Actividad 5.2

Objetivo(s)

2.7 Explicar la diferencia entre pruebas dinámicas y pruebas estáticas
2.8 Describir los beneficios e impacto de la calidad de las prácticas asociadas a pruebas estáticas.
2.9 Explicar el origen de las inspecciones como herramienta de pruebas estáticas.
2.10 Describir las diferencias entre revisiones informales, caminatas estructuradas, inspecciones e inspecciones automáticas.
2.11 Describir la relación de las herramientas de análisis estático y el código fuente.
2.12 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.

Especificaciones de entrega

- **Modalidad:** Individual
- **Medio de realización/entrega:** Liga al repositorio GIT que contiene todos los activos.
- **Nombre del entregable:** Matrícula de estudiante_Número de actividad. Ejemplo: A02234_A5.2.pdf siguiendo las indicaciones de los pasos 8, 9 y 10

Ejecución TC1

```
Compute Qty: 1 of product: "Corn" at price: $13.55
Compute Qty: 2 of product: "French fries" at price: $18.32
Compute Qty: 3 of product: "Ground beef meat burger" at price: $11.73
Compute Qty: 1 of product: "Hazelnut in black ceramic bowl" at price: $27.35
Compute Qty: 2 of product: "Cherry" at price: $14.35
Compute Qty: 5 of product: "Homemade bread" at price: $17.48
Compute Qty: 2 of product: "Smoothie with chia seeds" at price: $25.26
Compute Qty: 10 of product: "Corn" at price: $13.55
Compute Qty: 2 of product: "Rustic breakfast" at price: $21.32
Compute Qty: 3 of product: "Sandwich with salad" at price: $22.48
Compute Qty: 20 of product: "Raw legums" at price: $17.11
Compute Qty: 4 of product: "Fresh stawberry" at price: $28.59
Compute Qty: 1 of product: "Pears juice" at price: $19.49
Compute Qty: 8 of product: "Green smoothie" at price: $17.68
Compute Qty: 9 of product: "Cuban sandwiche" at price: $18.5
Compute Qty: 2 of product: "Hazelnut in black ceramic bowl" at price: $27.35
Compute Qty: 1 of product: "Tomatoes" at price: $26.03
Compute Qty: 2 of product: "Plums" at price: $19.18
Compute Qty: 3 of product: "Fresh blueberries" at price: $21.01
Compute Qty: 5 of product: "Green smoothie" at price: $17.68
Compute Qty: 4 of product: "Corn" at price: $13.55


Total cost for all sales: $2481.86
Elapsed Time: 0.00101185 seconds
```

Ejecución TC2

```
Compute Qty: 2 of product: "Corn" at price: $13.55
Compute Qty: 4 of product: "French fries" at price: $18.32
Compute Qty: 64 of product: "Ground beef meat burger" at price: $11.73
Compute Qty: 2 of product: "Hazelnut in black ceramic bowl" at price: $27.35
Compute Qty: 3 of product: "Sweet fresh stawberry" at price: $29.45
Compute Qty: 456 of product: "Homemade bread" at price: $17.48
Compute Qty: 2 of product: "Smoothie with chia seeds" at price: $25.26
Compute Qty: 5 of product: "Corn" at price: $13.55
Compute Qty: 645 of product: "Plums" at price: $19.18
Compute Qty: -35 of product: "Fresh blueberries" at price: $21.01
Compute Qty: 2 of product: "Green smoothie" at price: $17.68
Compute Qty: 465 of product: "Corn" at price: $13.55
Compute Qty: 20 of product: "French fries" at price: $18.32
Compute Qty: 4 of product: "Ground beef meat burger" at price: $11.73
Compute Qty: 1 of product: "Hazelnut in black ceramic bowl" at price: $27.35
Compute Qty: 131 of product: "Sweet fresh stawberry" at price: $29.45
Compute Qty: 9 of product: "Homemade bread" at price: $17.48
Compute Qty: 13 of product: "Smoothie with chia seeds" at price: $25.26
Compute Qty: 678 of product: "Corn" at price: $13.55
Compute Qty: 334 of product: "Plums" at price: $19.18
Compute Qty: 3445 of product: "Fresh blueberries" at price: $21.01
Compute Qty: -123 of product: "Green smoothie" at price: $17.68
Compute Qty: 445 of product: "Corn" at price: $13.55


Total cost for all sales: $166568.23
Elapsed Time: 0.00100493 seconds
```

Ejecución TC3

```
Compute Qty: 64 of product: "Ground beef meat burger" at price: $11.73
Compute Qty: 2 of product: "Hazelnut in black ceramic bowl" at price: $27.35
Compute Qty: 3 of product: "Sweet fresh stawberry" at price: $29.45
Compute Qty: 456 of product: "Homemade bread" at price: $17.48
Compute Qty: 2 of product: "Smoothie with chia seeds" at price: $25.26
Compute Qty: 5 of product: "Corn" at price: $13.55
Compute Qty: 645 of product: "Plums" at price: $19.18
Compute Qty: -35 of product: "Fresh blueberries" at price: $21.01
Compute Qty: 2 of product: "Green smoothie" at price: $17.68
Compute Qty: 465 of product: "Corn" at price: $13.55
Error: Frijoles not found in Dictionary
Error: Dictionary provided doesn't exist
Invalid record: Qty = 100, Price = False, Product: Frijoles
Compute Qty: 4 of product: "Ground beef meat burger" at price: $11.73
Compute Qty: 1 of product: "Hazelnut in black ceramic bowl" at price: $27.35
Compute Qty: 131 of product: "Sweet fresh stawberry" at price: $29.45
Compute Qty: 9 of product: "Homemade bread" at price: $17.48
Compute Qty: 13 of product: "Smoothie with chia seeds" at price: $25.26
Compute Qty: 678 of product: "Corn" at price: $13.55
Compute Qty: 334 of product: "Plums" at price: $19.18
Compute Qty: 3445 of product: "Fresh blueberries" at price: $21.01
Compute Qty: -123 of product: "Green smoothie" at price: $17.68
Compute Qty: 445 of product: "Corn" at price: $13.55


Total cost for all sales: $165235.37
Elapsed Time: 0.00107193 seconds
```

Pylint examination

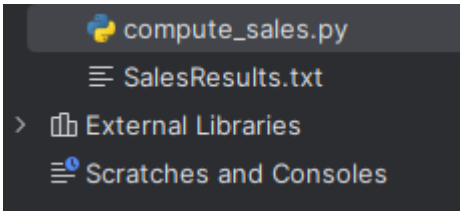
```
************* Module computeSales
computeSales.py:54:0: C0301: Line too long (116/100) (line-too-long)
computeSales.py:1:0: C0103: Module name "computeSales" doesn't conform to snake_case naming style (invalid-name)
computeSales.py:43:4: R1705: Unnecessary "elif" after "return", remove the leading "el" from "elif" (no-else-return)
computeSales.py:37:0: R1710: Either all return statements in a function should return an expression, or none of them sho
uld. (inconsistent-return-statements)
computeSales.py:106:4: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the code inside it (no-
else-return)

-----------------------------------
Your code has been rated at 9.44/10
```

Flake8 examination

```
computeSales.py:54:80: E501 line too long (116 > 79 characters)
computeSales.py:64:46: E231 missing whitespace after ','
computeSales.py:69:80: E501 line too long (85 > 79 characters)
computeSales.py:72:80: E501 line too long (86 > 79 characters)
computeSales.py:122:20: E225 missing whitespace around operator
```

Fixing issues



Module name "computeSales" doesn't conform to snake_case naming style (invalid-name)

```
2 usages
def read_filename_from_params(index):
    """Reads a file name by index from the command-line arguments."""
    if len(sys.argv) != 3:
        print("Invalid program call"
              "\nExample of usage: python compute_sales.py priceCatalogue.json salesRecord.json")
        sys.exit(1)
    return sys.argv[index]
```

Line too long (116/100) (line-too-long)

```
4 usages
def validate_field(obj, field_name):
    """Validates that a dictionary contains a given key"""
    if not obj:
        print("Error: Dictionary provided doesn't exist")
        return False
    if field_name in obj:
        return obj[field_name]

    print(f"Error: {field_name} not found in Dictionary")
    return False
```

Unnecessary "else" after "return", remove the "else" and de-indent the code inside it (no-else-return)

```
2 usages
def get_file_path(filename):
    """Gets the file path of the provided file name."""
    main_directory = "data_source"
    path = ""
    if not filename:
        print("File name was not provided")
        sys.exit(1)
    if "TC1" in filename:
        path = f"{main_directory}/TC1/{filename}"
    elif "TC2" in filename:
        path = f"{main_directory}/TC2/{filename}"
    elif "TC3" in filename:
        path = f"{main_directory}/TC3/{filename}"
    return path
```

Unnecessary "elif" after "return", remove the leading "el" from "elif" (no-else-return)
Either all return statements in a function should return an expression, or none of them should. (inconsistent-return-statements)

After fixing the pylint issues

```
--------------------------------------------------------------------
Your code has been rated at 10.00/10 (previous run: 9.89/10, +0.11)
```

Fixing Flake8 issues

```
2 usages
def read_filename_from_params(index):
    """Reads a file name by index from the command-line arguments."""
    if len(sys.argv) != 3:
        print("Invalid program call"
              "\nExample of usage: "
              "python compute_sales.py priceCatalogue.json salesRecord.json")
        sys.exit(1)
    return sys.argv[index]
```

E501 line too long (97 > 79 characters)

```
1 usage
def get_total_cost(prices, sales):
    """Calculates the total cost of sales"""
    total = 0
    print("::Details of sales::\n")
    for sale in sales:
        qty = process_qty(validate_field(sale, field_name: "Quantity"))
        product = validate_field(sale, field_name: "Product")
        price_record = validate_field(prices, product)
        price = process_price(validate_field(price_record, field_name: "price"))
        if qty and product and price:
            print(f"Compute Qty: {qty} of product: \"{product}\" "
                  f"at price: ${price}")
            total += (qty * price)
        else:
            print(f"Invalid record: Qty = {qty}, Price = {price}, "
                  f"Product: {product}")
    print("\n")
    return total
```

E501 line too long (85 > 79 characters)
E501 line too long (86 > 79 characters)

```
1 usage
def get_total_cost(prices, sales):
    """Calculates the total cost of sales"""
    total = 0
    print("::Details of sales::\n")
    for sale in sales:
        qty = process_qty(validate_field(sale, field_name: "Quantity"))
        product = validate_field(sale, field_name: "Product")
        price_record = validate_field(prices, product)
```

E231 missing whitespace after ','

```
sales = read_json_file(get_file_path(sales_file_name))
total_cost = get_total_cost(prices, sales)
elapsed_time = time.time() - start_time
sales_results = (f"Total cost for all sales: ${total_cost:.2f}"
                 f"\nElapsed Time: {elapsed_time:.8f} seconds")
```

E225 missing whitespace around operator

After fixing the highlighter issues and running flake8 on my script, I got no more feedback

```
\Documents\Personal\Maestría\Pruebas de software\Semana 5\Actividad 5.2>flake8 compute_sales.py

\Documents\Personal\Maestría\Pruebas de software\Semana 5\Actividad 5.2>
```