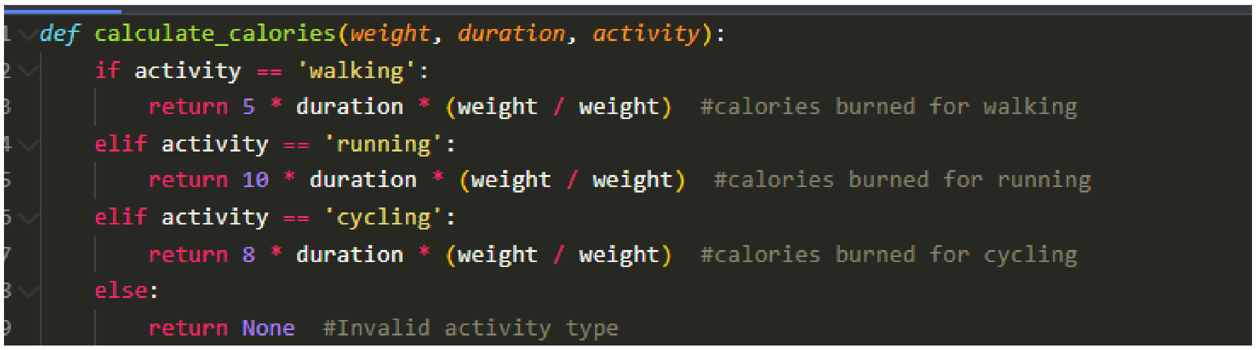
PROBLEM 1: Fitness Tracker

1. Create a function calculate_calories(activity, duration, weight) that calculates the calories burned for a given activity.

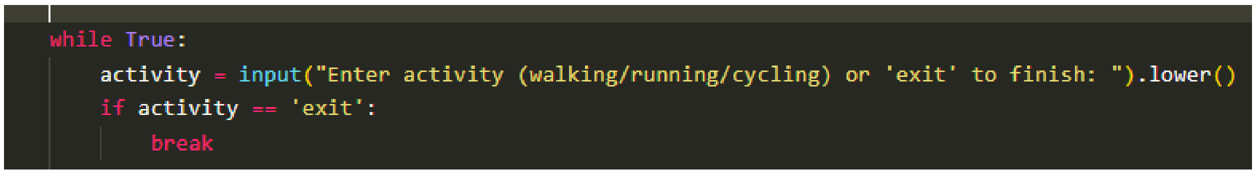
   It has 3 parameters to take into account the weight, duration, activity.



```python
def calculate_calories(weight, duration, activity):
    if activity == 'walking':
        return 5 * duration * (weight / weight)  #calories burned for walking
    elif activity == 'running':
        return 10 * duration * (weight / weight)  #calories burned for running
    elif activity == 'cycling':
        return 8 * duration * (weight / weight)  #calories burned for cycling
    else:
        return None  #Invalid activity type
```

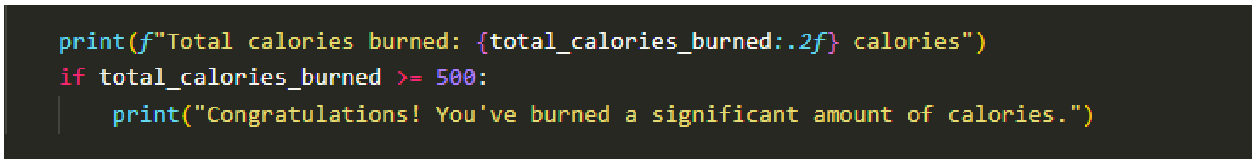
2. Use a loop to allow the user to input multiple activities for the same day.

   While loop was used.



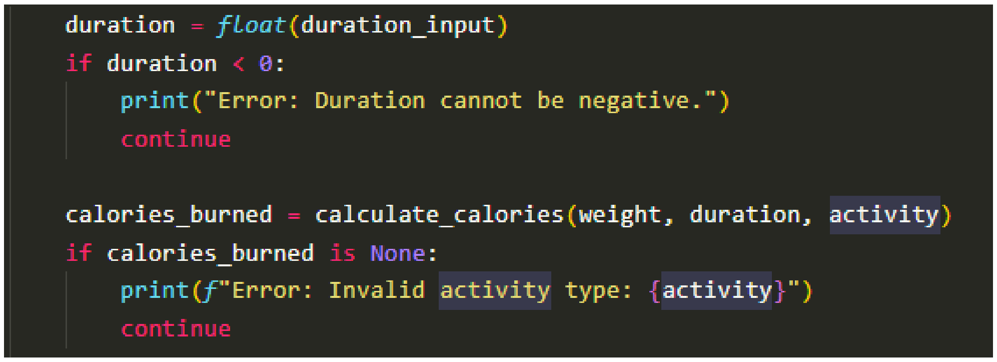
```python
while True:
    activity = input("Enter activity (walking/running/cycling) or 'exit' to finish: ").lower()
    if activity == 'exit':
        break
```

3. Use conditionals to check if the user has achieved their daily goal.



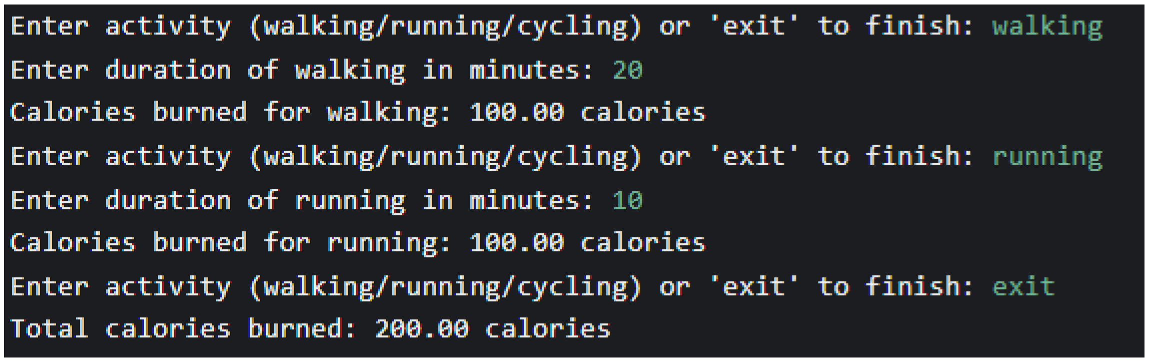
```python
print(f"Total calories burned: {total_calories_burned:.2f} calories")
if total_calories_burned >= 500:
    print("Congratulations! You've burned a significant amount of calories.")
```

4. Ensure proper error handling for invalid inputs (e.g., negative durations or weights).



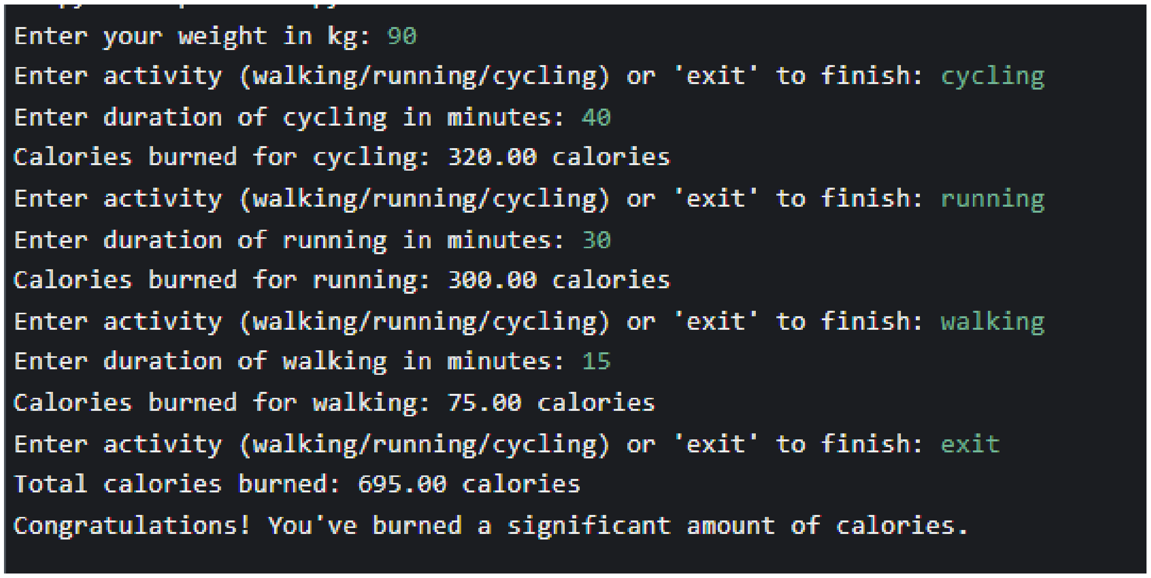
```python
duration = float(duration_input)
if duration < 0:
    print("Error: Duration cannot be negative.")
    continue

calories_burned = calculate_calories(weight, duration, activity)
if calories_burned is None:
    print(f"Error: Invalid activity type: {activity}")
    continue
```

Case 1: Valid Inputs



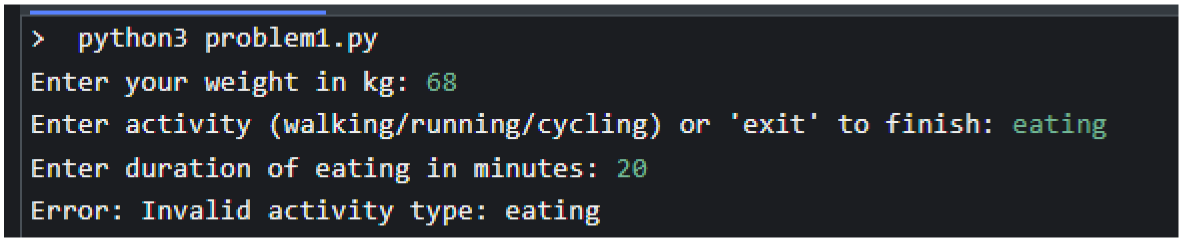
```
Enter activity (walking/running/cycling) or 'exit' to finish: walking
Enter duration of walking in minutes: 20
Calories burned for walking: 100.00 calories
Enter activity (walking/running/cycling) or 'exit' to finish: running
Enter duration of running in minutes: 10
Calories burned for running: 100.00 calories
Enter activity (walking/running/cycling) or 'exit' to finish: exit
Total calories burned: 200.00 calories
```

Case 2: Valid Inputs with Multiple Activities



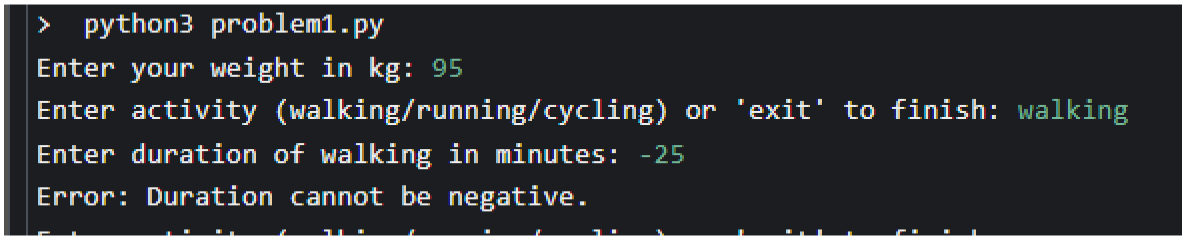
```
Enter your weight in kg: 90
Enter activity (walking/running/cycling) or 'exit' to finish: cycling
Enter duration of cycling in minutes: 40
Calories burned for cycling: 320.00 calories
Enter activity (walking/running/cycling) or 'exit' to finish: running
Enter duration of running in minutes: 30
Calories burned for running: 300.00 calories
Enter activity (walking/running/cycling) or 'exit' to finish: walking
Enter duration of walking in minutes: 15
Calories burned for walking: 75.00 calories
Enter activity (walking/running/cycling) or 'exit' to finish: exit
Total calories burned: 695.00 calories
Congratulations! You've burned a significant amount of calories.
```

Case 3: Invalid Activity Type



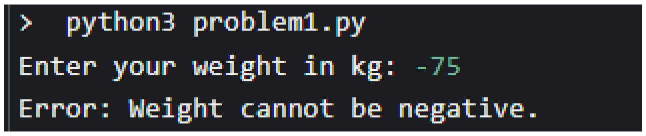
```
>  python3 problem1.py
Enter your weight in kg: 68
Enter activity (walking/running/cycling) or 'exit' to finish: eating
Enter duration of eating in minutes: 20
Error: Invalid activity type: eating
```

Case 4: Negative Duration



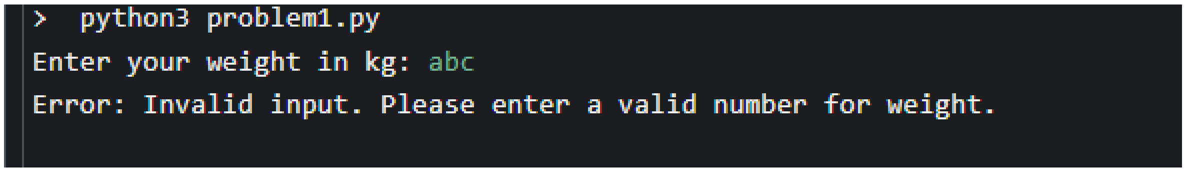
```
>  python3 problem1.py
Enter your weight in kg: 95
Enter activity (walking/running/cycling) or 'exit' to finish: walking
Enter duration of walking in minutes: -25
Error: Duration cannot be negative.
```

Case 5: Negative Weight



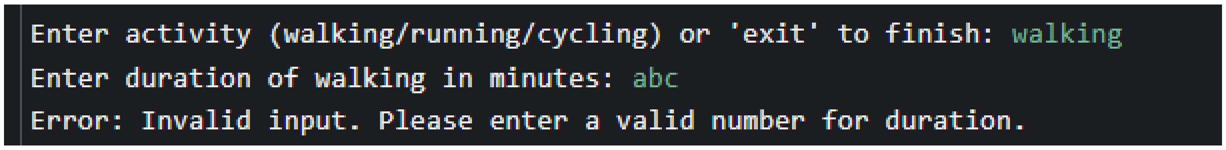
```
>  python3 problem1.py
Enter your weight in kg: -75
Error: Weight cannot be negative.
```

Case 6: Non numerical input for weight



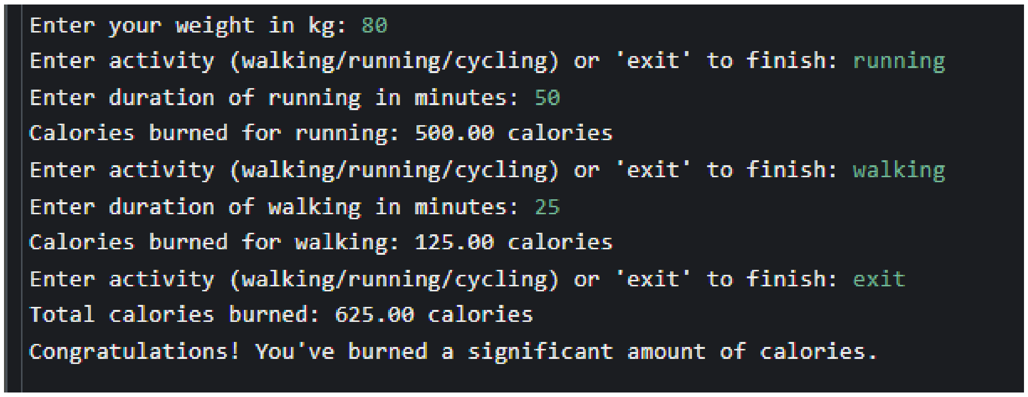
```
>  python3 problem1.py
Enter your weight in kg: abc
Error: Invalid input. Please enter a valid number for weight.
```

Case 7: Non numerical input for duration



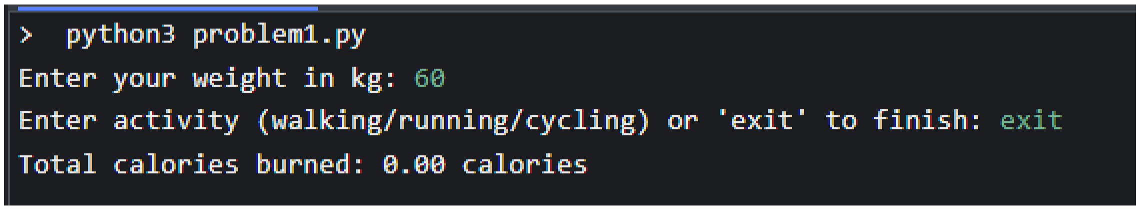
```
Enter activity (walking/running/cycling) or 'exit' to finish: walking
Enter duration of walking in minutes: abc
Error: Invalid input. Please enter a valid number for duration.
```

Case 8: Valid inputs with exact 500 calories



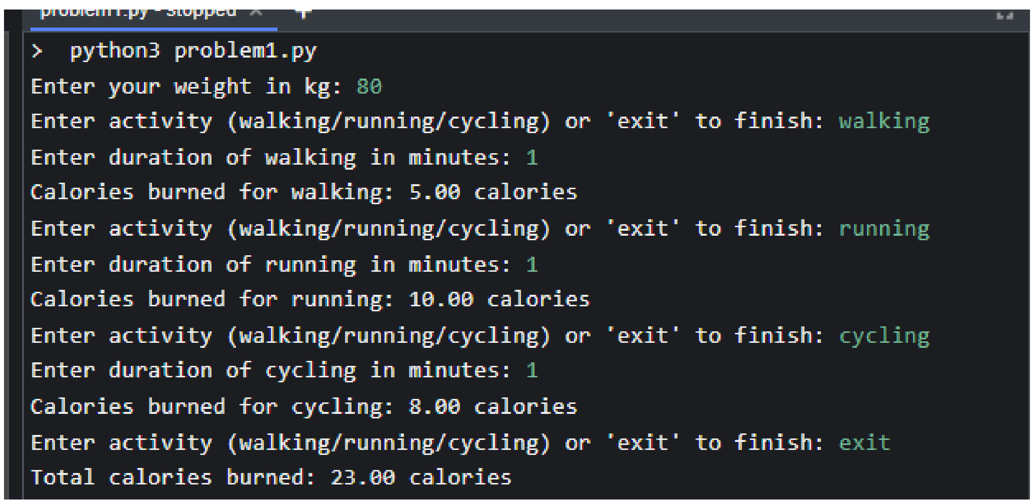
```
Enter your weight in kg: 80
Enter activity (walking/running/cycling) or 'exit' to finish: running
Enter duration of running in minutes: 50
Calories burned for running: 500.00 calories
Enter activity (walking/running/cycling) or 'exit' to finish: walking
Enter duration of walking in minutes: 25
Calories burned for walking: 125.00 calories
Enter activity (walking/running/cycling) or 'exit' to finish: exit
Total calories burned: 625.00 calories
Congratulations! You've burned a significant amount of calories.
```

Case 9: No activities Needed



```
>  python3 problem1.py
Enter your weight in kg: 60
Enter activity (walking/running/cycling) or 'exit' to finish: exit
Total calories burned: 0.00 calories
```

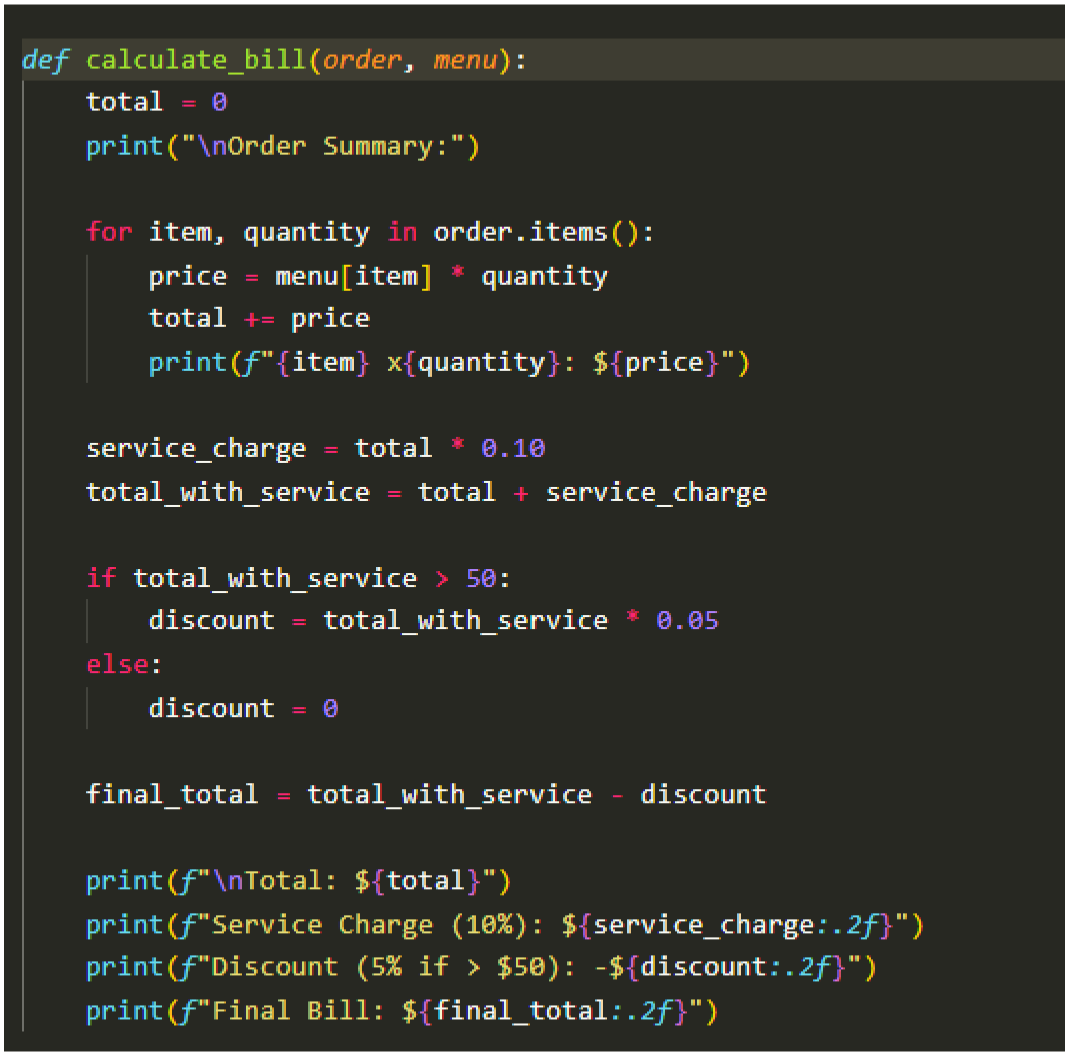
Case 10: Edge case with minimum duration



```
>  python3 problem1.py
Enter your weight in kg: 80
Enter activity (walking/running/cycling) or 'exit' to finish: walking
Enter duration of walking in minutes: 1
Calories burned for walking: 5.00 calories
Enter activity (walking/running/cycling) or 'exit' to finish: running
Enter duration of running in minutes: 1
Calories burned for running: 10.00 calories
Enter activity (walking/running/cycling) or 'exit' to finish: cycling
Enter duration of cycling in minutes: 1
Calories burned for cycling: 8.00 calories
Enter activity (walking/running/cycling) or 'exit' to finish: exit
Total calories burned: 23.00 calories
```

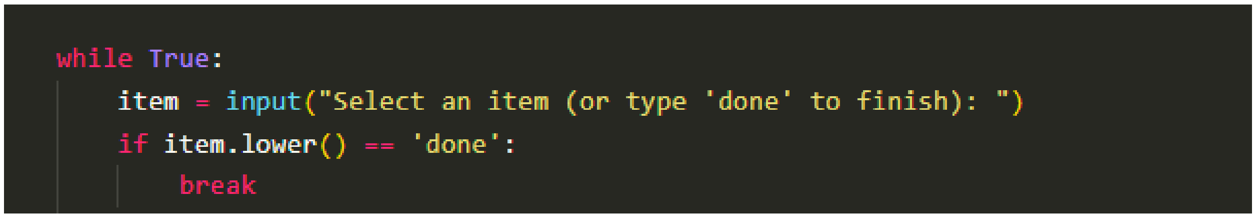
Problem 4: Restaurant Order Management

1. Create a function calculate_bill(order) that computes the total bill based on the user's order



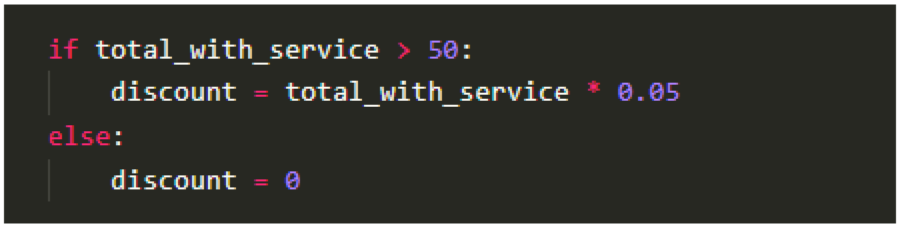
```python
def calculate_bill(order, menu):
    total = 0
    print("\nOrder Summary:")

    for item, quantity in order.items():
        price = menu[item] * quantity
        total += price
        print(f"{item} x{quantity}: ${price}")

    service_charge = total * 0.10
    total_with_service = total + service_charge

    if total_with_service > 50:
        discount = total_with_service * 0.05
    else:
        discount = 0

    final_total = total_with_service - discount

    print(f"\nTotal: ${total}")
    print(f"Service Charge (10%): ${service_charge:.2f}")
    print(f"Discount (5% if > $50): -${discount:.2f}")
    print(f"Final Bill: ${final_total:.2f}")
```

2. Use a loop to allow the user to add multiple items to their order.



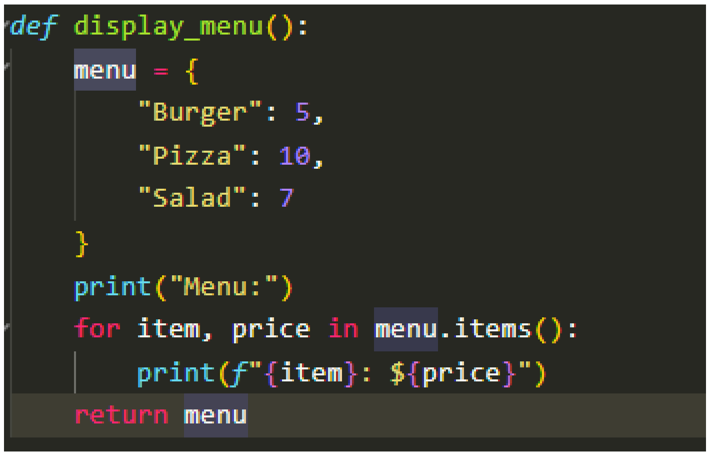
```python
while True:
    item = input("Select an item (or type 'done' to finish): ")
    if item.lower() == 'done':
        break
```

3. Use conditionals to apply discounts and calculate the final amount.



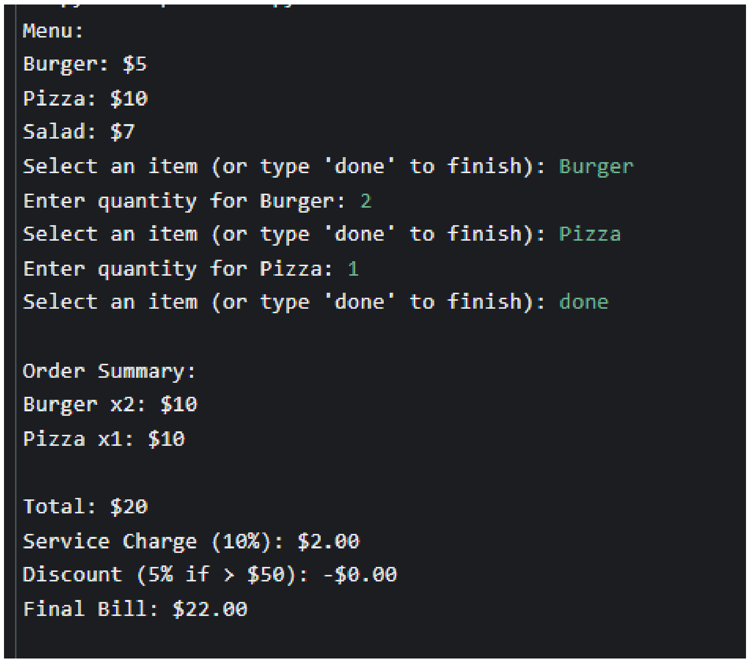
```python
if total_with_service > 50:
    discount = total_with_service * 0.05
else:
    discount = 0
```

4. Display the bill in clear format, showing each item, quantity, price, and the total.



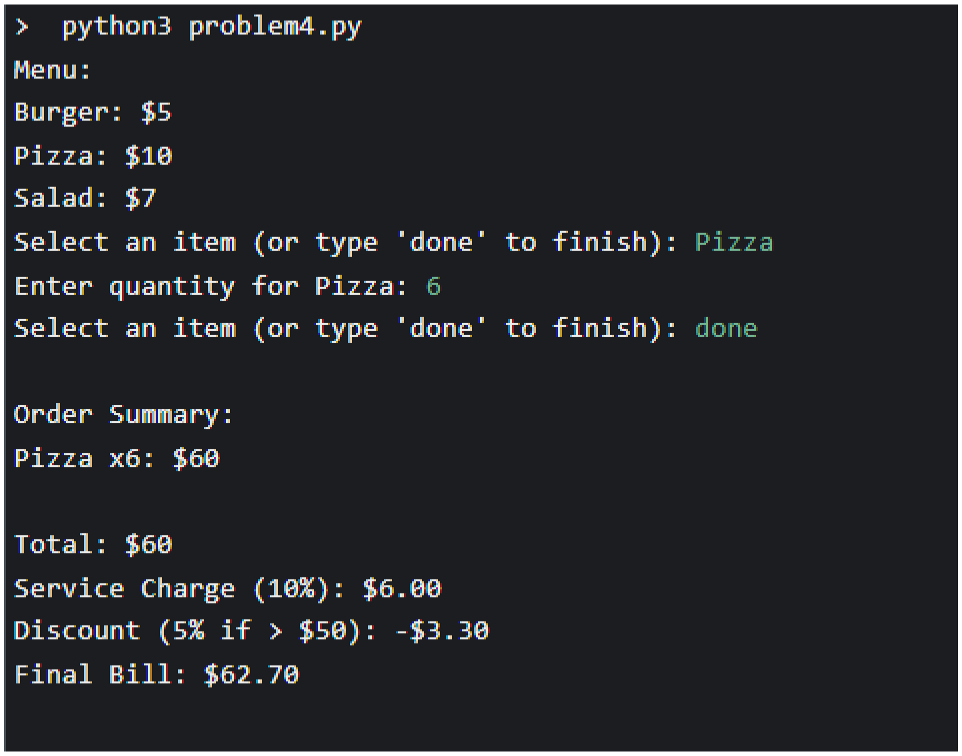
```python
def display_menu():
    menu = {
        "Burger": 5,
        "Pizza": 10,
        "Salad": 7
    }
    print("Menu:")
    for item, price in menu.items():
        print(f"{item}: ${price}")
    return menu
```

Case 1: Basic Ordering



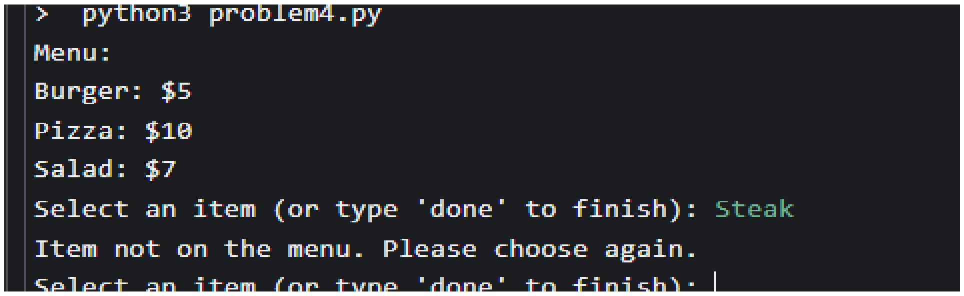
```
Menu:
Burger: $5
Pizza: $10
Salad: $7
Select an item (or type 'done' to finish): Burger
Enter quantity for Burger: 2
Select an item (or type 'done' to finish): Pizza
Enter quantity for Pizza: 1
Select an item (or type 'done' to finish): done

Order Summary:
Burger x2: $10
Pizza x1: $10

Total: $20
Service Charge (10%): $2.00
Discount (5% if > $50): -$0.00
Final Bill: $22.00
```

Case 2: Order Exceeding 50$



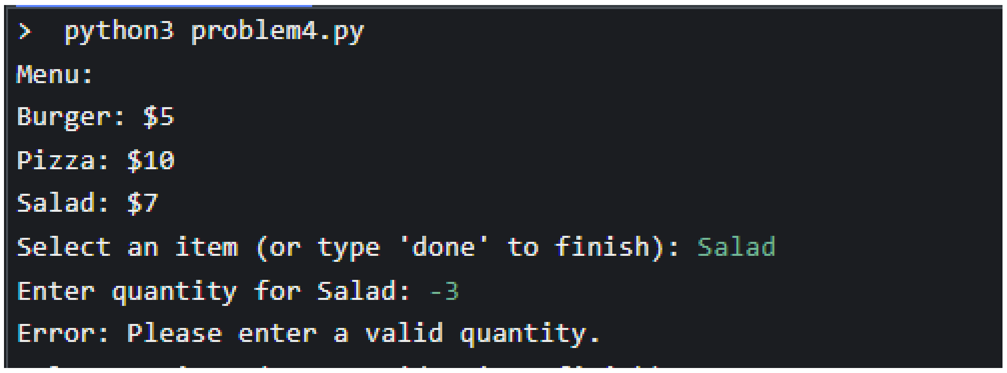
```
>  python3 problem4.py
Menu:
Burger: $5
Pizza: $10
Salad: $7
Select an item (or type 'done' to finish): Pizza
Enter quantity for Pizza: 6
Select an item (or type 'done' to finish): done

Order Summary:
Pizza x6: $60

Total: $60
Service Charge (10%): $6.00
Discount (5% if > $50): -$3.30
Final Bill: $62.70
```

Case 3: Invalid Item



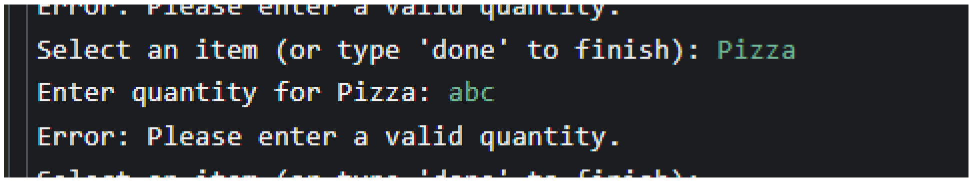
```
>  python3 problem4.py
Menu:
Burger: $5
Pizza: $10
Salad: $7
Select an item (or type 'done' to finish): Steak
Item not on the menu. Please choose again.
Select an item (or type 'done' to finish):
```

Case 4: Negative Quantity



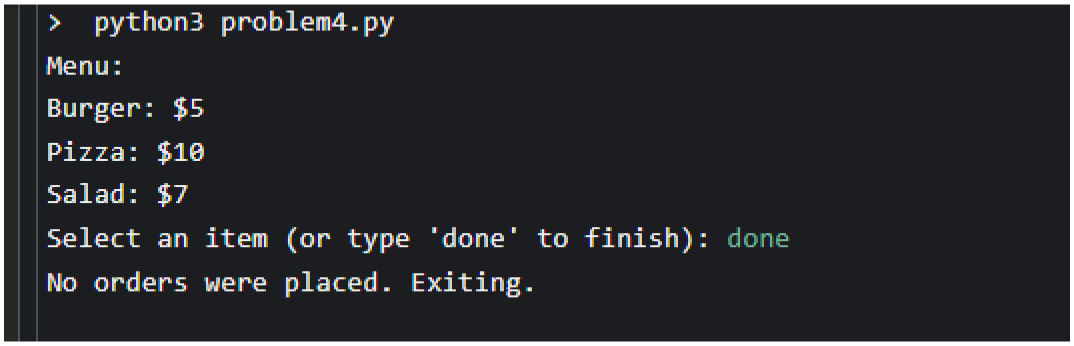
```
>  python3 problem4.py
Menu:
Burger: $5
Pizza: $10
Salad: $7
Select an item (or type 'done' to finish): Salad
Enter quantity for Salad: -3
Error: Please enter a valid quantity.
```

Case 5: Non-numeric



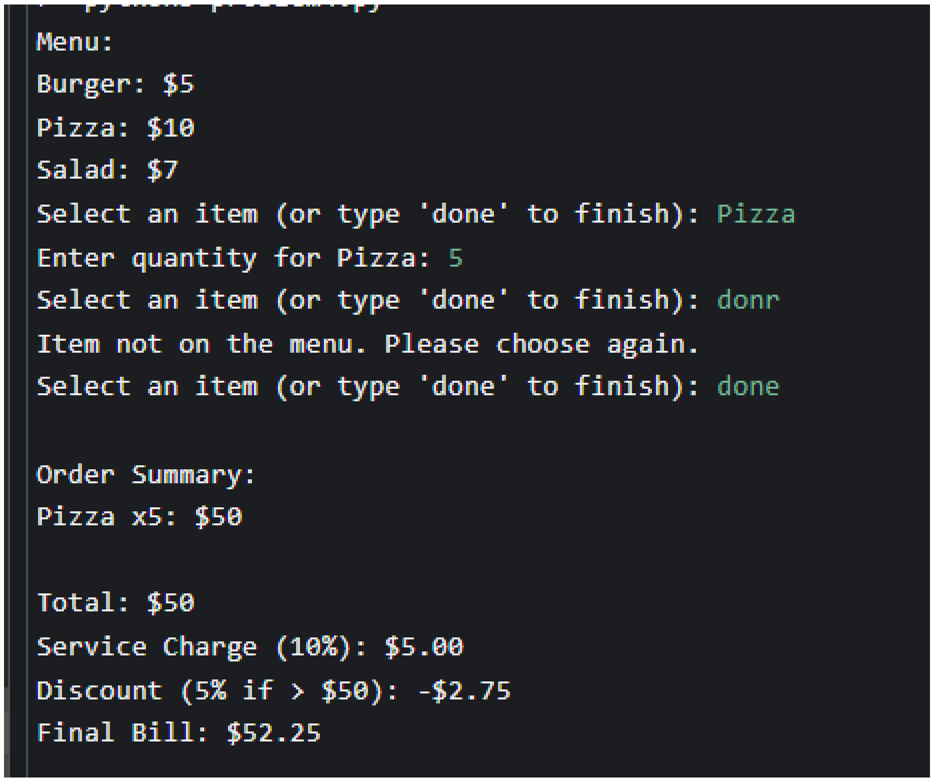
```
Error: Please enter a valid quantity.
Select an item (or type 'done' to finish): Pizza
Enter quantity for Pizza: abc
Error: Please enter a valid quantity.
Select an item (or type 'done' to finish):
```

Case 6: No Orders



```
>  python3 problem4.py
Menu:
Burger: $5
Pizza: $10
Salad: $7
Select an item (or type 'done' to finish): done
No orders were placed. Exiting.
```

Case 7: Edge Case with Exact 50$



```
>  python3 problem4.py
Menu:
Burger: $5
Pizza: $10
Salad: $7
Select an item (or type 'done' to finish): Pizza
Enter quantity for Pizza: 5
Select an item (or type 'done' to finish): donr
Item not on the menu. Please choose again.
Select an item (or type 'done' to finish): done

Order Summary:
Pizza x5: $50

Total: $50
Service Charge (10%): $5.00
Discount (5% if > $50): -$2.75
Final Bill: $52.25
```