7-2-2025

Tecnológico de Monterrey

Gustavo Andrés García Anguiano

Pruebas de software y aseguramiento de la calidad

Actividad 5.2

INDICE

Compute sales

```
"""
This script calculates the total sales cost
based on a price catalogue and a sales record.
It handles invalid data and provides a detailed summary.

Usage:
    python computeSales.py <price_catalogue.json> <sales_data.json>

Output:
    - Displays total sales and execution time on the console.
    - Saves the results to 'SalesResults.txt'.
"""
import json
import argparse
import sys
import time


def load_json_file(filename):
    """Reads a JSON file and returns its content."""
    try:
        with open(filename, 'r', encoding='utf-8-sig') as file:
            return json.load(file)
    except FileNotFoundError:
        print(
            f"Error: The file '{filename}' was not found.",
            file=sys.stderr
            )
    except json.JSONDecodeError:
        print(
            f"Error: The file '{filename}' contains invalid JSON.",
            file=sys.stderr)
    return None


def calculate_total_cost(price_catalogue, sales_data):
    """Computes the total sales cost from the given sales data."""
    total_cost = 0
    errors = []
    price_dict = {item["title"]: item["price"] for item in price_catalogue}
    for sale in sales_data:
        product = sale.get("Product")
        quantity = sale.get("Quantity")
        if not product:
            errors.append(
                "Warning: Sale record with a missing product name."
                )
            continue
        if product not in price_dict:
            errors.append(
                f"Warning: '{product}' is not listed in the price catalogue."
                )
            continue
        try:
            price = price_dict[product]
            total_cost += price * quantity
        except TypeError:
            errors.append(f"Error: Invalid price format for '{product}'.")
    return total_cost, errors


def main():
    """Main function to parse arguments, load data, and compute total cost."""
    parser = argparse.ArgumentParser(
        description="Compute total sales cost from JSON files."
    )
```

```
    parser.add_argument(
        "price_catalogue",
        help="Path to the price catalogue JSON file."
        )
    parser.add_argument(
        "sales_data",
        help="Path to the sales data JSON file."
        )
    args = parser.parse_args()
    start_time = time.time()
    price_catalogue = load_json_file(args.price_catalogue)
    sales_data = load_json_file(args.sales_data)
    if not price_catalogue or not sales_data:
        sys.exit(1)
    total_cost, errors = calculate_total_cost(price_catalogue, sales_data)
    execution_time = time.time() - start_time
    with open("SalesResults.txt", "w", encoding='utf-8') as result_file:
        result_file.write(f"Total Sales: ${total_cost:.2f}\n")
        result_file.write(f"Execution Time: {execution_time:.4f} seconds\n")
        if errors:
            result_file.write("\nWarnings and Errors:\n")
            for error in errors:
                result_file.write(error + "\n")
    print(f"Total Sales: ${total_cost:.2f}")
    print(f"Execution Time: {execution_time:.4f} seconds")
    if errors:
        print("\nWarnings and Errors:")
        for error in errors:
            print(error)


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