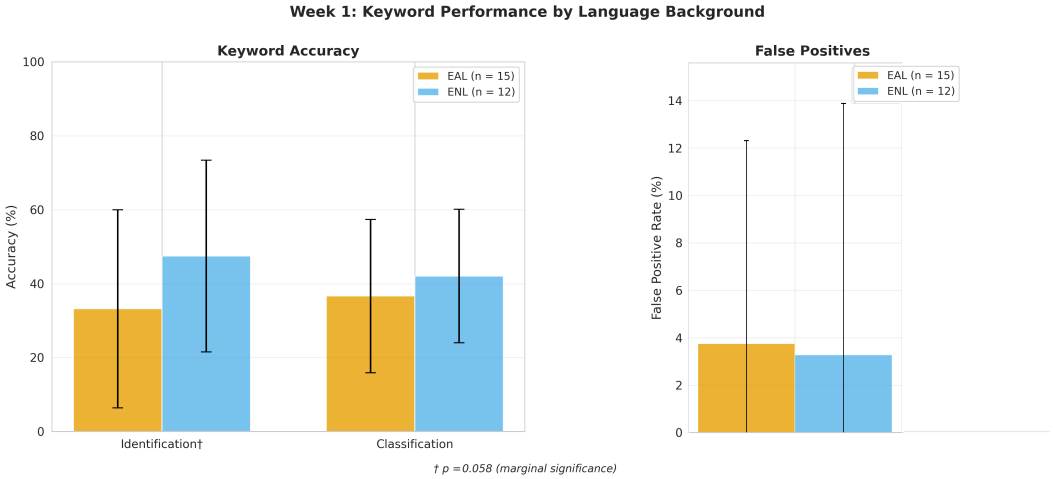


Figure 3. Week 1 keyword performance by language background. Left panel: EAL students identified fewer keywords than ENL students (33.23% vs. 47.46%), a medium effect approaching significance (p = 0.058, d = −0.54); classification accuracy was low for both groups (37–42%), indicating that keyword type discrimination was challenging, regardless of language background. Right panel: false positive rates were similarly low and comparable across groups, suggesting that both groups were initially cautious, rather than guessing. Error bars represent standard deviations. † p = 0.058 (marginal significance).

To examine changes in keyword performance over time, we analysed the nine students who completed both Week 1 and Week 6 assessments using Wilcoxon signed-rank tests. Contrary to expectations that performance would improve with six weeks of intensive instruction, identification accuracy remained virtually unchanged from Week 1 (M = 53.62%, SD = 18.06%) to Week 6 (M = 54.11%, SD = 26.19%), representing an increase of only 0.48 percentage points, W = 18.00, p = 1.000, d = 0.02. Similarly, classification accuracy showed a

minimal change, increasing from 42.38% (*SD* = 10.29%) to 43.40% (*SD* = 17.97%), $W = 18.00$, $p = 1.000$, $d = 0.07$.

Table 3. Keyword performance across time points and language groups. Key finding: identification accuracy remained below 50% at both time points for all groups; EAL students identified approximately 14 percentage points fewer keywords than ENL students at Week 1, a medium effect approaching significance ($p = 0.058$, $d = -0.54$).

Sample	Time	*n*	Keywords Identified M (SD)	Identification Accuracy M (SD)	Classification Accuracy M (SD)
Cross-sectional samples					
All students	Week 1	27	9.15 (6.18)	39.80 (26.85)	39.35 (19.10)
EAL students	Week 1	15	7.64 (6.16)	33.23 (26.78)	36.64 (20.71)
ENL students	Week 1	12	10.92 (5.96)	47.46 (25.92)	42.06 (18.03)
All students	Week 6	13	11.08 (6.29)	48.16 (27.35)	44.92 (17.69)
EAL students	Week 6	3	12.67 (5.77)	55.07 (25.10)	33.33 (29.54)
ENL students	Week 6	6	12.33 (6.68)	53.62 (29.06)	48.43 (8.75)
Longitudinal sample (matched pairs)					
All students	Week 1	9	12.33 (4.15)	53.62 (18.06)	42.38 (10.29)
All students	Week 6	9	12.44 (6.02)	54.11 (26.19)	43.40 (17.97)
EAL students	Week 1	3	10.67 (3.21)	46.38 (13.98)	38.80 (9.14)
EAL students	Week 6	3	12.67 (5.77)	55.07 (25.10)	33.33 (29.54)
ENL students	Week 1	6	13.17 (4.58)	57.25 (19.91)	44.17 (11.16)
ENL students	Week 6	6	12.33 (6.68)	53.62 (29.06)	48.43 (8.75)

Note. EAL = English as an additional language; ENL = English as a native language. Total keywords = 23. Week 6 cross-sectional sample excludes 4 students with unknown language background who only completed Week 6.

An unexpected finding emerged in the false positive rates: students showed a substantial increase in incorrectly identifying non-keywords as keywords, from a mean of 3.80% (*SD* = 10.45%) at Week 1 to 16.96% (*SD* = 33.32%) at Week 6, representing an increase of 13.16 percentage points. Although this change did not reach statistical significance ($W = 4.50$, $p = 0.250$), the effect size was moderate ($d = 0.39$), with the lack of significance likely attributable to the small sample size and high variability. This pattern suggests that students may have adopted a more liberal clicking strategy by Week 6, possibly attempting to maximise keyword identification at the expense of precision.

4.3.2. Clicking Strategies and Precision

An important methodological consideration concerns the possibility that students could inflate identification accuracy through indiscriminate clicking, rather than discriminative keyword recognition. To examine this, we analysed students' overall clicking behaviour and calculated precision: the proportion of items clicked as keywords that were actually keywords (i.e., correct identifications/[correct identifications + false positives]).

At Week 1, students clicked a mean of 35.41 items (*SD* = 23.01, range: 4–107) from the 61 available clickable items, representing 58.04% of all items. Of these clicks, 10.50 items (*SD* = 7.48) were designated as keywords (either correctly or incorrectly), with 9.15 being correct identifications and 1.35 being false positives. This yielded a mean precision of 86.05% (*SD* = 27.04%), indicating that most students were relatively selective in their keyword identification at Week 1.

By Week 6, clicking behaviour had become considerably more liberal. Students clicked a mean of 48.62 items (*SD* = 36.62, range: 14–131), representing 79.70% of available items—a substantial increase from Week 1. More concerning, the number of items designated as keywords increased to 18.31 (*SD* = 17.32), with 11.08 being correct and 7.23 being false positives. Precision dropped to 73.22% (*SD* = 34.44%), reflecting a shift towards less discriminative clicking strategies.

Individual variation was substantial at both time points, with some students adopting extremely liberal strategies. At Week 1, 14 students (52%) clicked more than 30 items (>49% of all items), with the most extreme case involving 107 clicks (175% of available items, suggesting multiple clicks per item). At Week 6, 7 students (54%) clicked more than 30 items, with two students exceeding 100 clicks. Notably, whilst these liberal clickers achieved higher raw identification counts, their precision often suffered dramatically. For instance, one Week 6 student who clicked 131 items achieved 16 correct identifications but also produced 39 false positives, yielding only 32.7% precision.

The decline in precision from Week 1 to Week 6, coupled with the dramatic increase in false positives reported earlier, provides converging evidence that students adopted more indiscriminate clicking strategies over time, rather than developing more accurate keyword recognition. This pattern suggests that the modest increase in correct identifications at Week 6 (from 9.15 to 11.08) may reflect a shift in response strategy (clicking more items), rather than improved discriminative ability. The finding that identification accuracy showed no significant improvement for matched pairs (Week 1: 53.62%, Week 6: 54.11%, $p = 1.000$), whilst precision declined substantially, reinforces this interpretation: students were clicking more but not learning better.

These findings highlight a limitation of the open-ended clicking task design: it permitted response strategies that could inflate accuracy metrics without requiring genuine keyword recognition. Future research might benefit from forced-choice designs or confidence ratings to better distinguish discriminative knowledge from guessing behaviour. Nevertheless, the overall pattern of low identification accuracy (below 50% at both time points) and poor classification accuracy (40%) remains clear, even when liberal clicking strategies are accounted for, indicating that keyword recognition posed substantial challenges for novice programmers, regardless of their response approach.

4.3.3. Research Question 4: Language Background Differences in Keyword Performance

At Week 1, EAL students demonstrated lower keyword identification accuracy ($M = 33.23\%$, $SD = 26.78\%$) compared to ENL students ($M = 47.46\%$, $SD = 25.92\%$), representing a difference of 14.23 percentage points. This difference approached statistical significance, $U = 47.00$, $p = 0.058$, $d = -0.54$, indicating a medium effect size (see Figure 3). This finding aligns with the cognitive load results, where EAL students reported a significantly higher intrinsic load, suggesting that the task was both subjectively and objectively more challenging for students learning programming in an additional language.

Classification accuracy showed a similar pattern, with EAL students achieving 36.64% ($SD = 20.71\%$) compared to ENL students' 42.06% ($SD = 18.03\%$), though this difference was not statistically significant, $U = 43.00$, $p = 0.620$, $d = -0.28$. Notably, false positive rates were comparable across language groups (EAL: $M = 3.76\%$, $SD = 8.56\%$; ENL: $M = 3.29\%$, $SD = 10.59\%$), $U = 99.00$, $p = 0.339$, $d = 0.05$, indicating that both groups exercised similar levels of caution in their keyword selections at Week 1.

The modest classification accuracy across both language groups (approximately 37–42%) suggests that distinguishing between repurposed English keywords and constructed programming terms posed considerable difficulty for all novice programmers, regardless of language background. This finding warrants further investigation into the specific sources of classification errors, which we explore in the following section, examining keyword type effects.

4.3.4. Keyword Type Classification Analysis

To examine whether familiarity with English aided keyword classification, we analysed students' accuracy in correctly classifying repurposed English keywords (e.g., `class`,

`if`, `while`; 19 total) versus constructed programming-specific terms (e.g., `def`, `__init__`; 4 total). Given ENL students' presumably greater familiarity with repurposed English words, we hypothesised they would show an advantage in classifying these terms correctly compared to constructed keywords.

Contrary to this hypothesis, the Week 1 results revealed that students classified constructed keywords more accurately than repurposed keywords. Overall, students correctly classified 45.37% of constructed keywords (SD = 39.85%) compared to 32.36% of repurposed keywords (SD = 26.63%), representing a 13.01 percentage point difference. Wilcoxon signed-rank tests confirmed that this difference was statistically significant, W = 25.00, p = 0.005, d = 0.61, with a medium-to-large effect size. However, an important limitation must be noted: the task contained substantially fewer constructed keywords (4) than repurposed keywords (19), which may have influenced these accuracy rates independently of keyword characteristics. The smaller set size reduces memory demands and the potential for confusion, making it difficult to disentangle the effects of set size from the inherent properties of keyword types.

This pattern of higher accuracy for constructed keywords held across both language groups (see Figure 4). EAL students classified constructed keywords at 35.00% (SD = 42.05%) versus repurposed at 25.26% (SD = 25.57%), a difference of 9.74 percentage points. ENL students showed a larger difference: 58.33% (SD = 34.27%) versus 41.23% (SD = 26.25%) for repurposed, representing 17.11 percentage points. However, the magnitude of this difference did not vary significantly between language groups, U = 78.00, p = 0.570.

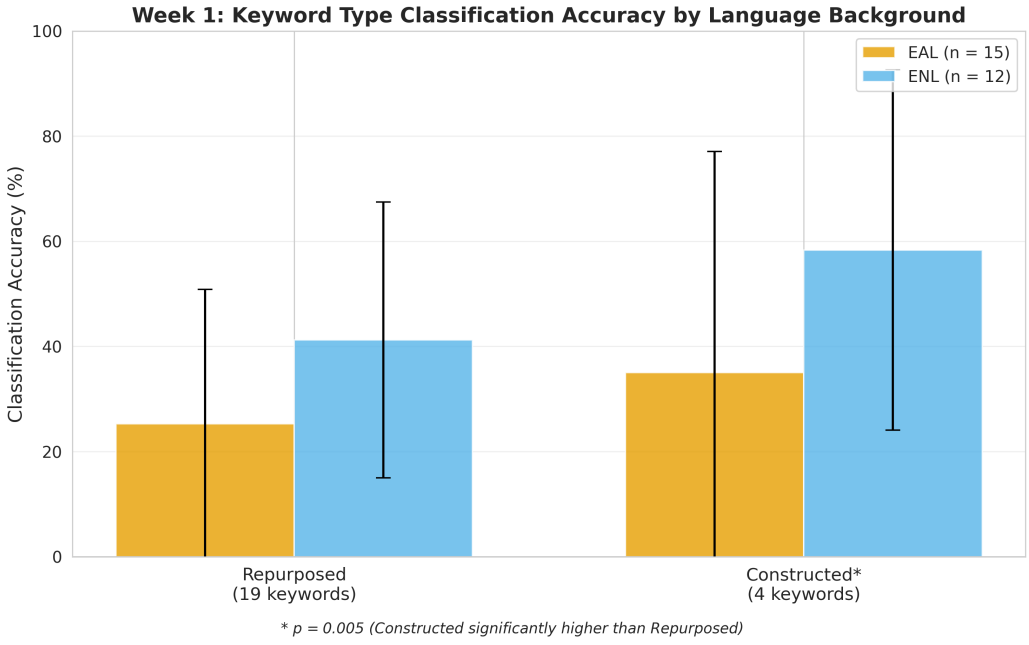


Figure 4. Week 1 keyword type classification accuracy by language background. Across both groups, students classified the four constructed keywords (e.g., `def`, `__init__`) more accurately than the 19 repurposed English keywords (e.g., `class`, `while`), a statistically significant difference (p = 0.005, d = 0.61). ENL students showed a larger gap between keyword types (17.1 percentage points) than EAL students (9.7 percentage points), though this between-group difference was not significant. Note that unequal set sizes (4 constructed vs. 19 repurposed) confound the interpretation of this pattern. The error bars represent standard deviations. * p = 0.005.

For matched pairs, the pattern of higher constructed keyword accuracy persisted from Week 1 to Week 6, though neither keyword type showed a statistically significant improvement (see Figure 5). Repurposed keyword accuracy showed a non-significant

decrease from 46.78% (*SD* = 21.61%) to 42.69% (*SD* = 30.63%), $W = 13.50$, $p = 0.570$, $d = -0.18$. Constructed keyword accuracy showed a non-significant increase from 61.11% (*SD* = 30.90%) to 66.67% (*SD* = 39.53%), $W = 8.00$, $p = 0.781$, $d = 0.14$. The persistence of this accuracy difference across six weeks of instruction is noteworthy, though its interpretation remains complicated by the unequal set sizes.

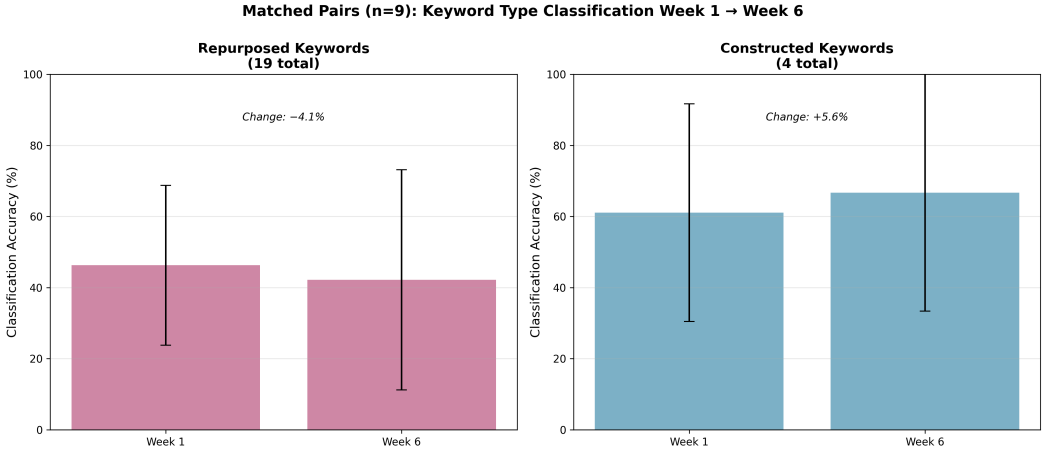


Figure 5. Matched pairs ($n = 9$): keyword type classification accuracy Week 1 → Week 6. The advantage for constructed over repurposed keywords persisted across both time points, with neither type showing statistically significant change (repurposed: −4.1 percentage points, $p = 0.570$; constructed: +5.6 percentage points, $p = 0.781$). The absence of improvement across six weeks of intensive instruction suggests keyword type classification requires considerably longer to consolidate than the course duration provides. Note that unequal set sizes (4 constructed vs. 19 repurposed) limit interpretation. The error bars represent standard deviations.

The observed accuracy difference between keyword types could arise from several factors that are difficult to disentangle in the current design. Most prominently, the unequal set sizes (4 constructed keywords versus 19 repurposed keywords) represent a significant confound: smaller sets impose lower memory demands, reduce opportunities for confusion amongst similar items, and may receive more concentrated instructional attention per item. Any or all of these factors could account for the higher constructed keyword accuracy independent of the keywords' inherent characteristics.

However, characteristics specific to keyword types may also contribute to the observed pattern. Constructed keywords are orthographically distinctive, with unconventional character combinations (e.g., double underscores in `__init__`) or abbreviated forms (e.g., `def`) that visually mark them as programming-specific, rather than natural language. Conversely, repurposed English keywords may create semantic interference: students must inhibit everyday meanings (e.g., "class" as a group of students) to access programming-specific meanings (e.g., a blueprint for objects). This interference would affect both EAL and ENL students, though the pattern of slightly larger differences for ENL students (17.11 versus 9.74 percentage points) hints that stronger everyday associations with these English words might paradoxically hinder, rather than help, programming-specific classification—though this difference between groups was not statistically significant.

The significant accuracy difference favouring constructed keywords challenges simplistic assumptions that linguistic familiarity with English keywords necessarily aids programming comprehension. However, without a design that controls for set size, we cannot determine whether this pattern reflects the cognitive burden of semantic differentiation for repurposed terms, the advantages of learning a smaller and more distinctive set, or some combination of both factors. Future research employing balanced sets of repur-

posed and constructed keywords would be necessary to disentangle these explanations. These findings and their limitations are discussed further in the Discussion section.

4.3.5. Duplicates Analysis: A Measure of Cognitive Load

To examine whether students engaged in complete and systematic keyword identification, we analysed their performance on duplicate keyword instances. The task contained 10 unique keywords that appeared multiple times throughout the code, yielding 13 duplicate instances. If students identified a keyword once (e.g., `if`), thorough processing would lead them to identify all subsequent instances. Conversely, missing duplicates of keywords already identified suggests incomplete processing and provides a behavioural indicator of cognitive load (Figure 6).

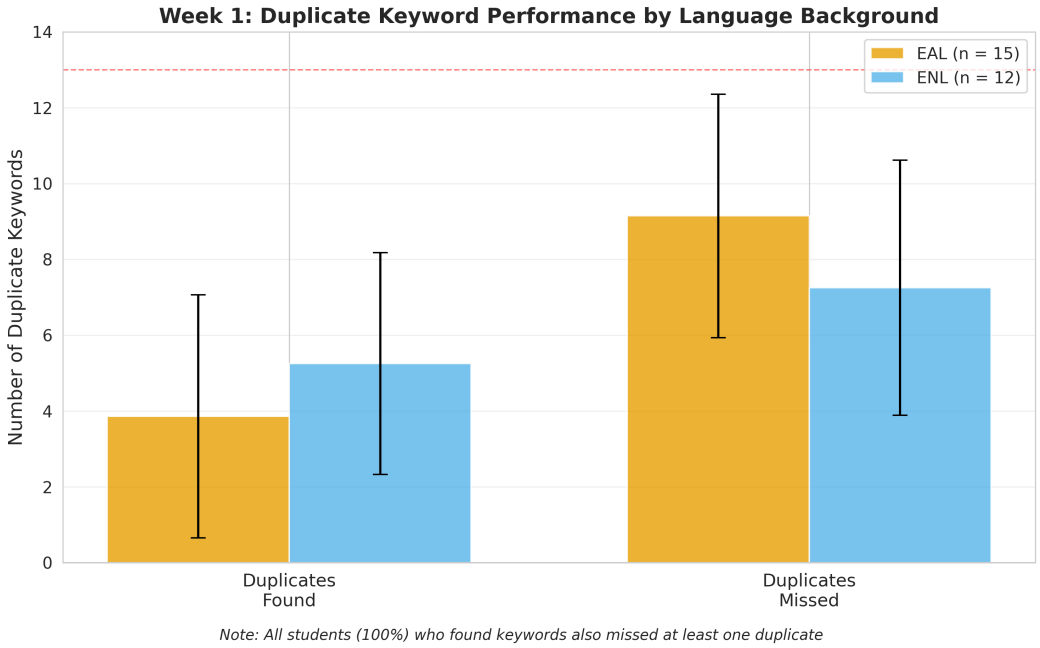


Figure 6. Week 1 duplicate keyword performance by language background. All students (100%) who identified at least one unique keyword also missed at least one duplicate instance—a universal pattern providing behavioural evidence of incomplete processing consistent with working memory constraints. EAL students missed more duplicates on average (M = 9.14) than ENL students (M = 7.25), a medium effect (d = 0.58) that did not reach statistical significance (p = 0.122). Error bars represent standard deviations. The dashed line indicates the maximum possible (13 duplicate instances).

At Week 1, students found a mean of 4.50 duplicate keywords (SD = 3.10) whilst missing 8.27 duplicates (SD = 3.35). Strikingly, 100% of students who identified at least one unique keyword (20 out of 27 students) also missed at least one duplicate of those same keywords. This universal pattern provides strong behavioural evidence that the keyword recognition task imposed substantial cognitive demands, leading to incomplete or inconsistent processing even for keywords that students had already encountered.

EAL students found fewer duplicates (M = 3.86, SD = 3.21) and missed more duplicates (M = 9.14, SD = 3.21) compared to ENL students (found: M = 5.25, SD = 2.93; missed: M = 7.25, SD = 3.36) (Figure 7). Whilst the difference in missed duplicates approached a medium effect size (U = 114.00, p = 0.122, d = 0.58), it did not reach statistical significance, likely due to the small sample size and high variability. However, the pattern aligns with the cognitive load findings, suggesting that EAL students experienced greater processing demands that interfered with systematic keyword detection.

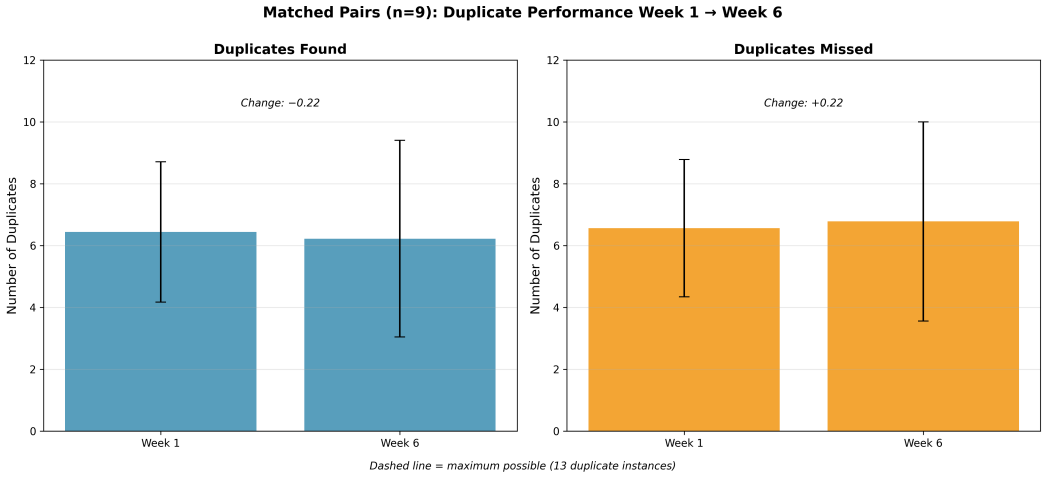


Figure 7. Matched pairs ($n = 9$): duplicate keyword performance Week 1 → Week 6. Students showed a negligible change in duplicates found or missed across the six weeks ($p = 1.000$, $d = 0.09$), and 100% of students who identified keywords at Week 6 still missed at least one duplicate—mirroring the Week 1 pattern exactly. The persistence of this universal incomplete processing suggests that the cognitive demands of systematic code scanning remain substantial well beyond initial exposure. The error bars represent standard deviations. The dashed line indicates the maximum possible (13 duplicate instances).

Contrary to expectations that students would develop more systematic processing strategies with experience, matched pairs showed no improvement from Week 1 to Week 6. Students continued to miss a similar proportion of duplicates at both time points (Week 1: $M = 6.56$, $SD = 2.19$; Week 6: $M = 6.78$, $SD = 3.31$), $W = 7.00$, $p = 1.000$, $d = 0.09$. All 12 students who found keywords at Week 6 also missed at least one duplicate (100%), mirroring the Week 1 pattern. This persistent incomplete processing, even after six weeks of intensive instruction, suggests that the cognitive demands of the task remained substantial throughout the course.

We examined whether missed duplicates correlated with self-reported cognitive load at Week 1. Although missed duplicates showed a weak positive trend with extraneous load ($r = 0.271$, $p = 0.190$), no correlations reached statistical significance (intrinsic: $r = -0.192$, $p = 0.357$; germane: $r = -0.108$, $p = 0.609$). The absence of strong correlations between this behavioural measure and self-reported cognitive load may reflect the multi-faceted nature of cognitive processing or measurement timing differences, as cognitive load was assessed after task completion, whilst duplicate detection occurred during the task itself.

4.4. Task Completion Time

4.4.1. Time Required for Keyword Recognition

At Week 1, students required a mean of 67.16 s ($SD = 42.86$, range: 9.21–224.52) to complete the keyword recognition task. The substantial variability in completion times suggests considerable individual differences in the task approach, with some students working quickly, whilst others engaged in more deliberate, time-intensive processing. Language background did not significantly influence completion time at Week 1 (EAL: $M = 66.33$ s, $SD = 33.43$; ENL: $M = 68.20$ s, $SD = 54.02$, $U = 103.00$, $p = 0.542$, $d = -0.04$), indicating that both groups approached the task with similar temporal investment, despite the differential cognitive load experienced.

At Week 6, the mean completion time increased to 89.89 s ($SD = 71.31$, range: 16.02–292.57), representing a 22.73 s increase from Week 1. For the nine matched pairs, the increase of 14.84 s from Week 1 ($M = 84.87$, $SD = 53.82$) to Week 6 ($M = 99.71$, $SD = 79.14$) was not statistically significant, $W = 18.00$, $p = 0.652$, $d = 0.14$. This unexpected pattern, that is,

students taking longer, rather than becoming faster with experience, contrasts with typical skill development trajectories and may reflect several factors. Students may have adopted more thorough but time-intensive strategies, experienced fatigue effects in the intensive course format, or faced a Week 6 task that, despite an identical structure, proved more challenging in the specific code context (cats, rather than dogs). The increased variability at Week 6 (larger standard deviations) further suggests diverging strategic approaches to the task over time.

4.4.2. Research Question 3: Relationships Between Time and Cognitive Load

To examine whether the time investment was related to cognitive processing demands, we calculated Spearman correlations between task completion time and the three cognitive load dimensions at each of the time points. Contrary to expectations that a longer task duration would be associated with a higher cognitive load, no significant correlations emerged at either Week 1 or Week 6 (see Table 4).

Table 4. Spearman correlations between task completion time and cognitive load. Key finding: no significant relationships were found at either time point, indicating that time-on-task was not a reliable proxy for cognitive processing demands in this context.

Time Point	Cognitive Load	*r*	*p*	*n*
Week 1	Intrinsic Load	0.105	0.617	25
	Germane Load	0.121	0.564	25
	Extraneous Load	−0.079	0.709	25
Week 6	Intrinsic Load	0.169	0.600	12
	Germane Load	0.304	0.336	12
	Extraneous Load	−0.193	0.548	12

Note. All correlations non-significant ($p > 0.33$). Spearman's rho used due to small sample sizes and non-normal distributions.

At Week 1, correlations between time and all three cognitive load dimensions were weak and non-significant (intrinsic: $r = 0.105$, $p = 0.617$; germane: $r = 0.121$, $p = 0.564$; extraneous: $r = -0.079$, $p = 0.709$, $n = 25$). The Week 6 pattern was similarly null, with the strongest correlation being a weak-to-moderate positive association between time and germane load ($r = 0.304$, $p = 0.336$) that did not reach statistical significance. The remaining Week 6 correlations were negligible (intrinsic: $r = 0.169$, $p = 0.600$; extraneous: $r = -0.193$, $p = 0.548$, $n = 12$).

The absence of time–cognitive load relationships suggests that task duration was not a straightforward indicator of processing difficulty in this context. Students experiencing high cognitive load did not necessarily work more slowly, nor did those working quickly necessarily experience a lower load. This dissociation may reflect compensatory strategies, with some students working quickly despite high load, whilst others invested time in careful processing that reduced subjective difficulty. Alternatively, the relationship between objective task duration and subjective cognitive load may be moderated by individual differences in processing speed, working memory capacity, or meta-cognitive awareness. The null findings underscore the importance of using multiple measurement approaches—both behavioural and self-report—to capture the complexity of cognitive processing during programming tasks.

5. Discussion

This study examined cognitive load and keyword recognition accuracy amongst novice programming students during an intensive six-week Python course, with particular attention to differences between English as an additional language (EAL) versus English as

a native language (ENL) learners. Contrary to expectations of progressive improvement and reduced cognitive load, our results revealed remarkable stability in keyword recognition performance alongside an unexpected increase in extraneous cognitive load. EAL students reported significantly higher intrinsic cognitive load and showed marginally lower keyword identification accuracy, suggesting that linguistic background creates additional barriers beyond general novice difficulties. These patterns challenge assumptions about how novices develop keyword recognition skills and how cognitive load evolves with programming experience.

5.1. The Paradox of Increasing Extraneous Load (RQ1)

Perhaps the most striking finding is that extraneous cognitive load significantly increased, rather than decreasing, from Week 1 to Week 6 (large effect, $d = 0.99$), contradicting standard Cognitive Load Theory predictions. In CLT, extraneous cognitive load specifically refers to cognitive effort imposed by factors *external* to the learning content itself—most commonly, poor instructional design, suboptimal interface presentation, or task format demands (Chandler & Sweller, 1992; Sweller, 1988). Crucially, extraneous load is considered reducible through better design, unlike intrinsic load, which is inherent to the material. The observed increase, therefore, warrants a careful consideration of what, specifically, was driving the elevated burden at Week 6.

Several distinct sources of extraneous load are plausible and worth distinguishing. First, the assessment format itself, an interactive clickable interface with no explicit systematic strategy cue, may have become more cognitively burdensome as students grew more aware of the complexity they were navigating. At Week 1, novices may have approached the task without fully grasping its demands; by Week 6, greater familiarity exposed the difficulty of exhaustive, systematic scanning. Second, learner fatigue in the intensive course context represents a plausible contributor: compressed learning timelines reduce opportunities for consolidation, and sustained cognitive effort across multiple weeks can generate commission errors and elevated subjective load independent of task design (Tamminen et al., 2025). Third, response strategy changes may have played a role; the shift toward more liberal clicking at Week 6 (mean clicks increasing from 35.41 to 48.62) would itself generate more decision-making demands, increasing the subjective sense of task burden without reflecting any change in instructional design. Finally, heightened metacognitive awareness may have increased: students who were initially too overwhelmed to perceive complexity may have rated load lower at Week 1 precisely because they were not genuinely engaging with the task's demands.

It is important to note a methodological constraint on interpreting the EL items. Research on cognitive load measurement with interactive learning media has demonstrated that subjective load ratings can diverge from objective performance measures, particularly when learners adopt varying engagement strategies (Skulmowski & Rey, 2020). The Klepsch et al. (2017) items for extraneous load (e.g., "During this task it was exhausting to find the important information") could plausibly capture fatigue and strategy-based effort as much as design-imposed burden, meaning the observed increase should be interpreted as a signal of increased subjective difficulty, rather than a definitive indicator of poor task design.

Clicking behaviour patterns provide supporting evidence for the metacognitive awareness interpretation. Week 1 showed relatively high precision (86.05%), suggesting selective clicking, whilst Week 6 precision dropped (73.22%) as clicks increased—consistent with deeper engagement with uncertainty, rather than confident guessing. Language background may moderate these trajectories: ENL students, with more automatic English processing might show different patterns than EAL students still managing both language

processing and programming concepts concurrently. Future research with larger samples and process-tracing methods (such as think-aloud protocols or eye-tracking) could disentangle these contributing sources.

5.2. Persistent Difficulties with Keyword Recognition (RQ1, RQ2)

Keyword identification and classification accuracy showed no significant improvement from Week 1 to Week 6 (both $p = 1.000$), with students correctly identifying approximately 40–50% of keywords at both time points despite daily exposure to Python keywords. From a schema theory perspective (Sweller, 1988), we would expect students to progressively build mental representations enabling increasingly automatic recognition. The persistence of low accuracy is consistent with the possibility that keyword recognition schemas develop more slowly than six weeks of intensive instruction provides, although this interpretation must be tempered by the small matched sample ($n = 9$) and limited statistical power. It is equally plausible that task context effects or response strategy shifts account for the apparent stasis and that longer-term studies with larger samples would reveal a gradual improvement not detectable here.

Several factors may contribute to these difficulties. First, re-purposed English keywords create the potential for semantic interference, through which everyday meanings compete with programming-specific meanings during processing (Cunnings & Fujita, 2024). Students encountering `class` must inhibit the everyday meaning to access the programming meaning, requiring executive control and working memory resources. Importantly, this inhibition demand is asymmetric: research on native and non-native language comprehension demonstrates that semantic interference effects are stronger and more persistent for non-native speakers, who rely more heavily on memory retrieval processes, rather than automatic lexical access (Cunnings & Fujita, 2024). This provides a psycholinguistic basis for the observed EAL performance disadvantage that extends beyond general language proficiency. Second, keywords appear embedded within complex syntactic structures where meaning depends on context, making recognition cognitively distinct from understanding function. Third, the finding that 100% of students who identified keywords also missed duplicate instances may suggest fundamental attention and working memory constraints that prevent systematic, exhaustive processing.

The declining precision alongside stable accuracy provides additional insight: students clicked more items at Week 6 (48.62 versus 35.41) with reduced discrimination (73.22% versus 86.05% precision). The increase in false positives (3.80% to 16.96%) suggests that students adopted more indiscriminate clicking —possibly from uncertainty or desperation— rather than developing more accurate knowledge. These findings suggest that explicit, sustained attention to keywords may be necessary throughout novice instruction, rather than assuming incidental acquisition through programming practice.

5.3. The Constructed Versus Repurposed Keyword Paradox (RQ2)

Students classified constructed keywords (e.g., `def`, `__init__`) significantly more accurately than repurposed English keywords ($p = 0.005$, $d = 0.61$) at both time points. However, this finding must be interpreted cautiously due to unequal set sizes: only 4 constructed versus 19 repurposed keywords. Smaller sets impose lower memory demands and reduce confusion, making it impossible to determine whether the accuracy advantage reflects genuine differences in keyword processing or simply advantages of learning a smaller set. Future research with balanced sets is necessary to disentangle these explanations.

Nevertheless, if the effect reflects keyword characteristics, several mechanisms might contribute. Constructed keywords possess orthographic distinctiveness (e.g., double underscores in `__init__`) that aids recognition and memory. More importantly, repurposed

keywords may create semantic interference, as students must inhibit well-established everyday meanings. The pattern of larger ENL advantages for constructed over repurposed keywords (17.11 versus 9.74 percentage points for EAL, although non-significant) hints that stronger everyday English associations might paradoxically increase interference, challenging assumptions about linguistic advantage in programming. Regardless of the mechanism, the pattern suggests that English proficiency may not provide unqualified advantages if repurposed keywords create semantic interference.

5.4. Time, Behavioural Measures, and Language Differences (RQ3, RQ4)

The absence of significant correlations between task completion time and self-reported cognitive load (all $p > 0.33$) reveals dissociation between objective and subjective measures, likely reflecting individual differences in processing speed, strategic choices, and the non-linear relationship between time and load. Some students work quickly despite a high load, whilst others invest time in careful, low-load processing.

Critically, temporal patterns differed by language background. Although the overall sample showed an increase in completion time (67.16 to 89.89 s), this increase was driven entirely by EAL students. The ENL students' times remained essentially stable (68.20 to 69.08 s). This differential pattern suggests fundamentally different trajectories: at Week 1, EAL students' brief times may have reflected incomplete engagement or confusion about task requirements, whilst at Week 6, an improved conceptual understanding led to more careful deliberation requiring extended language processing time. This pattern underscores that EAL students navigate qualitatively different cognitive demands with temporal costs that accumulate across programming activities.

The behavioural measure of missed duplicates provided compelling evidence of cognitive load through task performance itself. The finding that 100% of students who identified keywords also missed duplicate instances suggests universal incomplete processing that persisted from Week 1 to Week 6 ($p = 1.000$). When students identify `if` in one location but fail to identify it elsewhere, this pattern reveals attention lapses and working memory limitations that directly reflect cognitive strain. The universality and persistence of this pattern underscore the substantial cognitive demands that keyword recognition imposed throughout the course.

5.5. Educational Equity Implications (RQ4)

EAL students reported a significantly higher intrinsic cognitive load ($p = 0.030$, $d = 0.91$) and marginally lower keyword identification accuracy ($p = 0.058$, $d = -0.54$), documenting empirical evidence of linguistic barriers that persisted despite sufficient English proficiency for university admission. The higher intrinsic load suggests that understanding keyword meanings and functions was more difficult when processing in an additional language, while the 14.23 percentage point difference in identification accuracy represents a substantively meaningful performance gap that could compound across complex programming tasks.

Stable ENL completion times versus increased EAL times reveal additional dimensions of linguistic barriers: EAL students face temporal costs that extend beyond those experienced by ENL students, potentially affecting everything from assignment completion to examination performance. These findings align with broader evidence that language proficiency in the medium of instruction is a strong predictor of academic outcomes and that EAL students consistently face performance gaps even when controlling for prior achievement (Cho & Bridgeman, 2012; Moore & Waters, 2020; Schudde et al., 2023). Critically, the barriers documented here persisted despite the sufficient English proficiency for university admission, consistent with the finding of Prather et al. (2025) that even

EAL students using GenAI tools to support their Python learning were frustrated by the English-based nature of programming keywords and terminology.

Despite these barriers, the EAL students in this sample successfully progressed through the computing programme, suggesting resilience that, nevertheless, comes at costs to learning efficiency, confidence, and wellbeing (Raj et al., 2019). The field's efforts to broaden participation in computing must extend beyond traditional diversity dimensions to explicitly address how English-based programming languages create systematic disadvantages for the substantial and growing population of students learning to program in an additional language (B. Becker, 2019; Diethelm & Goschler, 2014). Support structures should include extended time accommodations, explicit instruction on keywords, rather than assuming obviousness, multilingual keyword reference materials, and institutional recognition that EAL students may require more processing time without indicating lesser ability or computational aptitude.

5.6. Limitations

The most consequential limitation of this study is sample attrition. Of 77 enrolled students, 42 initiated the Week 1 survey, 27 completed it sufficiently for inclusion, and only 13 completed the Week 6 survey; yielding just 9 matched pairs for longitudinal analysis. This attrition trajectory ($77 \rightarrow 27 \rightarrow 9$) severely constrains inferential strength: the matched sample is insufficient to detect all but large effects, and language-specific longitudinal analyses (EAL $n = 3$, ENL $n = 6$) are exploratory in the strictest sense. All longitudinal findings should be interpreted accordingly, as suggestive patterns, rather than generalisable conclusions. Cross-sectional comparisons at Week 1 ($n = 27$) are better powered but remain limited for subgroup analyses.

A fundamental limitation of the binary EAL/ENL classification used in this study warrants explicit discussion. Treating language background as a dichotomy collapses substantial within-group diversity: EAL students in this sample ranged from those with less than one year of English experience to those with more than five years, spoke a variety of home languages (predominantly Nepali), and had acquired English through different pathways (formal schooling, immersion, self-study). This heterogeneity means the EAL group likely includes individuals whose English proficiency and processing automaticity differ dramatically—with some approaching ENL-level fluency and others still effortfully decoding English text in real time. As a consequence, effect sizes for EAL versus ENL comparisons may simultaneously underestimate the true impact of language background on less proficient students whilst obscuring meaningful variation within the EAL group itself.

Future research should employ continuous or multi-level proficiency measures, rather than a binary classification. Promising options include standardised English proficiency scores (e.g., IELTS or TOEFL band scores), self-rated proficiency on validated scales, or behavioural measures of processing automaticity. These would allow the testing of whether proficiency level moderates keyword recognition and cognitive load in a graded fashion—a theoretically richer question than whether EAL and ENL groups differ on average. Investigating whether different first language backgrounds (e.g., languages with shared Latin roots versus non-alphabetic languages such as Nepali or Chinese) moderate outcomes would provide further insight into the specific linguistic mechanisms involved.

To assess whether the two ambiguous EAL cases (students who reported English as their home language and had spoken English for over five years, yet self-identified as EAL) influenced the primary findings, a sensitivity analysis was conducted that excluded these students ($n = 13$ EAL vs. $n = 12$ ENL). The key Week 1 EAL versus ENL comparisons were re-run using Mann–Whitney U tests. The intrinsic load difference remained significant and

strengthened slightly (M = 4.42 vs. 3.12, U = 120.00, p = 0.023, d = 1.01). The identification accuracy difference, which had been marginal in the full sample (p = 0.058), reached clear significance in the sensitivity analysis (M = 26.76% vs. 48.19%, U = 34.50, p = 0.018, d = −0.81). These results indicate that the two borderline cases were attenuating, rather than inflating, the observed EAL effects and that the primary findings are robust to their inclusion or exclusion.

The accuracy-based metrics used in this study carry an important limitation that requires acknowledgement, alongside the clicking strategy analysis. The open-ended interactive task format permitted response strategies that could inflate raw identification counts without reflecting genuine discriminative keyword knowledge. As documented in the Results, students clicked substantially more items at Week 6 (mean 48.62 versus 35.41) with a corresponding drop in precision (73.22% versus 86.05%), raising the possibility that apparent accuracy gains in the cross-sectional sample partly reflect a shift in response criterion, rather than improved keyword knowledge. This is consistent with signal detection theory, which distinguishes between a participant's *discriminability* (genuine ability to distinguish keywords from non-keywords) and their *response criterion* (willingness to click any given item). Standard accuracy metrics conflate these two components, making it difficult to determine whether students who identified more keywords at Week 6 genuinely knew more, or simply adopted a more liberal threshold for clicking. Precision analysis partially addresses this, but future designs would benefit from forced-choice formats, confidence ratings, or explicit signal detection analyses that model discriminability and response criterion separately.

The analysis of clicking strategies revealed an important insight into task design. Some students adopted extremely liberal strategies (>100 clicks), and individual cases highlighted the complexity of interpretation. An ENL student demonstrated a perfect Week 1 performance but clicked on most items at Week 6, suggesting task misunderstanding. Additionally, some students selected non-keywords that were English words, possibly misunderstanding the task as "identify English words", rather than "identify Python keywords." This confusion highlights the need for students to learn the distinction between "English word" and "Python keyword," which may not be immediately obvious to novices.

The unequal numbers of re-purposed (19) versus constructed (4) keywords create a fundamental confound that prevents definitive conclusions about keyword type effects. The open-ended clicking task allowed liberal response strategies that complicated interpretation, although precision analysis addresses this partially. Sample size limitations, particularly at Week 6 (13 students, 3 EAL), reduced statistical power. Language classification based on self-report may not fully capture linguistic complexity, and retrospectively assessed cognitive load may not accurately reflect moment-to-moment experiences. The single-institution intensive course context limits generalisability, and the Week 1 versus Week 6 code contexts (dogs versus cats), whilst structurally identical, might have differentially affected difficulty.

5.7. Theoretical and Practical Implications

These findings challenge Cognitive Load Theory assumptions about smooth, progressive schema development, suggesting that novice learning may involve periods of increased subjective difficulty as students become aware of complexities that were initially overlooked, or as intensive instruction creates cumulative fatigue. The persistent keyword recognition difficulties after 18 h of face to face instruction supplemented by videos and self-study materials, suggests that schemas for fundamental elements form more slowly than instructional timelines assume, with implications for expert–novice gap theories. The potential for semantic interference from re-purposed keywords points to an

un-theorised aspect: how existing linguistic knowledge might interfere with, rather than support, programming-specific meaning acquisition.

Practically, educators cannot assume that students will acquire keyword knowledge incidentally; explicit, sustained attention throughout introductory courses appears necessary. Several concrete strategies emerge from these findings.

Keyword scaffolding and dedicated instruction: Rather than treating keywords as implicit vocabulary absorbed through exposure, introductory courses should include dedicated keyword recognition activities: keyword sorting tasks (matching keywords to their functions), cloze-style exercises in which students complete code with missing keywords, and annotated code reading with explicit keyword highlighting. Keywords should be introduced explicitly at each conceptual milestone and revisited as new ones are added, rather than assuming prior keywords are consolidated.

Reduced keyword density in early examples: Code samples used in teaching should minimise the number of distinct keywords appearing simultaneously for novice programmers. Worked examples for early concept instruction might deliberately restrict the keyword set, gradually expanding density as students consolidate prior vocabulary. This directly addresses intrinsic load management by limiting the number of novel elements requiring simultaneous processing (Sweller, 2010).

Systematic code-reading strategy instruction: The universal pattern of missed duplicates (100% of students at both time points) suggests that novices do not naturally develop systematic scanning habits. Explicit instruction in structured code-reading approaches—such as reading line-by-line, colour-coding all instances of a given keyword, or using a keyword checklist—may help students overcome the working memory limitations that prevent exhaustive processing.

Multilingual keyword resources: For EAL students, multilingual reference materials providing translations or plain-language explanations of English-derived keywords in students' home languages would reduce the dual burden of language processing and concept acquisition. Keyword glossaries paired with annotated code examples in simplified English, or in students' native languages where feasible, can reduce the extraneous load imposed by unfamiliar vocabulary without compromising engagement with programming concepts (Prather et al., 2025).

Extended time accommodations: The temporal costs documented for EAL students—increased task duration at Week 6 without a corresponding accuracy advantage—suggests that EAL learners may require more processing time for programming tasks without this need indicating less ability. Formal extended time accommodations in programming assessments, analogous to those already provided in other language-intensive disciplines, are warranted and should be framed as recognition of additional cognitive demands, rather than remediation.

Intensive course monitoring: The significant increase in extraneous load across the six-week course signals that the accumulated cognitive burden in intensive programming formats is substantial. Educators should build explicit consolidation periods, consider whether the compressed format allows adequate time for keyword schemas to form, and use brief mid-course keyword recognition checks to identify students falling behind in this foundational skill before difficulties compound.

At the curriculum level, introductory programming should include explicit keyword recognition objectives with adequate time and resources. Six weeks of intensive instruction, at least in the current format, appears insufficient for robust schema development, with implications for curriculum pacing. Assessment design should favour formats that demand discriminative knowledge, rather than permitting liberal response strategies.

5.8. Future Research Directions

Studies employing balanced sets of re-purposed and constructed keywords are needed to test the semantic interference hypothesis independently of set size effects. Longitudinal research extending beyond six weeks would clarify whether difficulties represent persistent barriers or slower-than-expected development. Alternative measurement approaches (eye-tracking, think-aloud protocols, forced-choice formats) could address current task limitations. Intervention research comparing instructional approaches could identify effective methods for improving keyword recognition. Cross-linguistic research that compares learning in native versus additional languages would strengthen the evidence on linguistic barriers. Finally, research on broader schema development aspects would contextualise these keyword-specific findings within overall programming expertise development. A theoretically motivated extension concerns whether semantic interference operates asymmetrically across keyword sub-types: if re-purposed keywords with strong, well-consolidated everyday meanings (e.g., `class`, `return`) generate greater interference for ENL students, whose associations are more automatic and entrenched, this could produce a subset of items where EAL students outperform ENL students, analogous to interference effects documented in other domains where prior knowledge impedes relearning (Cunnings & Fujita, 2024). Identifying such a third keyword category would require designs that isolate individual items, measure association strength, and include sample sizes sufficient to detect crossover effects.

6. Conclusions

This study set out to examine how novice programmers develop keyword recognition skills during intensive instruction and whether language background affects this development. The findings paint a sobering picture: after six weeks of intensive Python instruction—representing approximately 18 h of direct teaching plus substantial additional practice—students showed no measurable improvement in their ability to identify or classify Python keywords. This persistent difficulty occurred despite daily exposure to keywords in lectures, readings, exercises, and programming assignments, suggesting that keyword recognition schemas develop far more slowly than instructional timelines typically assume.

The theoretical contribution of this work lies in raising questions about assumptions regarding how quickly novices develop fundamental programming knowledge. With the important caveat that the matched sample was small (n = 9) and the statistical power was limited, the observed pattern is consistent with schema development for even basic elements like keywords requiring considerably more time and deliberate practice than six weeks provides. Whether this reflects a genuine developmental plateau, slower-than-expected schema formation, or simply the constraints of a small intensive sample remains an open question for larger longitudinal studies to address. The cognitive demands of keyword recognition showed no sign of diminishing across the study period, whatever the underlying cause. The increase in extraneous cognitive load over time, whether reflecting cumulative fatigue or evolving meta-cognitive awareness, further complicates simplistic narratives about progressive skill development in intensive learning contexts.

The educational equity implications are equally significant. EAL students faced measurably higher cognitive load and lower identification accuracy throughout the course, with temporal costs that extended beyond those experienced by ENL students. These barriers persisted despite English proficiency sufficient for university admission, documenting that linguistic diversity creates obstacles to programming comprehension that computing education must address explicitly. The field's efforts to broaden participation in computing must extend beyond traditional diversity dimensions to consider how English-based

programming languages create systematic disadvantages for the substantial population of students learning to program in a language other than English.

Practically, these findings argue for a fundamental reconsideration of how keyword knowledge is addressed in introductory programming instruction. Educators cannot assume that keyword recognition develops automatically through programming practice; rather, explicit, sustained attention to keywords throughout introductory courses appears necessary. For EAL students, targeted support that acknowledges linguistic barriers while maintaining high expectations is essential—not as remediation for deficits, but as recognition that English-based programming languages impose additional, measurable cognitive costs on multilingual learners.

The persistent universal pattern of incomplete duplicate processing—affecting every student at both time points—suggests that the cognitive demands of systematic code reading exceed novices' available attentional and working memory resources. This finding implies that code-reading strategies require explicit instruction and scaffolding, rather than assuming students will develop effective approaches independently.

Looking forward, this research opens important questions about programming language design, instructional approaches, and temporal expectations for novice learning. If six weeks of intensive instruction proves insufficient for keyword recognition mastery, how should computing curricula be structured to provide adequate time and support? If re-purposed English keywords create semantic interference (though our findings remain confounded by set size), what implications follow for language design decisions? How can computing education better serve the growing population of multilingual students without compromising learning outcomes?

These questions demand empirical investigation, rather than assumptions. Keywords, often dismissed as trivial or obvious, represent a more substantial challenge for novice programmers than the field has recognised—a hidden burden that shapes learning trajectories and affects students differentially based on linguistic background. Addressing this challenge requires taking keyword knowledge seriously as a learning objective deserving explicit instructional attention, adequate time, and equitable support structures. Only then can computing education fulfil its promise of broad, inclusive access to programming knowledge.

Author Contributions: Conceptualisation, R.M.; methodology, R.M. and C.S.; analysis, C.S.; investigation, R.M. and C.S.; resources, C.S. and R.M.; data curation, C.S.; writing—original draft preparation, R.M. and C.S.; writing—review and editing, R.M. and C.S.; visualization, R.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: The study was conducted in accordance with the Australian National Statement on Ethical Conduct in Human Research, and approved by the Human Research Ethics Committee of Southern Cross University (Approval number 2025/108, 7 July 2025).

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: The datasets presented in this article are not readily available because the data must be kept private as part of the ethical approval. Requests to access the datasets should be directed to Carolyn Seton.

Acknowledgments: We would like to acknowledge the participation of the students in Programming 1, without whose help this study would not have been possible. During the preparation of this manuscript/study, the authors used Claude.ai for the purposes of expressing some parts of the results. The authors have reviewed and edited the output and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare that there are no conflicts of interest.

References

Becker, B. (2019). Parlez-vous java bonjour la monde hello world: Barriers to programming language acquisition for non-native English speakers. In *PPIG* (pp. 40–52). Psychology of Programming Interest Group. Available online: https://www.ppig.org/files/2019-PPIG-30th-becker.pdf (accessed on 19 November 2025).

Becker, B. A. (2021, June). The roles of computing terminology in non-computing disciplines. In *Proceedings of the 26th ACM conference on innovation and technology in computer science education V. 2* (p. 659). ACM. Available online: https://dl.acm.org/doi/10.1145/3456565.3460073 (accessed on 19 November 2025).

Becker, B. A., Denny, P., Prather, J., Pettit, R., Nix, R., & Mooney, C. (2021, February). Towards assessing the readability of programming error messages. In *Proceedings of the 23rd Australasian computing education conference* (pp. 181–188). ACM. Available online: https://dl.acm.org/doi/10.1145/3441636.3442320 (accessed on 19 November 2025).

Cambridge Dictionary. (2025, November). *Meaning of 'class' in English*. Available online: https://dictionary.cambridge.org/dictionary/english/class (accessed on 20 November 2025).

Chandler, P., & Sweller, J. (1992). The split-attention effect as a factor in the design of instruction. *British Journal of Educational Psychology*, *62*(2), 233–246. [CrossRef]

Cho, Y., & Bridgeman, B. (2012). Relationship of TOEFL iBT® scores to academic performance: Some evidence from American universities. *Language Testing*, *29*(3), 421–442. [CrossRef]

Cooper, G., & Sweller, J. (1987). Effects of schema acquisition and rule automation on mathematical problem-solving transfer. *Journal of Educational Psychology*, *79*(4), 347–362. [CrossRef]

Cowan, N. (2001). The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and Brain Sciences*, *24*(1), 87–114; discussion 114–185. [CrossRef] [PubMed]

Cunnings, I., & Fujita, H. (2024). Subject-verb dependency formation and semantic interference in native and non-native language comprehension. *Applied Psycholinguistics*, *45*(1), 81–109. [CrossRef]

Diethelm, I., & Goschler, J. (2014, November). On human language and terminology used for teaching and learning CS/informatics. In *Proceedings of the 9th workshop in primary and secondary computing education* (pp. 122–123). ACM. Available online: https://dl.acm.org/doi/10.1145/2670757.2670765 (accessed on 19 November 2025).

Guo, P. J. (2018). Non-native english speakers learning computer programming: Barriers, desires, and design opportunities. In *Proceedings of the 2018 CHI conference on human factors in computing systems—CHI '18* (pp. 1–14). ACM Press. Available online: http://dl.acm.org/citation.cfm?doid=3173574.3173970 (accessed on 18 February 2020).

Izu, C., Schulte, C., Aggarwal, A., Cutts, Q., Duran, R., Gutica, M., Heinemann, B., Kraemer, E., Lonati, V., Mirolo, C., & Weeda, R. (2019, December). Fostering program comprehension in novice programmers—Learning activities and learning trajectories. In *Proceedings of the working group reports on innovation and technology in computer science education* (pp. 27–52). ACM. Available online: https://dl.acm.org/doi/10.1145/3344429.3372501 (accessed on 20 December 2025).

Kalyuga, S., Ayres, P., Chandler, P., & Sweller, J. (2003). The expertise reversal effect. *Educational Psychologist*, *38*(1), 23–31. [CrossRef]

Klepsch, M., Schmitz, F., & Seufert, T. (2017). Development and validation of two instruments measuring intrinsic, extraneous, and germane cognitive load. *Frontiers in Psychology*, *8*, 1997. [CrossRef]

Lakens, D. (2013). Calculating and reporting effect sizes to facilitate cumulative science: A practical primer for *t*-tests and ANOVAs. *Frontiers in Psychology*, *4*, 863. [CrossRef]

Martono, N. P., & Ohwada, H. (2025). Learning programming for non-native english-speaking students: Insight from Japanese students. In *Proceedings of the 56th ACM technical symposium on computer science education V. 2* (pp. 1539–1540). Association for Computing Machinery. [CrossRef]

Mason, R. (2012). *Designing introductory programming courses: The role of cognitive load* [Doctoral dissertation, Southern Cross University]. Available online: https://researchportal.scu.edu.au/esploro/outputs/doctoral/Designing-introductory-programming-courses–the-role-of-cognitive-load/991012821409802368 (accessed on 19 November 2025).

Mason, R., & Seton, C. (2020, February). Assessing international students: The role of cognitive load. In *Proceedings of the twenty-second Australasian computing education conference* (pp. 160–166). ACM. Available online: https://dl.acm.org/doi/10.1145/3373165.3373183 (accessed on 19 November 2025).

Mason, R., & Seton, C. (2021, February). Leveling the playing field for international students in IT courses. In *Proceedings of the 23rd Australasian computing education conference* (pp. 138–146). ACM. Available online: https://dl.acm.org/doi/10.1145/3441636.3442316 (accessed on 19 November 2025).

Mason, R., Seton, C., & Cooper, G. (2016). Applying cognitive load theory to the redesign of a conventional database systems course. *Computer Science Education*, *26*(1), 68–87. [CrossRef]

Miller, G. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, *63*(2), 81–97. [CrossRef]

Moore, B., & Waters, A. (2020). The effect of linguistic modification on English as a second language (ESL) nursing student retention. *International Journal of Nursing Education Scholarship*, *17*(1), 20190116. [CrossRef]

Morrison, B. B., Margulieux, L. E., Ericson, B., & Guzdial, M. (2016). Subgoals help students solve parsons problems. In *Proceedings of the 47th ACM technical symposium on computing science education* (pp. 42–47). Association for Computing Machinery. Available online: https://doi-org.ezproxy.scu.edu.au/10.1145/2839509.2844617 (accessed on 16 November 2025).

Munasinghe, B., Bell, T., & Robins, A. (2021, February). Teachers' understanding of technical terms in a computational thinking curriculum. In *Proceedings of the 23rd Australasian computing education conference* (pp. 106–114). ACM. Available online: https://dl.acm.org/doi/10.1145/3441636.3442311 (accessed on 19 November 2025).

Paas, F., Tuovinen, J. E., Tabbers, H., & Van Gerven, P. W. M. (2003). Cognitive load measurement as a means to advance cognitive Load Theory. *Educational Psychologist*, *38*(1), 63–71. [CrossRef]

Power, K. (2023, March). *Managing cognitive load for EAL - and all - students*. Available online: https://www.timeshighereducation.com/campus/managing-cognitive-load-eal-and-all-students (accessed on 20 November 2025).

Prather, J., Reeves, B. N., Denny, P., Leinonen, J., MacNeil, S., Luxton-Reilly, A., Orvalho, J., Alipour, A., Alfageeh, A., Amarouche, T., Kimmel, B., Wright, J., Blank, M., & Barbre, G. (2025). Breaking the programming language barrier: Multilingual prompting to empower non-native English learners. In *Proceedings of the 27th australasian computing education conference* (pp. 74–84). ACM.

Qualtrics. (2025). *Qualtrics [Software]*. Available online: https://www.qualtrics.com (accessed on 19 November 2025).

Raj, A. G. S., Zhang, H., Abhyankar, V., Mukerjee, S., Zhang, E., Williams, J., Halverson, R., & Patel, J. M. (2019, November). Impact of bilingual CS education on student learning and engagement in a data structures course. In *Proceedings of the 19th koli calling international conference on computing education research* (pp. 1–10). ACM. Available online: https://dl.acm.org/doi/10.1145/3364510.3364518 (accessed on 19 November 2025).

Roussel, S., Joulia, D., Tricot, A., & Sweller, J. (2017). Learning subject content through a foreign language should not ignore human cognitive architecture: A cognitive load theory approach. *Learning and Instruction*, *52*, 69–79. [CrossRef]

Scherer, R., Siddiq, F., & Sánchez Viveros, B. (2020). A meta-analysis of teaching and learning computer programming: Effective instructional approaches and conditions. *Computers in Human Behavior*, *109*, 106349. [CrossRef]

Schudde, L., Callahan, R., & Kwon, Y. (2023). *Language and postsecondary trajectories: How "ever-english learner" status shapes college student outcomes*. The University of Texas at Austin. Available online: https://repositories.lib.utexas.edu/handle/2152/121163 (accessed on 20 December 2025).

Simon, H., & Gilmartin, K. (1973). A simulation of memory for chess positions. *Cognitive Psychology*, *5*(1), 29–46. [CrossRef]

Skulmowski, A., & Rey, G. D. (2020). Subjective cognitive load surveys lead to divergent results for interactive learning media. *Human Behavior and Emerging Technologies*, *2*(2), 149–157. [CrossRef]

Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. *Cognitive Science*, *12*(2), 257–285. Available online: http://onlinelibrary.wiley.com/doi/10.1207/s15516709cog1202_4/abstract (accessed on 11 April 2011). [CrossRef] [PubMed]

Sweller, J. (1994). Cognitive load theory, learning difficulty, and instructional design. *Learning and Instruction*, *4*(4), 295–312. [CrossRef]

Sweller, J. (2010). Element interactivity and intrinsic, extraneous, and germane cognitive load. *Educational Psychology Review*, *22*(2), 123–138. [CrossRef]

Tamminen, J., Toiviainen, P., Buiatti, M., Feurra, M., Sarasso, P., Rosanova, M., & Bortoletto, M. (2025). Extended cognitive load induces fast neural responses leading to commission errors. *eNeuro*, *12*(2), ENEURO.0354-24.2024. [CrossRef]

Xie, B., Loksa, D., Nelson, G. L., Davidson, M. J., Dong, D., Kwik, H., Tan, A. H., Hwa, L., Li, M., & Ko, A. J. (2019). A theory of instruction for introductory programming skills. *Computer Science Education*, *29*(2–3), 205–253. [CrossRef]

Zaiontz, C. (2024). *Real statistics using excel*. Available online: https://real-statistics.com (accessed on 4 March 2026).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.