



# **Instituto Tecnológico y de Estudios Superiores de Monterrey**



## **Pruebas de Software y Aseguramiento de la Calidad**

### **Análisis de problemas: 5.2 Ejercicio de programación 2 y análisis estático**

Julio Adrián Quintana García - A01793661

## Objetivo(s)

- 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.

## Instrucciones

En esta actividad vas a generar un repositorio para este ejercicio de programación en GIT.

Revisa las indicaciones para los programas a implementar:

1. Implementa los programas indicados al final del documento usando el lenguaje Python.
2. Sigue el estándar de codificación PEP-8.
3. Verifica la correcta ejecución de tus programas generando pruebas de cada ejercicio usando los recursos indicados. Documenta los resultados.
4. Instala el paquete flake8 usando PIP, <https://luminousmen.com/post/python-static-analysis-tools>.
5. Si tienes dudas del uso, revisa el tutorial de Flake8: <https://flake8.pycqa.org/en/latest/>.
6. Verifica que tus programas no generen errores o problemas usando pylint.  
The error code of flake8 are:
  - E\*\*\*/W\*\*\*: Errors and warnings of pycodestyle
  - F\*\*\*: Detections of PyFlakes
  - C9\*\*\*: Detections of circulate complexity by McCabe-script
7. Arregla todos los detalles que encontraste flake8 y verifica que tu programa sigue funcionando correctamente.
8. Al terminar carga tus programas en el repositorio personal que generaste. El proyecto deberá de llamarse: Matrícula de estudiante\_Número de actividadA5.2
9. Sube la liga del repositorio en la tarea de Canvas.

10. Sube los archivos fuente de la tarea a Canvas.

### Actividad 5.2. Ejercicio de programación 2

Programming Exercise	Description	Practice	Test Cases and Evidence
1. Compute sales	<p>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</p> <p>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.</p> <p>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.</p> <p>Req 4. The name of the program shall be computeSales.py</p> <p>Req 5. The minimum format to invoke the program shall be as follows: python computeSales.py priceCatalogue.json salesRecord.json</p> <p>Req 6. The program shall manage files having from hundreds of items to thousands of items.</p> <p>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.</p> <p>Req 8. Be compliant with PEP8.</p>	<ul style="list-style-type: none"><li>•Control structures</li><li>Console Input output</li><li>• Mathematical computation</li><li>• File management</li><li>• Error handling</li></ul>	Record the execution. Use files included in the assignment

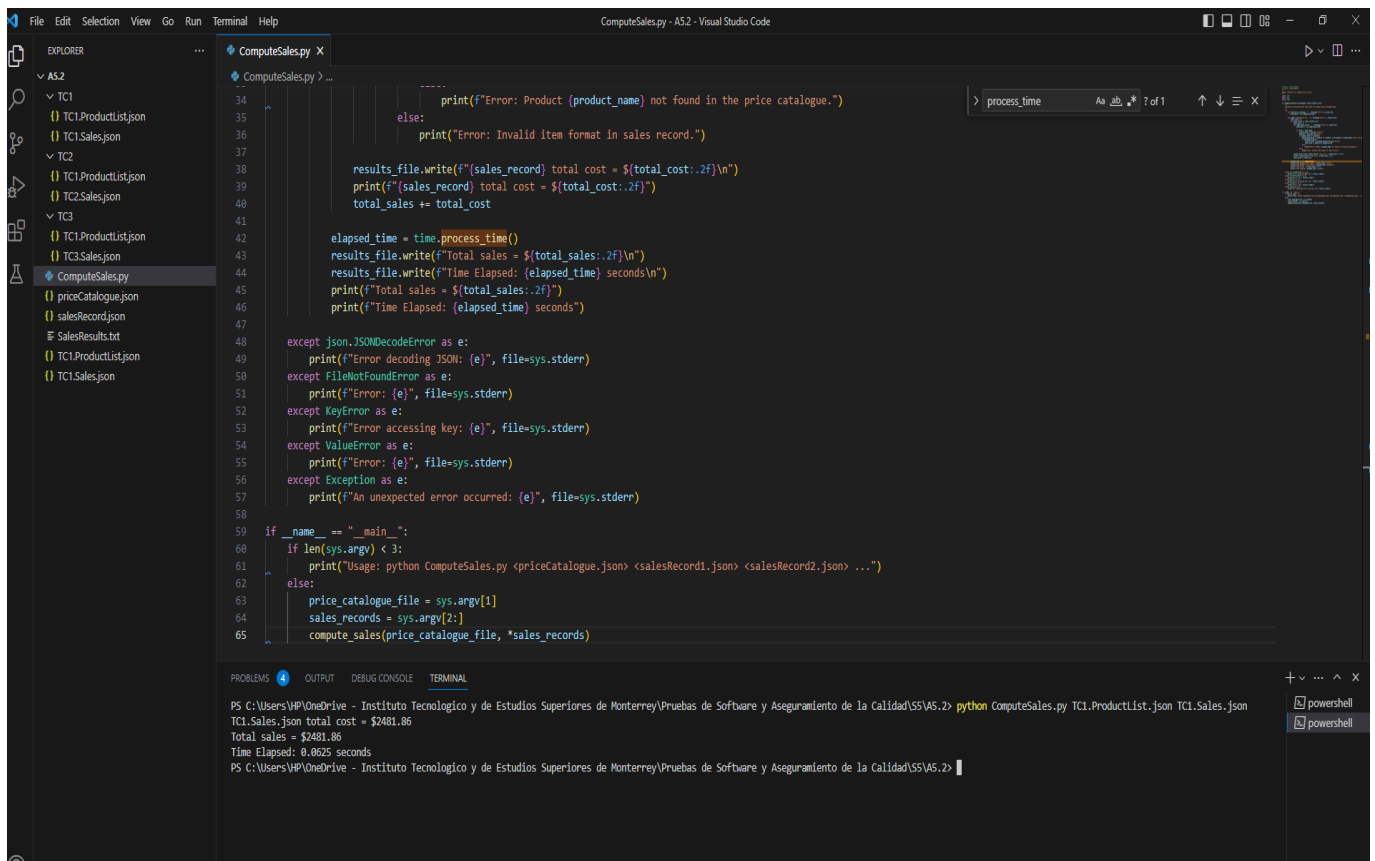
## Proceso de Desarrollo

El proceso de desarrollo del script "ComputeSales.py" ha abarcado la implementación de pruebas estáticas y dinámicas para garantizar la calidad del código. Las pruebas dinámicas implican la ejecución del código, mientras que las estáticas analizan el código sin ejecutarlo, detectando posibles defectos.

La calidad de las prácticas estáticas, como las inspecciones y revisiones, impacta significativamente en el ciclo de desarrollo al identificar errores tempranamente, reduciendo costos y tiempos de corrección.

Las herramientas de análisis estático, empleadas en el proceso, automatizan la detección de problemas, incluyendo defectos, vulnerabilidades y violaciones de estándares. Estas herramientas proporcionan retroalimentación instantánea, permitiendo a los desarrolladores mejorar la fiabilidad y mantenibilidad del código.

En resumen, el desarrollo de "ComputeSales.py" ha incorporado prácticas de análisis estático, demostrando su efectividad para garantizar la calidad y confiabilidad del software mediante la detección y corrección proactiva de errores.



The image shows a screenshot of the Visual Studio Code editor with the file "ComputeSales.py" open. The editor has a dark theme. On the left, the Explorer sidebar shows a project structure with folders "AS2", "TC1", "TC2", and "TC3", each containing "ProductList.json" and "Sales.json" files. The "ComputeSales.py" file is selected in the Explorer. The main editor area displays the Python code for "ComputeSales.py". The code includes logic for processing sales records, calculating total costs, and handling various exceptions like JSON decoding errors, file not found, key errors, and value errors. It also includes a main function that takes command-line arguments for the input files. At the bottom, the Output window shows the execution results of the script, including the total sales and the time elapsed.

```
34         print(f"Error: Product {product_name} not found in the price catalogue.")
35     else:
36         print("Error: Invalid item format in sales record.")
37
38     results_file.write(f"(sales_record) total cost = ${total_cost:.2f}\n")
39     print(f"(sales_record) total cost = ${total_cost:.2f}")
40     total_sales += total_cost
41
42     elapsed_time = time.process_time()
43     results_file.write(f"Total sales = ${total_sales:.2f}\n")
44     results_file.write(f"Time Elapsed: {elapsed_time} seconds\n")
45     print(f"Total sales = ${total_sales:.2f}")
46     print(f"Time Elapsed: {elapsed_time} seconds")
47
48 except json.JSONDecodeError as e:
49     print(f"Error decoding JSON: {e}", file=sys.stderr)
50 except FileNotFoundError as e:
51     print(f"Error: {e}", file=sys.stderr)
52 except KeyError as e:
53     print(f"Error accessing key: {e}", file=sys.stderr)
54 except ValueError as e:
55     print(f"Error: {e}", file=sys.stderr)
56 except Exception as e:
57     print(f"An unexpected error occurred: {e}", file=sys.stderr)
58
59 if __name__ == "__main__":
60     if len(sys.argv) < 3:
61         print("Usage: python ComputeSales.py <priceCatalogue.json> <salesRecord1.json> <salesRecord2.json> ...")
62     else:
63         price_catalogue_file = sys.argv[1]
64         sales_records = sys.argv[2:]
65         compute_sales(price_catalogue_file, *sales_records)
```

Output window content:

```
PS C:\Users\VP\OneDrive - Instituto Tecnológico y de Estudios Superiores de Monterrey\Pruebas de Software y Aseguramiento de la Calidad\SS\AS.2> python ComputeSales.py TC1.ProductList.json TC1.Sales.json
TC1.Sales.json total cost = $2481.86
Total sales = $2481.86
Time Elapsed: 0.0625 seconds
PS C:\Users\VP\OneDrive - Instituto Tecnológico y de Estudios Superiores de Monterrey\Pruebas de Software y Aseguramiento de la Calidad\SS\AS.2>
```

Imagen 1.1 – Compute Sales

## **Bibliografía:**

- PEP 0 – Index of Python Enhancement Proposals (PEPS) | Peps.python.org. (s. f.-b). <https://peps.python.org/>
- Python JSON Archives. (2021, 8 diciembre). PYnative. <https://pynative.com/python/json/>
- The Python tutorial. (s. f.-b). Python documentation. <https://docs.python.org/3/tutorial/index.html>
- Python Static Analysis Tools. (s. f.). Blog | iamluminousmen. <https://luminousmen.com/post/python-static-analysis-tools>