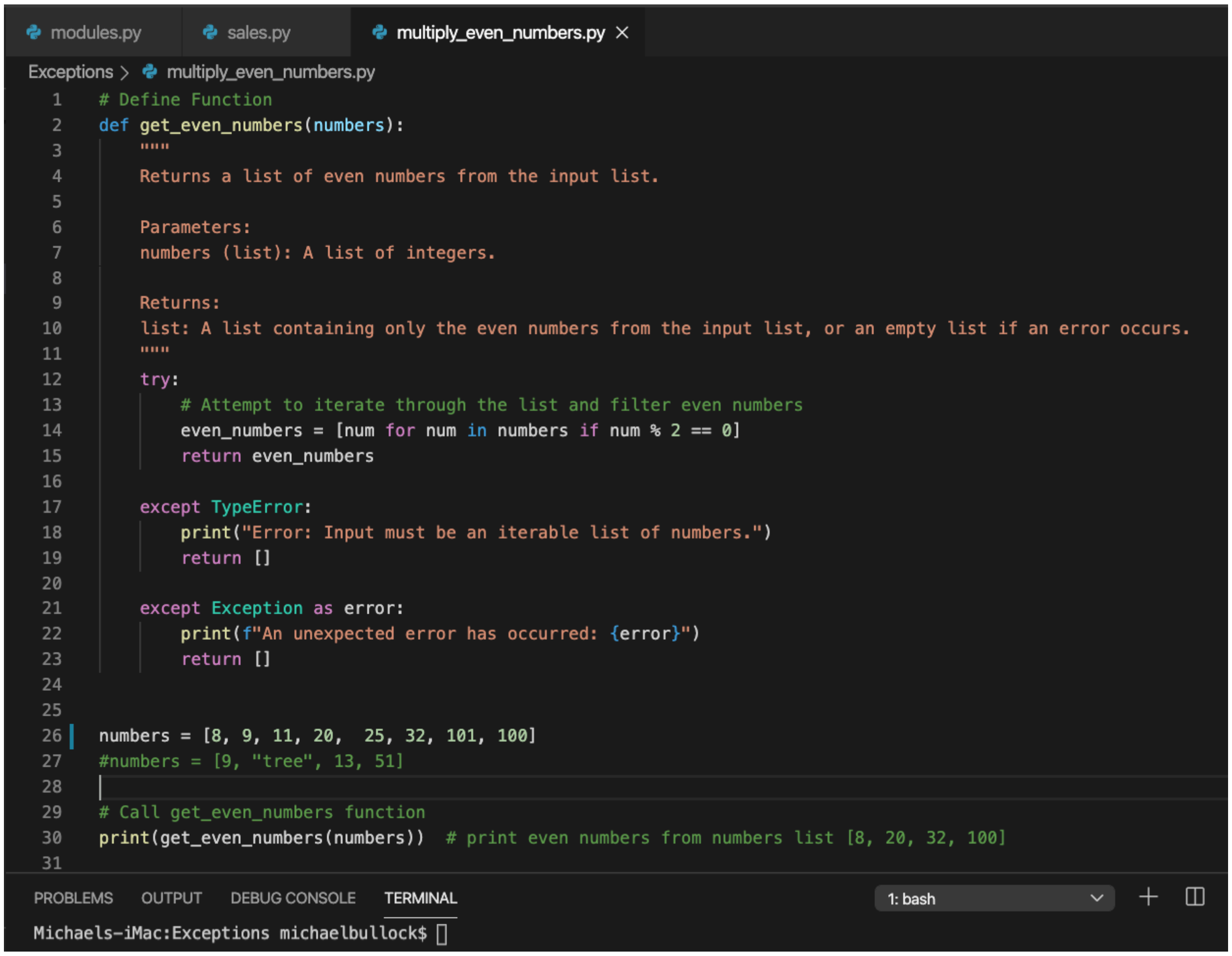
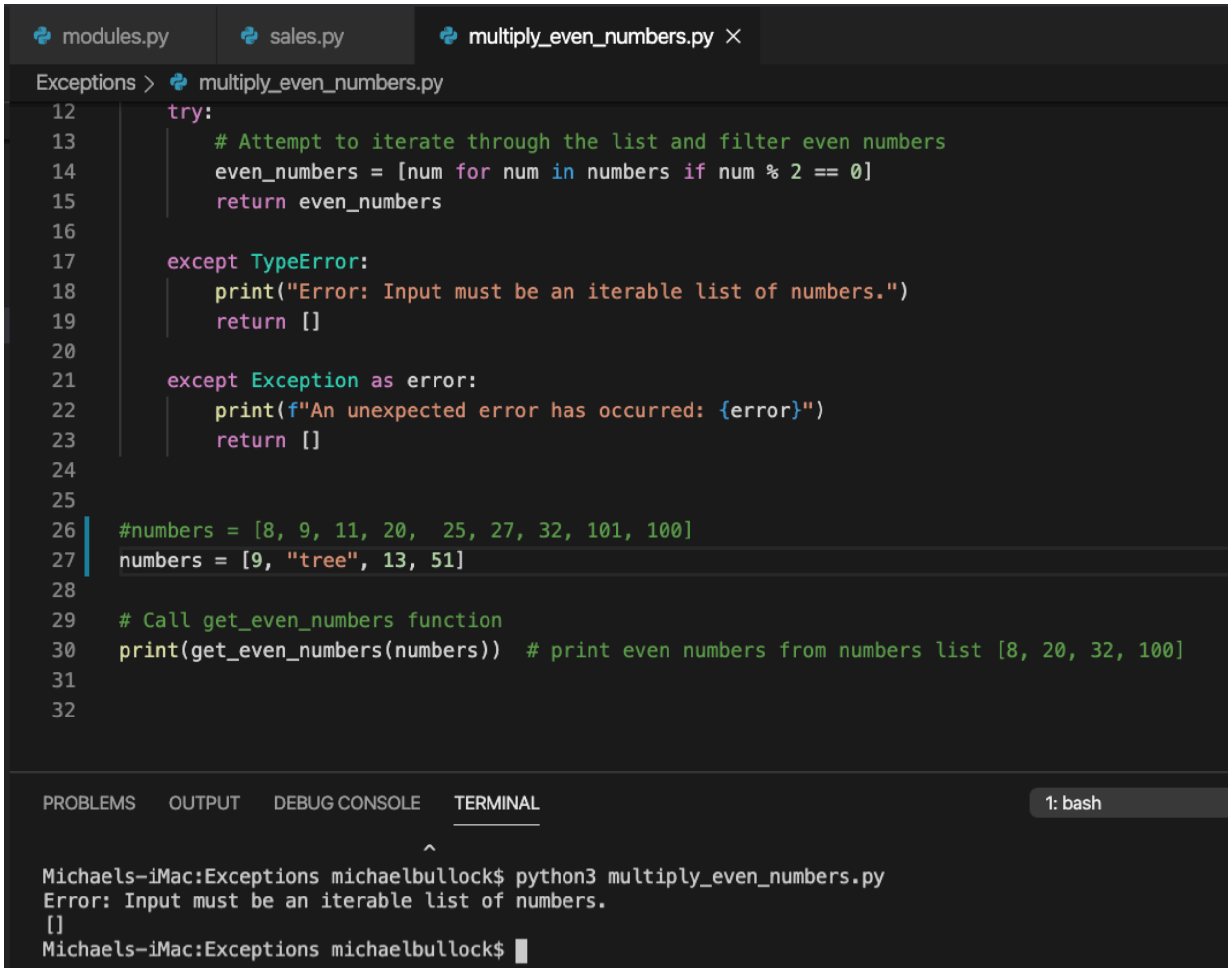
Michael Bullock
Python on AWS
Exceptions



```python
# Define Function
def get_even_numbers(numbers):
    """
    Returns a list of even numbers from the input list.

    Parameters:
    numbers (list): A list of integers.

    Returns:
    list: A list containing only the even numbers from the input list, or an empty list if an error occurs.
    """
    try:
        # Attempt to iterate through the list and filter even numbers
        even_numbers = [num for num in numbers if num % 2 == 0]
        return even_numbers

    except TypeError:
        print("Error: Input must be an iterable list of numbers.")
        return []

    except Exception as error:
        print(f"An unexpected error has occurred: {error}")
        return []


numbers = [8, 9, 11, 20,  25, 32, 101, 100]
#numbers = [9, "tree", 13, 51]

# Call get_even_numbers function
print(get_even_numbers(numbers))  # print even numbers from numbers list [8, 20, 32, 100]

```

```
Michaels-iMac:Exceptions michaelbullock$ 
```

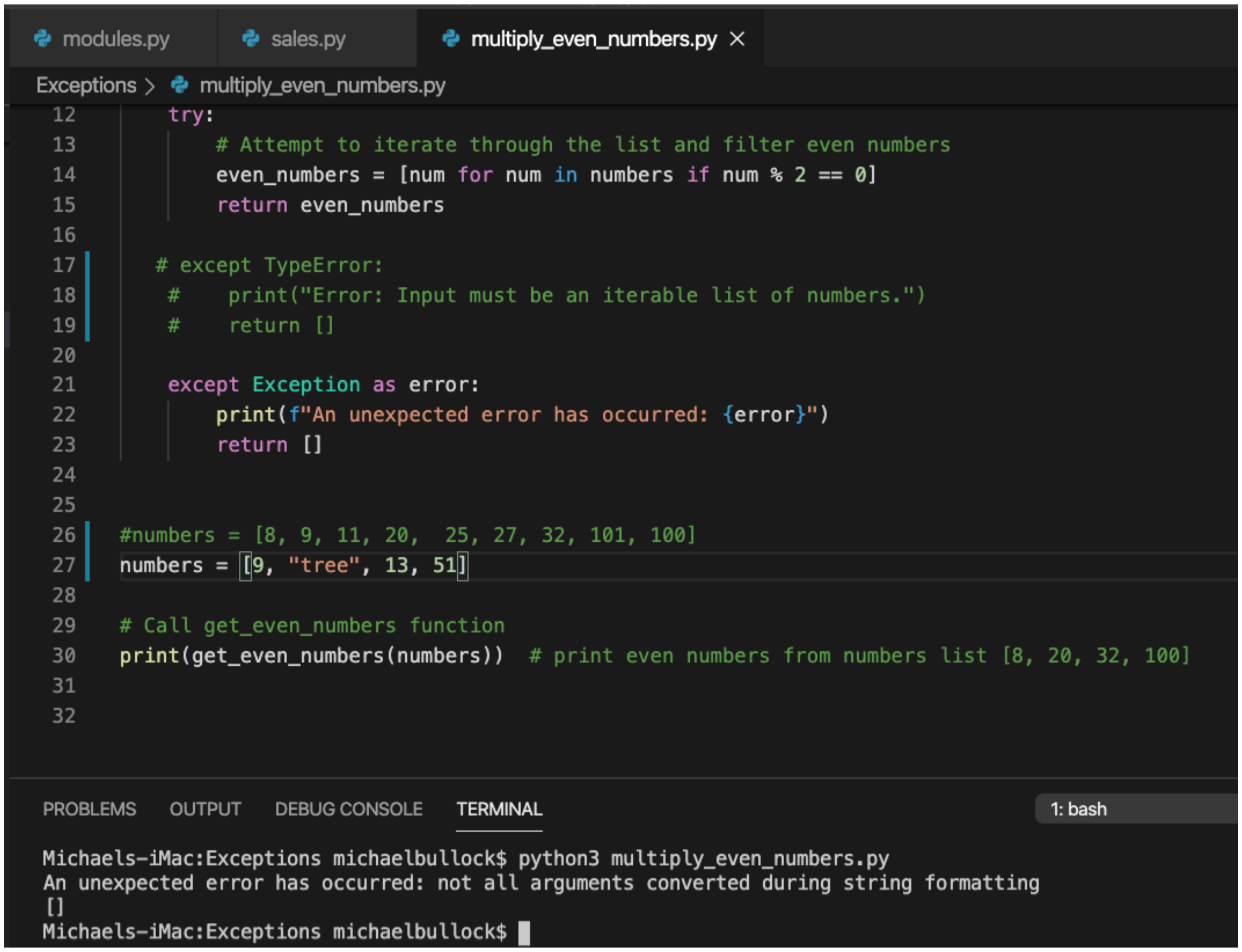
TypeError Example:



```python
    try:
        # Attempt to iterate through the list and filter even numbers
        even_numbers = [num for num in numbers if num % 2 == 0]
        return even_numbers

    except TypeError:
        print("Error: Input must be an iterable list of numbers.")
        return []

    except Exception as error:
        print(f"An unexpected error has occurred: {error}")
        return []


#numbers = [8, 9, 11, 20,  25, 27, 32, 101, 100]
numbers = [9, "tree", 13, 51]

# Call get_even_numbers function
print(get_even_numbers(numbers))  # print even numbers from numbers list [8, 20, 32, 100]


```

```
Michaels-iMac:Exceptions michaelbullock$ python3 multiply_even_numbers.py
Error: Input must be an iterable list of numbers.
[]
Michaels-iMac:Exceptions michaelbullock$ 
```

Exception Error Example:



modules.py | sales.py | multiply_even_numbers.py

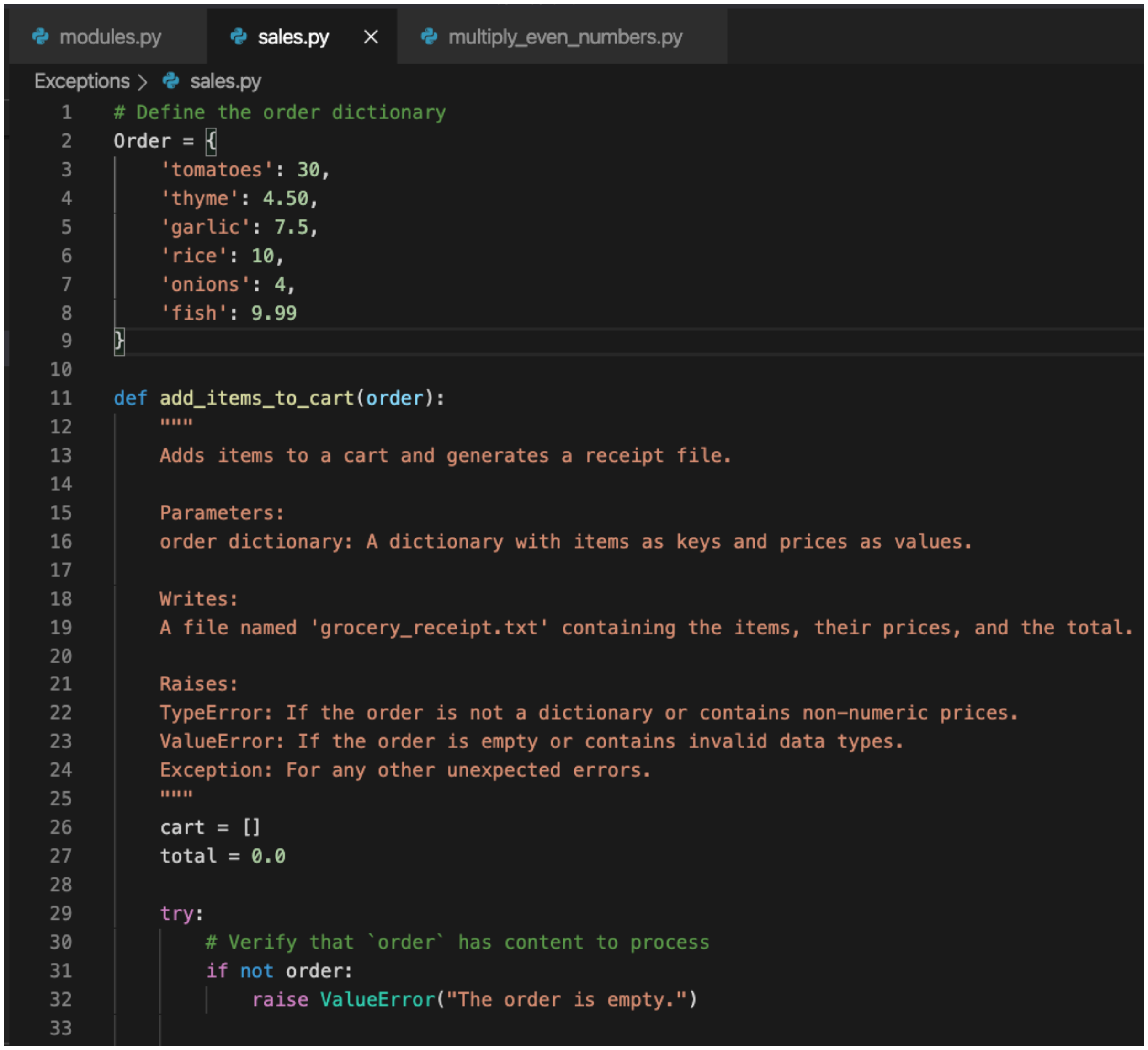
Exceptions > multiply_even_numbers.py

```python
    try:
        # Attempt to iterate through the list and filter even numbers
        even_numbers = [num for num in numbers if num % 2 == 0]
        return even_numbers

   # except TypeError:
    #    print("Error: Input must be an iterable list of numbers.")
    #    return []

    except Exception as error:
        print(f"An unexpected error has occurred: {error}")
        return []


#numbers = [8, 9, 11, 20,  25, 27, 32, 101, 100]
numbers = [9, "tree", 13, 51]

# Call get_even_numbers function
print(get_even_numbers(numbers))  # print even numbers from numbers list [8, 20, 32, 100]
```

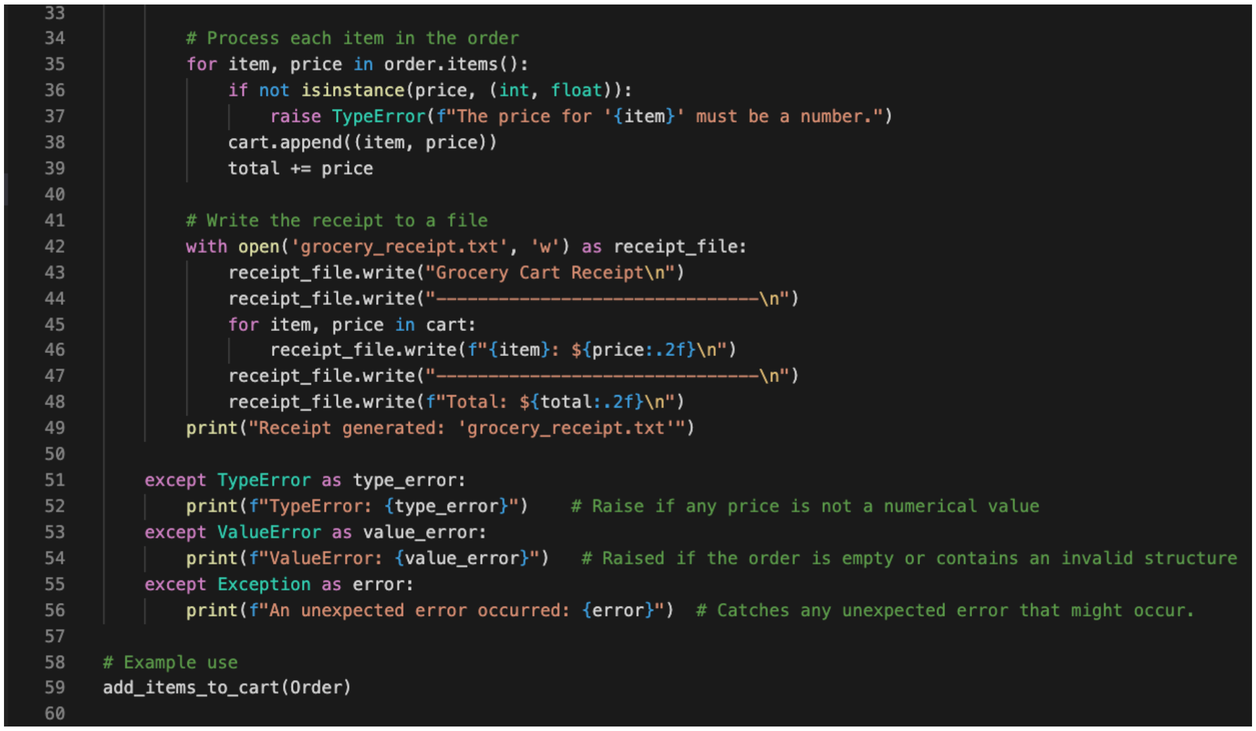
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: bash

```
Michaels-iMac:Exceptions michaelbullock$ python3 multiply_even_numbers.py
An unexpected error has occurred: not all arguments converted during string formatting
[]
Michaels-iMac:Exceptions michaelbullock$ 
```

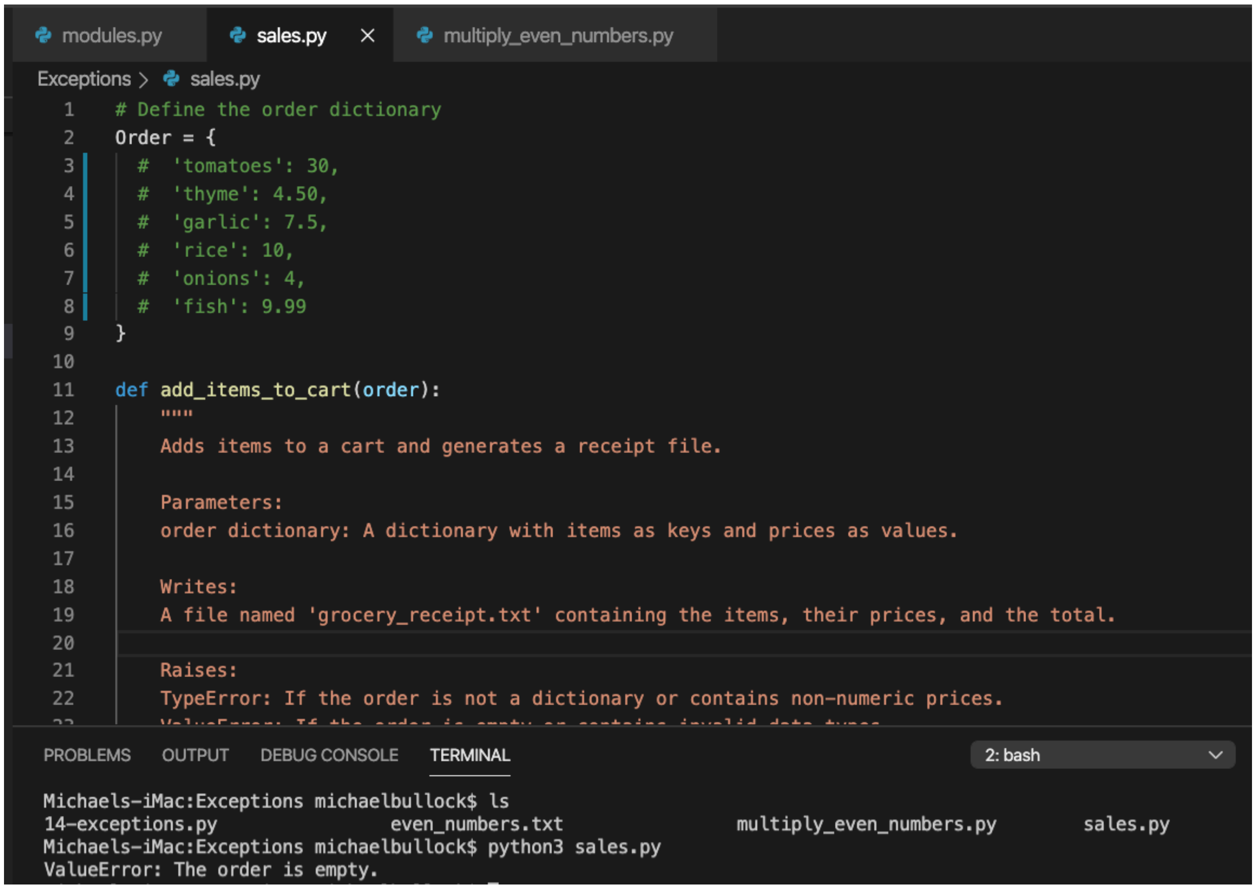
Exceptions: sales.py



```python
# Define the order dictionary
Order = {
    'tomatoes': 30,
    'thyme': 4.50,
    'garlic': 7.5,
    'rice': 10,
    'onions': 4,
    'fish': 9.99
}

def add_items_to_cart(order):
    """
    Adds items to a cart and generates a receipt file.

    Parameters:
    order dictionary: A dictionary with items as keys and prices as values.

    Writes:
    A file named 'grocery_receipt.txt' containing the items, their prices, and the total.

    Raises:
    TypeError: If the order is not a dictionary or contains non-numeric prices.
    ValueError: If the order is empty or contains invalid data types.
    Exception: For any other unexpected errors.
    """
    cart = []
    total = 0.0

    try:
        # Verify that `order` has content to process
        if not order:
            raise ValueError("The order is empty.")

```


```python

        # Process each item in the order
        for item, price in order.items():
            if not isinstance(price, (int, float)):
                raise TypeError(f"The price for '{item}' must be a number.")
            cart.append((item, price))
            total += price

        # Write the receipt to a file
        with open('grocery_receipt.txt', 'w') as receipt_file:
            receipt_file.write("Grocery Cart Receipt\n")
            receipt_file.write("-------------------------------\n")
            for item, price in cart:
                receipt_file.write(f"{item}: ${price:.2f}\n")
            receipt_file.write("-------------------------------\n")
            receipt_file.write(f"Total: ${total:.2f}\n")
        print("Receipt generated: 'grocery_receipt.txt'")

    except TypeError as type_error:
        print(f"TypeError: {type_error}")    # Raise if any price is not a numerical value
    except ValueError as value_error:
        print(f"ValueError: {value_error}")   # Raised if the order is empty or contains an invalid structure
    except Exception as error:
        print(f"An unexpected error occurred: {error}")  # Catches any unexpected error that might occur.

# Example use
add_items_to_cart(Order)

```

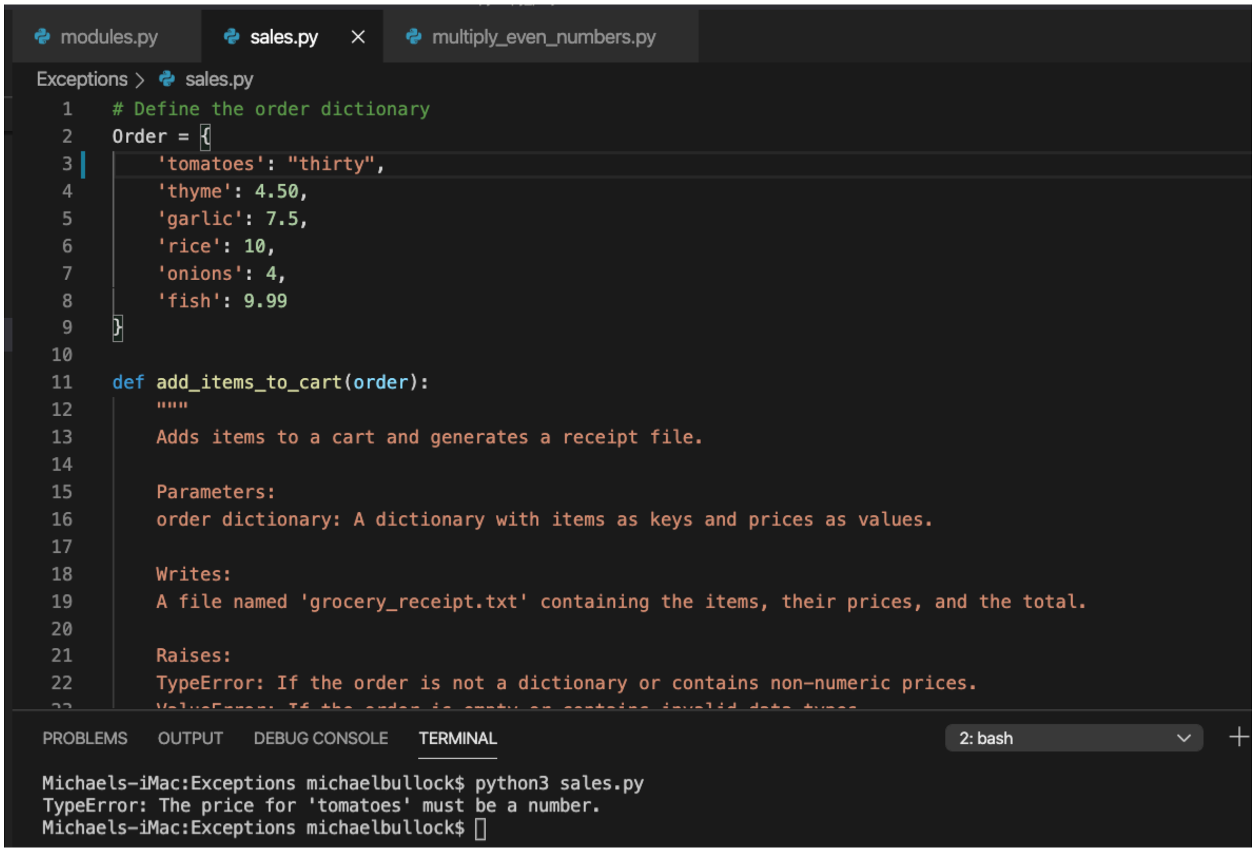
Value Error example:



```python
# Define the order dictionary
Order = {
  #  'tomatoes': 30,
  #  'thyme': 4.50,
  #  'garlic': 7.5,
  #  'rice': 10,
  #  'onions': 4,
  #  'fish': 9.99
}

def add_items_to_cart(order):
    """
    Adds items to a cart and generates a receipt file.

    Parameters:
    order dictionary: A dictionary with items as keys and prices as values.

    Writes:
    A file named 'grocery_receipt.txt' containing the items, their prices, and the total.

    Raises:
    TypeError: If the order is not a dictionary or contains non-numeric prices.
```

```
Michaels-iMac:Exceptions michaelbullock$ ls
14-exceptions.py              even_numbers.txt              multiply_even_numbers.py      sales.py
Michaels-iMac:Exceptions michaelbullock$ python3 sales.py
ValueError: The order is empty.
```

Type Error Example:



```python
# Define the order dictionary
Order = {
    'tomatoes': "thirty",
    'thyme': 4.50,
    'garlic': 7.5,
    'rice': 10,
    'onions': 4,
    'fish': 9.99
}

def add_items_to_cart(order):
    """
    Adds items to a cart and generates a receipt file.

    Parameters:
    order dictionary: A dictionary with items as keys and prices as values.

    Writes:
    A file named 'grocery_receipt.txt' containing the items, their prices, and the total.

    Raises:
    TypeError: If the order is not a dictionary or contains non-numeric prices.
    ValueError: If the order is empty or contains invalid data types.
```

```
Michaels-iMac:Exceptions michaelbullock$ python3 sales.py
TypeError: The price for 'tomatoes' must be a number.
Michaels-iMac:Exceptions michaelbullock$ 
```