# 

# Project Title: DSLR Camera Sales

# Program: Cost and Billing

# Source Code: Python

## 

# GitHub Repository Link:

https://github.com/mahi2008324/BILLING\_project.git

**BY**

**KOTHA MAHEEDHAR**

**Table of Contents:**

**Chapter 1:** Introduction………………………………………………………………………………………pgno:2

**Chapter 2:** Understanding the Concept………………………………………………………………..pgno:4

**Chapter 3:** Input Handling and Discount Logic……………………………………………………..pgno:5

**Chapter 4:** Steps for Calculations………………………………………………………………………….pgno:6

**Chapter 5:** Output and Billing Summary………………………………………………………………..pgno:7

**Chapter 6:** Potential Enhancements and Code……………………………………………………….pgno:8

## Chapter 1:

## Introduction to the DSLR Camera Sales Program

This program is designed to calculate the total cost a customer must pay when purchasing DSLR cameras from a retail store. It takes into account the number of cameras bought, applies a tiered discount based on quantity, adds applicable taxes, and then produces a detailed billing summary. The primary goal is to automate the billing process, ensuring accuracy and efficiency in retail sales.

In many retail environments, calculating final prices can become complex due to varying discounts and tax components. This program addresses that challenge by simplifying the computation into clear, logical steps. It is particularly useful for business owners and sales staff who want to quickly determine the payable amount for customers buying multiple units, all while factoring in government-mandated taxes and volume-based discounts.

The key components of the program include:

* **Pricing:** The fixed cost of a single DSLR camera, set here as ₹30,000.
* **Tax Rates:** Central and state taxes, specifically CGST and SGST at 14% each, which are standard in many regions for goods and services.
* **Input Handling:** The program prompts the user to enter the quantity of cameras required, allowing dynamic calculations.
* **Discount Logic:** Discounts vary by purchase quantity, incentivizing larger orders and reflecting common retail practices.
* **Billing Summary:** After calculations, a clear output displays each cost component and the final payable amount.

Overall, this program serves as a practical example of how simple programming techniques can facilitate retail billing processes and enhance customer service by delivering transparent and precise price information.

Chapter 2:

Understanding the Program Variables and Constants

The program defines several key variables and constants that form the foundation for all the calculations involved in determining the total cost for purchasing DSLR cameras. These elements are crucial for ensuring the accuracy of pricing, tax application, and discounts.

First, the **DSLR\_camera\_cost** constant is set to *30,000 INR*, representing the base price of a single camera. This value is chosen to reflect a typical mid-range DSLR camera price in the given market, providing a realistic context for the billing calculations.

Next, the program defines two tax rate constants: **SGST** and **CGST**, each assigned a value of *14/100*. These are stored as floating-point numbers (0.14) to facilitate precise arithmetic operations when calculating tax amounts. Representing percentages as floats is a standard practice in programming because it allows direct multiplication with monetary values without additional conversions. This approach simplifies the logic and minimizes rounding errors.

The input variable **required\_items** is an integer value entered by the user, representing how many DSLR cameras the customer wishes to purchase. This variable directly influences the discount percentage and the total cost calculation.

Each of these variables plays an essential role in the calculations:

* **DSLR\_camera\_cost:** Basis for total camera price before tax and discount.
* **SGST and CGST:** Applied separately to the total cost to compute the final tax amounts according to government standards.
* **required\_items:** Determines the scale of the purchase, directly affecting the discount rate, which incentivizes larger orders.

By setting these parameters explicitly and clearly, the program maintains transparency and ease of modification, such as updating tax rates or base prices without altering the core logic.

## 

## Chapter3:

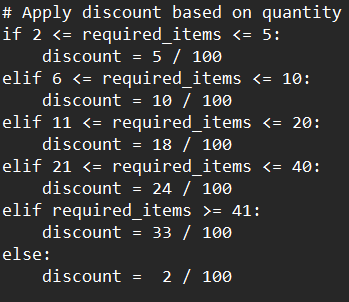
## Input Handling and Discount Logic

The program begins by capturing user input for the number of DSLR cameras to be purchased through the line:

required\_items = int(input("No. of required items: "))

Here, the input() function prompts the user to enter a quantity, which is then converted into an integer using int(). This approach assumes valid numeric input, but in practical scenarios, additional validation would be necessary to ensure robustness. For instance, handling non-integer inputs or negative values would prevent runtime errors and enhance user experience.

Following the input, the program uses a series of conditional statements to determine the applicable discount rate based on the quantity purchased. This tiered discount system is implemented as follows:

* **1 item:** 2% discount (default for below 2 units)
* **2 to 5 items:** 5% discount
* **6 to 10 items:** 10% discount
* **11 to 20 items:** 18% discount
* **21 to 40 items:** 24% discount
* **41 or more items:** 33% discount

Each discount is represented as a floating-point fraction (e.g., 5 / 100 for 5%), which enables direct multiplication with the total price to calculate the precise discount amount. This method avoids ambiguity compared to using whole percentages and maintains consistency in arithmetic operations.

The graduated discount structure rewards bulk buyers with increasingly larger discounts, encouraging higher volume purchases. Such tiering mimics real-world retail strategies where economies of scale and competitive pricing provide customers with financial incentives.

Although the current script captures the quantity and applies discounts effectively, it lacks explicit input validation. Adding checks to confirm that input is a positive integer would prevent logical errors and improve resilience. For example, using a try-except block and verifying that required\_items is greater than zero would ensure robust handling of user inputs.

## Chapter 4:

## Calculations for Total Cost Including Taxes and Discounts

The core calculation section of the program systematically determines the final amount payable by the customer by breaking down the costs into several components and combining them logically. The process follows a clear order of operations to ensure accuracy and transparency.

### Step 1: Calculating Total Camera Cost

Initially, the program computes the total cost of the DSLR cameras before applying any taxes or discounts. This is done by multiplying the number of cameras requested (required\_items) by the fixed unit cost (DSLR\_camera\_cost).

For example, 

### Step 2: Computing SGST and CGST

Next, the program calculates the tax amounts separately for State GST (SGST) and Central GST (CGST). Both taxes are applied on the pre-discounted total camera cost, ensuring compliance with tax regulations. These amounts are computed as:

* **SGST:** DSLR\_SGST = DSLR\_total\_cost × SGST
* **CGST:** DSLR\_CGST = DSLR\_total\_cost × CGST

### Step 3: Calculating Discount Amount

The discount is a percentage based on the quantity bracket the purchase falls into. This discount is applied only on the original total camera cost, not on the taxes.



### Step 4: Deriving the Total Payable Amount

Finally, the program derives the total payable amount by adding both tax components to the total cost and then subtracting the discount amount:

This ordering of operations—applying taxes before subtracting the discount—ensures that taxes are calculated on the gross amount, complying with typical tax regulations where discounts reduce the final sale price but do not reduce the tax base.

Through this structured approach, the program transparently outlines each cost component, enhancing clarity for both the retailer and the customer.



## 

## Chapter 5:

## Output and Billing Summary Presentation

After completing all calculations, the program presents a clear and organized billing summary to the user through a series of print() statements. This output serves as the final communication of the transaction details, ensuring the customer understands the cost breakdown before making a payment.

The summary begins with a clearly labeled header:

print("------------ Billing Summary --------------------")

This line visually separates the billing information from other output, making it easier to read and comprehend.

Each key component of the pricing is displayed on its own line, accompanied by descriptive labels and formatted with the Indian Rupee symbol **₹** to indicate currency. The displayed items include:

* **Total Camera Cost:** The combined price of all cameras before taxes and discounts.
* **SGST (14%):** The state tax amount calculated on the base price.
* **CGST (14%):** The central tax component, also based on the base price.
* **Discount Applied:** The percentage discount along with the exact monetary amount subtracted.
* **Total Payable Amount:** The final amount that the customer must pay after adding taxes and subtracting the discount.

For example, the discount line prints the percentage alongside the value to reinforce transparency about the savings offered:

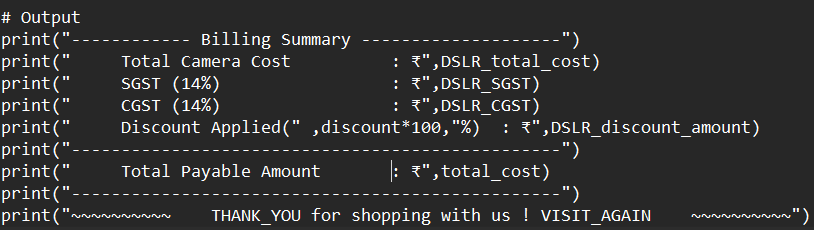
print(" Discount Applied(" ,discount,") : ₹",DSLR\_discount\_amount)

Though this method is straightforward, displaying the discount rate as a decimal (e.g., 0.05) could be confusing for some users; converting it to a percentage format (like "5%") would improve clarity.

Finally, the billing summary is concluded with a visually distinct separator and a polite thank-you message:

print("-------------------------------------------------")  
print(".<.<.< THANK\_YOU AND VISIT\_AGAIN >.>.>.")

This closing line enhances user experience by adding a friendly and professional touch to the interaction.



### 

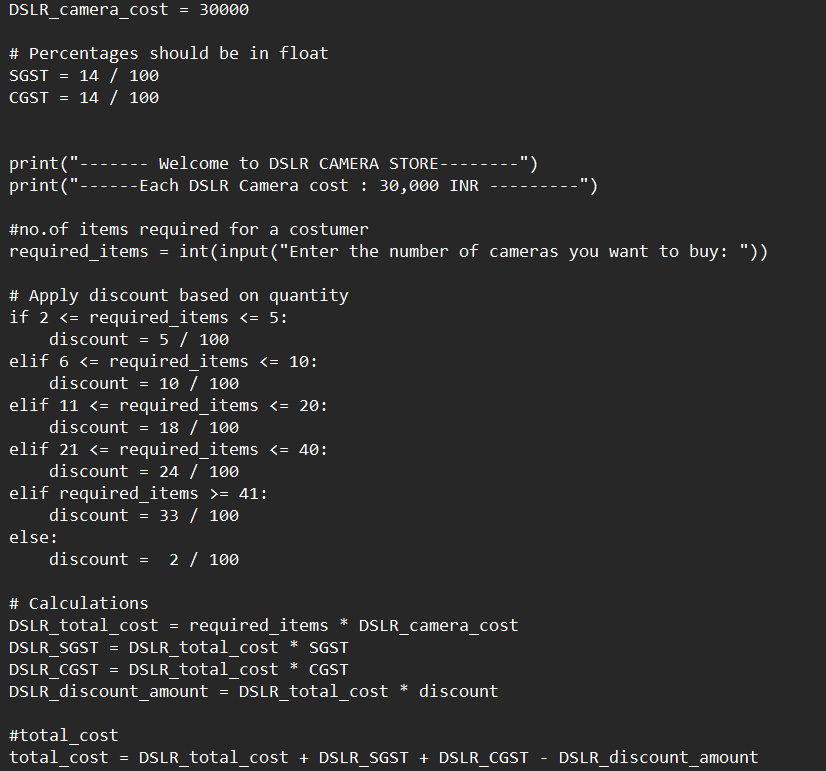
### Chapter 6:

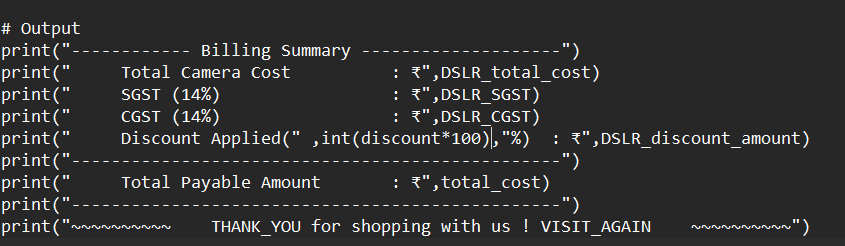
### Potential Enhancements

* **Decimal Formatting:** Formatting monetary values to two decimal places would look more professional, using Python’s format() or f-strings with :.2f specifier.
* **Input Validation Feedback:** Giving immediate feedback if the user enters invalid data (like negative numbers or non-integers) would prevent confusion and errors.
* **Graphical User Interface (GUI):** Developing a simple GUI could improve usability by offering buttons for input and displaying the billing summary in a neatly formatted window instead of the console.

In summary, the program’s output section achieves a clear and structured presentation of the billing summary, which is essential for customer confidence and retail transparency.

# 7.Total Code





# 8.Sample output:

