Tecnológico de Monterrey
Maestría en Inteligencia Artificial Aplicada

Aseguramiento de la Calidad de Software

Actividad 5.2

A01793818 Jorge Ariel Bermúdez Tellería

He realizado el código que han solicitado y me dio los resultados esperados:

In [10]:
```python
import json
import time
import sys

def load_data(file_path):
    try:
        with open(file_path, 'r') as file:
            data = json.load(file)
        return data
    except FileNotFoundError:
        print(f"Advertencia: El archivo {file_path} no existe. Se utilizarán valores predeterminados.")
        return {}  # Devolver un diccionario vacío en caso de que el archivo no exista
    except json.JSONDecodeError:
        print(f"Error: No se pudo decodificar el archivo {file_path}. Asegúrate de que sea un archivo JSON válido.")
        sys.exit(1)

def compute_total_cost(catalog, sales):
    total_cost = 0
    for sale in sales:
        product_name = sale['Product']
        quantity = sale['Quantity']

        # Buscar el producto en el catálogo por nombre
        matching_products = [product for product in catalog if product['title'] == product_name]

        if matching_products:
            product = matching_products[0]
            total_cost += product['price'] * quantity
        else:
            print(f"Advertencia: El producto {product_name} no se encuentra en el catálogo.")

    return total_cost


def main():
    # Asignar nombres de archivo directamente
    catalog_file = 'TC1/priceCatalogue_default.json'
    sales_file = 'TC1/salesRecord_default.json'

    start_time = time.time()

    catalog = load_data(catalog_file)
    sales = load_data(sales_file)

    total_cost = compute_total_cost(catalog, sales)

    end_time = time.time()
    elapsed_time = end_time - start_time

    # Imprimir resultados en pantalla
    print(f"Costo total de las ventas: {total_cost}")
    print(f"Tiempo transcurrido: {elapsed_time} segundos")

    # Guardar resultados en SalesResults.txt
    with open('SalesResults.txt', 'w') as result_file:
        result_file.write(f"Costo total de las ventas: {total_cost}\n")
        result_file.write(f"Tiempo transcurrido: {elapsed_time} segundos\n")

if __name__ == "__main__":
    main()
```

```
Costo total de las ventas: 2481.8600000000006
Tiempo transcurrido: 0.010554790496826172 segundos
```

```python
import json
import time
import sys

def load_data(file_path):
    try:
        with open(file_path, 'r') as file:
            data = json.load(file)
        return data
    except FileNotFoundError:
        print(f"Advertencia: El archivo {file_path} no existe. Se utilizarán valores predeterminados.")
        return {}  # Devolver un diccionario vacío en caso de que el archivo no exista
    except json.JSONDecodeError:
        print(f"Error: No se pudo decodificar el archivo {file_path}. Asegúrate de que sea un archivo JSON válido.")
        sys.exit(1)

def compute_total_cost(catalog, sales):
    total_cost = 0
    for sale in sales:
        product_name = sale['Product']
        quantity = sale['Quantity']

        # Buscar el producto en el catálogo por nombre
        matching_products = [product for product in catalog if product['title'] == product_name]

        if matching_products:
            product = matching_products[0]
            total_cost += product['price'] * quantity
        else:
            print(f"Advertencia: El producto {product_name} no se encuentra en el catálogo.")

    return total_cost


def main():
    # Asignar nombres de archivo directamente
    catalog_file = 'TC2/priceCatalogue_default.json'
    sales_file = 'TC2/salesRecord_default.json'

    start_time = time.time()

    catalog = load_data(catalog_file)
    sales = load_data(sales_file)

    total_cost = compute_total_cost(catalog, sales)

    end_time = time.time()
    elapsed_time = end_time - start_time

    # Imprimir resultados en pantalla
    print(f"Costo total de las ventas: {total_cost}")
    print(f"Tiempo transcurrido: {elapsed_time} segundos")

    # Guardar resultados en SalesResults.txt
    with open('SalesResults.txt', 'w') as result_file:
        result_file.write(f"Costo total de las ventas: {total_cost}\n")
        result_file.write(f"Tiempo transcurrido: {elapsed_time} segundos\n")

if __name__ == "__main__":
    main()
```

```
Costo total de las ventas: 166568.22999999998
Tiempo transcurrido: 0.02984333038330078 segundos
```

In [19]:

```python
import json
import time
import sys

def load_data(file_path):
    try:
        with open(file_path, 'r') as file:
            data = json.load(file)
        return data
    except FileNotFoundError:
        print(f"Advertencia: El archivo {file_path} no existe. Se utilizarán valores predeterminados.")
        return {}  # Devolver un diccionario vacío en caso de que el archivo no exista
    except json.JSONDecodeError:
        print(f"Error: No se pudo decodificar el archivo {file_path}. Asegúrate de que sea un archivo JSON válido.")
        sys.exit(1)

def compute_total_cost(catalog, sales):
    total_cost = 0
    for sale in sales:
        product_name = sale['Product']
        quantity = sale['Quantity']

        # Buscar el producto en el catálogo por nombre
        matching_products = [product for product in catalog if product['title'] == product_name]

        if matching_products:
            product = matching_products[0]
            total_cost += product['price'] * quantity
        else:
            print(f"Advertencia: El producto {product_name} no se encuentra en el catálogo.")

    return total_cost


def main():
    # Asignar nombres de archivo directamente
    catalog_file = 'TC3/priceCatalogue_default.json'
    sales_file = 'TC3/salesRecord_default.json'

    start_time = time.time()

    catalog = load_data(catalog_file)
    sales = load_data(sales_file)

    total_cost = compute_total_cost(catalog, sales)

    end_time = time.time()
    elapsed_time = end_time - start_time

    # Imprimir resultados en pantalla
    print(f"Costo total de las ventas: {total_cost}")
    print(f"Tiempo transcurrido: {elapsed_time} segundos")

    # Guardar resultados en SalesResults.txt
    with open('SalesResults.txt', 'w') as result_file:
        result_file.write(f"Costo total de las ventas: {total_cost}\n")
        result_file.write(f"Tiempo transcurrido: {elapsed_time} segundos\n")

if __name__ == "__main__":
    main()
```

```
Advertencia: El producto Elotes no se encuentra en el catálogo.
Advertencia: El producto Frijoles no se encuentra en el catálogo.
Costo total de las ventas: 165235.37
Tiempo transcurrido: 0.0017502307891845703 segundos
```

Luego he instalado flake8 y pylint, donde tuve muchos errores en flake8 y un puntaje de 5/10 en pylint.

```
In [11]: pip install flake8
```

```
Requirement already satisfied: flake8 in c:\users\usuario\anaconda3\lib\site-packages (3.9.2)
Requirement already satisfied: pyflakes<2.4.0,>=2.3.0 in c:\users\usuario\anaconda3\lib\site-packages (from flake8) (2.3.1)
Requirement already satisfied: mccabe<0.7.0,>=0.6.0 in c:\users\usuario\anaconda3\lib\site-packages (from flake8) (0.6.1)
Requirement already satisfied: pycodestyle<2.8.0,>=2.7.0 in c:\users\usuario\anaconda3\lib\site-packages (from flake8) (2.7.0)
Note: you may need to restart the kernel to use updated packages.
```

```
In [14]: !flake8 computeSales.py
```

```
computeSales.py:11:1: E302 expected 2 blank lines, found 1
computeSales.py:17:80: E501 line too long (103 > 79 characters)
computeSales.py:18:80: E501 line too long (86 > 79 characters)
computeSales.py:20:80: E501 line too long (116 > 79 characters)
computeSales.py:23:1: E302 expected 2 blank lines, found 1
computeSales.py:28:1: W293 blank line contains whitespace
computeSales.py:30:80: E501 line too long (96 > 79 characters)
computeSales.py:31:1: W293 blank line contains whitespace
computeSales.py:36:80: E501 line too long (93 > 79 characters)
computeSales.py:65:1: E305 expected 2 blank lines after class or function definition, found 1
computeSales.py:72:1: E402 module level import not at top of file
computeSales.py:72:1: F811 redefinition of unused 'json' from line 7
computeSales.py:73:1: E402 module level import not at top of file
computeSales.py:73:1: F811 redefinition of unused 'time' from line 8
computeSales.py:74:1: E402 module level import not at top of file
computeSales.py:74:1: F811 redefinition of unused 'sys' from line 9
computeSales.py:76:1: E302 expected 2 blank lines, found 1
computeSales.py:76:1: F811 redefinition of unused 'load_data' from line 11
computeSales.py:82:80: E501 line too long (103 > 79 characters)
computeSales.py:83:80: E501 line too long (86 > 79 characters)
computeSales.py:85:80: E501 line too long (116 > 79 characters)
computeSales.py:88:1: E302 expected 2 blank lines, found 1
computeSales.py:88:1: F811 redefinition of unused 'compute_total_cost' from line 23
computeSales.py:93:1: W293 blank line contains whitespace
computeSales.py:95:80: E501 line too long (96 > 79 characters)
computeSales.py:96:1: W293 blank line contains whitespace
computeSales.py:101:80: E501 line too long (93 > 79 characters)
computeSales.py:130:1: E305 expected 2 blank lines after class or function definition, found 1
computeSales.py:132:1: W391 blank line at end of file
```

In [15]:
```
!pip install pylint
```

```
Requirement already satisfied: pylint in c:\users\usuario\anaconda3\lib\site-packages (2.9.6)
Requirement already satisfied: isort<6,>=4.2.5 in c:\users\usuario\anaconda3\lib\site-packages (from pylint) (5.9.3)
Requirement already satisfied: colorama in c:\users\usuario\anaconda3\lib\site-packages (from pylint) (0.4.4)
Requirement already satisfied: mccabe<0.7,>=0.6 in c:\users\usuario\anaconda3\lib\site-packages (from pylint) (0.6.1)
Requirement already satisfied: toml>=0.7.1 in c:\users\usuario\anaconda3\lib\site-packages (from pylint) (0.10.2)
Requirement already satisfied: astroid<2.7,>=2.6.5 in c:\users\usuario\anaconda3\lib\site-packages (from pylint) (2.6.6)
Requirement already satisfied: lazy-object-proxy>=1.4.0 in c:\users\usuario\anaconda3\lib\site-packages (from astroid<2.7,>=2.
6.5->pylint) (1.6.0)
Requirement already satisfied: wrapt<1.13,>=1.11 in c:\users\usuario\anaconda3\lib\site-packages (from astroid<2.7,>=2.6.5->pyl
int) (1.12.1)
Requirement already satisfied: setuptools>=20.0 in c:\users\usuario\anaconda3\lib\site-packages (from astroid<2.7,>=2.6.5->pyli
nt) (61.2.0)
```

In [16]:
```
!pylint computeSales.py
```

```
************* Module computeSales
computeSales.py:17:0: C0301: Line too long (103/100) (line-too-long)
computeSales.py:20:0: C0301: Line too long (116/100) (line-too-long)
computeSales.py:28:0: C0303: Trailing whitespace (trailing-whitespace)
computeSales.py:31:0: C0303: Trailing whitespace (trailing-whitespace)
computeSales.py:82:0: C0301: Line too long (103/100) (line-too-long)
computeSales.py:85:0: C0301: Line too long (116/100) (line-too-long)
computeSales.py:93:0: C0303: Trailing whitespace (trailing-whitespace)
computeSales.py:96:0: C0303: Trailing whitespace (trailing-whitespace)
computeSales.py:132:0: C0305: Trailing newlines (trailing-newlines)
computeSales.py:1:0: C0103: Module name "computeSales" doesn't conform to snake_case naming style (invalid-name)
computeSales.py:1:0: C0114: Missing module docstring (missing-module-docstring)
computeSales.py:11:0: C0116: Missing function or method docstring (missing-function-docstring)
computeSales.py:23:0: C0116: Missing function or method docstring (missing-function-docstring)
computeSales.py:41:0: C0116: Missing function or method docstring (missing-function-docstring)
computeSales.py:72:0: W0404: Reimport 'json' (imported line 7) (reimported)
computeSales.py:72:0: C0413: Import "import json" should be placed at the top of the module (wrong-import-position)
computeSales.py:73:0: W0404: Reimport 'time' (imported line 8) (reimported)
computeSales.py:73:0: C0413: Import "import time" should be placed at the top of the module (wrong-import-position)
computeSales.py:74:0: W0404: Reimport 'sys' (imported line 9) (reimported)
computeSales.py:74:0: C0413: Import "import sys" should be placed at the top of the module (wrong-import-position)
computeSales.py:76:0: E0102: function already defined line 11 (function-redefined)
computeSales.py:76:0: C0116: Missing function or method docstring (missing-function-docstring)
computeSales.py:88:0: E0102: function already defined line 23 (function-redefined)
computeSales.py:88:0: C0116: Missing function or method docstring (missing-function-docstring)
computeSales.py:106:0: E0102: function already defined line 41 (function-redefined)
computeSales.py:106:0: C0116: Missing function or method docstring (missing-function-docstring)
computeSales.py:72:0: C0412: Imports from package json are not grouped (ungrouped-imports)
computeSales.py:73:0: C0412: Imports from package time are not grouped (ungrouped-imports)
computeSales.py:74:0: C0412: Imports from package sys are not grouped (ungrouped-imports)

-----------------------------------

Your code has been rated at 5.00/10
```

Hice varios cambios, entre ellos colocar documentación luego de cada definición.

In [25]:

```
"""Este módulo contiene funciones para calcular el costo total de las ventas."""

import json
import time
import sys

def load_data(file_path):
    """Carga datos desde un archivo JSON."""
    try:
        with open(file_path, 'r') as file:
            data = json.load(file)
        return data
    except FileNotFoundError:
        print(f"Advertencia: El archivo {file_path} no existe.Se utilizarán valores predeterminados.")
        return {}  # Devolver un diccionario vacío en caso de que el archivo no exista
    except json.JSONDecodeError:
        print(f"Error:No se decodifica el archivo {file_path}.Asegúrate de que sea un archivo JSON válido.")
        sys.exit(1)

def compute_total_cost(catalog, sales):
    total_cost = 0
    for sale in sales:
        product_name = sale['Product']
        quantity = sale['Quantity']

        # Buscar el producto en el catálogo por nombre
        matching_products = [product for product in catalog
                    if product['title'] == product_name]

        if matching_products:
            product = matching_products[0]
            total_cost += product['price'] * quantity
        else:
            print(f"Advertencia: El producto {product_name} no se encuentra en el catálogo.")

    return total_cost


def main():
    # Asignar nombres de archivo directamente
    catalog_file = 'TC3/priceCatalogue_default.json'
    sales_file = 'TC3/salesRecord_default.json'

    start_time = time.time()

    catalog = load_data(catalog_file)
    sales = load_data(sales_file)

    total_cost = compute_total_cost(catalog, sales)

    end_time = time.time()
    elapsed_time = end_time - start_time

    # Imprimir resultados en pantalla
    print(f"Costo total de las ventas: {total_cost}")
    print(f"Tiempo transcurrido: {elapsed_time} segundos")

    # Guardar resultados en SalesResults.txt
    with open('SalesResults.txt', 'w') as result_file:
        result_file.write(f"Costo total de las ventas: {total_cost}\n")
        result_file.write(f"Tiempo transcurrido: {elapsed_time} segundos\n")

if __name__ == "__main__":
    main()
```

```
Advertencia: El producto Elotes no se encuentra en el catálogo.
Advertencia: El producto Frijoles no se encuentra en el catálogo.
Costo total de las ventas: 165235.37
```

Ya en esta segunda inspección de calidad tuve un 8.05/10 en pylint y menos errores en flake8.

```
In [5]: !flake8 computeSales.py

computeSales.py:7:80: E501 line too long (80 > 79 characters)
computeSales.py:13:1: E302 expected 2 blank lines, found 1
computeSales.py:20:80: E501 line too long (103 > 79 characters)
computeSales.py:21:80: E501 line too long (86 > 79 characters)
computeSales.py:23:80: E501 line too long (110 > 79 characters)
computeSales.py:26:1: E302 expected 2 blank lines, found 1
computeSales.py:31:1: W293 blank line contains whitespace
computeSales.py:35:1: W293 blank line contains whitespace
computeSales.py:40:80: E501 line too long (93 > 79 characters)
computeSales.py:69:1: E305 expected 2 blank lines after class or function definition, found 1
computeSales.py:77:1: W391 blank line at end of file
```

```
In [4]: !pylint computeSales.py

************* Module computeSales
computeSales.py:20:0: C0301: Line too long (103/100) (line-too-long)
computeSales.py:23:0: C0301: Line too long (110/100) (line-too-long)
computeSales.py:31:0: C0303: Trailing whitespace (trailing-whitespace)
computeSales.py:35:0: C0303: Trailing whitespace (trailing-whitespace)
computeSales.py:77:0: C0305: Trailing newlines (trailing-newlines)
computeSales.py:1:0: C0103: Module name "computeSales" doesn't conform to snake_case naming style (invalid-name)
computeSales.py:26:0: C0116: Missing function or method docstring (missing-function-docstring)
computeSales.py:45:0: C0116: Missing function or method docstring (missing-function-docstring)

-------------------------------------------------------------------

Your code has been rated at 8.05/10 (previous run: 5.00/10, +3.05)
```

En la tercera mejora traté de disminuir la extensión de las líneas.

In [1]:

```python
"""Este módulo contiene funciones para calcular el costo total de las ventas."""

import json
import time
import sys

def load_data(file_path):
    """Carga datos desde un archivo JSON."""
    try:
        with open(file_path, 'r') as file:
            data = json.load(file)
        return data
    except FileNotFoundError:
        print(f"Advertencia: El archivo {file_path} no existe. Se utilizarán valores predeterminados.")
        return {}  # Devolver un diccionario vacío en caso de que el archivo no exista
    except json.JSONDecodeError:
        print(f"Error: No se decodifica el archivo {file_path}. Asegúrate de que sea un archivo JSON válido.")
        sys.exit(1)

def compute_total_cost(catalog, sales):
    """Calcula el costo total de las ventas."""
    total_cost = 0
    for sale in sales:
        product_name = sale['Product']
        quantity = sale['Quantity']

        # Buscar el producto en el catálogo por nombre
        matching_products = [product for product in catalog
                             if product['title'] == product_name]

        if matching_products:
            product = matching_products[0]
            total_cost += product['price'] * quantity
        else:
            print(f"Advertencia: El producto {product_name} no se encuentra en el catálogo.")

    return total_cost


def main():
    # Asignar nombres de archivo directamente
    catalog_file = 'TC1/priceCatalogue_default.json'
    sales_file = 'TC1/salesRecord_default.json'

    start_time = time.time()

    catalog = load_data(catalog_file)
    sales = load_data(sales_file)

    total_cost = compute_total_cost(catalog, sales)

    end_time = time.time()
    elapsed_time = end_time - start_time

    # Imprimir resultados en pantalla
    print(f"Costo total de las ventas: {total_cost}")
    print(f"Tiempo transcurrido: {elapsed_time} segundos")

    # Guardar resultados en SalesResults.txt
    with open('SalesResults.txt', 'w') as result_file:
        result_file.write(f"Costo total de las ventas: {total_cost}\n")
        result_file.write(f"Tiempo transcurrido: {elapsed_time} segundos\n")

if __name__ == "__main__":
    main()
```

```
Costo total de las ventas: 2481.8600000000006
Tiempo transcurrido: 0.0024068355560302734 segundos
```

Mejoró muy poco, solo subió a 8.29 en pylint.

In [7]:
```
!flake8 compute_Sales.py
```

```
compute_Sales.py:7:80: E501 line too long (80 > 79 characters)
compute_Sales.py:13:1: E302 expected 2 blank lines, found 1
compute_Sales.py:20:80: E501 line too long (103 > 79 characters)
compute_Sales.py:21:80: E501 line too long (86 > 79 characters)
compute_Sales.py:23:80: E501 line too long (110 > 79 characters)
compute_Sales.py:26:1: E302 expected 2 blank lines, found 1
compute_Sales.py:32:1: W293 blank line contains whitespace
compute_Sales.py:36:1: W293 blank line contains whitespace
compute_Sales.py:41:80: E501 line too long (93 > 79 characters)
compute_Sales.py:70:1: E305 expected 2 blank lines after class or function definition, found 1
compute_Sales.py:78:1: W391 blank line at end of file
```

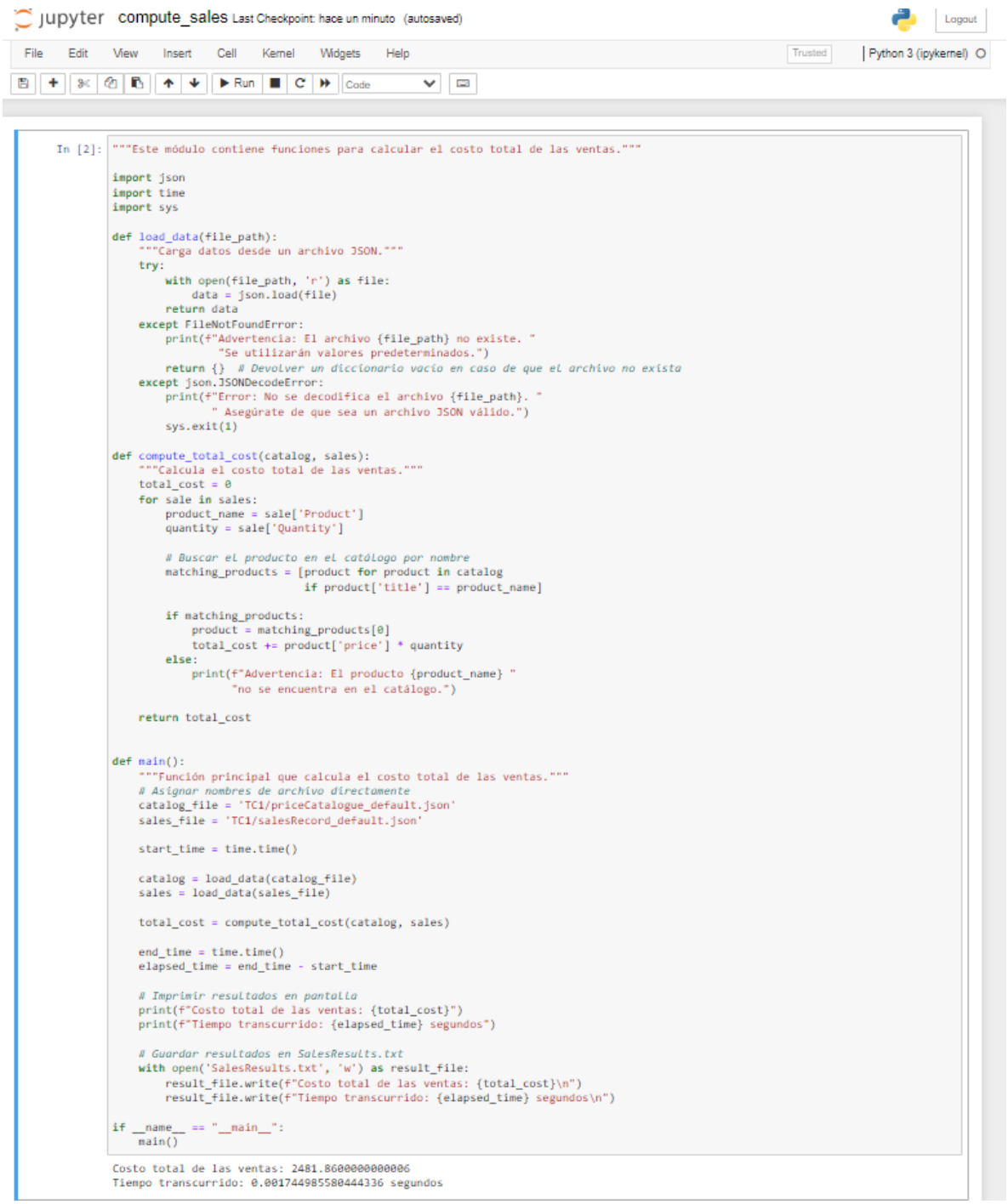
In [6]:
```
!pylint compute_Sales.py
```

```
************ Module compute_Sales
compute_Sales.py:20:0: C0301: Line too long (103/100) (line-too-long)
compute_Sales.py:23:0: C0301: Line too long (110/100) (line-too-long)
compute_Sales.py:32:0: C0303: Trailing whitespace (trailing-whitespace)
compute_Sales.py:36:0: C0303: Trailing whitespace (trailing-whitespace)
compute_Sales.py:78:0: C0305: Trailing newlines (trailing-newlines)
compute_Sales.py:1:0: C0103: Module name "compute_Sales" doesn't conform to snake_case naming style (invalid-name)
compute_Sales.py:46:0: C0116: Missing function or method docstring (missing-function-docstring)

-----------------------------------

Your code has been rated at 8.29/10
```

En la cuarta disminuí aún más algunas líneas que estaban aún más grande de la convención pep8, además le cambié el nombre al archivo ya que lo pasé de computeSales.py a compute_sales.py



Mejoro a 9.02 el puntaje de pylint, y menos errores en flake8.

```
In [11]: !flake8 compute_sales.py

compute_sales.py:7:80: E501 line too long (80 > 79 characters)
compute_sales.py:13:1: E302 expected 2 blank lines, found 1
compute_sales.py:21:17: E127 continuation line over-indented for visual indent
compute_sales.py:22:80: E501 line too long (86 > 79 characters)
compute_sales.py:25:16: E127 continuation line over-indented for visual indent
compute_sales.py:28:1: E302 expected 2 blank lines, found 1
compute_sales.py:34:1: W293 blank line contains whitespace
compute_sales.py:38:1: W293 blank line contains whitespace
compute_sales.py:74:1: E305 expected 2 blank lines after class or function definition, found 1
compute_sales.py:82:1: W391 blank line at end of file
```

```
In [10]: !pylint compute_sales.py

************* Module compute_Sales
compute_sales.py:34:0: C0303: Trailing whitespace (trailing-whitespace)
compute_sales.py:38:0: C0303: Trailing whitespace (trailing-whitespace)
compute_sales.py:82:0: C0305: Trailing newlines (trailing-newlines)
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 9.02/10 (previous run: 8.78/10, +0.24)
```

Eliminé la línea del final y coloqué dos líneas entre cada función.

In [3]:

```
"""Este módulo contiene funciones para calcular el costo total de las ventas."""

import json
import time
import sys


def load_data(file_path):
    """Carga datos desde un archivo JSON."""
    try:
        with open(file_path, 'r') as file:
            data = json.load(file)
        return data
    except FileNotFoundError:
        print(f"Advertencia: El archivo {file_path} no existe. "
              "Se utilizarán valores predeterminados.")
        return {}  # Devolver un diccionario vacío en caso de que el archivo no exista
    except json.JSONDecodeError:
        print(f"Error: No se decodifica el archivo {file_path}. "
              " Asegúrate de que sea un archivo JSON válido.")
        sys.exit(1)


def compute_total_cost(catalog, sales):
    """Calcula el costo total de las ventas."""
    total_cost = 0
    for sale in sales:
        product_name = sale['Product']
        quantity = sale['Quantity']

        # Buscar el producto en el catálogo por nombre
        matching_products = [product for product in catalog
                             if product['title'] == product_name]

        if matching_products:
            product = matching_products[0]
            total_cost += product['price'] * quantity
        else:
            print(f"Advertencia: El producto {product_name} "
                  "no se encuentra en el catálogo.")

    return total_cost


def main():
    """Función principal que calcula el costo total de las ventas."""
    # Asignar nombres de archivo directamente
    catalog_file = 'TC1/priceCatalogue_default.json'
    sales_file = 'TC1/salesRecord_default.json'

    start_time = time.time()

    catalog = load_data(catalog_file)
    sales = load_data(sales_file)

    total_cost = compute_total_cost(catalog, sales)

    end_time = time.time()
    elapsed_time = end_time - start_time

    # Imprimir resultados en pantalla
    print(f"Costo total de las ventas: {total_cost}")
    print(f"Tiempo transcurrido: {elapsed_time} segundos")

    # Guardar resultados en SalesResults.txt
    with open('SalesResults.txt', 'w') as result_file:
        result_file.write(f"Costo total de las ventas: {total_cost}\n")
        result_file.write(f"Tiempo transcurrido: {elapsed_time} segundos\n")


if __name__ == "__main__":
    main()
```

```
Costo total de las ventas: 2481.8600000000006
Tiempo transcurrido: 0.0019979476928710938 segundos
```

Ya con esto me quedaron solo tres errores en flake8 y uno en pylint, aunque en ambos no se quitaron los errores, aunque los quité. Terminé con un puntaje de 9.76 en pylint.

```
In [2]: !flake8 compute_sales.py
compute_sales.py:7:80: E501 line too long (80 > 79 characters)
compute_sales.py:23:80: E501 line too long (86 > 79 characters)
compute_sales.py:85:1: W391 blank line at end of file
```

```
In [1]: !pylint compute_sales.py
************* Module compute_sales
compute_sales.py:85:0: C0305: Trailing newlines (trailing-newlines)

--------------------------------------------------------------------

Your code has been rated at 9.76/10 (previous run: 9.02/10, +0.73)
```