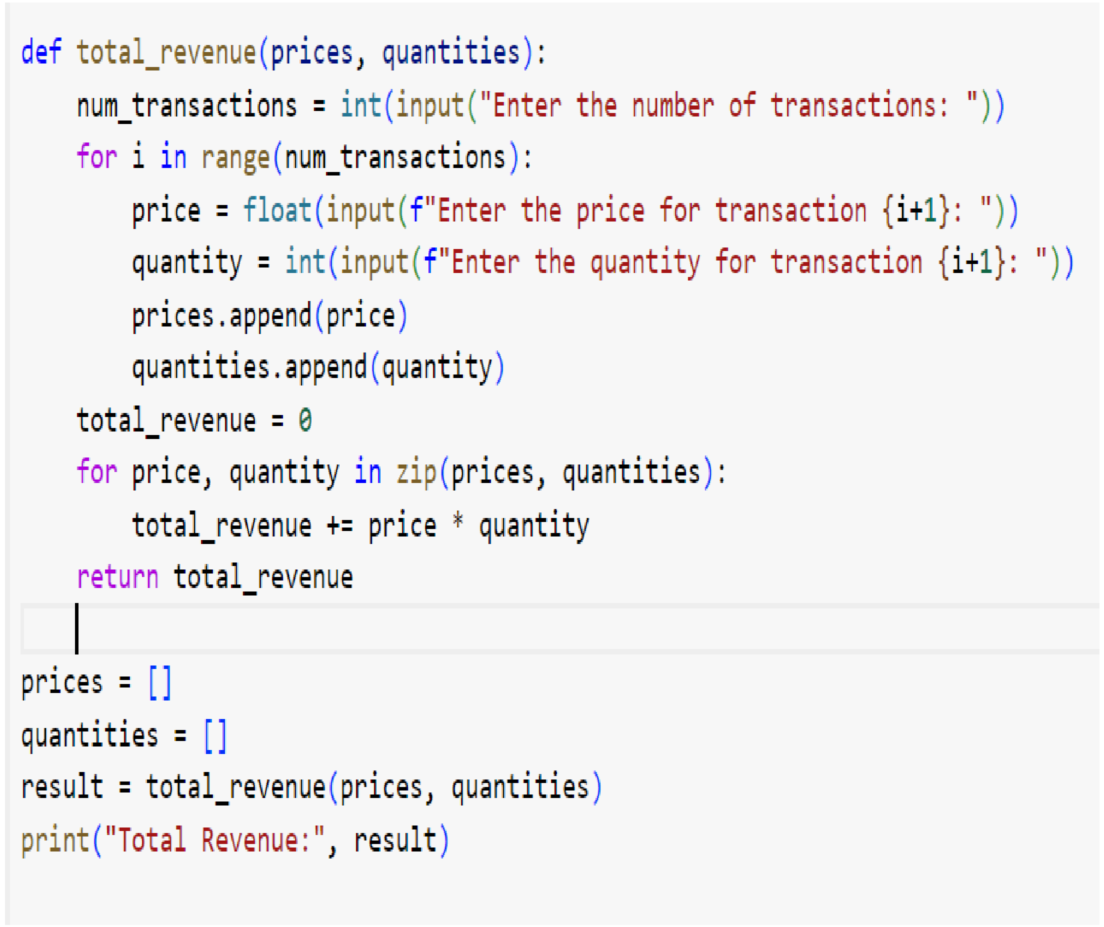
TOTAL REVENUE CALCULATION TOOL

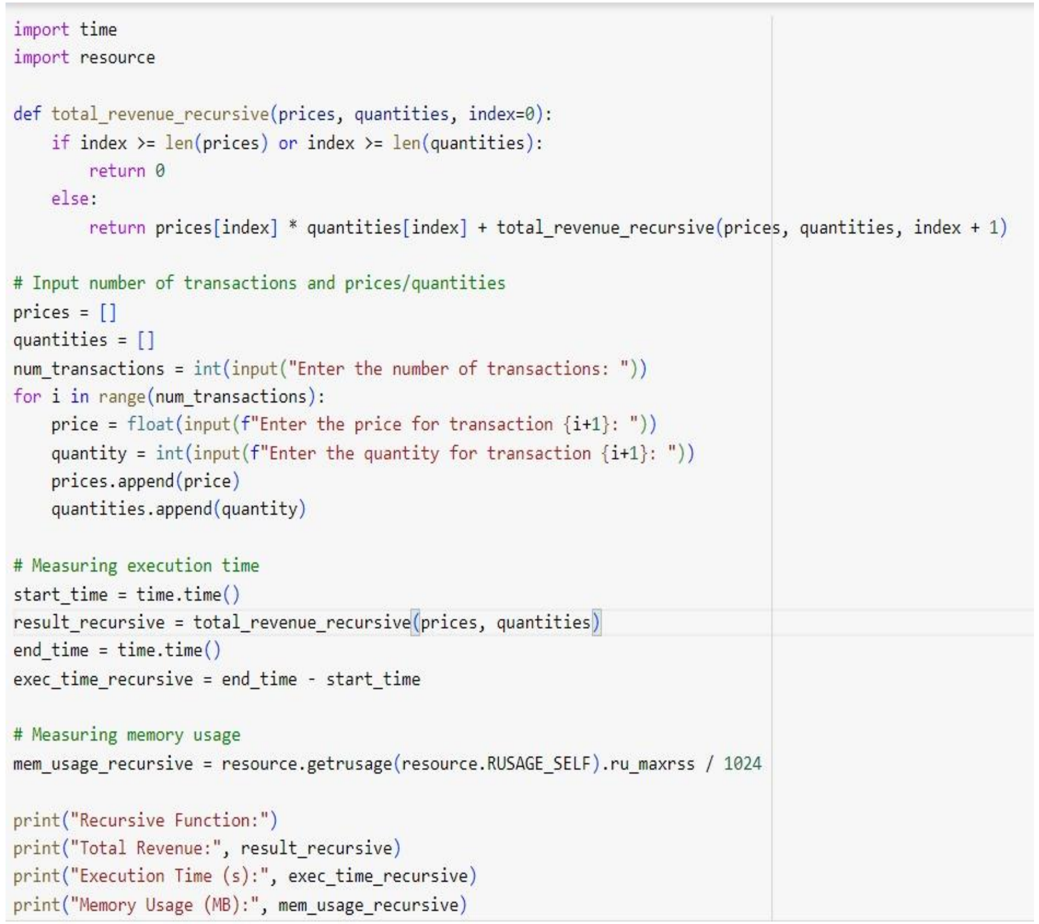
INTRODUCTION AND PURPOSE

This project proposes the development of a Sales Revenue Calculation Tool to automate the process of calculating total revenue generated from sales transactions with a reliable and efficient solution. By implementing both regular and recursive versions of the revenue calculation algorithm and offering a user-friendly interface, the tool will streamline the revenue calculation process and empower businesses to make data-driven decisions.

REGULAR VS RECURSIVE

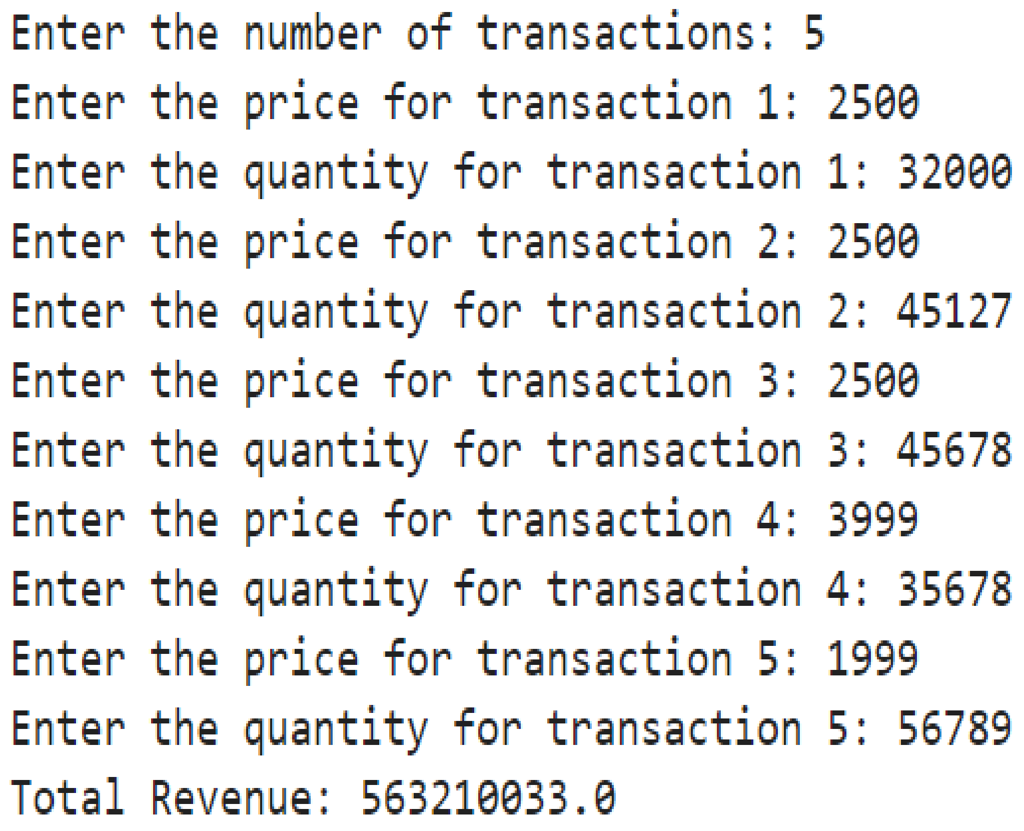


```python
def total_revenue(prices, quantities):
    num_transactions = int(input("Enter the number of transactions: "))
    for i in range(num_transactions):
        price = float(input(f"Enter the price for transaction {i+1}: "))
        quantity = int(input(f"Enter the quantity for transaction {i+1}: "))
        prices.append(price)
        quantities.append(quantity)
    total_revenue = 0
    for price, quantity in zip(prices, quantities):
        total_revenue += price * quantity
    return total_revenue

prices = []
quantities = []
result = total_revenue(prices, quantities)
print("Total Revenue:", result)
```

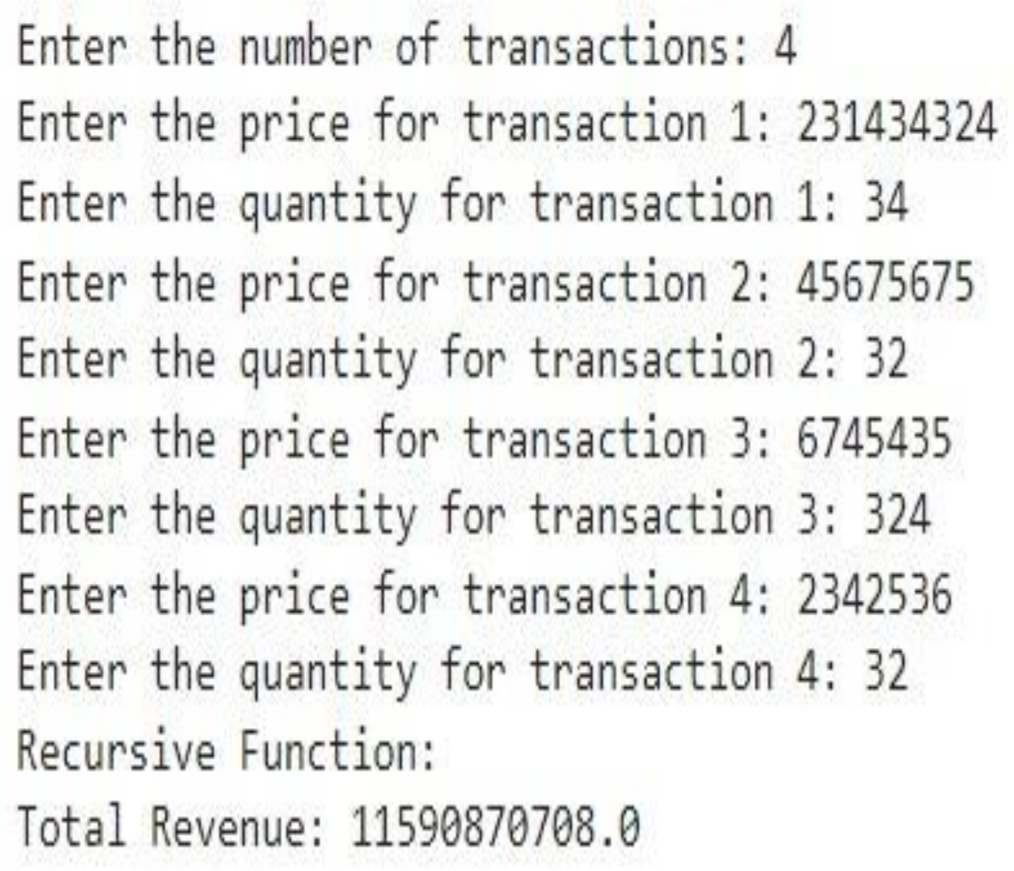


```python
import time
import resource

def total_revenue_recursive(prices, quantities, index=0):
    if index >= len(prices) or index >= len(quantities):
        return 0
    else:
        return prices[index] * quantities[index] + total_revenue_recursive(prices, quantities, index + 1)

# Input number of transactions and prices/quantities
prices = []
quantities = []
num_transactions = int(input("Enter the number of transactions: "))
for i in range(num_transactions):
    price = float(input(f"Enter the price for transaction {i+1}: "))
    quantity = int(input(f"Enter the quantity for transaction {i+1}: "))
    prices.append(price)
    quantities.append(quantity)

# Measuring execution time
start_time = time.time()
result_recursive = total_revenue_recursive(prices, quantities)
end_time = time.time()
exec_time_recursive = end_time - start_time

# Measuring memory usage
mem_usage_recursive = resource.getrusage(resource.RUSAGE_SELF).ru_maxrss / 1024

print("Recursive Function:")
print("Total Revenue:", result_recursive)
print("Execution Time (s):", exec_time_recursive)
print("Memory Usage (MB):", mem_usage_recursive)
```

TEST CASES USED AND THE RESULTS

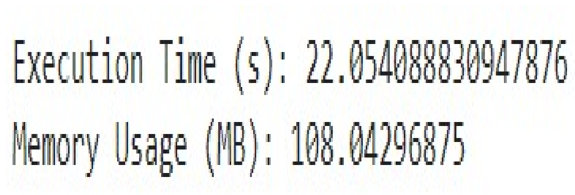


```
Enter the number of transactions: 5
Enter the price for transaction 1: 2500
Enter the quantity for transaction 1: 32000
Enter the price for transaction 2: 2500
Enter the quantity for transaction 2: 45127
Enter the price for transaction 3: 2500
Enter the quantity for transaction 3: 45678
Enter the price for transaction 4: 3999
Enter the quantity for transaction 4: 35678
Enter the price for transaction 5: 1999
Enter the quantity for transaction 5: 56789
Total Revenue: 563210033.0
```

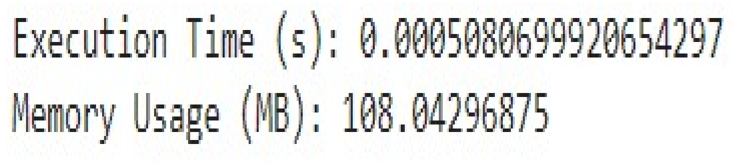


```
Enter the number of transactions: 4
Enter the price for transaction 1: 231434324
Enter the quantity for transaction 1: 34
Enter the price for transaction 2: 45675675
Enter the quantity for transaction 2: 32
Enter the price for transaction 3: 6745435
Enter the quantity for transaction 3: 324
Enter the price for transaction 4: 2342536
Enter the quantity for transaction 4: 32
Recursive Function:
Total Revenue: 11590870708.0
```

PERFORMANCE ANALYSIS



```
Execution Time (s): 22.054088830947876
Memory Usage (MB): 108.04296875
```



```
Execution Time (s): 0.0005080699920654297
Memory Usage (MB): 108.04296875
```

CONCLUSION & RECOMMENDATIONS

The choice of algorithm depends on the specific requirements and constraints of the problem. For large datasets or simple applications, recursive functions may suffice. Although the memory usage for both types of functions is the same, the execution time is much less when compared, which means that the recursive function is very fast and accurate in executing the program.