Learning Objectives

Learners will be able to...

- **Understand the capabilities and limitations of ChatGPT in terms of coding**
- **Learn how to effectively structure user queries and system messages to obtain accurate code examples, explanations, and solutions to coding problems**
- **Develop skills in adapting AI-generated code to specific use cases**

info

Make Sure You Know

You are familiar with Python.

Limitations

This is a gentle introduction. So there is a little bit of Python programming. The information might not be the most up to date as OpenAI releases new features.

Capabilities

ChatGPT can be a powerful tool for generating and understanding code snippets, but it is essential to recognize its capabilities and limitations to maximize its potential in coding applications. We will explore how ChatGPT can assist with coding tasks, the extent of its language-specific understanding, and the importance of considering its limitations when using it as a coding resource.

ChatGPT Capabilities in Coding:

1. **Generating code snippets**: ChatGPT can generate code snippets in various programming languages, including Python, JavaScript, and Java, among others.
2. **Providing explanations**: The AI can offer explanations for specific code segments or concepts, helping users understand the underlying logic and functionality.
3. **Debugging assistance**: ChatGPT can help identify and fix common coding issues or suggest improvements to existing code.
4. **Best practices**: The AI can provide guidance on coding best practices, such as naming conventions, code structure, and design patterns.

Limitations of ChatGPT in Coding:

1. **Incomplete or incorrect code**: Generated code snippets might be incomplete or contain syntax errors, requiring user intervention for correction.
2. **Context awareness**: While ChatGPT has a general understanding of programming languages, it may not be aware of the latest language features, updates, or specific libraries and frameworks.
3. **Limited understanding of complex codebases**: ChatGPT may struggle to comprehend large or complex codebases, making it less effective in certain real-world scenarios.
4. **Varying quality of responses:** The quality and accuracy of generated code snippets or explanations may vary, depending on the clarity of the user query and the AI's understanding of the given context.

Understanding the capabilities and limitations of ChatGPT in generating and understanding code is crucial for effectively using it as a coding resource. By recognizing what the AI can and cannot do, users can better leverage ChatGPT for coding tasks, while being mindful of its limitations and potential inaccuracies.

Code Snippet

The ChatGPT API can be a valuable resource for generating code snippets in various programming languages, helping users streamline their development process. In this lesson, we will discuss techniques for structuring user queries and system messages to obtain accurate code examples and tips for getting the most out of the ChatGPT API for coding purposes.

Techniques for Generating Code Snippets:

1. **Be explicit in user queries**: When requesting code snippets, specify the programming language, desired functionality, and any relevant context or constraints. The clearer the query, the more accurate and useful the generated code snippet will be.
2. **Leverage system messages**: Use a system message to set the context of the AI as a coding assistant with expertise in the programming language you are working with.
3. **Provide code examples**: If possible, include examples of input and output data, or provide a code skeleton for the AI to work with. This helps the AI generate more accurate and relevant code snippets.

There are two files on the left side. `temp.py` will be what we use to interact with the API. The other file `test.py`, will be used to paste in the code generated by the API call.
Consider the following API call to generate a Python code snippet:

```
response=openai.ChatCompletion.create(
  model="gpt-3.5-turbo",
  messages=[
        {"role": "system", "content": "You are a coding
        assistant with expertise in Python."},
        {"role": "user", "content": "Write a Python function to
        find the factorial of a given number using recursion."}
    ]
)

print(response['choices'][0]['message']['content'].strip())
```

In this example, the user query explicitly mentions the programming language (Python), the desired functionality (factorial), and the technique (recursion). The system message sets the context of the AI as a coding

assistant with expertise in Python. Copy and paste the code being generated to the bottom left box. The button below is provided to help run it.

Always review the generated code snippet for correctness, completeness, and adherence to best practices. Part of the review process involves testing the generated code snippet in your development environment to ensure it works as expected and to identify any potential issues or errors. If the generated code snippet does not fully meet your requirements, adapt and refine the code as needed, or provide additional context and constraints in your user query.

Context awareness

Context awareness plays a critical role in generating meaningful and accurate responses using the ChatGPT API interactions. By providing context through user queries and system messages, you can guide the AI to generate responses that address user intent, requirements, and constraints effectively. Before moving further into code explanation, lets discuss context awareness.

Providing context in ChatGPT API interactions ensures that the AI understands the user's intent, background information, and any specific requirements or constraints. By supplying adequate context, you can obtain more accurate, relevant, and helpful responses from the AI.

Methods for Providing Context:
1. **User queries**: Formulate clear and explicit user queries that include relevant information, constraints, and requirements.
2. **System messages**: Use system messages to set the initial context, define the AI's role, or provide domain-specific guidance.

Lets try going over it using non-coding example to better help with understanding! Please pay specific attention to the roles and content of our queries. For the following code you are provided a `message` variable and paste the following into the message variable Consider the following ambiguous user query:

```
[  {"role": "system", "content": "You are a helpful
        assistant."},  {"role": "user", "content": "What's the
        best way to cook it?"}]
```

In comparison, Adding context through user query:

```
[  {"role": "system", "content": "You are a helpful
        assistant."},  {"role": "user", "content": "What's the
        best way to cook quinoa?"}]
```

In this example, by specifying "quinoa" in the user query, the AI can generate a more accurate response regarding cooking methods for that specific ingredient.

Setting context through system message:

```
[  {"role": "system", "content": "You are a helpful assistant
        with expertise in cooking."},
{"role": "user", "content": "What's the best way to cook
        quinoa?"}]
```

By adding a system message indicating expertise in cooking, the AI is guided to provide more detailed and accurate cooking instructions for quinoa.

Combining context from user query and system message:

```
[  {"role": "system", "content": "You are a helpful assistant
        with expertise in plant-based diets."},
{"role": "user", "content": "What's the best way to cook quinoa
        for a vegan meal?"}]
```

In this example, both the system message and user query provide context that ensures the AI generates a response tailored to vegan meal preparation.

Providing Explanations for Code

In addition to generating code snippets, ChatGPT can provide explanations for specific code segments or concepts, helping users understand the underlying logic and functionality. Employing effective techniques for requesting explanations and working with the provided information, you can improve your understanding of various programming languages, libraries, and frameworks.

As we get to the rest of this lesson, we will keep bringing up the idea of being specific, providing context using system message. However, as you will notice that being specific in one scenario might be different in another case.

> Techniques for Requesting Code Explanations:
>
> 1. **Be specific:** Clearly identify the code segment or concept for which you need an explanation.
> 2. **Provide context**: Include any relevant background information, constraints, or requirements that may affect the explanation.
> 3. **Use system messages**: Set the context of the AI as a coding assistant with expertise in the programming language or domain you are working with.

The following API is a call to obtain an explanation for a Python code snippet. Copy and paste the following as the value for the `message` variable.

```
[
{"role": "system", "content": "You are a coding assistant with
        expertise in Python."},
{"role": "user", "content": "Explain the following Python code:
        `def square(x): return x * x`"}
    ]
```

The user query explicitly identifies the code segment and requests an explanation. The system message sets the context of the AI as a coding assistant with expertise in Python.

Some general tips for Working with Code Explanations:

- Review the explanation: Ensure the provided explanation is accurate, complete, and easy to understand.
- Test the code: If the explanation involves code execution or modification, test the code in your development environment to verify its behavior.
- Request further clarification: If the explanation is unclear or incomplete, ask follow-up questions or provide additional context to obtain a more comprehensive understanding.

Debugging With ChatGPT

Using the ChatGPT API for debugging assistance can help users identify and fix coding issues, leading to more robust and efficient code. By employing effective techniques for requesting debugging help and working with the provided solutions, you can enhance your development process and improve the quality of your code.

Similarly, when generating code, asking for assistance or explanation we must add specificity, context and make use of system messages.

Techniques for Requesting Debugging Assistance:

1. **Be specific**: Clearly describe the problem, error message, or code segment you need help debugging.
2. **Provide context**: Include any relevant background information, constraints, requirements, or expected behavior that may affect the debugging process.
3. **Use system messages:** Set the context of the AI as a coding assistant with expertise in the programming language, library, or framework you are working with.

In the code below , the user query explicitly mentions the problem (`IndexError`) and provides the code segment in question. The system message sets the context of the AI as a coding assistant with expertise in Python.

```
{"role": "system", "content": "You are a coding assistant with
        expertise in Python."},
{"role": "user", "content": "I have this Python code that throws
        an IndexError: `def get_element(lst, index): return
        lst[index]`. Can you help me understand the issue and
        suggest a fix?"}
```

Based on the response generated. Always make sure to review the message generated, test the updated code and if necessary request further clarification. If for some reason the response generated is not correct or is not what is wanted. Provide further clarification in term of context and specificity. It might also be helpful to put in the whole error message if error was not solved the first time instead of `IndexError`.

The ChatGPT API can be a valuable resource for debugging assistance, helping users identify and fix common coding issues or suggest improvements to existing code.

Best Practices

Using the ChatGPT API for best practices and code optimization can help developers improve their coding skills and enhance the quality of their code. By employing effective techniques for requesting guidance and working with the provided suggestions, you can elevate your programming expertise and create more efficient, maintainable, and robust code.

Consider the following API call to obtain guidance on Python naming conventions:

```
[
  {"role": "system", "content": "You are a coding assistant with
        expertise in Python."},
        {"role": "user", "content": "What are some best
        practices for naming variables and functions in
        Python?"}
    ]
```

In this example, the user query explicitly requests guidance on Python naming conventions. The system message sets the context of the AI as a coding assistant with expertise in Python.

Now consider the following API call to optimize a JavaScript code snippet:

```
[
        {"role": "system", "content": "You are a coding
        assistant with expertise in JavaScript."},
        {"role": "user", "content": "I have this JavaScript
        code: `function add(a, b) { return a + b; }`. Can you
        suggest any improvements or optimizations?"}
    ]
```

In this example, the user query provides a JavaScript code snippet and requests suggestions for improvements or optimizations. The system message sets the context of the AI as a coding assistant with expertise in JavaScript.

Techniques for Requesting Best Practices and Optimization:

1. **Be specific**: Clearly describe the area of code or programming concept for which you need guidance on best practices.
2. **Provide context**: Include any relevant background information, constraints, or requirements that may affect the implementation of best practices or optimization.
3. **Use system messages:** Set the context of the AI as a coding assistant with expertise in the programming language or domain you are working with.

Coding Exercise

You have been tasked with using the ChatGPT API to generate a Python code snippet. The Python function you need should take in a string as an input and return the same string but reversed. Your task includes the following:

1. **Create the API Call**: Write a Python script that uses the ChatGPT API to request a Python function for reversing a string. Remember to follow the techniques for generating code snippets mentioned in the lesson. Make sure the user query explicitly mentions the programming language (Python), the desired functionality (reversing a string), and the system message sets the context of the AI as a coding assistant with expertise in Python. Additionally, write a print statement to help you visualize the generated code.

```
response = openai.ChatCompletion.create(
  model="gpt-3.5-turbo",
  messages=[
        {"role": "system", "content": ""},# make sure to replace
        empty string
        {"role": "user", "content":"" }# make sure to replace
        empty string
    ]
)
```

2. **Test the Generated Code**: Once you receive the Python function from the API, copy and paste it into a separate Python file (test.py). Run this script to ensure the function works as expected. Try with different inputs to verify the correctness of the function.

3. **Evaluate and Refine:** Critically assess the quality of the generated Python function. Does it meet the specified requirements? Does it follow good coding practices? If there are any issues, refine the code as necessary or adjust your API call to provide more explicit instructions or constraints.