Instituto Tecnológico y de Estudios Superiores de Monterrey

Maestría en Inteligencia Artificial
Pruebas de software y aseguramiento de la calidad

Alumno: Luis Alfredo Negron Naldos
A01793865

11Febrero 2024

1. **Liga del Repositorio:** https://github.com/PosgradoMNA/actividades-de-aprendizaje-luisnegronnaldos/tree/main/Calidad_software/A01793865_A52

2. **Problema 1: computesales.py vs PyLint**

   - Primer Código:

```
-  import json
-  import time
-  import sys
-
-  def load_json_file(file_path):
-      try:
-          with open(file_path, 'r') as file:
-              data = json.load(file)
-          return data
-      except (FileNotFoundError, json.JSONDecodeError) as e:
-          print(f"Error loading {file_path}: {e}")
-          return None
-
-  def compute_total_cost(price_catalogue, sales_records):
-      total_cost = 0
-
-      for sale in sales_records:
-          sale_id = sale.get('SALE_ID', '')
-          product = sale.get('Product', '')
-          quantity = sale.get('Quantity', 0)
-
-          if isinstance(product, str):
-              product_id = product
-          elif isinstance(product, dict):
-              product_id = product.get('product', '')
-          else:
-              print(f"Error: Unexpected structure for 'Product' in
   SALE_ID {sale_id}.")
-
-          matching_products = [item for item in price_catalogue if
   item.get('title') == product_id]
-
-          if matching_products:
-              # Assuming there's only one matching product, use the
   first one
-              cost_per_item = matching_products[0]['price']
-              total_cost += quantity * cost_per_item
-          else:
-              print(f"Error: Product ID {product_id} not found in the
   price catalogue for SALE_ID {sale_id}.")
-
-      return total_cost
-
-  def main():
-      if len(sys.argv) != 3:
-          print("Usage: python importjson.py priceCatalogue.json
   salesRecord.json")
```

```
-           sys.exit(1)
-
-       start_time = time.time()
-
-       price_catalogue_file = sys.argv[1]
-       sales_record_file = sys.argv[2]
-
-       price_catalogue = load_json_file(price_catalogue_file)
-       sales_records = load_json_file(sales_record_file)
-
-       if price_catalogue is None or sales_records is None:
-           sys.exit(1)
-
-       total_cost = compute_total_cost(price_catalogue, sales_records)
-
-       end_time = time.time()
-       elapsed_time = end_time - start_time
-
-       print(f"Total Cost: ${total_cost:.2f}")
-       print(f"Time Elapsed: {elapsed_time:.4f} seconds")
-
-       with open('SalesResults.txt', 'w') as results_file:
-           results_file.write(f"Total Cost: ${total_cost:.2f}\n")
-           results_file.write(f"Time Elapsed: {elapsed_time:.4f}
    seconds\n")
-
-   if __name__ == "__main__":
-       main()
-
```

Resultado de PyLint:

```
(base) MacBook-Air-5:TC1 Luis$ pylint Compute_Sales.py
************* Module Compute_Sales
Compute_Sales.py:28:0: C0303: Trailing whitespace (trailing-whitespace)
Compute_Sales.py:36:0: C0301: Line too long (108/100) (line-too-long)
Compute_Sales.py:71:0: 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:1:0: C0114: Missing module docstring (missing-module-docstring)
Compute_Sales.py:5:0: C0116: Missing function or method docstring (missing-function-docstring)
Compute_Sales.py:10:4: C0103: Variable name "e" doesn't conform to snake_case naming style (invalid-
name)
Compute_Sales.py:14:0: C0116: Missing function or method docstring (missing-function-docstring)
Compute_Sales.py:40:0: C0116: Missing function or method docstring (missing-function-docstring)

-------------------------------------------------------------------
Your code has been rated at 8.16/10 (previous run: 8.37/10, -0.20)
```

Se logró corregir uno a uno los problemas presentados en el código, definiendo las variables observadas, los docstring solicitados, los cambio de nombre, etc. corregir el problema obteniendose el siguiente resultado en cada uno de los elementos marcados obteniendose el siguiente resultado.

(base) MacBook-Air-5:TC1 Luis$ pylint compute_sales.py

--
Your code has been rated at 10.00/10 (previous run: 9.80/10, +0.20)

Código Corregido

```
"Programa para realizar calculo de ventas por tickect"
import json
import time
import sys

def load_json_file(file_path):
    "define los archivos a ser cargados"
    try:
        with open(file_path, 'r') as file:
            data = json.load(file)
        return data
    except (FileNotFoundError, json.JSONDecodeError) as e_n:
        print(f"Error loading {file_path}: {e_n}")
        return None

def compute_total_cost(price_catalogue, sales_records):
    "calcula los costos totales"
    total_cost = 0

    for sale in sales_records:
        sale_id = sale.get('SALE_ID', '')
        product = sale.get('Product', '')
        quantity = sale.get('Quantity', 0)

        if isinstance(product, str):
            product_id = product
        elif isinstance(product, dict):
            product_id = product.get('product', '')
        else:
            print(f"Error: Unexpected structure for 'Product' in SALE_ID
{sale_id}.")
        matching_products = [item for item in price_catalogue if
item.get('title') == product_id]

        if matching_products:
            # Assuming there's only one matching product, use the first
one
            cost_per_item = matching_products[0]['price']
            total_cost += quantity * cost_per_item
        else:
            print(f"Error: Product ID {product_id} not found in the
catalogue {sale_id}.")

    return total_cost

def main():
    "funcion principal"
    if len(sys.argv) != 3:
```

```
        print("Usage: python importjson.py priceCatalogue.json
salesRecord.json")
        sys.exit(1)

    start_time = time.time()

    price_catalogue_file = sys.argv[1]
    sales_record_file = sys.argv[2]

    price_catalogue = load_json_file(price_catalogue_file)
    sales_records = load_json_file(sales_record_file)

    if price_catalogue is None or sales_records is None:
        sys.exit(1)

    total_cost = compute_total_cost(price_catalogue, sales_records)

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

    print(f"Total Cost: ${total_cost:.2f}")
    print(f"Time Elapsed: {elapsed_time:.4f} seconds")

    with open('SalesResults.txt', 'w') as results_file:
        results_file.write(f"Total Cost: ${total_cost:.2f}\n")
        results_file.write(f"Time Elapsed: {elapsed_time:.4f} seconds\n")

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

3. Problema 2: computesales.py vs Flake8

A continuación se muestra el resultado de Flake8 del código corregido, al igual que en el caso anterior se procedio a revisar cada uno de los errores reportados.

```
(base) MacBook-Air-5:TC1 Luis$ flake8 compute_sales.py --statistics
compute_sales.py:6:1: E302 expected 2 blank lines, found 1
compute_sales.py:16:1: E302 expected 2 blank lines, found 1
compute_sales.py:30:80: E501 line too long (85 > 79 characters)
compute_sales.py:31:80: E501 line too long (97 > 79 characters)
compute_sales.py:38:80: E501 line too long (90 > 79 characters)
compute_sales.py:42:1: E302 expected 2 blank lines, found 1
compute_sales.py:45:80: E501 line too long (81 > 79 characters)
compute_sales.py:71:1: E305 expected 2 blank lines after class or function definition, found 1
3     E302 expected 2 blank lines, found 1
1     E305 expected 2 blank lines after class or function definition, found 1
4     E501 line too long (85 > 79 characters)
```

Se logró uno a uno los problemas presentados en el código, donde la mayoria de los problemas eran por longitud de código, espacios en blanco y espacios requeridos entre funciones. Luego de corregir los errores se procedio a correr nuevamente la función dando como resultado que no hay errores.

```
(base) MacBook-Air-5:TC1 Luis$ flake8 compute_sales.py --statistics
```

(base) MacBook-Air-5:TC1 Luis$

El Código final es el siguiente:

```
"Programa para realizar calculo de ventas por tickect"
import json
import time
import sys

def load_json_file(file_path):
    "define los archivos a ser cargados"
    try:
        with open(file_path, 'r') as file:
            data = json.load(file)
        return data
    except (FileNotFoundError, json.JSONDecodeError) as e_n:
        print(f"Error loading {file_path}: {e_n}")
        return None

def compute_total_cost(price_catalogue, sales_records):
    "calcula los costos totales"
    total_cost = 0

    for sale in sales_records:
        sale_id = sale.get('SALE_ID', '')
        product = sale.get('Product', '')
        quantity = sale.get('Quantity', 0)

        if isinstance(product, str):
            product_id = product
        elif isinstance(product, dict):
            product_id = product.get('product', '')
        else:
            print("Error: Unexpected structure for Product")
        mat_prod = [item for item in price_catalogue
                    if item.get('title') == product_id]

        if mat_prod:
            # Assuming there's only one matching product, use the first
one
            cost_per_item = mat_prod[0]['price']
            total_cost += quantity * cost_per_item
        else:
            print(f"Error: Product ID {product_id} not found in
{sale_id}.")

    return total_cost

def main():
    "funcion principal"
    if len(sys.argv) != 3:
        print("Usage: python importjson.py price.json sales.json")
        sys.exit(1)
```

```
    start_time = time.time()

    price_catalogue_file = sys.argv[1]
    sales_record_file = sys.argv[2]

    price_catalogue = load_json_file(price_catalogue_file)
    sales_records = load_json_file(sales_record_file)

    if price_catalogue is None or sales_records is None:
        sys.exit(1)

    total_cost = compute_total_cost(price_catalogue, sales_records)

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

    print(f"Total Cost: ${total_cost:.2f}")
    print(f"Time Elapsed: {elapsed_time:.4f} seconds")

    with open('SalesResults.txt', 'w') as results_file:
        results_file.write(f"Total Cost: ${total_cost:.2f}\n")
        results_file.write(f"Time Elapsed: {elapsed_time:.4f} seconds\n")

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