LARGE LANGUAGE MODEL

Evaluating Regional-Language and Code-Mixed Explanations Generated by Large Language Models for Python Code Comprehension: A Comparative Study

Parshuram Manur[1], Srushti Limhan[1], Sarjerao Ghatge[2], Avani Deshpande[2]

[1]Department of Computer Applications and Management, Navsahyadri Group of Institutes, Savitribai Phule Pune University, Pune, Maharashtra, India
[2]Guide, Department of Computer Applications and Management, Navsahyadri Group of Institutes, Savitribai Phule Pune University, Pune, Maharashtra, India

Published November 2026

Abstract

Large Language Models (LLMs) are increasingly used to explain source code to novice programmers, yet most evaluations assume English-language explanations. Many learners in Maharashtra think about programming in Marathi while relying on English technical vocabulary, so a Marathi-English code-mixed explanation may be more accessible, but only if it remains technically correct. This study asks whether LLM-generated code-mixed explanations are as useful as English ones. Ten Python programs covering variables and data types, conditionals, loops, functions, lists and dictionaries, string manipulation, recursion, exception handling, searching and sorting, and object-oriented programming were each explained by one fixed model (Claude Sonnet 5.5) in English and in Marathi-English using equivalent prompts. The study has two components. Component A is a learner-perception pilot: the same 20 students reported, for each format, whether they understood the explanations easily or needed time. In the English format 8 of 20 students (40%) understood easily; in the Marathi-English format 14 of 20 (70%) did. The 30-percentage-point difference favours Marathi-English. Because the same students rated both formats, the appropriate test is McNemar's exact test; the response totals alone place its two-sided p-value between 0.031 and 0.238, so the difference is reported as a trend and not as a confirmed effect. Component B is a five-criterion rubric evaluation (technical correctness, completeness, clarity, terminology consistency and debugging usefulness, 0 to 4 each) by two raters; this component is fully designed but its scores have not yet been collected. The survey measures perceived understanding only, not learning gain or technical accuracy. The main contribution is a reproducible protocol and rubric for evaluating regional-language explanations of code, together with a first small perception baseline for Marathi-English.

Keywords: large language models; code explanation; Python programming education; code-mixed language; Marathi; multilingual NLP; code comprehension

1 Introduction

1.1 Background of artificial intelligence in programming education

Reading and explaining code is a core programming skill, and "Explain in Plain English" questions have long been used to develop and assess it [1]. Writing good explanations for many examples takes instructors considerable time, and student-written explanations vary in quality [2]. Generative AI offers explanations on demand. Early work found that most code explanations generated by a Codex

model were sensible [3], and explanations produced by LLMs have since been deployed inside course materials [4].

1.2 Role of large language models in code explanation

In a large course of about 1,000 students, LLM-created explanations were rated significantly easier to understand and more accurate than student-created ones [2]. Line-by-line LLM explanations have also been compared with expert and student explanations for Java examples [5]. A 2026 study of novices who prompted LLMs freely reported that the explanations for typical introductory programs were considerably accurate and complete, and that LLM access was associated with a statistically significant rise in confidence [6]. These studies evaluate English explanations. Whether the same quality holds in a regional language is the question this paper takes up.

1.3 Importance of regional languages in programming education

Non-native English speakers face barriers in programming, and many know more than they can articulate in English [7]. India's National Education Policy 2020 lists promoting multilingualism among its principles [8]. Yet in a study across ten Indian languages, many students preferred English for technical work, and when they did write in their own language they often mixed English words into it, as in Hindi-English and Kannada-English answers [9]. The realistic regional format is therefore code-mixed rather than purely regional. For Marathi, Marathi-English code-mixing is well attested in NLP resources, and Roman-script spelling variation is a known difficulty [10]. For Marathi-speaking programming learners, an explanation that keeps English technical terms inside Marathi sentences is plausibly the most natural register, but that is an assumption to test, not a finding.

1.4 Research problem

Evidence on regional-language LLM support in programming is concentrated on the input side: how well models generate code from non-English or code-mixed prompts [7], [11]-[12]. The output side is less studied: when an LLM itself writes a Marathi-English explanation of Python code, do learners find it easy to understand, and is it technically correct, complete, consistent in its terminology, and useful for debugging? Model performance is known to differ across Indian languages [9], so quality in Marathi-English cannot be assumed from English results.

1.5 Research objectives

1. Assemble a fixed set of 10 Python programs covering the topics in the code sample selection subsection.
2. Generate English and Marathi-English explanations of each program with one model, equivalent prompts and consistent settings.
3. Measure, through a small learner survey, how many students understood each format easily and how many needed time.
4. Define a five-criterion rubric, each criterion scored 0 to 4 (maximum 20 per explanation), for rater-based comparison of the two formats.
5. Catalogue recurring terminology and accuracy problems in the Marathi-English explanations (planned with the rubric evaluation).

1.6 Research questions

RQ1. Do more students report easy understanding of Marathi-English explanations than of English explanations? (answered in the Results section)

RQ2. How do total rubric scores (0 to 20) of English and Marathi-English explanations compare? (planned)

RQ3. On which of the five criteria, and for which program topics, do the two formats differ most? (planned)

RQ4. Which terminology and accuracy problems recur in Marathi-English explanations? (planned)

1.7 Scope of the study

The study evaluates AI-generated explanations of Python programs in two formats. It uses one model and ten programs. A small perception survey in which the same 20 students rated both formats was collected. It records perceived ease of understanding only, so the study makes no claim about learning gains or about technical correctness from the survey. Technical quality is assessed through the rater-scored rubric, which is designed but not yet applied. "Marathi-English" means Marathi sentence structure with English programming terms retained; the script choice is fixed in the prompt design subsection.

1.8 Research contribution

The contribution is methodological and small by design: a reproducible protocol, a five-criterion rubric for bilingual code explanations, a results framework ready for rater data, and a first perception baseline for Marathi-English that instructors in Maharashtra can use when judging whether such explanations are worth examining further. It differs from prior work in [7], [11]-[12], which examines prompts for code generation, by examining the explanations a model produces. The five criteria are proposed in this paper and are not a previously validated instrument.

2 Literature review

The closest verified work studies LLM code explanations in English, or non-English prompts for code generation; none found examines Marathi-English explanations written by the model. Sources were checked on arXiv, ACM proceedings pages, the ACL Anthology, FLAIRS proceedings and institutional pages during a targeted search in November 2026. It was not a systematic review. Table 1 lists each work; "Limitations" means limits relative to the present study, not flaws in the original work, except where the authors state them.

Table 1. Verified studies relevant to this research (2022 to 2026).

Study	Method	Main findings	Limitations here
[3] Sarsa, Denny, Hellas, Leinonen (2022), ICER. Peer-reviewed	Codex generated exercises and code explanations; qualitative and quantitative assessment	Most content novel and sensible; some usable as is; oversight still needed	English only; older model
[2] Leinonen et al. (2023), ITiCSE. Peer-reviewed	LLM vs student explanations in a course of about 1,000 students	LLM explanations rated significantly easier to understand and more accurate	English only; GPT-3 era model

Study	Method	Main findings	Limitations here
[4] MacNeil et al. (2023), SIGCSE. Peer-reviewed	Three explanation types shown beside code in an e-book, with feedback buttons	All types viewed; most students found them helpful; engagement varied	Preliminary; English; web-development code
[5] Lekshmi Narayanan et al. (2024), arXiv:2403.05538. Preprint	ChatGPT line-by-line explanations vs expert and student explanations (60 students), four Java examples	Prompt iteration mattered; extending the explanation once gave the best results	Four Java examples; English
[6] Sajib, Tamang, Chapagain, Rus (2026), FLAIRS-39. Peer-reviewed	Novices freely prompted LLMs for code explanations	Explanations for typical introductory programs considerably accurate and complete; access linked to higher confidence	English only
[7] Prather et al. (2025), ACE, arXiv:2412.12800. Peer-reviewed venue	Students prompted in Arabic, Chinese and Portuguese to generate code	Native-language prompting worked, but students struggled to specify terminology and concepts	Code generation, not explanation; no Marathi
[9] Smith IV, Kumar, Denny (2024), arXiv:2409.20297. Preprint	GPT-4o generated code from expert translations in ten Indian languages (180 faculty); 72 students in an online course	Most languages reached 75 percent or higher correctness; 78.8 percent of student answers were in English; mixed or mother-tongue answers succeeded 80 percent of the time vs 95.9 percent for English	Convenience sampling, unvalidated translations; student-side text, not LLM explanations
[13] Denny et al. (2024), ITiCSE. Peer-reviewed	Students explained code so an LLM could regenerate it	Some students succeeded with explanations in Chinese	Student-written; English focus
[15] Jordan, Ly, Soosai Raj (2024), SIGCSE. Known only as reported in [9]	Non-English programming exercises with GPT-3.5	Reported in [9] as weak for Tamil	Not read directly; exercises, not explanations
[11] Peng, Chai, Li (2024), HumanEval-XL, LREC-COLING. Peer-reviewed	Parallel prompts in 23 natural and 12 programming languages (22,080 prompts)	Benchmark contribution	Code generation only
[14] Singh et al. (2025), IndicEval-XL, arXiv:2502.19067. Preprint	Six Indic languages including Marathi, paired with 12 programming languages	Benchmark contribution	Code generation only
[17] Sheokand, Sawant (2025), CodeMixBench, arXiv:2505.05063. Preprint	Controlled code-mixing in prompts, including Hinglish	Benchmark contribution	Code generation only
[12] Pawagi, Gupta, Rolla, Saha, HinglishEval (IISc; venue unconfirmed)	Hinglish translation of HumanEval; open and closed models compared	Models generally more accurate on English prompts; Hinglish performance promising	Prompt side only

Study	Method	Main findings	Limitations here
[10] Chavan et al. (2023), My Boli, arXiv:2306.14030. Preprint	10 million Marathi-English sentences; models; three benchmarks of about 12,000 tweets	Code-mixed Marathi models outperformed baselines	Social-media text; Roman-script spelling varies
[16] Katzy et al. (2026), arXiv:2605.05902. Preprint	Five code LLMs generated comments in Dutch, English, Greek, Polish and Chinese; 12,500 comments open-coded into 26 error types	Quality dropped outside English (linguistic errors up to 15.1 times higher); no automatic metric was reliable; human judgment indispensable	Code comments for developers; no Marathi or code-mixed text

2.1 AI-Based programming education

Studies of LLMs in programming education have moved from generating exercises and explanations [3] to deploying explanations in course materials [4] and comparing them with student work [2]. Linking explaining code to LLM code generation has also been proposed as a way to build comprehension and prompting skills [13].

2.2 LLM-Generated code explanations

Explanations have been compared with student and expert versions [2], [5], offered in several types [4], and studied in novices' free use [6]. All of these studies are in English.

2.3 Multilingual AI systems

Multilingual evaluation of code models exists as parallel benchmarks [11], [14], but the task is generating code. Model quality varies between languages: weak Tamil performance is reported in [15] as relayed by [9], and a 2026 preprint finds that LLM-generated code comments degrade outside English and that automatic metrics are unreliable for them [16].

2.4 Regional-language programming support

For Indian languages, evidence covers prompts in Hinglish [17], [12], six Indic languages including Marathi [14], and student answers in ten languages [9]. Marathi-English code-mixing has NLP resources [10] but none specific to programming that I found.

2.5 Code comprehension and debugging

Code comprehension is classically assessed by explaining code in plain language [1], [13]. I found no study that rates LLM explanations for usefulness in debugging, so criterion 5 in this paper is a new proposal rather than an adaptation.

2.6 Accuracy of ai-generated explanations

Accuracy findings are encouraging but conditional: explanations were rated more accurate than students' [2], mostly sensible [3] and considerably accurate for typical introductory programs [6], yet oversight was still advised [3].

2.7 Language-related limitations

Learners struggled with terminology when prompting in their own language [7], preferred English for technical answers [9], and Roman-script Marathi has no single correct spelling [10]. These point to terminology consistency as the likeliest weak spot for code-mixed explanations.

2.8 Research gap

Prior work mostly tests what models do with non-English input. The nearest output-side study evaluates generated code comments in five non-Indian languages for developers [16]. I found no rubric-based or learner-based comparison of Marathi-English and English explanations of Python code written for learners. Because the search was targeted, this is a gap in the sources found, not proof that no such study exists.

3 Research design overview

The study is a paired comparative evaluation at the level of programs: each of 10 Python programs is explained twice by the same model, once in English and once in Marathi-English. Both explanations of a program share the same code, so differences can be attributed to the explanation format, subject to the limits in the ethical considerations and limitations subsection. Two components use these 20 explanations (Figure 1): Component A, a learner-perception survey (completed), and Component B, a five-criterion rubric evaluation by two raters (designed, scores pending).

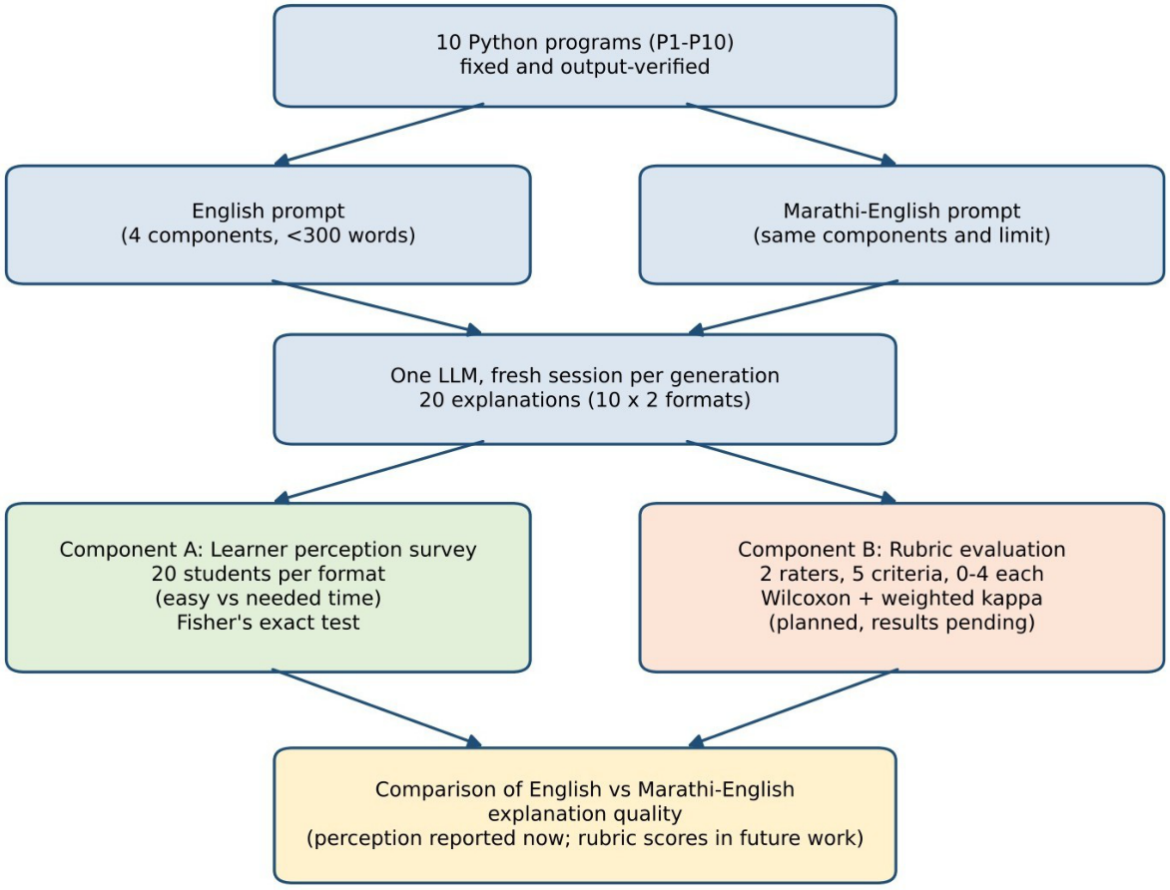


Figure 1. Overall study design: one program set, two prompt formats, one model, and two evaluation components.

4 Methodology

4.1 Research design

See the research design overview above. Component A reports perceived ease of understanding; Component B scores quality on five rubric criteria.

4.2 AI models and tools used

Model: Claude Sonnet 5.5 (Anthropic). One model was used for all 20 explanations. Interface: the Claude chat application (web and mobile). Generation dates and times and the model version string shown in the application were recorded for each explanation. Settings: the chat application does not let users adjust temperature, so default settings were used; no system prompt was added beyond the prompts in the prompt design subsection, and a fresh conversation was used for every generation so earlier outputs could not influence later ones. Execution: Python 3.13 was used to run each program and confirm its output before use.

4.3 Python code sample selection

The ten programs are short, textbook-style and written for this study, one per required topic. Each runs without errors; its output is shown so raters can check correctness claims. Programs were fixed before any explanation was generated and were not changed afterwards.

Table 2. The ten Python programs.

ID	Topic	Program summary	Verified output
P1	Variables and data types	Four variables of different types, type printing, age increment	Four type names, then "Asha is 22 years old"
P2	Conditional statements	Grade from marks with if/elif/else	First Class
P3	Loops	Sum of even numbers from 1 to 10	30
P4	Functions	Rectangle area with a default argument	15, then 4
P5	Lists and dictionaries	Average marks per student from a dictionary of lists	{'Amit': 75.0, 'Neha': 87.5}
P6	String manipulation	Case-insensitive palindrome check by slicing	Palindrome
P7	Recursion	Factorial of 5	120
P8	Exception handling	Division with try, except and finally	Division attempted, then Cannot divide by zero
P9	Searching and sorting	Bubble sort then linear search	[1, 2, 5, 9] 3
P10	Object-oriented programming	Account class with deposit and withdraw	Ravi 600

The full source of the ten programs is given in the accompanying file mca_programs_P1_P10.py. Two examples are shown below.

```
# P7: Recursion
def factorial(n):
    if n == 0:
        return 1
    return n * factorial(n - 1)
print(factorial(5))

# P8: Exception handling
def safe_divide(a, b):
    try:
        return a / b
    except ZeroDivisionError:
        return "Cannot divide by zero"
    finally:
        print("Division attempted")
print(safe_divide(10, 0))
```

4.4 English and Marathi-English prompt design

Marathi-English here means Marathi sentence structure written in Devanagari, with English programming terms (variable, loop, function, exception and similar) kept in Latin script. Devanagari was chosen because Roman-script Marathi has no single correct spelling [10], which would add a rating confound. Both prompts request the same four components and the same length limit; only the output language instruction differs.

4.4.1 English prompt

Explain the following Python program for a beginner. Include: (1) the overall purpose of the program, (2) a step-by-step walkthrough of how it works, (3) the exact output it produces, and (4) common mistakes a beginner might make and tips for debugging it. Keep the explanation under 300 words. Do not rewrite the code.
<PROGRAM>

4.4.2 Marathi-English prompt

खालील Python प, ोग, ामचे नविशKयांसाठी 5प4टीकरण मराठी-इंग,जी ि मL भाषेत Д ा. वाKयरचना मराठी असावी, पण variable, loop, function, condition, exception यांसारखे programming terms इंग,जीतच ठेवा. 5प4टीकरणात हे समािव4ट करा: (१) प, ोग, ामचा एकूण उ5 ेश, (२) तो टYYयाटYYयाने कसा चालतो, (३) तो नेमका कोणता output देतो, आिण (४) नविशKयांकडून होणा_या सामा`य चुका व debugging साठी ि टYस. 5प4टीकरण ३०० शêदांपेd ा कमी ठेवा. कोड पु`हा ि लहू नका.
<PROGRAM>

Fairness rests on four controls: identical program text, identical requested components, identical length limit and identical model settings. Because Marathi text tokenises differently from English, word counts are recorded for every explanation.

4.5 AI explanation generation procedure

1. Fix the ten programs and both prompts. Do not edit them after step 2.
2. For each program in order P1 to P10, open a fresh session and submit the English prompt with the program inserted.
3. Copy the full response unedited into the data sheet with its timestamp.
4. In another fresh session, repeat with the Marathi-English prompt.
5. Do not regenerate. If a response fails (empty or cut off), log it and regenerate once, keeping both.
6. Assign each explanation a random ID so raters do not see program order.

4.6 Evaluation criteria (component B)

- **Technical Correctness:** every statement about the program's behaviour and output is true.
- **Explanation Completeness:** all four requested components are present and cover the whole program.
- **Clarity and Understandability:** a beginner could follow it; sentences and structure are easy to read.
- **Programming Terminology Consistency:** terms are used correctly and the same concept keeps the same term throughout, with no mixed or wrongly translated terms.
- **Debugging Usefulness:** the mistakes and tips named are realistic for this program and would help locate or fix a fault.

4.7 Scoring rubric

Each criterion is scored 0 to 4 as an integer. Scores 1 and 3 are intermediate between the anchors shown. The total per explanation is the sum, with a maximum of 20.

Table 3. Scoring rubric anchors.

Criterion	4	2	0
Technical Correctness	No incorrect statement; output stated exactly	One minor error or imprecision that does not mislead	A major error about behaviour or output
Explanation Completeness	All four components present and the whole program covered	Two components missing or parts of the program skipped	Fewer than two components addressed
Clarity and Understandability	Easy for a beginner; well ordered	Understandable with effort; uneven structure	Confusing or hard to follow
Terminology Consistency	Correct and consistent throughout	Occasional inconsistency or an awkward mix of terms	Frequent wrong or conflicting terms
Debugging Usefulness	Specific, realistic, actionable tips for this program	Generic tips that fit many programs	No useful debugging content or misleading tips

The rubric is proposed here and is not a validated instrument. Before scoring, two raters score P1 in both languages together to calibrate; those scores are kept in a pilot sheet and excluded from the analysis if P1 is later re-scored.

$$S(i,\ L) = c_1 + c_2 + c_3 + c_4 + c_5 \tag{1}$$

where each criterion score c is an integer from 0 to 4 (maximum 20 per explanation).

$$\text{Percentage}\left(L\right) = \frac{\sum_{i=1}^{10} S\left(i,\ L\right)}{200} \times 100 \tag{2}$$

where i indexes the program (P1 to P10) and L is the format (English or Marathi-English).

4.8 Data collection procedure

Component A (learner survey). Twenty students read the explanations in both formats. The same 20 students rated both formats. For each format they were asked whether they understood the explanation easily or needed time to understand it, and answered in person by show of hands; the counts were recorded by the researcher. This binary response is the only survey measure analysed. Because responses were given in the open and not anonymously, and the same students saw both formats, they may have influenced one another and may show order effects. Only aggregate counts are reported. The survey was conducted on 11 August 2026 with MCA second-year students of the 2025-2027 batch.

Component B (rubric). Two raters fluent in Marathi and Python score all 20 explanations independently, using the random IDs, with the program output known. Language cannot be hidden from raters, so blinding applies only to program order and to the model's identity in notes. Each rater records five scores and free-text notes. Differences of more than one point on any criterion are discussed and resolved; unresolved ones keep both values and are reported. Error tags used: factual error, wrong term, mistranslated term, missing component, hallucinated output.

4.9 Data analysis techniques

Component A. The counts in each format are reported with percentages. Because the same students rated both formats, the responses are paired, and the appropriate test is McNemar's exact test on the discordant pairs (students easy in one format only). The per-student cross-tabulation was not recorded, so the test cannot be computed directly. Instead, the response totals are used to bound it: the number of students easy only in Marathi-English minus the number easy only in English must equal 14 - 8 = 6, which limits the possible pairings and therefore the possible p-values. The significance level is 0.05.

Component B (planned). Mean and standard deviation of totals and of each criterion per format; paired differences per program; the Wilcoxon signed-rank test on the ten paired totals (exact p-value, median difference, matched-pairs effect size) and per criterion with Holm correction; Cohen's weighted kappa [18] between raters per criterion, reported before consensus; counts of each error tag by format. Because n = 10, results are indicative only.

4.10 Ethical considerations and limitations

The model's terms of use are followed and its version recorded. Student responses are reported only in aggregate, as counts, with no names or personal identifiers. Limitations are listed in the limitations of the research.

5 Results

5.1 Learner perception survey (component A)

Table 4 and Figures 2 and 3 show the survey responses. In the English format, 8 of 20 students (40%) understood the explanations easily and 12 (60%) needed time. In the Marathi-English format, 14 of 20 students (70%) understood easily and 6 (30%) needed time.

Table 4. Learner perception of explanation format (same 20 students rated both formats).

Format	Understood easily	Needed time	% easy
English	8	12	40%
Marathi-English	14	6	70%
Difference (Marathi-English minus English)	+6	-6	+30 points

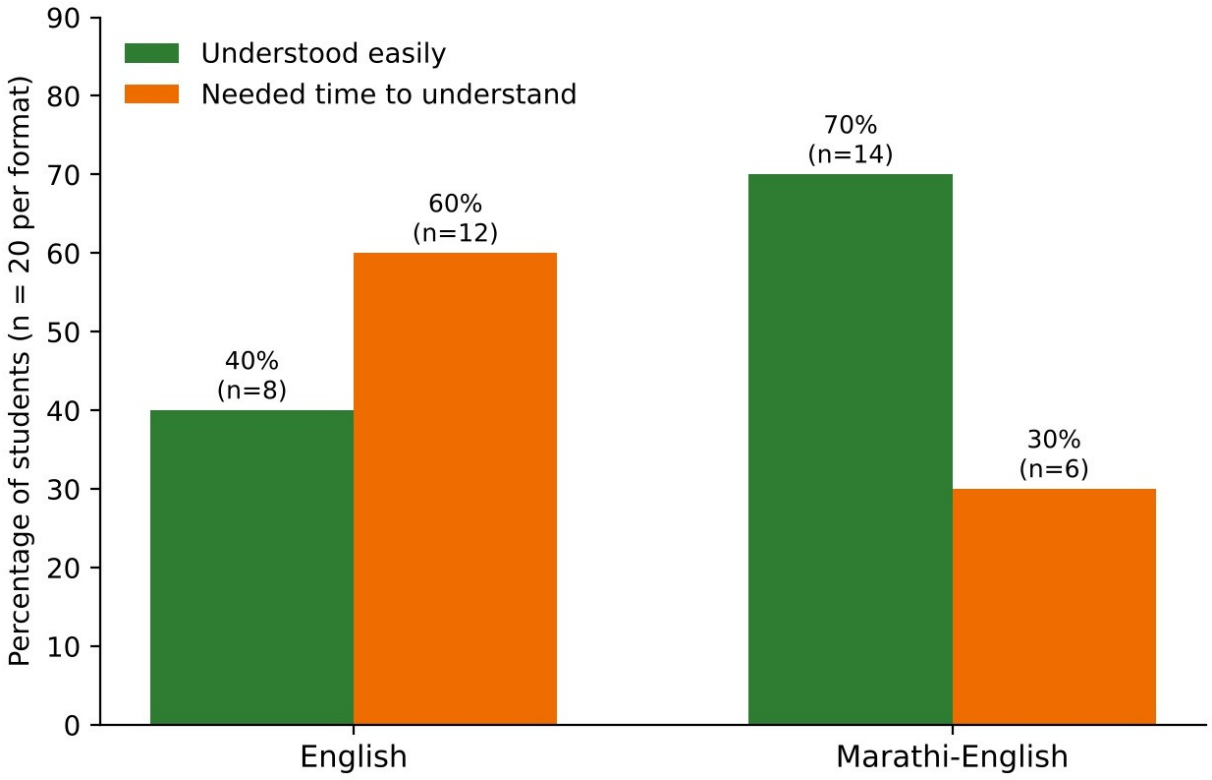


Figure 2. Percentage of students who understood each format easily or needed time (same 20 students; counts in brackets).

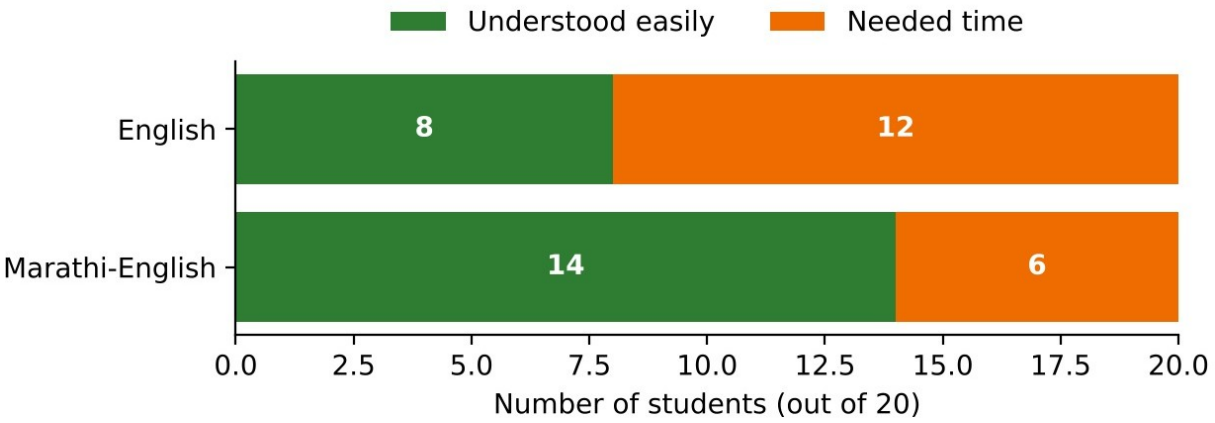


Figure 3. Number of students (out of 20) by response and format.

Because the same 20 students rated both formats, the 2 x 2 comparison of independent groups does not apply; the correct analysis is McNemar's exact test on paired responses. The totals imply that six more students found Marathi-English easy than found English easy (14 - 8 = 6). The per-student pairing was not recorded, so the exact p-value cannot be stated. It depends on how many students

changed their answer between formats. If the six extra students were the only ones who changed (no one found English easy but Marathi-English hard), the exact two-sided p is 0.031. If the maximum possible number changed (6 students easy in English only and 12 in Marathi-English only), the exact two-sided p is 0.238. Significance at the 0.05 level therefore cannot be established from the totals alone. The result is reported as a trend favouring Marathi-English, not as a confirmed effect, and it is not evidence that the formats are equivalent.

5.2 Rubric evaluation (component B)

Rubric scoring by two raters has not yet been performed. No scores, kappa values, p-values or error counts are reported for Component B, and none are estimated. The planned outputs are: a table of 20 explanation totals with word counts, a criterion-level table with Wilcoxon and Holm-adjusted p-values and weighted kappa, an error-tag table by format, a grouped bar chart of criterion means by format, and a paired dot plot of program totals. These will answer RQ2 to RQ4 once the data exist.

5.3 Answer to RQ1

More students reported easy understanding of the Marathi-English explanations (70%) than of the English ones (40%), a difference of 30 percentage points. Because the pairing was not recorded, the exact paired p-value lies between 0.031 and 0.238, so this is suggestive, not conclusive. RQ2 to RQ4 remain open until rater scores are collected.

6 Discussion

6.1 Interpretation of results

The survey direction is consistent with the idea that Marathi-speaking learners find code-mixed explanations easier to follow, which fits the observation in [9] that many Indian students mix English technical words into their own language. However, the difference cannot be confirmed as significant from the totals alone, the sample is small, and the survey measures a single binary perception. It cannot show that the Marathi-English explanations were more correct, only that more students said they found them easy.

6.2 Comparison with existing research

English LLM explanations have been rated accurate and understandable [2], [6]. Native-language use has been linked to terminology difficulty [7], and model performance varies across Indian languages [9]. The present perception result does not contradict these findings, because it does not test accuracy. The hypothesis that terminology consistency is the weakest criterion for Marathi-English, suggested by [7], [9] and [10], remains untested until the rubric scores exist.

6.3 Effectiveness of Marathi-English AI explanations

Here "effectiveness" can only mean perceived ease of understanding in the survey, plus rater-assessed quality once Component B is run. It does not mean student comprehension or learning, which were not measured.

6.4 Technical accuracy and terminology challenges

Perceived ease and technical correctness can diverge: an explanation can feel easy to read and still contain a wrong statement. This is why the rubric evaluation matters and why the survey result should not be used alone to endorse Marathi-English explanations. Concrete cases of wrong, mis-translated or inconsistent terms will be reported with program IDs after Component B.

6.5 Implications for AI-Based programming education

The study offers instructors a protocol for checking AI explanations in a regional language before use, and an early indication that learners may find Marathi-English explanations easier. Prior work advises oversight of generated explanations [3]; the rubric makes that oversight concrete. Any recommendation to use such explanations in teaching would need a larger study with comprehension measures.

6.6 Limitations of the research

- One model and one run; results may differ across models and versions.
- Ten short programs.
- The survey has 20 students and one binary question, answered by show of hands in person, and measures perceived ease only. The same students rated both formats, so responses are paired, but the pairing was not recorded and the paired test could not be computed. Open, non-anonymous responses and order effects may have influenced answers.
- Self-reported perception can be influenced by language preference and by not knowing whether an explanation is correct.
- The rubric is proposed here, is not validated, and has not yet been applied; raters cannot be blinded to language.
- One definition and script of code-mixing; Roman-script Marathi may behave differently.
- No measure of learning outcomes.
- The literature search was targeted, not systematic, and some sources were checked at abstract level only.

6.7 Future research implications

Future work should complete the rubric scoring, add several models and repeated generations, use longer programs, include Roman-script Marathi and other regional languages, validate the rubric with instructors, and run a larger study with Marathi-speaking learners that tests comprehension and debugging outcomes directly.

7 Conclusion

7.1 Summary of the research

This paper compares LLM-generated explanations of ten Python programs in English and Marathi-English. In a pilot survey in which the same 20 students rated both formats, 70% found the Marathi-English explanations easy to understand against 40% for English, a 30-point difference that is a trend and not confirmed as statistically significant (paired exact p between 0.031 and 0.238, depending on the unrecorded pairing). A five-criterion rubric evaluation by two raters is designed but not yet carried out.

7.2 Answers to research questions

RQ1: more students found Marathi-English explanations easy (70% vs 40%), not confirmed as significant at the 0.05 level. RQ2 to RQ4 are pending rubric scoring and no answer is claimed.

7.3 Main research contribution

A reproducible protocol: ten fixed programs with verified outputs, equivalent English and Marathi-English prompts, a five-criterion rubric scored 0 to 4, an analysis plan, and a first learner-perception baseline for Marathi-English code explanations.

7.4 Practical implications

Instructors in Maharashtra can use the protocol to check AI explanations in Marathi-English and may expect learners to find them at least as easy to follow, but accuracy must be verified before the explanations are used with students.

7.5 Future scope

Complete the rubric evaluation, run the protocol on more models and repeated generations, extend it to longer programs, Roman-script Marathi and other Indian languages, validate the rubric with instructors, and test comprehension and debugging outcomes in a larger study with Marathi-speaking learners.

References

[1] L. Murphy, S. Fitzgerald, R. Lister, and R. McCauley, "Ability to 'explain in plain English' linked to proficiency in computer-based programming," in *Proc*. 9th Annu. Int. Conf. Int. Computing Education Research (ICER '12), Auckland, New Zealand, 2012, pp. 111-118, doi: 10.1145/2361276.2361299.

[2] J. Leinonen, P. Denny, S. MacNeil, S. Sarsa, S. Bernstein, J. Kim, A. Tran, and A. Hellas, "Comparing code explanations created by students and large language models," in *Proc*. 2023 Conf. Innovation and Technology in Computer Science Education V. 1 (ITiCSE 2023), Turku, Finland, 2023, pp. 124-130, doi: 10.1145/3587102.3588785.

[3] S. Sarsa, P. Denny, A. Hellas, and J. Leinonen, "Automatic generation of programming exercises and code explanations using large language models," in *Proc*. 2022 ACM Conf. Int. Computing Education Research V. 1 (ICER 2022), Lugano, Switzerland, 2022, doi: 10.1145/3501385.3543957.

[4] S. MacNeil, A. Tran, A. Hellas, J. Kim, S. Sarsa, P. Denny, S. Bernstein, and J. Leinonen, "Experiences from using code explanations generated by large language models in a web software development e-book," in *Proc*. 54th ACM Tech. Symp. Computer Science Education V. 1 (SIGCSE 2023), Toronto, ON, Canada, 2023, pp. 931-937, doi: 10.1145/3545945.3569785.

[5] A. B. Lekshmi Narayanan, P. Oli, J. Chapagain, M. Hassany, R. Banjade, P. Brusilovsky, and V. Rus, "Explaining code examples in introductory programming courses: LLM vs humans," *EasyChair Preprint* 12117, 2024 (also arXiv:2403.05538). [Online]. Available: https://arxiv.org/abs/2403.05538

[6] M. I. Sajib, S. Tamang, J. Chapagain, and V. Rus, "A study on how free use of LLMs assists novices with code comprehension tasks," *Int*. FLAIRS Conf. Proc., vol. 39, no. 1, 2026, doi: 10.32473/flairs.39.1.141647.

[7] J. Prather et al., "Breaking the programming language barrier: Multilingual prompting to empower non-native English learners," in *Proc*. 27th Australasian Computing Education Conf. (ACE '25), Brisbane, Australia, Feb. 2025; arXiv:2412.12800. [Online]. Available: https://arxiv.org/abs/2412.12800

[8] Ministry of Education, Government of India, "National Education Policy 2020," *2020*. [Online]. Available: https://www.education.gov.in/sites/upload_files/mhrd/files/NEP_Final_English_0.pdf

[9] D. H. Smith IV, V. Kumar, and P. Denny, "Explain in Plain Language questions with Indic languages: Drawbacks, affordances, and opportunities," *arXiv:2409.20297*, Sep. 2024 (preprint). [Online]. Available: https://arxiv.org/abs/2409.20297

[10] T. Chavan, O. Gokhale, A. Kane, S. Patankar, and R. Joshi, "My Boli: Code-mixed Marathi-English corpora, pretrained language models and evaluation benchmarks," *arXiv:2306.14030*, Jul. 2023 (preprint). [Online]. Available: https://arxiv.org/abs/2306.14030

[11] Q. Peng, Y. Chai, and X. Li, "HumanEval-XL: A multilingual code generation benchmark for cross-lingual natural language generalization," in *Proc*. 2024 Joint Int. Conf. Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024), Torino, Italy, May 2024, pp. 8383-8394. [Online]. Available: https://aclanthology.org/2024.lrec-main.735

[12] M. Pawagi, A. Gupta, S. R. Rolla, and K. Saha, "HinglishEval: Evaluating the effectiveness of code-generation models on Hinglish prompts," *Indian Institute of Science*, Bengaluru, India (publication venue not confirmed). [Online]. Available: https://kiac.iisc.ac.in/wp-content/uploads/2025/06/HinglishEval.pdf

[13] P. Denny, D. H. Smith IV, M. Fowler, J. Prather, B. A. Becker, and J. Leinonen, "Explaining code with a purpose: An integrated approach for developing code comprehension and prompting skills," in *Proc*. 2024 Conf. Innovation and Technology in Computer Science Education V. 1 (ITiCSE 2024), Milan, Italy, 2024, pp. 283-289, doi: 10.1145/3649217.3653587.

[14] U. Singh, A. Sharma, N. Gupta, Deepakshi, and V. K. Jha, "IndicEval-XL: Bridging linguistic diversity in code generation across Indic languages," *arXiv:2502.19067*, Feb. 2025 (preprint). [Online]. Available: https://arxiv.org/abs/2502.19067

[15] M. Jordan, K. Ly, and A. G. Soosai Raj, "Need a programming exercise generated in your native language? ChatGPT's got your back: Automatic generation of non-English programming exercises using OpenAI GPT-3.5," in *Proc*. 55th ACM Tech. Symp. Computer Science Education V. 1 (SIGCSE 2024), 2024, pp. 618-624.

[16] J. Katzy et al., "Evaluating non-English developer support in machine learning for software engineering," *arXiv:2605.05902*, May 2026 (preprint). [Online]. Available: https://arxiv.org/abs/2605.05902

[17] M. Sheokand and P. Sawant, "CodeMixBench: Evaluating large language models on code generation with code-mixed prompts," *arXiv:2505.05063*, May 2025 (preprint). [Online]. Available: https://arxiv.org/abs/2505.05063

[18] J. Cohen, "A coefficient of agreement for nominal scales," *Educ*. Psychol. Meas., vol. 20, no. 1, pp. 37-46, 1960.