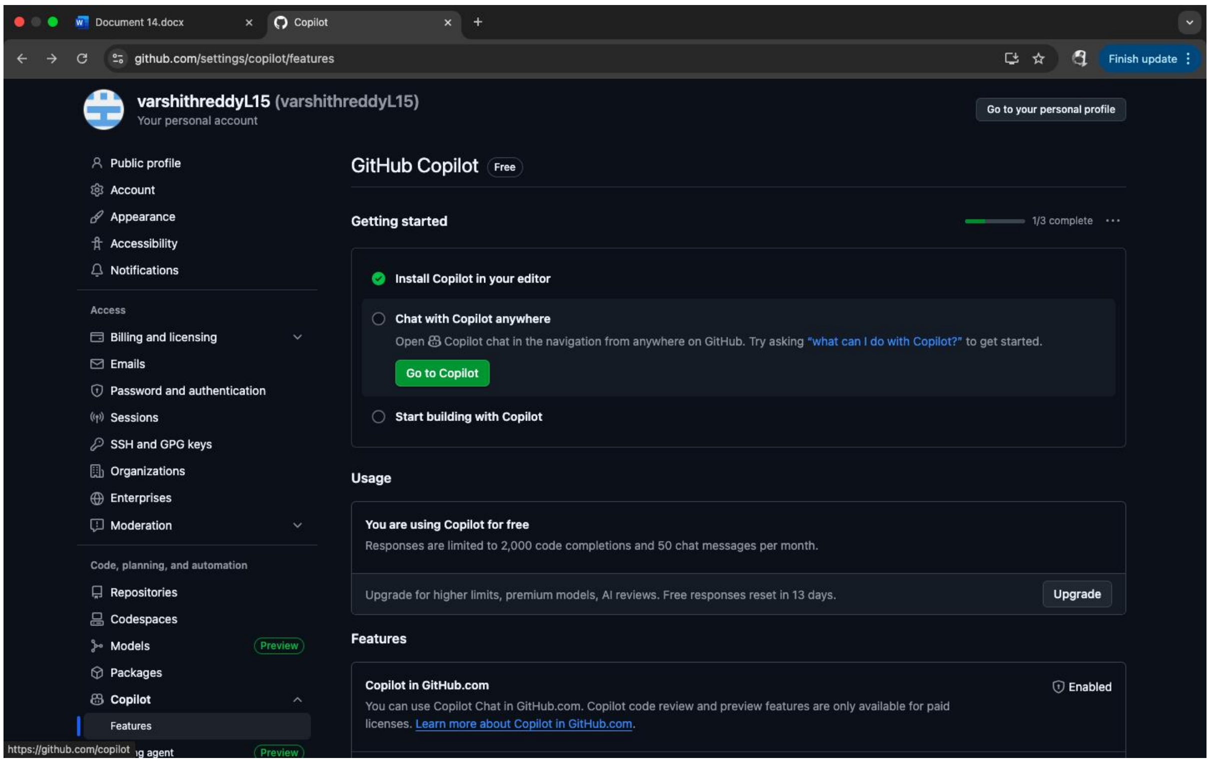
Htno: 2503A52L15

Varshith reddy .G

Objective: To explore AI-powered code assistants for writing Python classes, constructors, and methods through intelligent suggestions.

Suppose that you are hired as an intern at a tech company that develops inventory management systems. Your manager asks you to create a Product class and a Warehouse class with some basic methods. You have decided to use AI-powered code
suggestions to help speed up development and reduce syntax errors.



GitHub Copilot Free

Getting started — 1/3 complete

Install Copilot in your editor

Chat with Copilot anywhere

Open Copilot chat in the navigation from anywhere on GitHub. Try asking "what can I do with Copilot?" to get started.

Go to Copilot

Start building with Copilot

Usage

You are using Copilot for free

Responses are limited to 2,000 code completions and 50 chat messages per month.

Upgrade for higher limits, premium models, AI reviews. Free responses reset in 13 days. Upgrade

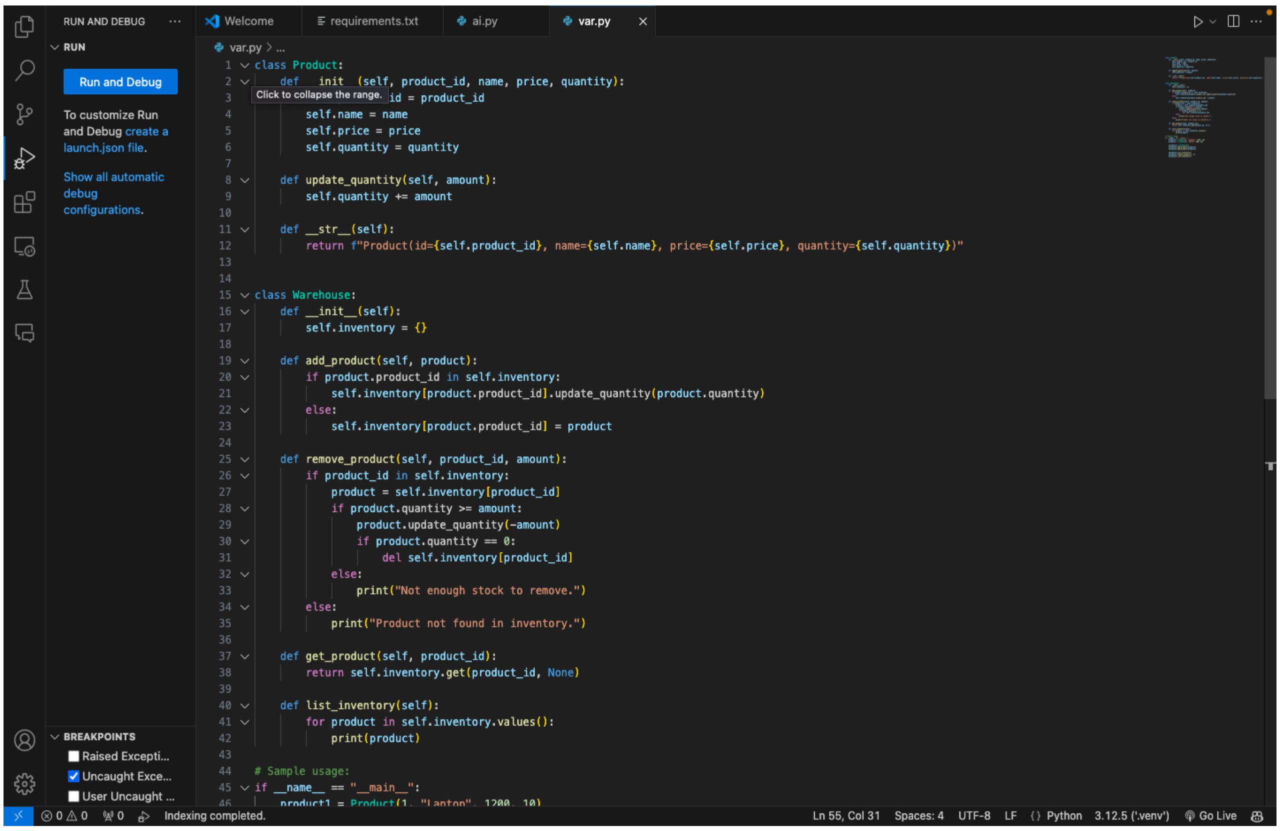
Features

Copilot in GitHub.com — Enabled

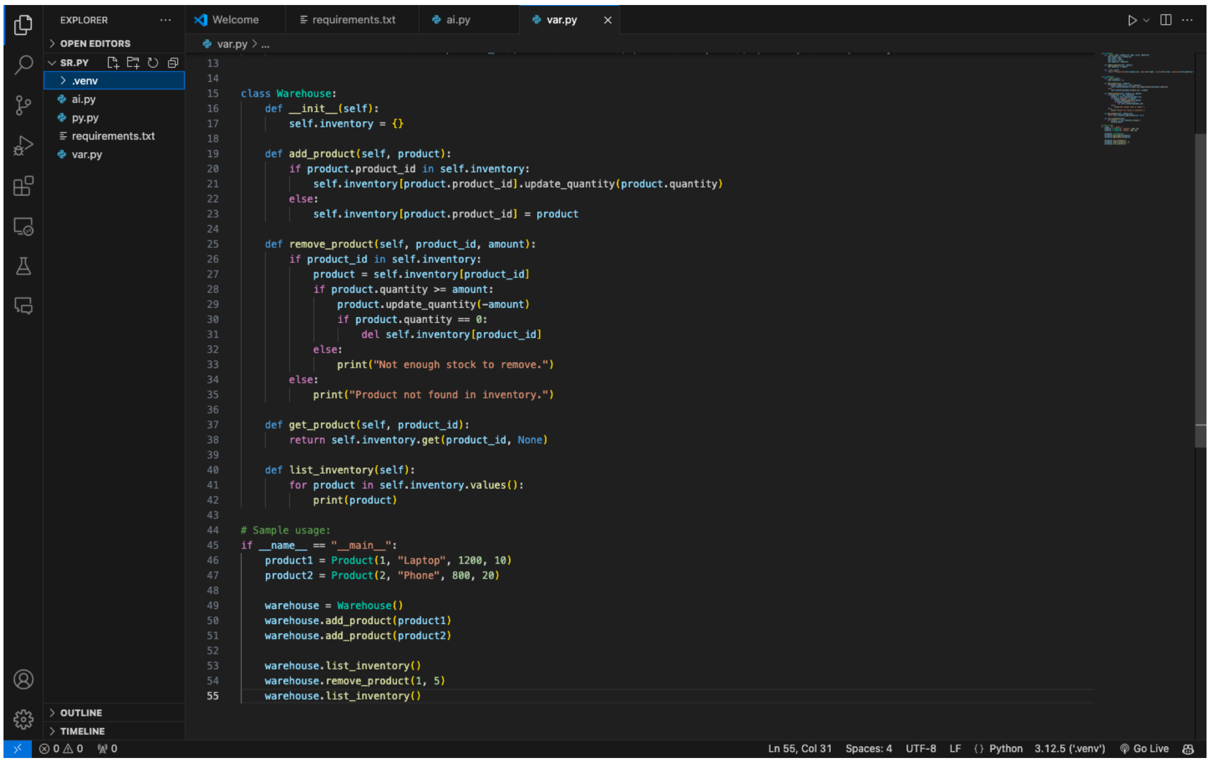
You can use Copilot Chat in GitHub.com. Copilot code review and preview features are only available for paid licenses. Learn more about Copilot in GitHub.com.

PROMPT:

Write a code that you are hired as an intern at a tech company that develops inventory management systems. Your manager asks you to create a Product class and a Warehouse class with some basic methods. You have decided to use AI-powered code
suggestions to help speed up development and reduce syntax errors.


RUN AND DEBUG
RUN
Run and Debug
To customize Run and Debug create a launch.json file.
Show all automatic debug configurations.
BREAKPOINTS
Raised Excepti...
Uncaught Exce...
User Uncaught ...
Welcome
requirements.txt
ai.py
var.py
var.py > ...
Click to collapse the range.
0 0 0 Indexing completed.
Ln 55, Col 31 Spaces: 4 UTF-8 LF {} Python 3.12.5 ('.venv') Go Live

```python
class Product:
    def __init__(self, product_id, name, price, quantity):
                               id = product_id
        self.name = name
        self.price = price
        self.quantity = quantity

    def update_quantity(self, amount):
        self.quantity += amount

    def __str__(self):
        return f"Product(id={self.product_id}, name={self.name}, price={self.price}, quantity={self.quantity})"


class Warehouse:
    def __init__(self):
        self.inventory = {}

    def add_product(self, product):
        if product.product_id in self.inventory:
            self.inventory[product.product_id].update_quantity(product.quantity)
        else:
            self.inventory[product.product_id] = product

    def remove_product(self, product_id, amount):
        if product_id in self.inventory:
            product = self.inventory[product_id]
            if product.quantity >= amount:
                product.update_quantity(-amount)
                if product.quantity == 0:
                    del self.inventory[product_id]
            else:
                print("Not enough stock to remove.")
        else:
            print("Product not found in inventory.")

    def get_product(self, product_id):
        return self.inventory.get(product_id, None)

    def list_inventory(self):
        for product in self.inventory.values():
            print(product)

# Sample usage:
if __name__ == "__main__":
    product1 = Product(1, "Laptop", 1200, 10)
```

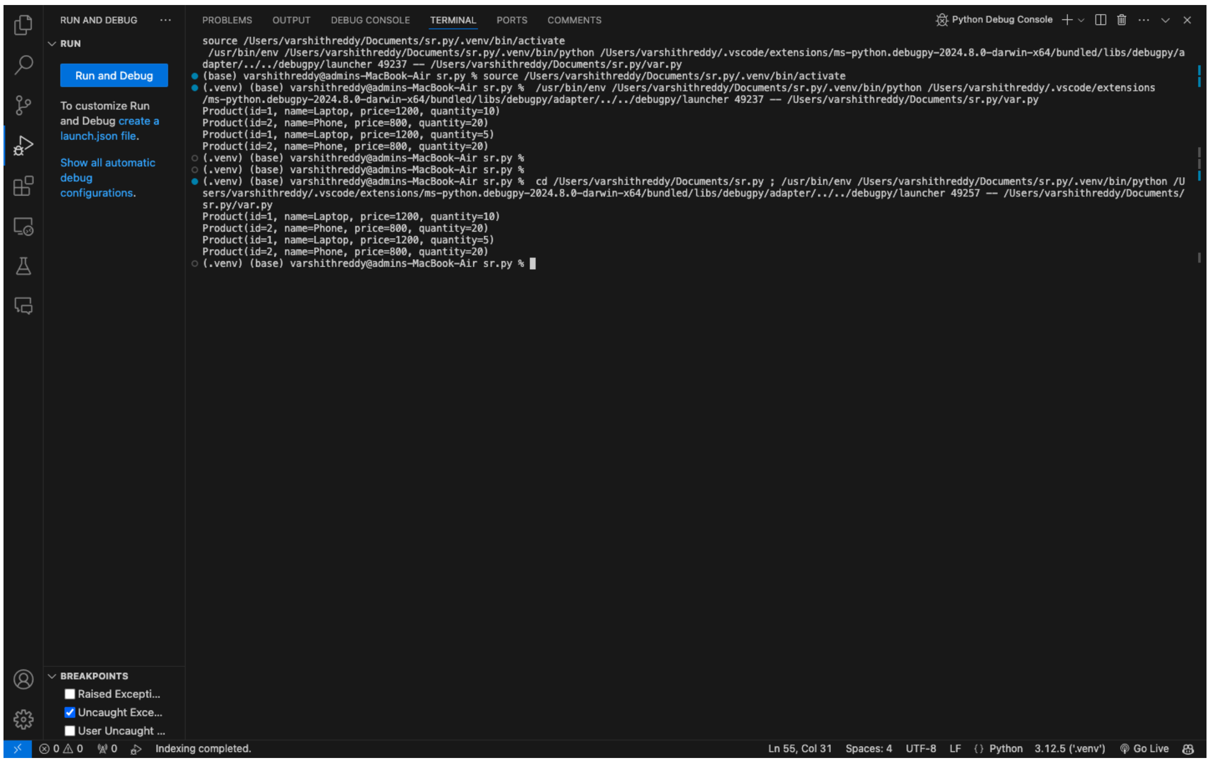


```python


class Warehouse:
    def __init__(self):
        self.inventory = {}

    def add_product(self, product):
        if product.product_id in self.inventory:
            self.inventory[product.product_id].update_quantity(product.quantity)
        else:
            self.inventory[product.product_id] = product

    def remove_product(self, product_id, amount):
        if product_id in self.inventory:
            product = self.inventory[product_id]
            if product.quantity >= amount:
                product.update_quantity(-amount)
                if product.quantity == 0:
                    del self.inventory[product_id]
            else:
                print("Not enough stock to remove.")
        else:
            print("Product not found in inventory.")

    def get_product(self, product_id):
        return self.inventory.get(product_id, None)

    def list_inventory(self):
        for product in self.inventory.values():
            print(product)

# Sample usage:
if __name__ == "__main__":
    product1 = Product(1, "Laptop", 1200, 10)
    product2 = Product(2, "Phone", 800, 20)

    warehouse = Warehouse()
    warehouse.add_product(product1)
    warehouse.add_product(product2)

    warehouse.list_inventory()
    warehouse.remove_product(1, 5)
    warehouse.list_inventory()
```

OBERVATION :

This code defines a simple warehouse management system using OOP concepts in Python. The Product class represents individual items with attributes like ID, name, price, and quantity, along with a method to update stock. The Warehouse class manages an inventory (stored as a dictionary), allowing products to be added, removed, retrieved, and listed. If a product already exists, its quantity is updated instead of duplicating it, and products are removed completely when their quantity reaches zero. The design is clean and functional for small-scale use, with readable outputs thanks to the __str__ method. However, it has limitations such as ignoring price updates when re-adding products, relying on print statements instead of proper error handling, exposing the inventory directly, and lacking search features beyond product IDs. Overall, it's a solid demo of inventory management with room for enhancements like exception handling, search by name, and support for price updates or persistence.

OUTPUT:



```
source /Users/varshithreddy/Documents/sr.py/.venv/bin/activate
 /usr/bin/env /Users/varshithreddy/Documents/sr.py/.venv/bin/python /Users/varshithreddy/.vscode/extensions/ms-python.debugpy-2024.8.0-darwin-x64/bundled/libs/debugpy/a
dapter/../../debugpy/launcher 49237 -- /Users/varshithreddy/Documents/sr.py/var.py
(base) varshithreddy@admins-MacBook-Air sr.py % source /Users/varshithreddy/Documents/sr.py/.venv/bin/activate
(.venv) (base) varshithreddy@admins-MacBook-Air sr.py %  /usr/bin/env /Users/varshithreddy/Documents/sr.py/.venv/bin/python /Users/varshithreddy/.vscode/extensions
/ms-python.debugpy-2024.8.0-darwin-x64/bundled/libs/debugpy/adapter/../../debugpy/launcher 49237 -- /Users/varshithreddy/Documents/sr.py/var.py
Product(id=1, name=Laptop, price=1200, quantity=10)
Product(id=2, name=Phone, price=800, quantity=20)
Product(id=1, name=Laptop, price=1200, quantity=5)
Product(id=2, name=Phone, price=800, quantity=20)
(.venv) (base) varshithreddy@admins-MacBook-Air sr.py %
(.venv) (base) varshithreddy@admins-MacBook-Air sr.py %
(.venv) (base) varshithreddy@admins-MacBook-Air sr.py %  cd /Users/varshithreddy/Documents/sr.py ; /usr/bin/env /Users/varshithreddy/Documents/sr.py/.venv/bin/python /U
sers/varshithreddy/.vscode/extensions/ms-python.debugpy-2024.8.0-darwin-x64/bundled/libs/debugpy/adapter/../../debugpy/launcher 49257 -- /Users/varshithreddy/Documents/
sr.py/var.py
Product(id=1, name=Laptop, price=1200, quantity=10)
Product(id=2, name=Phone, price=800, quantity=20)
Product(id=1, name=Laptop, price=1200, quantity=5)
Product(id=2, name=Phone, price=800, quantity=20)
(.venv) (base) varshithreddy@admins-MacBook-Air sr.py %
```