1. `def find\_alternate\_medicine(df):`: defines a function named `find\_alternate\_medicine` that takes a DataFrame `df` as an argument.

2. `if 'Medicine Name' not in df.columns:`: checks if the DataFrame `df` has a column named 'Medicine Name'.

3. `print("Error: 'Medicine Name' column not found in the DataFrame.")`: prints an error message if the 'Medicine Name' column is not found in the DataFrame.

4. `return`: exits the function if the 'Medicine Name' column is not found, as there's no point in proceeding further.

5. `if 'Alternate Medicines' not in df.columns:`: checks if the DataFrame `df` has a column named 'Alternate Medicines'.

6. `print("Error: 'Alternate Medicines' column not found in the DataFrame.")`: prints an error message if the 'Alternate Medicines' column is not found in the DataFrame.

7. `return`: exits the function if the 'Alternate Medicines' column is not found.

8. `medicine\_name\_input = input("Enter the name of the medicine: ")`: prompts the user to input the name of the medicine and assigns the input value to the variable `medicine\_name\_input`.

9. `matching\_medicines = df[df['Medicine Name'].str.contains(medicine\_name\_input, case=False)]`: filters the DataFrame `df` to find rows where the 'Medicine Name' column contains the value stored in `medicine\_name\_input`. It ignores the case sensitivity.

10. `if not matching\_medicines.empty:`: checks if there are any matching medicines found based on the user input.

11. `print("Matching medicines:")`: prints a header indicating the list of matching medicines.

12. `for i, (index, row) in enumerate(matching\_medicines.iterrows(), start=1):` iterates through the rows of the DataFrame `matching\_medicines`, assigning the index and row data to variables `index` and `row`, respectively. The `enumerate()` function is used to track the iteration count, starting from 1.

13. `print(f"{i}. {row['Medicine Name']}")`: prints the medicine name along with its corresponding index number.

14. `alternate\_medicine = row['Alternate Medicines']`: retrieves the value from the 'Alternate Medicines' column for the current row.

15. `if pd.notna(alternate\_medicine):` checks if the 'Alternate Medicines' value is not NaN (not a number).

16. `print(f" Alternate medicine: {alternate\_medicine}")`: prints the alternate medicine if it exists.

17. `else:`: indicates that if the 'Alternate Medicines' value is NaN, it means there is no alternate medicine available.

18. `print(" Given Medicine alternate is not available with us kindly contact the nearby Store ! ")`: prints a message indicating that the alternate medicine is not available.

19. `else:`: executes if no matching medicines are found.

20. `print(f"No medicine found matching '{medicine\_name\_input}'.")`: prints a message indicating that no medicines matching the user input were found in the DataFrame.

21. `find\_alternate\_medicine(df)`: calls the `find\_alternate\_medicine` function with the DataFrame `df` as an argument, initiating the execution of the function.