

# Computational Thinking AI-Assisted Programming

University of Engineering and Technology  
VIETNAM NATIONAL UNIVERSITY HANOI

# Content

---

- ▶ The Role of Artificial Intelligence in Computational Thinking
- ▶ Introduction to AI Assistants Ecosystem
- ▶ Learn Python Programming with AI
- ▶ Common Pitfalls and Challenges of AI
- ▶ Responsible Use of AI

# The Role of Artificial Intelligence in Computational Thinking

# Programming Is More Than Just Typing Code!

- ▶ Before computers, how did people solve complex problems?

Through COMPUTATIONAL THINKING!

- ▶ This is the systematic skill of analyzing and solving problems.
- ▶ Programming is merely the tool to "tell" the computer how to implement that solution.



# Computational Thinking: 4 Core Skills

---

## 1 **Decomposition**

Breaking down a large problem into smaller, more manageable parts.

*Example: to draw a basic cat, it's easier to focus on one part at a time, instead of trying to draw the entire cat all at once.*

## 2 **Pattern Recognition**

Identifying similarities, trends, or regularities in data.

*Example: Noting that all cats share common characteristics: eyes, a tail, fur,...*

## 3 **Abstraction**

Focusing on essential information, ignoring irrelevant details.

*Example: Individual cat specific characteristics: brown fur, a long tail, yellow eyes,... are not important for this task.*

## 4 **Algorithms**

Designing step-by-step instructions, or the rules to follow to solve the problem.

*Example: The precise steps to draw a basic cat.*

# Decomposition: AI Assisting "Divide and Conquer"

---

- ▶ AI can help to identify the distinct components of a larger problem.
- ▶ Get a structured starting point without being overwhelmed.
- ▶ Example Scenario:
  - ▶ **Problem:** "Write a simple student management program."
  - ▶ **Prompt:** "I want to write a student management program. What are the key functions I should consider?"
  - ▶ **AI Response (likely):**
    - ▶ Add new students.
    - ▶ Display student list.
    - ▶ Search for students by ID.
    - ▶ Delete students.

# Pattern Recognition & Abstraction

---

## ▶ Pattern Recognition:

- ▶ Provide AI with two similar code snippets and ask:  
"How can I reuse code for these two cases?"
- ▶ AI might suggest creating a common function or class, helping spot underlying patterns.

## ▶ Abstraction:

- ▶ AI can simplify complex concepts through abstraction.
- ▶ Asking "Explain 'Object-Oriented Programming' using a car as an example"
- ▶ AI maps an abstract idea to a familiar analogy, helping learners grasp the core principles more intuitively.

# Algorithms: AI as Your Learning Compass, Not Your Pilot

---

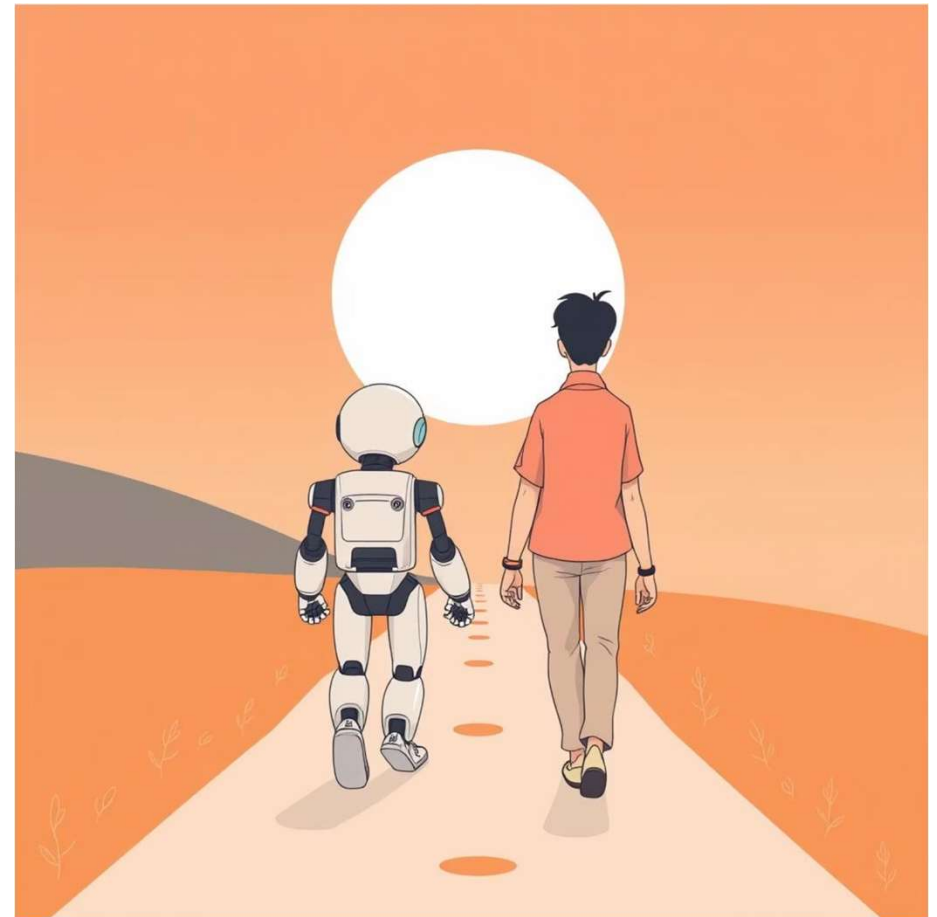
- ▶ AI can be an invaluable tool for mastering programming, but it's a double-edged sword.
- ▶ Ask AI: "What are common algorithms for sorting a list of numbers? Compare their pros and cons."
  - ▶ This encourages critical thinking about different approaches.
- ▶ Avoid asking: "Write code to sort this list for me." (At least while you're learning!)
  - ▶ This bypasses the crucial true understanding.
  - ▶ Learner would likely fail to evaluate a solution which is a must-have skill in real world scenarios.



# Where Does Ai Fit Into This Picture?

---

- ▶ **AI: Your Companion On The Thinking Journey**
  - ▶ AI doesn't replace your critical thinking.
  - ▶ It's a powerful tool that can assist you in all four steps of Computational Thinking.
  - ▶ Make your problem-solving process more efficient.
  - ▶ Our goal: Learn how to "command" AI



# AI: The Programmer's Powerful Assistant

---

- ▶ **Code Generation**
  - ▶ Generate code snippets or full functions from simple descriptions.
- ▶ **Code Suggestions**
  - ▶ Auto-complete lines of code as you type, predicting your next move.
- ▶ **Code Explanation**
  - ▶ Understand complex code sections with AI explanations.
- ▶ **Debugging**
  - ▶ Identify errors and suggest fixes, saving valuable time.
- ▶ **Documentation**
  - ▶ Automatically generate documentation for your functions and modules.
- ▶ **Learning Aid**
  - ▶ Ask for examples or explanations of new concepts, like "*Show me a for-loop example in Python.*"

# Introduction to AI Assistants Ecosystem

# ChatGPT: The General Coding Assistant

---

## ▶ How it Works:

- ▶ It's a question-and-answer format.
- ▶ You pose a prompt, and it generates a response.

## ▶ Strengths:

- ▶ Highly versatile, capable of answering almost anything.
- ▶ Ideal for brainstorming ideas and analyzing problems.

## ▶ When to Use It:

- ▶ When you need an explanation, an idea, or a comprehensive code sample.

# Quick Demo: From Idea to Code

---

## ► Quickly transform ideas to code

### The Prompt:

"Write a simple Python function called `tin_h_tong` that takes two numbers, `a` and `b`, and returns their sum."

This demonstrates AI's ability to quickly generate boilerplate code, freeing developers to focus on more complex logic and problem-solving.

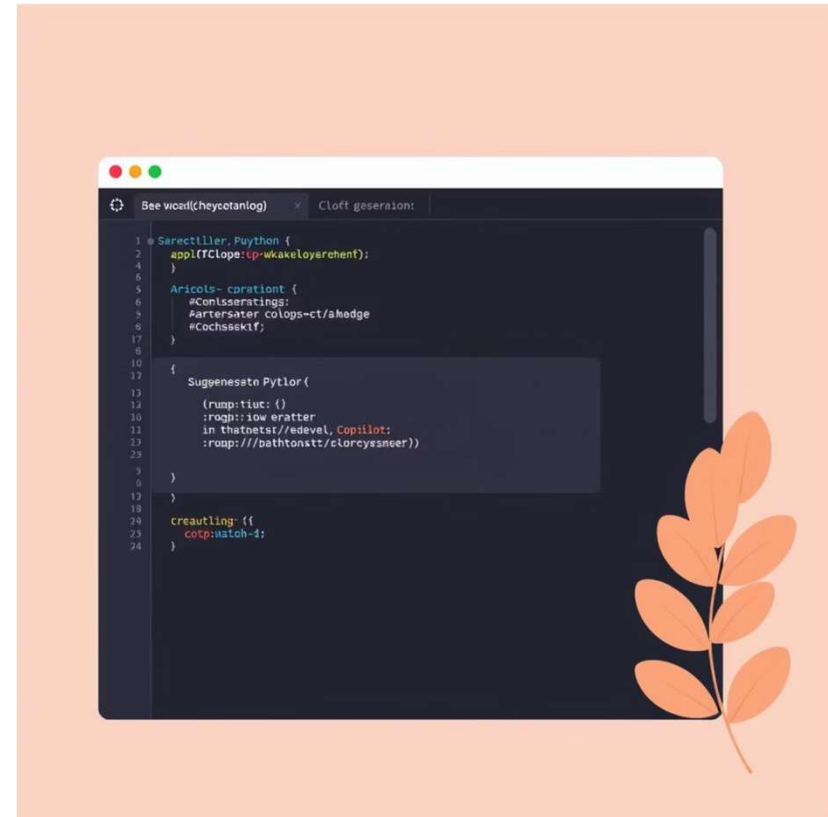
### AI Generated Result:

```
def tin_h_tong(a, b):  
    """ This function takes two numbers and returns their sum. """  
    return a + b
```

Notice how the AI not only generates the correct code but also includes a helpful docstring (description).

# GitHub Copilot: In-Editor Coding Assistant

- ▶ How it Works:
  - ▶ Integrates directly into code editor (VS Code,...).
  - ▶ Automatically suggests code as user type.
- ▶ Strengths:
  - ▶ Can be fast and convenient.
  - ▶ Understands the context of current project.
  - ▶ Significantly boosts coding speed.
- ▶ When to Use It:
  - ▶ When you know what you need to write and want to complete it faster.



# ChatGPT Vs. Copilot

---

Criterion	ChatGPT	GitHub Copilot
Format	Chat Interface	Integrated into Editor
Best For	Brainstorming, Explanations, Learning	Accelerating Code Writing
Strengths	Flexibility, Detailed Explanations	Speed, Contextual Awareness
Example	<i>"Explain data types in Python."</i>	Auto-completing a function as user type

- ▶ Use both! They complement each other.

# The AI Ecosystem for Programming

## ► Besides ChatGPT and Copilot:



<https://blog.scottlogic.com/2025/04/01/making-sense-of-the-ai-developer-tools-ecosystem.html>



# Learn Python Programming with AI

Leverage AI tools to accelerate learning and become a more effective programmer

# Demo: Decomposing a Problem with ChatGPT

---

## ▶ The Challenge:

- ▶ Write a small Python program to count the number of words in a sentence entered by the user.

## ▶ ChatGPT Prompt:

- ▶ `"I'm a new Python programming student. I want to write a program to count the words in a sentence. Help me break this problem down into smaller steps."`

## ▶ ChatGPT's Response (can be different on your machine):

- ▶ Step 1: Ask the user to input a sentence and store it in a variable. (Hint: use the `input()` function)
- ▶ Step 2: Split that sentence into a list of words. (Hint: use the `.split()` method)
- ▶ Step 3: Count the number of elements in the list you just created. (Hint: use the `len()` function)
- ▶ Step 4: Print the result to the screen."

# Demo: GitHub Copilot Auto-Completes Code

- ▶ Understanding the problem with ChatGPT's help, In VS Code, as you start typing, Copilot suggests completions.

Developer type:

```
# Ask the user to input a sentence and store it in a variable
```

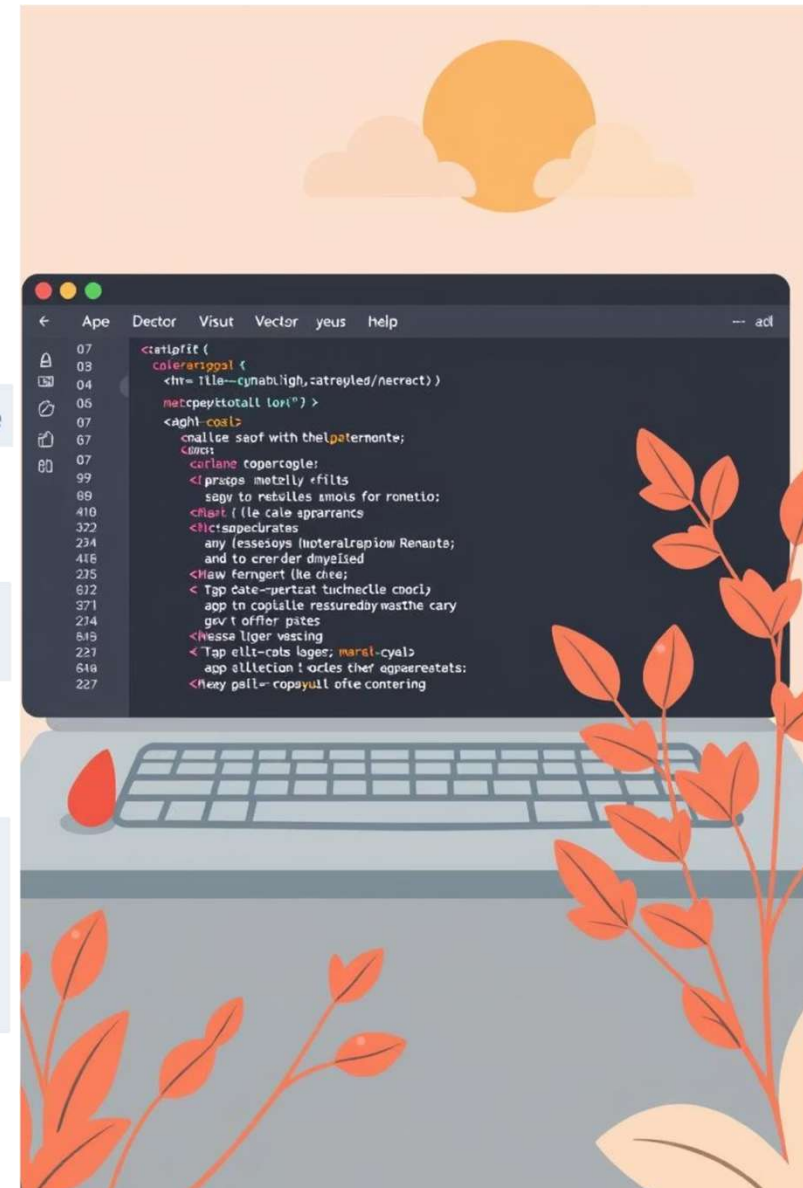
Copilot suggests:

```
# Ask the user to input a sentence and store it in a variable
user_sentence = input("Please enter a sentence: ")
```

Developer verifying the result:

```
# Ask the user to input a sentence and store it in a variable
user_sentence = input("Please enter a sentence: ")

# Print the sentence back to the user
print(f"The sentence you entered is: {user_sentence}")
```



# AI Helps Debug

---

## Scenario:

Suppose you wrote incorrect code and encountered an error:

```
Traceback (most recent call last):  File "main.py", line 4, in  
print("Number of words: " + count)TypeError: can only concatenate str  
(not "int") to str
```

## ChatGPT Prompt:

I encountered a `TypeError: can only concatenate str (not "int") to str` with this code. What does this error mean and how can I fix it?

# AI Explains Errors Like a Teacher

---

## ChatGPT's Potential Response:

"This error means you are trying to combine a string ("Number of words: ") with an integer (count). Python does not allow this directly.

**To fix it, you need to convert the `count` to a string before printing.**

- **Method 1 (using `str()`):** `print("Number of words: " + str(count))`
- **Method 2 (using f-string, more readable):** `print(f"Number of words: {count}")`

✓ **Remark:** AI not only fixes the problem but also explains the underlying issue and offers multiple solutions, enhancing your understanding.

# "Garbage In, Garbage Out"

---

## Effective Prompting Strategies



Role

"Act as a Python expert..."



Objective

"...explain the concept of for loops..."



Rules

"...using simple examples only."



Audience

"...for a programming beginner."

**Poor Prompt:**

"code word count"

**Good Prompt:**

"Write a Python function named `count_words` that takes a string as input and returns the number of words in that string."

# Prompt Engineering

---

- ▶ **Prompt patterns**
  - ▶ General prompting/Zero shot prompting
  - ▶ One-shot/Few-shot prompting
  - ▶ Role prompting
  - ▶ Chain-of-Thought (CoT) prompting
  - ▶ Code prompting
    - ▶ Prompts for writing code
    - ▶ Prompts for explaining code
    - ▶ Prompts for translating code
    - ▶ Prompts for debugging and reviewing code

....

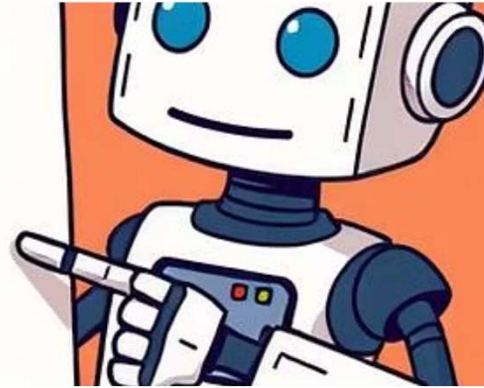
Google's prompting guide: <https://www.kaggle.com/whitepaper-prompt-engineering>

# Common Pitfalls and Challenges of AI



# When AI "Confidently" Lies

---



## Bias in Training Data

If the AI's training data contains biases (e.g., outdated or suboptimal code), the AI will learn and perpetuate these flaws, generating less-than-ideal code.

## Hallucination

AI can "invent" non-existent functions or concepts, like an `is_perfectly_even()` function in Python. New learners are particularly susceptible to believing these fabrications.

**Real-world example:** A lawyer once used ChatGPT for research and cited non-existent legal cases in court, leading to significant professional embarrassment.

# Don't Trust AI 100%! Understanding Hallucinations

---

- ▶ AI's hallucination: when the AI confidently generates incorrect or fabricated information. Examples:
  - ▶ Using a non-existent library or function.
  - ▶ Generating syntactically correct but logically flawed code.
  - ▶ Providing inaccurate explanations for concepts or errors.
- ▶ The key takeaway is critical: **always verify AI-generated content.**

**"AI is an assistant, not an absolute expert."**

# A Hard-Earned Lesson

---

## Real-World Example: When AI Goes Terribly Wrong


### The Scenario:

A programmer asked AI to write code to delete unnecessary files in a directory.

The AI, misunderstanding the context, generated a script that accidentally deleted critical system files.

### The Consequence:

The system crashed, leading to significant data loss.

 **The Lesson:** Never run AI-generated code - especially dangerous commands like file deletion or system modifications - without 100% understanding what it does.

# Responsible Use of AI

# Why Can't You Outsource Your Brain to AI?

- ▶ **AI can be wrong:** Only you understand the context and true requirements to fix its errors.
- ▶ **Creativity matters:** AI excels at imitation, but groundbreaking solutions stem from human ingenuity.
- ▶ **Debugging skills:** You learn most from fixing your own mistakes.
- ▶ **Foundational knowledge:** Without deep roots, you won't be able to build real world program.



# When to Use AI (and When Not To) in Programming?

---

## When to Use AI:

- ▶ Quickly generate example code.
- ▶ Learning new skills.
- ▶ Searching information from vast knowledge base.
- ▶ Explain coding errors/aid debugging.
- ▶ Automate repetitive coding tasks.

## When NOT to Use AI:

- ▶ Academic assignments (unless explicitly allowed).
- ▶ For sensitive or secure data tasks.
- ▶ When learning fundamental concepts. Wrestle with it first!

# AI: Collaborator or Boss?

---

## Passive Learner:

1. Copy problem statement.
2. Paste into ChatGPT.
3. Copy result.
4. Submit assignment.

❌ **Result:** 0 knowledge, 0 skills.

## Active Learner:

1. Analyze problem independently.
2. Ask AI for hints or guidance.
3. Write code yourself.
4. If errors occur, ask AI to understand and self-correct.

✅ **Result:** Deep understanding, skill development.

# Are You Over-Relying on AI?

---

1

## Blind Copy-Pasting

You copy and paste AI-generated code without understanding each line.

2

## Instant AI for Debugging

You ask AI for help immediately when encountering a small error, instead of thinking first.

3

## Inability to Explain Code

You can't explain the code AI just wrote for you.

4

## Knowledge Gap

Your grades are good, but you feel your fundamental knowledge is "empty".



If you check more than two boxes, you need to be cautious!



# 4 Golden Rules For Responsible Ai Use

---

## 1. Transparency

Always disclose when you use AI for assistance. Don't claim AI's work as your own.

## 2. Accountability

You are ultimately responsible for your work, not the AI.

## 3. Understanding

Always strive to understand what the AI generates before using it.

## 4. Academic Integrity

Adhere to your course and institutional regulations regarding AI usage.

# Always Verify the Results!

---

## DON'T BLINDLY TRUST

- ☐ Does the code run?
- ☐ Does the code run **correctly** for all cases? (e.g., sentence with multiple spaces, empty sentence...)
- ☐ Is there a better, more efficient way to write it?
- ☐ Do I truly understand every line of this code?

Always be the final reviewer.

# Citing AI In Academia

---

- ▶ Clear Attribution is Key
- ▶ When to cite?
  - ▶ When you use code, ideas, or explanations directly from AI in your assignments.
- ▶ How to cite?
  - ▶ This is a suggestion; always follow your instructor's guidelines.

```
// This code section was suggested/generated by OpenAI's ChatGPT on  
[date],// based on the prompt: "[Your exact prompt content]".// I have  
reviewed and modified it to fit the problem requirements.
```

- ▶ Honesty is always highly valued.

# First Steps In Programming

---

- ▶ **Focus on Fundamentals**
  - ▶ Computational thinking, data structures, and algorithms are your must-have knowledge.
- ▶ **Embrace Mistakes**
  - ▶ Errors are part of learning. Debugging your own code will embed concepts much more effectively than asking AI for solutions.
- ▶ **AI as a Study Supporter**
  - ▶ Discuss, debate, and learn from AI, but the final exam depends on your understanding, not its output.
- ▶ **Build Your Filter**
  - ▶ Learn to ask precise questions and critically evaluate AI's answers to determine their reliability and relevance.

# Key Takeaways

---

- ▶ Programming is about **thinking**; code is just a tool.
- ▶ AI is a powerful **assistant**, not a replacement.
- ▶ Be an **active** and **responsible** AI user.
- ▶ **Understanding, verification, and honesty** are key to success.
- ▶ A strong foundation will help you **ride the AI wave**, not be drowned by it.