Shopping Cart System

-Python Project

Sourav Kumar Singh

Introduction to the Shopping Cart Program:

The Shopping Cart Program is a Python-based application designed to simulate an online shopping experience. It offers a range of products across different categories, along with a coupon system for discounts, and incorporates basic user interaction for selecting products, applying discounts, and completing the checkout process.

Key Features

1. **Product Catalog**: The program features a diverse range of products categorized into three main sections:
    1. **Styling Category**: Includes fashion items like shoes, shirts, pants, and caps.
    2. **Self-Care Category**: Offers personal care products such as shampoos, conditioners, hair masks, and a hair serum.
    3. **College-Ready Category**: Contains items geared towards college students, like college bags and phone cases.

2. **Coupon System**: Each category comes with its own set of coupons that provide either a percentage discount or a flat rate off the total purchase price.

3. **User Interaction**: Through a text-based interface, users can browse products, add them to a shopping cart, and enter coupon codes for discounts.

4. **Checkout Process**: The program simulates a checkout process where users provide their contact information and an OTP verification is mimicked for order confirmation.

Target Audience

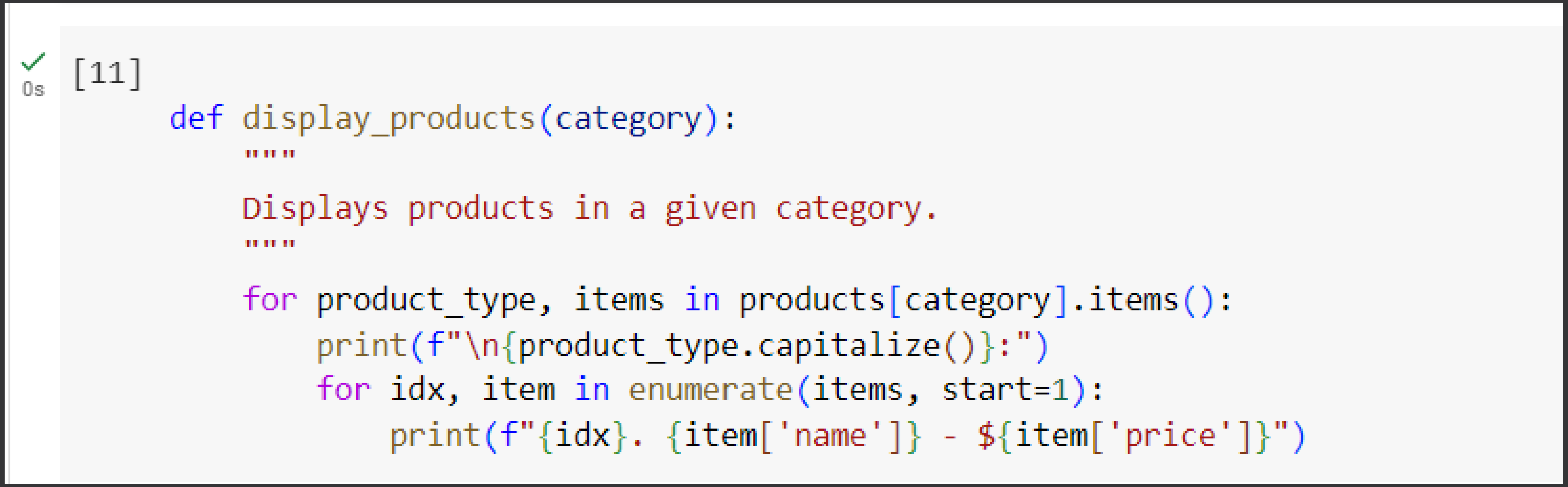
The project is ideal for students, beginner programmers, or anyone interested in understanding the basics of creating a simple e-commerce system in Python. It provides practical insights into how real-world applications function at a fundamental level.

Note:

This program is a basic simulation and does not include advanced features like a real-time database, payment processing, or an actual OTP system. It is meant for educational and demonstration purposes only.

display_products(category)

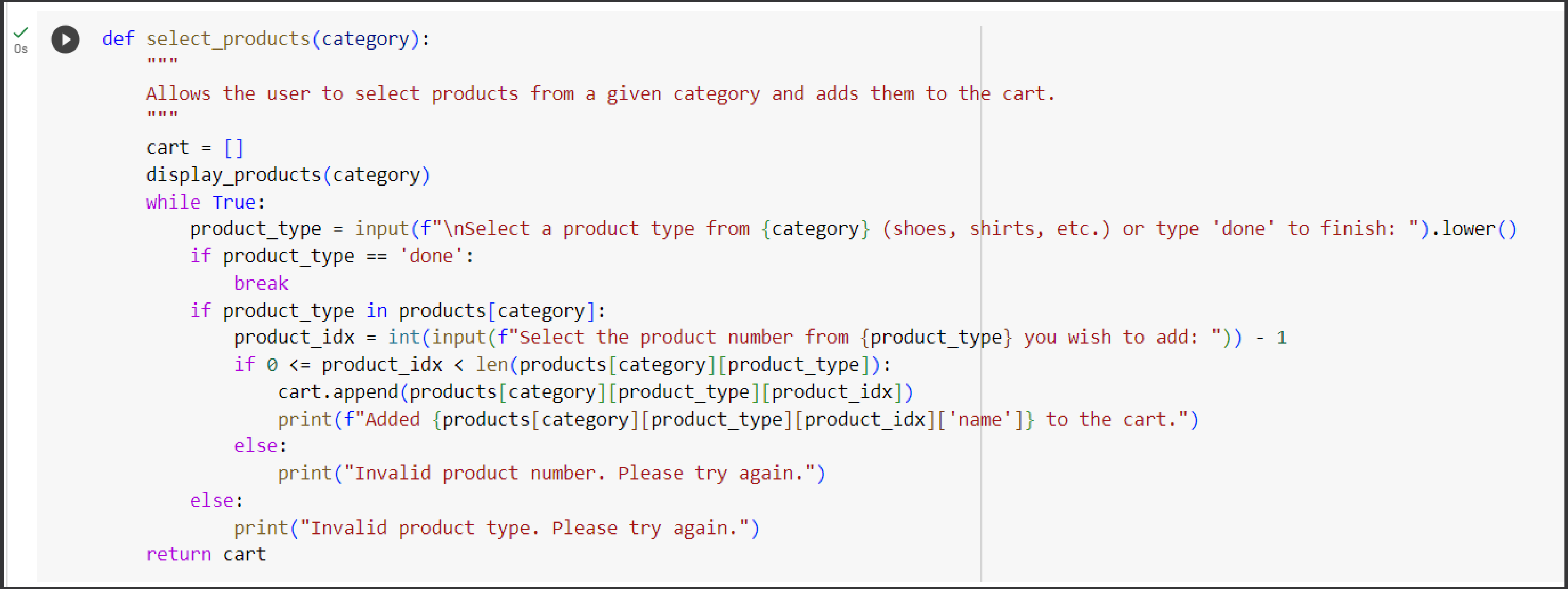
This function displays all the products available in a specified category. When you call this function with a category like 'styling', 'self_care', or 'college_ready', it lists all the products under that category with their names and prices.



```
[11]
def display_products(category):
    """
    Displays products in a given category.
    """
    for product_type, items in products[category].items():
        print(f"\n{product_type.capitalize()}:")
        for idx, item in enumerate(items, start=1):
            print(f"{idx}. {item['name']} - ${item['price']}")
```

select_products(category)

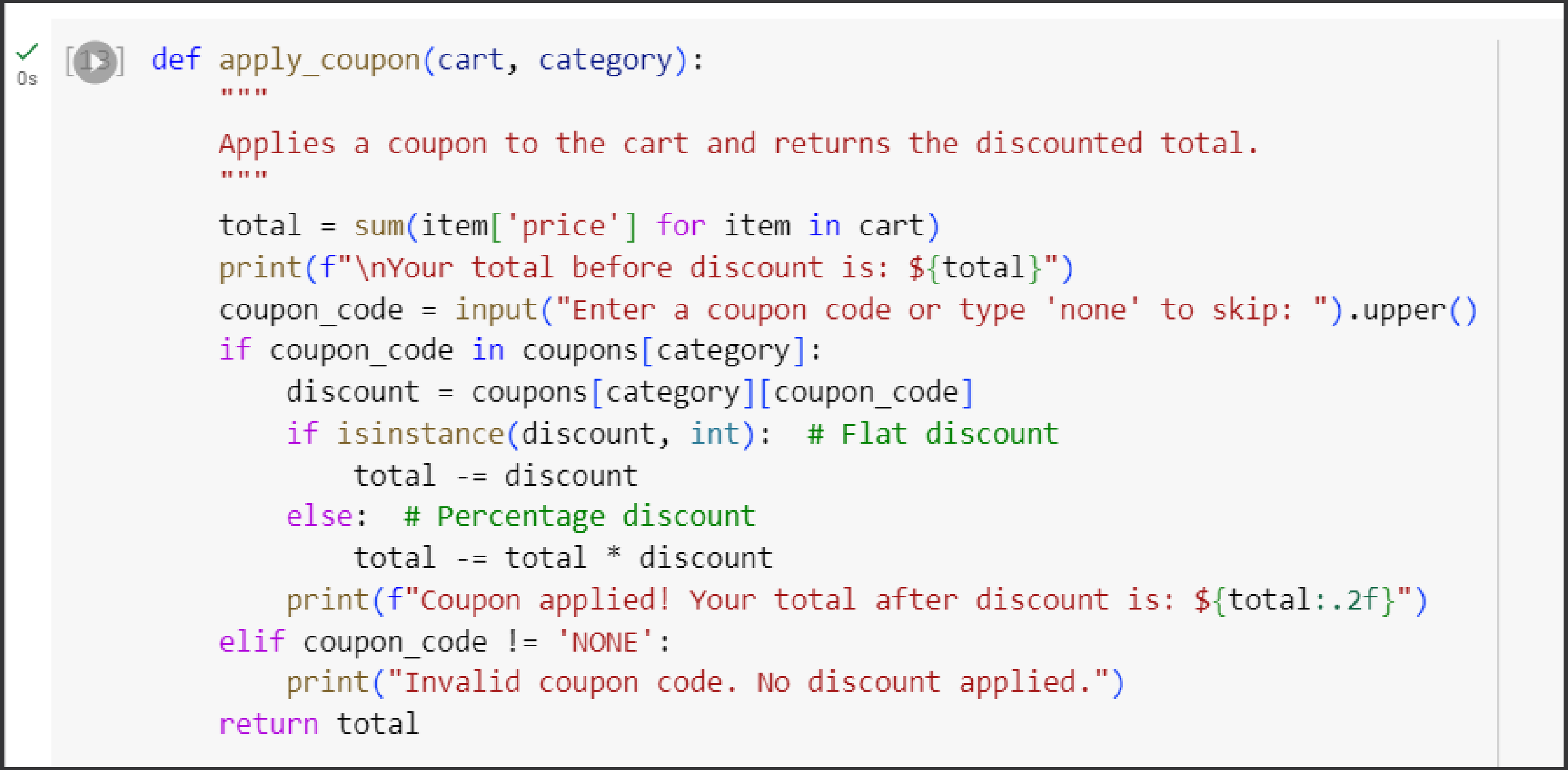
This function allows the user to select products from a given category. It first displays the products in the chosen category, then prompts the user to select a type of product (like 'shoes' or 'shirts' in the styling category). After the user selects a product type, it asks for the specific product number to add to the cart. The selected product is then added to the cart, which is a list of products.



```python
def select_products(category):
    """
    Allows the user to select products from a given category and adds them to the cart.
    """
    cart = []
    display_products(category)
    while True:
        product_type = input(f"\nSelect a product type from {category} (shoes, shirts, etc.) or type 'done' to finish: ").lower()
        if product_type == 'done':
            break
        if product_type in products[category]:
            product_idx = int(input(f"Select the product number from {product_type} you wish to add: ")) - 1
            if 0 <= product_idx < len(products[category][product_type]):
                cart.append(products[category][product_type][product_idx])
                print(f"Added {products[category][product_type][product_idx]['name']} to the cart.")
            else:
                print("Invalid product number. Please try again.")
        else:
            print("Invalid product type. Please try again.")
    return cart
```

apply_coupon(cart, category)

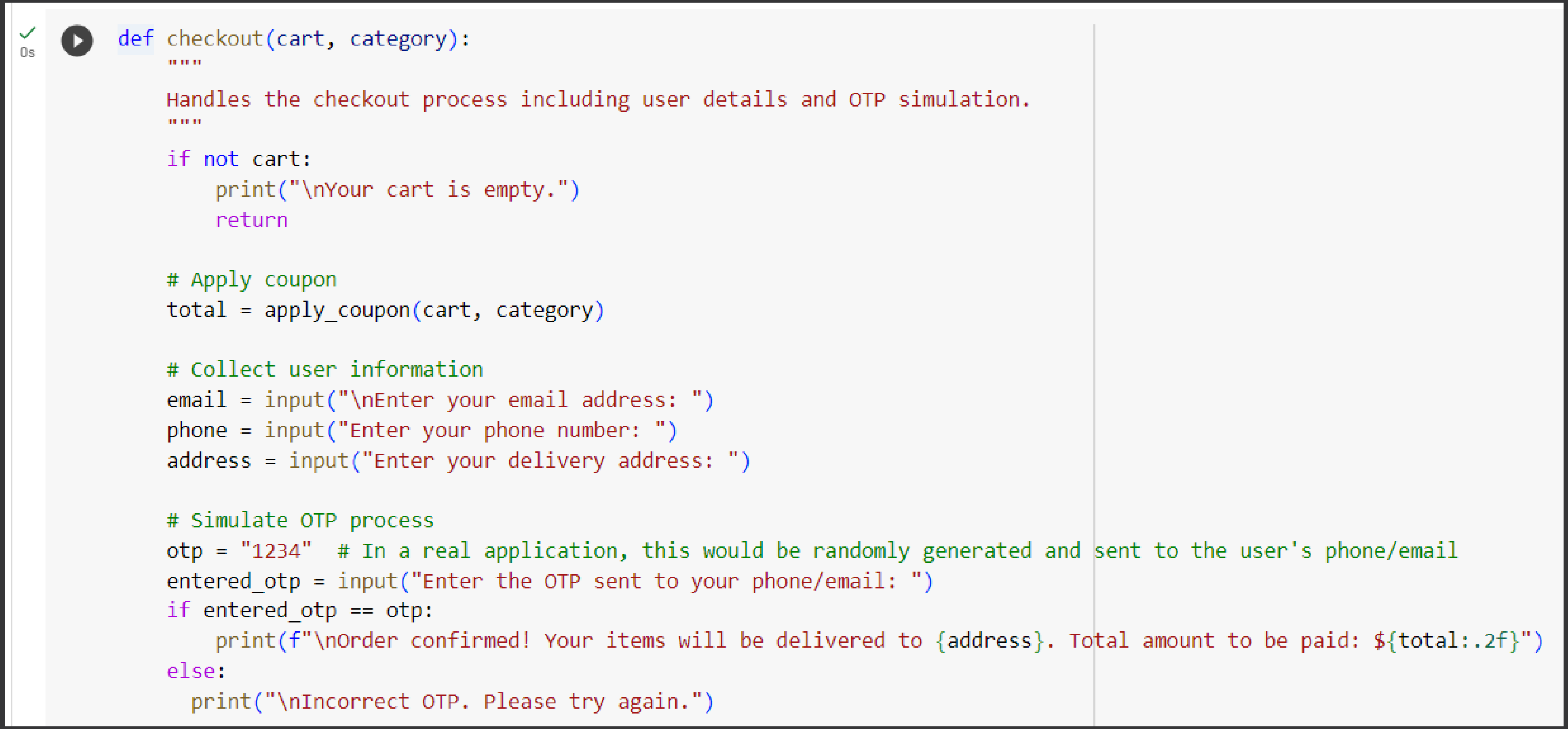
After the user has selected products and added them to their cart, this function applies a coupon code for a discount. It calculates the total price of the cart, prompts the user to enter a coupon code, and then applies the corresponding discount if the coupon is valid for the selected category. The function then returns the total amount after the discount has been applied.



```
def apply_coupon(cart, category):
    """
    Applies a coupon to the cart and returns the discounted total.
    """
    total = sum(item['price'] for item in cart)
    print(f"\nYour total before discount is: ${total}")
    coupon_code = input("Enter a coupon code or type 'none' to skip: ").upper()
    if coupon_code in coupons[category]:
        discount = coupons[category][coupon_code]
        if isinstance(discount, int):  # Flat discount
            total -= discount
        else:  # Percentage discount
            total -= total * discount
        print(f"Coupon applied! Your total after discount is: ${total:.2f}")
    elif coupon_code != 'NONE':
        print("Invalid coupon code. No discount applied.")
    return total
```

checkout(cart, category)

This function handles the checkout process. It only proceeds if the cart is not empty. The function first applies any coupon to the cart to get the total price. Then, it prompts the user to enter their email, phone number, and delivery address. After this, it simulates an OTP (One Time Password) process where it asks the user to enter a pre-defined OTP (for demonstration purposes). If the OTP is entered correctly, it confirms the order and displays the total amount to be paid along with the delivery address.



```python
def checkout(cart, category):
    """
    Handles the checkout process including user details and OTP simulation.
    """
    if not cart:
        print("\nYour cart is empty.")
        return

    # Apply coupon
    total = apply_coupon(cart, category)

    # Collect user information
    email = input("\nEnter your email address: ")
    phone = input("Enter your phone number: ")
    address = input("Enter your delivery address: ")

    # Simulate OTP process
    otp = "1234"  # In a real application, this would be randomly generated and sent to the user's phone/email
    entered_otp = input("Enter the OTP sent to your phone/email: ")
    if entered_otp == otp:
        print(f"\nOrder confirmed! Your items will be delivered to {address}. Total amount to be paid: ${total:.2f}")
    else:
      print("\nIncorrect OTP. Please try again.")
```

Each function is designed to handle a specific part of the shopping process, making the code modular and easier to manage. This structure also makes it simpler to update individual parts of the program in the future, such as adding new products or changing how discounts are calculated.