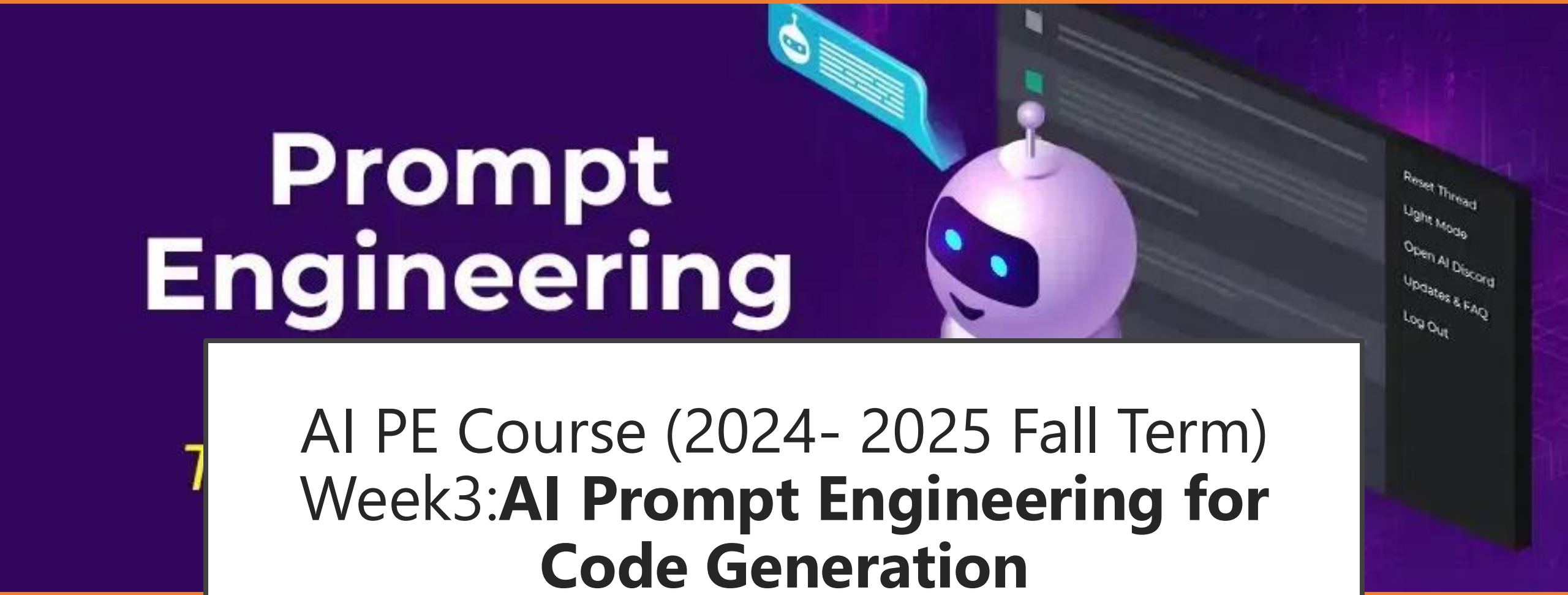


AI PE Course (2024- 2025 Fall Term)
Week3:**AI Prompt Engineering for Code Generation**

Google Classroom code: d5hsxq3

3d Grade IT Students

Lecturer: Mohamamd Salim Al-Othman

Tishk International University

Tishk International University

Contents

- Introduction
- Types of Prompts for Code Generation
- Prompt Engineering Techniques for Code generation
- Applications of Prompt Engineering for Code Generation
- Best 5 AI Code Generators Apps
- Conclusion

Introduction

What is prompt engineering?

Prompt engineering is the process of designing prompts that help large language models (**LLMs**) to generate the desired output. Prompts can be simple or complex, and they can be provided in a variety of formats, such as natural language descriptions, examples, or code templates.

Why is prompt engineering important for code generation?

Prompt engineering is important for code generation because **LLMs** need clear and concise prompts to generate **high-quality code**. Without effective prompt engineering, LLMs may generate code that is **incorrect**, **inefficient**, or **difficult to read and maintain.**

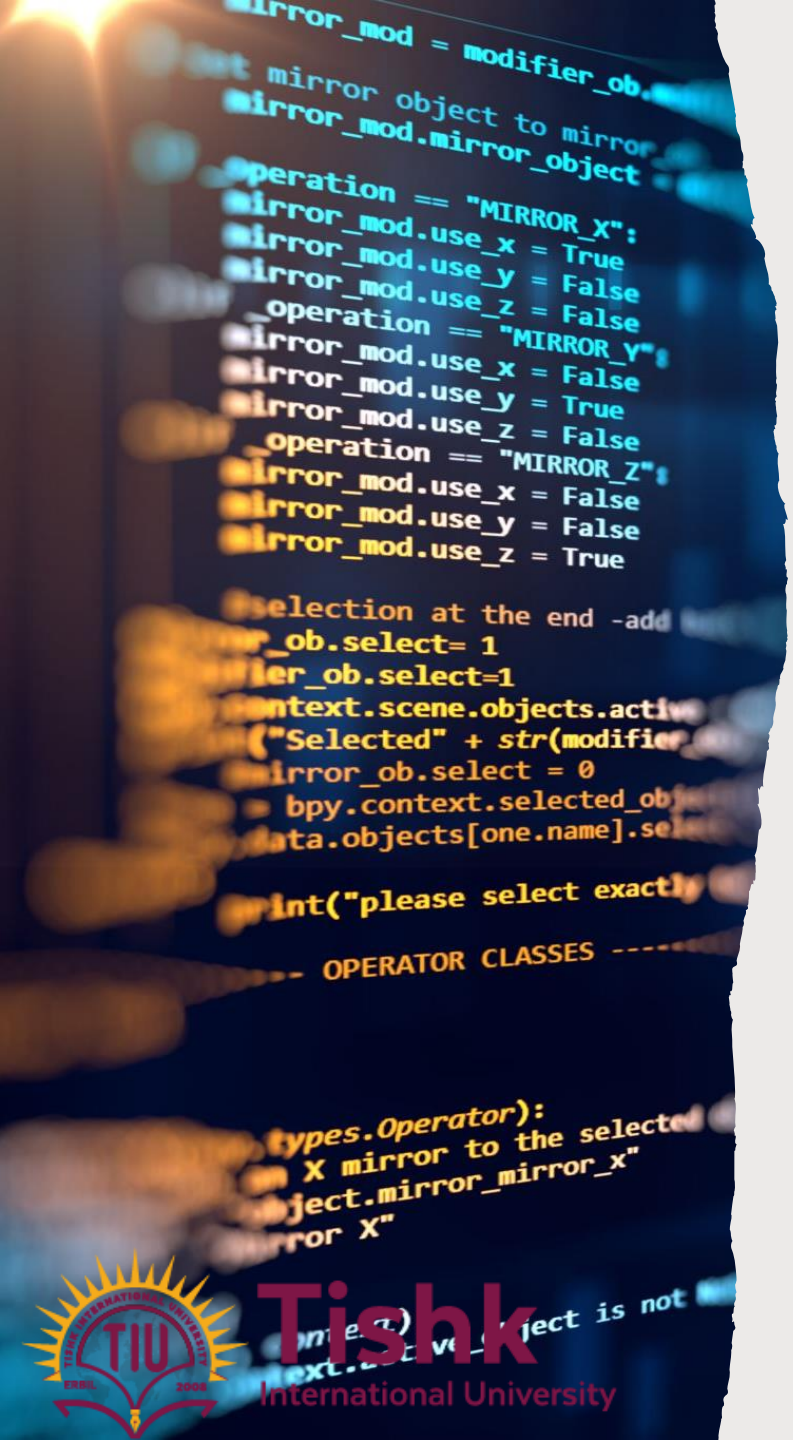
Benefits of using prompt engineering for code generation:

- **Increased productivity:** Prompt engineering can help developers to write code more quickly and efficiently by automating repetitive tasks and generating code from natural language descriptions.
- **Improved code quality:** Prompt engineering can help developers to write higher-quality code by reducing the risk of errors and ensuring that the generated code is consistent with the desired style and conventions.
- **Reduced costs:** Prompt engineering can help developers to reduce the cost of software development by reducing the need for *manual code generation and testing*.

Some of the common challenges faced in prompt engineering for code generation include:

- **Providing clear and concise prompts**: It can be difficult to provide LLMs with prompts that are clear, concise, and unambiguous. This is especially challenging for complex tasks.
- **Debugging generated code**: Generated code may not always be correct or efficient. It is important to carefully debug generated code before using it in *production*.

Tishk International University



Types of prompts for code generation

There are four main types of prompts for code generation:

- **Simple prompts:** Simple prompts provide a general description of the desired code. For example, the prompt "*Generate a function that takes a list of numbers and returns the sum of the numbers*" is a simple prompt.
- **Complex prompts**: Complex prompts provide more detailed information about the desired code, such as input and output formats, expected behavior, and constraints. For example, the prompt "*Generate a function that takes a list of numbers and returns the sum of the numbers, but only counts the numbers that are greater than 10*" is a complex prompt.
- **Multi-stage prompts:** Multi-stage prompts break down the code generation task into multiple smaller tasks. This can be helpful for complex tasks or tasks that require the generation of multiple code snippets.
- **Interactive prompts:** Interactive prompts allow the user to interact with the language model to refine the generated code. This can be helpful for tasks where it is difficult to provide a complete and unambiguous prompt in advance.

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

There are **four** main prompt engineering **techniques** for code generation:

- **Using examples and templates:** Providing the language model with examples of the desired code or templates that it can use to generate new code can help to improve the quality of the generated code.
- **Pre-training the language model on code:** Training the language model on a large dataset of code can help it to improve its understanding of code syntax and semantics, which can lead to better code generation results.
- **Using natural language descriptions:** Providing the language model with natural language descriptions of the desired code can allow it to generate code without the need for explicit code examples or templates.
- **Using code generation tools**: There are a number of specialized code generation tools available that can help users to create and refine prompts for code generation.

Tishk International University

Applications Of Prompt Engineering For Code Generation

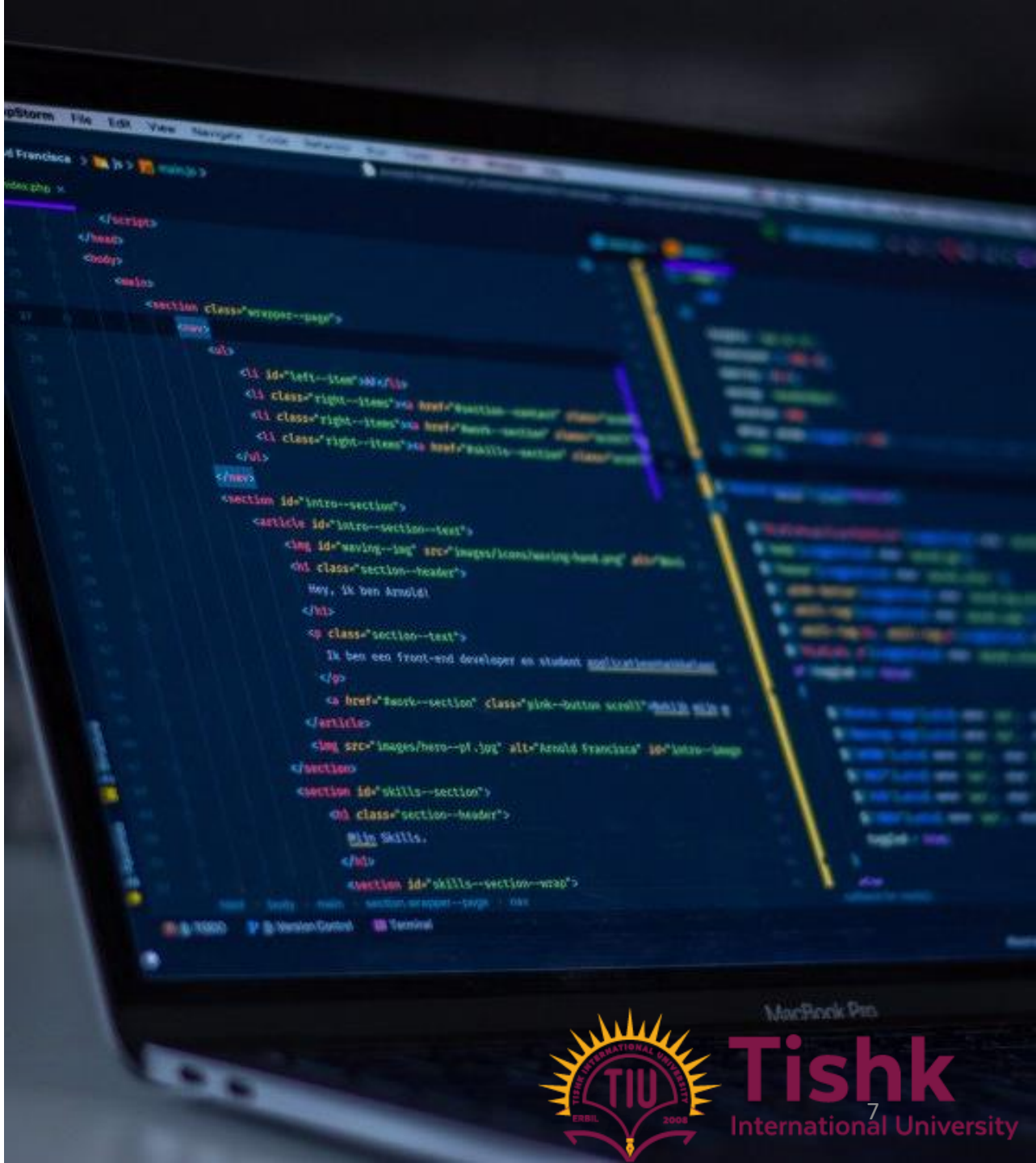
There are **four main applications** of prompt engineering for **code generation**:

- **Automated code generation:** Prompt engineering can be used to automate the generation of code from natural language descriptions or examples. This can be useful for tasks such as *generating boilerplate code or generating code for specific libraries or frameworks*.
- **Code repair and refactoring**: Prompt engineering can be used to *repair or refactor existing code*. This can be useful for tasks such as *fixing bugs, improving the efficiency of code, or making code more readable and maintainable*.
- **Code synthesis**: Prompt engineering can be used to *synthesize new code from multiple existing* code snippets. This can be useful for tasks such as *creating new libraries or frameworks, or generating code for complex tasks that are difficult to code manually.*
- **Code summarization and documentation**: Prompt engineering can be used to *summarize or document existing code*. This can be useful for tasks such as *generating API documentation or explaining how a piece of code works*.

Tishk International University

5 Best AI Code Generators (October 2023)

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



Tishk International University

1. GitHub Copilot

- Developed by GitHub in collaboration with OpenAI, GitHub Copilot represents the next level in AI-powered programming assistance.
- This tool functions like a **virtual pair programmer** that aids developers in writing better code at an expedited pace.

2. Replit GhostWriter

- Replit GhostWriter, as a product of Replit, is another impactful AI-based coding assistant designed to aid programmers in writing efficient and high-quality code.
- GhostWriter stands out for its ability to complete the code in real-time as the developer types, reducing the amount of time spent on writing boilerplate code and hunting down syntax errors.

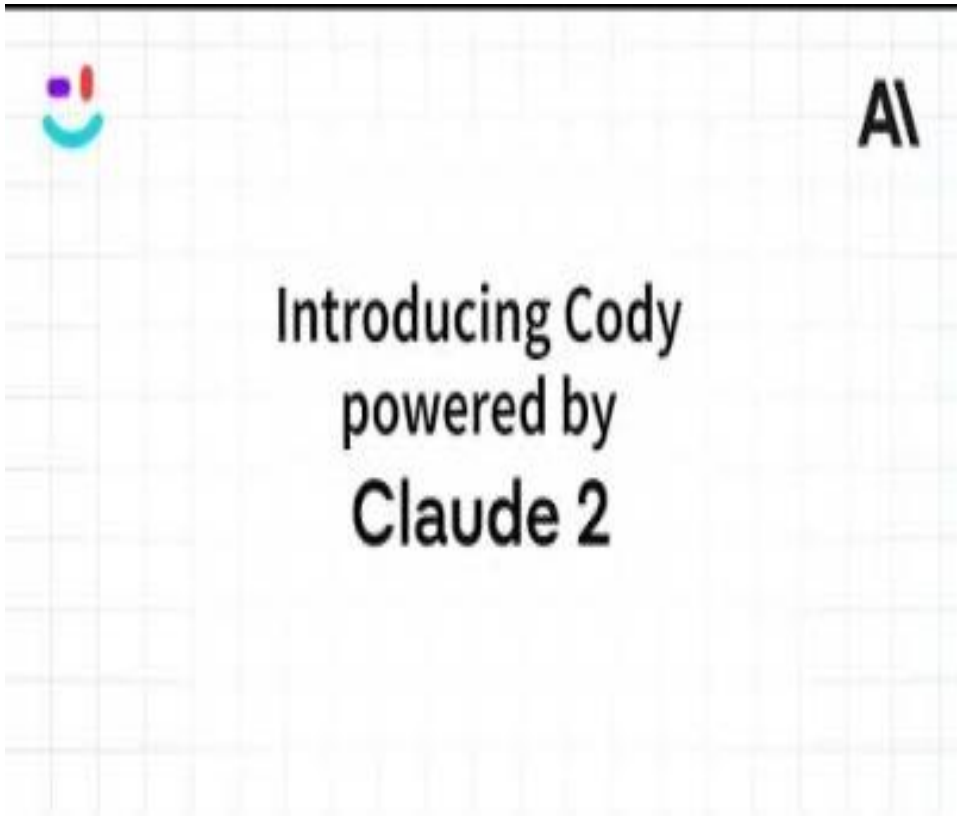
3. Amazon CodeWhisperer

- Amazon's CodeWhisperer revolutionizes the coding process by offering real-time suggestions ranging from snippets to entire functions, thanks to its vast knowledge from billions of lines of code.
- This facilitates smoother coding, even with unfamiliar APIs, and ensures code quality by highlighting suggestions sourced from open-source data, granting easy access to relevant project repositories and licenses.
- Furthermore, it prioritizes code security by pinpointing vulnerabilities, providing instant solutions, and ensuring alignment with esteemed security benchmarks such as those by OWASP.

4. Cody by Sourcegraph

- Cody is another AI-driven coding assistant, this one developed by Sourcegraph. The tool offers an impressive set of features that extend beyond the scope of code completion. Cody can be a boon to developers by providing automated code reviews and even identifying and fixing potential bugs in the code.

- Cody's main strength lies in its capability to understand the context in which the code is written, allowing it to provide meaningful and relevant suggestions and reviews. This can result in enhanced code quality and reduced debugging time, making the coding process more efficient.



Tishk International University

5. Tabnine

- Tabnine stands out as a powerful AI code assistant developed by Codota. The tool uses machine learning algorithms to predict and suggest code completions, aiming to make coding faster, more efficient, and less prone to errors.
- One of Tabnine's impressive features is its compatibility with **over 20** programming languages. This, along with its integration capabilities with various code editors, makes **TabNine** a versatile tool for developers across different platforms.
- Furthermore, its deep learning capabilities allow it to provide highly relevant code suggestions, making it a beneficial tool in any developer's toolkit.

Conclusion

- Prompt engineering is a powerful technique that can be used to improve the quality, productivity, and cost-effectiveness of code generation.
- By understanding the different types of prompts, prompt engineering techniques, and applications of prompt engineering for code generation, developers can use prompt engineering to generate better code, faster and cheaper.
- The role of AI in coding and software development is rapidly expanding. These AI-powered code generators are blazing the trail by providing powerful, intelligent, and intuitive tools to both seasoned developers and newcomers alike. They not only speed up the process of writing code but also make it more accessible to a broader audience, expanding the capabilities of individuals and organizations.
- From creating fully functioning eCommerce websites to converting audio commands into code, these AI-powered tools have opened up new opportunities and possibilities.
- Whether you're a seasoned developer seeking a smart assistant, or a beginner looking for a way to kickstart your coding journey, there's an AI code generator out there for you. Explore these options, and you might find a tool that significantly improves your coding efficiency and broadens your development horizons.

References

[1]: Cao, Yiwei, Pengcheng He, Yanyan Lan, and Xu Su. "Prompt Engineering for Code Generation: A Survey." In Proceedings of the 2023 International Conference on Learning Representations (ICLR), 2023.

[2]: Zhang, Qi, Tao Liu, and Wenhui Wang. "Prompt Engineering for Code Generation: A Comprehensive Guide." In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2023.

[3]: Zhang, Hongyu, Xiangyu Guo, and Wenhui Wang. "Prompt Engineering for Code Generation: A Tutorial." In Proceedings of the 2023 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA), 2023.

[4]: Zhang, Jian, Pengcheng He, and Xu Su. "Prompt Engineering for Code Generation: An Empirical Study." In Proceedings of the 2023 IEEE/ACM Conference on Automated Software Engineering (ASE), 2023.

[5]: Cao, Yiwei, Pengcheng He, Yanyan Lan, and Xu Su. "Prompt Engineering for Code Generation: A Practitioner's Guide." In Proceedings of the 2023 ACM SIGPLAN International Workshop on AI for Code Generation, Optimization, and Refactoring (AICGOR), 2023.

[6]: Unite.AI. "Best AI Code Generators." Unite.AI, 2023. https://www.unite.ai/best-ai-code-generators/.