Learning Objectives

Learners will be able to...

- **Understand the benefits of integrating ChatGPT API with development environments and tools**
- **Learn how to connect ChatGPT API with popular code editors**
- **Explore the integration of ChatGPT API with project management and collaboration tools**
- **Discover the potential of using ChatGPT API with Continuous Integration and Continuous Deployment (CI/CD) pipelines:**

---

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 data as OpenAI releases new features.

Integrating ChatGPT API

In today's fast-paced software development world, integrating powerful tools like the ChatGPT API with development environments and tools can significantly improve productivity, streamline workflows, and enhance collaboration. Some of those could be code editors like Sublime Text PyCharm. Some could be Project Management and Collaboration Tools like jira and github. Lets delve more into some of those benefits.

> Integrating the ChatGPT API with your development tools can offer numerous advantages, including:
>
> - Accessing AI-powered assistance directly within your preferred environment.
> - Accelerating problem-solving and brainstorming sessions.
> - Facilitating communication and collaboration within development teams.
> - Automating code reviews and generating test cases.
> - Tailoring the API to address unique requirements and challenges.

Developers often spend a significant portion of their time working within code editors like Visual Studio Code, Atom, or Sublime Text. By integrating the ChatGPT API into these editors, developers can access AI-powered assistance without leaving their preferred environment. The integration process typically involves:

1. Creating an API key from the OpenAI platform.
2. Installing an extension or plugin, or configuring the editor to connect with the ChatGPT API.
3. Setting up key bindings or commands to access the API conveniently.

Integrating ChatGPT API with Code Editors and IDEs

Let's tackle how to integrate the ChatGPT API with popular code editors and Integrated Development Environments (IDEs) to provide AI-powered assistance directly within your preferred coding environment. We will go over how to do it with 2 examples but the concepts can be applied to other environments as well.

For our first example, we can tackle the intergration using **Jupyter**.

Jupyter is an open-source web application that allows users to create and share documents containing live code, equations, visualizations, and narrative text. To integrate the ChatGPT API with Jupyter, follow these steps:

Ensure that you have the required Python libraries installed (openai and ipywidgets). If not, you can install them using pip. For that there is an additional terminal tab provided on the left box. Run the following command:

```
pip install openai ipywidgets
```

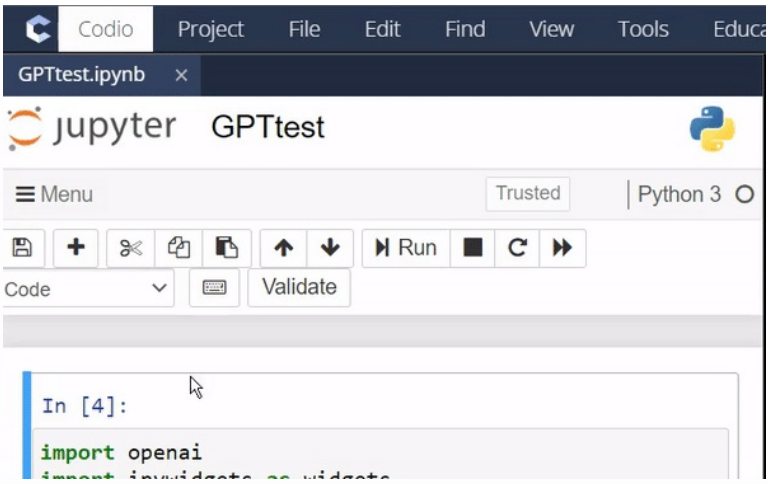
after running the following command you can import the necessary libraries in the Jupyter notebook.

```
import openai
import ipywidgets as widgets
from IPython.display import display
```

Next we can set our API key.

```
openai.api_key = "your_api_key_here"
```

To retrieve your api key you can go click on `Codio ->preferences -> environmental variable`.



codioTOkey

After retrieving your key you can copy and paste it as the value for the `openai.api_key`.

Now that we have all the tools needed we can start interacting with the API inside Jupyter.

Lets create a function to call the ChatGPT API:

```
def chatgpt_query(prompt):
    response = openai.Completion.create(
        engine="gpt-3.5-turbo",
        prompt=prompt,
        max_tokens=50,
        n=1,
        stop=None,
        temperature=0.5,
    )
    return response.choices[0].text.strip()
```

Create a simple interactive widget to interact with the ChatGPT API:

```
input_widget = widgets.Textarea(placeholder="Enter your question
        or prompt here...")
output_widget = widgets.Label()
submit_button = widgets.Button(description="Submit")

def on_submit(_):
    prompt = input_widget.value
    response = chatgpt_query(prompt)
    output_widget.value = response

submit_button.on_click(on_submit)

display(input_widget)
display(submit_button)
display(output_widget)
```

you can access ChatGPT API assistance within Jupyter by typing a question or prompt into the input widget and pressing Enter. The AI-generated response will appear below the input widget. we've added a submit_button widget. The on_click method is used to bind the on_submit function to the button. When you click the "Submit" button, the ChatGPT API query will be executed, and the response will be displayed in the output_widget.

Other IDES

The concepts beING applied to other environments.

Atom Integration:
Atom is another widely used open-source code editor with a strong community and a rich ecosystem of packages. To integrate the ChatGPT API with Atom, follow these steps:

1. Install the “chatgpt-api-client” package from the Atom package manager.
2. Configure the package by providing your ChatGPT API key.
3. Set up key bindings or commands to access the ChatGPT API from within the editor.
   After the integration is complete, you can access ChatGPT API assistance within Atom using the configured key bindings or commands.

Sublime Text Integration:
Sublime Text is a lightweight, fast, and customizable code editor popular among developers. To integrate the ChatGPT API with Sublime Text, follow these steps:

1. Install the “ChatGPT API Client” package using Package Control, the Sublime Text package manager.
2. Configure the package by providing your ChatGPT API key.
3. Set up key bindings or commands to access the ChatGPT API from within the editor.

ChatGPT API with project management and collaboration tools

We will explore how to integrate the ChatGPT API with GitHub to improve collaboration and streamline workflows. GitHub, a popular platform for version control and project management, can benefit from ChatGPT's capabilities to enhance various aspects of project management and development. This integration can provide intelligent suggestions for code, help with documentation, and assist in code review processes. In this tutorial, we will demonstrate how to use the ChatGPT API within a GitHub Actions workflow to automate some of these tasks.

Prrequisite

you should have a basic understanding of GitHub, Git, and GitHub Actions. You will also need an OpenAI API key to access the ChatGPT API.

First, let's create a GitHub Actions workflow. In your GitHub repository, create a `.github/workflows` directory and add a new YAML file called `chatgpt_integration.yml`. The contents of the file should be as follows:

```yaml
name: ChatGPT Integration

on:
  push:
    branches:
      - main
  pull_request:
    branches:
      - main

jobs:
  chatgpt_integration:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout code
      uses: actions/checkout@v2

    - name: Set up Python
      uses: actions/setup-python@v2
      with:
        python-version: 3.9

    - name: Install dependencies
      run: |
        python -m pip install --upgrade pip
        pip install openai

    - name: Run ChatGPT API integration
      run: python chatgpt_integration.py
      env:
        OPENAI_API_KEY: ${{ secrets.OPENAI_API_KEY }}
```

This workflow triggers whenever there is a push or pull request to the `main` branch. It sets up a Python environment, installs the required dependencies, and runs a script called `chatgpt_integration.py`, which we will create next.

ChatGPT API Integration Script

Create a new Python file in your repository called `chatgpt_integration.py`. This script will interact with the ChatGPT API and perform necessary actions based on the user's requirements. Here is a simple example:

```
import openai
import os

# Initialize the OpenAI API
openai.api_key = os.environ["OPENAI_API_KEY"]

def chatgpt_prompt(prompt):
    response = openai.ChatCompletion.create(
        model="gpt-3.5-turbo",
        messages=[
            {"role": "system", "content": "You are a helpful
assistant."},
            {"role": "user", "content": prompt},
        ],
    )

    assistant_message = [msg for msg in response['choices'][0]
['messages'] if msg['role'] == 'assistant']
    return assistant_message[0]['content']

# Example: Get a code suggestion from ChatGPT
code_suggestion_prompt = "Write a Python function to calculate
the factorial of a number using recursion."
code_suggestion = chatgpt_prompt(code_suggestion_prompt)
print("Code Suggestion:")
print(code_suggestion)
```

the chatgpt_prompt function sends a conversation with two messages to the ChatGPT API: one system message that sets the context (“You are a helpful assistant.”) and one user message containing the actual prompt. The response is then filtered to extract the assistant’s message and return its content.

This approach allows for more interactive and context-aware responses from the ChatGPT API. You can easily extend the conversation by adding more messages to the messages parameter, which can help you obtain more detailed or context-specific responses for your tasks.

You can extend this integration to cover other aspects of project management and development, such as generating code snippets based on issue descriptions, assisting in the code review process by providing suggestions and feedback, or helping with documentation by generating docstrings or markdown files.

With the combination of GitHub and ChatGPT API, the possibilities are endless, enabling teams to collaborate more effectively, streamline workflows, and improve the overall quality of their software projects.