In [2]:
```
#ebook.py
class EBook:
    def __init__(self, title, author, publication_date, genre, pric
        self.title = title
        self.author = author
        self.publication_date = publication_date
        self.genre = genre
        self.price = price

    def __str__(self):
        return f"{self.title} by {self.author} ({self.genre}) - ${s

#customer.py
class Customer:
    def __init__(self, name, email, phone, loyalty_member):
        self.name = name
        self.email = email
        self.phone = phone
        self.loyalty_member = loyalty_member

    def __str__(self):
        return f"Customer {self.name} ({self.email})"

#shopping_cart.py
class ShoppingCart:
    def __init__(self):
        self.items = []

    def add_item(self, ebook):
        self.items.append(ebook)

    def apply_discount(self, loyalty_member, bulk_purchase):
        discount = 0
        if loyalty_member:
            discount = 0.10  #10% discount for loyalty members
        if bulk_purchase:
            discount = max(discount, 0.20)  #20% discount for bulk 
        return discount

    def __str__(self):
        return f"Cart contains {len(self.items)} items"

#order.py
class Order:
    def __init__(self, customer, items, order_date, discount):
        self.customer = customer
        self.items = items
        self.order_date = order_date
        self.discount = discount

    def generate_invoice(self):
        #calculate total price before discount
        total_price = sum(item.price for item in self.items)

        #apply discount
        discounted_price = total_price * (1 - self.discount)

        #calculate VAT
        vat = discounted_price * 0.08  # VAT is 8%

        #calculate final price
        final_price = discounted_price + vat

        #generate invoice string
        invoice = f"Invoice for {self.customer.name}:\n"
        invoice += f"Items: {', '.join([item.title for item in self
        invoice += f"Total: ${total_price:.2f}\n"
        invoice += f"Discount: {self.discount * 100}%\n"
        invoice += f"VAT: ${vat:.2f}\n"
        invoice += f"Final Price: ${final_price:.2f}\n"

        return invoice

#payment.py
class Payment:
    def __init__(self, amount, method):
        self.amount = amount
        self.method = method

    def process_payment(self, total_amount):
        self.amount = total_amount  #set the payment amount to the
        return f"Payment of ${self.amount:.2f} processed via {self.

#main.py (test script)

#create some e-books
ebook1 = EBook("Python for Beginners", "John Doe", "2023-05-01", "P
ebook2 = EBook("Advanced Python", "Jane Doe", "2024-06-15", "Progra

#create a customer (assuming they are a loyalty program member)
customer1 = Customer("Alice", "alice@example.com", "123-456-7890", 

#create a shopping cart and add e-books
cart = ShoppingCart()
cart.add_item(ebook1)
cart.add_item(ebook2)

#apply a discount (loyalty member, not bulk purchase)
discount = cart.apply_discount(customer1.loyalty_member, False)

#create an order
order = Order(customer1, cart.items, "2024-11-04", discount)

#generate the invoice
invoice = order.generate_invoice()
print(invoice)
```

```
#extract the final price from the invoice (after discount and VAT)
final_price = float(invoice.split("Final Price: $")[-1].strip())

#process the payment with the final price
payment = Payment(0, "Credit Card")  #initialize payment with zero 
print(payment.process_payment(final_price))  #process payment with 
```

```
Invoice for Alice:
Items: Python for Beginners, Advanced Python
Total: $65.99
Discount: 10.0%
VAT: $4.75
Final Price: $64.14

Payment of $64.14 processed via Credit Card
```

In [6]:
```
class ShoppingCart:
    """
    Represents a shopping cart containing e-books.
    """
    def __init__(self):
        self.items = []  # List of e-books in the cart

    def add_item(self, ebook):
        self.items.append(ebook)

    def remove_item(self, ebook):
        if ebook in self.items:
            self.items.remove(ebook)

    def create_order(self):
        # Create an order with the current items in the cart
        order = Order(self.items)
        return order
```

In [7]:
```
class Order:
    """
    Represents an order containing a list of e-books.
    """
    def __init__(self, items):
        self.items = items
        self.order_date = "2024-11-06"  # Example order date
        self.total_price = self.calculate_total_price()
        self.discount = 0
        self.vat_rate = 0.08  # VAT of 8%

    def calculate_total_price(self):
        return sum(item.price for item in self.items)

    def apply_loyalty_discount(self):
        if isinstance(self.items[0], EBook):  # Check if the first
            self.discount = 0.10  # 10% loyalty discount for demons
            self.total_price -= self.total_price * self.discount

    def apply_bulk_discount(self):
        if len(self.items) >= 5:
            self.discount = 0.20  # 20% bulk discount
            self.total_price -= self.total_price * self.discount

    def generate_invoice(self):
        vat = self.total_price * self.vat_rate
        final_price = self.total_price + vat
        invoice_details = f"Items: {', '.join([item.title for item
        invoice_details += f"Total: ${self.total_price:.2f}\n"
        invoice_details += f"Discount: {self.discount * 100}%\n"
        invoice_details += f"VAT: ${vat:.2f}\n"
        invoice_details += f"Final Price: ${final_price:.2f}"
        return invoice_details
```

In [10]:
```
class Payment:
    """
    Represents a payment for an order.
    """
    def __init__(self, order, payment_method, amount):
        self.order = order
        self.payment_method = payment_method
        self.amount = amount

    def process_payment(self):
        if self.amount >= self.order.total_price:
            return f"Payment of ${self.amount:.2f} processed via {s
        else:
            return "Insufficient funds. Payment not processed."
```

In [11]:
```
class EBook:
    """
    Represents an e-book in the store.
    """
    def __init__(self, title, author, genre, price):
        self.title = title
        self.author = author
        self.genre = genre
        self.price = price

    def __str__(self):
        return f"{self.title} by {self.author}, {self.genre}, ${sel
```

In [12]:
```
# Test the Customer class and add e-books to cart
ebook1 = EBook("Python for Beginners", "Alice Smith", "Programming"
ebook2 = EBook("Advanced Python", "Bob Brown", "Programming", 26.00

# Create a Customer without providing phone and loyalty_member
customer = Customer("Alice", "alice@example.com")

# Add e-books to the shopping cart
customer.add_to_cart(ebook1)
customer.add_to_cart(ebook2)

# Create an order from the shopping cart
order = customer.shopping_cart.create_order()

# Apply any discounts if applicable
order.apply_loyalty_discount()  # For loyalty member discount
order.apply_bulk_discount()  # For bulk purchase discount

# Generate invoice
invoice = order.generate_invoice()
print(invoice)

# Process payment
payment = Payment(order, "Credit Card", 64.14)
print(payment.process_payment())
```

```
Items: Python for Beginners, Advanced Python
Total: $59.39
Discount: 10.0%
VAT: $4.75
Final Price: $64.14
Payment of $64.14 processed via Credit Card
```

In [ ]: