Instituto Tecnológico de Estudios Superiores de Monterrey

TC4017 Pruebas de Software y Aseguramiento de la Calidad

Semana 5 - Actividad 1
Ejercicio de Programación 2

Ronald David Zuniga Sánchez A01686240

11/Febrero/2024

Introducción

El presente informe resume desarrollo de un ejercicio de programación en Python con el objetivo de implementar conceptos pruebas de software para el aseguramiento de calidad tomando como referencia principal el estándar de codificación para este lenguaje, PEP 8, y utilizando los conceptos de pruebas dinámicas y estáticas con Flake 8. Se reconocen los atributos principales de un estándar de codificación para identificar errores y su importancia para el aseguramiento de calidad en la construcción de sistemas de software.

Github Link: https://github.com/rzunick/MNA-Pruebas-de-Software-RDZ/tree/main/Tarea%204_Ejercicio%20de%20Programaci%C3%B3n%202

Descripción

Programming Exercise	Description	Practice	Test Cases and Evidence
1. Compute sales	Req1. The program shall be invoked from a command line. The program shall receive two files as parameters. The first file will contain information in a JSON format about a catalogue of prices of products. The second file will contain a record for all sales in a company. Req 2. The program shall compute the total cost for all sales included in the second JSON archive. The results shall be print on a screen and on a file named SalesResults.txt. The total cost should include all items in the sale considering the cost for every item in the first file. The output must be human readable, so make it easy to read for the user. Req 3. The program shall include the mechanism to handle invalid data in the file. Errors should be displayed in the console and the execution must continue. Req 4. The name of the program shall be computeSales.py Req 5. The minimum format to invoke the program shall be as follows: python computeSales.py priceCatalogue.json salesRecord.json Req 6. The program shall manage files having from hundreds of items to thousands of items. Req 7. The program should include at the end of the execution the time elapsed for the execution and calculus of the data. This number shall be included in the results file and on the screen. Req 8. Be compliant with PEP8.	• Control structures • Console Input output • Mathematical computation • File management • Error handling	Record the execution. Use files included in the assignment.

Código Inicial

Se propone un programa diseñado para calcular el costo total de ventas a partir de dos archivos JSON, uno que contiene un catálogo de precios de productos y otro con registros de ventas.

El código comienza importando los módulos json, sys, y time, necesarios para trabajar con archivos JSON, interactuar con el sistema, y medir el tiempo de ejecución del programa, respectivamente.

Se define función **load_json_file(filename).** Esta función intenta abrir y leer un archivo JSON dado su nombre de archivo (filename). Si el archivo se carga correctamente, devuelve su contenido. Si ocurre un error porque el archivo no se encuentra o contiene JSON inválido, se imprime un mensaje de error y se devuelve None.

```python
import json
import sys
import time

def load_json_file(filename):
    """Load JSON data from a file."""
    try:
        with open(filename, 'r') as file:
            return json.load(file)
    except FileNotFoundError:
        print(f"Error: File '{filename}' not found.")
    except json.JSONDecodeError:
        print(f"Error: File '{filename}' contains invalid JSON.")
    return None
```

Se define función **calculate_total_sales(prices, sales).** Calcula el costo total de ventas. Primero, transforma la lista de precios en un diccionario para facilitar la búsqueda de precios por título de producto. Luego, para cada registro de venta, busca el precio del producto correspondiente y multiplica este precio por la cantidad vendida, sumando al costo total. Si un producto no se encuentra en el catálogo, imprime una advertencia.

Se define función **write_results_to_file(filename, total_cost, elapsed_time)**. Escribe el costo total de las ventas y el tiempo de ejecución en un archivo llamado según el argumento filename. El formato es legible, mostrando el costo total con dos decimales y el tiempo transcurrido en segundos.

```python
def calculate_total_sales(prices, sales):
    """Calculate the total sales cost."""
    total_cost = 0
    prices_dict = {product["title"]: product["price"] for product in prices}
    for sale in sales:
        product_title = sale.get('Product')
        quantity = sale.get('Quantity', 0)
        product_price = prices_dict.get(product_title)
        if product_price is not None:
            total_cost += product_price * quantity
        else:
            print(f"Warning: Product '{product_title}' not found in the catalogue.")
    return total_cost

def write_results_to_file(filename, total_cost, elapsed_time):
    """Write the sales results to a file."""
    with open(filename, 'w') as file:
        file.write(f"Total Sales Cost: ${total_cost:.2f}\n")
        file.write(f"Time Elapsed: {elapsed_time:.2f} seconds\n")
```

Se define función **main(prices_file, sales_file).** Es la función principal que orquesta el flujo del programa. Mide el tiempo de inicio, carga los datos de los archivos JSON de precios y ventas, calcula el costo total de las ventas, mide el tiempo de ejecución, imprime los resultados en la consola y los escribe en un archivo. Si los archivos JSON no se pueden cargar, termina la ejecución sin realizar más acciones.

Se define segemento de ejecución condicional **__name__== "__main__".** Esta sección verifica si el script se está ejecutando como programa principal. Comprueba que se hayan proporcionado exactamente dos argumentos de línea de comandos (además del nombre del script), que deben ser los nombres de archivo del catálogo de precios y el registro de ventas, respectivamente. Si no se cumplen estas condiciones, imprime cómo usar el programa y sale con un código de error. Si se cumplen, llama a la función main con los nombres de archivo proporcionados.

```python
def main(prices_file, sales_file):
    start_time = time.time()

    prices_data = load_json_file(prices_file)
    sales_data = load_json_file(sales_file)

    if prices_data is None or sales_data is None:
        return

    total_cost = calculate_total_sales(prices_data, sales_data)
    elapsed_time = time.time() - start_time

    print(f"Total Sales Cost: ${total_cost:.2f}")
    print(f"Time Elapsed: {elapsed_time:.2f} seconds")

    write_results_to_file('SalesResults.txt', total_cost, elapsed_time)

if __name__ == '__main__':
    if len(sys.argv) != 3:
        print("Usage: python computeSales.py priceCatalogue.json salesRecord.json")
        sys.exit(1)

    prices_file = sys.argv[1]
    sales_file = sys.argv[2]

    main(prices_file, sales_file)
```

Flujo de ejecución:

Al ejecutar el programa desde la línea de comandos con los archivos adecuados, este realiza los siguientes pasos:

1. Mide el tiempo de inicio.
2. Carga los datos de los archivos JSON especificados.
3. Calcula el costo total de las ventas basándose en estos datos.
4. Mide el tiempo total de ejecución.
5. Imprime el costo total de las ventas y el tiempo de ejecución en la consola.
6. Escribe estos mismos resultados en un archivo denominado SalesResults.txt.

Evaluación de Análisis de Errores

Análisis con Pylint

```
PS C:\Users\rdzun\source\repos\compute_Sales\compute_Sales> pylint compute_Sales.py
************* Module compute_Sales
compute_Sales.py:57:0: C0303: Trailing whitespace (trailing-whitespace)
compute_Sales.py:60:0: C0303: Trailing whitespace (trailing-whitespace)
compute_Sales.py:1:0: C0103: Module name "compute_Sales" doesn't conform to snake_case naming style (invalid-name)
compute_Sales.py:1:0: C0114: Missing module docstring (missing-module-docstring)
compute_Sales.py:8:13: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
compute_Sales.py:32:9: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
compute_Sales.py:36:0: C0116: Missing function or method docstring (missing-function-docstring)
compute_Sales.py:36:9: W0621: Redefining name 'prices_file' from outer scope (line 58) (redefined-outer-name)
compute_Sales.py:36:22: W0621: Redefining name 'sales_file' from outer scope (line 59) (redefined-outer-name)

-----------------------------------
Your code has been rated at 8.00/10
```

Se corrijen los puntos identificados para cumplimiento con formato PEP 8.

```
PS C:\Users\rdzun\source\repos\compute_Sales\compute_Sales> pylint compute_Sales.py
************* Module compute_Sales
compute_Sales.py:13:58: C0303: Trailing whitespace (trailing-whitespace)
compute_Sales.py:1:0: C0103: Module name "compute_Sales" doesn't conform to snake_case naming style (invalid-name)

--------------------------------------------------------------------
Your code has been rated at 9.56/10 (previous run: 8.89/10, +0.67)
```

Análisis con Flake8 (mypy)

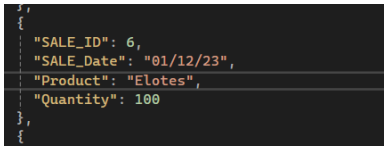
```
PS C:\Users\rdzun\source\repos\compute_Sales\compute_Sales> mypy compute_Sales.py
Success: no issues found in 1 source file
PS C:\Users\rdzun\source\repos\compute_Sales\compute_Sales>
```

Ejecución y Resultados

Ejecución 1

```
PS C:\Users\rdzun\source\repos\compute_Sales\compute_Sales> python compute_Sales.py TC1.ProductList.json TC1.Sales.json
Total Sales Cost: $2481.86
Time Elapsed: 0.00 seconds
PS C:\Users\rdzun\source\repos\compute_Sales\compute_Sales> python compute_Sales.py TC1.ProductList.json TC2.Sales.json
Total Sales Cost: $166568.23
Time Elapsed: 0.00 seconds
PS C:\Users\rdzun\source\repos\compute_Sales\compute_Sales> python compute_Sales.py TC1.ProductList.json TC3.Sales.json
Warning: Product 'Elotes' not found in the catalogue.
Warning: Product 'Frijoles' not found in the catalogue.
Total Sales Cost: $165235.37
Time Elapsed: 0.02 seconds
```

Se identifican 2 registros con Product ID diferentes a los registrados en la base de datos de precios TC1.Productlist: “Elotes”, y “Frijoles”.



Se determina actualizar los (2) únicos registros de ventas modificando el Product ID a su versión en inglés para que coincida con los registros en el Product List.

- “Elotes” a “Corn”
- “Frijoles” a “Green beans”

```
PS C:\Users\rdzun\source\repos\compute_Sales\compute_Sales> python compute_Sales.py TC1.ProductList.json TC3.Sales.json
Warning: Product 'Elotes' not found in the catalogue.
Warning: Product 'Frijoles' not found in the catalogue.
Total Sales Cost: $165235.37
Time Elapsed: 0.02 seconds
PS C:\Users\rdzun\source\repos\compute_Sales\compute_Sales> python compute_Sales.py TC1.ProductList.json TC3.Sales.json
Total Sales Cost: $169469.37
Time Elapsed: 0.00 seconds
```