AI PE Course (2025- 2026 Fall Term)

Week6:**AI Prompt Engineering for Code Generation**

MS TEAM Classroom code: ysliir3

3d Grade IT Students

Lecturer: Mohamamd Salim Al-Othman

Tishk International University

Tishk International University

Contents

- Introduction
- Types of Prompts
- Techniques
- Applications
- Safety and Limitaions
- Tools (5 AI Code Generators)
- Mini Lab + Exercises
- Summary

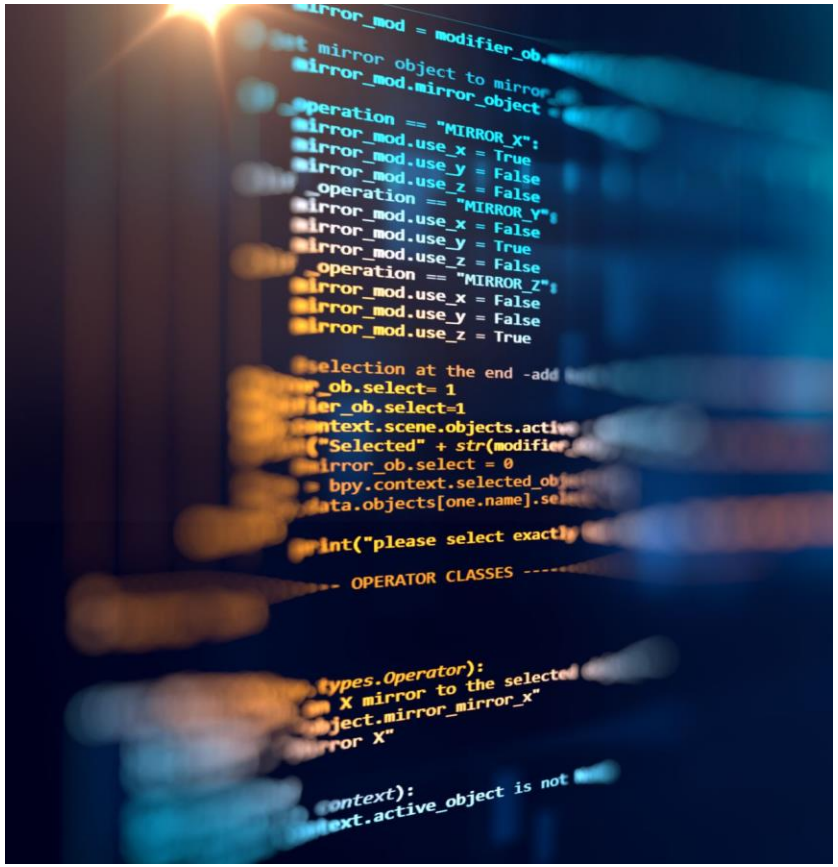
Common Challenges

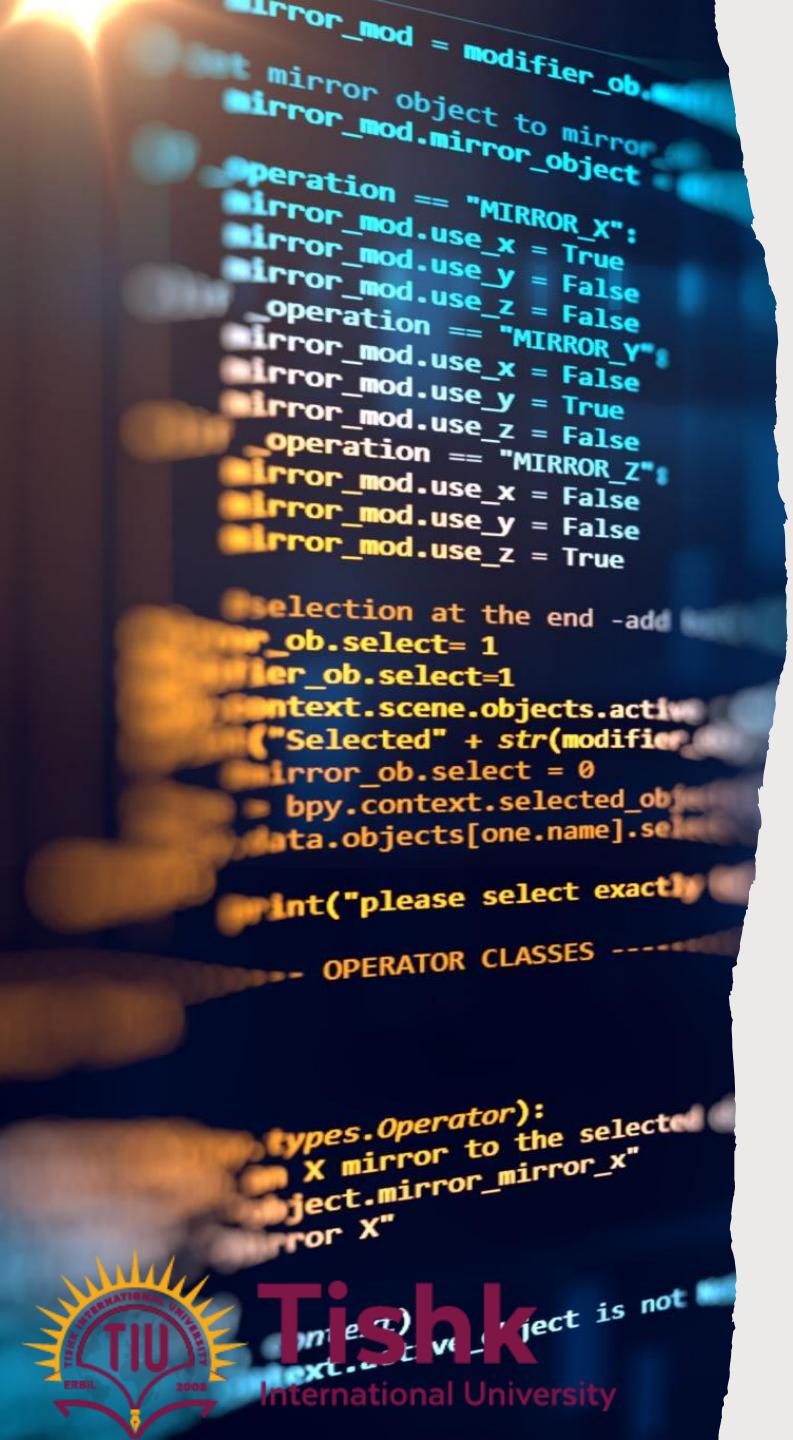
Ambiguous prompts lead to incorrect or incomplete code.

AI may generate syntactically correct but logically wrong code.

Complex tasks require multi-step guidance.

Debugging AI-generated code still requires developer expertise.





Types of prompts for code generation

1. Simple Prompt

Short description of a task.

Example: "Generate a Python function to calculate factorial."

2. Complex Prompt

Detailed specification including constraints, inputs, and expected outputs.

Example: "Write a Dart function that sums a list but only counts numbers > 10."

3. Multi-Stage Prompt

Break large tasks into sequential steps.

Example:
"Step 1: Generate a data model.
Step 2: Generate a REST API client.
Step 3: Generate Flutter UI for displaying the data."

4. Interactive Prompt

Iterative refinement through follow-up instructions.
User: "Add validation." AI: Updates code accordingly.

Prompt engineering **techniques** for **code generation**

1. Use Examples & Templates

- Provide patterns or code skeletons to guide the model.

2. Use Natural Language Descriptions

- Clear, direct instructions increase accuracy.

3. Use Structured Prompts

- Include: *context* → *instruction* → *constraints* → *format* → *examples.*

4. Use Specialized Code Tools

- Copilot, CodeWhisperer, Tabnine, etc., optimize the model's understanding of syntax and libraries.

Tishk International University

Applications Of Prompt Engineering For Code Generation

There are **four main applications** of prompt engineering for **code generation**:

1. Automated Code Generation

- Generate functions, classes, UI components, API services.

2. Code Repair & Refactoring

- Optimize for readability, performance, or best practices.

3. Code Synthesis

- Combine multiple snippets to produce complex code.

4. Code Explanation & Documentation

- Generate API docs or explain legacy code line by line.

Tishk International University

AI Code Assistants: Safety & Limitations

1. Never paste sensitive information

- Do **not** enter passwords, API keys, database credentials, student data, or internal project files.
- AI tools store prompts for model improvement, so sensitive data may become exposed.

2. AI-generated code must be reviewed

- Treat AI output as a *draft*, not final code.
- Always check logic, performance, security, and compatibility with your project.
- AI can produce code that compiles but contains hidden bugs.

3. AI may hallucinate functions, classes, or libraries

- Sometimes the model invents:
  - Non-existent methods
  - Fake APIs
  - Wrong library names
  - Unsupported Flutter/Dart syntax
- Students should verify output against official documentation.

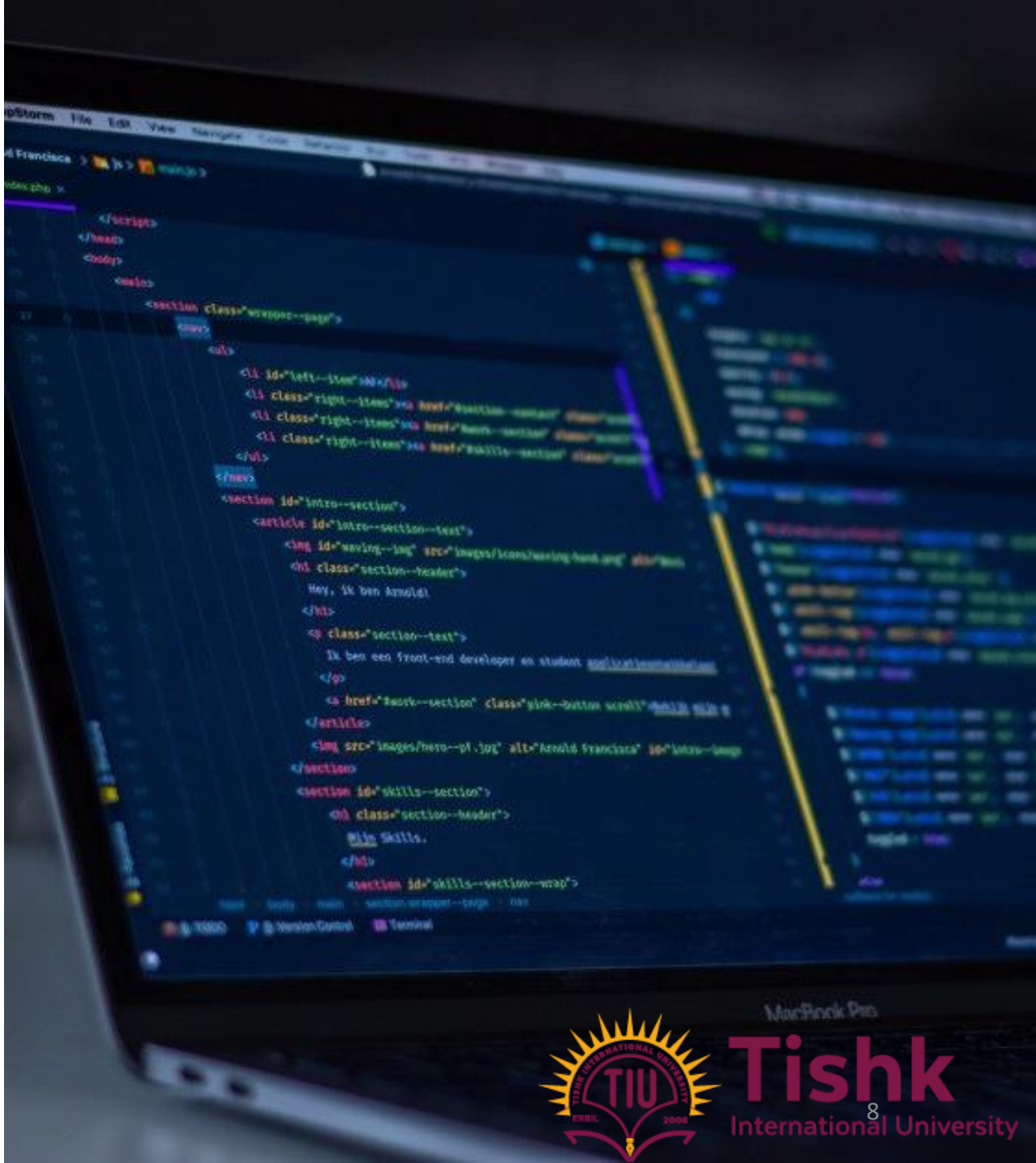
4. Human responsibility remains essential

- AI helps accelerate coding, but the developer is responsible for correctness, security, and maintainability.
- Use AI to support thinking, not replace it.

Tishk International University

5 Best AI Code Generators (October 2026)

AI code generators streamline coding, automate routine tasks, and suggest code snippets :)



Tishk International University

Top 5 AI Code Generators 2026

Industry leaders for developer productivity & code quality

RANK	TOOL	CATEGORY	KEY FEATURES	PRICING
1	⚡ GitHub Copilot	Productivity Leader	Real-time completions • Multi-IDE support • GPT-4 powered • Code workspace	$10-19/mo
2	✨ Cursor	AI-Native IDE	Full IDE environment • Chat + code • Web integration • VSCode familiar	Free + Premium
3	🧠 Claude Code	Agentic	Complex multi-step tasks • Deep reasoning • Full file editing • Artifact generation	API-based
4	CodeWhisperer	AWS Ecosystem	Security scanning • AWS optimization • License tracking • Vulnerability detection	Free
5	✔ Qodo	Quality-First	Multi-agent system • Test generation • Code review • PR analysis	Free + Enterprise

2026 Trends: AI code generators are evolving toward multi-agent systems with security-first architectures. The top performers combine real-time IDE integration, full SDLC coverage, local execution options, and privacy-by-design principles. Selection depends on your workflow: IDE-integrated tools (Copilot, Cursor) for daily coding; agentic tools (Claude Code, Qodo) for complex tasks; specialized tools (CodeWhisperer) for specific ecosystems.

2026 Trends in AI Code Generation & Software Development

6 key trends in AI-driven code generation & software development for 2026

Agentic AI Workflows

AI tools can autonomously plan → generate → test → refactor code

Domain-Specific Code Models

Models fine-tuned for specific domains (mobile apps, banking, security)

AI-Native Development Platforms (ANDPs)

Platforms integrating AI to speed up app design & deployment

Security, Governance & Ethical Coding

AI-generated code requires security review & provenance checks

Production-Scale Adoption

Organizations are deploying AI coding tools to full production workflows

Rise of New Developer Roles

Prompt engineers, LLM security auditors, hybrid AI-augmented developers

Tishk International University

Ai Lab Tools & Activities for This Week

1. AI Coding Assistants

- **Cursor AI**
  Hands-on testing of an AI-driven coding workspace with agentic features.
- **GitHub Copilot**
  Inline code suggestions, refactoring, and debugging support inside the IDE.

2. Google AI for Code Generation

- **Gemini via DartPad**
  You have already tested Gemini's code-generation capabilities using DartPad
  as part of your **Lab Task** for this course (Dart/Flutter focus).

3. Google AI Studio

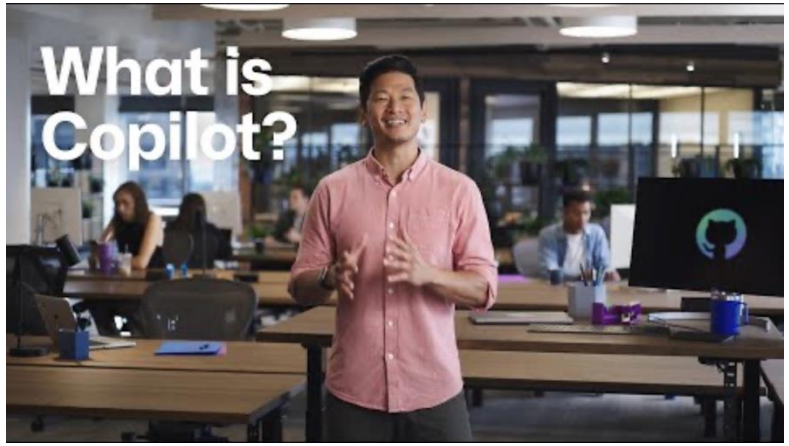
- Building prompts, testing models, and generating code in a controlled environment.
- Exploring model parameters and evaluating output quality.

4. Google AI Labs

- Experimenting with cutting-edge AI features and experimental tools.
- Understanding how AI agents perform planning and multi-step code generation.

Tishk International University

Demos ☺



Conclusion

- Clear prompts produce cleaner, more accurate code.
- Use structured prompts for complex tasks.
- AI helps generate code, but human review is essential.
- Multiple tools exist — choose based on context and constraints.
- Prompt engineering is now a core skill for modern developers.

Tishk International University

References

- **[1]** V. Corso, L. Mariani, D. Micciucci, and O. Riganelli, "Generating Java Methods: An Empirical Assessment of Four AI-Based Code Assistants," *arXiv preprint arXiv:2402.08431*, 2024.
- **[2]** J. H. Klemmer *et al.*, "Using AI Assistants in Software Development: A Qualitative Study on Security Practices and Concerns," *arXiv preprint arXiv:2405.06371*, 2024.
- **[3]** J. Li *et al.*, "An Exploratory Study on Fine-Tuning Large Language Models for Secure Code Generation," *arXiv preprint arXiv:2408.09078*, 2024.
- **[4]** S. Torka, "Optimizing AI-Assisted Code Generation," *arXiv preprint arXiv:2412.10953*, 2024.
- **[5]** A. Sergeyuk, "Using AI-based Coding Assistants in Practice," *Information and Software Technology*, vol. 171, 2025, doi: 10.1016/j.infsof.2024.107347.