Generative AI for Programming Education: Can ChatGPT Facilitate the Acquisition of Fundamental Programming Skills for Novices?

Shakib Sadat Shanto
Department of Computer Science,
American International
University-Bangladesh (AIUB)
Dhaka, Bangladesh
shakibsss080@gmail.com

Zishan Ahmed
Department of Computer Science,
American International
University-Bangladesh (AIUB)
Dhaka, Bangladesh
zishanahmed599@gmail.com

Akinul Islam Jony
Department of Computer Science,
American International
University-Bangladesh (AIUB)
Dhaka, Bangladesh
akinul@aiub.edu

ABSTRACT

Modern Generative AI (GAI) systems like ChatGPT have sparked much interest in their potential to revolutionize programming education, especially for beginners. However, the existing empirical data regarding the effectiveness of technologies like ChatGPT as autonomous programming tutors is presently limited. The present study investigates the capacity of ChatGPT to facilitate the acquisition of fundamental programming skills for novice programmers without human assistance. This study puts forth a conceptual framework (APEC - Adaptive Programming Education via ChatGPT) that integrates both bottom-up and top-down approaches, incorporating ChatGPT as the principal instructor for the study of programming. An empirical study was undertaken to assess the usefulness of ChatGPT as a tool for teaching novice programmers a new programming language. This empirical study was conducted on 20 undergraduate students. To provide an expert assessment of the quality of the responses, a survey was conducted with three programming experts proficient in Python. The survey findings indicate that ChatGPT is proficient in explaining core principles such as variables, data types, and control statements through conversational exchanges, adopting an intelligent and logical methodology. Nevertheless, certain constraints arise when dealing with increasingly complex topics.

CCS CONCEPTS

• **Computing methodologies → Artificial intelligence**; • **Social and professional topics → Computer science education**.

KEYWORDS

Generative AI, ChatGPT, Programming Education, Educational Technology, Higher Education

ACM Reference Format:
Shakib Sadat Shanto, Zishan Ahmed, and Akinul Islam Jony. 2024. Generative AI for Programming Education: Can ChatGPT Facilitate the Acquisition of Fundamental Programming Skills for Novices?. In *Proceedings of 3rd International Conference on Computing Advancements - ICCA 2024 (ICCA 2024).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Artificial intelligence has reached a critical juncture with the advent of advanced GAI systems such as ChatGPT. ChatGPT has undergone extensive training using a vast and varied corpus of textual data from various sources on the internet [4, 15]. By extracting patterns and structures from extensive textual data, these models have developed a notable ability to generate coherent and contextually appropriate responses to prompts in natural language. Even so, it is crucial to recognize that despite their high level of intelligence, these systems do not possess actual cognition or intentionality. The responses provided by the model are mainly derived from the identification and extrapolation of statistical patterns within the data it has been trained on. The social effects surrounding the implementation of AI systems, which possess significant capabilities but are constrained in certain aspects, continue to be a subject of intense discussion and controversy [20]. While accepting the limitations of a particular entity or concept, it is imperative to acknowledge its significant usefulness within appropriate boundaries equally. Domains such as customer service [21], creative writing assistance [19], and programming education [24] exemplify areas in which these technologies can enhance human talents and increase productivity. However, the emergence of sophisticated generative artificial intelligence systems like ChatGPT has generated significant attention to their capacity to revolutionize educational methodologies in various fields [2, 14], including the teaching of computer programming [3]. Nevertheless, despite the considerable attention and speculations around these emerging technologies, significant academic studies still need to be investigated for the effectiveness of tools such as ChatGPT in reliably teaching fundamental coding abilities without human assistance. This study focuses to fill the existing research gap by conducting a comprehensive examination of ChatGPT's effectiveness as a distinct programming teacher for beginner students who are learning the basics of Python.

This study aims to investigate the effectiveness of ChatGPT in facilitating independent learning of expertise in the Python programming language among students who have no prior knowledge in programming. This study examines the accuracy and standard of ChatGPT's explanations regarding fundamental concepts and its ability to provide informative sample code. This study presents

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ICCA 2024, October 17-18, 2024, AIUB, Dhaka, Bangladesh
© 2024 Association for Computing Machinery.
ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

and assesses a well-defined conceptual framework that integrates bottom-up and top-down approaches to facilitate independent learning using ChatGPT. In addition, this study evaluates the effectiveness of the proposed framework through an empirical study of 20 undergraduate students divided into two groups - one taught programming in the traditional manner and the other using the proposed framework. Moreover, a survey of three Python experts was undertaken to gain their perspectives on the framework and the response of ChatGPT in assisting novices to learn programming fundamentals independently.

The key findings indicate that ChatGPT offers reasonable assistance regarding fundamental Python concepts. However, it needs to improve in fostering the development of more advanced skills. Experts recommend integrating human tutoring with ChatGPT to maximize outcomes. This research generates novel insights into ChatGPT's promise and pitfalls as an independent programming tutor, establishing an empirical foundation to guide the thoughtful and strategic incorporation of generative AI in computer science education.

2 LITERATURE REVIEW

Learning computer programming languages has been extensively studied within education and cognitive science research. A consistent observation evident across multiple prior studies is that acquiring proficiency in programming and effectively instructing foundational concepts poses substantial difficulties [13, 18]. Research findings also indicate that students have challenges in the areas of reading [12], writing [10], and programming code design [23]. This challenge arises from the diverse array of advanced skills required, including critical thinking and problem-solving abilities, proficiency in computer programming syntax and practical expertise in debugging and implementing solutions.

Several approaches have been suggested and assessed to facilitate the early phases of acquiring programming skills [7]. Effective pedagogical tactics include the utilization of visual block-based languages as a precursor to shifting towards text-based coding [9], the implementation of problem-based learning using real-world illustrations [11], and the provision of scaffolding to facilitate the development of planning and tracing abilities [25]. Research has also demonstrated that pair programming and interactive instructors can effectively enhance engagement and foster skill development among those new to a particular domain [6].

The evolution of programming language learning has paralleled technical improvements, wherein generative AI tools such as ChatGPT have emerged as promising facilitators. Several studies have investigated the potential of utilizing ChatGPT in programming instruction within an experimental framework [3]. When using AI-based tools and settings for programming learning, students possess the capacity to submit problems to the AI tool, thereby obtaining prompt feedback and solutions. This enables a customized educational experience per the student's unique learning pace. AI-powered tools have the potential to assist students with coding tasks through the provision of ideas, error detection, and the automation of code development [1]. This approach can potentially enhance students' ability to produce both efficient and precise code, hence minimizing the time and effort needed to fulfill programming tasks. The utilization of AI-driven tools and environments has the potential to enhance student engagement and motivation through interactive interaction and individualized assistance and feedback in the process of acquiring programming skills [27].

So, investigating self-directed learning of programming fundamentals exclusively through AI tools such as ChatGPT, without human guidance, presents a promising domain for research exploration. This study targets to investigate the potential of ChatGPT in facilitating the independent acquisition of fundamental ideas in a specific programming language among novice students. The focus of this study revolves around basic research questions:

(1) Can ChatGPT effectively teach beginners a programming language without human help?
(2) How accurately can ChatGPT explain programming concepts and provide sample code to novice learners compared to human tutors?
(3) How do experts view the quality of programming guidance provided by ChatGPT compared to traditional methods?
(4) What differences can be observed in learning outcomes on programming competence between students taught through the proposed framework and those learning via traditional pedagogical methods?

3 METHODOLOGY

Figure 1 depicts the sequential procedures utilized in the research. The methodology consists of four main components: (1) development of a conceptual framework for learning programming using ChatGPT, (2) generation and analysis of ChatGPT responses to programming-related prompts, (3) expert evaluation of ChatGPT's teaching quality through a survey, and (4) an empirical study comparing traditional learning methods with the proposed framework.

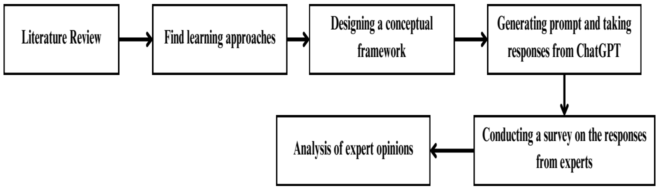

Literature Review
Find learning approaches
Designing a conceptual framework
Generating prompt and taking responses from ChatGPT
Conducting a survey on the responses from experts
Analysis of expert opinions

Figure 1: Methodological Approach of the study

3.1 Programming Language Learning Approaches

There are two distinct methodologies for acquiring proficiency in a programming language: the bottom-up approach (BUA) and the top-down approach (TDA).

3.1.1 Bottom-Up Approach (BUA). Learning the fundamentals of a programming language—variables, data types, functions, and grammar—is given top priority in the bottom-up method. With a thorough understanding of these foundational concepts, students can gradually use them to develop their skills in a logical manner, building a strong technical base. This method is frequently used in programming books for beginners and educational institutions [16].

3.1.2 Top-Down Approach (TDA). In order to help students comprehend language functionality, the top-down method gives priority to an educational technique where students are first introduced to sample programs [17]. The application of TDA expedites the process of acquiring language; it takes less time than BUA. With this method of learning, students may build real-world projects faster, which increases their level of engagement. A defined curriculum or set of learning objectives is given to students, with an emphasis on understanding, modifying, and validating code. TDA encourages a thorough comprehension of programming blocks as opposed to isolated code segments. All things considered, the top-down approach initially downplays technological complexity in favor of thorough conception and realistic application.

3.1.3 Approach Used by ChatGPT. ChatGPT is an instructional framework that teaches programming languages to beginners by skillfully combining top-down and bottom-up methods. By offering basic explanations and details, ChatGPT uses a bottom-up methodology to help users develop a solid understanding of programming principles. Simultaneously, it applies a top-down approach by giving students real-world coding scenarios to understand how the language functions within the context of practical applications. When this dual strategy is used, students receive a comprehensive education. Students are provided with a solid foundation in programming fundamentals through methodical explanations. Combining top-down and bottom-up approaches ensures that students gain the ability to write functional code as well as a thorough understanding of grammar and structure.

3.2 Framework for Learning Programming Using ChatGPT

A conceptual framework called APEC (Adaptive Programming Education via ChatGPT) has been proposed to facilitate learning programming using ChatGPT. This framework primarily targets individuals new to programming but can also be adapted to accommodate those with varying levels of expertise. While initially designed for beginners, the framework has the potential to benefit learners at all proficiency levels who have a strong inclination toward programming. This framework enables dynamic and inclusive learning environments by utilizing ChatGPT as a guiding tool without human assistance. It proves valuable to anyone interested in programming, regardless of their existing knowledge in the field. Figure 2 shows the proposed framework APEC.

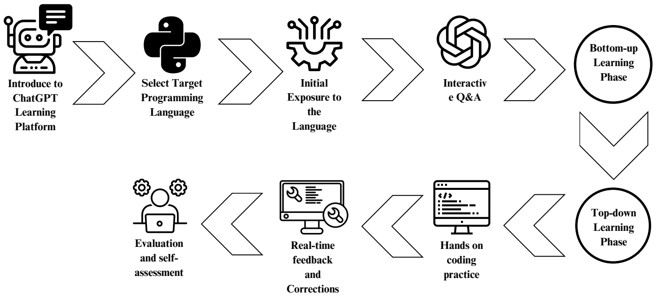

Introduce to ChatGPT Learning Platform
Select Target Programming Language
Initial Exposure to the Language
Interactiv e Q&A
Bottom-up Learning Phase
Top-down Learning Phase
Hands on coding practice
Real-time feedback and Corrections
Evaluation and self-assessment

Figure 2: APEC (Adaptive Programming Education via ChatGPT) Framework

(1) Introduction to ChatGPT Learning Platform:
Familiarize students with the ChatGPT platform as a self-learning resource for programming.
(2) Selecting the Targeted Programming Language:
Students choose the specific programming language they want to learn.
(3) Initial Exposure to the Language:
ChatGPT provides an overview of the selected programming language, introducing basic concepts and terminology.
(4) Interactive Q&A Sessions:
Students engage in interactive sessions with ChatGPT, asking questions related to programming language fundamentals.
(5) Bottom-Up Learning (BUA) Phase:
ChatGPT explains foundational concepts, syntax, and data structures, assisting students in building a solid base.
(6) Top-Down Learning (TDA) Phase:
ChatGPT presents sample coding scenarios, allowing students to observe practical implementations of language concepts.
(7) Hands-On Coding Practice:
Students apply their learning by actively writing code with ChatGPT's guidance.
(8) Real-Time Feedback and Corrections:
ChatGPT provides instant feedback on code accuracy and helps correct errors, encouraging iterative learning.
(9) Evaluation and Self-Assessment:
Students evaluate their understanding and coding skills by solving coding challenges independently.

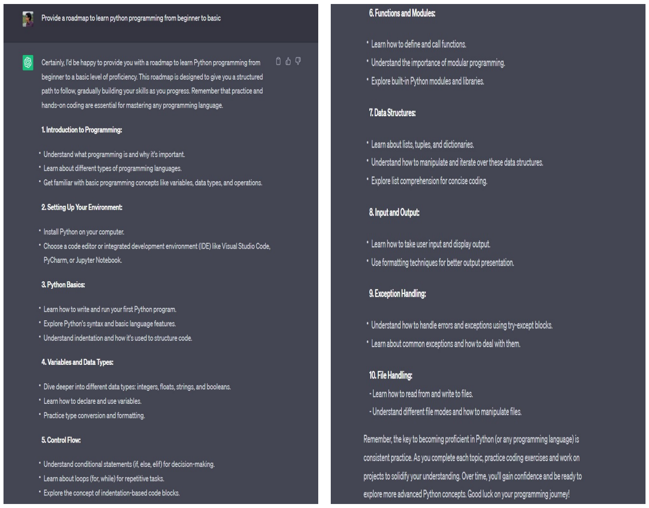
The proposed framework integrates two instructional approaches commonly employed in programming education: a foundational approach, which prioritizes the comprehension of fundamental concepts, and a practical approach, which emphasizes applying these concepts in real-world contexts. Using ChatGPT as an instructional tool, our objective is to assist people new to programming. This methodology tackles two prominent inquiries: Is it possible for ChatGPT to provide programming training to beginner learners without any human intervention? Moreover, can ChatGPT's explanations and code examples attain a level of quality comparable to that offered by human tutors?

The initial query was established to assess the independent instructional capabilities of ChatGPT. Our study aimed to evaluate the usefulness of learning programming exclusively through interactions with ChatGPT without the presence of human teachers. To evaluate the effectiveness of the technique, ChatGPT was utilized as a tool to provide a full teaching resource for individuals seeking to develop expertise in Python programming. This resource covers fundamental topics at an introductory level. The quality of the generated solutions was further assessed by incorporating expert comments and perspectives.

The following query examines the distinct instructional approaches utilized by ChatGPT and human educators. This study will evaluate the potential of ChatGPT in explaining programming ideas and offering informative code snippets. Subsequently, a comparison will be made between these strategies and those employed by human educators, facilitating an assessment of ChatGPT's instructional effectiveness concerning human capabilities.

3.3 Using ChatGPT for Learning Python Programming Language

To evaluate the ability of ChatGPT to offer concise and organized instructional assistance for individuals who are new to programming languages, the following questions were asked. Python was selected as the programming language for this research. Using a syntactically simple programming language, such as Python, compared to a complex language like Java, has proven to enhance students' understanding and learning of programming principles [8]. Initially, ChatGPT's guidance was requested to obtain a systematic plan to pursue learning proficiency in Python programming as an individual with limited experience in the field. Subsequently, the system was asked to instruct on diverse Python modules encompassing topics such as variables and data types, data structures, as well as input and output mechanisms.



Provide a roadmap to learn python programming from beginner to basic

Certainly, I'd be happy to provide you with a roadmap to learn Python programming from beginner to a basic level of proficiency. This roadmap is designed to give you a structured path to follow, gradually building your skills as you progress. Remember that practice and hands-on coding are essential for mastering any programming language.

1. Introduction to Programming:

- Understand what programming is and why it's important.
- Learn about different types of programming languages.
- Get familiar with basic programming concepts like variables, data types, and operations.

2. Setting Up Your Environment:

- Install Python on your computer.
- Choose a code editor or integrated development environment (IDE) like Visual Studio Code, PyCharm, or Jupyter Notebook.

3. Python Basics:

- Learn how to write and run your first Python program.
- Explore Python's syntax and basic language features.
- Understand indentation and how it's used to structure code.

4. Variables and Data Types:

- Dive deeper into different data types: integers, floats, strings, and booleans.
- Learn how to declare and use variables.
- Practice type conversion and formatting.

5. Control Flow:

- Understand conditional statements (if, else, elif) for decision-making.
- Learn about loops (for, while) for repetitive tasks.
- Explore the concept of indentation-based code blocks.

(a) Part-1

6. Functions and Modules:

- Learn how to define and call functions.
- Understand the importance of modular programming.
- Explore built-in Python modules and libraries.

7. Data Structures:

- Learn about lists, tuples, and dictionaries.
- Understand how to manipulate and iterate over these data structures.
- Explore list comprehension for concise coding.

8. Input and Output:

- Learn how to take user input and display output.
- Use formatting techniques for better output presentation.

9. Exception Handling:

- Understand how to handle errors and exceptions using try-except blocks.
- Learn about common exceptions and how to deal with them.

10. File Handling:

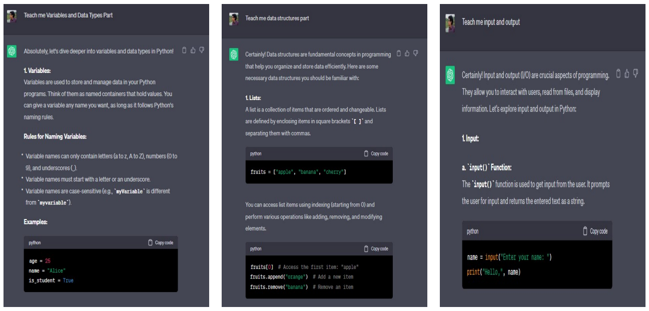
- Learn how to read from and write to files.
- Understand different file modes and how to manipulate files.

Remember, the key to becoming proficient in Python (or any programming language) is consistent practice. As you complete each topic, practice coding exercises and work on projects to solidify your understanding. Over time, you'll gain confidence and be ready to explore more advanced Python concepts. Good luck on your programming journey!

(b) Part-2

Figure 3: Beginner Python learning roadmap response from ChatGPT. *Prompt: Provide a roadmap to learn Python programming from beginner to basic.*

By assessing the output created by ChatGPT, its efficacy in facilitating the understanding of foundational Python concepts can be evaluated. Furthermore, ChatGPT was asked to offer practical coding exercises in Python for novice learners in order to assess its efficacy as an instructor in facilitating programming education through active engagement in practice. Finally, the suitability of ChatGPT for delivering real-time feedback and corrections while debugging a specific issue was assessed.



Teach me Variables and Data Types Part

Absolutely, let's dive deeper into variables and data types in Python!

1. Variables:
Variables are used to store and manage data in your Python programs. Think of them as named containers that hold values. You can give a variable any name you want, as long as it follows Python's naming rules.

Rules for Naming Variables:

- Variable names can only contain letters (a to z, A to Z), numbers (0 to 9), and underscores (_).
- Variable names must start with a letter or an underscore.
- Variable names are case-sensitive (e.g., `myVariable` is different from `myvariable`).

Examples:

```python
age = 25
name = "Alice"
is_student = True
```

(a)

Teach me data structures part

Certainly! Data structures are fundamental concepts in programming that help you organize and store data efficiently. Here are some necessary data structures you should be familiar with:

1. Lists:
A list is a collection of items that are ordered and changeable. Lists are defined by enclosing items in square brackets `[ ]` and separating them with commas.

```python
fruits = ["apple", "banana", "cherry"]
```

You can access list items using indexing (starting from 0) and perform various operations like adding, removing, and modifying elements.

```python
fruits[0]  # Access the first item: "apple"
fruits.append("orange")  # Add a new item
fruits.remove("banana")  # Remove an item
```

(b)

Teach me input and output

Certainly! Input and output (I/O) are crucial aspects of programming. They allow you to interact with users, read from files, and display information. Let's explore input and output in Python:

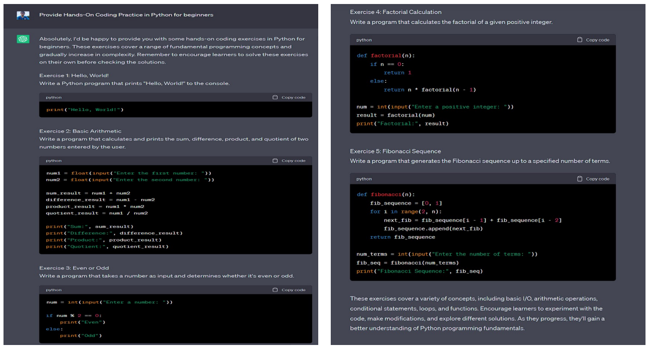
1. Input:

a. `input()` Function:
The `input()` function is used to get input from the user. It prompts the user for input and returns the entered text as a string.

```python
name = input("Enter your name: ")
print("Hello,", name)
```

(c)

Figure 4: Samples of teaching each part of the ChatGPT. Here, ChatGPT has been prompted to provide instruction on each component of its own roadmap that has been offered. *Prompt: Teach me each part of the provided roadmap.*



Provide Hands-On Coding Practice in Python for beginners

Absolutely, I'd be happy to provide you with some hands-on coding exercises in Python for beginners. These exercises cover a range of fundamental programming concepts and gradually increase in complexity. Remember to encourage learners to solve these exercises on their own before checking the solutions.

Exercise 1: Hello, World!
Write a Python program that prints "Hello, World!" to the console.

```python
print("Hello, World!")
```

Exercise 2: Basic Arithmetic
Write a program that calculates and prints the sum, difference, product, and quotient of two numbers entered by the user.

```python
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))

sum_result = num1 + num2
difference_result = num1 - num2
product_result = num1 * num2
quotient_result = num1 / num2

print("Sum:", sum_result)
print("Difference:", difference_result)
print("Product:", product_result)
print("Quotient:", quotient_result)
```

Exercise 3: Even or Odd
Write a program that takes a number as input and determines whether it's even or odd.

```python
num = int(input("Enter a number: "))

if num % 2 == 0:
    print("Even")
else:
    print("Odd")
```

(a)

Exercise 4: Factorial Calculation
Write a program that calculates the factorial of a given positive integer.

```python
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)

num = int(input("Enter a positive integer: "))
result = factorial(num)
print("Factorial:", result)
```

Exercise 5: Fibonacci Sequence
Write a program that generates the Fibonacci sequence up to a specified number of terms.

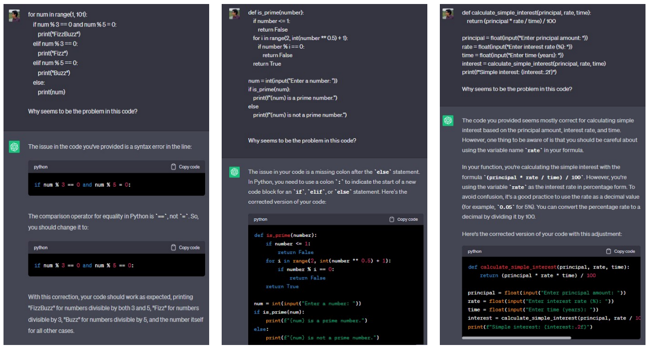
```python
def fibonacci(n):
    fib_sequence = [0, 1]
    for i in range(2, n):
        next_fib = fib_sequence[i - 1] + fib_sequence[i - 2]
        fib_sequence.append(next_fib)
    return fib_sequence

num_terms = int(input("Enter the number of terms: "))
fib_seq = fibonacci(num_terms)
print("Fibonacci Sequence:", fib_seq)
```

These exercises cover a variety of concepts, including basic I/O, arithmetic operations, conditional statements, loops, and functions. Encourage learners to experiment with the code, make modifications, and explore different solutions. As they progress, they'll gain a better understanding of Python programming fundamentals.

(b)

Figure 5: Hands-on coding sample response from ChatGPT. *Prompt: Provide Hands-On Coding Practice in Python for beginners.*



```
for num in range(1, 101):
  if num % 3 == 0 and num % 5 = 0:
    print("FizzBuzz")
  elif num % 3 == 0:
    print("Fizz")
  elif num % 5 == 0:
    print("Buzz")
  else:
    print(num)
```

Why seems to be the problem in this code?

The issue in the code you've provided is a syntax error in the line:

```python
if num % 3 == 0 and num % 5 = 0:
```

The comparison operator for equality in Python is `==`, not `=`. So, you should change it to:

```python
if num % 3 == 0 and num % 5 == 0:
```

With this correction, your code should work as expected, printing "FizzBuzz" for numbers divisible by both 3 and 5, "Fizz" for numbers divisible by 3, "Buzz" for numbers divisible by 5, and the number itself for all other cases.

(a)

```
def is_prime(number):
  if number <= 1:
    return False
  for i in range(2, int(number ** 0.5) + 1):
    if number % i == 0:
      return False
  return True

num = int(input("Enter a number: "))
if is_prime(num):
  print(f"{num} is a prime number.")
else
  print(f"{num} is not a prime number.")
```

Why seems to be the problem in this code?

The issue in your code is a missing colon after the `else` statement. In Python, you need to use a colon `:` to indicate the start of a new code block for an `if`, `elif`, or `else` statement. Here's the corrected version of your code:

```python
def is_prime(number):
    if number <= 1:
        return False
    for i in range(2, int(number ** 0.5) + 1):
        if number % i == 0:
            return False
    return True

num = int(input("Enter a number: "))
if is_prime(num):
    print(f"{num} is a prime number.")
else:
    print(f"{num} is not a prime number.")
```

(b)

```
def calculate_simple_interest(principal, rate, time):
  return (principal * rate / time) / 100

principal = float(input("Enter principal amount: "))
rate = float(input("Enter interest rate (%): "))
time = float(input("Enter time (years): "))
interest = calculate_simple_interest(principal, rate, time)
print(f"Simple interest: {interest:.2f}")
```

Why seems to be the problem in this code?

The code you provided seems mostly correct for calculating simple interest based on the principal amount, interest rate, and time. However, one thing to be aware of is that you should be careful about using the variable name `rate` in your formula.

In your function, you're calculating the simple interest with the formula `(principal * rate / time) / 100`. However, you're using the variable `rate` as the interest rate in percentage form. To avoid confusion, it's a good practice to use the rate as a decimal value (for example, `0.05` for 5%). You can convert the percentage rate to a decimal by dividing it by 100.

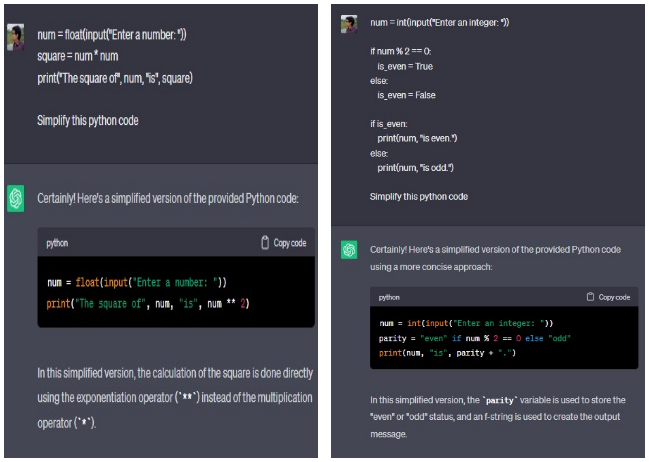
Here's the corrected version of your code with this adjustment:

```python
def calculate_simple_interest(principal, rate, time):
    return (principal * rate * time) / 100

principal = float(input("Enter principal amount: "))
rate = float(input("Enter interest rate (%): "))
time = float(input("Enter time (years): "))
interest = calculate_simple_interest(principal, rate / 10
print(f"Simple interest: {interest:.2f}")
```

(c)

Figure 6: Real-time coding corrections sample response from ChatGPT. ChatGPT was presented with a series of erroneous codes and requested to assist in identifying and rectifying the issues. The codes contained both theoretical and syntactical mistakes. *Prompt: Real-Time Feedback and Corrections.*



```
num = float(input("Enter a number: "))
square = num * num
print("The square of", num, "is", square)
```

Simplify this python code

Certainly! Here's a simplified version of the provided Python code:

```python
num = float(input("Enter a number: "))
print("The square of", num, "is", num ** 2)
```

In this simplified version, the calculation of the square is done directly using the exponentiation operator (`**`) instead of the multiplication operator (`*`).

(a)

```
num = int(input("Enter an integer: "))

if num % 2 == 0:
  is_even = True
else:
  is_even = False

if is_even:
  print(num, "is even.")
else:
  print(num, "is odd.")
```

Simplify this python code

Certainly! Here's a simplified version of the provided Python code using a more concise approach:

```python
num = int(input("Enter an integer: "))
parity = "even" if num % 2 == 0 else "odd"
print(num, "is", parity + ".")
```

In this simplified version, the `parity` variable is used to store the "even" or "odd" status, and an f-string is used to create the output message.

(b)

Figure 7: Real-time coding feedback sample response from ChatGPT. Multiple codes were sent to ChatGPT, which have the potential for additional improvement or simplification. The provided code snippets were written inefficiently.

Figure 3-7 demonstrates the sample responses provided by ChatGPT for given prompts.

4 ANALYSIS AND DISCUSSIONS

4.1 Survey Questions

A survey was conducted to accurately evaluate ChatGPT's quality in various tasks, such as interactive programming, providing structured roadmaps, and code debugging. The survey involved developing a customized collection of questions closely aligned with the framework's main objectives. These well-crafted questions, aligned with research goals, were used in a survey of three Python experts. Their insights focused on evaluating the quality and effectiveness of ChatGPT's responses.

The Likert Scale was employed as the measurement scale for the survey questions. The Likert Scale is widely recognized as a reliable and valid instrument for assessing individuals' subjective opinions and attitudes in an organized manner [26]. The survey employs a scale that gives participants different response alternatives to indicate their degree of agreement or disagreement with a given topic. Participants in the study offer a numerical value to their responses, which spans from "Strongly Disagree" (1) to "Strongly Agree" (5) [22]. This scale is standard in surveys and questionnaires to get a wide range of thoughts and perspectives from participants. Then, these data can be examined and interpreted to make well-informed decisions. Table 1 shows the survey questions given to three specialists in the Python programming language to assess how well ChatGPT helped beginners learn programming.

4.2 Expert Opinion Analysis

Following the collection of feedback from three Python experts, their answers to each survey question were carefully examined and contrasted to identify commonalities and discrepancies. The combination of expert viewpoints unveils common trends and noteworthy specifics. This procedure ultimately facilitates a comprehensive exploration of expert opinions regarding the efficacy of ChatGPT in the domain of programming education, enabling an analysis of the experts' feedback. Three educators who instruct university-level programming courses participated in the survey. No participant-specific personal information was gathered or disclosed. All three experts were drawn from the same university, which may have led to some similarity in their perspectives and expertise.

Table 1: Survey questionnaire

Question
1. Main ChatGPT can effectively teach beginners the following Python concepts: i) Variables and data types ii) Control Statements iii) Functions and Modules iv) Data Structures v) Exception Handling vi) File Handling
2. ChatGPT has the potential to teach beginners programming without teachers -
3. ChatGPT can be used to personalize the learning experience for each beginner-
4. ChatGPT can be used to provide feedback to beginners in a timely and constructive way-
5. ChatGPT can effectively help beginners with Hands-On Coding Practice -
6. ChatGPT can provide Real-Time Feedback and Corrections for beginners –
7. ChatGPT can help promote deep conceptual understanding and real-life implementation capabilities of learners –
8. ChatGPT maintains a Bottom-Up Approach while generating responses –
9. ChatGPT maintains a Top-Down Approach while generating responses –
10. The quality of programming guidance provided by ChatGPT is better compared to traditional methods –

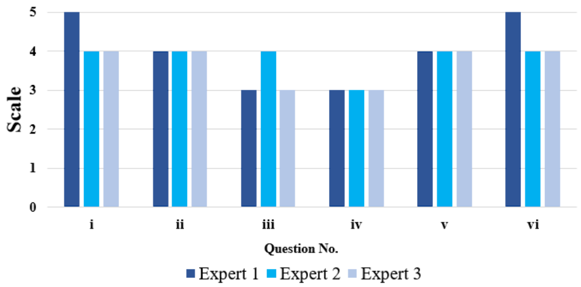


Figure 8: Expert Responses to Question 1. Question 1 has 6 sub-questions, from i – vi.

Figure 8 illustrates expert responses to ChatGPT's ability to teach foundational Python concepts. Experts generally agree that ChatGPT effectively teaches basic concepts like variables, data types, control statements, exception handling, and file handling. However, they express neutrality regarding ChatGPT's effectiveness in teaching functions, modules, and data structures, indicating areas for improvement. Further analysis revealed specific challenges in explaining complex topics. For instance, ChatGPT struggled to provide in-depth explanations of advanced data structures like balanced trees or graph algorithms. Its explanations of concepts such as deep vs. shallow copying in Python and decorators often lacked depth and practical examples. When explaining decorators, ChatGPT provided basic definitions but failed to comprehensively illustrate how they modify function behavior at runtime. These observations align with the experts' neutral stance on ChatGPT's ability to teach advanced concepts effectively, highlighting the need for human oversight and supplementary resources when using ChatGPT for teaching complex programming topics.

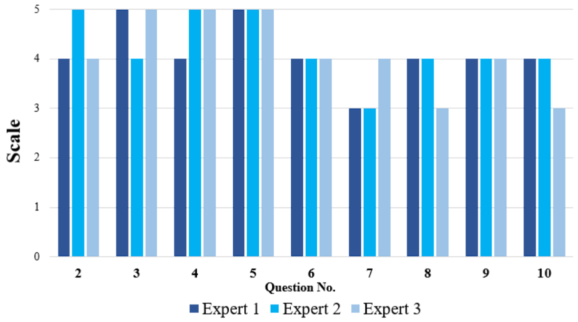


Figure 9: Expert Responses to Questions 2-10.

Figure 9 depicts the experts' responses concerning survey questions "2" to "10". In general, all three experts provided mostly good responses to the majority of the questions. In questions 2, 3, and 4, the expert assigns ratings that span from "Agree" (4) to "Strongly Agree" (5). Based on their responses, those surveyed concur or strongly agree that ChatGPT has the potential to facilitate programming education for beginners in the absence of instructors. Furthermore, they acknowledge its ability to tailor the learning process to individual beginners and deliver timely and constructive feedback. In response to issue 5, all three experts express a strong agreement regarding the effectiveness of ChatGPT in facilitating hands-on coding practice for novices. All respondents indicated agreement in both questions 6 and 9, with an average rate of 4. The observation suggests that ChatGPT possesses the capacity to offer real-time feedback and corrections to individuals who are new to programming. Additionally, it maintains a top-down approach while generating responses. In relation to questions "8" and "10", it is observed that expert 1 and expert 2 express agreement with a rating of "4", whilst expert 3 expresses a neutral stance with a rating of "3". Based on agreement among experts, ChatGPT follows a Bottom-Up Approach in its response generation. Furthermore, the programming guidance offered by ChatGPT exhibits superior quality when compared to conventional methodologies. In contrast, the majority of experts express a neutral stance, neither agreeing nor disagreeing (3), regarding question 7. This suggests that ChatGPT has limitations in fostering profound conceptual comprehension and practical application skills among learners, indicating the need for further improvement.

5 EMPIRICAL STUDY

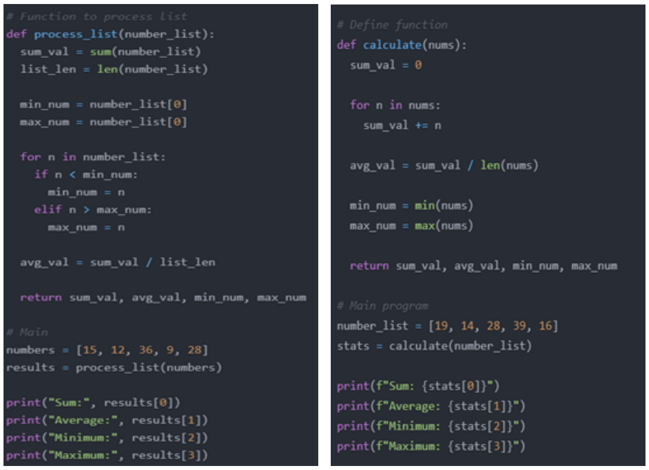
An empirical study was conducted to evaluate the effect of the framework in a real-life educational scenario. The participants were 20 undergraduate students who were absolute beginners in programming. The participants were divided into two groups of 10 students each. One group was taught basic Python programming through traditional methods in an educational setting consisting of lectures and lab work. The other group was instructed to learn programming basics using the proposed framework. The learning phase for both groups lasted two months. After the two-month period, the participants took a lab test which assessed their learning outcomes and programming performance. This post-intervention lab test enabled a comparison of the learning outcomes and skills attained between the two experimental groups after the two-month learning phase. Outcomes of this test provided data to examine the effectiveness of learning basic Python programming either through the traditional lecture and lab structure or through the proposed new framework. The implementation of this study required minimal computational resources. ChatGPT interactions were conducted using the OpenAI API, which can be run on standard consumer-grade hardware.

A lab test was chosen as the assessment method, as prior research has found it to accurately evaluate programming skills [5]. The lab test was 30 minutes in duration. For the empirical study, students in both groups used standard desktop computers equipped with Python 3.8 and PyCharm Community Edition as the integrated development environment (IDE). Students were required to implement, run, and debug their programs within the allotted time. The programs were submitted electronically. Students had access to standard Python documentation materials. The assigned programming task covered all basic Python fundamentals while keeping the scope focused in order to be completed within the time constraints.

Question

- Write a Python program with the following features:
  - Define a function that takes in a list of numbers as an argument.
  - Inside the function: a) Calculate the sum of all the numbers b) Calculate the average of the numbers c) Find the maximum and minimum values d) Return these statistics from the function
  - In the main program: a) Create a list with at least 5 integers b) Call the stats function, passing the list as an argument c) Print the results returned from the function - including: i) The sum of the numbers ii) The average iii) The minimum and maximum values
  - Handle any errors appropriately with try/except blocks

Figure 10 illustrates the responses provided by two students from two distinct groups. An examination of the answers makes it apparent that both codes yielded the anticipated outputs through different approaches. An analysis of the responses furthermore



```python
# Function to process list
def process_list(number_list):
  sum_val = sum(number_list)
  list_len = len(number_list)

  min_num = number_list[0]
  max_num = number_list[0]

  for n in number_list:
    if n < min_num:
      min_num = n
    elif n > max_num:
      max_num = n

  avg_val = sum_val / list_len

  return sum_val, avg_val, min_num, max_num

# Main
numbers = [15, 12, 36, 9, 28]
results = process_list(numbers)

print("Sum:", results[0])
print("Average:", results[1])
print("Minimum:", results[2])
print("Maximum:", results[3])
```

(a) Traditional Group

```python
# Define function
def calculate(nums):
  sum_val = 0

  for n in nums:
    sum_val += n

  avg_val = sum_val / len(nums)

  min_num = min(nums)
  max_num = max(nums)

  return sum_val, avg_val, min_num, max_num

# Main program
number_list = [19, 14, 28, 39, 16]
stats = calculate(number_list)

print(f"Sum: {stats[0]}")
print(f"Average: {stats[1]}")
print(f"Minimum: {stats[2]}")
print(f"Maximum: {stats[3]}")
```

(b) Framework Group

Figure 10: Sample answers from two students from two different groups

reveals that both students satisfied all features of the question with the exception of error handling. While the methodology differed marginally, the solutions proffered were largely similar in fulfilling the key objectives.

Each programming task on the lab test was designed to have a "perfect solution" that would demonstrate the student's competence in utilizing one or a combination of the Python concepts taught. Full credit was awarded for program submissions that reflected this perfect solution. Partial credit (half of the points allotted) was granted for submissions that produced the correct output but did not demonstrate the use of the expected Python concept(s). The grading process consisted of two stages [8]. First, an auto grader filtered out programs with runtime errors, compile-time errors, or incorrect output. Then, educators manually examined the remaining submissions that were executed successfully and produced accurate output. These submissions were scored based on the extent to which they matched the predefined perfect solution, with points awarded accordingly. Submissions reflecting the perfect solution were awarded a score of 3 points. Partially correct solutions demonstrating accurate output but not the expected technique was given 2 points. Incorrect programs that failed to compile, generate runtime errors, or produce the right output were assigned a score of 1 point. This 3-point scale was used to assess students' programming solutions for each lab test task. Then, we proceed to learning outcome comparison between the two groups.

As evidenced by Figure 11 and Figure 12, students in the two groups demonstrated comparable overall programming competence. In the traditional instruction group (participants 1-10), scores ranged from 1 to 3, with a mode score of 2. The mean score for this group was 2.3. Similarly, the framework group (participants 11-20) had scores spanning 1 to 3, with a modal score of 3. The average score in this group was 2.2.

Upon closer analysis, 50% of students in the traditional cohort scored a 3, reflecting the full implementation of the expected programming solution. Likewise, 50% of the framework group attained a perfect score of 3 on the assessment. 30% of the traditional group scored 2 points, indicating a partially correct solution, and 20% of framework students received 2 points. Additionally, while only 20% of the traditional group scored 1 point, indicating incorrect or failed solutions, 30% of framework students received 1 point. While some variances existed in the distribution of scores, students in both instructional approaches exhibited comparable programming aptitude overall as measured by the lab test. The two groups had very similar average scores, with the traditional approach having a slightly higher mean score of 2.3 versus 2.2 in the framework cohort. The number of students achieving perfect scores is the same in both experimental groups. Further study with larger sample sizes would be valuable to obtain additional insights into differences in programming outcomes between the two pedagogical methods. However, based on this preliminary quantitative analysis, both approaches demonstrated similar ability in strengthening novice programming skills.

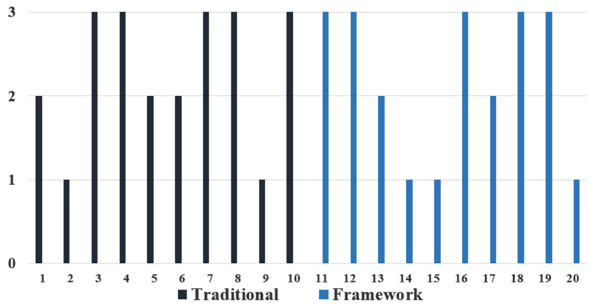


Figure 11: Evaluation of the coding task of the two groups

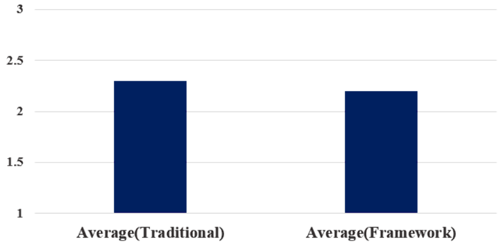


Figure 12: Average score of two groups

The proposed ChatGPT-based self-directed learning framework demonstrates significant promise in strengthening novice programming skills. A key advantage is enabling 24/7 access to an AI assistant that can provide customized explanations, interactive coding practice, and real-time feedback. Through this approach, newcomers to programming can learn core concepts and work hands-on by writing code with guidance from ChatGPT. As per the test results, the framework facilitates effective self-paced learning. Learners can tailor sessions based on their gaps and queries. This allows for covering fundamentals systematically while also attempting practical implementations. ChatGPT assists novices in not just understanding terminology or syntax but also in actually developing coding capabilities. Its ability to identify errors and suggest fixes is valuable for debugging. Additionally, the empirical study highlights that self-motivated novice can successfully acquire elementary programming competence via this framework. While traditional instruction provided more structured learning paths and immediate face-to-face clarification, the ChatGPT-based approach offered greater flexibility in learning pace and constant availability of assistance.

6 CONCLUSION

The study investigated the effectiveness of ChatGPT as a self-sufficient instructor for teaching fundamental programming concepts to novices, specifically focusing on Python. The findings suggest that ChatGPT can effectively explain basic concepts like variables, data types, and control statements, but struggles with more complex topics like functions, modules, and data structures.

The empirical study comparing traditional classroom instruction to the proposed ChatGPT-based framework showed similar programming outcomes between the two groups. However, the study had limitations, including a small sample size, single evaluation point, and focus only on technical skills. While ChatGPT shows promise as an initial learning tool, it needs improvement to progress learners to advanced proficiency levels without human tutoring. Incorporating human instruction, especially for complex topics, is still crucial.

This study has several limitations that should be addressed in future research. Firstly, the small sample size (20 students) limits the generalizability of the findings. A larger-scale study with more diverse participants would provide more robust results. Secondly, the study focused on a single programming language (Python) and basic concepts; future work should explore ChatGPT's effectiveness across multiple languages and more advanced topics. Thirdly, the evaluation was conducted at a single point in time, limiting our understanding of long-term learning outcomes. Longitudinal studies are needed to assess knowledge retention and skill development over time. Finally, while this study focused primarily on technical skills, future research should also consider soft skills development and student engagement factors when evaluating AI-assisted learning approaches. Additionally, long-term study is necessary to fully evaluate the relative strengths of each approach across various learning outcomes, including conceptual understanding, problem-solving skills, and long-term knowledge retention.

ACKNOWLEDGMENTS

Generative AI tools were utilized to generate sections of this Work, including text, tables, data, and also used for editing, formatting, and grammar correction in the paper.

REFERENCES

[1] Zishan Ahmed, Shakib Sadat Shanto, and Akinul Islam Jony. 2024. Potentiality of generative AI tools in higher education: Evaluating ChatGPT's viability as a teaching assistant for introductory programming courses. *STEM Education* 4, 3 (2024), 165–182.

[2] Zishan Ahmed, Shakib Sadat Shanto, Most Humayra Khanom Rime, Md Kishor Morol, Nafiz Fahad, Md Jakir Hossen, and Md Abdullah-Al-Jubair. 2024. The Generative AI Landscape in Education: Mapping the Terrain of Opportunities, Challenges and Student Perception. *IEEE Access* (2024).

[3] Som Biswas. 2023. Role of ChatGPT in Computer Programming.: ChatGPT in Computer Programming. *Mesopotamian Journal of Computer Science* 2023 (2023), 8–16.

[4] Ali Borji. 2023. A categorical archive of chatgpt failures. *arXiv preprint arXiv:2302.03494* (2023).

[5] AT Chamillard and Kim A Braun. 2000. Evaluating programming ability in an introductory computer science course. In *Proceedings of the thirty-first SIGCSE technical symposium on Computer science education*. 212–216.

[6] Bassey Isong. 2014. A Methodology for Teaching Computer Programming: first year students' perspective. *International journal of modern education and computer science* 6, 9 (2014), 15.

[7] Md Golam Jamil and Sakirulai Olufemi Isiaq. 2019. Teaching technology with technology: approaches to bridging learning and teaching gaps in simulation-based programming education. *International Journal of Educational Technology in Higher Education* 16, 1 (2019), 25.

[8] Theodora Koulouri, Stanislao Lauria, and Robert D Macredie. 2014. Teaching introductory programming: A quantitative evaluation of different approaches. *ACM Transactions on Computing Education (TOCE)* 14, 4 (2014), 1–28.

[9] Yuhan Lin and David Weintrop. 2021. The landscape of Block-based programming: Characteristics of block-based environments and how they support the transition to text-based programming. *Journal of Computer Languages* 67 (2021), 101075.

[10] Raymond Lister, Elizabeth S Adams, Sue Fitzgerald, William Fone, John Hamer, Morten Lindholm, Robert McCartney, Jan Erik Moström, Kate Sanders, Otto Seppälä, et al. 2004. A multi-national study of reading and tracing skills in novice programmers. *ACM SIGCSE Bulletin* 36, 4 (2004), 119–150.

[11] Valéria F Martins, Ilana de Almeida Souza Concilio, and Marcelo de Paiva Guimarães. 2018. Problem based learning associated to the development of games for programming teaching. *Computer Applications in Engineering Education* 26, 5 (2018), 1577–1589.

[12] Michael McCracken, Vicki Almstrum, Danny Diaz, Mark Guzdial, Dianne Hagan, Yifat Ben-David Kolikant, Cary Laxer, Lynda Thomas, Ian Utting, and Tadeusz Wilusz. 2001. A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. In *Working group reports from ITiCSE on Innovation and technology in computer science education*. 125–180.

[13] Rodrigo Pessoa Medeiros, Geber Lisboa Ramalho, and Taciana Pontual Falcão. 2018. A systematic literature review on teaching and learning introductory programming in higher education. *IEEE Transactions on Education* 62, 2 (2018), 77–90.

[14] Md Mostafizer Rahman and Yutaka Watanobe. 2023. ChatGPT for education and research: Opportunities, threats, and strategies. *Applied Sciences* 13, 9 (2023), 5783.

[15] Partha Pratim Ray. 2023. ChatGPT: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope. *Internet of Things and Cyber-Physical Systems* (2023).

[16] Daisuke Saito and Tsuneo Yamaura. 2013. A new approach to programming language education for beginners with top-down learning. In *Proceedings of 2013 IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE)*. IEEE, 752–755.

[17] Daisuke Saito and Tsuneo Yamaura. 2014. Applying the top-down approach to beginners in programming language education. In *2014 International Conference on Interactive Collaborative Learning (ICL)*. IEEE, 311–318.

[18] Wong Yoke Seng and Maizatul Hayati Mohamad Yatim. 2014. Computer game as learning and teaching tool for object oriented programming in higher education institution. *Procedia-Social and Behavioral Sciences* 123 (2014), 215–224.

[19] Shakib Sadat Shanto, Zishan Ahmed, and Akinul Islam Jony. [n.d.]. Enriching Learning Process with Generative AI: A Proposed Framework to Cultivate Critical Thinking in Higher Education using Chat GPT. *Tuijin Jishu/Journal of Propulsion Technology* 45, 1 ([n. d.]), 2024.

[20] Shakib Sadat Shanto, Zishan Ahmed, and Akinul Islam Jony. 2023. PAIGE: A generative AI-based framework for promoting assignment integrity in higher education. *STEM Education* 3, 4 (2023), 288–305.

[21] Agus Dedi Subagja, Abu Muna Almaududi Ausat, Ade Risna Sari, M Indre Wanof, and Suherlan Suherlan. 2023. Improving customer service quality in MSMEs through the use of ChatGPT. *Jurnal Minfo Polgan* 12, 1 (2023), 380–386.

[22] Gail M Sullivan and Anthony R Artino Jr. 2013. Analyzing and interpreting data from Likert-type scales. *Journal of graduate medical education* 5, 4 (2013), 541–542.

[23] Josh Tenenberg and Sally Fincher. 2005. Students designing software: a multi-national, multi-institutional study. *Informatics in Education* 4, 1 (2005), 143–162.

[24] Haoye Tian, Weiqi Lu, Tsz On Li, Xunzhu Tang, Shing-Chi Cheung, Jacques Klein, and Tegawendé F Bissyandé. 2023. Is ChatGPT the ultimate programming assistant–how far is it? *arXiv preprint arXiv:2304.11938* (2023).

[25] Jacqueline Whalley and Nadia Kasto. 2014. A qualitative think-aloud study of novice programmers' code writing strategies. In *Proceedings of the 2014 conference on Innovation & technology in computer science education*. 279–284.

[26] Huiping Wu and Shing-On Leung. 2017. Can Likert scales be treated as interval scales?—A simulation study. *Journal of social service research* 43, 4 (2017), 527–532.

[27] Ramazan Yilmaz and Fatma Gizem Karaoglan Yilmaz. 2023. The effect of generative artificial intelligence (AI)-based tool use on students' computational thinking skills, programming self-efficacy and motivation. *Computers and Education: Artificial Intelligence* 4 (2023), 100147.