AI Code Generation Overview

[(https://github.com/learn-co-curriculum/phase-2-ai-code-generation-overview)](https://github.com/learn-co-curriculum/phase-2-ai-code-generation-overview) [(https://github.com/learn-co-curriculum/phase-2-ai-code-generation-overview/issues/new)](https://github.com/learn-co-curriculum/phase-2-ai-code-generation-overview/issues/new)

Learning Goals

- Describe how AI code generation differs from code snippets.
- Explore how software engineers use these tools.

Introduction

As a software engineer, you'll have to write boilerplate code from time to time. Boilerplate code refers to repetitive code that is needed for running standard procedures. You may have already used code snippet extensions such as **ES6 Code Snippets (https://marketplace.visualstudio.com/items?itemName=xabikos.JavaScriptSnippets&ssr=false#review-details)** or the auto-complete feature in your code editor to generate boilerplate code.

In this lesson, we'll be looking at how AI supercharges code generation and in what contexts you'd want to use AI tools like **Tabnine (https://www.tabnine.com/)**, **Intellicode (https://visualstudio.microsoft.com/services/intellicode/)**, or **GitHub Copilot (https://github.com/features/copilot)**.

What Makes AI Code Generation Better?

Code snippet extensions are programmed to understand domain specific patterns and fill in boilerplate code. For example, if you're using a **React snippet extension (https://marketplace.visualstudio.com/items?itemName=dsznajder.es7-react-js-snippets)** it will have a **list of commands (https://github.com/ults-io/vscode-react-javascript-snippets/blob/HEAD/docs/Snippets.md)** to generate code. Typing in `rfc` would generate a standard function signature for a functional component.

Auto-complete features scan the project you're working in and provides suggestions based on data types, function definitions, and package dependencies. These completion features are usually limited to a few lines.

AI assistant tools use AI models that are trained on billions of lines of code from open source repositories, books, articles, and other publicly available sources. Languages that are better represented in these sources give better suggestions. Some tools like GitHub Copilot use the [OpenAI Codex model (https://openai.com/blog/openai-codex)](https://openai.com/blog/openai-codex) while others such as Tabnine use [their own models (https://www.tabnine.com/blog/announcing-tabnine-next-generation/)](https://www.tabnine.com/blog/announcing-tabnine-next-generation/). The models can connect with your project repo to gain more context and provide better suggestions.

All of this additional context allows these tools to offer some of the following features:

1. **Better Code Completion:** The AI tools can provide multi-line suggestions, function suggestions, and contextual suggestions if you provide access to private repos.
2. **Fast Test Generation:** AI assistants can also generate tests based on function signatures or doc strings. For complex tests, it's better to scaffold your own definitions first before using the AI suggestions.
3. **In-depth Code Analysis:** There are several types of analysis options provided by these tools. They can check for unsafe patterns, suggest common optimizations, and rewrite code to align with newer syntax.
4. ***************************************Language Translation:*** Migrations can be made easier by using AI assistants to translate between coding languages or even framework conventions.

Although these tools provide lots of great features, they aren't without their downsides. Let's look at some limitations of these tools and how to effectively use them while keeping these limitations in mind.

Limitations of AI Assisted Code Generation

As discussed previously, AI models can't think or reason. They use the data they've been trained on to make reasonable predictions about what to generate. This means that they're only as good as the data they've been trained on. AI models don't have access to private systems, lag behind on data collection (outdated APIs), uses answers from sites like StackOverflow which often don't have enough context, and doesn't have access to paid books. If you're working on specialized systems or are solving novel problems, AI assistants likely won't be able to generate useful code for you.

AI code generation works best for simple, pure functions where the inputs and outputs are clearly defined and there's no communication with external services. Software engineers primarily use AI assistants to take care of repetitive, boilerplate code. You can check out the following Hacker News threads to get a sense of how software engineers are using these tools:

- **Study finds AI assistants help developers produce code likely to be buggy (https://news.ycombinator.com/item?id=34137990)** .
- **How AI coding companions will change the way developers work (https://news.ycombinator.com/item?id=36019109)** .

And finally, larger corporations are banning access to external AI assistants due to security and privacy concerns. The laws and regulations are yet to catch up with the recent boom in AI services and systems so it's still unclear how exactly software engineers will be using AI for their work in the future. AI models that are used internally to develop systems or make decisions can be used within existing laws and regulations. These use cases are likely to be relevant for the foreseeable future even with upcoming policy changes.

Conclusion

In this lesson, we have learned AI code generation is different from regular code snippet generation and autocompletion tools. Although they provide some powerful features, their limitations necessitates exercising caution when using them in projects.

Resources

- **Developer sentiment around AI/ML (https://stackoverflow.co/labs/developer-sentiment-ai-ml/)** by StackOverflow.
- **StackOverflow 2023 Survey on AI (https://survey.stackoverflow.co/2023/#ai)** .
- Hacker News threads:
  - **Study finds AI assistants help developers produce code likely to be buggy (https://news.ycombinator.com/item?id=34137990)** .
  - **How AI coding companions will change the way developers work (https://news.ycombinator.com/item?id=36019109)** .
- **GitHub Copilot (https://github.com/features/copilot)** .
- **Tabnine (https://www.tabnine.com/)** .
- **Intellicode (https://visualstudio.microsoft.com/services/intellicode/)** .