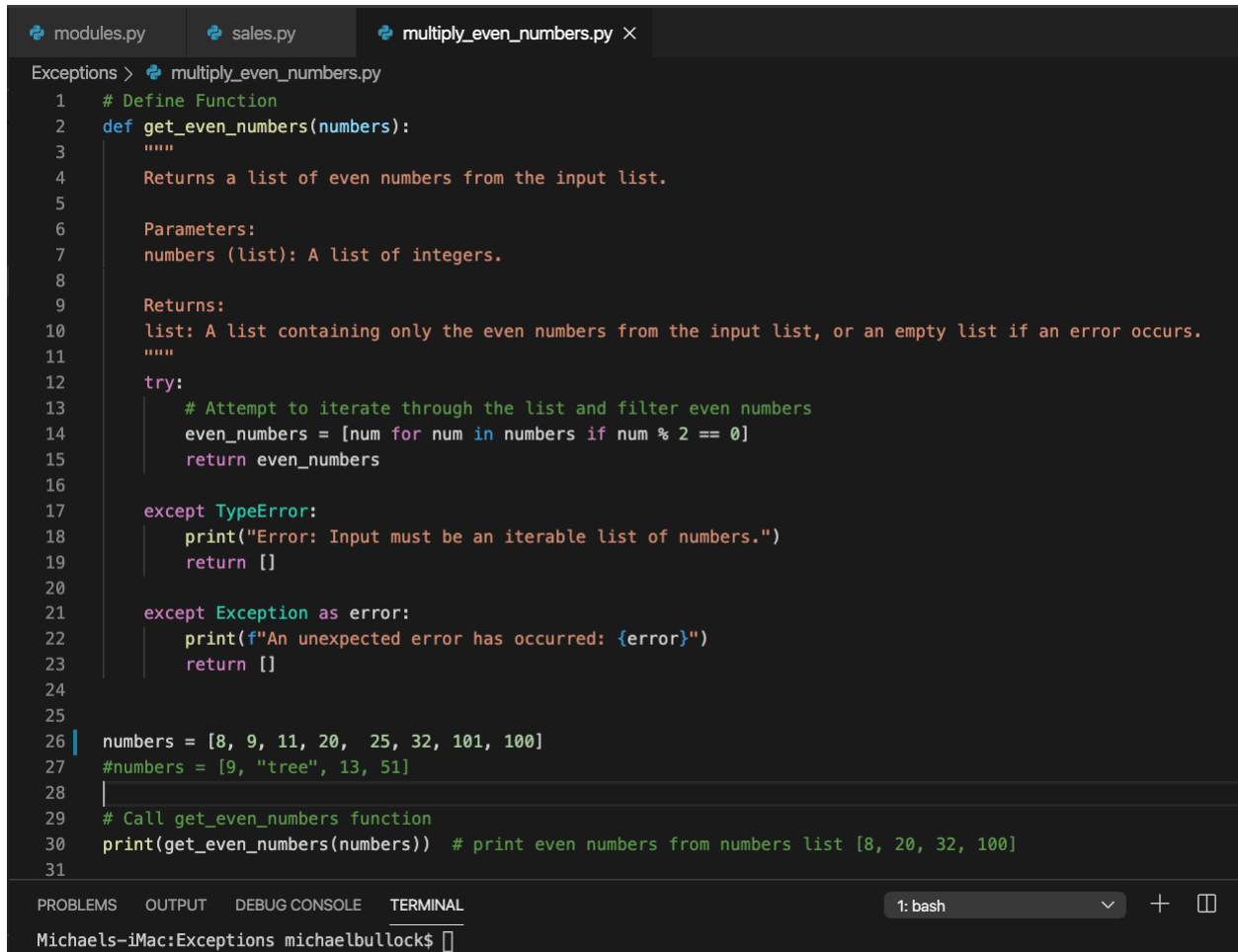


Michael Bullock
Python on AWS
Exceptions



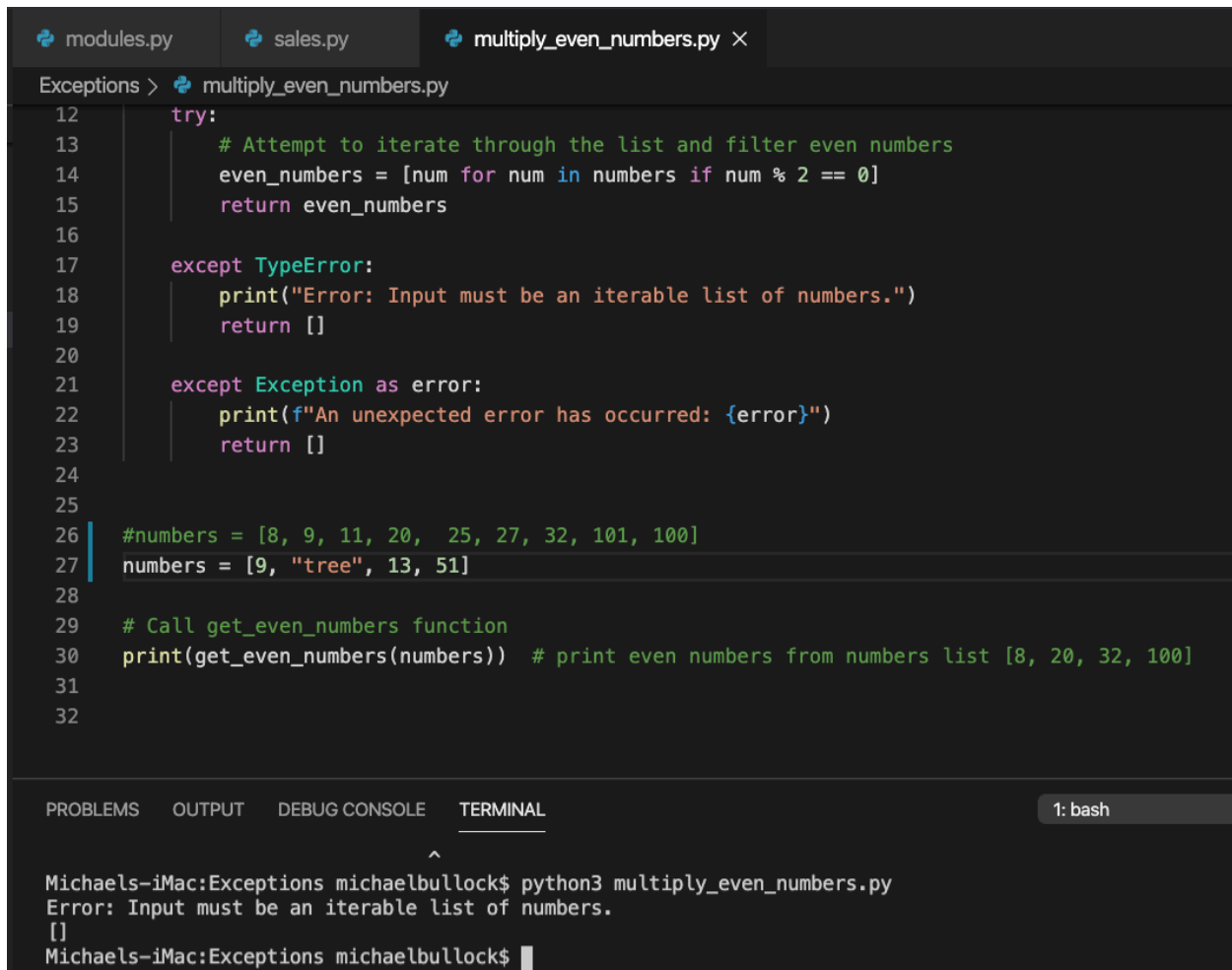
The image shows a code editor with three tabs: `modules.py`, `sales.py`, and `multiply_even_numbers.py`. The `multiply_even_numbers.py` tab is active, displaying a Python script. The script defines a function `get_even_numbers` that filters even numbers from a list. It includes docstrings for the function's purpose, parameters, and return values. The function uses a `try` block to handle `TypeError` and a general `Exception`. Below the function, there are two test cases: one with a list of integers and one with a list containing a string. The script ends with a call to `print` to display the results. At the bottom, a terminal window is open, showing the command prompt `Michael's-iMac:Exceptions michaelbullock$`.

```
1  # Define Function
2  def get_even_numbers(numbers):
3      """
4      Returns a list of even numbers from the input list.
5
6      Parameters:
7      numbers (list): A list of integers.
8
9      Returns:
10     list: A list containing only the even numbers from the input list, or an empty list if an error occurs.
11     """
12     try:
13         # Attempt to iterate through the list and filter even numbers
14         even_numbers = [num for num in numbers if num % 2 == 0]
15         return even_numbers
16
17     except TypeError:
18         print("Error: Input must be an iterable list of numbers.")
19         return []
20
21     except Exception as error:
22         print(f"An unexpected error has occurred: {error}")
23         return []
24
25
26 numbers = [8, 9, 11, 20, 25, 32, 101, 100]
27 #numbers = [9, "tree", 13, 51]
28
29 # Call get_even_numbers function
30 print(get_even_numbers(numbers)) # print even numbers from numbers list [8, 20, 32, 100]
31
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: bash

Michael's-iMac:Exceptions michaelbullock\$

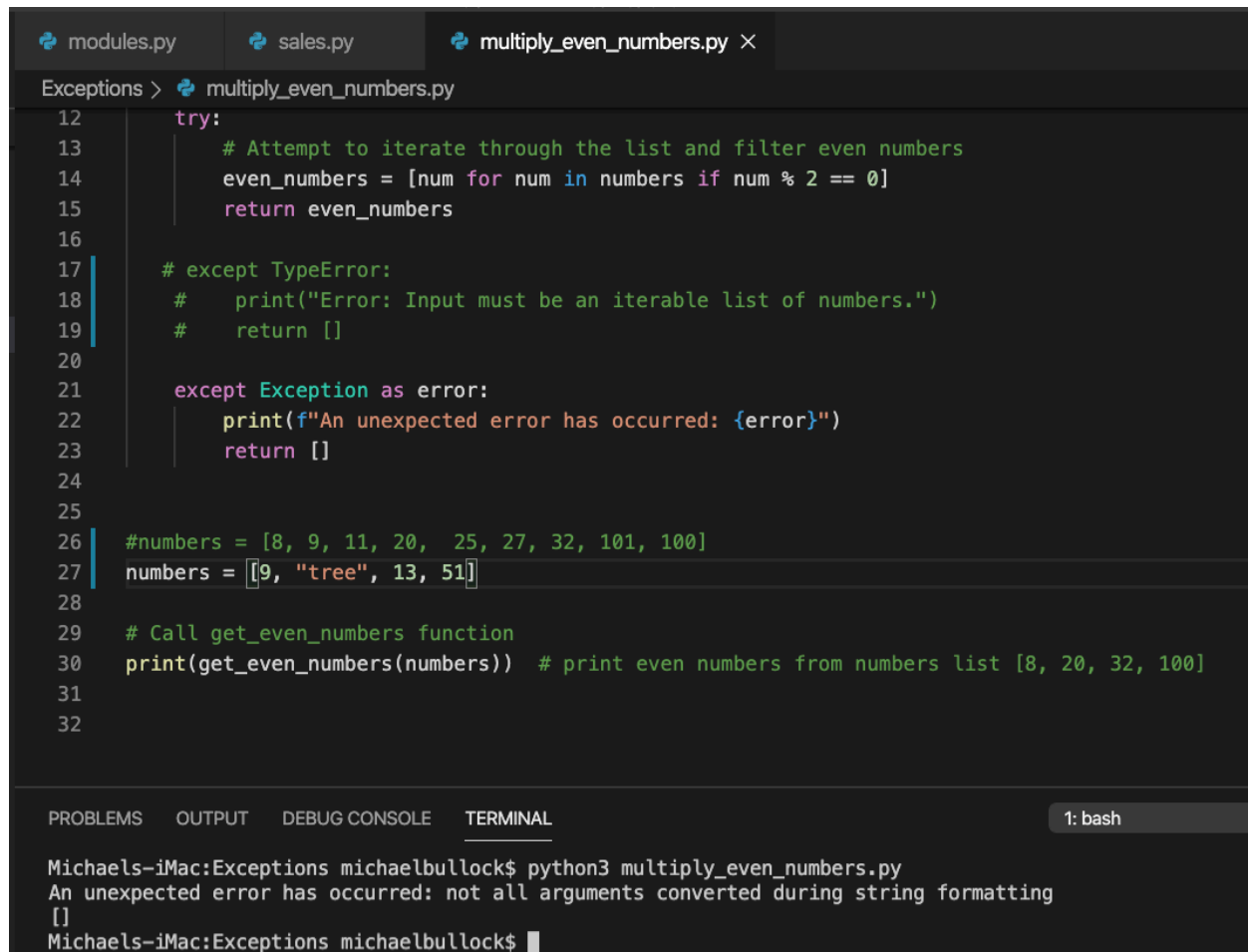
TypeError Example:



```
modules.py sales.py multiply_even_numbers.py ×
Exceptions > multiply_even_numbers.py
12 try:
13     # Attempt to iterate through the list and filter even numbers
14     even_numbers = [num for num in numbers if num % 2 == 0]
15     return even_numbers
16
17 except TypeError:
18     print("Error: Input must be an iterable list of numbers.")
19     return []
20
21 except Exception as error:
22     print(f"An unexpected error has occurred: {error}")
23     return []
24
25
26 #numbers = [8, 9, 11, 20, 25, 27, 32, 101, 100]
27 numbers = [9, "tree", 13, 51]
28
29 # Call get_even_numbers function
30 print(get_even_numbers(numbers)) # print even numbers from numbers list [8, 20, 32, 100]
31
32

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: bash
^
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:



The image shows a code editor with three tabs: `modules.py`, `sales.py`, and `multiply_even_numbers.py`. The `multiply_even_numbers.py` tab is active, showing a Python script. The script defines a function `get_even_numbers` that filters even numbers from a list. It includes a `try` block for filtering and an `except` block to handle errors. The script then defines a list `numbers` and calls the `get_even_numbers` function. The terminal at the bottom shows the output of running the script, which is an empty list `[]` due to a formatting error in the `print` statement.

```
Exceptions > multiply_even_numbers.py
12     try:
13         # Attempt to iterate through the list and filter even numbers
14         even_numbers = [num for num in numbers if num % 2 == 0]
15         return even_numbers
16
17     # except TypeError:
18     #     print("Error: Input must be an iterable list of numbers.")
19     #     return []
20
21     except Exception as error:
22         print(f"An unexpected error has occurred: {error}")
23         return []
24
25
26     #numbers = [8, 9, 11, 20, 25, 27, 32, 101, 100]
27     numbers = [9, "tree", 13, 51]
28
29     # Call get_even_numbers function
30     print(get_even_numbers(numbers)) # print even numbers from numbers list [8, 20, 32, 100]
31
32
```

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

```
modules.py  sales.py  multiply_even_numbers.py

Exceptions > sales.py

1  # Define the order dictionary
2  Order = {
3      'tomatoes': 30,
4      'thyme': 4.50,
5      'garlic': 7.5,
6      'rice': 10,
7      'onions': 4,
8      'fish': 9.99
9  }

10
11 def add_items_to_cart(order):
12     """
13     Adds items to a cart and generates a receipt file.
14
15     Parameters:
16     order dictionary: A dictionary with items as keys and prices as values.
17
18     Writes:
19     A file named 'grocery_receipt.txt' containing the items, their prices, and the total.
20
21     Raises:
22     TypeError: If the order is not a dictionary or contains non-numeric prices.
23     ValueError: If the order is empty or contains invalid data types.
24     Exception: For any other unexpected errors.
25     """
26     cart = []
27     total = 0.0
28
29     try:
30         # Verify that `order` has content to process
31         if not order:
32             raise ValueError("The order is empty.")
33
```

```

33
34     # Process each item in the order
35     for item, price in order.items():
36         if not isinstance(price, (int, float)):
37             raise TypeError(f"The price for '{item}' must be a number.")
38         cart.append((item, price))
39         total += price
40
41     # Write the receipt to a file
42     with open('grocery_receipt.txt', 'w') as receipt_file:
43         receipt_file.write("Grocery Cart Receipt\n")
44         receipt_file.write("-----\n")
45         for item, price in cart:
46             receipt_file.write(f"{item}: ${price:.2f}\n")
47         receipt_file.write("-----\n")
48         receipt_file.write(f"Total: ${total:.2f}\n")
49     print("Receipt generated: 'grocery_receipt.txt'")
50
51 except TypeError as type_error:
52     print(f"TypeError: {type_error}") # Raise if any price is not a numerical value
53 except ValueError as value_error:
54     print(f"ValueError: {value_error}") # Raised if the order is empty or contains an invalid structure
55 except Exception as error:
56     print(f"An unexpected error occurred: {error}") # Catches any unexpected error that might occur.
57
58 # Example use
59 add_items_to_cart(Order)
60

```

Value Error example:

modules.py sales.py multiply_even_numbers.py

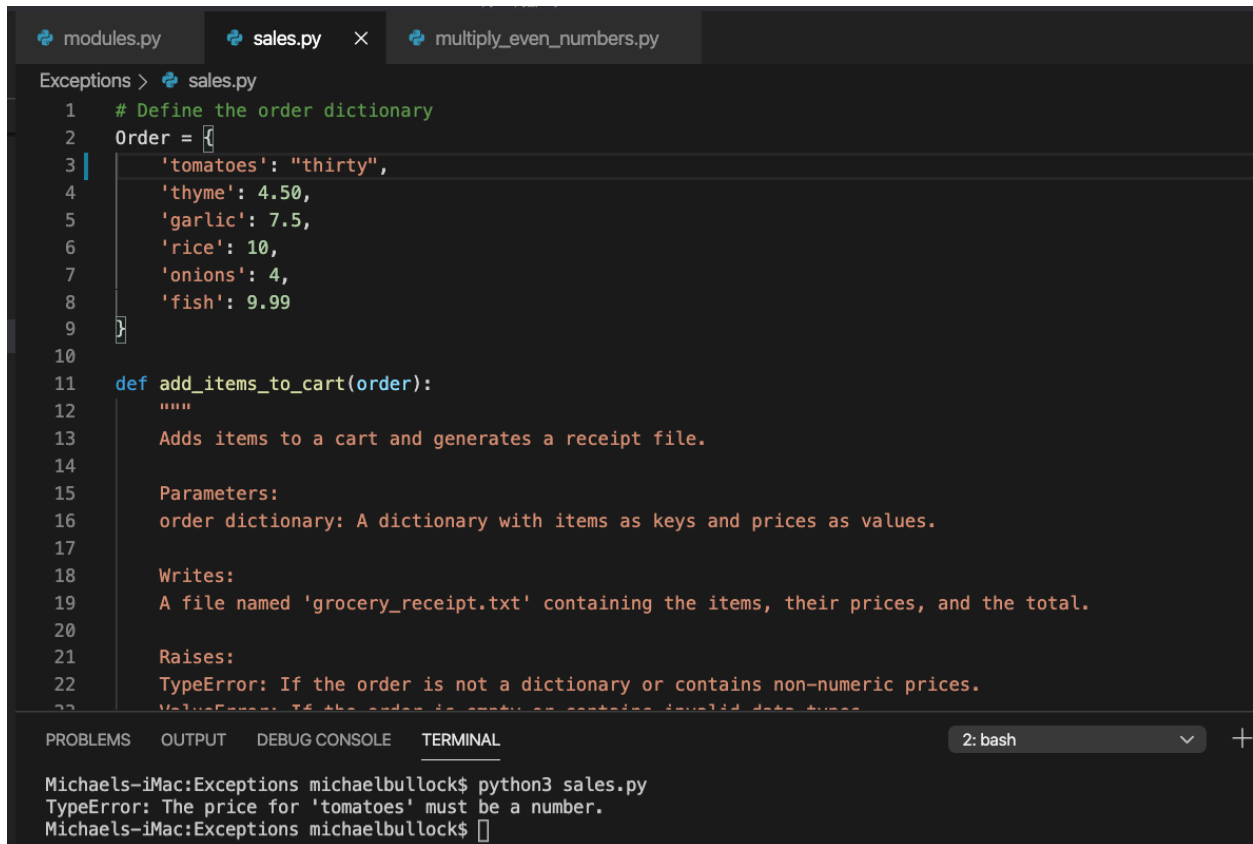
Exceptions > sales.py

```
1 # Define the order dictionary
2 Order = {
3     # 'tomatoes': 30,
4     # 'thyme': 4.50,
5     # 'garlic': 7.5,
6     # 'rice': 10,
7     # 'onions': 4,
8     # 'fish': 9.99
9 }
10
11 def add_items_to_cart(order):
12     """
13     Adds items to a cart and generates a receipt file.
14
15     Parameters:
16     order dictionary: A dictionary with items as keys and prices as values.
17
18     Writes:
19     A file named 'grocery_receipt.txt' containing the items, their prices, and the total.
20
21     Raises:
22     TypeError: If the order is not a dictionary or contains non-numeric prices.
23     ValueError: If the order is empty or contains invalid data types.
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 2: bash

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:



The screenshot shows a code editor with three tabs: `modules.py`, `sales.py` (active), and `multiply_even_numbers.py`. The `sales.py` file contains the following code:

```
1 # Define the order dictionary
2 Order = {
3     'tomatoes': "thirty",
4     'thyme': 4.50,
5     'garlic': 7.5,
6     'rice': 10,
7     'onions': 4,
8     'fish': 9.99
9 }
10
11 def add_items_to_cart(order):
12     """
13     Adds items to a cart and generates a receipt file.
14
15     Parameters:
16     order dictionary: A dictionary with items as keys and prices as values.
17
18     Writes:
19     A file named 'grocery_receipt.txt' containing the items, their prices, and the total.
20
21     Raises:
22     TypeError: If the order is not a dictionary or contains non-numeric prices.
23     ValueError: If the order is empty or contains invalid data types.
```

The terminal at the bottom shows the command `python3 sales.py` being executed, which results in a `TypeError`:

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