ACTIVITY 5.2
PROGRAMMING EXERCISES 2

TC4017 Software testing and quality assurance

Student:
Arturo Alfonso Gallardo Jasso   A01795510

Professors:
Dr. Gerardo Padilla Zárate
Sajith Delgado Miranda

Activity instructions

1. Implement the programs from this document using the Python language.
2. Follow the PEP-8 coding standard.
3. Verify the correct execution of the programs by generating tests of each exercise using the indicated resources. Document the results.
4. Install flake8 package using PIP, https://luminousmen.com/post/python-static-analysis-tools
5. Verify that your programs do not generate errors or issues using pylint.
   There are 5 kind of message types :
   * (C) convention, for programming standard violation
   * (R) refactor, for bad code smell
   * (W) warning, for python specific problems
   * (E) error, for probable bugs in the code
   * (F) fatal, if an error occurred which prevented pylint from doing
   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

6. Fix all the issues found by flake8 and verify that your program continues to work correctly.
7. When finished, upload your programs to your personal repository.
8. The project should be named: StudentID_ActivityNumberA5.2
9. Submit the repository link in the Canvas assignment.
10. Also, upload the source files of the task to Canvas.

Repository

The project folder and a copy of this PDF report can be found in the following link:

https://github.com/AAGJ89/MNA_TC4017_SW.git

Platform details

Python 3.10.11

Visual Studio Code 1.96.4

Problem 1. Compute statistics

1.1 Software Requirements

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.

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.

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 name of the program shall be computeSales.py

Req 5. The minimum format to invoke the program shall be as follows: python computeSales.py priceCatalogue.json TCXsalesRecord.json*

Req 6. The program shall manage files having from hundreds of items to thousands of items.

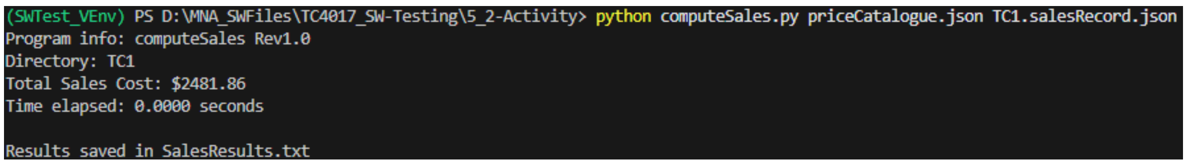
Req 7. The program should include at the end of the execution the time elapsed for the execution and calculus of the data. This number shall be included in the results file and on the screen.

Req 8. Be compliant with PEP8.

*The requirement was not clear, and the name of the files were wrong. I modified the names as requested by the requirements and added TCX as part of the format to call each folder independently.

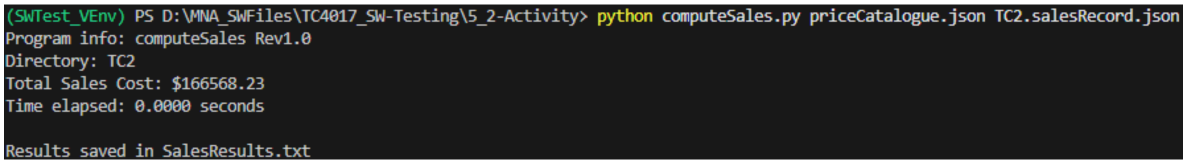
1.2 Code Highlights

- The program successfully invokes the file using the necessary format.
- The program computes total cost of the second file selected, prints the results on screen and stores them into "SalesResults.txt" file.
- The program prints the time elapsed, from the beginning of the execution until complete the storage of the data.
- The program manages 4 types of errors.
- Refer to Figure 1, 2, and 3 to see compliance of the program with Requirements 1, 2, 3, 4, 5, 6, 7.



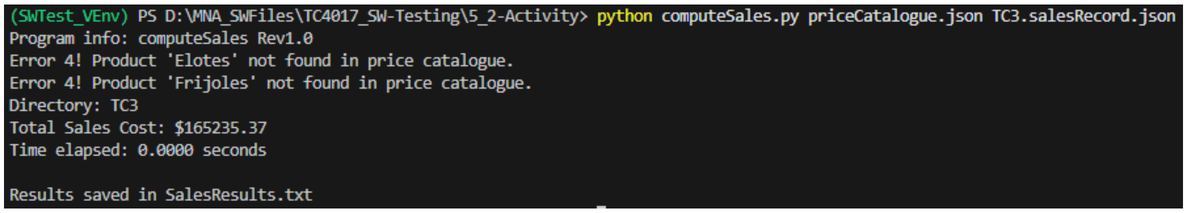
```
(SWTest_VEnv) PS D:\MNA_SWFiles\TC4017_SW-Testing\5_2-Activity> python computeSales.py priceCatalogue.json TC1.salesRecord.json
Program info: computeSales Rev1.0
Directory: TC1
Total Sales Cost: $2481.86
Time elapsed: 0.0000 seconds

Results saved in SalesResults.txt
```

Figure 1



```
(SWTest_VEnv) PS D:\MNA_SWFiles\TC4017_SW-Testing\5_2-Activity> python computeSales.py priceCatalogue.json TC2.salesRecord.json
Program info: computeSales Rev1.0
Directory: TC2
Total Sales Cost: $166568.23
Time elapsed: 0.0000 seconds

Results saved in SalesResults.txt
```

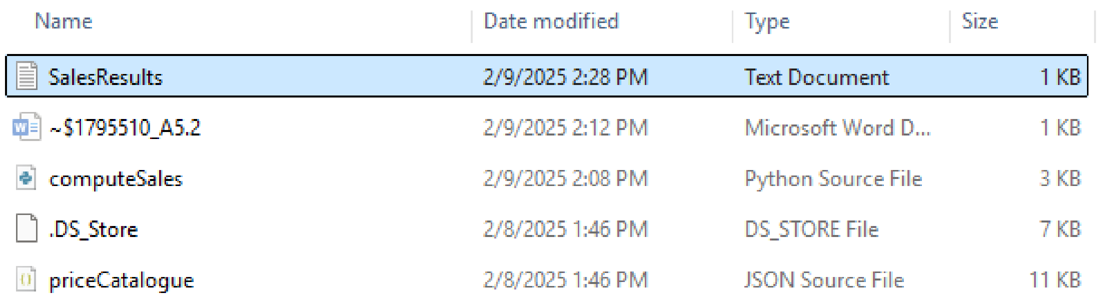
Figure 2



```
(SWTest_VEnv) PS D:\MNA_SWFiles\TC4017_SW-Testing\5_2-Activity> python computeSales.py priceCatalogue.json TC3.salesRecord.json
Program info: computeSales Rev1.0
Error 4! Product 'Elotes' not found in price catalogue.
Error 4! Product 'Frijoles' not found in price catalogue.
Directory: TC3
Total Sales Cost: $165235.37
Time elapsed: 0.0000 seconds

Results saved in SalesResults.txt
```

Figure 3

- The program stores the results in “SalesResults.txt”, see Figure 4.

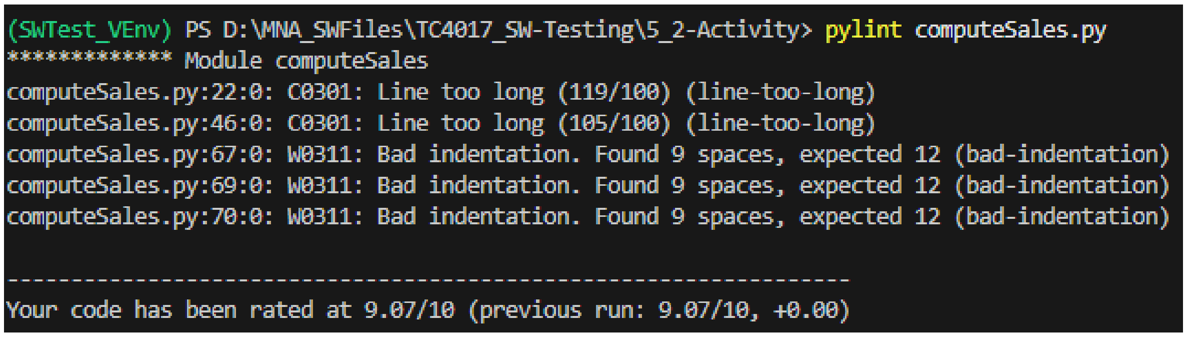


Name	Date modified	Type	Size
SalesResults	2/9/2025 2:28 PM	Text Document	1 KB
~$1795510_A5.2	2/9/2025 2:12 PM	Microsoft Word D...	1 KB
computeSales	2/9/2025 2:08 PM	Python Source File	3 KB
.DS_Store	2/8/2025 1:46 PM	DS_STORE File	7 KB
priceCatalogue	2/8/2025 1:46 PM	JSON Source File	11 KB

Figure 4

1.3 Pylint report

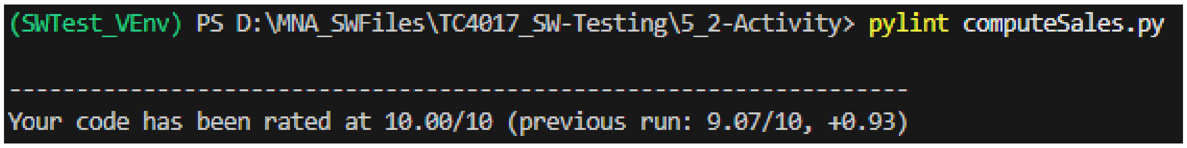
The report of the first version of the code is given by Figure 5.



```
(SWTest_VEnv) PS D:\MNA_SWFiles\TC4017_SW-Testing\5_2-Activity> pylint computeSales.py
************* Module computeSales
computeSales.py:22:0: C0301: Line too long (119/100) (line-too-long)
computeSales.py:46:0: C0301: Line too long (105/100) (line-too-long)
computeSales.py:67:0: W0311: Bad indentation. Found 9 spaces, expected 12 (bad-indentation)
computeSales.py:69:0: W0311: Bad indentation. Found 9 spaces, expected 12 (bad-indentation)
computeSales.py:70:0: W0311: Bad indentation. Found 9 spaces, expected 12 (bad-indentation)

------------------------------------------------------------------
Your code has been rated at 9.07/10 (previous run: 9.07/10, +0.00)
```

Figure 5

The score was 9.07 of 10.00. Changes were made to have 10.0



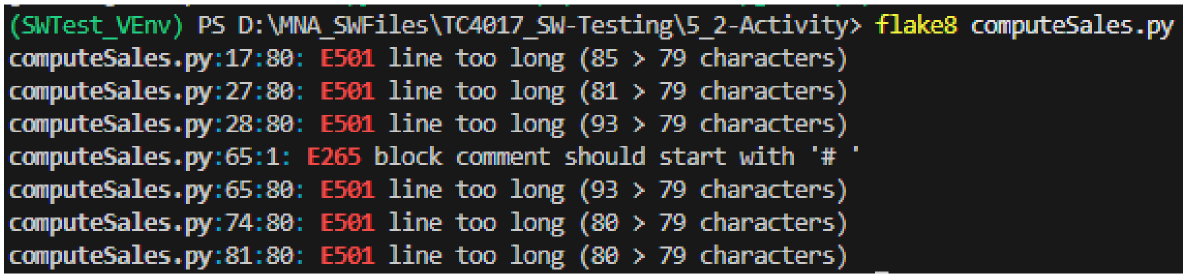
```
(SWTest_VEnv) PS D:\MNA_SWFiles\TC4017_SW-Testing\5_2-Activity> pylint computeSales.py

--------------------------------------------------------------------
Your code has been rated at 10.00/10 (previous run: 9.07/10, +0.93)
```

Figure 6

The program meets Requirement 8 for Pylint.

1.4 Flake8 report

The report of the second version of the code is given by Figure 6.



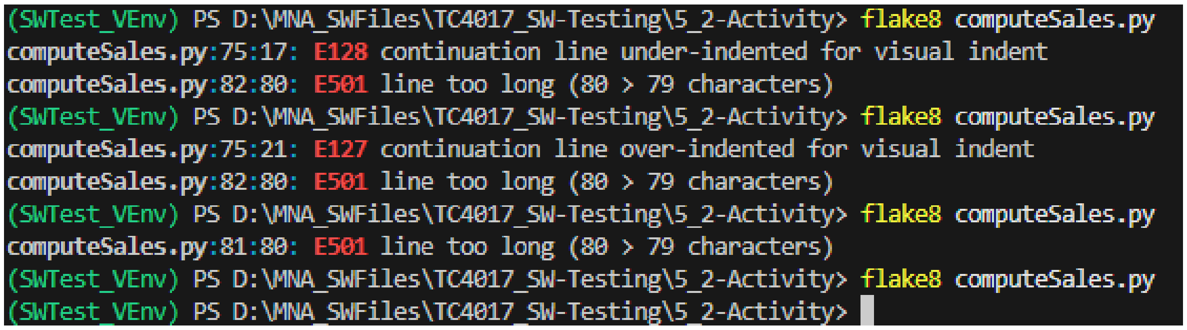
```
(SWTest_VEnv) PS D:\MNA_SWFiles\TC4017_SW-Testing\5_2-Activity> flake8 computeSales.py
computeSales.py:17:80: E501 line too long (85 > 79 characters)
computeSales.py:27:80: E501 line too long (81 > 79 characters)
computeSales.py:28:80: E501 line too long (93 > 79 characters)
computeSales.py:65:1: E265 block comment should start with '# '
computeSales.py:65:80: E501 line too long (93 > 79 characters)
computeSales.py:74:80: E501 line too long (80 > 79 characters)
computeSales.py:81:80: E501 line too long (80 > 79 characters)
```

Figure 7

E265: The comment should have space after #. Actually, this was an obsoleted instruction that I forgot to remove so I deleted the line.

E501 is an error about a line too long. I made the changes to reduce the length of the lines below 79. During this, I got a new error that was E128 and E127 due wrong indexing when splitting a row in two. I decided to make it shorter.

Figure 8 contains the multiple interactions until being complaint with flake8.



```
(SWTest_VEnv) PS D:\MNA_SWFiles\TC4017_SW-Testing\5_2-Activity> flake8 computeSales.py
computeSales.py:75:17: E128 continuation line under-indented for visual indent
computeSales.py:82:80: E501 line too long (80 > 79 characters)
(SWTest_VEnv) PS D:\MNA_SWFiles\TC4017_SW-Testing\5_2-Activity> flake8 computeSales.py
computeSales.py:75:21: E127 continuation line over-indented for visual indent
computeSales.py:82:80: E501 line too long (80 > 79 characters)
(SWTest_VEnv) PS D:\MNA_SWFiles\TC4017_SW-Testing\5_2-Activity> flake8 computeSales.py
computeSales.py:81:80: E501 line too long (80 > 79 characters)
(SWTest_VEnv) PS D:\MNA_SWFiles\TC4017_SW-Testing\5_2-Activity> flake8 computeSales.py
(SWTest_VEnv) PS D:\MNA_SWFiles\TC4017_SW-Testing\5_2-Activity>
```

Figure 8

References

Spolsky, J. (2005). *Making Wrong Code Look Wrong – Joel on Software*.

Van Rossum, G., Warsaw, B. & Coghlan, Alyssa (2013). PEP 8 – Style Guide for Python Cod. Convención de codificación de Python - PEP8 https://peps.python.org/pep-0008/

Lutz, M. (2013). *Learning Python, 5th Edition.* 5ed.

Pylint Tutorial – How to Write Clean Python

Flake8: Your Tool For Style Guide Enforcement