



Pruebas de Software y aseguramiento de la calidad

Actividad 5.2. Ejercicio de programación 2 y análisis estático

Alumno: Roberto Perezcano Hernández

Matrícula: A01730502

Fecha: 15 de febrero de 2026

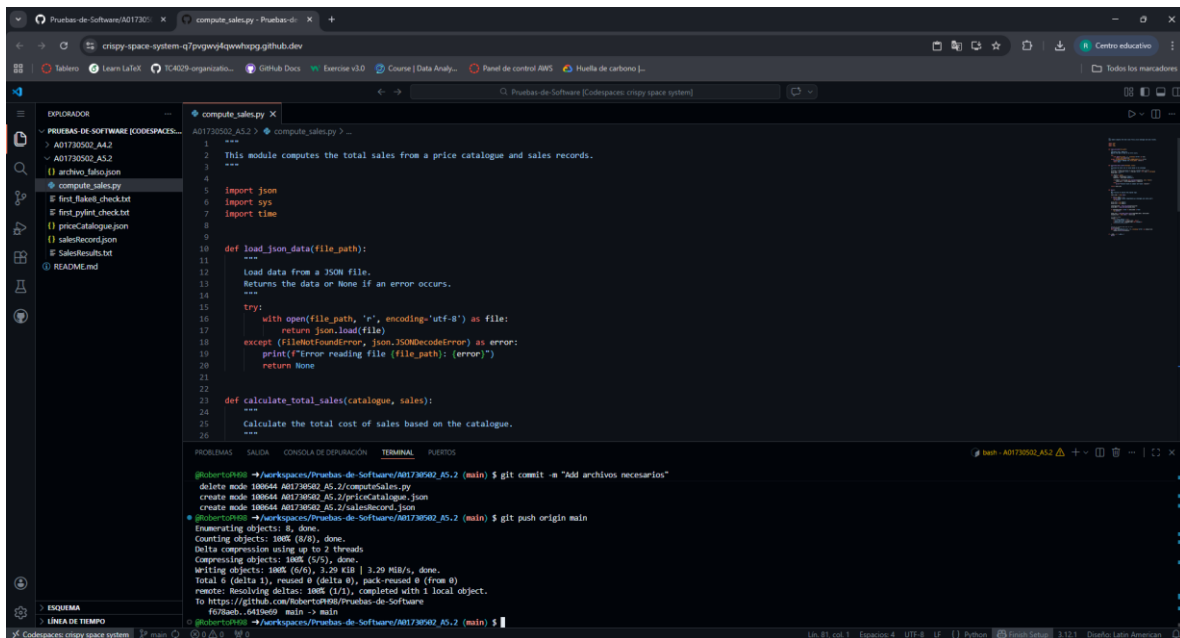
Contenido

Objetivos.....	3
Creación del programa Python para la actividad.....	3
Análisis Estático (Prevención de Defectos).....	3
Caso de Prueba 1: Ejecución Estándar Completa	3
Caso de Prueba 2: Manejo de Datos Inválidos (Req 3)	4
Caso de Prueba 3: Robustez de Archivos (Manejo de Excepciones)	4
Conclusión	4

Objetivos

- Explicar la diferencia entre pruebas dinámicas y pruebas estáticas
- Describir los beneficios e impacto de la calidad de las prácticas asociadas a pruebas estáticas.
- Explicar el origen de las inspecciones como herramienta de pruebas estáticas.
- Describir las diferencias entre revisiones informales, caminatas estructuradas, inspecciones e inspecciones automáticas.
- Describir la relación de las herramientas de análisis estático y el código fuente.
- Experimentar con el uso de herramientas de análisis estático en el código fuente.

Creación del programa Python para la actividad



```
1 """
2 This module computes the total sales from a price catalogue and sales records.
3 """
4
5 import json
6 import sys
7 import time
8
9
10 def load_json_data(file_path):
11     """
12     Load data from a JSON file.
13     Returns the data or None if an error occurs.
14     """
15     try:
16         with open(file_path, 'r', encoding='utf-8') as file:
17             return json.load(file)
18     except (FileNotFoundException, json.JSONDecodeError) as error:
19         print(f"Error reading file {file_path}: {error}")
20         return None
21
22
23 def calculate_total_sales(catalogue, sales):
24     """
25     Calculate the total cost of sales based on the catalogue.
26     """
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

```
@RobertoPH98 ➔ /workspaces/Pruebas-de-Software/A01730502_A5.2 (main) $ git commit -m "Add archivos necesarios"
delete mode 100644 A01730502_A5.2/computeSales.py
create mode 100644 A01730502_A5.2/priceCatalogue.json
create mode 100644 A01730502_A5.2/salesRecord.json
@RobertoPH98 ➔ /workspaces/Pruebas-de-Software/A01730502_A5.2 (main) $ git push origin main
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 2 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 3.20 KiB | 3.20 MiB/s, done.
Total 6 (delta 1), reused 0 (delta 0), pack-reused 8 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/RobertoPH98/Pruebas-de-Software
   627aeb..6529d9  main -> main
```

Análisis Estático (Prevención de Defectos)

Se aplicaron herramientas de análisis estático sobre el código fuente para garantizar la adherencia a estándares industriales.

Caso de Prueba 1: Ejecución Estándar Completa

Validar que el programa procese correctamente archivos JSON válidos con cientos de ítems.

```
@RobertoPH98 ➔ /workspaces/Pruebas-de-Software/A01730502_A5.2 (main) $ python compute_sales.py priceCatalogue.json salesRecord.json
--- Sales Report ---
Total Sales Cost: $2,481.86
Execution Time: 0.000294 seconds
```

Caso de Prueba 2: Manejo de Datos Inválidos (Req 3)

Verificar que el programa maneje productos no encontrados en el catálogo sin detener la ejecución

```
@RobertoPH98 →/workspaces/Pruebas-de-Software/A01730502_A5.2 (main) $ python compute_sales.py priceCatalogue.json salesRecord.json
Invalid record or product not found: {'SALE_ID': 1, 'SALE_Date': '01/12/23', 'Product': 'Raw', 'Quantity': 1}
--- Sales Report ---
Total Sales Cost: $2,464.75
Execution Time: 0.000317 seconds
-----
```

Caso de Prueba 3: Robustez de Archivos (Manejo de Excepciones)

Validar el comportamiento del sistema ante la ausencia de archivos de entrada.

```
@RobertoPH98 →/workspaces/Pruebas-de-Software/A01730502_A5.2 (main) $ python compute_sales.py archivo_falso.json salesRecord.json
Invalid record or product not found: {'SALE_ID': 9, 'SALE_Date': '02/12/23', 'Product': 'Pears juice', 'Quantity': 1}
Invalid record or product not found: {'SALE_ID': 9, 'SALE_Date': '02/12/23', 'Product': 'Green smoothie', 'Quantity': 8}
Invalid record or product not found: {'SALE_ID': 9, 'SALE_Date': '02/12/23', 'Product': 'Cuban sandwich', 'Quantity': 9}
Invalid record or product not found: {'SALE_ID': 10, 'SALE_Date': '02/12/23', 'Product': 'Hazel nut in black ceramic bowl', 'Quantity': 2}
Invalid record or product not found: {'SALE_ID': 10, 'SALE_Date': '02/12/23', 'Product': 'Tomatoes', 'Quantity': 1}
Invalid record or product not found: {'SALE_ID': 10, 'SALE_Date': '02/12/23', 'Product': 'Plums', 'Quantity': 2}
Invalid record or product not found: {'SALE_ID': 10, 'SALE_Date': '02/12/23', 'Product': 'Fresh blueberries', 'Quantity': 3}
Invalid record or product not found: {'SALE_ID': 10, 'SALE_Date': '02/12/23', 'Product': 'Green smoothie', 'Quantity': 5}
Invalid record or product not found: {'SALE_ID': 10, 'SALE_Date': '02/12/23', 'Product': 'Corn', 'Quantity': 4}
--- Sales Report ---
Total Sales Cost: $0.00
Execution Time: 0.000600 seconds
-----
```

Conclusión

El uso de herramientas de análisis estático como Flake8 y Pylint permitió identificar y corregir problemas de estilo y posibles errores lógicos sin necesidad de ejecutar el programa.

Esto demuestra que la detección temprana de defectos reduce el impacto en la calidad final del producto.

La implementación del código bajo el estándar PEP 8 asegura que el software sea legible, mantenible y profesional, cumpliendo con las convenciones reconocidas en la industria del desarrollo con Python.

El uso de GIT y GitHub para organizar el trabajo en carpetas específicas y realizar commits estructurados garantizó la integridad de los artefactos y el control de versiones, elementos críticos para la administración de procesos de software.

Por último, la ejecución de casos de prueba verificó que el programa computeSales.py no solo fuera estéticamente correcto, sino también robusto en el manejo de errores y eficiente en el procesamiento de datos a gran escala.