# 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. This assignment will require a little bit of git knowledge. 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, Jupyter Notebook and 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 jupyter the concepts can be applied to other environments as well.

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

Before we get started, if for some reason Jupyter is not open on the left panel please refresh your page.

**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:

Before we get started, if for some reason Jupyter is not open on the left panel please refresh your page.

Ensure that you have the required Python libraries installed (openai and ipywidgets). If not, you can install them using pip. Run the following command in Jupyter:

```
!pip3 install openai
!pip3 install --upgrade openai
!pip3 install openai ipywidgets
```

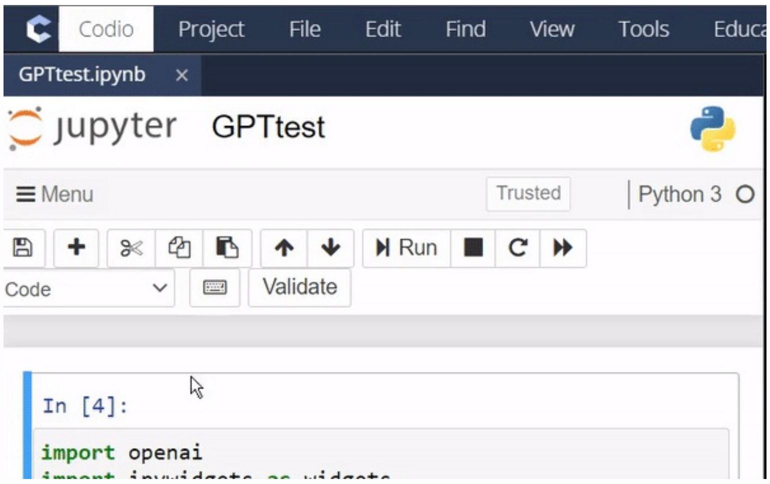
after running the following command you can import the necessary libraries in the Jupyter notebook.

```
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`.



```
In [4]:
import openai
```

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(prompts):
    # example of generating a completion
    response = openai.Completion.create(
      engine="text-davinci-002",
      prompt=prompts,
      max_tokens=60
)
    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.

**Step 1**: Create a New Repository
Create a new directory on your local machine and initialize it as a Git repository:

Use the terminal for the following:

```
mkdir my_project
cd my_project
git init
```

**Step 2**: Create the GitHub Actions Workflow
1. In your local Git repository, create a directory for GitHub Actions:

```
mkdir -p .github/workflows
```

2. Create a new YAML file for the ChatGPT integration workflow:

```
touch .github/workflows/chatgpt_integration.yml
```

For this page as you run the `touch` command to run your code, it will create a new file. You can access those file by clicking on the file button ->open file then you type in the file name you want to open. In the following case it should be `chatgpt_integration.yml`.

Once Open paste the following inside the text area:

```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_KEY: "${{ secrets.OPENAI_KEY }}"
```

**Step 3**: Create the ChatGPT API Integration Script
1. In your local Git repository, create a new Python file:

```
touch chatgpt_integration.py
```

1. Open the `chatgpt_integration.py` file and add the following content:

```
import openai
import os

# Initialize the OpenAI API
openai.api_key = os.getenv('OPENAI_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},
        ],
    )
    return (response['choices'][0]['message']
        ['content'].strip())
    #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)
```

**Step 4:** Commit Changes and Push to GitHub
1. Stage the files for commit:

```
git add .
```

2.Commit changes

```
git commit -m "Initial commit with ChatGPT integration."
```

3. Create a new repository on GitHub. You can do this by visiting https://github.com/new and following the instructions. Do not initialize the new repository with a README, .gitignore, or License. This empty repository will receive the contents of your local repository.

4. Once your new GitHub repository is created, you'll be shown a repository URL. It will look like this: https://github.com//.git. Copy this URL, replace and with your GitHub username and repository name, and run the following commands:

```
git remote add origin git@github.com:<username>/<repository>.git
git branch -M main
git push -u origin main
```

You now have a new GitHub repository with a GitHub Actions workflow that integrates with the ChatGPT API! Please remember to add your OpenAI API key to the GitHub repository secrets. You can do this by going to your GitHub repository, clicking on **“Settings” -> “Secrets” -> “New repository secret”**, then adding “OPENAI_KEY” as the name and your actual OpenAI API key as the value.

Please note that to use SSH with GitHub, you must have an SSH key set up on your GitHub account. You can find out how to do this in the GitHub Docs here.

# Coding Exercise

### Build a Basic Code Review Assistant with the ChatGPT API

Create a function named `chatgpt_code_review` with the following specifications:

- The function should take one argument: a file path.
- This function should use the ChatGPT API to review the code in the file.
- The function should return a string of feedback on the code.

Here is some python that can help with reading a file outline of the function:

```python
# Path to your python file
file_path = code

with open(file_path, 'r') as file:
code = file.read()
```

A `temp.py` file is in the same directory as you `exerc.py` file. Feel free to use it to test your code.

```python
print(chatgpt_code_review("temp.py"))
```

**Before submitting, your code remove your print statement.** The submit button will be running your code, we don't want to run it twice which might cause delays and lead to a `timeout error`.