Building a GUI-Based Shopping Cart System In Python

1. Import Libraries

i. Why do we give tkinter a nickname (tk) when importing it?
To make it shorter and easier to type in the code.

ii. What practical tasks do messagebox, ttk, and filedialog each handle inside a GUI application?

- `messagebox`: shows pop-up alerts (e.g., login failed).
- `ttk`: gives better-looking buttons, tables, etc.
- `filedialog`: lets users pick a file from a folder.

iii. When working with CSV files, why is using Python's csv module safer or easier than manually splitting text on commas?
It avoids problems with commas inside data (like names or text).

2. Sample Product List and User Data

i. Why might a list of small dictionaries be a better way to store multiple products than one large dictionary of lists?
It's easier to add, find, or remove one product at a time.

ii. What security problems could arise from keeping passwords in plain text inside a dictionary, and how would you protect them in a real application?
Someone can see passwords easily. Use hashing to protect them.

iii. If you need to find the price of the item whose ID equals 3, which two dictionary keys would you use inside the product list to locate and display it?
Use the `id` key to find it, then `price` key to get its price.

3. Login Module

i. Why does the function retrieve text from the on-screen entry fields instead of relying on global variables?
Because it checks what the user typed in real time.

ii. What advantage does the dictionary method .get() provide when looking up a username, compared with direct bracket indexing?
It avoids an error if the username is not found.

iii. Why is it important to close the login window before opening the main application interface after a successful login?
To save memory and keep the screen clean.

4. Add to Cart Module

i. Why do we check for an empty selection before entering the loop, instead of letting the loop run with no items?
To stop adding nothing or getting an error.

ii. What is the benefit of converting the price text to a numeric value right away, instead of keeping it as a string until checkout time?
So we can add prices correctly without errors.

iii. How would the program's behaviour change if we refreshed the cart inside the loop (after each item) instead of once at the end?
It would slow down the program and refresh too many times.

5. Refresh Cart Function

i. Why do we completely clear the cart display before inserting the current items, instead of trying to edit rows in place?
It's simpler and avoids old or wrong data.

ii. What might happen to the total price label if we forgot to run the calculation step after updating the cart table?
It would show the wrong total amount.

iii. Why is it helpful to format each price with exactly two decimal places in the cart display and total label?
It looks clean and shows exact money values.

6. Calculate Total Function

i. Why is it useful to place the price-adding logic inside its own function instead of repeating the addition code every time we need the total?
To avoid writing the same code again and again.

ii. If one product in the cart were missing a price field or contained text instead of a number, how would that affect the total calculation—and how could you guard against it?
It would give an error. Use try/except to handle it.

iii. Explain in your own words what the loop inside this function is doing and why it doesn't need to reference product names or categories.
It just adds up prices, so names/categories are not needed.

7. Sort Cart Function

i. What is the purpose of supplying a "key" rule (lambda) to the list's sort method, and how does it differ from sorting plain strings or numbers?
It tells Python exactly what to sort by in each product.

ii. If the user accidentally requests a sort field that isn't handled (for example "name"), what would the function do today, and how could you make it more robust?
It would do nothing or crash. Use an else block with a warning.

iii. Why does the code call the refresh routine after sorting instead of before, and what would the user experience be if that call was omitted?
So users can see the new sorted list. Without it, screen won't update.

8. Export Bill Function

i. Why is it important to stop the function early when the cart is empty instead of letting the user pick a filename?
To avoid saving a blank file.

ii. What could happen if you forgot to include the header row before writing the product lines?
The CSV will open without column titles, which looks confusing.

iii. Explain why using a file dialog is safer than hard-coding a filename inside the program.
User can choose location and name, avoiding overwrite or errors.

9. Display Product Function

i. Why does the function wipe the table clean before inserting the new products, instead of trying to update rows that might already be there?
To avoid showing old or duplicate data.

ii. What is the advantage of formatting each price to exactly two decimal places from a user-experience perspective?
It looks neat and professional like a real price tag.

iii. If you wanted to add a fourth column that shows each product's numerical ID, what changes would you need to make inside the loop?
Add one more column and include product[`id`] in insert.

10. Search Product Function

i. Why do we convert both the search text and the product fields to lowercase before doing the comparison?
To make search case-insensitive.

ii. Explain the logic used to decide whether a product belongs in the results list. Why do we search in both the name and the category?
To give better results. User may search by type or name.

iii. If you wanted the search function to also match on price (for example, the user types "50"), what extra condition would you add, and how might that change the display?
Check if search text is in the price too. More items will show up.

11. Apply Discount Function

i. Why is a "try/except" block necessary when converting the discount text to a number, and what would happen without it?
To stop crashes if user types wrong input.

ii. Explain how the formula 1 – rate / 100 ensures the correct discounted price is applied to each item.
It gives the part user pays after discount.

iii. If you wanted to prevent users from entering percentages above 100 or below 0, where and how would you add that check?
Check the number after converting, and show error if it's out of range.

12. Checkout Function

i. Why does the function return immediately if the cart is empty instead of letting the code continue?
To avoid errors and inform the user clearly.

ii. Explain why it's important to clear the cart after a successful checkout. What problems might occur if you skipped this step?
Old items would still be there for the next order.

iii. How does formatting the total amount to two decimal places improve clarity for the shopper?
It shows the exact final amount like real receipts.

13. Main Window of Shopping Cart App

i. Why are some widgets declared global inside the setup function?
So other functions can use them too.

ii. How does the grid layout system decide where each widget appears?
By using row and column numbers in code.

iii. Tree-view tables vs ordinary labels or buttons — what special job do Tree-views perform in this interface, and what features make them suitable for that job?
They can show multiple rows and columns like tables.

14. Build Login Window

i. Why should the password field display asterisks instead of the actual characters the user types?
To keep the password hidden and private.

ii. What happens if the “Login” button is not linked to any function?

Clicking it will do nothing.

iii. Why does the program need an “event loop” at the end of the setup, and what would occur if that loop were omitted?

Without it, the window will close instantly and not work.