



Tecnológico
de Monterrey



MAESTRIA EN INTELIGENCIA
ARTIFICIAL APLICADA

Tarea 5.2

Ejercicio de programación 2

BERNAGA TORRES, ASTRID R. A01793080

PRUEBAS DE SOFTWARE Y
ASEGURAMIENTO DE LA CALIDAD

PROFR. DR GERARDO PADILLA ZÁRATE

PROFR. TUTOR. MTRO. YAMIL ADRIÁN ELÍAS SOTO



compute_sales.py

```
...idad\5.2 Ejercicio de Programacion 2\compute_sales.py 1
"""
Este módulo calcula el total de ventas a partir de un catálogo de precios y un
registro de ventas, ambos proporcionados en archivos JSON. Calcula el total de
ventas, maneja datos inválidos y reporta el tiempo total de ejecución del
cálculo.
"""

import json
import sys
import time

def read_json_file(filename):
    """
    Lee y devuelve el contenido de un archivo JSON.

    Args:
        filename (str): La ruta al archivo JSON a leer.

    Returns:
        dict/list: Contenido del archivo JSON.
    """
    try:
        with open(filename, 'r', encoding='utf-8') as file:
            return json.load(file)
    except FileNotFoundError:
        print(f"Error: El archivo {filename} no fue encontrado.")
        return None
    except json.JSONDecodeError:
        print(f"Error: Archivo {filename} no es válido JSON.")
        return None

def calculate_sales(prices, sales):
    """
    Calcula el total de ventas basado en los precios y el registro de ventas.

    Args:
        prices (dict): Un diccionario con los precios de los productos.
        sales (list): Lista de registros de ventas.

    Returns:
        float: El total de las ventas.
    """
    total_sales = 0
    products_not_found = set() # Guarda los productos no encontrados
    for sale in sales:
        product_title = sale.get('Product')
        quantity = sale.get('Quantity', 0)
        if product_title in prices:
            total_sales += prices[product_title] * quantity
        else:
            products_not_found.add(product_title)
            # Añade el producto no encontrado

    # Imprime los productos no encontrados
    if products_not_found:
        print("Productos no encontrados en el catálogo:")
        print(", ".join(products_not_found))

    return total_sales

def main(product_list_filename, sales_filename):
    """
    Función principal que ejecuta el cálculo del total de ventas.

    Lee los archivos de lista de productos y registros de ventas, calcula el
    """
```

compute_sales.py

```
...idad\5.2 Ejercicio de Programacion 2\compute_sales.py 2
total de ventas combinando estos datos, e imprime y guarda el resultado.

Args:
    product_list_filename (str): Ruta al archivo JSON de la lista de
        productos.
    sales_filename (str): Ruta al archivo JSON de los registros de ventas.
"""
start_time = time.time()

product_list = read_json_file(product_list_filename)
if product_list is None:
    return

sales_record = read_json_file(sales_filename)
if sales_record is None:
    return

prices = {item['title']: item['price'] for item in product_list}

total_sales = calculate_sales(prices, sales_record)

elapsed_time = time.time() - start_time

result_string = (f"Total ventas: ${total_sales:.2f}\n"
                f"Tiempo ejecución: {elapsed_time:.2f}s")
print(result_string)

with open('sales_results.txt', 'w', encoding='utf-8') as file:
    file.write(result_string)

if __name__ == '__main__':
    if len(sys.argv) != 3:
        print("Uso: python compute_sales.py product_list.json "
              "sales_record.json")
    else:
        main(sys.argv[1], sys.argv[2])
```

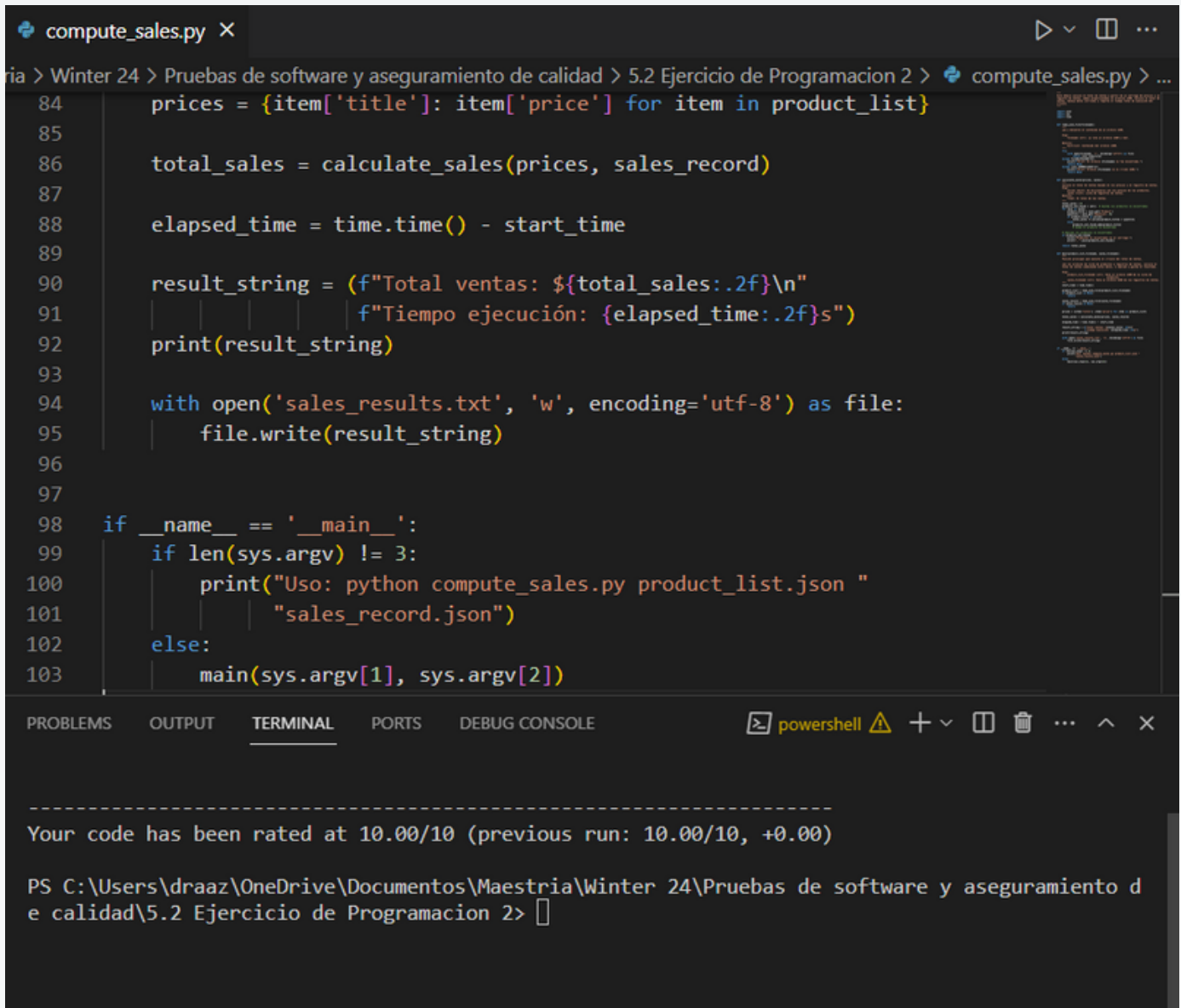
Código representa un módulo de Python que realiza el cálculo del total de ventas a partir de un catálogo de precios y un registro de ventas, ambos proporcionados en archivos JSON.

La función *read_json_file* se encarga de leer y cargar el contenido de un archivo JSON, manejando adecuadamente casos de archivos no encontrados o inválidos.

La función *calculate_sales* realiza el cálculo del total de ventas utilizando los precios y el registro de ventas dados como entrada, manejando también la situación en la que un producto no está en el catálogo como ocurre en el ejercicio 3.

Finalmente, la función principal *main* coordina la lectura de los archivos, el cálculo de las ventas totales, la medición del tiempo de ejecución y la generación de un archivo de resultados.

PYLINT



```
compute_sales.py X
ria > Winter 24 > Pruebas de software y aseguramiento de calidad > 5.2 Ejercicio de Programacion 2 > compute_sales.py > ...
84 prices = {item['title']: item['price'] for item in product_list}
85
86 total_sales = calculate_sales(prices, sales_record)
87
88 elapsed_time = time.time() - start_time
89
90 result_string = (f"Total ventas: ${total_sales:.2f}\n"
91                 f"Tiempo ejecución: {elapsed_time:.2f}s")
92 print(result_string)
93
94 with open('sales_results.txt', 'w', encoding='utf-8') as file:
95     file.write(result_string)
96
97
98 if __name__ == '__main__':
99     if len(sys.argv) != 3:
100         print("Uso: python compute_sales.py product_list.json "
101              "sales_record.json")
102     else:
103         main(sys.argv[1], sys.argv[2])

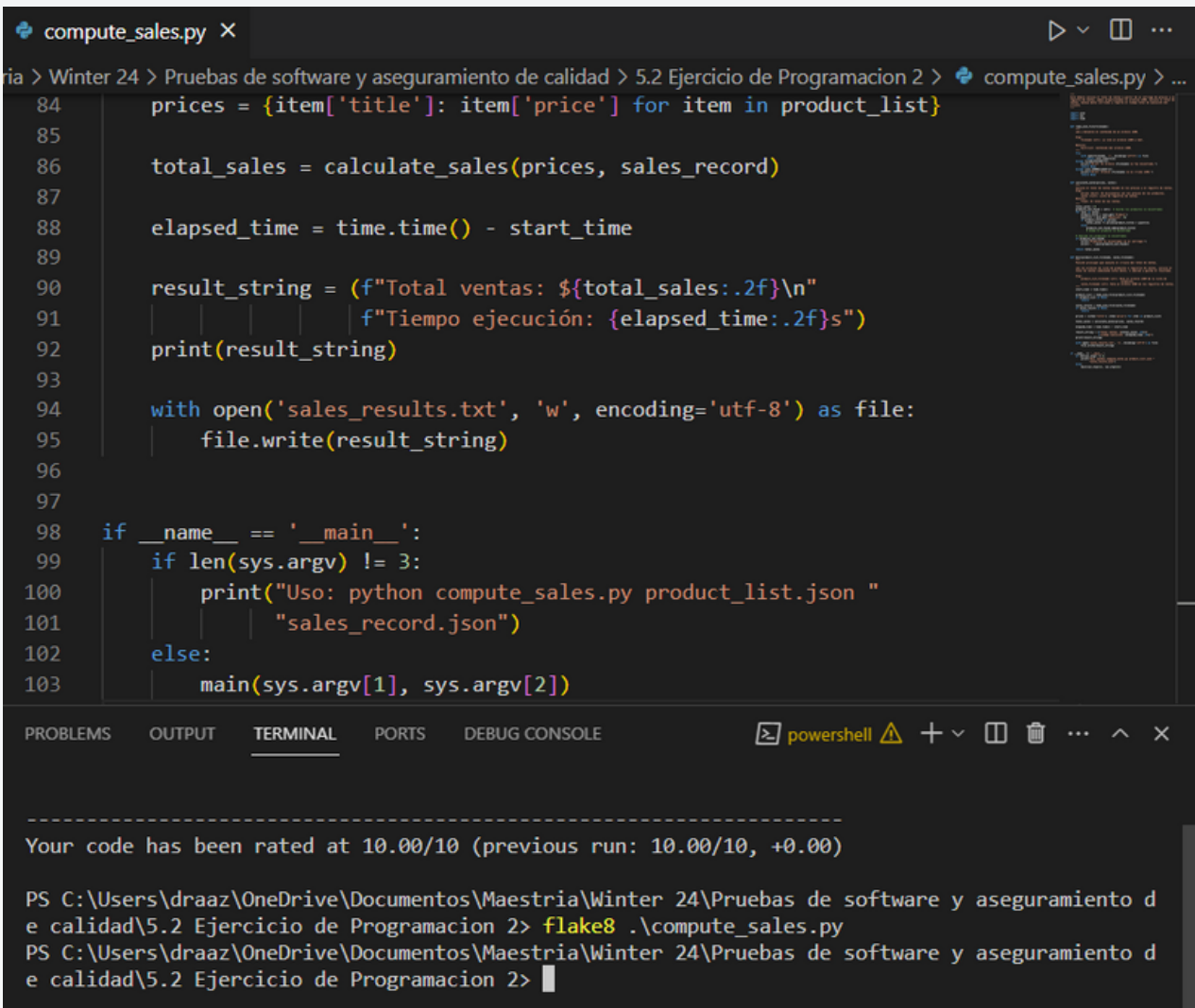
PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE powershell
-----
Your code has been rated at 10.00/10 (previous run: 10.00/10, +0.00)

PS C:\Users\draaz\OneDrive\Documentos\Maestria\Winter 24\Pruebas de software y aseguramiento d
e calidad\5.2 Ejercicio de Programacion 2> 
```

El código ha sido calificado con una puntuación perfecta de 10.00/10.

Esto significa que pylint no ha encontrado ningún problema en el código en términos de convenciones de estilo, errores o posibles problemas.

FLAKE8



The screenshot shows a VS Code editor window with a file named `compute_sales.py` open. The code is a Python script that calculates total sales and execution time. The terminal at the bottom shows the command `flake8 .\compute_sales.py` being executed, and the output indicates that the code is clean with a score of 10.00/10.

```
84 prices = {item['title']: item['price'] for item in product_list}
85
86 total_sales = calculate_sales(prices, sales_record)
87
88 elapsed_time = time.time() - start_time
89
90 result_string = (f"Total ventas: ${total_sales:.2f}\n"
91                 f"Tiempo ejecución: {elapsed_time:.2f}s")
92 print(result_string)
93
94 with open('sales_results.txt', 'w', encoding='utf-8') as file:
95     file.write(result_string)
96
97
98 if __name__ == '__main__':
99     if len(sys.argv) != 3:
100         print("Uso: python compute_sales.py product_list.json "
101              "sales_record.json")
102     else:
103         main(sys.argv[1], sys.argv[2])
```

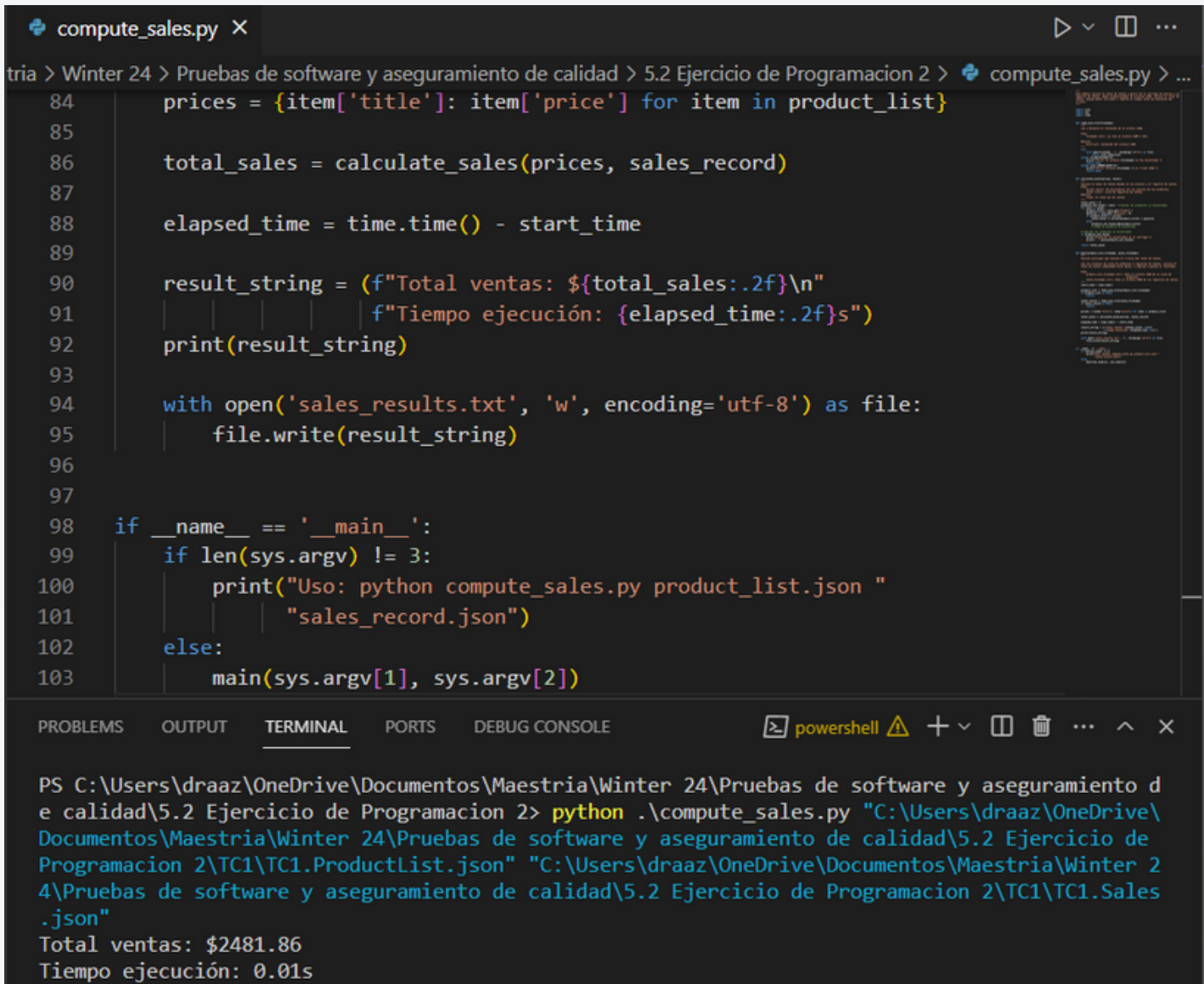
PROBLEMS OUTPUT **TERMINAL** PORTS DEBUG CONSOLE

Your code has been rated at 10.00/10 (previous run: 10.00/10, +0.00)

PS C:\Users\draaz\OneDrive\Documentos\Maestria\Winter 24\Pruebas de software y aseguramiento de calidad\5.2 Ejercicio de Programacion 2> flake8 .\compute_sales.py
PS C:\Users\draaz\OneDrive\Documentos\Maestria\Winter 24\Pruebas de software y aseguramiento de calidad\5.2 Ejercicio de Programacion 2> █

También se ejecutó flake8 y no se generaron comentarios, lo que indica que el código cumple con las convenciones de estilo definidas.

CAPTURA DE PANTALLA TC1



The screenshot displays a Visual Studio Code window with a Python file named `compute_sales.py` and its execution output in the terminal.

Python Code:

```
84 prices = {item['title']: item['price'] for item in product_list}
85
86 total_sales = calculate_sales(prices, sales_record)
87
88 elapsed_time = time.time() - start_time
89
90 result_string = (f"Total ventas: ${total_sales:.2f}\n"
91                | f"Tiempo ejecución: {elapsed_time:.2f}s")
92 print(result_string)
93
94 with open('sales_results.txt', 'w', encoding='utf-8') as file:
95     file.write(result_string)
96
97
98 if __name__ == '__main__':
99     if len(sys.argv) != 3:
100         print("Uso: python compute_sales.py product_list.json "
101              | "sales_record.json")
102     else:
103         main(sys.argv[1], sys.argv[2])
```

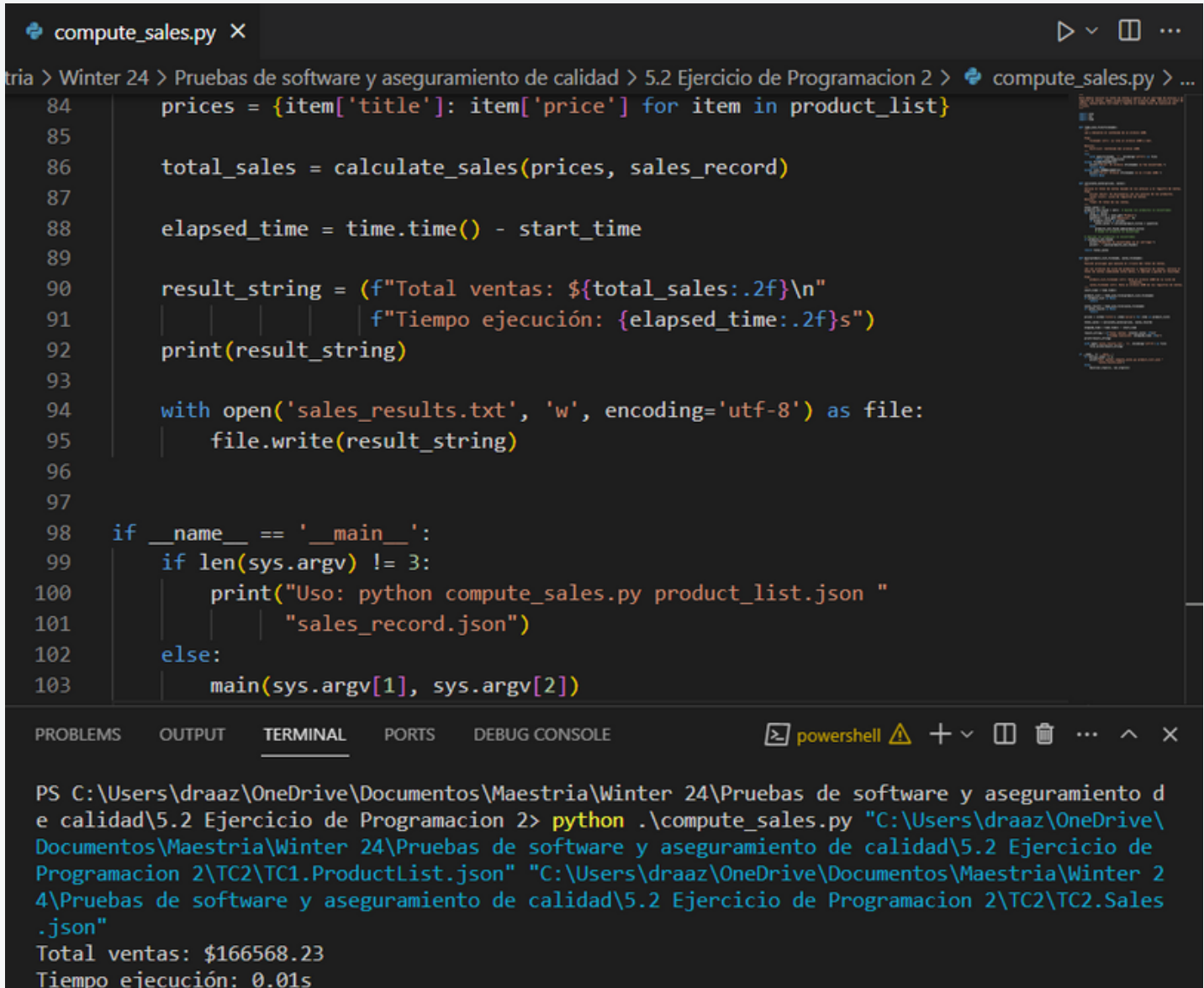
Terminal Output:

```
PS C:\Users\draaz\OneDrive\Documentos\Maestria\Winter 24\Pruebas de software y aseguramiento d
e calidad\5.2 Ejercicio de Programacion 2> python .\compute_sales.py "C:\Users\draaz\OneDrive\
Documentos\Maestria\Winter 24\Pruebas de software y aseguramiento de calidad\5.2 Ejercicio de
Programacion 2\TC1\TC1.ProductList.json" "C:\Users\draaz\OneDrive\Documentos\Maestria\Winter 2
4\Pruebas de software y aseguramiento de calidad\5.2 Ejercicio de Programacion 2\TC1\TC1.Sales
.json"
Total ventas: $2481.86
Tiempo ejecución: 0.01s
```

Prueba con TC1:

- Total de ventas: \$2481.86
- Tiempo de ejecución: 0.01 segundos

CAPTURAS DE PANTALLA TC2



The image shows a screenshot of a code editor and a terminal window. The code editor displays a Python script named `compute_sales.py`. The script calculates the total sales from a list of products and records the execution time. The terminal window shows the command to run the script with two JSON files as arguments, and the output of the script.

```
compute_sales.py X
tria > Winter 24 > Pruebas de software y aseguramiento de calidad > 5.2 Ejercicio de Programacion 2 > compute_sales.py > ...
84     prices = {item['title']: item['price'] for item in product_list}
85
86     total_sales = calculate_sales(prices, sales_record)
87
88     elapsed_time = time.time() - start_time
89
90     result_string = (f"Total ventas: ${total_sales:.2f}\n"
91                     | f"Tiempo ejecución: {elapsed_time:.2f}s")
92     print(result_string)
93
94     with open('sales_results.txt', 'w', encoding='utf-8') as file:
95         file.write(result_string)
96
97
98     if __name__ == '__main__':
99         if len(sys.argv) != 3:
100             print("Uso: python compute_sales.py product_list.json "
101                  | "sales_record.json")
102         else:
103             main(sys.argv[1], sys.argv[2])
```

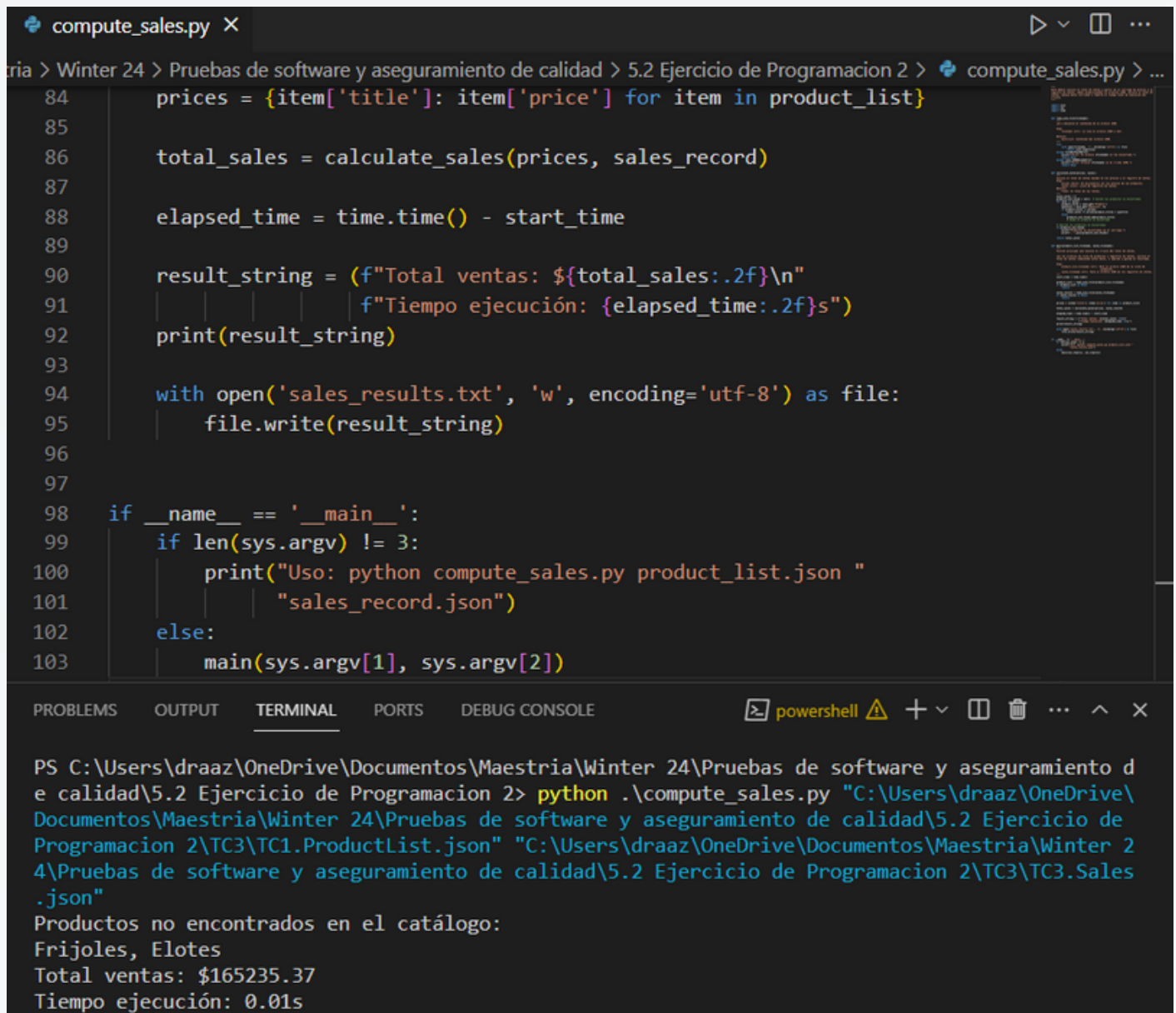
PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE powershell

```
PS C:\Users\draaz\OneDrive\Documentos\Maestria\Winter 24\Pruebas de software y aseguramiento d
e calidad\5.2 Ejercicio de Programacion 2> python .\compute_sales.py "C:\Users\draaz\OneDrive\
Documentos\Maestria\Winter 24\Pruebas de software y aseguramiento de calidad\5.2 Ejercicio de
Programacion 2\TC2\TC1.ProductList.json" "C:\Users\draaz\OneDrive\Documentos\Maestria\Winter 2
4\Pruebas de software y aseguramiento de calidad\5.2 Ejercicio de Programacion 2\TC2\TC2.Sales
.json"
Total ventas: $166568.23
Tiempo ejecución: 0.01s
```

Prueba con TC2:

- Total de ventas: \$166568.23
- Tiempo de ejecución: 0.01 segundos

CAPTURAS DE PANTALLA TC3



```
compute_sales.py X
tria > Winter 24 > Pruebas de software y aseguramiento de calidad > 5.2 Ejercicio de Programacion 2 > compute_sales.py > ...
84     prices = {item['title']: item['price'] for item in product_list}
85
86     total_sales = calculate_sales(prices, sales_record)
87
88     elapsed_time = time.time() - start_time
89
90     result_string = (f"Total ventas: ${total_sales:.2f}\n"
91                     | f"Tiempo ejecución: {elapsed_time:.2f}s")
92     print(result_string)
93
94     with open('sales_results.txt', 'w', encoding='utf-8') as file:
95         file.write(result_string)
96
97
98 if __name__ == '__main__':
99     if len(sys.argv) != 3:
100         print("Uso: python compute_sales.py product_list.json "
101              | "sales_record.json")
102     else:
103         main(sys.argv[1], sys.argv[2])

PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE
PS C:\Users\draaz\OneDrive\Documentos\Maestria\Winter 24\Pruebas de software y aseguramiento de calidad\5.2 Ejercicio de Programacion 2> python .\compute_sales.py "C:\Users\draaz\OneDrive\Documentos\Maestria\Winter 24\Pruebas de software y aseguramiento de calidad\5.2 Ejercicio de Programacion 2\TC3\TC1.ProductList.json" "C:\Users\draaz\OneDrive\Documentos\Maestria\Winter 24\Pruebas de software y aseguramiento de calidad\5.2 Ejercicio de Programacion 2\TC3\TC3.Sales.json"
Productos no encontrados en el catálogo:
Frijoles, Elotes
Total ventas: $165235.37
Tiempo ejecución: 0.01s
```

Prueba con TC3:

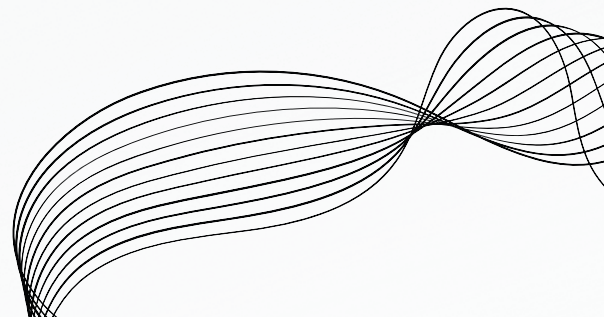
- Productos no encontrados en el catálogo: Frijoles, Elotes
- Total de ventas: \$165235.37
- Tiempo de ejecución: 0.01 segundos



ANALISIS GENERAL

- Los resultados de las pruebas indican que el código funciona correctamente y produce resultados esperados.
- Las pruebas se ejecutan en un tiempo de ejecución muy corto (0.01 segundos), lo que sugiere que el rendimiento del código es adecuado para los conjuntos de datos proporcionados.
- En el caso de la prueba TC3, el código identifica correctamente los productos que no están en el catálogo y maneja la situación adecuadamente, imprimiendo los productos no encontrados.

En general, el código está escrito con los estándares esperados, sin errores detectados por pylint o flake8, y funciona como se espera según los resultados de las pruebas proporcionados.





LIGA DE GITHUB

<https://github.com/Astroцитos/A01793080A5.2>

