computers

MDPI

Article

PyChatAI: Enhancing Python Programming Skills—An Empirical Study of a Smart Learning System

Manal Alanazi [1], Ben Soh [1,*], Halima Samra [2] and Alice Li [3]

[1] Department of Computer Science and Information Technology, La Trobe University, Melbourne 3086, Australia; manalmohammedg.alanazi@latrobe.edu.au
[2] Computer Science & Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia; hsamra@kau.edu.sa
[3] La Trobe Business School, La Trobe University, Melbourne 3086, Australia; a.li@latrobe.edu.au
* Correspondence: b.soh@latrobe.edu.au

Abstract: This paper presents strategies for effectively integrating AI tools into programming education and provides recommendations for enhancing student learning outcomes through intelligent educational systems. Learning computer programming is a cognitively demanding task that requires dedication, logical reasoning, and persistence. Many beginners struggle with debugging and often lack effective problem-solving strategies. To address these issues, this study investigates PyChatAI—a bilingual, AI-powered chatbot designed to support novice Python programmers by providing real-time feedback, answering coding-related questions, and fostering independent problem-solving skills. PyChatAI offers continuous, personalised assistance and is particularly beneficial for students who prefer remote or low-pressure learning environments. An empirical evaluation employing a Solomon Four-Group design revealed significant improvements across all programming skill areas, with especially strong gains in theoretical understanding, code writing, and debugging proficiency.

Keywords: artificial intelligence; ChatGPT; natural language processing (NLP); OpenAI API; Python programming; computer programming; smart learning systems; Saudi Arabia

Academic Editor: Ananda Maiti

Received: 20 March 2025
Revised: 15 April 2025
Accepted: 20 April 2025
Published: 23 April 2025

Citation: Alanazi, M.; Soh, B.; Samra, H.; Li, A. PyChatAI: Enhancing Python Programming Skills—An Empirical Study of a Smart Learning System. *Computers* **2025**, *14*, 158. https://doi.org/10.3390/computers14050158

1. Introduction

With the rapid advancement of artificial intelligence (AI) and the ongoing digital transformation across industries, the demand for individuals with specialised expertise in computer science and programming has become increasingly urgent. Programming skills are now recognised as essential competencies for navigating the future workforce, driving innovation, and contributing to sustained economic growth. In Saudi Arabia, this shift is evident in the substantial rise in enrolment in computer science programs, in alignment with national strategic goals, such as Saudi Vision 2030.

The country has made significant progress in AI education, with 86% of universities offering undergraduate programs, 56% offering master's programs, and 9% offering PhD programs in the field. Between 2019 and 2023, more than 38,000 students graduated with specialisations in AI, including 6500 who completed their studies abroad. Remarkably, the number of AI graduates increased by 42% from 2022 to 2023 [1].

Despite these encouraging developments, challenges in computer programming education remain. A smart learning system—capable of providing real-time, AI-driven, and individualised support—holds great potential to address these gaps and enhance the effectiveness of programming instruction.

These challenges include regional disparities in educational quality, traditional and often passive teaching methods, language barriers, and limited access to instructional resources. Such obstacles significantly hinder student engagement, contributing to high dropout rates and inconsistent learning outcomes. Addressing these issues requires more than curriculum reform; it necessitates the development of intelligent, responsive, and student-centred learning environments [2].

Recent research indicates that while Saudi students generally possess strong digital literacy and positive attitudes toward technology, their use of digital tools for academic purposes remains limited. Factors such as geographic location, family support, and English language proficiency significantly influence their adoption of educational technologies [3]. Notably, many female computer science students in Saudi Arabia still rely on traditional study methods—such as handwritten notes, which often lack interactivity and real-time feedback mechanisms [4].

In response to these persistent challenges, this study introduces PyChatAI—a bilingual, AI-powered smart learning system designed specifically to support novice Python programmers. Unlike general-purpose AI tools, PyChatAI delivers culturally relevant, curriculum-aligned assistance tailored to the unique needs of Saudi learners. With features such as real-time feedback, adaptive guidance, and proactive error detection, it offers a novel and timely solution within the expanding landscape of AI-enhanced education.

2. Motivation and Research Objectives

Learning computer programming is a complex and demanding process that requires persistence, logical reasoning, and strong problem-solving skills. One of the most common challenges faced by beginners is debugging—an often frustrating task that can lead to the development of ineffective coding habits and reduced learner confidence [5].

To address these difficulties, AI-powered educational technologies—particularly smart learning systems, such as intelligent tutoring systems and chatbots—have emerged as promising solutions. These tools enhance student engagement by delivering personalised, real-time feedback, clarifying complex programming concepts, and fostering independent learning [6]. Tools such as ChatGPT, for example, have been well-received for their ability to simplify programming tasks and offer adaptive, context-sensitive responses [7,8].

However, many existing AI tools remain limited by their reactive design; they only provide assistance when explicitly prompted by the user. This restricts their capacity to proactively identify when students are struggling and intervene accordingly [9]. In addition, most tools are not curriculum-aligned, lack bilingual support, and are not tailored to specific educational contexts—making them less effective for non-English-speaking learners, particularly in Saudi Arabia.

To overcome these limitations, this study introduces PyChatAI—a smart learning system specifically designed to support beginner Python programmers. PyChatAI offers bilingual support (Arabic and English), integrates proactive error detection, and incorporates culturally relevant scaffolding based on Vygotsky's Zone of Proximal Development (ZPD) [10]. The tool aims to deliver context-aware, responsive assistance that promotes both technical skill development and learner motivation, in alignment with Saudi Arabia's national education and digital transformation goals.

2.1. Research Objectives

This study aims to achieve the following:

- Evaluate the effectiveness of PyChatAI in improving programming skills among novice learners.
- Identify factors influencing students' acceptance of smart learning systems using the Technology Acceptance Model (TAM3).

- Explore the role of AI-driven feedback and real-time support in facilitating personalised learning.
- Assess the potential of culturally adapted AI tools in supporting digital education goals, particularly in Saudi Arabia.

2.2. Research Questions

1. Does PyChatAI significantly improve students' programming skills across key dimensions (e.g., code writing and problem-solving)?
2. How do students perceive the usefulness and ease of use of PyChatAI?
3. What is the impact of the bilingual and proactive features of PyChatAI on user engagement and learning outcomes?
4. Can PyChatAI be effectively integrated into existing programming courses in a way that aligns with educational objectives in Saudi Arabia?

3. Literature Review

3.1. Existing Research and Tools

The integration of AI tools into computer programming education has rapidly advanced with the emergence of technologies such as ChatGPT, Pyo, Python-Bot, and Revision-Bot. These tools offer a range of functionalities designed to improve coding proficiency, provide real-time feedback, and enhance student engagement.

ChatGPT, a generative pre-trained transformer (GPT-4), represents a significant innovation in AI technology. Integrated into computer programming education, it assists students by generating code snippets, offering debugging support, and suggesting coding strategies. These features promote sustainable coding practices and enrich the learning experience [11–13]. ChatGPT also facilitates multiple concurrent interactions, creating a supportive, non-judgmental environment that encourages students to seek help comfortably. Additionally, its capabilities—such as error detection, automated code generation, and real-time assistance—enhance coding efficiency by reducing the time and effort needed for assignments [11].

However, ChatGPT is not specialised for programming education. It provides static responses that do not adapt to individual learners' levels and lacks capabilities for direct code execution or real-time code evaluation [11]. Despite these limitations, its educational impact is broad. It has gained considerable academic attention for its potential to transform teaching and learning through interactive, AI-driven approaches [14–16]. Many students perceive ChatGPT as a human-like assistant, appreciating its ability to simplify complex concepts and improve their learning experience [17].

Empirical studies underscore the effectiveness of ChatGPT in programming education. For example, one study found that students using ChatGPT-enhanced learning methods improved their exam scores from 48.33 to 74.47 [18]. Engaging with ChatGPT has also been linked to higher motivation and the development of cleaner, more structured code, with fewer rule violations, reduced complexity, and improved adherence to coding conventions [19,20].

Beyond individual assistance, AI-powered systems such as ChatGPT-based virtual agents are being developed to simulate programmer-like problem-solving techniques. These agents provide interactive and adaptive learning scenarios that enhance students' technical skills, critical thinking, and collaborative learning.

Building on the foundation established by ChatGPT, several specialised tools have been introduced:

- Pyo focuses on introductory Python programming. It offers real-time chatbot support to explain syntax, concepts, and debugging strategies [21]. While useful, it lacks deep personalisation and automated code assessment.
- Python-Bot and Revision-Bot support programming practice, revision, and exam preparation. These tools provide interactive feedback and automatic assessments of code quality, syntax, and logic. They also integrate with Learning Management Systems (LMSs) to track student progress [7,22]. However, their reliance on predefined question banks limits their adaptability to individual learning trajectories.
- KIAAA serves as an AI-driven intelligent tutoring system (ITS) with a focus on automation-related programming tasks. It offers customised learning paths and supports 3D simulation-based interactive execution [23].
- VoiceBots employ voice-based technology to support multimedia programming education, particularly in HTML and CSS. While they enhance accessibility and interactivity, they lack deep integration with core programming instruction.

While these tools have contributed significantly to the advancement of computer programming education, many still lack key features, such as proactive engagement, adaptive learning, and real-time code execution. Among these, adaptive learning is especially critical, as it allows AI systems to personalise instruction based on a student's existing knowledge, learning pace, and areas of difficulty. By dynamically tailoring feedback and content delivery, adaptive AI tools can offer more effective support than static, one-size-fits-all approaches. This adaptability enhances learning efficiency, increases student motivation, and helps reduce cognitive overload—particularly in complex domains, such as programming.

To better understand the current landscape, Table 1 presents a comparative overview of these AI-based educational tools, outlining their respective strengths and limitations in supporting computer programming education.

Table 1. Comparison of tools for learning computer programming.

Feature	ChatGPT	KIAAA	Pyo	VoiceBots	Xatkit
Year	**2023-2024-2025**	**2023**	**2022**	**2024**	**2020-2021-2024**
Primary Purpose	General AI assistant for various tasks, including programming.	AI-driven Intelligent Tutoring System (ITS) for automation programming education.	AI-based assistant for Python programming support.	Enhances multimedia programming education through VoiceBot technology.	AI-powered programming learning environment.
AI Model Used	GPT-4 (Natural Language Processing, NLP).	Help-DKT (AI cognitive model for student progress tracking).	Built on Rasa (Conversational AI framework).	IBM Watson AI Services.	NLP-based machine learning models.
Specialisation in Programming	Not specific to programming education.	Focused on automation-related programming.	Specialises in introductory Python programming.	Emphasises HTML and CSS in multimedia programming.	Supports multiple languages, including Python, Java, and C++.
Response to Coding Queries	Provides general programming explanations and debugging assistance.	Generates custom programming tasks and dynamically adapts to student responses.	Explains Python syntax, concepts, and debugging strategies.	Responds to multimedia programming inquiries in real time.	Provides immediate feedback, debugging support, and adaptive learning.
Real-Time Interaction	No direct code execution.	Supports interactive execution in a 3D simulation environment.	Offers real-time programming assistance via chatbot.	Provides instant feedback through WhatsApp integration.	Delivers live chatbot responses with adaptive learning.

Table 1. *Cont.*

Feature	ChatGPT	KIAAA	Pyo	VoiceBots	Xatkit
Adaptability to Student Level	Static responses, no adaptation to user expertise.	Monitors and adjusts difficulty dynamically based on student progress.	Provides guidance based on errors but lacks deep personalisation.	Customisable responses based on student engagement and feedback.	Adapts difficulty based on learning patterns.
Automated Code Evaluation	No real-time assessment.	AI-powered evaluation with automated feedback.	Identifies syntax errors and offers hints via multiple-choice prompts.	Offers automated responses but lacks code execution analysis.	Performs syntax validation, logic analysis, and efficiency checks.
Simulation & Hands-on Learning	No simulation capabilities.	3D virtual learning environment for practical coding tasks.	Encourages logical code structuring through guided exercises.	Provides interactive learning experiences in multimedia programming.	Supports hands-on coding exercises and problem-solving scenarios.
Integration with Educational Systems	No direct LMS or IDE support.	Compatible with IDEs and automation software.	Embedded in an online Python learning platform.	Integrated with WhatsApp and Learning Management Systems (LMS).	Supports LMS integration, classroom analytics, and API connectivity.
Assessment & Feedback	Provides only text-based explanations, no evaluation metrics.	AI-driven real-time feedback with automated assessment.	Offers hints, explanations, and solutions for debugging exercises.	Provides real-time feedback with additional learning resources.	Automated grading, performance tracking, and skill improvement recommendations.
Ref	[11,14,20,24–27]	[28]	[22]	[29]	[30–32]
Feature	**Revision-Bot**	**Python-Bot**	**Iris**	**e-JAVA Chatbot**	**PerFuSIT**
Year	**2022**	**2020**	**2024**	**2020**	**2024**
Primary Purpose	AI-based chatbot for personalised programming revision and learning support.	AI-based chatbot for programming practice and revision.	AI tutor in Artemis for personalised coding help	Virtual tutor for learning JAVA programming.	Adaptive Learning, Personalised Tutoring
AI Model Used	SnatchBot API with NLP and AIML (Artificial Intelligence Markup Language).	SnatchBot API with NLP and AIML.	GPT-3.5-Turbo with advanced prompting.	Rule-based chatbot with text-matching techniques.	Fuzzy Logic, Machine Learning
Specialisation in Programming	Python-focused, designed for exam practice and interactive problem-solving.	Python-focused, designed for exam practice and interactive problem-solving.	CS1 programming exercises, no direct solutions.	Focused on JAVA, particularly control structures.	Python, Java, C++
Response to Coding Queries	AI-driven question-answer chatbot with a predefined question bank.	AI-driven question-answer chatbot with a predefined question bank.	Hints, counter-questions, and explanations.	Provides instant responses using pre-defined patterns.	Real-Time Feedback, Debugging Assistance
Real-Time Interaction	AI-powered chatbot with interactive feedback.	AI-powered chatbot with interactive feedback.	Instant, context-aware responses.	Offers immediate feedback through chat-based input.	Dynamic Adjustments, Live Support

Table 1. *Cont.*

Feature	ChatGPT	KIAAA	Pyo	VoiceBots	Xatkit
Adaptability to Student Level	Tracks student progress and customises revision content accordingly.	Tracks student progress and customises revision content accordingly.	Adjusts to problem context, not skill level.	Limited adaptability; follows structured responses	Beginner to Advanced, Skill-Based Progression
Automated Code Evaluation	AI-driven assessment of code quality, syntax, and logic.	AI-driven assessment of code quality, syntax, and logic.	Uses Artemis's automated feedback.	Generates code samples but does not evaluate user code.	Syntax Analysis, Logical Error Detection
Simulation & Hands-on Learning	Practice-based revision tool with interactive coding challenges.	Practice-based revision tool with interactive coding challenges.	Real-time guidance, no simulations	Supports learning via structured problem-solving.	Interactive Coding Exercises, Virtual Labs
Integration with Educational Systems	Web-based chatbot with LMS compatibility and social media deployment.	Web-based chatbot with LMS compatibility and social media deployment.	Fully embedded in Artemis.	Standalone tool; not integrated with LMS or e-learning platforms.	LMS Compatibility, API Integration
Assessment & Feedback	AI-powered self-evaluation, student progress tracking, and score prediction.	AI-powered self-evaluation, student progress tracking, and score prediction.	Provides hints, not direct grading.	Provides structured responses but lacks personalised feedback.	Performance Tracking, Personalised Reports
Ref	[33]	[21]	[34]	[35]	[36]

3.2. Broader Educational Context

The rapid advancement in AI technologies has profoundly impacted higher education, particularly in the domain of programming instruction. AI-powered tools such as ChatGPT have garnered significant attention for their potential to enhance teaching and learning outcomes in academic settings [12]. By offering immediate feedback, assisting with coding tasks, and promoting interactive learning, these tools foster greater student engagement through AI-driven instructional methods [15,16].

Beyond programming-specific applications, AI-based tutors, such as Iris and the e-JAVA Chatbot, have been integrated into broader educational contexts. Iris provides real-time guidance and personalised feedback for CS1 programming exercises, while the *e*-JAVA Chatbot supports structured problem-solving in Java programming environments [21]. These tools demonstrate the versatility of AI in various learning contexts, although they still lack proactive engagement capabilities that would allow them to anticipate and respond to student difficulties without prompting.

Moreover, the integration of AI in education is reshaping the role of educators—from traditional content deliverers to mentors who facilitate conceptual understanding and guide students in developing critical problem-solving skills [7,23].

3.3. Theoretical Frameworks

The development of AI-based educational tools is frequently grounded in established learning theories, with Vygotsky's Zone of Proximal Development (ZPD) serving as a key theoretical framework. ZPD highlights the value of scaffolding—providing structured support that enables learners to accomplish tasks they would find difficult to complete unaided. This guided approach fosters the development of cognitive and problem-solving skills by gradually promoting learner autonomy [10].

In the context of computer programming education, integrating adaptive support mechanisms into educational tools can address common challenges faced by novice programmers, such as planning program logic and debugging complex code [24,37]. AI-driven systems that are designed around ZPD principles deliver personalised assistance that bridges the gap between what students can do independently and what they can accomplish with appropriate guidance, thereby enhancing both learning effectiveness and student confidence.

3.4. Gaps in Research

Despite the growing success of AI-driven tools, such as ChatGPT, Pyo, and Python-Bot, several important gaps persist in the current research and application landscape:

- Lack of Proactive Engagement: Most existing tools—including ChatGPT and Python-Bot—are reactive in nature, responding only when prompted by the user. They do not proactively detect or intervene when students encounter difficulties during coding tasks.
- Limited Adaptability: Tools such as ChatGPT provide static responses that do not adjust to a learner's skill level or learning trajectory. While systems such as KIAAA offer dynamic difficulty adjustments, they are not specifically designed for programming education [23].
- Absence of Real-Time Code Execution and Evaluation: Many AI-powered tools, including ChatGPT and VoiceBots, lack built-in capabilities for executing and evaluating code in real time, limiting their usefulness for hands-on programming practice [11,15].
- Integration Challenges: Although some tools can interface with Learning Management Systems (LMSs), seamless integration with Integrated Development Environments (IDEs) and other educational platforms remains limited, hindering a cohesive and streamlined learning experience [7,22].

These limitations underscore the need for specialised AI tools that deliver personalised, real-time support tailored to the unique demands of programming education—particularly at the introductory level.

To address these gaps, this study introduces PyChatAI, a purpose-built, bilingual chatbot developed for novice Python programmers. Building upon the foundational capabilities of ChatGPT, PyChatAI extends its functionality by offering interactive, adaptive programming support that aligns with learners' skill levels, educational contexts, and curriculum requirements. It is designed to proactively assist students in overcoming common programming challenges, thereby enhancing both learning outcomes and student engagement.

4. Methodology

4.1. Introduction to Methodology

This study aims to evaluate the effectiveness of PyChatAI, an AI-powered educational tool developed to support and enhance introductory Python programming instruction. This methodology section details the participant demographics, system development process, technical implementation, and the experimental framework used to assess the impact of PyChatAI on student learning outcomes. To ensure a robust and comprehensive evaluation, this study employs a mixed-methods approach, integrating both quantitative and qualitative techniques. These include the Solomon Four-Group Experimental Design, the Technology Acceptance Model (TAM3) questionnaire, and structured interviews with faculty members.

4.2. Participants and Setting

This research was conducted at the College of Computer Science and Information Technology at Jouf University, Saudi Arabia. The study involved 300 female first-year

undergraduate students enrolled in introductory Python programming courses during the first semester of the 2024 academic year. Participants were between 18 and 22 years old, and the majority had limited or no prior programming experience, with Python serving as their first formal programming language. To ensure the validity of the findings and minimise selection bias, participants were randomly assigned to control and experimental groups using a computer-generated randomisation method, ensuring a balanced group distribution and unbiased comparisons.

4.3. Experimental Design and Evaluation

To rigorously assess the effectiveness of PyChatAI, this study employs the Solomon Four-Group design—a well-established experimental methodology renowned for its ability to control for pre-test sensitisation and isolate the true effects of an intervention. This design is particularly effective in distinguishing whether observed improvements are attributable to the intervention itself or are influenced by participants' prior exposure to testing conditions.

4.3.1. The Structure of the Solomon Four-Group Design

The experimental setup consists of four groups: two control groups and two experimental groups. The groups vary based on whether participants received a pre-test and whether they were exposed to the intervention (use of PyChatAI). The structure of this design is summarised in Table 2.

Table 2. Solomon Four-Group design.

Group	Pre-Test	Intervention (Using PyChatAI)	Post-Test	Sample Size
Control Group One	No	No	Yes	50
Control Group Two	Yes	No	Yes	50
Experimental Group One	Yes	Yes	Yes	50
Experimental Group Two	No	Yes	Yes	50

4.3.2. Explanation of Design Components

- Pre-test: Measures participants' baseline programming knowledge and skills prior to the intervention. This establishes a reference point and helps assess whether initial proficiency influences the effectiveness of PyChatAI.
- Intervention (use of PyChatAI): represents the experimental conditions in which students interacted with PyChatAI during programming sessions to receive real-time assistance, debugging support, and contextual code feedback.
- Post-test: administered following the intervention to evaluate changes in programming performance, enabling a comparative analysis across all four groups.

4.3.3. Purpose of Research Design

The Solomon Four-Group design enables a rigorous evaluation of PyChatAI by facilitating balanced comparisons across groups exposed to different conditions. Specifically, the design isolates and measures two critical factors:

1. Effect of the Intervention:
   By comparing outcomes between the experimental groups (who used PyChatAI) and the control groups (who did not), this study quantitatively assessed the direct impact of PyChatAI on students' computer programming skills. This comparison provides robust evidence regarding the efficacy of the application in enhancing programming competencies.

2. Effect of Pre-testing:
   The inclusion of both pre-tested and non-pre-tested groups controls for potential testing effects that might influence post-test performance. This allowed this study to examine whether exposure to the pre-test itself contributes to improvements in computer programming skills, independent of the intervention.

4.3.4. Research Process Flowchart

Figure 1 illustrates the research process for evaluating PyChatAI. The process began with the pre-test, followed by the random assignment of 300 students into control and experimental groups. The experimental group received the PyChatAI intervention, while the control group continued with traditional instructional methods. After a four-month intervention phase, both groups completed a post-test to assess changes in computer programming skills.

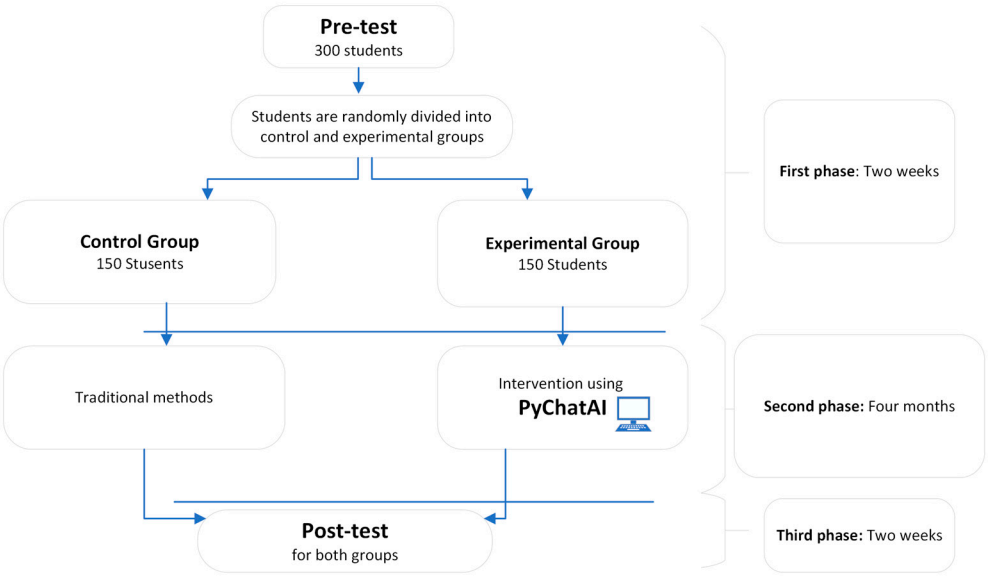

Figure 1. Flowchart illustrating the research process for evaluating PyChatAI.

4.4. System Development and Technical Implementation

PyChatAI was developed as a domain-specific smart learning system, integrating AI-driven feedback and real-time debugging support that leverages the OpenAI API for generating natural language responses tailored to Python programming tasks. The system provides real-time feedback, structured learning prompts, and debugging support to enhance the learning experience for novice programmers. The PyChatAI system development involves the following:

- Programming Language: Python 3.6+.
- User Interface (UI): developed using Tkinter for graphical interface design.
- AI Integration: utilises OpenAI API for generating context-specific programming responses.
- System Requirements:
  a. Operating Systems: Windows, macOS, or Linux.
  b. Dependencies: Tkinter, OpenAI API, and Pyperclip for clipboard operations.

Figures 2–6 illustrate various interface features of PyChatAI, including the AI Question interface (Figure 2), input windows for coding challenges (Figure 4), and predefined prompts guiding students through Python concepts (Figure 6). These interfaces were designed to ensure usability and interactive engagement for beginner programmers.

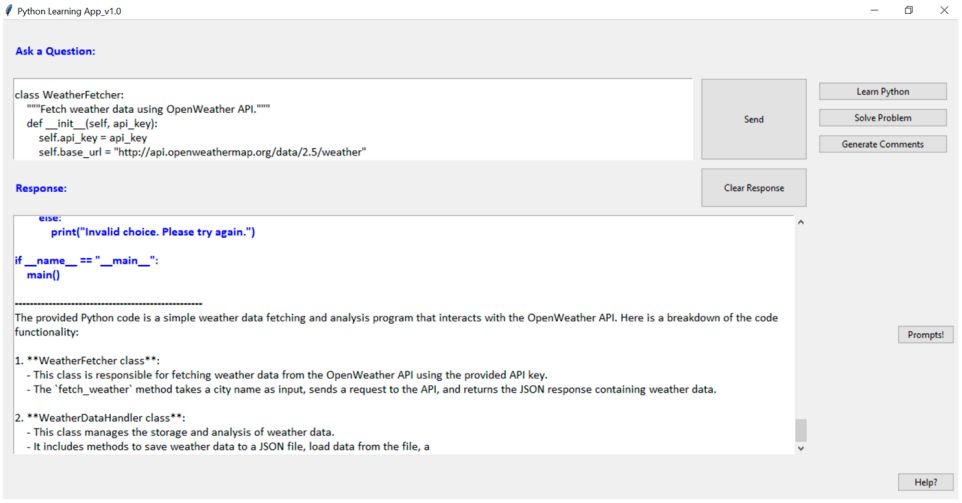

Figure 2. Screenshot of the Python Learning App_v1.0, illustrating code input, explanation, and interactive features for learning how to fetch data using the OpenWeather API.

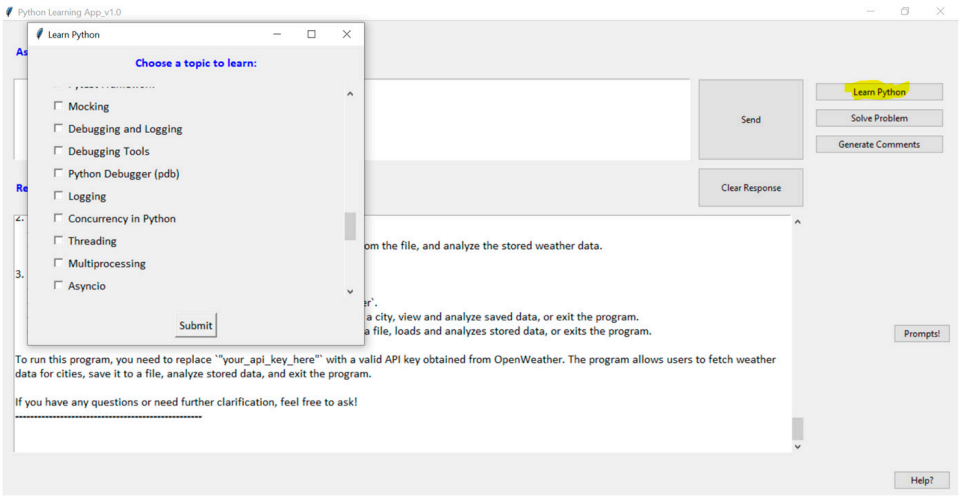

Figure 3. Screenshot of the "Learn Python" feature interface, displaying topic selection for interactive learning in the Python Learning App.

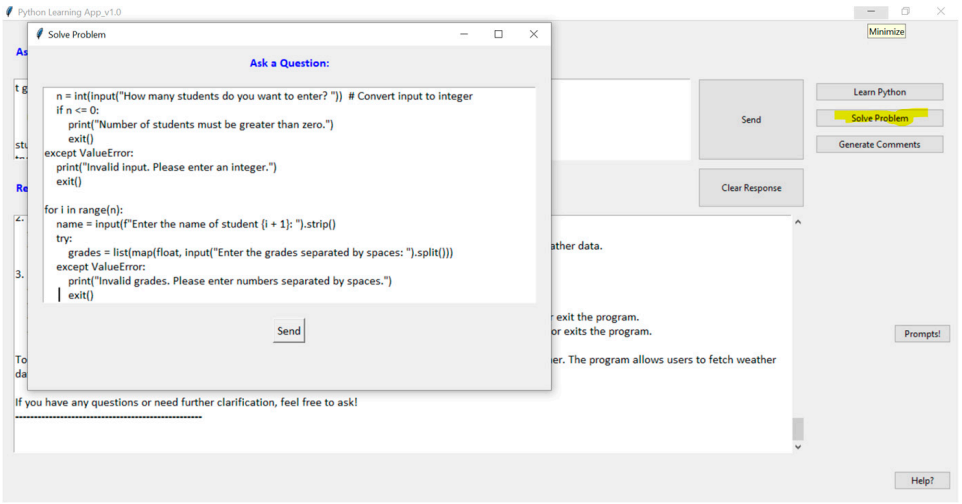

Figure 4. Screenshot of "Solve Problem" feature displaying a Python script input interface for entering student data and grades with basic error handling.

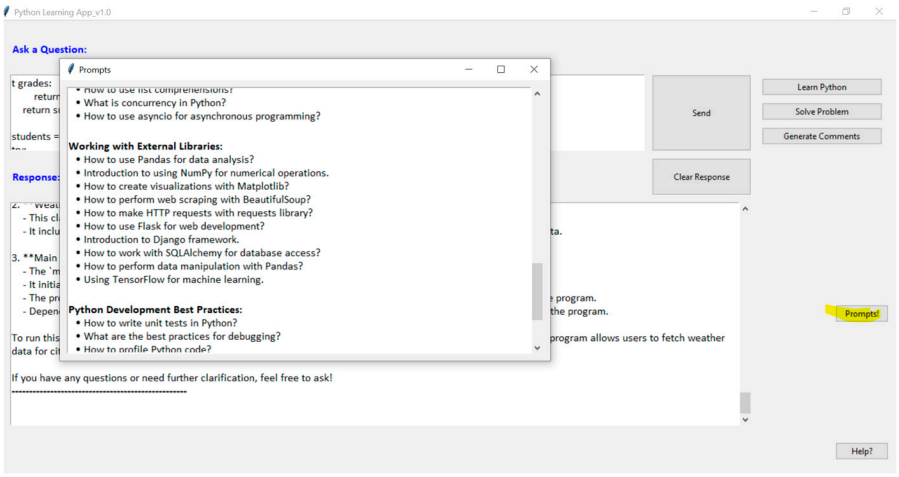

Figure 5. Screenshot of "Prompts" feature showing a categorised list of Python learning questions, covering external libraries and best development practices.

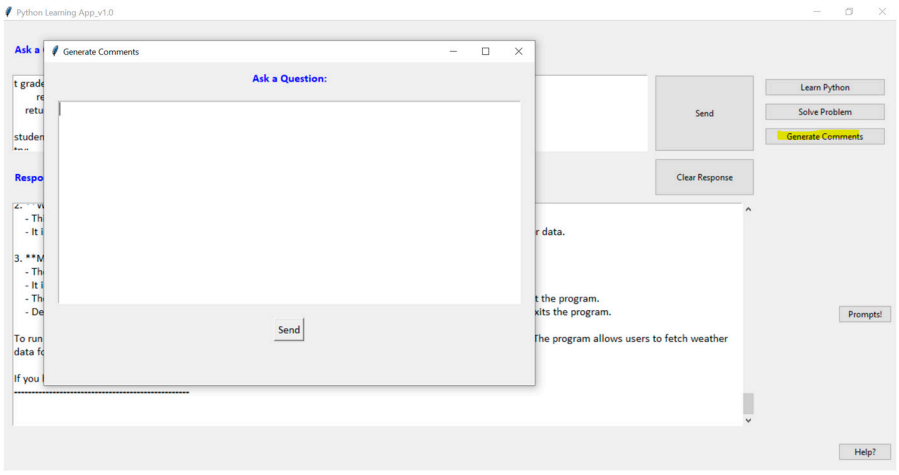

Figure 6. Screenshot of "Generate Comments" window in the Python Learning App, allowing users to input code and receive AI-generated comments to enhance code clarity.

4.4.1. Technical Contribution of PyChatAI

PyChatAI is not merely an OpenAI API wrapper but a domain-specific, adaptive learning system built with tailored features to support Python education in multilingual and culturally diverse contexts. The system was custom developed with the following technical contributions:

- Custom Prompt Engineering: PyChatAI uses domain-optimised prompt templates that scaffold learning based on student inputs, reducing hallucinations and ensuring responses are pedagogically aligned with Python course content.
- No Custom Dataset Training: The system does not train its own models but uses pre-trained OpenAI models. However, its performance is fine-tuned at runtime through controlled prompt engineering and contextual filtering.
- Adaptive Layer Built on OpenAI API: PyChatAI adds a real-time logic layer that monitors engagement patterns (e.g., frequent syntax errors and repeated queries) and offers tailored scaffolding. This adaptivity goes beyond simple request–response models.
- Arabic/English Multilingual Interface: the chatbot dynamically detects the language and adjusts UI elements, error messages, and responses accordingly.
- Debugging Assistant: this analyses common logic and syntax errors using pattern-matching and suggests improvements proactively based on detected coding issues.

4.4.2. Innovation Compared to Existing Tools

Unlike tools such as ChatGPT, Python-Bot, or Pyo, PyChatAI is purpose-built for academic Python instruction in Saudi Arabia. Table 3 provides a comparative analysis of PyChatAI, ChatGPT, and Python-Bot/Pyo across six key features relevant to educational support in Python programming. This table clearly shows that PyChatAI offers a more comprehensive and education-specific feature set than ChatGPT and Python-Bot/Pyo, especially in areas such as proactive feedback, adaptive support, bilingual interaction, and curriculum integration.

Table 3. Comparison of PyChatAI and other AI tools (ChatGPT and Python-Bot/Pyo) based on key educational features.

Feature	PyChatAI	ChatGPT	Python-Bot/Pyo
Proactive Error Detection	Yes	No	No
Adaptive Scaffolding	Yes	No	No
Arabic/English Support	Yes (bilingual UI/feedback)	Limited	No
Curriculum-Aligned Prompts	Yes	No	No
Context-Aware Learning Feedback	Yes	No	No
Built-in Debugging Assistance	Yes	Basic (if prompted)	No

4.4.3. Pedagogical Framework

The system incorporates Vygotsky's Zone of Proximal Development (ZPD) by adjusting the level of assistance based on the complexity of the task and user engagement. This scaffolding approach ensures students can solve problems they may not be able to handle independently, gradually building their confidence and skill.

4.5. PyChatAI System Features and Development Phases

The system was developed in two phases to progressively enhance its educational capabilities:

1. Phase 1: Core Functionalities
    a. Live Question Assistance: Users can submit Python-related questions and receive instant responses. Contextual filters ensure all responses remain specific to Python programming.
    b. Multilingual Support: The system supports both English and Arabic, making it accessible to a diverse learner base. The interface and chatbot dynamically adjust to the selected language.
2. Phase 2: Enhanced Features
    a. Predefined Learning Prompts: interactive prompts guide users through fundamental Python concepts, such as loops, conditionals, and functions (Figure 5).
    b. Problem Description Form: students describe their coding challenges, and PyChatAI generates a basic logic framework to aid problem-solving.
    c. Interactive Debugging Assistance: the tool detects syntax errors and provides corrective suggestions in real-time (Figure 3).
    d. Code Analysis and Feedback: students can submit code to receive a fully commented version, helping them understand coding best practices and improve code readability (Figure 4).

4.6. Data Collection and Analysis

The evaluation of PyChatAI involves both quantitative and qualitative methods to provide a comprehensive understanding of its educational impact.

1. Quantitative Data Collection:
    - Pre-tests and Post-tests: programming assessments administered before and after the intervention to measure changes in coding skills and problem-solving abilities.
    - Technology Acceptance Model (TAM3) Questionnaire: To investigate factors influencing student acceptance of PyChatAI, this study incorporates a TAM3-based questionnaire. This model evaluates key variables, such as perceived usefulness, ease of use, and behavioural intention to continue using the application.
2. Qualitative Data Collection:
    - Faculty Interviews: To gain a comprehensive understanding of the educational value of PyChatAI, faculty perspectives were gathered through structured interviews. These interviews focused on the effectiveness of PyChatAI in enhancing programming instruction, its integration into the curriculum, and potential areas for improvement.

This multi-faceted methodological approach enables a comprehensive evaluation of PyChatAI from both student and educator perspectives. It directly addresses the primary research objective of this study: to determine the effectiveness of PyChatAI in enhancing computer programming outcomes and to explore the key factors influencing its acceptance and adoption in educational settings.

5. Results

5.1. Pre-Test: Equivalence of Study Groups

To ensure the validity of this study's findings, it was essential to establish baseline equivalence between the control and experimental groups prior to the intervention. A pre-test was conducted across six core programming skill dimensions: theoretical understanding, code writing, software design, problem-solving and analysis, software testing and debugging, and overall programming competency.

An independent samples *t*-test was employed to compare mean scores between the groups. Assumptions of normality and homogeneity of variance were assessed using the Shapiro–Wilk and Levene's tests, respectively, and were satisfactorily met, confirming the suitability of the *t*-test for this analysis. In addition, effect sizes (Cohen's d) were calculated to determine the practical significance of any observed differences. The results indicate no statistically significant differences between the control and experimental groups across all measured domains. As summarised in Table 4, these findings affirm the initial equivalence of the groups, providing a reliable basis for evaluating the impact of the PyChatAI intervention.

Table 4. The results of the independent samples *t*-test for the pre-test of programming skills.

Skill	Group	N	Mean	Std. Deviation	T-Value	DF	*p*-Value
Theoretical Understanding	Control	150	2.17	1.14	0.593	298	0.554
	Experimental	150	2.26	1.56			
Code Writing Skills	Control	150	1.98	1.13	1.252	298	0.212
	Experimental	150	1.82	1.08			
Software Design	Control	150	1.77	1.13	1.534	298	0.126
	Experimental	150	1.56	1.21			
Problem Solving	Control	150	1.51	1.02	0.001	298	0.999
	Experimental	150	1.51	1.28			
Software Testing	Control	150	1.83	1.11	1.539	298	0.062
	Experimental	150	2.05	1.42			
Total Skills	Control	150	9.25	2.79	0.12	298	0.904
	Experimental	150	9.20	3.85			

Key Findings from the Pre-Test:

- For theoretical understanding, the control group (M = 2.17; SD = 1.14) and the experimental group (M = 2.26; SD = 1.56) showed no significant difference, t(298) = 0.593 and $p = 0.554$, indicating comparable baseline knowledge between the two groups.
- Similarly, in code-writing skills, the control group (M = 1.98; SD = 1.13) and the experimental group (M = 1.82; SD = 1.08) did not differ significantly, t(298) = 1.252 and $p = 0.212$, suggesting equivalent starting levels for this skill.
- In software design, no significant difference was observed between the control group (M = 1.77; SD = 1.13) and the experimental group (M = 1.56; SD = 1.21), t(298) = 1.534 and $p = 0.126$, confirming the groups' similarity prior to the intervention.
- For problem-solving skills, both groups had identical mean scores (M = 1.51), with no significant difference, t(298) = 0.001 and $p = 0.999$, further supporting the equivalence of the groups.
- In software testing and debugging, while the experimental group scored slightly higher (M = 2.05; SD = 1.42) than the control group (M = 1.83; SD = 1.11), this difference was not statistically significant, t(298) = 1.539 and $p = 0.062$.
- Finally, for total programming skills, the control group (M = 9.25; SD = 2.79) and the experimental group (M = 9.20; SD = 3.85) showed no significant difference, t(298) = 0.12 and $p = 0.904$, confirming overall equivalence across all measured skills.

5.2. Post-Test Results: Impact of PyChatAI

Programming skills were assessed post-intervention using a comprehensive programming skills test conducted after a four-month period. The results for both the control and experimental groups were analysed using the independent samples *t*-test to identify any significant differences between the two groups. This statistical method enabled a comparison of means to determine the impact of the PyChatAI application on the development of programming skills.

Table 5 presents the *t*-test results, highlighting the differences in programming skills across the measured dimensions for the two groups. These findings provide critical insights into the effectiveness of the intervention in enhancing programming skills among female computer science students at Jouf University.

Key Findings from the Post-Test:

The results, as summarised in Table 4, indicate statistically significant differences between the control and experimental groups across all measured dimensions of programming skills ($p < 0.001$ for all skills).

Table 5. Independent samples *t*-test results comparing programming skills between two groups.

Skill	Group	N	Mean	Std. Deviation	T-Value	DF	*p*-Value
Theoretical Understanding	Control Experimental	150 150	2.51 3.99	1.01 1.05	12.402	298	0.001>
Code Writing Skills	Control Experimental	150 150	2.31 4.02	1.01 1.01	14.719	298	0.001>
Software Design	Control Experimental	150 150	2.19 4.02	0.89 1.17	15.211	298	0.001>
Problem Solving	Control Experimental	150 150	2.09 4.03	0.99 1.21	15.096	298	0.001>
Software Testing	Control Experimental	150 150	2.22 4.00	0.90 1.12	15.120	298	0.001>
Total Skills	Control Experimental	150 150	11.33 20.05	2.53 4.40	21.068	298	0.001>

- The experimental group had a mean score of 3.99 in theoretical understanding compared to 2.51 for the control group, with a t-value of 12.402 and a p-value < 0.001, indicating a significant improvement due to the intervention.
- In code-writing skills, the experimental group achieved a mean score of 4.02, substantially higher than the control group's 2.31, with a t-value of 14.719 and a p-value < 0.001.
- The experimental group outperformed the control group in software design, with a mean score of 4.02 compared to 2.19, reflected by a t-value of 15.211 and a p-value < 0.001.
- In problem-solving, the experimental group's mean score of 4.03 significantly exceeded the control group's 2.09, with a t-value of 15.096 and a p-value < 0.001.
- The experimental group achieved a mean score of 4.00 in software testing, higher than the control group's 2.22, with a t-value of 15.120 and a p-value < 0.001.
- The total programming skill score showed a marked improvement in the experimental group, with a mean of 20.05 compared to 11.33 in the control group and a t-value of 21.068 with a p-value < 0.001, confirming the significant overall effectiveness of the PyChatAI application in enhancing programming skills.

Figures 7 and 8 show a comparison of computer programming skills between the control and experimental groups, demonstrating consistent and substantial improvements across all dimensions for students using PyChatAI. Figure 8 illustrates the distribution of computer programming skill scores, highlighting the higher median scores and reduced variance in the experimental group, which suggests both improvements and consistency in learning outcomes. Figure 9 compares the mean scores across different computer programming skills, clearly showing the superior performance of the experimental group in all categories.

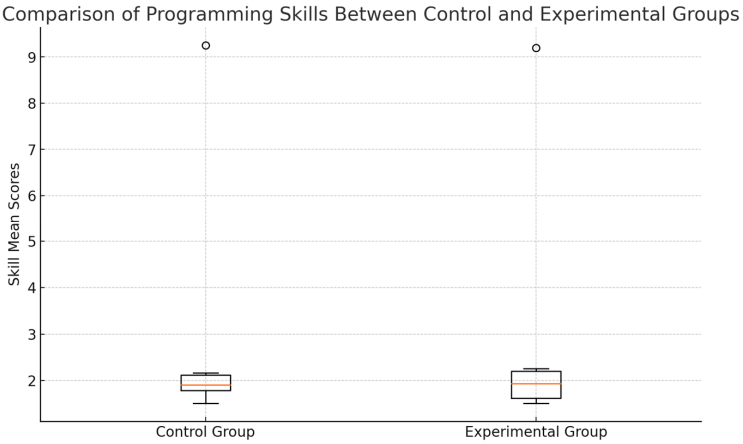

Figure 7. Boxplot comparison of programming skill mean scores between control and experimental groups, highlighting score distributions and outliers.

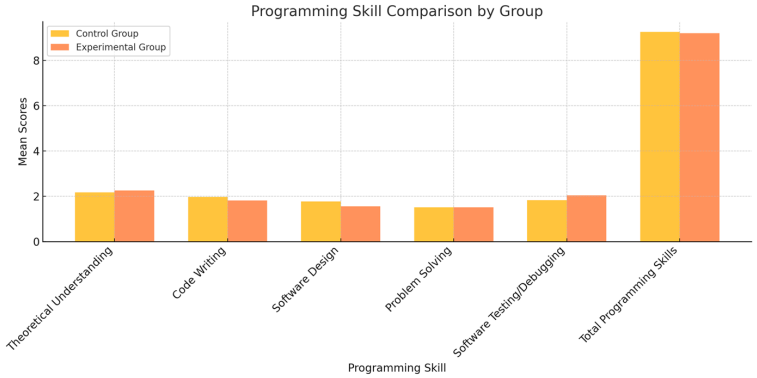

Figure 8. Mean programming skill scores by group, comparing control and experimental groups across individual skills and total performance.

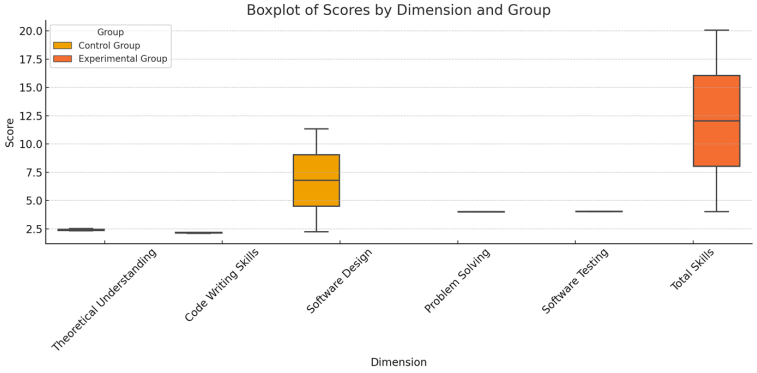

Figure 9. Boxplot showing differences in programming skill scores between control and experimental groups across six dimensions.

6. Discussion

6.1. Interpretation of Pre-Test Results

The pre-test results confirm that both the control and experimental groups possessed equivalent baseline skills across all dimensions of computer programming, including theoretical understanding, code writing, software design, problem-solving, and software testing. This statistical equivalence is crucial, as it ensures that any differences observed in the post-test are the result of the PyChatAI intervention, rather than pre-existing disparities between the groups. By establishing that both groups started with comparable skill levels, this study maintains strong internal validity, allowing for a clear attribution of post-intervention improvements to the use of the AI-powered educational tool. This baseline similarity provides a solid foundation for evaluating the impact of PyChatAI and supports the reliability of the conclusions of this study.

6.2. Impact of PyChatAI on Computer Programming Skills

The post-test results reveal a significant positive impact of PyChatAI as a smart learning system, demonstrating its effectiveness in the development of computer programming skills among the experimental group. The data demonstrate substantial improvements across all measured dimensions, indicating that the intervention effectively enhanced students' abilities in both theoretical knowledge and practical application.

In the areas of theoretical understanding and code writing, the experimental group showed marked improvements compared to the control group. Theoretical understanding is fundamental for comprehending computer programming principles, and the observed gains suggest that PyChatAI successfully facilitated students' grasp of complex concepts. Similarly, the improvement in code-writing skills indicates that the AI tool effectively

supported students in translating their theoretical knowledge into practical coding abilities. This is likely due to the ability of PyChatAI to provide instant feedback, suggest code modifications, and guide students through common programming errors, fostering a more interactive and engaging learning experience.

The experimental group's superior performance in problem-solving and software testing further underscores the effectiveness of PyChatAI as a smart learning system that fosters computational thinking and coding proficiency. Problem-solving is a critical skill in computer programming, requiring not only technical knowledge but also analytical thinking and creativity. The significant gains in this area suggest that the interactive features of PyChatAI, such as real-time debugging support and tailored problem-solving prompts, played a key role in helping students develop these competencies. Likewise, improvements in software testing and debugging skills highlight the role of the tool in enhancing students' error detection and correction abilities, which are essential for writing efficient and reliable code.

The most compelling evidence of the impact of PyChatAI is reflected in the total programming skill score. The experimental group achieved a mean score of 20.05, nearly double that of the control group's 11.33, indicating comprehensive support across multiple dimensions of computer programming. This dramatic improvement suggests that PyChatAI not only helps students understand individual concepts but also fosters a more holistic development of computer programming proficiency.

These findings are visually supported by Figures 9 and 10, which illustrate the differences in mean scores across various computer programming dimensions. The visual data reinforce the robustness of the intervention, clearly showing the experimental group's superior performance in all skill areas. The consistent improvements across multiple dimensions highlight the ability of PyChatAI to provide an integrated learning experience that addresses both foundational and advanced programming skills.

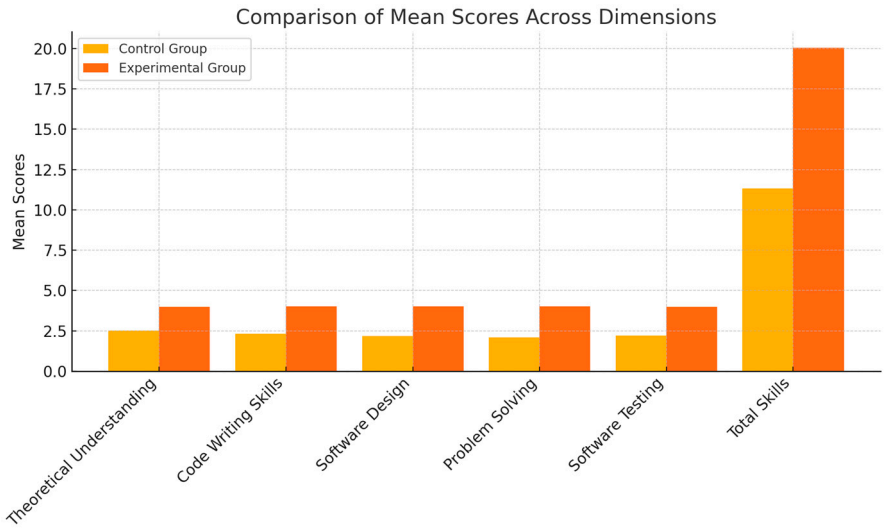

Figure 10. Bar chart comparing mean programming scores across dimensions for control and experimental groups.

6.3. Implications for Educational Practice

The significant improvements observed in the experimental group suggest that AI-powered smart learning systems, such as PyChatAI, have the potential to transform computer programming education by providing real-time feedback and personalised assistance. One of the key advantages of PyChatAI is its ability to provide real-time feedback and assistance. Unlike traditional teaching methods, which often involve delayed feedback,

PyChatAI offers instant responses to students' queries. This immediacy fosters continuous engagement, helps students correct mistakes as they occur, and reduces the frustration commonly associated with programming challenges.

Moreover, the adaptive support mechanisms of PyChatAI enable personalised learning experiences. The tool tailors its feedback and guidance based on individual student needs, making it particularly effective for novice programmers who may struggle with the steep learning curve associated with programming. This level of personalisation helps students build confidence and maintain motivation, which are critical factors in successful learning.

The effectiveness of the tool in improving problem-solving and software debugging skills also highlights its role in promoting critical thinking and analytical abilities. These skills are not only essential for programming but are also valuable in a wide range of academic and professional contexts. By fostering these competencies, PyChatAI contributes to the development of students' overall cognitive abilities, preparing them for more complex tasks and challenges in the future.

While the results of this study demonstrate the effectiveness of PyChatAI as a smart learning system, several limitations should be acknowledged. First, this study focused exclusively on female students at a single institution (Jouf University in Saudi Arabia). While this focus provided valuable insights into a specific demographic, it also limits the generalisability of the findings. Future research should include more diverse student populations, encompassing different genders, institutions, and cultural contexts, to determine whether the effectiveness of PyChatAI is consistent across various groups.

In addition, while PyChatAI showed significant improvements in Python programming, it remains unclear whether similar results would be observed in other programming languages or more advanced programming topics. Future studies should explore the applicability of the tool across different programming environments and complexity levels to better understand its broader potential.

Lastly, the current study focused on the short-term impact of PyChatAI over a four-month period. While the results are promising, longitudinal studies are needed to examine the sustained impact of AI-assisted learning on computer programming skills. Understanding whether the improvements observed in this study persist over time would provide valuable insights into the long-term benefits of integrating AI tools into educational settings.

In summary, the findings of this study highlight the transformative potential of AI-powered educational tools, such as PyChatAI, in enhancing computer programming skills. By providing real-time feedback, fostering critical thinking, and offering personalised learning experiences, PyChatAI represents a significant advancement in computer programming education. However, further research is needed to explore its applicability in broader contexts and to assess the long-term impact of AI-assisted learning on student outcomes.

7. Conclusions

In response to the growing demand for personalised, inclusive programming education, this study introduced PyChatAI, a bilingual AI-powered learning system designed to support novice Python learners through real-time feedback, adaptive guidance, and proactive problem-solving support.

Using a Solomon Four-Group design with 300 female students at Jouf University, this study demonstrated that PyChatAI significantly enhanced programming competencies across key areas—including code writing, theoretical understanding, and debugging. Compared to control groups, students using PyChatAI nearly doubled their overall programming performance. Unlike general-purpose tools, such as ChatGPT, PyChatAI offers curriculum-aligned, proactive support tailored to learners' progress. TAM3 responses confirmed high student acceptance, citing ease of use and perceived usefulness, while

faculty feedback underscored its value in large classroom settings. The system's bilingual capabilities also helped reduce language barriers, supporting broader educational equity and aligning with the digital transformation goals of Saudi Vision 2030.

The outcomes of this study suggest significant relevance beyond the immediate setting. PyChatAI's design offers a model for developing AI-powered educational systems that are culturally localised, curriculum-aligned, and proactively assistive features that go beyond what existing tools, such as ChatGPT or Python-Bot, provide. As AI becomes increasingly integrated into education, such smart systems may play a transformative role in building equitable and personalised learning environments.

Despite these promising results, this study has certain limitations. It focused exclusively on female students at a single institution, limiting the generalisability of the findings. In addition, this study only evaluated short-term gains over a four-month period. Future research should aim for the following:

- Examine the system's applicability across diverse learner populations and institutions.
- Test PyChatAI in other programming languages and advanced computing topics.
- Explore long-term retention and the sustained impact of AI-assisted learning.
- Investigate how integrating real-time performance analytics and reinforcement learning could further personalise the learning experience.

In conclusion, PyChatAI demonstrates strong potential as a scalable and effective smart learning system capable of transforming computer programming education. By building on the strengths of existing tools, such as ChatGPT—while introducing proactive engagement, bilingual support, and alignment with local curricula—PyChatAI offers a uniquely adaptable solution for supporting novice programmers. Its proven ability to enhance student outcomes highlights the promise of AI-driven educational technologies in fostering more inclusive, personalised, and adaptive learning environments in computer programming education both in Saudi Arabia and globally.

Author Contributions: Conceptualization, M.A. and B.S.; methodology, M.A. and B.S.; software, M.A.; validation, M.A., B.S. and H.S.; formal analysis, M.A., B.S. and H.S.; investigation, M.A., B.S. and H.S.; resources, M.A., B.S. and A.L.; data curation, M.A. and H.S.; writing—original draft preparation, M.A. and B.S.; writing—review and editing, B.S., A.L. and H.S.; visualization, M.A., B.S. and H.S.; supervision, B.S. and A.L.; project administration, M.A., B.S., H.S. and A.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in the study are included in the article; further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Alharthi, F. Challenges related to computer programming education in Saudi Arabia. *Proc. Eng.* **2024**, *6*, 37–44. [CrossRef]
2. Alothman, M.; Robertson, J.; Michaelson, G. Computer usage and attitudes among Saudi Arabian undergraduate students. *Comput. Educ.* **2017**, *110*, 127–142. [CrossRef]
3. Alanazi, A.; Li, A.; Soh, B. Effects on Saudi Female Student Learning Experiences in a Programming Subject Using Mobile Devices: An Empirical Study. *Int. J. Interact. Mob. Technol.* **2023**, *17*, 44–58. [CrossRef]
4. Alasmari, O.A.; Singer, J.; Ada, M.B. Do current online coding tutorial systems address novice programmer difficulties? In Proceedings of the 15th International Conference on Education Technology and Computers, Barcelona, Spain, 26–28 September 2023; pp. 242–248.
5. Zhong, X.; Zhan, Z. An intelligent tutoring system for programming education based on informative tutoring feedback: System development, algorithm design, and empirical study. *Interact. Technol. Smart Educ.* **2024**. *ahead-of-print*.
6. Abdulla, S.; Ismail, S.; Fawzy, Y.; Elhag, A. Using ChatGPT in Teaching Computer Programming and Studying Its Impact on Students Performance. *Electron. J. E-Learn.* **2024**, *22*, 66–81. [CrossRef]

7. Mekthanavanh, V.; Meksavanh, B.; Bouddy, S. ChatGPT Enhances Programming Skills of Computer Engineering Students. *Souphanouvong Univ. J. Multidiscip. Res. Dev.* **2024**, *10*, 1–9. [CrossRef]
8. Stone, I. Investigating the Use of ChatGPT to Support the Learning of Python Programming Among Upper Secondary School Students: A Design-Based Research Study. In Proceedings of the 2024 Conference on United Kingdom & Ireland Computing Education Research, Manchester, UK, 5–6 September 2024; p. 1.
9. Maguire, J.; Cutts, Q. Supporting the Computing Science Education Research Community with Rolling Reviews. In Proceedings of the United Kingdom & Ireland Computing Education Research Conference, Glasgow, UK, 3–4 September 2020; ACM: New York, NY, USA, 2020.
10. Van Der Stuyf, R.R. Scaffolding as a teaching strategy. *Adolesc. Learn. Dev.* **2002**, *52*, 5–18.
11. Yilmaz, R.; Yilmaz, F.G.K. The effect of generative artificial intelligence (AI)-based tool use on students' computational thinking skills, programming self-efficacy and motivation. *Comput. Educ. Artif. Intell.* **2023**, *4*, 100147. [CrossRef]
12. Rahman, M.; Watanobe, Y. ChatGPT for education and research: Opportunities, threats, and strategies. *Appl. Sci.* **2023**, *13*, 5783. [CrossRef]
13. Silva, C.A.G.D.; Ramos, F.N.; De Moraes, R.V.; Santos, E.L.D. ChatGPT: Challenges and benefits in software programming for higher education. *Sustainability* **2024**, *16*, 1245. [CrossRef]
14. Tick, A. Exploring ChatGPT's Potential and Concerns in Higher Education. In Proceedings of the 2024 IEEE 22nd Jubilee International Symposium on Intelligent Systems and Informatics (SISY), Pula, Croatia, 19–21 September 2024.
15. Wieser, M.; Schöffmann, K.; Stefanics, D.; Bollin, A.; Pasterk, S. Investigating the role of ChatGPT in supporting text-based programming education for students and teachers. In Proceedings of the International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, Lausanne, Switzerland, 23–25 October 2023; Springer Nature: Cham, Switzerland, 2023; pp. 40–53.
16. Ma, B.; Chen, L.; Konomi, S.I. Enhancing programming education with ChatGPT: A case study on student perceptions and interactions in a Python course. In Proceedings of the International Conference on Artificial Intelligence in Education, Recife, Brazil, 8–12 July 2024; Springer: Berlin/Heidelberg, Germany, 2024; pp. 113–126.
17. Jalon, J.B.; Chua, G.A.; Torres, M.D.L. ChatGPT as a learning assistant: Its impact on students learning and experiences. *Int. J. Educ. Math. Sci. Technol.* **2024**, *12*, 1603–1619. [CrossRef]
18. Akçapınar, G.; Sidan, E. AI chatbots in programming education: Guiding success or encouraging plagiarism. *Discov. Artif. Intell.* **2024**, *4*, 87. [CrossRef]
19. Sandu, R.; Gide, E.; Elkhodr, M. The role and impact of ChatGPT in educational practices: Insights from an Australian higher education case study. *Discov. Educ.* **2024**, *3*, 71. [CrossRef]
20. Haindl, P.; Weinberger, G. Does ChatGPT help novice programmers write better code? Results from static code analysis. *IEEE Access* **2024**, *12*, 114146–114156. [CrossRef]
21. Okonkwo, C.W.; Ade-Ibijola, A. Python-bot: A chatbot for teaching python programming. *Eng. Lett.* **2020**, *29*, 25.
22. Carreira, G.; Silva, L.; Mendes, A.J.; Oliveira, H.G. Pyo, a chatbot assistant for introductory programming students. In Proceedings of the 2022 International Symposium on Computers in Education (SIIE), Coimbra, Portugal, 17–19 November 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1–6.
23. Abolnejadian, M.; Alipour, S.; Taeb, K. Leveraging chatgpt for adaptive learning through personalized prompt-based instruction: A cs1 education case study. In Proceedings of the Extended Abstracts of the CHI Conference on Human Factors in Computing Systems, Honolulu, HI, USA, 11–16 May 2024; Association for Computing Machinery: New York, NY, USA, 2024; pp. 1–8.
24. Popovici, M.-D. ChatGPT in the classroom. Exploring its potential and limitations in a functional programming course. *Int. J. Hum. Comput. Interact.* **2024**, *40*, 7743–7754. [CrossRef]
25. Vukojičić, M.; Krstić, J. ChatGPT in programming education: ChatGPT as a programming assistant. *InspirED Teach. Voice* **2023**, *2023*, 7–13.
26. Xue, Y.; Chen, H.; Bai, G.R.; Tairas, R.; Huang, Y. Does ChatGPT help with introductory programming? An experiment of students using ChatGPT in cs1. In Proceedings of the 46th International Conference on Software Engineering: Software Engineering Education and Training, Lisbon, Portugal, 14–20 April 2024; pp. 331–341.
27. Güner, H.; Er, E.; Akçapinar, G.; Khalil, M. From chalkboards to AI-powered learning. *Educ. Technol. Soc.* **2024**, *27*, 386–404.
28. Eilermann, S.; Wehmeier, L.; Niggemann, O.; Deuter, A. KIAAA: An AI assistant for teaching programming in the field of automation. In Proceedings of the 2023 IEEE 21st International Conference on Industrial Informatics (INDIN), Lemgo, Germany, 18–20 July 2023; pp. 1–7.
29. Essel, H.B.; Vlachopoulos, D.; Nunoo-Mensah, H.; Amankwa, J.O. Exploring the impact of VoiceBots on multimedia programming education among Ghanaian university students. *Br. J. Educ. Technol.* **2025**, *56*, 276–295. [CrossRef]
30. Daniel, G.; Cabot, J.; Deruelle, L.; Derras, M. Xatkit: A multimodal low-code chatbot development framework. *IEEE Access* **2020**, *8*, 15332–15346. [CrossRef]
31. Daniel, G.; Cabot, J. The software challenges of building smart chatbots. In Proceedings of the 2021 IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), Madrid, Spain, 25–28 May 2021; pp. 324–325.

32. Daniel, G.; Cabot, J. Applying model-driven engineering to the domain of chatbots: The Xatkit experience. *Sci. Comput. Program.* **2024**, *232*, 103032. [CrossRef]
33. Okonkwo, C.W.; Ade-Ibijola, A. Revision-Bot: A Chatbot for Studying Past Questions in Introductory Programming. *IAENG Int. J. Comput. Sci.* **2022**, *49*, 644.
34. Bassner, P.; Frankford, E.; Krusche, S. Iris: An ai-driven virtual tutor for computer science education. In Proceedings of the 2024 on Innovation and Technology in Computer Science Education, Milan, Italy, 8–10 July 2024; Association for Computing Machinery: New York, NY, USA, 2024; Volume 1, pp. 394–400.
35. Daud, S.H.M.; Teo, N.H.I.; Zain, N.H.M. Ejava chatbot for learning programming language: Apost-pandemic alternative virtual tutor. *Int. J.* **2020**, *8*, 3290–3298.
36. Chrysafiadi, K.; Virvou, M. PerFuSIT: Personalized Fuzzy Logic Strategies for Intelligent Tutoring of Programming. *Electronics* **2024**, *13*, 4827. [CrossRef]
37. Biňas, M.; Pietriková, E. Impact of virtual assistant on programming novices' performance, behavior and motivation. *Acta Electrotech. Inform.* **2022**, *22*, 30–36. [CrossRef]