Programa

# -\*- coding: utf-8 -\*-

"""calculateSales.py

Automatically generated by Colab.

Original file is located at

https://colab.research.google.com/drive/1okpv7m4Y\_MEyEsfSoB5GnSkEcN7SYIeE

Calcula el costo total de ventas tomando como base el cat\u00e1logo de precios.

Este programa toma dos archivos JSON como entrada:

1. priceCatalogue.json - Contiene información de los precios de productos.

2. salesRecord.json - Contiene un registro de las ventas realizadas.

El resultado se muestra en pantalla y se guarda en SalesResults.txt.

"""

import json

import sys

import time

import glob

from google.colab import drive

drive.mount('/content/drive', force\_remount=True)

# Definir constantes para las rutas de archivos

PRODUCTS\_FILE = (

"/content/drive/MyDrive/Calidad\_software/Semana5/TC1.ProductList.json"

)

SALES\_FILES\_PATTERN = (

"/content/drive/MyDrive/Calidad\_software/Semana5/TC\*.Sales.json"

)

RESULT\_FILE = (

"/content/drive/MyDrive/Calidad\_software/Semana5/SalesResults.txt"

)

def load\_json(file\_path):

"""

Carga un archivo JSON y maneja errores.

"""

try:

with open(file\_path, "r", encoding="utf-8") as file:

return json.load(file)

except FileNotFoundError:

print(f"Error: No se encontró el archivo {file\_path}.")

except json.JSONDecodeError:

print(f"Error: {file\_path} no tiene un formato JSON válido.")

return None

def build\_price\_catalogue(product\_list):

"""

Crea un diccionario de precios basado en el título

del producto.

"""

return {product["title"]: product["price"] for

product in product\_list}

def calculate\_total\_sales(price\_catalogue, sales\_records):

"""

Calcula el costo total de ventas basado en el catálogo de precios.

"""

total\_sales = 0.0

detailed\_sales = []

errors = []

for sale in sales\_records:

product\_name = sale.get("Product")

quantity = sale.get("Quantity")

if product\_name not in price\_catalogue:

errors.append(

f"Producto no encontrado en catálogo: {product\_name}"

)

continue

if not isinstance(quantity, (int, float)) or quantity <= 0:

errors.append(

f"Cantidad inválida para producto {product\_name}: {quantity}"

)

continue

price = price\_catalogue[product\_name]

total\_cost = price \* quantity

total\_sales += total\_cost

detailed\_sales.append(

f"Producto: {product\_name}, Cantidad: {quantity}, "

f"Precio unitario: ${price:.2f}, Costo total: ${total\_cost:.2f}"

)

return total\_sales, detailed\_sales, errors

def main():

"""Función principal del programa."""

start\_time = time.time()

# Cargar lista de productos y construir catálogo de precios

product\_data = load\_json(PRODUCTS\_FILE)

if product\_data is None:

sys.exit(1)

price\_catalogue = build\_price\_catalogue(product\_data)

# Cargar todas las ventas de los archivos TCx.Sales.json

sales\_data = []

for sales\_file in glob.glob(SALES\_FILES\_PATTERN):

sales = load\_json(sales\_file)

if sales:

sales\_data.extend(sales)

if not sales\_data:

print("Error: No se encontraron datos de ventas válidos.")

sys.exit(1)

# Calcular ventas totales

total\_sales, detailed\_sales, errors = calculate\_total\_sales(

price\_catalogue, sales\_data

)

elapsed\_time = time.time() - start\_time

# Generar informe

result\_text = (

f"Costo total de ventas: ${total\_sales:.2f}\n\n"

"Detalles de ventas:\n" + "\n".join(detailed\_sales) +

f"\n\nTiempo de ejecución: {elapsed\_time:.4f} segundos\n"

)

if errors:

result\_text += "\nErrores encontrados:\n" + "\n".join(errors) + "\n"

print(result\_text)

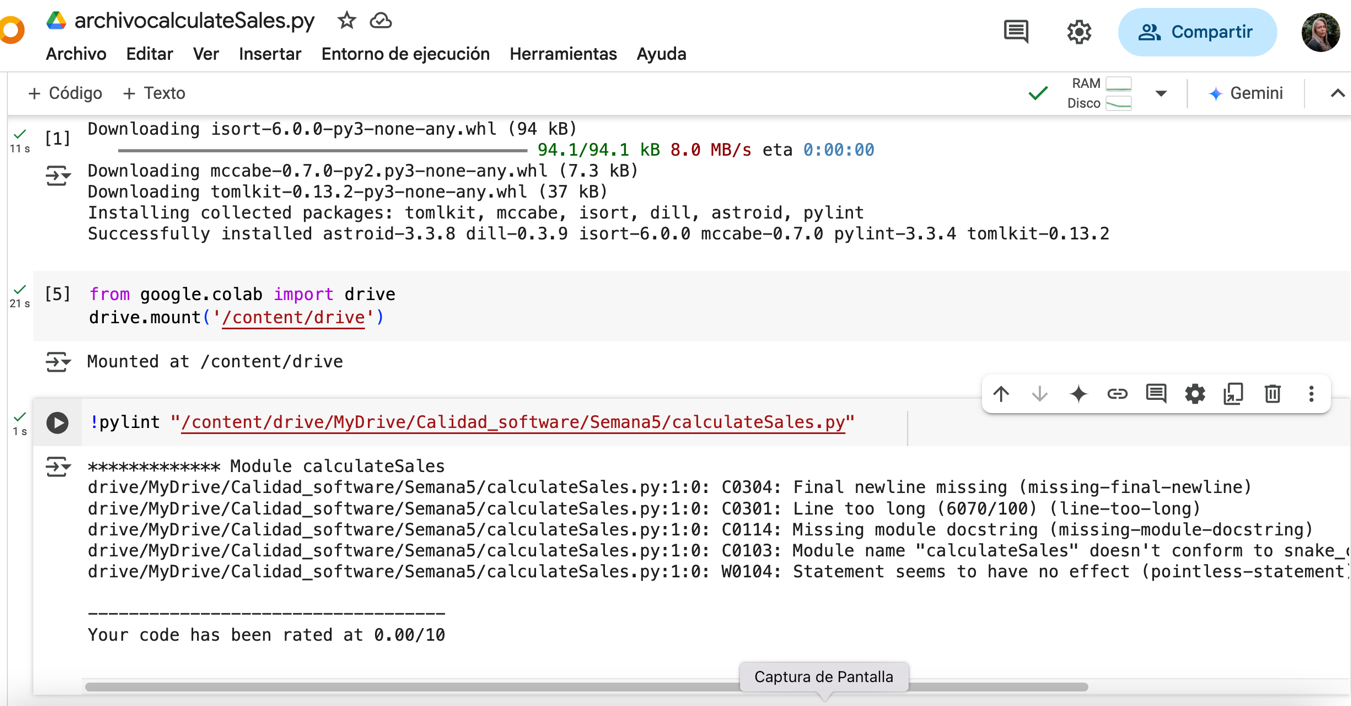
with open(RESULT\_FILE, "w", encoding="utf-8") as result\_file:

result\_file.write(result\_text)

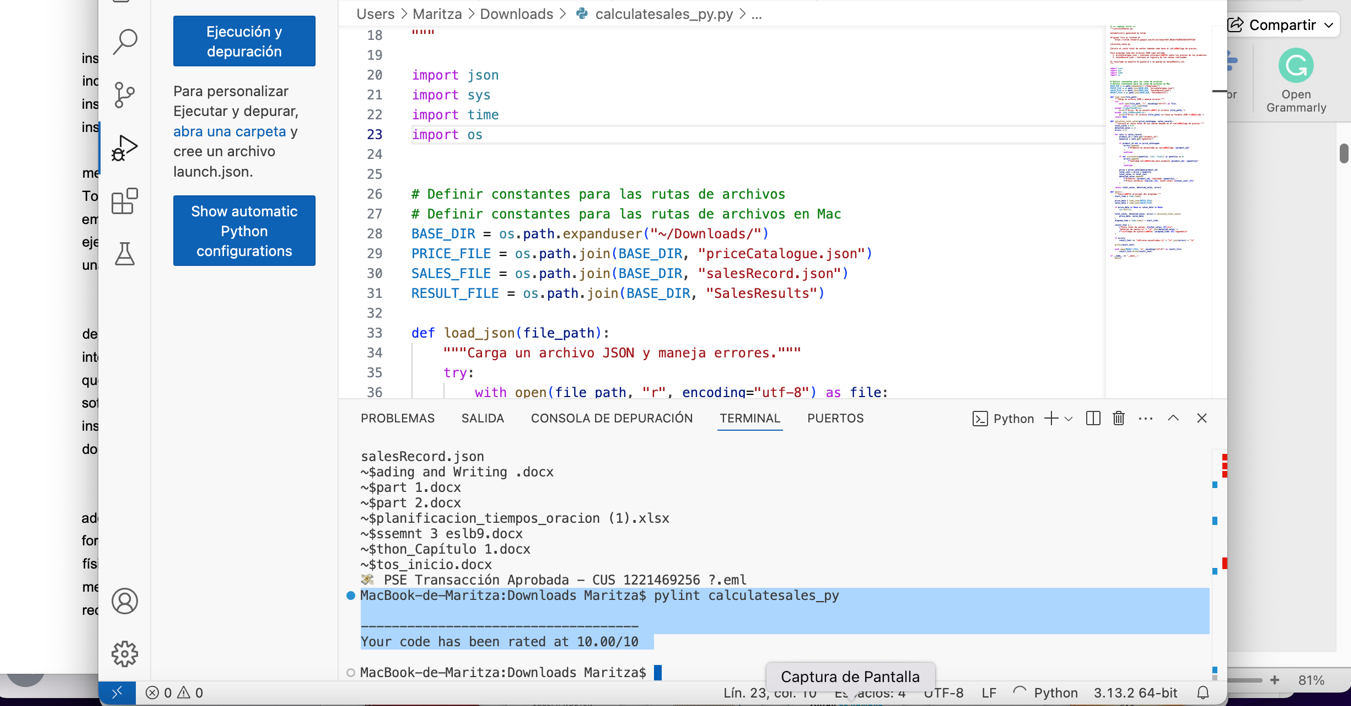
if \_\_name\_\_ == "\_\_main\_\_":

main()

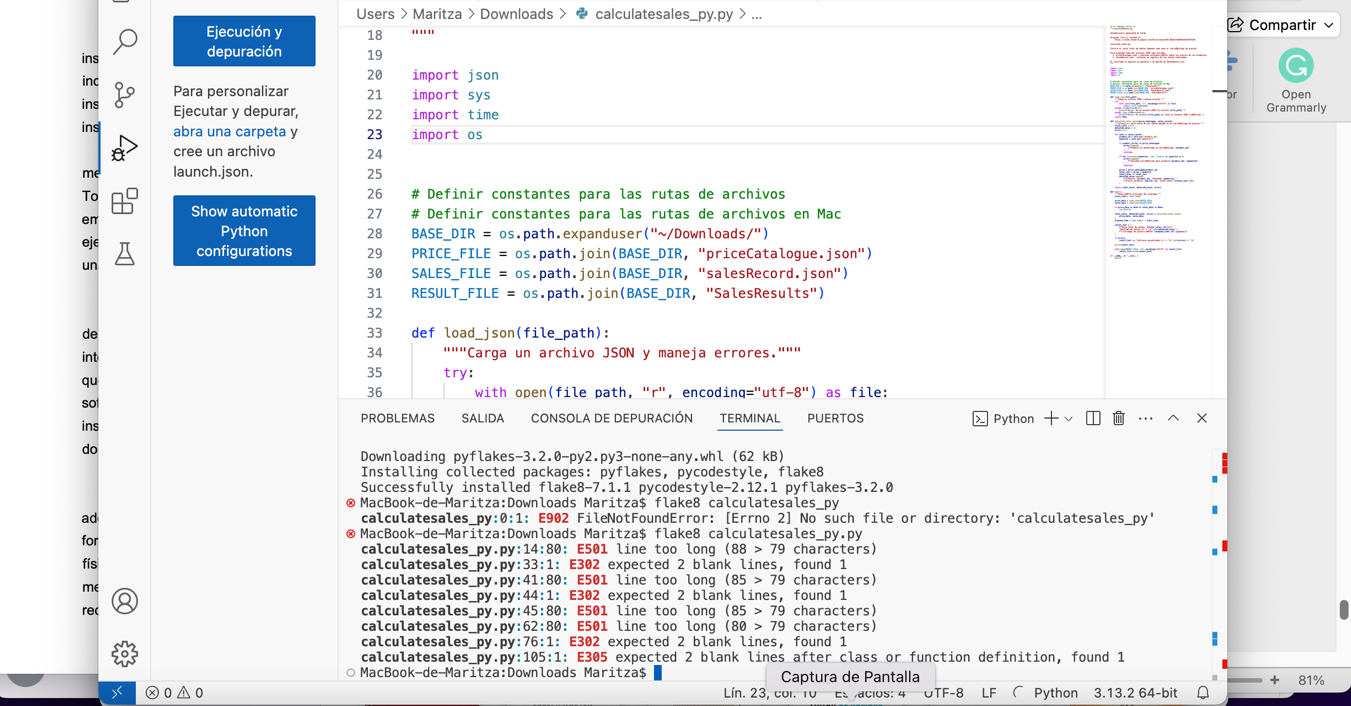
Al Ejecutar el comando Pylint para las validaciones dinámicas se encontró:



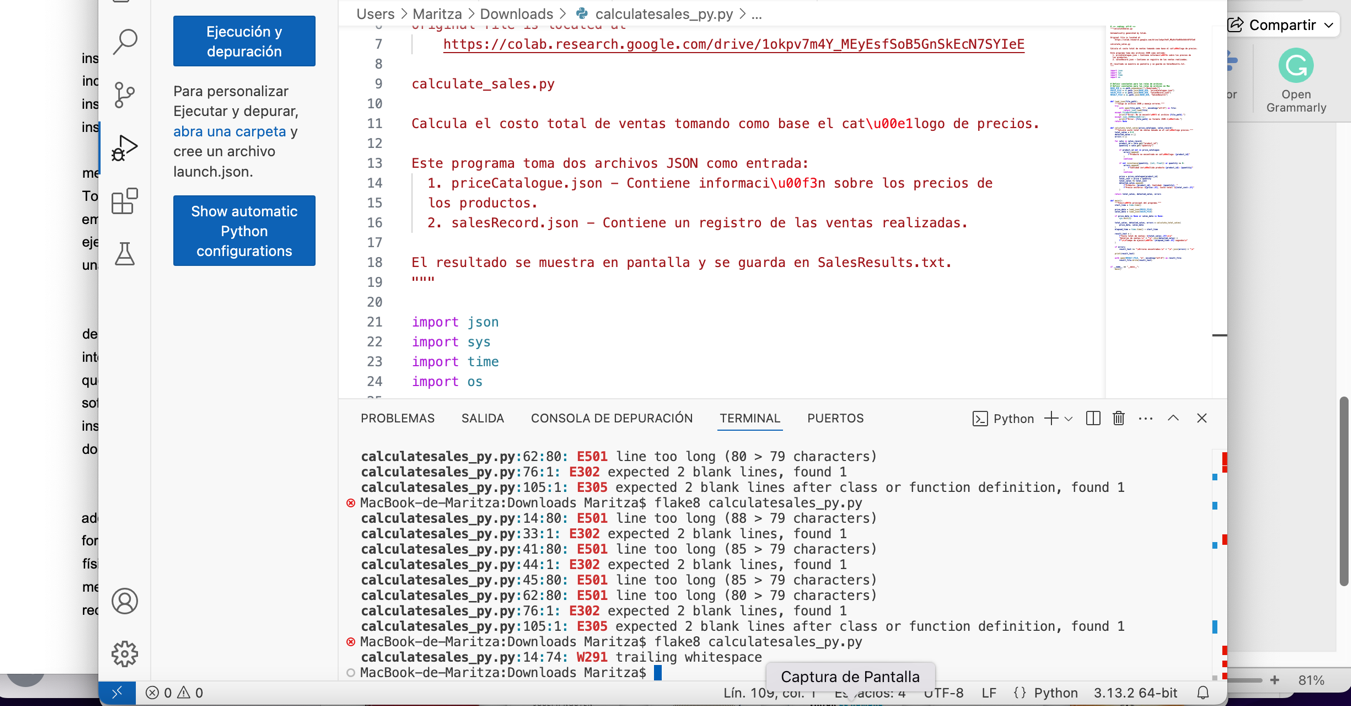
En colab no me ejecuta correctamente, lo realice en Visual y me salió todo ok



Al ejecutar Flake8 salio por visual:



Se ajustaron cosas se subsanaron, pero sale otro error



Ultima ejecución eliminando espacios y reduciendo el texto

