Pruebas de software y aseguramiento de la calidad

Emmanuel Francisco González Velázquez

A01364577

Tarea

5.2 Ejercicio de programación 2

Repositorio de Github:

https://github.com/A01364577/A01364577_ActividadA5.2

Código Programa previo a revisión

```
import sys
import json
import time

def parse_arguments():
    #Parse command-line arguments.
    if len(sys.argv) != 3:
        print("Usage: python computeSales.py priceCatalogue.json salesRecord.json")
        sys.exit(1)
    return sys.argv[1], sys.argv[2]

def load_json(filename):
    #Load data from a JSON file.
    try:
        with open(filename, 'r') as file:
            data = json.load(file)
        return data
    except FileNotFoundError:
        print(f"File '{filename}' not found.")
        sys.exit(1)
    except json.JSONDecodeError:
        print(f"Invalid JSON format in '{filename}'.")
        sys.exit(1)

def compute_total_cost(price_catalogue, sales_record):
    #Compute the total cost for all sales.
    total_cost = 0
    for sale in sales_record:
        product_name = sale['Product']
        quantity = sale['Quantity']
        # Find product price in the catalogue
        for item in price_catalogue:
            if item['title'] == product_name:
                price = item['price']
                total_cost += price * quantity
                break  # Once found, no need to continue searching
        else:
```

```
            print(f"Price for product '{product_name}' not found in catalogue.")
    return total_cost

def main():
    start_time = time.time()

    # Parse command-line arguments
    catalogue_file, sales_file = parse_arguments()

    # Load JSON files
    price_catalogue = load_json(catalogue_file)
    sales_record = load_json(sales_file)

    # Compute total cost
    total_cost = compute_total_cost(price_catalogue, sales_record)

    # Output results
    print("Total cost of all sales:", total_cost)
    with open("SalesResults.txt", "w") as output_file:
        output_file.write(f"Total cost of all sales: {total_cost}")

    # Display execution time
    end_time = time.time()
    execution_time = end_time - start_time
    print("Execution time:", execution_time, "seconds")

if __name__ == "__main__":
    main()
```

Errores detectados con Pylint

C:\Users\efgv1\Documents\Tec - IA\Software_quality\Tarea 5.2\TC3\TC3>pylint computeSales.py
********** Module computeSales
computeSales.py:1:0: C0114: Missing module docstring (missing-module-docstring)
computeSales.py:1:0: C0103: Module name "computeSales" doesn't conform to snake_case naming style (invalid-name)
computeSales.py:5:0: C0116: Missing function or method docstring (missing-function-docstring)
computeSales.py:12:0: C0116: Missing function or method docstring (missing-function-docstring)
computeSales.py:15:13: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
computeSales.py:25:0: C0116: Missing function or method docstring (missing-function-docstring)
computeSales.py:41:0: C0116: Missing function or method docstring (missing-function-docstring)
computeSales.py:56:9: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)

Your code has been rated at 8.22/10

Errores detectados con Flake 8

.\computeSales.py:5:1: E302 expected 2 blank lines, found 1
.\computeSales.py:6:5: E265 block comment should start with '# '
.\computeSales.py:8:80: E501 line too long (83 > 79 characters)
.\computeSales.py:12:1: E302 expected 2 blank lines, found 1
.\computeSales.py:13:5: E265 block comment should start with '# '
.\computeSales.py:25:1: E302 expected 2 blank lines, found 1
.\computeSales.py:26:5: E265 block comment should start with '# '
.\computeSales.py:38:80: E501 line too long (80 > 79 characters)
.\computeSales.py:41:1: E302 expected 2 blank lines, found 1
.\computeSales.py:64:1: E305 expected 2 blank lines after class or function definition, found 1

Corrections performed in the code

```
"""
This module computes the total cost of sales based on a price catalogue and sales record.
"""

import sys
import json
import time


def parse_arguments():

  """
  Parse command-line arguments.

  Returns:
    str: Filename of the price catalogue JSON file.
    str: Filename of the sales record JSON file.
  """
  if len(sys.argv) != 3:
    print("python computeSales.py priceCatalogue.json salesRecord.json")
    sys.exit(1)
  return sys.argv[1], sys.argv[2]


def load_json(filename):
  """
  Load data from a JSON file.

  Args:
    filename (str): The filename of the JSON file.

  Returns:
    dict: Data loaded from the JSON file.
  """
  try:
    with open(filename, 'r', encoding='utf-8') as file:
      data = json.load(file)
    return data
  except FileNotFoundError:
    print(f"File '{filename}' not found.")
    sys.exit(1)
```

```
    except json.JSONDecodeError:
        print(f"Invalid JSON format in '{filename}'.")
        sys.exit(1)


def compute_total_cost(price_catalogue, sales_record):
    """
    Compute the total cost for all sales.

    Args:
        price_catalogue (list): List of products with their prices.
        sales_record (list): List of sales records.

    Returns:
        float: Total cost of all sales.
    """
    total_cost = 0
    for sale in sales_record:
        product_name = sale['Product']
        quantity = sale['Quantity']
        # Find product price in the catalogue
        for item in price_catalogue:
            if item['title'] == product_name:
                price = item['price']
                total_cost += price * quantity
                break  # Once found, no need to continue searching
        else:
            print(f"Price of '{product_name}' not found in catalogue.")
    return total_cost


def main():
    """
    Main function.
    """
    start_time = time.time()

    # Parse command-line arguments
    catalogue_file, sales_file = parse_arguments()

    # Load JSON files
    price_catalogue = load_json(catalogue_file)
    sales_record = load_json(sales_file)

    # Compute total cost
```

```
    total_cost = compute_total_cost(price_catalogue, sales_record)

    # Output results
    print("Total cost of all sales:", total_cost)
    with open("SalesResults.txt", "w", encoding='utf-8') as output_file:
        output_file.write(f"Total cost of all sales: {total_cost}\n")

    # Display execution time
    end_time = time.time()
    execution_time = end_time - start_time
    print("Execution time:", execution_time, "seconds")


if __name__ == "__main__":
    main()
```

Results after correction of flake8 evaluation

No more errors detected

Results after correction of pylint evaluation

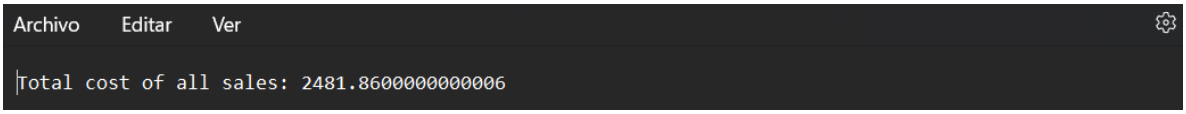
Your code has been rated at 10.00/10 (previous run: 8.22/10, +1.78)

Test case run in TC1

Result in CMD window

```
C:\Users\efgv1\Documents\Tec - IA\Software_quality\Tarea 5.2\TC1\TC1>python compute_sales.py TC1.ProductList.json TC1.Sa
les.json
Total cost of all sales: 2481.8600000000006
Execution time: 0.0010461807250976562 seconds
```

Generated .txt



Test case run in TC2

Result in CMD window

```
C:\Users\efgv1\Documents\Tec - IA\Software_quality\Tarea 5.2\TC2\TC2>python compute_sales.py TC1.ProductList.json TC2.Sales.jso
n
Total cost of all sales: 166568.22999999998
Execution time: 0.00090146064758300078 seconds
```

Generated .txt

Test case run in TC3

Result in CMD window

```
C:\Users\efgv1\Documents\Tec - IA\Software_quality\Tarea 5.2\TC3\TC3>python compute_sales.py TC1.ProductList.json TC3.Sales.jso
n
Price of 'Elotes' not found in catalogue.
Price of 'Frijoles' not found in catalogue.
Total cost of all sales: 165235.37
Execution time: 0.002936840057373047 seconds
```

Generated .txt