Code Explanation: Investor ITR & GST Calculator (Detailed)

This document provides a detailed, line-by-line explanation of the Python code used in the "Investor ITR & GST Calculator" project.

1. `calculator.py` - The Core Logic

This file is the heart of the application, containing all the business logic for the financial calculations.

`Trade` Class

This class models a single transaction.

```python
class Trade:
    def __init__(self, date: datetime, trade_type: str, stock: str,
                 qty: int, price: float, brokerage: float, dividend: float = 0):
        self.date = date
        self.trade_type = trade_type.upper()
        self.stock = stock
        self.qty = qty
        self.price = price
        self.brokerage = brokerage
        self.dividend = dividend
        self.remaining_qty = qty
```

- **Line 19**: Defines the `Trade` class.
- **Line 20**: The constructor for the class. It takes all the details of a single transaction as arguments.
- **Lines 21-27**: These lines initialize the attributes of the `Trade` object. `trade_type` is converted to uppercase to ensure consistency.
- **Line 28**: `remaining_qty` is initialized with the full quantity of the trade. This value will be decremented as the trade is matched in the FIFO process.

`MatchedTrade` Class

This class models a matched pair of a BUY and a SELL trade.

```python
class MatchedTrade:
    def __init__(self, buy_trade: Trade, sell_trade: Trade, matched_qty: int):
        self.buy_date = buy_trade.date
        self.sell_date = sell_trade.date
        # ... (other attributes)
```

- **Line 35**: Defines the `MatchedTrade` class.

- **Line 36**: The constructor takes a `buy_trade` object, a `sell_trade` object, and the quantity that has been matched between them.
- **Lines 37-43**: These lines store the basic details of the matched trade.
- **Line 44**: `buy_value` and `sell_value` are calculated by multiplying the price by the matched quantity.
- **Line 45**: `gain` is calculated as the difference between the sell value and the buy value, minus the brokerage fees.
- **Line 48**: `days_held` calculates the holding period of the investment.
- **Line 49**: `is_ltcg` is a boolean that is `True` if the holding period is more than 365 days.
- **Line 50**: `gain_type` is set to "LTCG" or "STCG" based on the `is_ltcg` flag.
- **Line 53**: `gst_on_brokerage` is calculated as 18% of the total brokerage.

`InvestorCalculator` Class

This class manages the overall calculation process.

`load_csv_data` method

- **Line 81**: This method reads the uploaded CSV file into a pandas DataFrame.
- **Lines 84-87**: It checks if all the required columns are present in the DataFrame.
- **Line 90**: It iterates through each row of the DataFrame.
- **Lines 92-102**: It extracts the data from each row, converts it to the correct data type, and creates a `Trade` object.
- **Line 103**: The newly created `Trade` object is appended to the `self.trades` list.

`calculate_fifo_matching` method

- **Lines 114-118**: The trades are grouped by stock into a dictionary.
- **Lines 121-122**: The trades for each stock are sorted by date.
- **Line 128**: A `buy_queue` is created to hold the BUY trades for the current stock.
- **Line 130**: The code iterates through the trades for the current stock.
- **Line 131**: If the trade is a BUY, it is added to the `buy_queue`.
- **Line 134**: If the trade is a SELL, the code enters a `while` loop to match it with the BUY trades in the `buy_queue`.
- **Line 139**: The `matched_qty` is the smaller of the remaining quantity of the BUY trade and the SELL trade.
- **Line 142**: A `MatchedTrade` object is created with the matched details.
- **Lines 146-147**: The `remaining_qty` of the BUY and SELL trades are updated.
- **Line 150**: If the `remaining_qty` of a BUY trade becomes zero, it is removed from the `buy_queue`.

`calculate_summary` method

- **Lines 160-163**: This method calculates the total STCG, LTCG, GST, and dividends by summing up the values from the `self.matched_trades` and `self.trades` lists.
- **Line 166**: The total brokerage is calculated.
- **Line 169**: The final taxable income is calculated.
- **Lines 171-181**: A dictionary is returned with all the summary calculations.

2. `app.py` - The User Interface

This file uses the Streamlit library to create the web application.

`main` function

- **Line 15**: `st.set_page_config` configures the page title, icon, and layout.
- **Lines 22-32**: `st.title` and `st.markdown` are used to display the main title and description of the application.
- **Line 35**: `with st.sidebar:` creates a sidebar for the file uploader and instructions.
- **Line 38**: `st.file_uploader` creates the file upload widget.
- **Line 70**: The code checks if a file has been uploaded.
- **Line 72**: An instance of `InvestorCalculator` is created.
- **Line 76**: `calculator.process_portfolio` is called to perform the calculations. This is wrapped in `st.spinner` to show a loading message.
- **Line 80**: If the calculations are successful, `st.success` displays a success message.
- **Lines 84-105**: `st.metric` is used to display the summary of the tax calculations in a visually appealing way.
- **Lines 111-128**: A pandas DataFrame is created to display the final tax calculation in a table.
- **Lines 130-136**: `st.info` is used to display additional details.
- **Lines 142-153**: `st.selectbox` and `st.number_input` are used to create filters for the detailed trade analysis table.
- **Lines 156-164**: The filters are applied to the results DataFrame.
- **Lines 167-172**: The filtered results are displayed in a table using `st.dataframe`.
- **Lines 178-199**: `st.download_button` creates buttons to download the detailed results and the summary as CSV files.
- **Lines 202-217**: `st.bar_chart` is used to create charts for portfolio insights.
- **Line 220**: If no trades could be matched, `st.warning` displays a warning message.
- **Line 223**: If an error occurs during processing, `st.error` displays an error message.

- **Lines 234-265**: If no file is uploaded, this section displays instructions and key features of the application.
- **Lines 268-274**: A footer is added to the page with a disclaimer.
- **Line 277**: `if __name__ == "__main__":` ensures that the `main()` function is called only when the script is executed directly.