



Actividad 5.2

Alumna:

Guadalupe Esmeralda González Maldonado (Ao1795767)

Maestría en Inteligencia Artificial Aplicada

**Materia: Pruebas de software y aseguramiento de la
calidad**

Profesor Titular: Dr. Gerardo Padilla Zárate

Febrero 2025

Contenido

ANÁLISIS DE FLAKE8.....3

Conclusión de Flake8:.....3

EJERCICIO DE PROGRAMACIÓN 23

 TC13

 TC24

 TC35

CONCLUSIÓN5

Análisis de Flake8

Para asegurar la calidad del código, se ejecutó Flake8 como herramienta de análisis estático. A lo largo del desarrollo, se encontraron y corrigieron los siguientes errores:

Errores de espaciado y líneas en blanco (E302, E303, E305):

Se ajustaron los espacios entre funciones y estructuras para cumplir con PEP8.

Errores de formato (W292, W391):

Se corrigieron líneas en blanco al final del archivo y la falta de una nueva línea final.

Llamadas a Flake8 dentro del código:

Se implementó la opción `--lint` para permitir que el usuario analice el código con Flake8 de manera automática.

Conclusión de Flake8:

El código final cumple con los estándares de PEP8, lo que garantiza su claridad, mantenibilidad y conformidad con buenas prácticas de desarrollo en Python.

Ejercicio de programación 2

TC1

- **Costo total de ventas:** 2481.86
- **Tiempo de ejecución:** 0.001986 segundos
- **Errores detectados:** Ninguno
- **Observaciones:**
 - El programa procesó correctamente los datos sin errores.
 - Todos los productos en `salesRecord.json` coincidieron con los de `priceCatalogue.json`.

```
computeSales.py X {} salesRecord.json {} PriceCatalogue.json
computeSales.py > read_json_file
1 import sys
2 import json
3 import time
4 import os
5
6
7 def read_json_file(filename):
8     """Lee un archivo JSON y retorna su contenido."""
9     try:
10         with open(filename, 'r', encoding='utf-8') as file:
11             return json.load(file)
12     except (FileNotFoundError, json.JSONDecodeError) as e:
13         print(f"Error al leer {filename}: {e}")
14         return None
15
16
17 def create_price_dict(price_catalogue):
18     """Convierte la lista de productos en un diccionario con precios."""
19     price_dict = {}
20     for item in price_catalogue:
21         product_name = item.get("title")
22         price = item.get("price")
23         if product_name and isinstance(price, (int, float)):
```

PROBLEMS 7 OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + - - - ^ x

```
PS C:\Users\52899\Downloads\A5.2 Archivos de Apoyo> "C:/Program Files (x86)/Microsoft Visual Studio/Shared/Python39_64/python.exe" "c:/Users/52899/Downloads/A5.2 Archivos de Apoyo/computeSales.py"
Uso: python computeSales.py priceCatalogue.json salesRecord.json
PS C:\Users\52899\Downloads\A5.2 Archivos de Apoyo> python computeSales.py "C:\Users\52899\Downloads\A5.2 Archivos de Apoyo\TC1\PriceCatalogue.json" "C:\Users\52899\Downloads\A5.2 Archivos de Apoyo\TC1\salesRecord.json"
Costo total de ventas: 2481.86
Tiempo de Ejecución: 0.001986 segundos
PS C:\Users\52899\Downloads\A5.2 Archivos de Apoyo>
```

TC2

- Costo total de ventas: 169478.22
- Tiempo de ejecución: 0.001067 segundos
- Errores detectados:
 - Cantidad inválida para Fresh blueberries:-35
 - Cantidad inválida para Green smoothie:-123
- Observaciones:
 - El programa identificó correctamente cantidades negativas y reportó los errores.
 - A pesar de los errores, el programa continuó ejecutándose sin interrupciones.

```
PS C:\Users\52899\Downloads\A5.2 Archivos de Apoyo> python computeSales.py PriceCatalogue.json "C:\Users\52899\Downloads\A5.2 Archivos de Apoyo\TC2\TC2.Sales.json"
Costo total de ventas: 169478.22
Tiempo de Ejecución: 0.001067 segundos
Errores encontrados:
- Cantidad inválida para Fresh blueberries: -35
- Cantidad inválida para Green smoothie: -123
PS C:\Users\52899\Downloads\A5.2 Archivos de Apoyo>
```

TC3

- **Costo total de ventas:** 168145.36
- **Tiempo de ejecución:** 0.002035 segundos
- **Errores detectados:**
 - Producto no encontrado: Elotes
 - Producto no encontrado: Frijoles
 - Cantidad inválida para Fresh blueberries:-35
 - Cantidad inválida para Green smoothie:-123
- **Observaciones:**
 - Además de cantidades inválidas, se encontraron productos que no estaban en el catálogo.
 - El programa manejó ambos tipos de errores correctamente y generó un reporte detallado.

```
PS C:\Users\52899\Downloads\A5.2 Archivos de Apoyo> python computeSales.py PriceCatalogue.json "C:\Users\52899\Downloads\A5.2 Archivos de Apoyo\TC3\TC3.Sales.json"
Costo total de ventas: 168145.36
Tiempo de Ejecución: 0.002035 segundos
Errores encontrados:
- Producto no encontrado: Elotes
- Cantidad inválida para Fresh blueberries: -35
- Producto no encontrado: Frijoles
- Cantidad inválida para Green smoothie: -123
PS C:\Users\52899\Downloads\A5.2 Archivos de Apoyo> |
```

Conclusión

1. **Funcionamiento Correcto:**
 - El programa ejecutó con éxito los tres casos de prueba, demostrando su capacidad para calcular ventas y detectar errores sin interrupciones.
2. **Manejo de Errores:**
 - Identificó y reportó cantidades inválidas (negativas) en TC2 y TC3.
 - Detectó productos inexistentes en TC3, lo que asegura una validación adecuada de los datos.
3. **Eficiencia:**
 - El tiempo de ejecución fue muy bajo en todas las pruebas, lo que demuestra que el programa es eficiente incluso con grandes volúmenes de datos.
4. **Sugerencias de Mejora:**

- **Mejor manejo de errores:** En lugar de solo reportar errores, podría implementarse una opción para corregirlos automáticamente.
- **Log de errores:** Guardar errores en un archivo separado podría facilitar su revisión.
- **Validación previa:** Antes de procesar los datos, validar la existencia de productos y valores numéricos podría evitar errores tempranos.