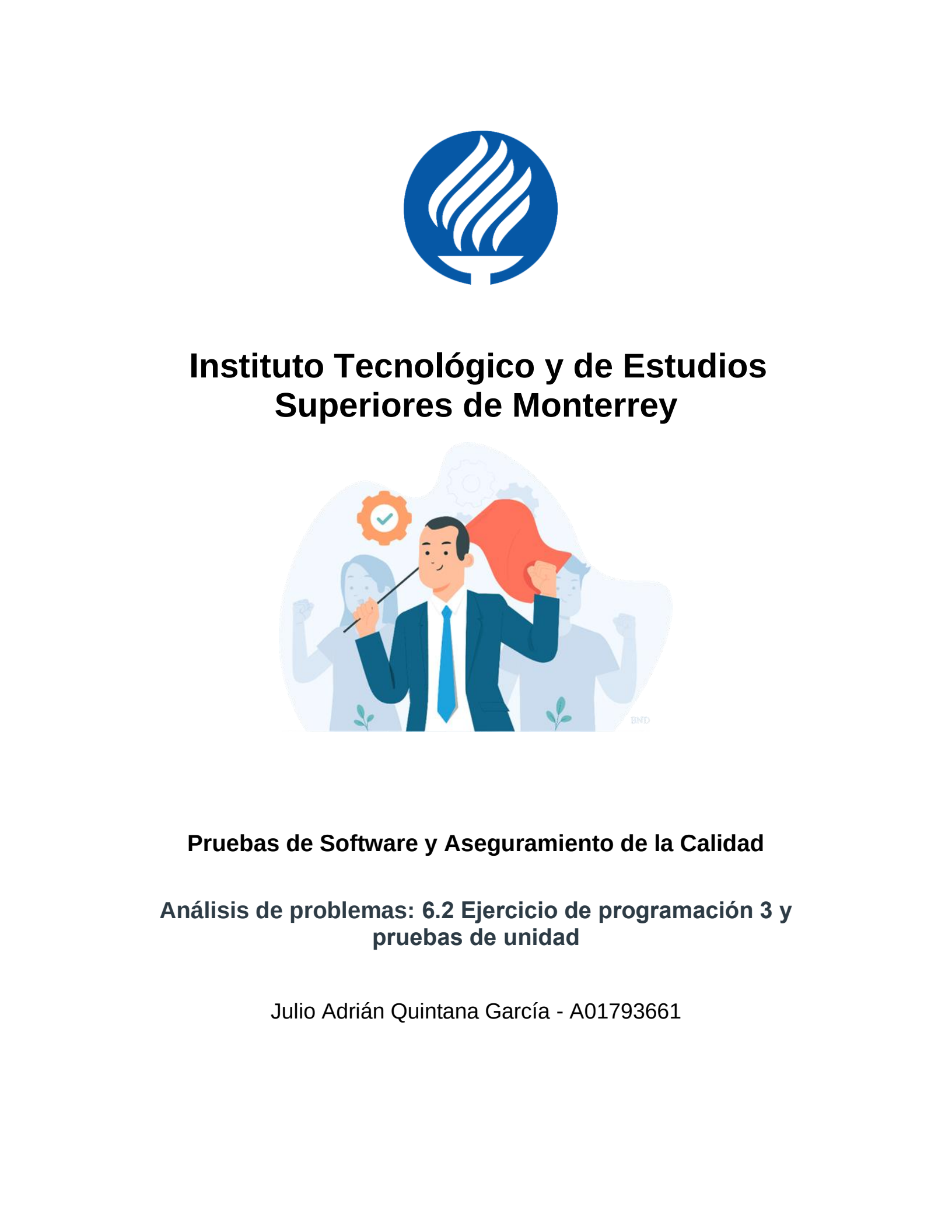

Instituto Tecnológico y de Estudios Superiores de Monterrey

Pruebas de Software y Aseguramiento de la Calidad

Análisis de problemas: 6.2 Ejercicio de programación 3 y pruebas de unidad

Julio Adrián Quintana García - A01793661

Objetivo(s)

- Explicar los fundamentos del desarrollo de pruebas unitarias
- 2.19 Desarrollar pruebas unitarias para fragmentos de programas usando las mejores prácticas recomendadas.

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. Genera un repositorio para los ejercicios de programación en GIT.
2. Revisa las indicaciones para los programas a implementar.
3. Implementa los programas, indicados al final del documento, usando el lenguaje Python.
4. Sigue el estándar de codificación PEP-8.
5. Verifica la correcta ejecución de tus programas generando pruebas de cada ejercicio usando los recursos indicados. Documenta los resultados.
6. Instala el paquete flake8 usando PIP, https://luminousmen.com/post/python-static-analysis-tools.
7. Si tienes dudas del uso, revisa el tutorial de Flake8: https://flake8.pycqa.org/en/latest/.
8. 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
9. Arregla todos los detalles que encontraste flake8 y verifica que tu programa sigue funcionando correctamente.
10. Al terminar carga tus programas en el repositorio personal que generaste. El proyecto deberá de llamarse: Matrícula de estudiante_Número de actividadA6.2
11. Sube la liga del repositorio en la tarea de Canvas.
12. Sube los archivos fuente de la tarea a Canvas.

Actividad 6.2. Ejercicio de programación 3

Programming Exercise	Description	Practice	Test Cases and Evidence
1. Reservation System	Req 1. Implement a set of classes in Python that implements two abstractions: 1. Hotel 2. Reservation 3. Customers Req 2. Implement a set of methods to handle the next persistent behaviors (stored in files): 1. Hotels a. Create Hotel b. Delete Hotel c. Display Hotel information d. Modify Hotel Information e. Reserve a Room f. Cancel a Reservation 2. Customer a. Create Customer b. Delete a Customer c. Display Customer Information d. Modify Customer Information 3. Reservation a. Create a Reservation (Customer, Hotel) b. Cancel a Reservation You are free to decide the attributes within each class that enable the required behavior 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 code coverage for all unit tests should accumulate at least 85% of line coverage. Req 5. 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 6. Be compliant with PEP8.	•Control structures Console Input output • Mathematical computation • File management • Error handling	Record the execution. Use files included in the assignment

	Req 7. The source code must show no warnings using Fleak and PyLint.		

Proceso de Desarrollo

Explicación de los fundamentos del desarrollo de pruebas unitarias:

Las pruebas unitarias son esenciales en el desarrollo de software, ya que permiten verificar el comportamiento correcto de pequeñas unidades de código de forma aislada. Se escriben casos de prueba que evalúan el funcionamiento esperado de estas unidades, lo que mejora la calidad del software al identificar y corregir errores temprano en el ciclo de desarrollo.

Prueba unitaria para la clase Reserva:

Una prueba puede verificar si el método __str__ de la clase Reserva devuelve la representación correcta de la reserva.

Prueba unitaria para la clase Cliente:

Se puede realizar una prueba para el método to_dict de la clase Cliente para garantizar que la conversión del cliente a un diccionario sea precisa.

Prueba unitaria para la clase ManejadorHoteles:

Una prueba puede validar si el método crear_hotel del ManejadorHoteles puede crear correctamente un nuevo hotel.

Prueba unitaria para la clase Hotel:

Se puede crear una prueba para el método agregar_reserva de la clase Hotel para asegurar que las reservas se añadan correctamente.

Prueba unitaria para la clase ManejadorClientes:

Una prueba puede confirmar que el método crear_cliente del ManejadorClientes puede crear un nuevo cliente correctamente.

Análisis de Errores de Pylint – PEP 8:

Los errores de Pylint, como "line too long" y "too-few-public-methods", señalan incumplimientos de las convenciones de estilo definidas en PEP 8, lo que mejora la legibilidad y mantenibilidad del código.

Análisis de Errores de Flake:

Los errores de Flake, como "line too long", indican que una línea de código excede la longitud máxima permitida según las convenciones de estilo, lo que puede afectar la claridad del código.

Correcto Diseño de Casos de Prueba. Incluir casos negativos:

Es importante diseñar casos de prueba que cubran tanto los escenarios esperados como los inesperados. Incluir casos negativos garantiza que el código maneje adecuadamente situaciones inesperadas, mejorando la robustez y fiabilidad del software.

Cobertura de líneas por clase:

Se debe apuntar a una cobertura de al menos el 85% por clase, asegurando que la mayoría de las líneas de código sean ejecutadas durante las pruebas unitarias para garantizar una mayor confiabilidad y calidad del software.

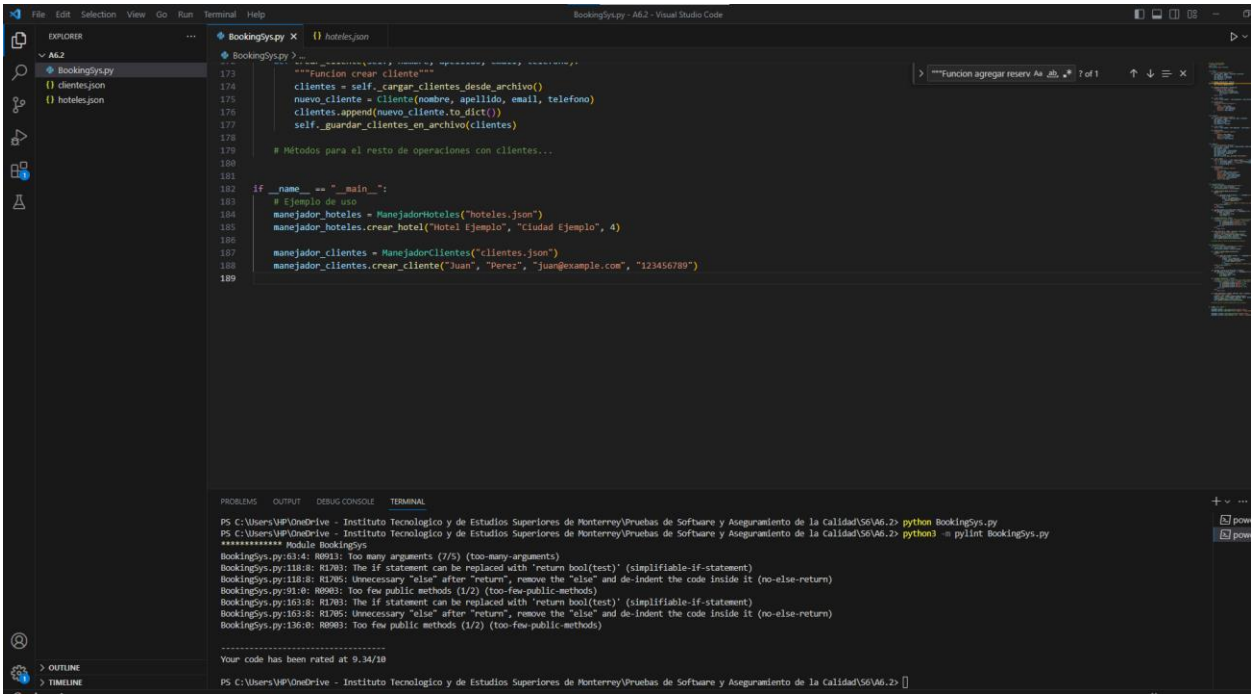

Imagen 1.1 – Primeros errores pylint

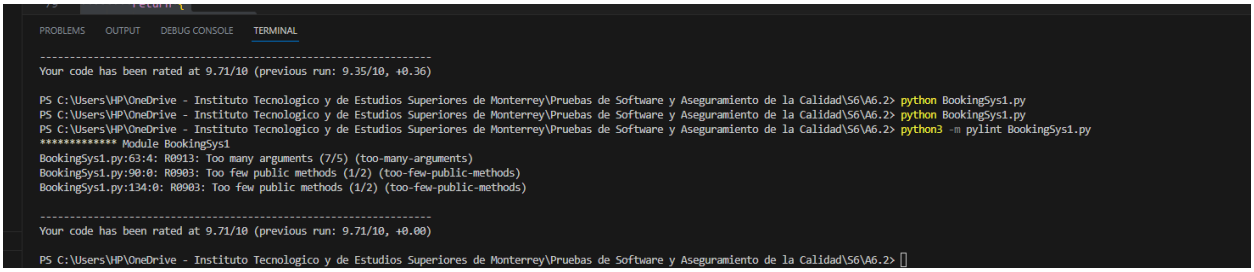

Imagen 1.2 – proceso de corrección de errores pylint

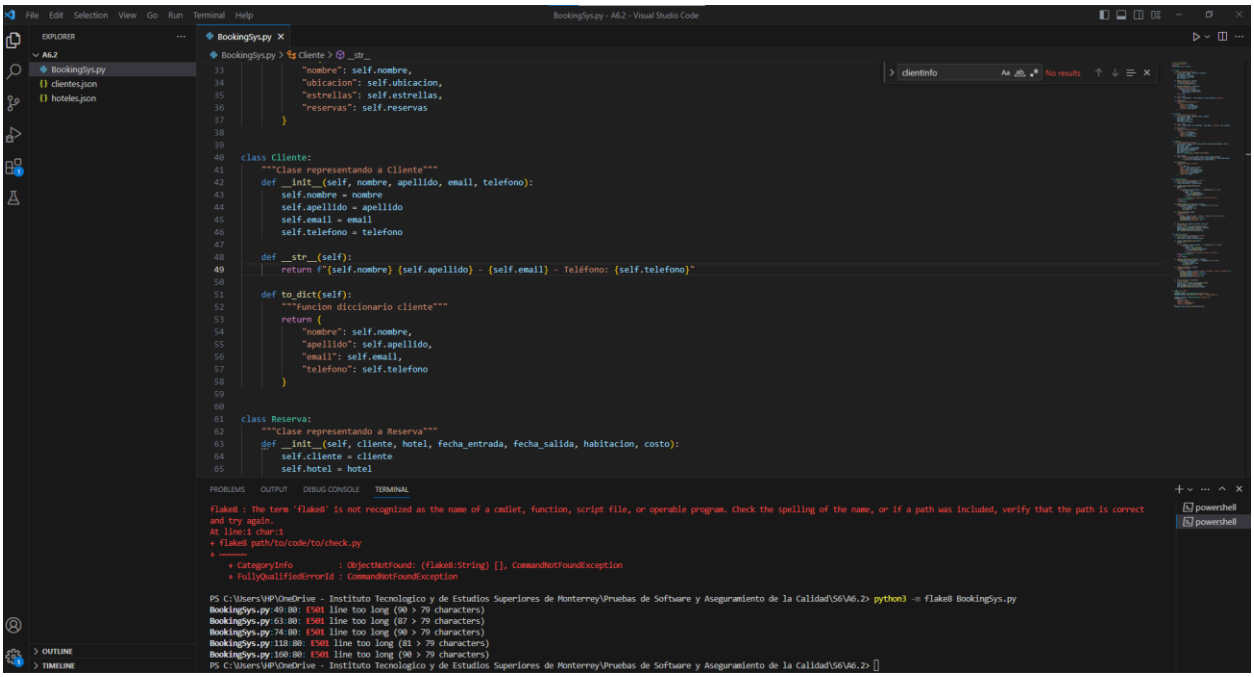

Imagen 1.3 – proceso de corrección de errores Flake8

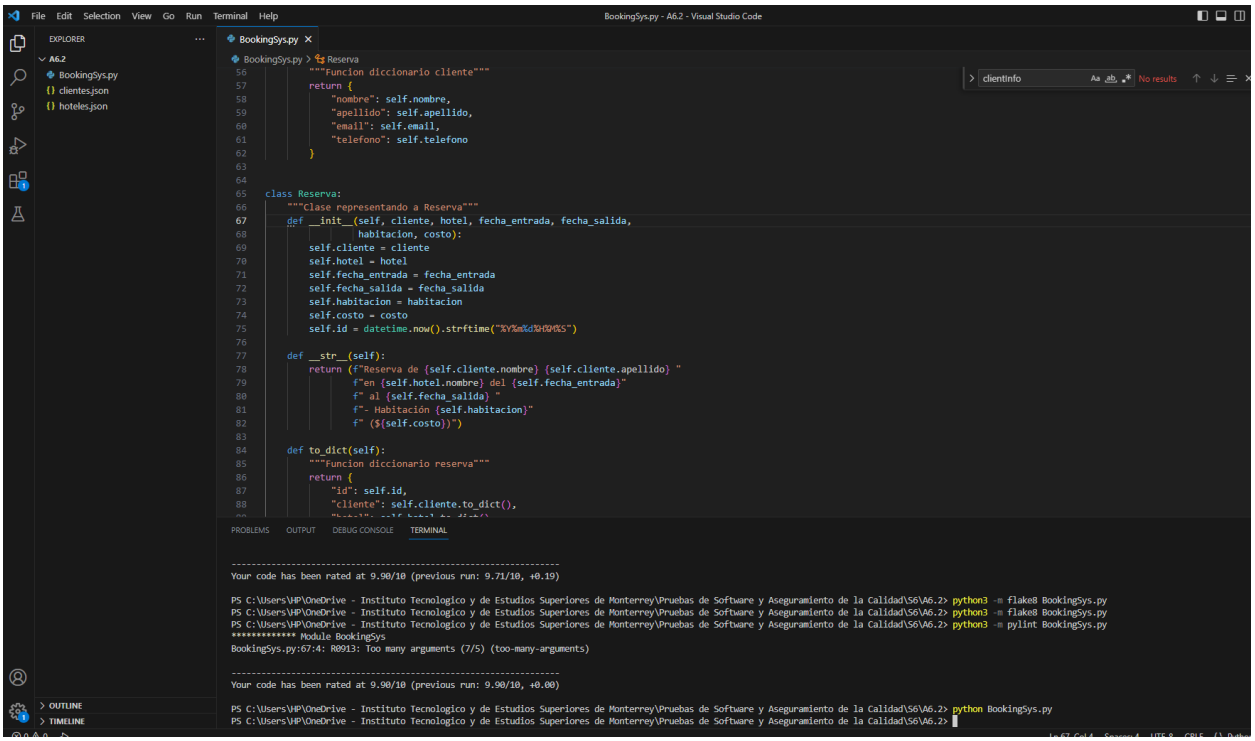

Imagen 1.4 – proceso de corrección de errores pylint y flake8

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