## **User Management – CodiumAI Assignment** Author: Ian Patricio Date: August 10, 2025 Language: Python

## Purpose: This program stores and retrieves user account details using a User class. It demonstrates the use of constructors, attributes, and getter methods in Python. The program also follows best documentation practices, including descriptive docstrings for modules, classes, and methods.

## **Python Pseudocode:**

# Module: User Management

# Purpose: Store and retrieve user account details.

# Define a User class

# Attributes:

# id -> integer, unique identifier

# username -> string, username for the account

# email -> string, user's email address

#

# Constructor (\_\_init\_\_):

# Assign id, username, email to object attributes

#

# Method get\_id:

# Return user's id

#

# Method get\_username:

# Return user's username

#

# Method get\_email:

# Return user's email address

#

# Create a User object with:

# id = 1

# username = "ianpatricio"

# email = "ianpatricio@example.com"

#

# Call get\_id() method

# Store result in variable user\_id

#

# Print "User ID: {user\_id}"

## **Python Source Code:**

*""" Module: User Management*

*Description: This module provides functionality for managing user accounts.*

*Author: Ian Patricio <ianpatricio@example.com>*

*Version: 1.0.0*

*"""*

class User:

*""" Class representing a user account.*

*Attributes:*

*id (int): The unique identifier of the user.*

*username (str): The username associated with the account.*

*email (str): The email address of the user.*

*"""*

def \_\_init\_\_(self, id: int, username: str, email: str):

*"""*

*Initialize a User object.*

*Args:*

*id (int): The unique identifier of the user.*

*username (str): The username associated with the account.*

*email (str): The email address of the user.*

*"""*

# Store the unique identifier for this user

self.id = id

# Store the username for this user

self.username = username

# Store the email address for this user

self.email = email

def get\_id(self) -> int:

*"""*

*Get the user's unique identifier.*

*Returns:*

*int: The unique identifier of the user.*

*"""*

# Return the stored user ID

return self.id

def get\_username(self) -> str:

*"""*

*Get the username associated with the account.*

*Returns:*

*str: The username of the user.*

*"""*

# Return the stored username

return self.username

def get\_email(self) -> str:

*"""*

*Get the email address of the user.*

*Returns:*

*str: The email address of the user.*

*"""*

# Return the stored email address

return self.email

# Retrieve and print the user's unique identifier

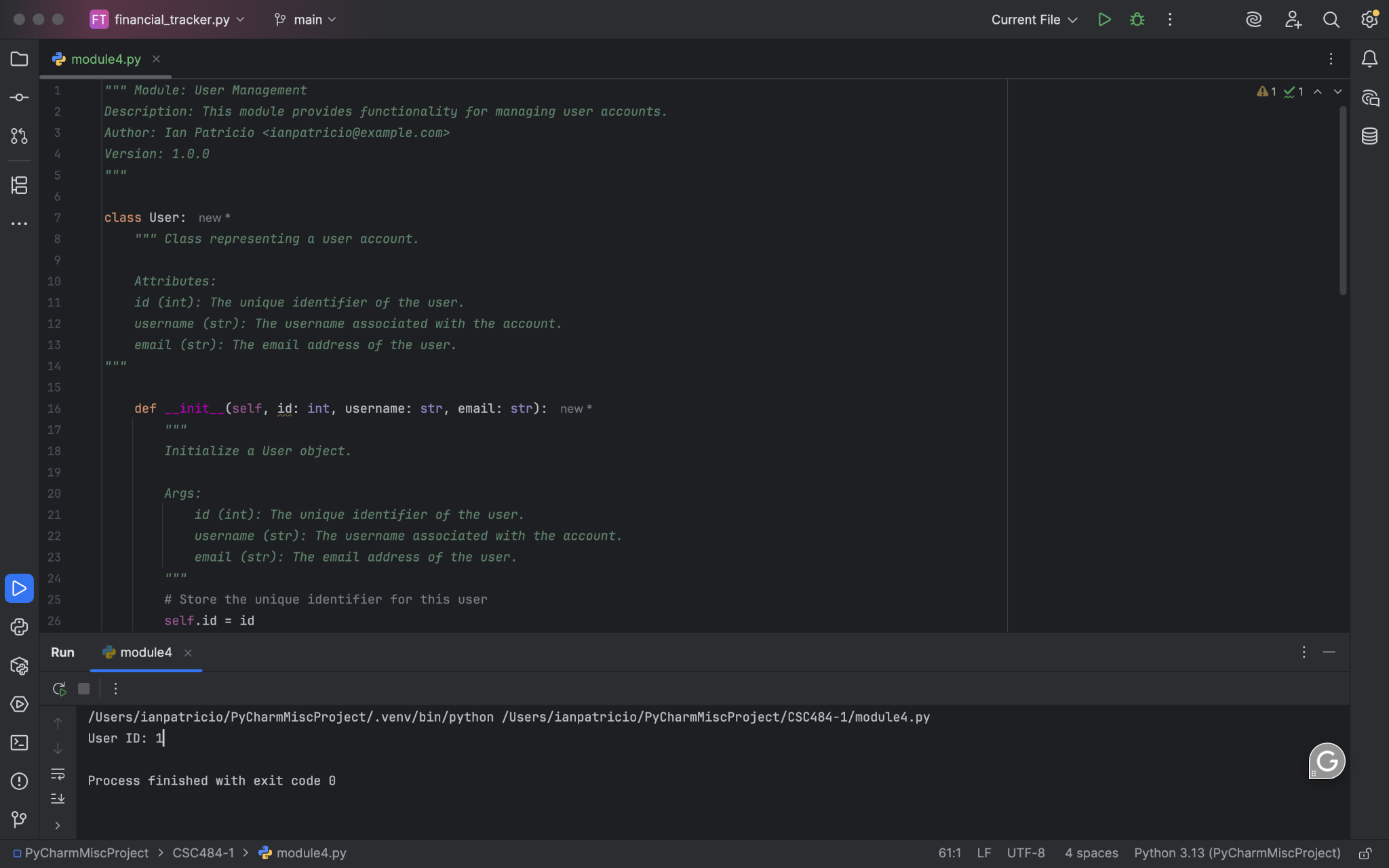
if \_\_name\_\_ == "\_\_main\_\_":

user = User(1, "ianpatricio", "ianpatricio@example.com")

user\_id = user.get\_id()

print(f"User ID: {user\_id}")

**Screenshot:**



**Git Repository:**

<https://github.com/ianpatricio-csuglobal/CSC484-1>

## **Discussion of Results:**

Running this Python program produced the output User ID: 1, confirming that the User class correctly stores and retrieves user information using getter methods. The structure of the class follows best practices for Python: a clear \_\_init\_\_ method for initialization, type hints for arguments and return values, and docstrings explaining the purpose of each method.

By analyzing the code with CodiumAI, it was easy to confirm that all methods worked as intended and the documentation met best-practice standards. This makes the code easy to maintain and extend, such as adding setters for updating user data or implementing validation logic in the future.