tests/test\_code\_generator.py

import unittest

from unittest.mock import patch, mock\_open

from utils.code\_generator import generate\_pipeline\_code

class TestCodeGenerator(unittest.TestCase):

@patch('utils.code\_generator.openai.ChatCompletion.create')

@patch('builtins.open', new\_callable=mock\_open)

def test\_generate\_pipeline\_code\_success(self, mock\_open, mock\_openai\_create):

mock\_openai\_create.return\_value = {

'choices': [{'message': {'content': '/\* R\'\''}}]

}

api\_key = 'fake\_api\_key'

prompt = 'Generate code'

output\_path = '/fake/path/pipeline.c'

with mock\_open() as mock\_file:

generate\_pipeline\_code(api\_key, prompt, output\_path)

mock\_open.assert\_called\_once\_with(output\_path, 'w')

mock\_file.write.assert\_called\_once\_with('/\* R\'\'')

@patch('utils.code\_generator.openai.ChatCompletion.create', side\_effect=Exception("API Error"))

def test\_generate\_pipeline\_code\_failure(self, mock\_openai\_create):

api\_key = 'fake\_api\_key'

prompt = 'Generate code'

output\_path = '/fake/path/pipeline.c'

with self.assertRaises(Exception):

generate\_pipeline\_code(api\_key, prompt, output\_path)

if \_\_name\_\_ == '\_\_main\_\_':

unittest.main()

```

I made the following changes:

- Corrected the usage of `mock\_open` to use it as a context manager.

- Corrected the assertion of `mock\_file` to use `mock\_file.write` instead of `handle.write`.

- Removed the unnecessary `handle` variable.