Journal of Computer Languages 86 (2026) 101384

Contents lists available at ScienceDirect

Journal of Computer Languages

journal homepage: www.elsevier.com/locate/cola

Teaching introductory programming in top universities: A global study of languages, paradigms, assessment, and AI

João Pedro Gomes [a,*], Redney Monteiro [b], André Silva Moreira [b], Vítor Mendonça [c], Anabela Pinho [b], Carlos R. Cunha [d]

[a] Instituto Politécnico de Bragança, Campus de Santa Apolónia, 5300-253, Bragança, Portugal
[b] Research Centre in Digitalization and Intelligent Robotics (CeDRI), Instituto Politécnico de Bragança, Campus de Santa Apolónia, 5300-253, Bragança, Portugal
[c] Applied Management Research Unit (UNIAG), Instituto Politécnico de Bragança, Campus de Santa Apolónia, 5300-253, Bragança, Portugal
[d] Research Centre in Digitalization and Intelligent Robotics (CeDRI), SusTEC, Instituto Politécnico de Bragança, Campus de Santa Apolónia, 5300-253, Bragança, Portugal

ARTICLE INFO

Keywords:
Programming languages
Programming paradigms
Assessment
AI
Computer science
Higher education
Curriculum design

ABSTRACT

This study examines the first two introductory computer science courses (CS1 and CS2) at the world's top universities, focusing on the choice of programming languages and paradigms, assessment types and weights, and policies regarding student use of AI tools. Data were compiled from 150 programming courses across 83 institutions ranked in the Shanghai 2023 Global Ranking of Academic Subjects for Computer Science & Engineering. The analysis reveals that the prevailing languages are Python in CS1 (32.5 %) and Java in CS2 (56.7 %). Typical transitions from CS1 to CS2 involve moving from Python or C to Java. There are some regional preferences: Asian institutions favor lower-level languages like C and C++, whereas European universities explore functional programming languages like Haskell. Regarding programming paradigms, CS1 emphasizes imperative approaches, sometimes introducing object-oriented concepts later in the course, while CS2 consolidates object-oriented programming. However, some CS1 courses adopt a functional paradigm. Course assessments are typically divided into several categories, with a preference for exams and various types of continuous assessments. However, there is also significant use of projects, quizzes, and labs. Examinations usually have the highest weight in the final grade. Course-level policies on students' use of generative AI tools reveal very different approaches, from complete prohibition to active encouragement. These findings offer valuable insights into how leading universities approach programming education in a multidimensional view that includes languages, paradigms, assessments, and AI tools policies.

1. Introduction

The Teaching of Programming plays a fundamental role in training professionals in computer science. More than just technical knowledge, programming courses provide an environment for developing transversal skills such as abstraction, logical thinking, and problem-solving. The first introductory programming course, often referred to as Computer Science 1 (CS1), focuses on the fundamentals of programming and computational thinking [1,2]. The subsequent course, hereafter referred to as Computer Science 2 (CS2), has less consensus on its specific content and may include advanced programming topics, new paradigms, algorithms, data structures, or other concepts, depending on the institution [1]. In CS1, the language used can significantly influence students' learning, shaping their preparation for more advanced courses and future professional challenges.

There is no consensus on the best approach to teaching introductory programming [3], as there are several factors that influence this choice, such as the ease of learning the language, pedagogical reasons, the programming paradigm, the relevance of the language in the job market, and even the individual preferences and previous experience of the instructors. Several studies have presented sets of objective criteria for consideration when selecting programming languages to adopt [4–7].

The choice of programming language appears to influence student learning, as some languages, due to their complexity in syntax and

* Corresponding author.
E-mail addresses: jpgomes@ipb.pt (J.P. Gomes), redneymonteiro@ipb.pt (R. Monteiro), andre-moreira@ipb.pt (A.S. Moreira), mendonca@ipb.pt (V. Mendonça), apinho@ipb.pt (A. Pinho), crc@ipb.pt (C.R. Cunha).

https://doi.org/10.1016/j.cola.2025.101384
Received 3 September 2025; Received in revised form 30 November 2025; Accepted 23 December 2025
Available online 24 December 2025

semantics, may overwhelm beginners and distract their focus from understanding fundamental programming concepts [8]. Such complexity and distraction can impact students' confidence in and perception of their programming skills.

The programming paradigm used in the courses also influences both the learning process and the way problems are solved. Imperative programming, based on sequences of instructions that alter states, leads students to focus on the details of implementation, favoring algorithmic reasoning. In object-oriented programming, the focus is on modelling a problem as a set of entities that relate to each other, reflecting real-world systems, and allows the development of abstraction and modularity skills. In functional programming, focused on mathematical data transformations, students develop the ability to decompose and generalize problems. Learning to work with each one, understanding their strengths and applicability, empowers students to develop their programming abilities. It is essential that curricula address multiple paradigms, allowing the integration of complementary perspectives [9] and providing students with tools for their future careers, at a time when software development often draws on concepts from different programming paradigms [10].

CS Curricula 2023 [11], a curriculum guidance for computer science courses, developed in collaboration with the Association for Computing Machinery (ACM), the IEEE-Computer Society (IEEE-CS), and the Association for the Advancement of Artificial Intelligence (AAAI), defines core competencies and essential knowledge areas, prioritizing learning outcomes and pedagogical flexibility over specific programming languages. Regarding the Software Development Fundamentals (SDF), the document mentions that SDF is "generally covered in introductory courses, often called CS1 and CS2" and that the SDF knowledge area "assumes a contemporary programming language with built-in support for common data types including associative data types like dictionaries/maps as the vehicle for introducing students to programming (e.g., Python, Java). However, this is not to discourage using older or lower-level languages for SDF— the knowledge units […] can be suitably adapted for the actual language used." [11]. This supports the focus on CS1 and CS2 as the natural context to investigate language, paradigm, and pedagogical trends.

In addition to choosing languages and paradigms, one of the challenges institutions currently face is determining how to effectively integrate generative AI into an educational context. On the one hand, it offers unique opportunities for learning programming, enhancing learning by acting as tireless and always available personalized tutors, allowing students to follow their own pace. Fan et al. [12] demonstrate how the use of AI-assisted pair programming significantly increases intrinsic motivation, reduces anxiety in coding, and improves student performance. On the other hand, a passive attitude from the student and excessive dependence on these tools can negatively affect autonomy, performance, and the ability to solve problems without assistance [13]. This requires teachers and institutions to define how they intend to integrate these tools into the education system and to establish clear guidelines for students on how they can use them, both for learning purposes and to maintain academic integrity in the assessment process.

There is also considerable diversity in assessment methodologies between courses, institutions, and contexts. Even the medium on which the exam is taken is not agreed upon, and there is still frequent debate about whether coding exams should be taken with pen and paper or on a computer [14]. Formative assessment, on the other hand, has several advantages, motivating and engaging students more and reducing the anxiety associated with final exams [15].

The best universities often set trends and best practices in higher education. Identifying the programming languages, paradigms, and methodologies that they use in CS1 and CS2 enables us to understand patterns that could serve as a benchmark for decision-makers at other institutions and future research, contributing to an overall improvement in programming education and, consequently, in computer science education.

This article extends and revises a conference paper presented at WorldCIST 2025 [16]. The original study primarily focused on the distribution of programming languages in CS1 and CS2, analyzing language transitions between courses, and regional variations across top-ranked universities. The selection of these universities was based on the Shanghai Ranking, due to its established international recognition and focus on objective, transparent, and research-based metrics, ensuring that the study reflects high-quality institutions representative of different regions and educational contexts. The subject-specific 2023 Global Ranking of Academic Subjects about Computer Science & Engineering was used to identify the best institutions in the field relevant to the study.

In this revised version, the scope of analysis has been significantly expanded to address three topics: (i) how are programming paradigms distributed in CS1 and CS2, and what alignment patterns emerge; (ii) what assessment categories and grade weights are used, and how do they differ between CS1 and CS2; and (iii) what types of course-level policies exist for the use of generative AI by the students, an aspect increasingly relevant to programming education. These additions offer a more comprehensive view of the various approaches employed in introductory programming courses.

This study complements previous regional and survey-based research by applying a systematic analysis to primary data sources from a worldwide selection of top universities. While previous studies have successfully employed document analysis in specific national contexts or relied on voluntary surveys of instructors for global coverage, this research leverages the objective metrics of the Shanghai Ranking (2023). This approach provides a baseline of curricular trends specifically in the world's leading computer science departments. Furthermore, this study offers a multidimensional view of the curriculum that goes beyond isolated language statistics, adding new dimensions and simultaneously examining languages, paradigms, assessment, and AI policies.

The following sections review related works and contextualize the study (Section 2), describe the methodology and used data (Section 3), present and discuss the results (Section 4), and conclude with a summary of the main contributions and suggestions for future research (Section 5).

2. Related work

The choice of programming languages for introductory programming courses has been the subject of study at various higher education institutions. Several previous works present a comprehensive overview of the languages and practices adopted in different regional and institutional contexts, providing insights into the trends and criteria used to select these languages.

2.1. Regional and national surveys

In 2017, Murphy, Crick and Davenport [17] surveyed the introductory programming courses offered in 80 Computer Science degree programs at universities in the United Kingdom, analyzing the languages, paradigms, and tools used in these courses. Java appears to be the dominant language, being used in more than half of the courses, despite Python being considered more accessible for both teaching and learning, and the fact that students are generally not programming experts. The C family of languages (C, C++, and C#) are used in almost a third of cases. The prevailing paradigms were object-oriented (50 %), followed by procedural (33.8 %), and, lastly, functional (8.75 %).

In 2018, Avouris [18] also surveyed higher education institutions in Greece, using a public database containing information on programs of study, courses and recommended textbooks, with a focus on pedagogical approaches, programming languages, and tools used. C was identified as the most used language in the programming courses analyzed, present in 37 % of the courses, followed by MATLAB (17 %) and Basic (11 %). The

remaining languages had representations of 6 % or less, reflecting the differences concerning educational systems in other countries.

Additionally, in 2018, Ezenwoye [19] mapped the most popular languages in introductory computer science courses in higher education and the factors that influenced their adoption in the North American context. Java was identified as the most popular language (41.94 %), followed by Python (26.45 %) and C++ (19.35 %). The study analyzed the factors considered important when choosing a language, with the most frequently mentioned being the language's characteristics (26.19 %), ease of learning (18.81 %), and job opportunities (14.76 %). The study also shows that, most of the time (58.27 %), curricula have three additional programming courses beyond the introductory one.

In 2019, Becker [20] conducted an exhaustive study of introductory programming courses in higher education institutions in Ireland, including universities, institutes of technology, and private colleges. The most widely used language identified was Java, which is present in 49 % of courses, including a course that starts with Scratch. The following languages were Python (28 %), JavaScript (18 %), and C and C# (both with 10 %). The study also assessed whether courses used only one programming language and found that when courses used Python or Java, they typically did not use any other language. Examining the reported paradigms, the responses were categorized as object-oriented (43 %), procedural (23 %), functional (15 %), logical (8 %), other or mixed (8 %), and 5 % used no particular paradigm. The main paradigms corroborate the results of the 2016 UK survey mentioned above [17]. The study also explored the reasons for choosing the languages: relevance to the industry, availability/cost for students, and pedagogical benefits. Other assessments were course structure, instructors' perceptions of students' difficulties in learning the language, and perceptions of the usefulness of the language for teaching fundamental programming concepts. The instructors considered C++, Java, and C to be the most difficult to learn, and all the languages were considered helpful in learning the fundamental concepts.

Realizing the limited literature available on the languages used in introductory programming courses in management information systems (MIS) curricula, in 2021 Smith and Jones [21] surveyed US business college MIS programs. The findings revealed that the most popular languages in use were Python (36 %), followed by Java (21 %), Visual Basic (13 %), and C# (11 %). The study showed that the adoption of Python in most courses was made very recently, and that the adoption of Visual Basic has been declining significantly, whereas it was previously the most widely used language. When asked if there were plans to change the language, 85 % responded negatively.

Siegfried et al. have updated and expanded the historical Reid List, created by Richard Reid in the early 1990s to track the programming languages in introductory computer science courses at universities across the United States. The most recent analysis [22], published in 2021, shows that CS1 language choice is predominantly Java, Python, C++, and C, with one of these languages used by 88 % of the surveyed institutions. There has been a consistent increase in Python, mainly to the detriment of Java and C++. In CS2, the main choices are even clearer, with a significant predominance of Java, followed at a distance by C++. The study also identifies patterns of continuity and transition between CS1 and CS2, showing that most of the programs retain the same language across both courses (54 %), and the most common transition is from Python to Java.

2.2. Global surveys

More recently, in 2024, Mason et al. [23] conducted a global survey of introductory programming courses, involving instructors from 18 countries across six continents. The study identified Python and Java as the most widely used programming languages, each accounting for around 40 % of the courses analyzed. Other languages, such as C++ and C, also appeared with significant frequency (12 % and 10 % respectively). Also mentioned, but more marginally, were JavaScript, Processing, Racket, Scala, Go, and Bash, the latter two used in conjunction with another language. Among the factors influencing the choice of languages, pedagogical benefits were the most important, followed by platform independence and cost/accessibility for students. Relevance to industry was also mentioned, but to a lesser extent than in previous regional studies. The study also revealed regional variations, with Python dominating the Americas and Java being more prevalent in Oceania. They also asked what the taught paradigm was, and the results show that the most common was procedural (54 %), followed by object-oriented (41 %) and functional (5 %). Additionally, it examined programming environments and the impact of remote teaching during the pandemic. Nevertheless, it was pointed out that the responses collected were "very US-dominated" (54 % of the institutions), a limitation of the study.

These studies highlight the diversity of practices in teaching introductory programming, influenced by pedagogical, regional, and institutional factors. However, most rely on data reported from voluntary surveys. Analyzing the syllabi of the best universities in the Shanghai Ranking offers a methodologically distinct opportunity to identify patterns in the practices of the most prestigious and globally relevant institutions. This includes not only consolidating data on programming language and paradigm to track temporal shifts but also providing new insights into assessment practices and the emerging role of AI tools.

3. Methodology

The overall methodology followed a structured, multi-step process involving sampling and scoping, data acquisition, classification and categorization, and analysis (Fig. 1). Acquiring the data needed for this study proved laborious and presented a few obstacles. First, institutions were selected using the 2023 Shanghai Ranking for Computer Science and Engineering. The most relevant undergraduate program at each institution was then identified through institutional websites, followed by the selection of the first two introductory programming courses, designated CS1 and CS2, according to their chronological position in the curriculum. Detailed courses information was collected and organized, and additional searches were conducted when data was ambiguous or insufficient, to identify programming languages, paradigms, assessments methods, and policies regarding students' use of generative AI. Finally, a descriptive statistical analysis was performed on the structured and coded data.

3.1. Selection of institutions from the Shanghai Ranking

The selection of universities was based on the Shanghai Ranking, one of the most influential world rankings. The 2023 Global Ranking of Academic Subjects (GRAS) in the discipline of Computer Science and Engineering was used to identify the most relevant institutions to programming education. This ranking uses a "range of objective academic indicators and third-party data to measure the performance of the world's universities in their respective disciplines, covering five broad categories of evaluation, such as World-Class Faculty, World-Class Production, High-Quality Research, Research Impact and International Collaboration" [24].

While there are other global rankings, such as Times Higher Education (THE) or QS World University Rankings (QS), these include some subjective measures, such as academic and employer reputation through surveys. The Shanghai Ranking approach is more objective, quantitative and research-focused, based on the quality and impact of scientific production, international collaboration and quantity and quality of faculty and researchers involved in Computer Science Projects.

3.2. Identification of CS1 and CS2 courses

The next step involved identifying the most significant degree program at each institution. The institutions' websites were consulted to

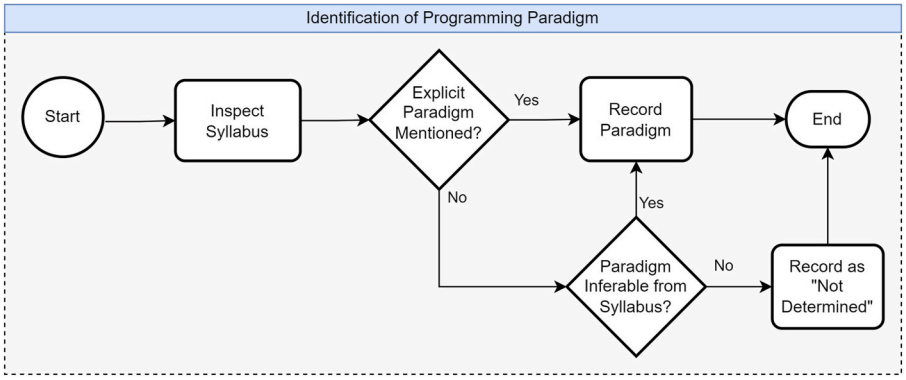

Fig. 1. Flowchart of the decision process for classifying programming paradigms used in CS1 and CS2 courses.

identify undergraduate programmes with "Computer Science" in their name or with thematically related names (e.g., Informatics) and collected information on their curricular structure. When multiple programmes existed in the field, the one most directly related to "Computer Science" was selected based on the name and analysis of the syllabus, prioritizing those with a stronger focus on the programming subject.

Then, based on the syllabus, the existing programming-related courses were identified. The two introductory courses usually have the term "Programming" in their names (e.g., "Introduction to Programming", "Programming I", "Algorithm and Programming", "Object-Oriented Programming"); however, this is not always the case (e.g., "Computational Thinking", "Object-Oriented Development").

The first two programming courses were identified based on their chronological position within the curriculum. Typically, the first programming-related course is introduced in the first term of the degree, and the second course in the second term.

For each course, the description, topics/content, learning outcomes, assessment criteria, learning tools, and information about the use of generative AI were collected and recorded. Data was stored and organized using a spreadsheet.

The collected information was reviewed to identify the programming languages. When information was incomplete or ambiguous, additional searches were conducted, including contacting faculty staff via email. To mitigate language barriers, the official English versions of the websites were referenced when available, and machine translation tools were used when necessary.

It was possible to successfully identify the programming languages in 150 courses, comprising 83 introductory programming courses and 67 s-level programming courses, from a total of 83 institutions across 14 countries, representing four continents: North America, Asia, Europe, and Oceania (Table 1).

Table 1
Number of courses successfully verified by country.

Country	Total Institutions	Total Courses	CS1	CS2
United States	27	54	27	27
China	23	36	23	13
Australia	8	14	8	6
United Kingdom	7	14	7	7
Canada	5	10	5	5
Singapore	3	6	3	3
South Korea	2	2	2	0
Switzerland	2	4	2	2
Saudi Arabia	1	2	1	1
Denmark	1	1	1	0
China-Macau	1	2	1	1
Belgium	1	1	1	0
Germany	1	2	1	1
Finland	1	2	1	1
Total	*83*	*150*	*83*	*67*

3.3. Classification of languages, paradigms, and assessments

A descriptive statistical analysis was conducted to identify the distribution of the different programming languages in CS1 and CS2, the progression from the first to the second, and the influence of geographical factors and institutional ranking. Tables and graphs were created to illustrate the results and support further discussion.

The identification of programming paradigms relied not only on the programming language but also on the course description, topics, and learning outcomes. This was necessary, since some languages are multi-paradigm, requiring identification of the paradigm used in the course. In that case, the classification used a structured decision process (Fig. 2): (i) if there is an apparent reference to a paradigm in the syllabus, then record the paradigm (e.g., "Object-oriented software development." → object-oriented paradigm); (ii) if there are no direct reference and the language is multi-paradigm, then infer from the syllabus (e.g., "Java programming language is used to introduce fundamental programming concepts, including conditionals, loops, arrays, functions" → imperative paradigm); (iii) if no evidence was available, then the paradigm was recorded as *Not Determined* (e.g., "Programming 2 - This class is the second course in computer programming, taught using Java." → Not Determined). For each course, all paradigms supported by textual evidence were recorded.

Assessment methods were grouped into broader categories (see Table 5 in the Results section for details). Then, for each course, all the assessment categories used by it were marked.

3.4. Data availability and ambiguities

A significant limitation was the variability in data availability and clarity across institutions. No data from some institutions seems to be publicly available, and only partial data was found from others. The information provided differed significantly, and it was not possible to access the syllabus at some institutions because it was not publicly available or because the language barrier made it impossible to understand how to obtain the information, even with the assistance of translators. In other cases, the programming language was not explicitly indicated, even with access to the syllabus.

Of the 100 institutions analyzed, there were 17 institutions, all in China, with no accessible information about any of the initial courses due to the absence of publicly accessible syllabi. In addition, there were 16 institutions where the CS2 programming language could not be verified.

Another point to note is that some courses use multiple programming languages. In seven cases, two programming languages were identified as being taught within the same course, while in three courses students could choose one out of two or three languages. For these ambiguous cases, a structured decision tree was applied, which will be explained in the related sections later.

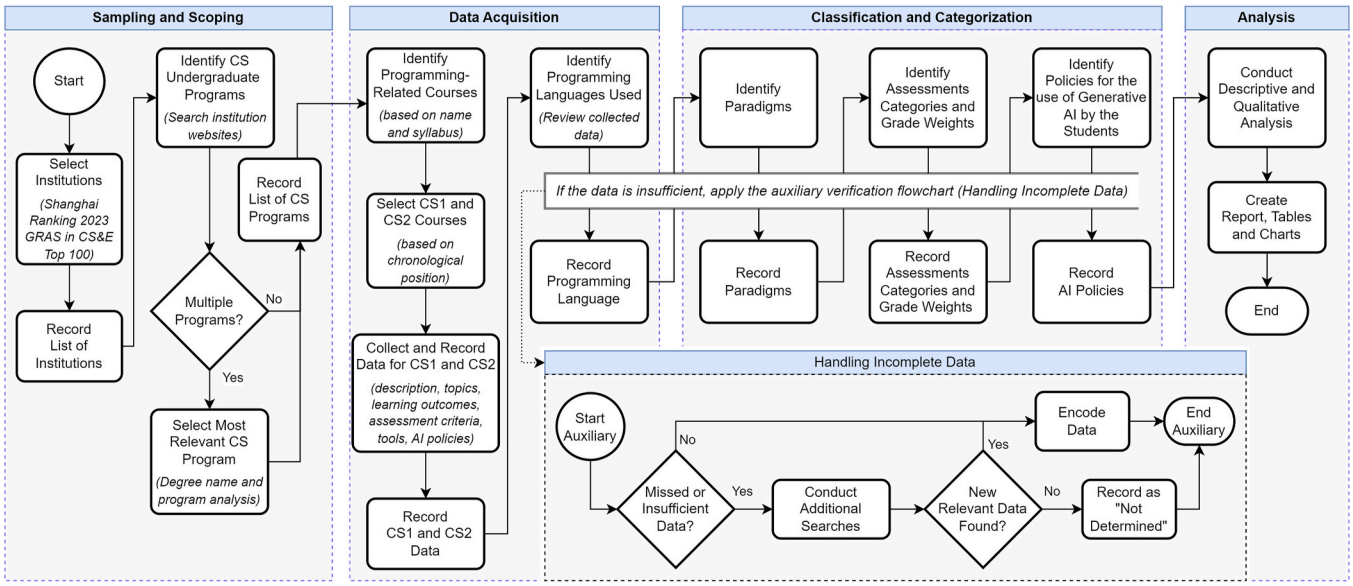

Fig. 2. Flowchart of the decision process for classifying programming paradigms used in CS1 and CS2 courses.

4. Results and discussion

This section analyses the programming languages used in introductory computer science courses, focusing on their distribution across undergraduate programs, the transitions between languages from CS1 to CS2, and the regional variations in the adoption of programming languages.

4.1. Distribution of programming languages

Table 2 presents the programming languages used in the two introductory computer science courses, along with their frequency of use in CS1 and CS2. Additionally, it shows the number of times when both courses within the same undergraduate program have the same language.

Table 2
Distribution of programming languages used in introductory courses of computer science undergraduate programs.

Language	Total	CS1	CS2	Both
Java	54 (36.0 %)	16 (19.3 %)	38 (56.7 %)	10
Python	32 (21.3 %)	27 (32.5 %)	5 (7.5 %)	1
C	24 (16.0 %)	14 (16.9 %)	10 (14.9 %)	0
C++	16 (10.7 %)	10 (12.0 %)	6 (9.0 %)	2
Haskell	4 (2.7 %)	3 (3.6 %)	1 (1.5 %)	0
C AND C++	3 (2.0 %)	0 (0.0 %)	3 (4.5 %)	0
Assembly	2 (1.3 %)	2 (2.4 %)	0 (0.0 %)	0
JavaScript	2 (1.3 %)	1 (1.2 %)	1 (1.5 %)	0
(C AND Python) OR Python	1 (0.7 %)	1 (1.2 %)	0 (0.0 %)	0
Haskell AND Kotlin/Java AND C	1 (0.7 %)	1 (1.2 %)	0 (0.0 %)	0
Java OR Python OR Matlab	1 (0.7 %)	1 (1.2 %)	0 (0.0 %)	0
OCaml	1 (0.7 %)	1 (1.2 %)	0 (0.0 %)	0
Python OR C	1 (0.7 %)	1 (1.2 %)	0 (0.0 %)	0
Python OR F#	1 (0.7 %)	1 (1.2 %)	0 (0.0 %)	0
Python	1 (0.7 %)	1 (1.2 %)	0 (0.0 %)	0
Racket	1 (0.7 %)	1 (1.2 %)	0 (0.0 %)	0
Ruby	1 (0.7 %)	1 (1.2 %)	0 (0.0 %)	0
Source/JavaScript	1 (0.7 %)	1 (1.2 %)	0 (0.0 %)	0
C#	1 (0.7 %)	0 (0.0 %)	1 (1.5 %)	0
Javascript	1 (0.7 %)	0 (0.0 %)	1 (1.5 %)	0
OCaml OR (C AND C++ AND Assembly)	1 (0.7 %)	0 (0.0 %)	1 (1.5 %)	0
Total	*150*	*83*	*67*	*13*

Most courses use a single mandatory programming language (94 %). However, six courses use two languages (C and C++, C, and Python) or even more (C and C++ and Assembly, Haskell and Kotlin/Java, and C). In five cases, students can choose the course they enroll in, determining the programming language they will learn (Java or Python or MATLAB, Python or C, Python or F#, Python or Python and C, OCaml or C and C++ and Assembly).

Java, Python, C, and C++ are the most chosen programming languages for the two initial courses. Together, they are mandatory in 131 (87.3 %) courses. This count rises to 137 (91.3 %) of the courses when considering elective options.

Java is the most widely used language in the two introductory programming courses, which are mandatory in 55 (36.7 %) of the courses analyzed. It is significantly the most common language in CS2, used in 38 courses (56.7 %), and the second most common in the initial programming course in at least 16 (19.3 %) courses. Of the 13 institutions that use the same language in both introductory courses, 10 use Java.

Python is the second most common language, leading as the choice for introductory programming courses with at least 27 occurrences (32.5 %). However, it is less frequently chosen for CS2, dropping to fourth place with only 5 (7.5 %).

C and C++, sometimes used together, are the third and fourth most used languages, respectively, with a balanced presence in CS1 and CS2. As a mandatory language, C appears in 27 courses (18.0 %), being used 14 times in CS1 (16.9 %) and 13 times in CS2 (19.4 %). C++, on the other hand, is mandatory in 19 (12.7 %) courses, with 10 in CS1 (12.7 %) and 9 in CS2 (13.5 %). Two institutions use C++ in both courses.

In total, 15 programming languages have been identified. Besides the ones mentioned, OCaml, JavaScript, Assembly, and Haskell are taught in 2–5 courses, in that order. Additionally, C#, F#, Kotlin, MATLAB, Racket, Ruby, and Source, a family of sublanguages of JavaScript, are only mentioned once.

Compared to previous regional studies, these results corroborate the trend identified by Siegfried et al. [22], regarding the growing adoption of Python in CS1 to the detriment of Java and C++. It is now evident that Python has replaced Java as the first programming language, at least at top universities. In CS2, the clear predominance of Java identified is also aligned with what had already been observed in Siegfried et al. [22]. However, the present analysis reveals a more pronounced trend on the first language choice at elite institutions compared to the global survey of Mason et al., where Python and Java each accounted for

approximately 40 % of courses.

The temporal evolution in the preference for the first programming language is also visible when comparing these results with earlier studies. In 2017, Murphy et al. [17] reported the dominance of Java (>50 %) in UK universities, also reported by Ezenwoye [19] in the US in 2018 (Java 41.94 %, Python 26.45 %). This shift in preference for Python in CS1 observed in the present study reinforces the trend already noted by Smith and Jones [21] in US MIS programs in 2021 (Python 36 %, Java 21 %), confirming that the adoption of Python has consolidated in recent years, particularly in introductory programming courses at leading institutions.

The prevalence of Java, Python, C, and C++ in introductory programming courses reveals a preference for imperative and object-oriented programming paradigms. This choice is not surprising, as these are paradigms widely used in industry.

The contrasting distribution of Python in CS1 and Java in CS2 highlights a consistent pattern, but the available data does not allow for an explanation of why this occurs. Some well-known technical differences between the languages are potential influences, such the smoother learning curve and support for multiple paradigms of Python, or Java's emphasis on object-oriented concepts from the very beginning. However, other factors may also have influence, including teaching preferences, institutional requirements, explicit learning objectives, perceived alignment with employment expectations, or old departmental practices. To determine which factors actually guide the choice of languages in CS1 and CS2, it would be necessary to collect additional types of data, such as interviews with teachers and coordinators or qualitative analyses of course documentation. Without additional information, any attempt to justify these differences remains speculative.

The initial descriptive analysis suggests that universities with the highest rankings tend to use Python in CS1 (average ranking of approximately 24.5). Java is widely used in both CS1 and CS2; but universities that use it in CS1 tend to have a lower average ranking than those that use Python. However, inferential statistical analysis using the Kruskal-Wallis test did not indicate a statistically significant difference or correlation between university rankings and the programming languages chosen for CS1 ($H = 7.85$, $df = 4$, $p = 0.097$) and CS2 ($H = 2.65$, $df = 3$, $p = 0.449$). Therefore, based on the available data, there is insufficient statistical evidence to suggest that the choice of programming language is correlated with the university's position in the ranking. For these analyses, only universities with unique programming languages in each course were considered, so the lack of statistical significance observed in the tests may be partially explained by the limited sample size.

4.2. Transition between CS1 and CS2

Only the 61 institutions where data were available for both courses were included in the analysis of transitions between the programming languages in CS1 and CS2 (Fig. 3). Therefore, not all the data in Table 2 is represented in this figure.

The most common transition is from Python to Java, occurring in 12 institutions, followed by C to Java (6 institutions), Python to C (5), and Python to C++ (4).

Several less common transitions exist, such as Haskell to Java, Assembly to Java, and Ruby to C#. Additionally, there are isolated cases where institutions start with less conventional languages, such as Racket or OCaml, and then transition to more traditional languages, like C or Java.

Using the same programming language in both courses is less common, observed in only 13 (21.3 %) institutions. As mentioned, Java is the language where this occurs most frequently, with ten instances. The

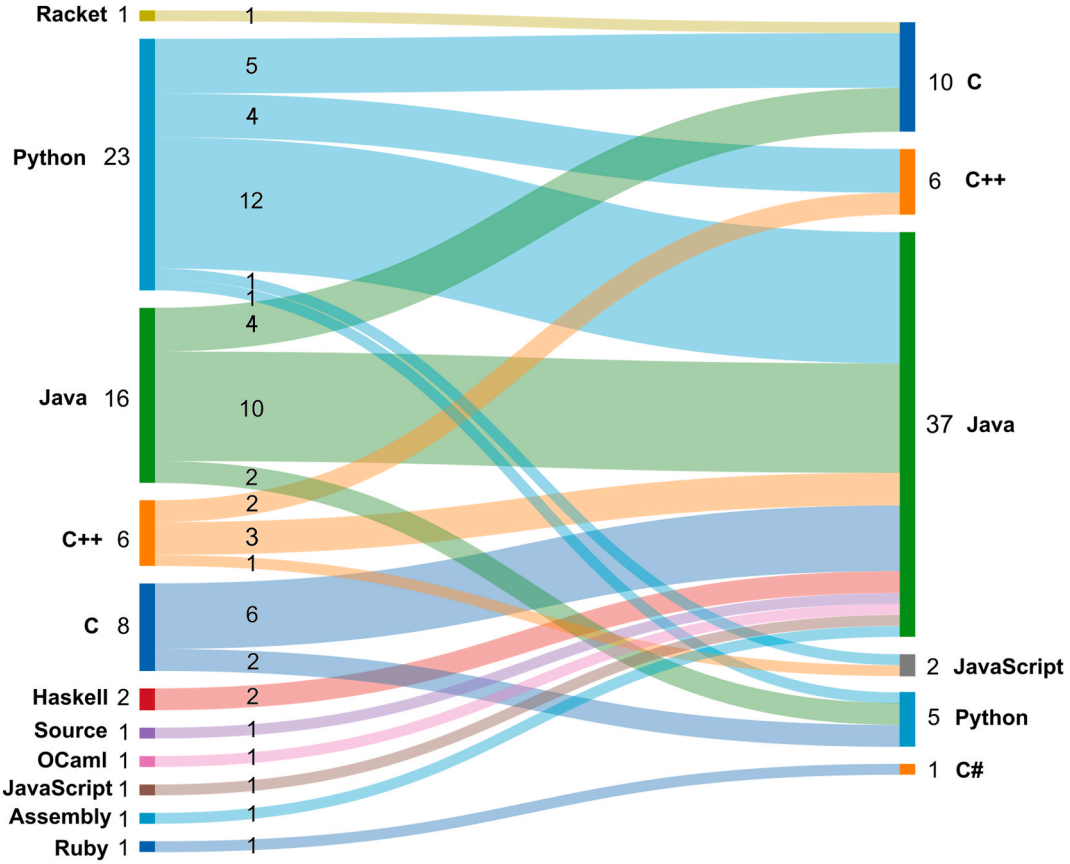

Fig. 3. Transition of programming languages between CS1 and CS2 courses. The width of each band represents the number of institutions transitioning from one language to another.

other situations happen with C++ twice and Python once.

The patterns observed show similarities to the study by Siegfried et al. [22] on the most frequent transition from CS1 to CS2, which also reported a move from Python to Java. However, there is a significant difference in the percentage of programs that maintain the same language between CS1 and CS2, since while Siegfried et al. [22] reported it to be 54 %, this study shows that in top institutions it is only 21.3 %.

4.3. Regional differences

To determine if there were regional differences in the choice of programming languages, the data from the institutions were combined by continent, considering only the five most represented languages in each region. This resulted in six different languages: Java, Python, C, C++, Haskell and JavaScript.

Fig. 4 visually illustrates these differences, showing the overall preferences of each region in CS1, CS2, and the combined results.

Overall, Java is prevalent in all regions, particularly in Europe and North America, where its adoption rates reach 42.9 % and 36.6 %, respectively. It is also significantly the most used language in CS2 across all regions. However, it is rarely used as a first language in Asia (6.7 %) and Oceania (11.1 %). Python is also widely used globally, mainly as an introductory language. Asia prefers traditional, low-level languages such as C and C++, which are taught in 48 % of institutions. On the other hand, Europe has only adopted C in 7.1 % of institutions, ignoring C++. Haskell is used as the first language in 17.6 % of European institutions. The inclusion of functional languages is not as evident in other regions and has no expression in Asia or North America.

Of the seven languages that only appear in one of the institutions, four are related to European institutions (C#, F#, Kotlin, Ruby), two are North American (MATLAB, Racket), and one is Asian (Source).

Fig. 5 shows two heatmaps that map the percentage distribution of languages used by each country, ordered geographically from west to east, to visualize regional differences in the choice of programming languages in Computer Science courses (CS1 and CS2). The maps show regional variations in the adoption of languages such as Python, Java, and C++. The comparison between the heatmaps also reveals that CS1 has a greater dispersion in the most used languages (e.g., the adoption of C in China). At the same time, for CS2, there is an apparent global adoption of Java as the preferred language.

It should be noted that these regional observations are based on limited samples in some countries (e.g., $n < 5$ in Belgium, Denmark, Finland, Germany, Saudi Arabia, Singapore, South Korea, and Switzerland) and should be interpreted cautiously.

The existence of regional variations has also been verified by the analysis of previous studies, but not necessarily the same ones. For example, C is the introductory language identified as the preferred choice by Asian institutions and had also been mentioned by Avouris [18] as the main language in Greek institutions, contrasting with the trend in Europe, which, according to the present study, prefers Python and Java.

The presence of functional languages in European institutions (17.6 % in CS1) had already been observed in previous studies. Murphy et al. [17] reported 8.75 % in the UK, and Becker [20] reported 15 % in Ireland, while Mason et al. [23] only mentioned 5 % globally.

4.4. Programming paradigms

Table 3 shows all the individual programming paradigms observed in the analyzed courses, along with the number of CS1 and CS2 courses that use each paradigm. When a single course adopts multiple paradigms, it is counted in each relevant category, so the total may exceed the number of different courses analyzed. Applied paradigms were identified in 109 courses, 53 of which were from CS1 and 56 from CS2.

In the introductory CS1 course, almost all courses use the imperative paradigm (~90 %). It is the only paradigm covered in 31 of the courses, predominantly using the Python language but also C in some cases. There is a significant presence of object orientation in about one-third

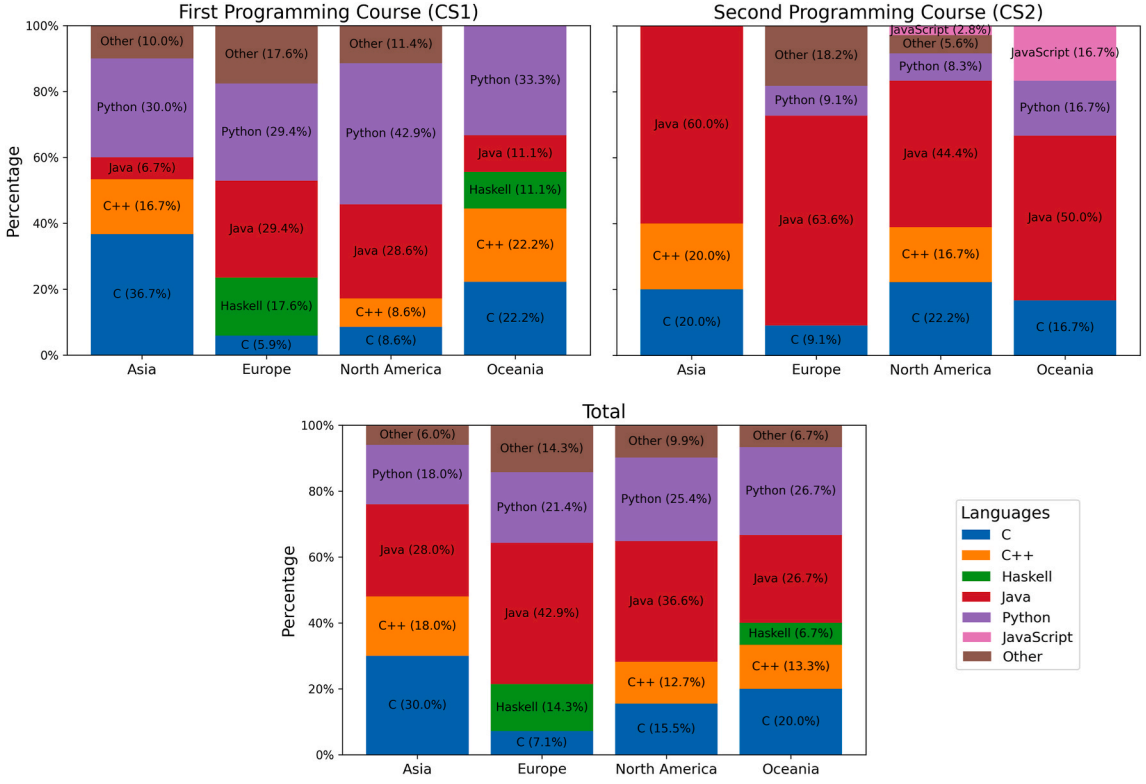

Fig. 4. Percentage distribution of the top programming languages across regions for the first programming course (CS1), the second programming course (CS2), and the overall total.

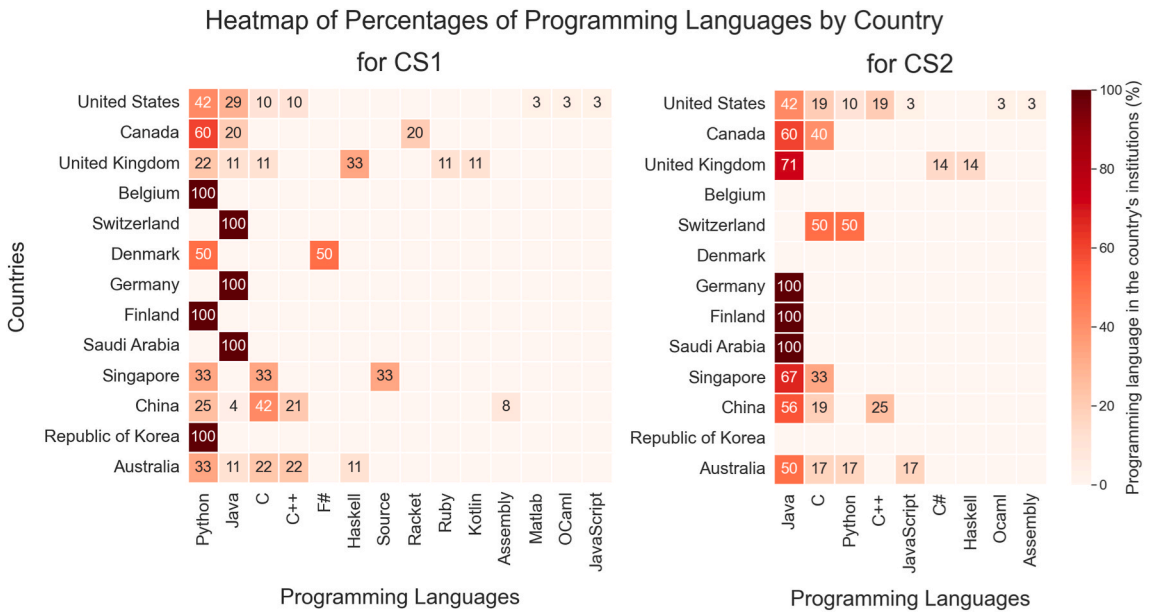

Fig. 5. Country distribution of programming languages in CS1 and CS2 courses. The heatmap highlights the prevalence of specific languages by country, with darker shades indicating higher usage.

Table 3
Distribution of individual paradigms used in CS1 and CS2. One course can have multiple paradigms.

Paradigm	Total courses	CS1 courses	CS2 courses
Imperative	109 (94.8 %)	53 (89.8 %)	56 (100 %)
Object-Oriented	71 (61.7 %)	21 (35.6 %)	50 (89.3 %)
Functional	9 (7.8 %)	8 (13.6 %)	1 (1.8 %)
Parallel	1 (0.9 %)	0 (0 %)	1 (1.8 %)
Total of different courses analyzed	*115*	*59*	*56*

Table 4
Distribution of multi-paradigms used in CS1 and CS2.

Paradigm	Total courses	CS1 courses	CS2 courses
Object-Oriented + Imperative	69 (60 %)	20 (33.9 %)	49 (87.5 %)
Functional + Imperative	2 (1.7 %)	1 (1.7 %)	1 (1.8 %)
Functional + Imperative + Object-Oriented	2 (1.7 %)	1 (1.7 %)	1 (1.8 %)
Imperative + Object-Oriented + Parallel	1 (0.9 %)	0 (0 %)	1 (1.8 %)
Total of different courses analyzed	*115*	*59*	*56*

Table 5
Distribution of the different types of assessment by category.

Assessment Category	Assessment Types
Test/Exam	Final Exam, Midterm Exam, Exam, Examination, Test, In-class Test, Term Test, Mid-Semester Test
Continuous Assessments	Assignments, Homework, Homework + Prep, Coursework, Problem Sets, Writing Sessions, Exercises, Continuous Assessment
Project/Portfolio	Project, Portfolio, Machine Project, Capstone project, Creative projects, Mini-projects, Midterm Coding Project, Final Coding Project
Quizzes	Quizzes, Tutorial Quiz
Lab/Practical	Laboratory, Lab, Lab Assessment, Lab Reports, Practical Assessment, Skills Assessment, Practical Skills Assessment, Practicals
Participation/ Attendance	Participation, Attendance, Recitation attendance, Workshops, In-Class Activities

(~36 %) of CS1 courses, all of which also involve imperative programming. Typically, object-oriented programming is introduced near the middle or at the end of the course. Six CS1 courses use functional programming as their primary paradigm, with Haskell, Racket, or OCaml as supporting languages. There are also explicit references to concurrent/parallel programming, using Java, in a single CS2 course.

In CS1, the dominant paradigm is imperative, using Python and C, while in CS2, the dominant paradigm is object-oriented, with Java and C++, a typical pedagogical transition.

The predominant transition flows between CS1 and CS2 on each program are from imperative to object-oriented. There is considerable continuity in the object-oriented paradigm between CS1 and CS2, given that many courses in CS1 have already transitioned from an imperative to an object-oriented approach. The transition from the functional paradigm in CS1 is almost always to object-oriented in CS2.

In CS1, most courses (~63 %) use only one paradigm, primarily imperative but also functional, with 22 multi-paradigm courses. In CS2, almost all courses (52, ~93 %) are multi-paradigm, primarily combining object-oriented with imperative programming, but four courses use a single paradigm (imperative).

The table of paradigm combinations in each course (Table 4), shows that in CS1, nearly two-thirds (~63 %) of courses employ only one paradigm, while in CS2, the vast majority (~91 %) are multi-paradigm. This is a typical progression, where a basic paradigm is introduced in the introductory course, followed by an increase in complexity through the integration of other paradigms in the second course.

In both CS1 and CS2, almost all multi-paradigm courses combine imperative and object-oriented paradigms. This combination prevails in both courses, but usually with the imperative paradigm dominating in CS1 and the object-oriented paradigm in CS2. Marginally, some options also include functional and parallel paradigms.

One of the unusual situations that stands out is the combination of contrasting languages, levels, and paradigms, namely functional programming in Haskell, functional and procedural programming in Kotlin, object-oriented programming in Kotlin and Java, imperative programming in C, and low-level programming in Assembly, among others. This approach is used in the Computing Practical 1 course at Imperial College London, one of the two courses analyzed that last longer than one semester. Given the diversity of content, this approach is feasible only in a

year-long course, as is the case.

4.5. Assessment types and grade weights

Just as there is diversity in the syllabus content, languages, and paradigms used in the various courses, the assessment methodologies found in the 66 courses with assessment information available are also extremely disparate, both in terms of the type of elements, their terminology, their quantity, and their relative weights in the final classification.

To enable a more robust comparison between courses and institutions, as well as between CS1 and CS2, the assessment types found were categorized as indicated in Table 5. A separate category for Quizzes was maintained, as they constitute a specific short-term mechanism that differs from both tests and other elements of continuous assessment. This classification was based only on the presence of the element and not on its weight in the final classification.

Fig. 6 compares these aggregated assessments categories used in CS1 and CS2. The values indicate the presence of assessment elements in each category, expressed as a percentage of the total number of CS1 and CS2 courses, totaling 33 in each.

The aggregate analysis reveals that traditional formal assessment methods, such as mid-term tests and final examinations, continue to be the most frequent element (CS1: 88 %; CS2: 85 %). Tests and exams, both mid-term and final, can take various forms, including paper and digital tests, and some assessments last up to 4 h. In other cases, the final examination is optional, with the mark being replaced by the average of the mid-term tests. However, several courses no longer use this type of assessment, preferring instead to focus mainly on projects/portfolios, as well as on other types of continuous assessment, including quizzes.

The second most used category includes several elements of continuous assessment, in the form of small practical assignments, challenges, programming tasks, or problem-solving exercises. However, their frequency in CS2 has fallen significantly compared to CS1 (CS1: 73 %; CS2: 58 %).

Each of the remaining categories appears in less than half of the courses.

Projects are typically more complex, allowing students to apply their knowledge in a more integrated way and covering various topics from the course. They have more extended deadlines and can sometimes be done in groups. Some courses use the aggregation of projects in the form of portfolios. The use of projects as an assessment element has a similar prevalence in CS1 (39 %) and CS2 (42 %).

Quizzes are used as a tool for periodic feedback. They are very useful for both students and instructors, as they encourage continued study and allow for the assessment of students' understanding of a particular topic and their progress throughout the course. Except for three cases, quizzes are almost always used as a complement to other forms of continuous assessment. This form of assessment is slightly more frequent in CS2 (CS1: 27 %; CS2: 36 %).

Sometimes, practical performance in a laboratory setting is also assessed, either through a test or by requiring the preparation of a report. This type of assessment has an identical frequency in both courses (27 %).

It is also possible to find cases that value attendance and class participation. One might anticipate higher values in introductory courses, to provide more guidance and support for students who are starting to learn programming and encourage their participation. However, the number of classes that value them is greater in CS2 (21 %) than in CS1 (12 %). There may be other incentives for participation and attendance that are not explicitly described in the available information, or even mandatory.

The study examined both the diversity of evaluation elements and the relative importance of each element in the final grade. Therefore, information on the weight of each evaluation element in the final classification was collected and a statistical analysis of the data was conducted, as illustrated in Fig. 7.

An analysis of the weights assigned to the different assessment categories shows that tests and exams are the most valued elements in the courses, with the highest average weight (0.44) and median of 0.5 in both CS1 and CS2. Continuous assessment is the next most frequent, with an average weight similar to projects (~0.2), but with greater consistency, especially in CS1 (median 0.2).

When projects are considered in the assessment, they tend to have a relevant average weight (~0.2). However, they are not as frequent as the previous categories.

Quizzes and laboratory practices complement the main assessment elements, occurring with some regularity but with a lower weight, with average below 0.1 and a median of 0. Participation and attendance have the lowest weight and frequency, with minimal relevance in the final grade.

4.6. Policies for the use of generative AI

Many universities advertise general rules on the use of generative AI by their students. For this study, the analysis considered only rules for using AI explicitly defined for each course. The sample is limited to 15 courses, those with this information publicly available (Table 6). Six of them forbid any AI use, and seven permit it only partially. However, it is pertinent to highlight two courses, both in CS1, where the use of generative AI is not only permitted but, in fact, encouraged.

In the Introduction to Computer Programming course [25] at City University of Hong Kong, China, students are expected to test their code using an AI-assisted interactive development environment during laboratory activities. Furthermore, in the assignments, it is stated that "Students will put theory into practice and become proficient in AI-assisted programming.". Assessment is divided into Continuous Assessment (50 %), consisting of Quizzes (20 %) and Assignments (30 %), and Examination (50 %).

In the Introduction to Computer Science I course [26] at the

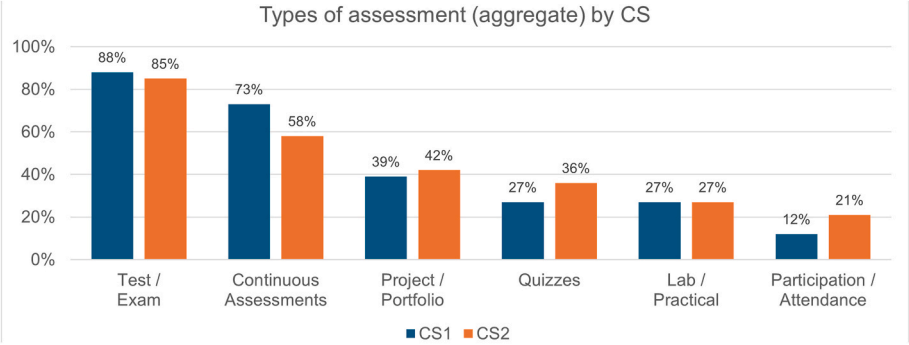

Fig. 6. Distribution of assessment categories in CS1 and CS2. The bars show the percentage of courses that include each category in their assessment methodologies.

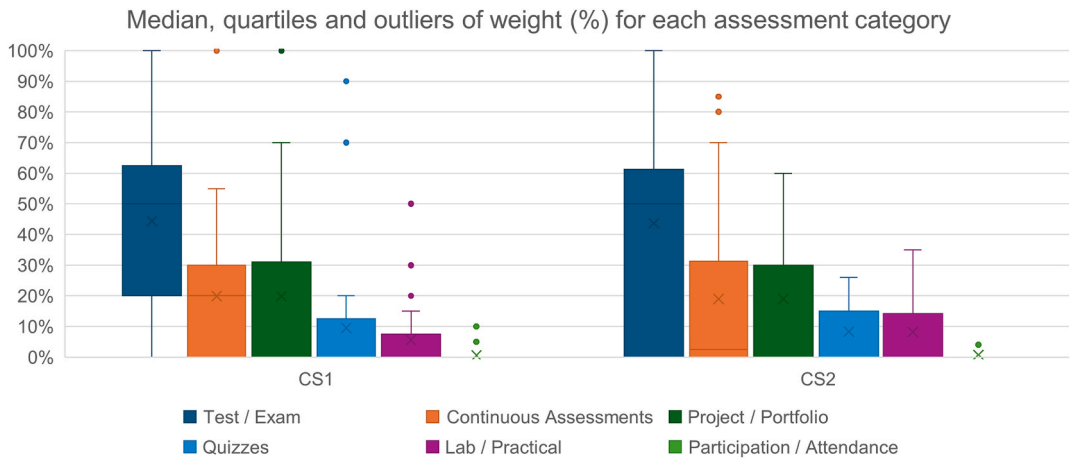

Fig. 7. Median, quartiles, and outliers of the weight (%) of each assessment category in the final grade in CS1 and CS2.

Table 6
Distribution of language paradigms used in introductory courses of computer science undergraduate programs.

Paradigm	Total courses	CS1 courses	CS2 courses
Allowed	2 (13.3 %)	2 (13.3 %)	0 (0 %)
Partially allowed	7 (46.7 %)	3 (37.5 %)	4 (57.1 %)
Not allowed	6 (40 %)	3 (37.5 %)	3 (42.9 %)
Total of different courses analyzed	*15*	*8*	*7*

University of Illinois at Urbana–Champaign, USA, the encouragement to use generative AI is even more explicit. Its statement on AI reads: "Software development is changing rapidly due to the incredible capabilities of generative AI. CS 124 is embracing AI coding agents to ensure you are prepared for the future of programming.", followed by a reasoned justification for this choice made in the course. Another difference between this course and others is that it is taught online, without lectures, based on daily lessons that students follow individually and in tutorials involving small groups. The assessment also differs from the traditional final exam, which does not exist. Instead, assessment is based on daily homework (10 %), small programming and debugging problems, weekly quizzes (70 %), a combination of multiple-choice questions, programming questions, and debugging challenges with automatic correction and feedback, and the Machine Project (MP) (20 %), which involves developing an Android application. As for the programming language, students have the option to choose between Java and Kotlin during the course.

For courses where the use of AI is only partially permitted, there is a balance in the approach taken. Its use is permitted in exploration and experimentation environments (such as laboratories, portfolios, and assignments), but is prohibited in summative assessment contexts, such as tests and exams. Assessments that permit the use of AI generally have a lower percentage in the final grade. The general idea is to allow the use of AI as a learning tool while ensuring that, ultimately, students can solve challenges independently, thereby demonstrating the acquisition of individual skills.

The possibility of completing assignments using AI raises some concerns because, on the one hand, it can serve as support for more motivated students, allowing them to explore different implementation approaches and receive feedback on autonomous implementations. But, on the other hand, it can result in excessive dependence among less interested students, compromising the development of algorithmic thinking and problem-solving skills. A particularly interesting solution to address this ambiguity is the one implemented by Redekopp, M., in a CS1 course at the University of Southern California's (CS124 Introduction to Programming) [27]. The multiple version assignments strategy provides students with two versions of each homework assignment. The first version must be implemented directly in the Codio editor without an AI plugin, allowing students to consult the bibliography and ask questions to instructors, but without resorting to AI. In the second version, students use the code from the first version with an AI platform, utilizing prompts to improve it and resolve any issues, and then copy the final version to Codio. The final grade for the assignment is based on the best of the two versions. However, if a violation of the rules is detected in the first version, the student will be penalized. This strategy requires students to work independently in the first version, to avoid penalties. And, simultaneously, promotes the correct use of generative AI in the second version, enabling them to obtain a good grade.

Despite the small sample size, the analysis indicates a lack of consensus on how to integrate generative AI into programming education and how to balance innovation with academic integrity. The challenges are significant, but debating how to find this balance is inevitable.

The diversity found in the choice of languages, paradigms, assessment strategies, and AI usage policies reflects the complexity of teaching programming and the attempt to balance pedagogical factors, labour market needs, individual responsibility, institutional traditions, and emerging trends.

5. Conclusion

This study analyses global trends in introductory Computer Science courses (CS1 and CS2) at top-ranked academic institutions, selected from the 2023 Shanghai Ranking. These institutions of academic excellence, with geographical and cultural diversity, often set trends in higher education. The study aimed to identify patterns and contribute to the debate on best practices in programming courses, helping to inform future decisions in the CS1 and CS2 curricula.

The results corroborate some trends identified in previous surveys of introductory programming courses, while providing additional insights specific to top academic institutions. Java and Python dominate introductory courses, accounting for more than half of the choices. Java's hegemony in CS1, reported in studies from the 2010s, has been replaced by Python at leading universities, validating the evolution observed in studies over the last decade. Now, Python stands out as the most used language in CS1, reflecting its smooth learning curve and flexibility. Java leads in CS2 and is also the most frequent when a single language is used in both courses. A language transition typically occurs between CS1 and CS2, typically from Python or C to Java. There are some regional particularities, such as the use of functional languages in Europe (e.g., Haskell) and low-level languages in Asia (e.g., C, C++).

Introduction to paradigms follows a classic approach, starting with imperative programming in CS1 to object-oriented programming in CS2.

However, although CS1 classes typically start with imperative programming, object-oriented programming is also introduced in more than one-third of CS1 courses. Only a small number of institutions adopt functional paradigms, mainly European ones and in CS1.

Traditional examinations are the most frequent and more heavily weighted components in both CS1 and CS2, followed by continuous assessments. Still, projects and labs are widely adopted, with quizzes emerging as valuable formative tools. Although with little impact on the final grade, there are also incentives for attendance and participation, particularly in CS2.

The teaching of programming is expected to be influenced by emerging trends, such as AI-based tools and the development of new programming languages. AI-based code generation tools have the potential to significantly change the teaching and learning of programming. Specifically, the challenge of Generative AI introduces an unprecedented variable that curricula are only beginning to address. The analysis of generative AI policies reveals contrasting decisions, despite the small sample size. While some courses strictly prohibit the use of AI tools, others allow them in exploratory contexts but not in summative assessments, and a few encourage their use.

Future studies may expand the dataset to include missing data and other institutions, such as those characterized by their smaller size, agility, and innovative approaches, in addition to or instead of solely focusing on top-ranked universities.

It is important to note that direct comparisons with previous surveys must consider methodological differences and that the populations differ significantly. These methodological and sampling differences may partially explain the variations in specific results.

These findings highlight that while there are some identified trends, top institutions maintain their pedagogical identities, and diversity remains a cross-cutting factor across all examined dimensions.

CRediT authorship contribution statement

João Pedro Gomes: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Redney Monteiro:** Writing – review & editing, Resources, Methodology, Investigation, Data curation, Conceptualization. **André Silva Moreira:** Writing – review & editing, Validation, Supervision, Methodology, Conceptualization. **Vítor Mendonça:** Writing – review & editing, Methodology, Conceptualization. **Anabela Pinho:** Writing – review & editing, Methodology, Conceptualization. **Carlos R. Cunha:** Writing – review & editing, Writing – original draft, Validation, Supervision, Resources, Project administration, Methodology, Investigation, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported by FCT - Fundação para a Ciência e Tecnologia, I.P. by projects: CeDRI, UID/05757/2025 (DOI: 10.54499/UID/05757/2025) and UID/PRR/05757/2025 (DOI: 10.54499/UID/PRR/05757/2025); SusTEC, LA/P/0007/2020 (DOI: 10.54499/LA/ P/0007/2020).

Data availability

Data will be made available on request.

References

[1] M. Hertz, What do "CS1" and "CS2" mean?: investigating differences in the early courses, in: Proceedings of the 41st ACM Technical Symposium on Computer Science Education, ACM, Milwaukee Wisconsin USA, 2010, pp. 199–203, https://doi.org/10.1145/1734263.1734335.
[2] R.P. Medeiros, G.L. Ramalho, T.P. Falcão, A systematic literature review on teaching and learning introductory programming in higher education, IEEE Trans. Educ. 62 (2019) 77–90, https://doi.org/10.1109/TE.2018.2864133.
[3] M.A. Foughali, Some thoughts on teaching introductory programming and the first language dilemma (Discussion Paper), in: Proceedings of the 23rd Koli Calling International Conference on Computing Education Research, ACM, Koli Finland, 2023, pp. 1–8, https://doi.org/10.1145/3631802.3631812.
[4] K.R. Parker, T.A. Ottaway, J.T. Chao, Criteria for the selection of a programming language for introductory courses, IJKL 2 (2006) 119, https://doi.org/10.1504/IJKL.2006.009683.
[5] Y. Prokop, E. Trofimenko, N. Loginova, A. Zadereyko, M. Gerganov, Multivariate analysis when choosing the first programming language studied in universities, in: 2019 IEEE 2nd Ukraine Conference on Electrical and Computer Engineering (UKRCON), 2019, pp. 1224–1228, https://doi.org/10.1109/UKRCON.2019.8879810.
[6] MS Farooq, SA Khan, F Ahmad, S Islam, A Abid, An Evaluation Framework and Comparative Analysis of the Widely Used First Programming Languages, PLoS ONE 9 (2) (2014) 1–25, https://doi.org/10.1371/journal.pone.0088941.
[7] R.M. Kaplan, Choosing a first programming language, in: Proceedings of the 2010 ACM Conference on Information Technology Education, Association for Computing Machinery, New York, NY, USA, 2010, pp. 163–164, https://doi.org/10.1145/1867651.1867697.
[8] T. Koulouri, S. Lauria, R.D. Macredie, Teaching introductory programming: a quantitative evaluation of different approaches, ACM Trans. Comput. Educ. 14 (26) (2015) 1–26:28, https://doi.org/10.1145/2662412.
[9] L. Gewali, J. Minor, Multi-Paradigm Approach for Teaching Programming, 2006.
[10] M.S. Samuel, An Insight into Programming Paradigms and their Programming Languages, vol. 1, 2017.
[11] A.N. Kumar, R.K. Raj, S.G. Aly, M.D. Anderson, B.A. Becker, R.L. Blumenthal, E. Eaton, S.L. Epstein, M. Goldweber, P. Jalote, D. Lea, M. Oudshoorn, M. Pias, S. Reiser, C. Servin, R. Simha, T. Winters, Q. Xiang, Computer Science Curricula 2023, ACM, New York, NY, USA, 2024, https://doi.org/10.1145/3664191.
[12] G. Fan, D. Liu, R. Zhang, L. Pan, The impact of AI-assisted pair programming on student motivation, programming anxiety, collaborative learning, and programming performance: a comparative study with traditional pair programming and individual approaches, International Journal of STEM Education 12 (2025) 16, https://doi.org/10.1186/s40594-025-00537-3.
[13] S. Rojas-Galeano, J. Tejada, F. Marmolejo-Ramos, Between tool and trouble: student attitudes toward AI in programming education. http://arxiv.org/abs/2508.05999, 2025. https://doi.org/10.48550/arXiv.2508.05999.
[14] D. Harris, G.S. Von Itzstein, D. Kelly, E. Smith, IDE vs. pen-and-paper showdown – Performance and perceptions of testing first-year IT students' programming skills, in: Proceedings ASCILITE 2023, Open Access Publishing Association, 2023, pp. 430–434, https://doi.org/10.14742/apubs.2023.541.
[15] J. Thangaraj, M. Ward, F. O'Riordan, A systematic review of formative assessment to support students learning computer programming, OASIcs 112 (2023), https://doi.org/10.4230/OASICS.ICPEC.2023.7. ICPEC 2023. 112, 7:1-7:13.
[16] J.P. Gomes, R. Monteiro, A. Moreira, V. Mendonça, A. Pinho, C.R. Cunha. Programming Language Choices for Introductory Computer Science Courses: Global Trends from the 2023 Shanghai Ranking. Emerging Trends in Information Systems and Technologies. WorldCIST 2025. Lecture Notes in Networks and Systems, vol. 1578, pp. 281–294. Springer, Cham (2026). doi:10.1007/978-3-032-00701-8_24.
[17] E. Murphy, T. Crick, J.H. Davenport, An analysis of introductory programming courses at UK universities, Programming 1 (2017) 18, https://doi.org/10.22152/programming-journal.org/2017/1/18.
[18] N. Avouris, Introduction to computing: a survey of courses in Greek higher education institutions, in: Proceedings of the 22nd Pan-Hellenic Conference on Informatics, Association for Computing Machinery, New York, NY, USA, 2018, pp. 64–69, https://doi.org/10.1145/3291533.3291549.
[19] O. Ezenwoye, What language? - the choice of an introductory programming language, in: 2018 IEEE Frontiers in Education Conference (FIE), 2018, pp. 1–8, https://doi.org/10.1109/FIE.2018.8658592.
[20] B.A. Becker, A survey of introductory programming courses in Ireland, in: Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education, Association for Computing Machinery, New York, NY, USA, 2019, pp. 58–64, https://doi.org/10.1145/3304221.3319752.
[21] T.C. Smith, L. Jones, First course programming languages within US business college MIS curricula, Journal of Information Systems Education 32 (4) (2021) 283–293.
[22] R.M. Siegfried, K.G. Herbert-Berger, K. Leune, J.P. Siegfried, Trends of commonly used programming languages in CS1 and CS2 learning, in: 2021 16th International Conference on Computer Science & Education (ICCSE), 2021, pp. 407–412, https://doi.org/10.1109/ICCSE51940.2021.9569444.
[23] R. Mason, Simon, B.A. Becker, T. Crick, J.H. Davenport, A global survey of introductory programming courses, in: Proceedings of the 55th ACM Technical Symposium on Computer Science Education, vol. 1, ACM, Portland OR USA, 2024, pp. 799–805, https://doi.org/10.1145/3626252.3630761.
[24] ShanghaiRanking's global ranking of academic subjects. https://www.shanghairanking.com/rankings/gras/2023/RS0210 last accessed 2024/08/06.

[25] City university of Hong Kong: CS1302: introduction to computer programming - detailed course information. https://www.cityu.edu.hk/catalogue/ug/202526/course/CS1302.pdf last accessed 2025/08/11.

[26] G. Challen, University of Illinois at Urbana-Champaign, CS 124: Fall 2025 Syllabus. https://www.cs124.org/syllabus/Fall2025 last accessed 2025/08/11.

[27] M. Redekopp, University of Southern California, Fall '25 CSCI 102 - introduction to programming. https://bytes.usc.edu/cs102/ last accessed 2025/08/12.