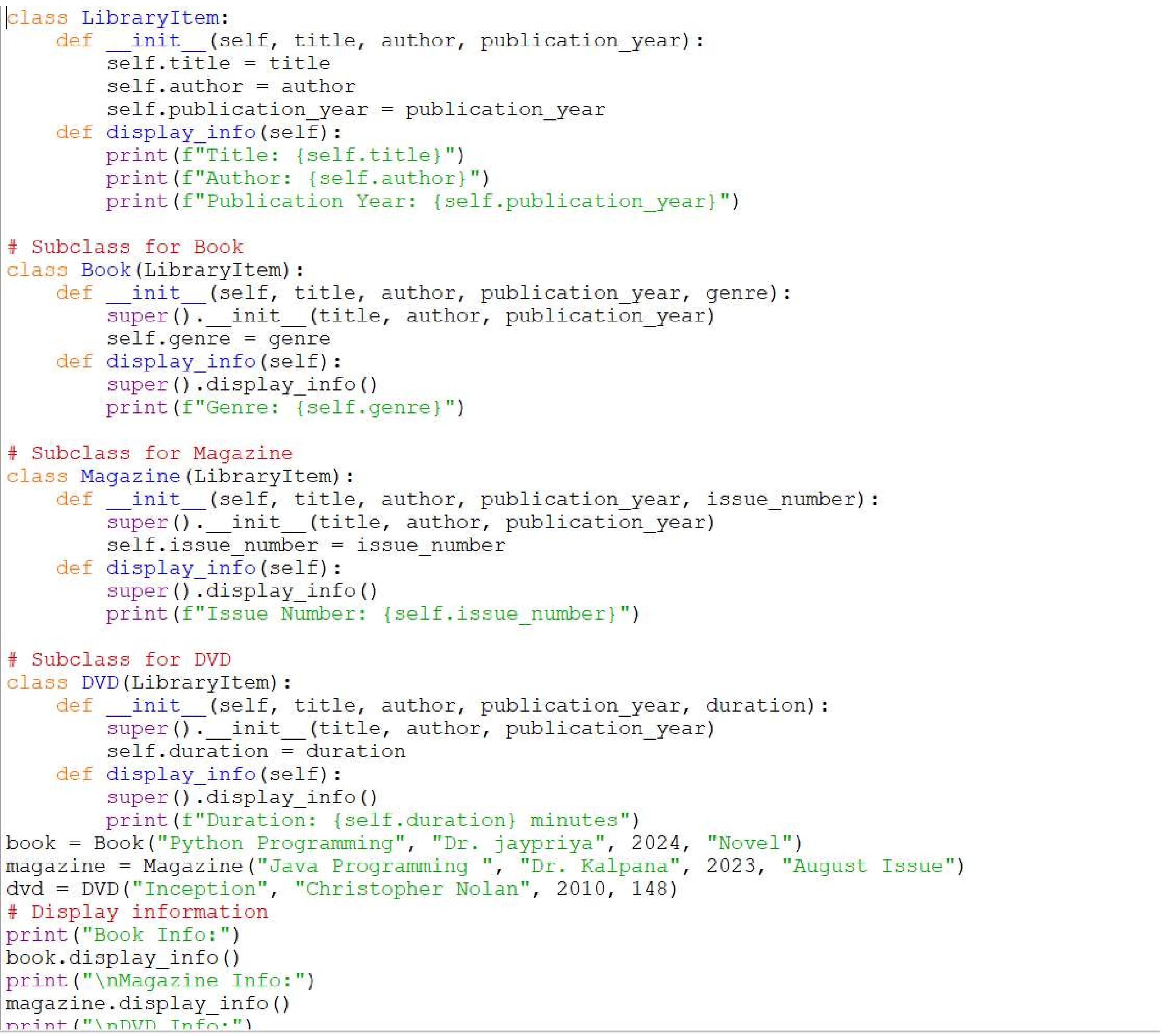
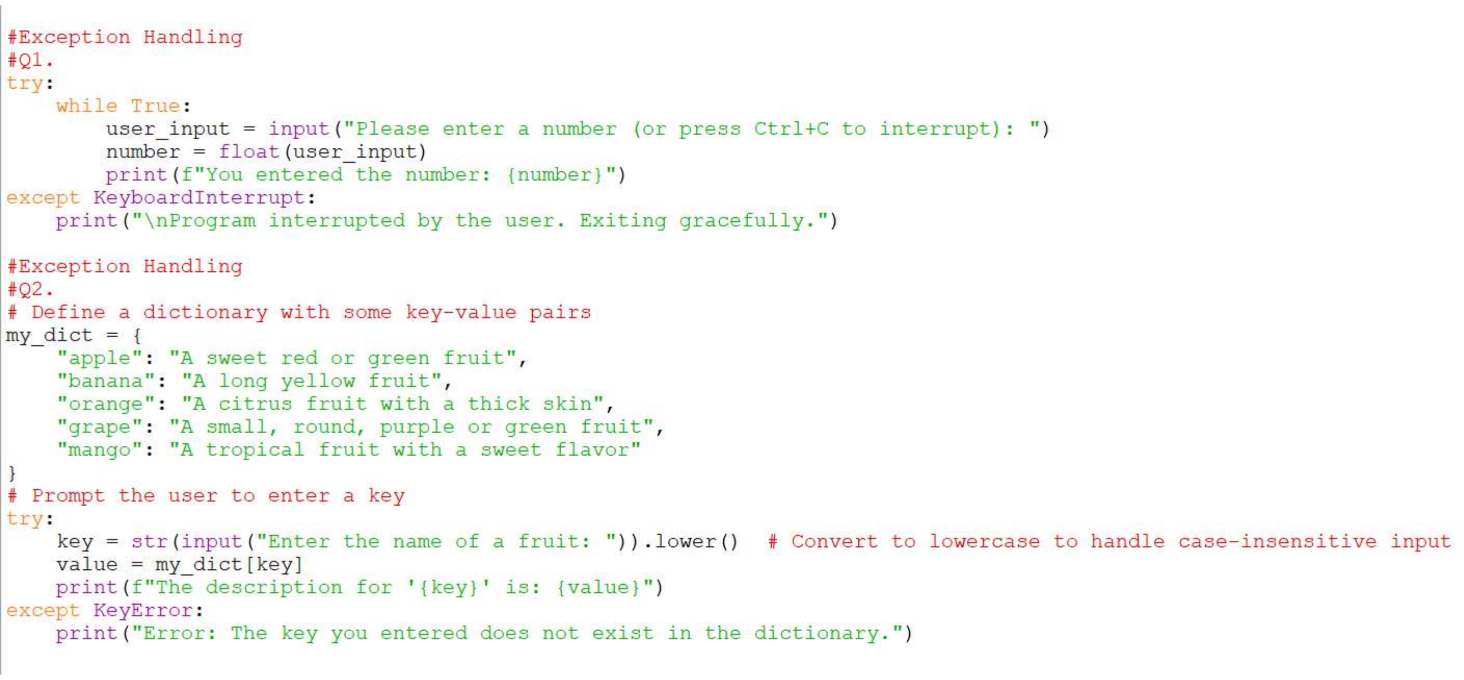


```python
class LibraryItem:
    def __init__(self, title, author, publication_year):
        self.title = title
        self.author = author
        self.publication_year = publication_year
    def display_info(self):
        print(f"Title: {self.title}")
        print(f"Author: {self.author}")
        print(f"Publication Year: {self.publication_year}")

# Subclass for Book
class Book(LibraryItem):
    def __init__(self, title, author, publication_year, genre):
        super().__init__(title, author, publication_year)
        self.genre = genre
    def display_info(self):
        super().display_info()
        print(f"Genre: {self.genre}")

# Subclass for Magazine
class Magazine(LibraryItem):
    def __init__(self, title, author, publication_year, issue_number):
        super().__init__(title, author, publication_year)
        self.issue_number = issue_number
    def display_info(self):
        super().display_info()
        print(f"Issue Number: {self.issue_number}")

# Subclass for DVD
class DVD(LibraryItem):
    def __init__(self, title, author, publication_year, duration):
        super().__init__(title, author, publication_year)
        self.duration = duration
    def display_info(self):
        super().display_info()
        print(f"Duration: {self.duration} minutes")
book = Book("Python Programming", "Dr. jaypriya", 2024, "Novel")
magazine = Magazine("Java Programming ", "Dr. Kalpana", 2023, "August Issue")
dvd = DVD("Inception", "Christopher Nolan", 2010, 148)
# Display information
print("Book Info:")
book.display_info()
print("\nMagazine Info:")
magazine.display_info()
print("\nDVD Info:")
```

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

    def calculate_total_cost(self):
        return self.price * self.quantity

# Subclass for PhysicalProduct
class PhysicalProduct(Product):
    def __init__(self, name, price, quantity, weight, shipping_cost_per_kg):
        super().__init__(name, price, quantity)
        self.weight = weight
        self.shipping_cost_per_kg = shipping_cost_per_kg
    def calculate_total_cost(self):
        product_cost = super().calculate_total_cost()
        shipping_cost = self.weight * self.shipping_cost_per_kg * self.quantity
        return product_cost + shipping_cost

# Subclass for DigitalProduct
class DigitalProduct(Product):
    def __init__(self, name, price, quantity, file_size):
        super().__init__(name, price, quantity)
        self.file_size = file_size
    def calculate_total_cost(self):
        return super().calculate_total_cost()

# Example usage
physical_product = PhysicalProduct("Laptop", 1000, 2, 2.5, 10)  # weight in kg, shipping cost per kg
digital_product = DigitalProduct("E-book", 15, 3, 2)  # file size in MB

# Calculate and display total cost
print(f"Total cost for physical product: ${physical_product.calculate_total_cost()}")
print(f"Total cost for digital product: ${digital_product.calculate_total_cost()}")
```



```
#Exception Handling
#Q1.
try:
    while True:
        user_input = input("Please enter a number (or press Ctrl+C to interrupt): ")
        number = float(user_input)
        print(f"You entered the number: {number}")
except KeyboardInterrupt:
    print("\nProgram interrupted by the user. Exiting gracefully.")

#Exception Handling
#Q2.
# Define a dictionary with some key-value pairs
my_dict = {
    "apple": "A sweet red or green fruit",
    "banana": "A long yellow fruit",
    "orange": "A citrus fruit with a thick skin",
    "grape": "A small, round, purple or green fruit",
    "mango": "A tropical fruit with a sweet flavor"
}
# Prompt the user to enter a key
try:
    key = str(input("Enter the name of a fruit: ")).lower()  # Convert to lowercase to handle case-insensitive input
    value = my_dict[key]
    print(f"The description for '{key}' is: {value}")
except KeyError:
    print("Error: The key you entered does not exist in the dictionary.")
```