DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF TECHNOLOGY
Pandit Deendayal Energy University
SESSION 2026-27

Advanced Python Programming Lab-23CP301P

Name: Aayush J Shah
Roll No: 23BCP001
Division: 1 Group: 1

EXPERIMENT 1

Text File Analysis for Product Reviews

Objective:

To develop a Python script that reads multiple customer review files from a directory, extracts structured data using regular expressions, computes the average rating per product, identifies the top-rated products, and writes a detailed summary to a text file.

Task Description:

Consider a scenario where you are working as a data scientist for a large e-commerce company. Your team is responsible for analyzing customer feedback data, which is stored in multiple text files. Each text file contains customer reviews for different product categories. Your task is to write a Python script that performs the following operations: Read the contents of all the text files in a given directory. For each review, extract the following information:

- Customer ID (a 6-digit alphanumeric code)
- Product ID (a 10-digit alphanumeric code)
- Review date (in the format ”YYYY-MM-DD”)
- Review rating (an integer between 1 and 5)
- Review text (the actual feedback provided by the customer)

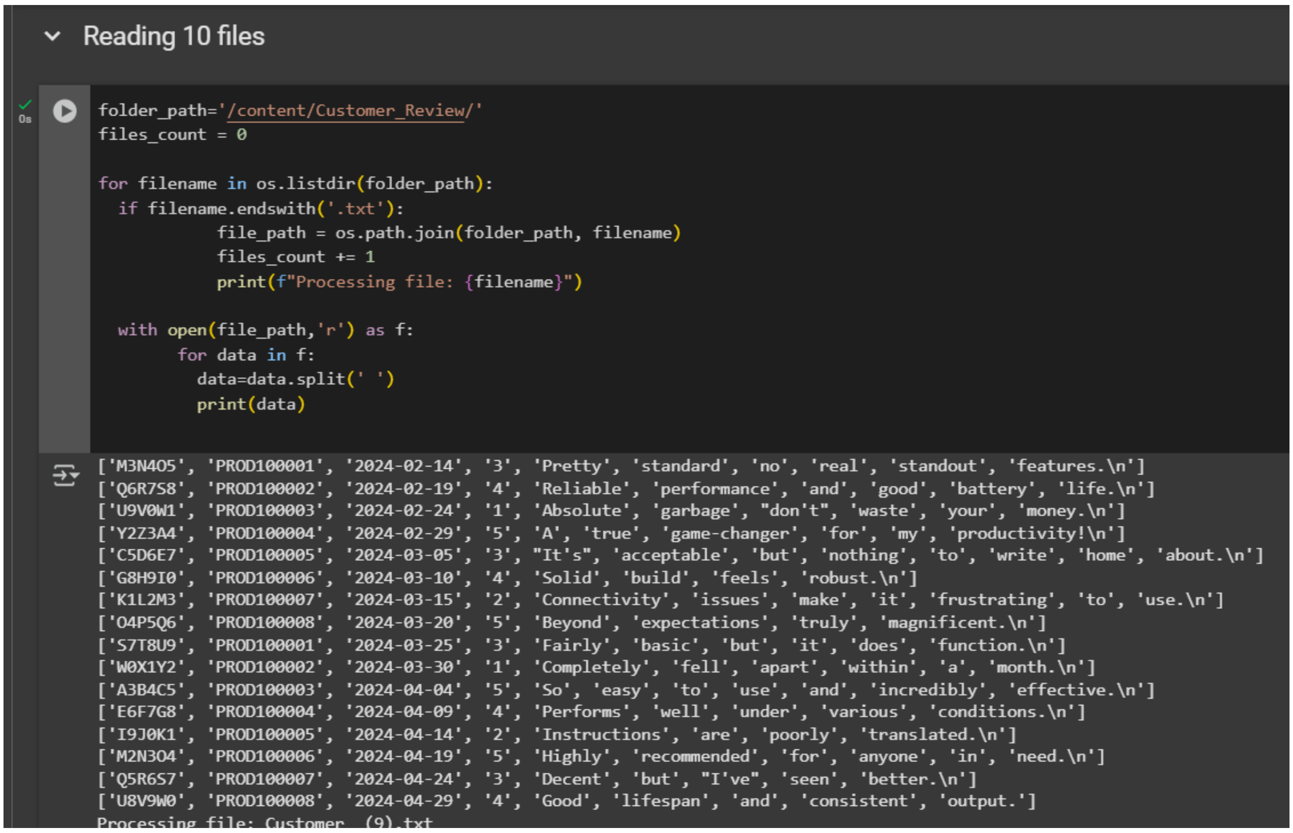
Calculate the average review rating for each product and store it in a dictionary where the product ID is the key and the average rating is the value. Determine the top 3 products with the highest average review ratings. Create a new text file named ”summary.txt” and write the following information into it:

- The total number of reviews processed.
- The total number of valid reviews (reviews with all required information extracted successfully).
- The total number of invalid reviews (reviews with missing or incorrect information).
- The product ID and average rating of the top 3 products with the highest average ratings

Your Python script should be robust, handling any potential errors or exceptions during the file-handling process. In addition, you should implement efficient algorithms to handle large volumes of data without consuming excessive memory or processing time. Write the Python script to achieve the above objectives and provide detailed comments explaining each step of your implementation.

CODE AND OUTPUT

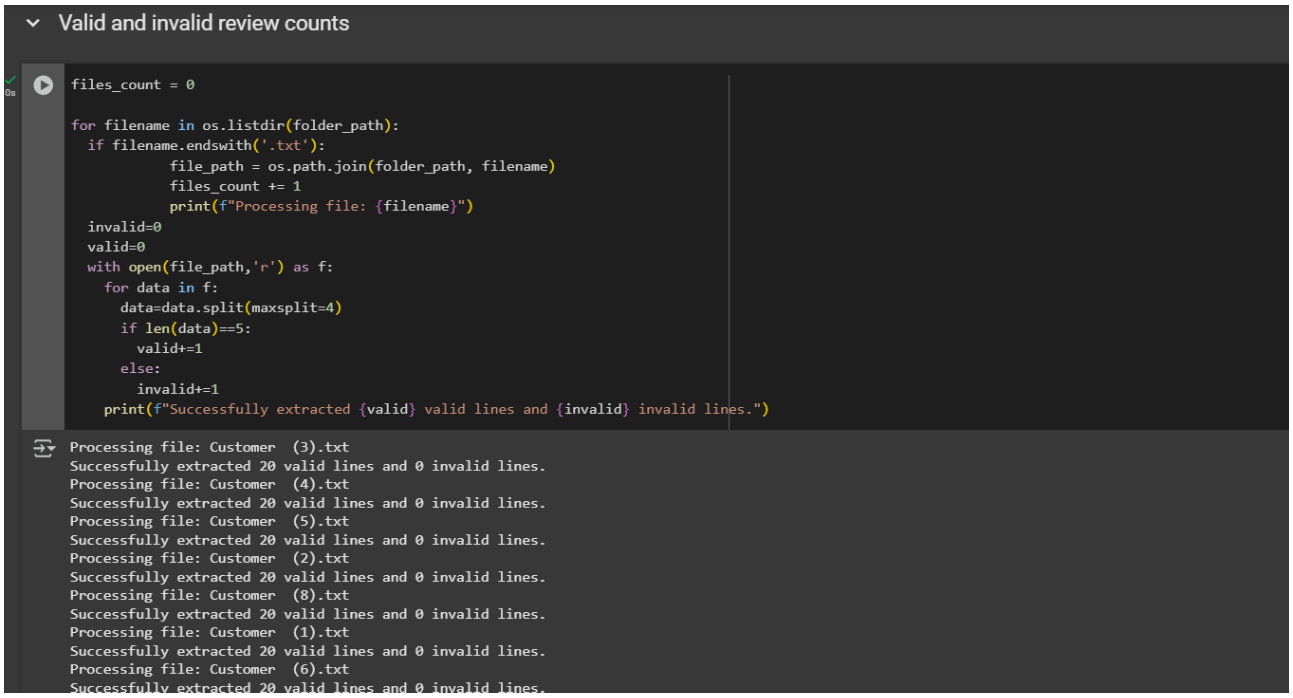
```python
import os
from collections import defaultdict
```



Reading 10 files

```python
folder_path='/content/Customer_Review/'
files_count = 0

for filename in os.listdir(folder_path):
  if filename.endswith('.txt'):
          file_path = os.path.join(folder_path, filename)
          files_count += 1
          print(f"Processing file: {filename}")

  with open(file_path,'r') as f:
        for data in f:
          data=data.split(' ')
          print(data)
```

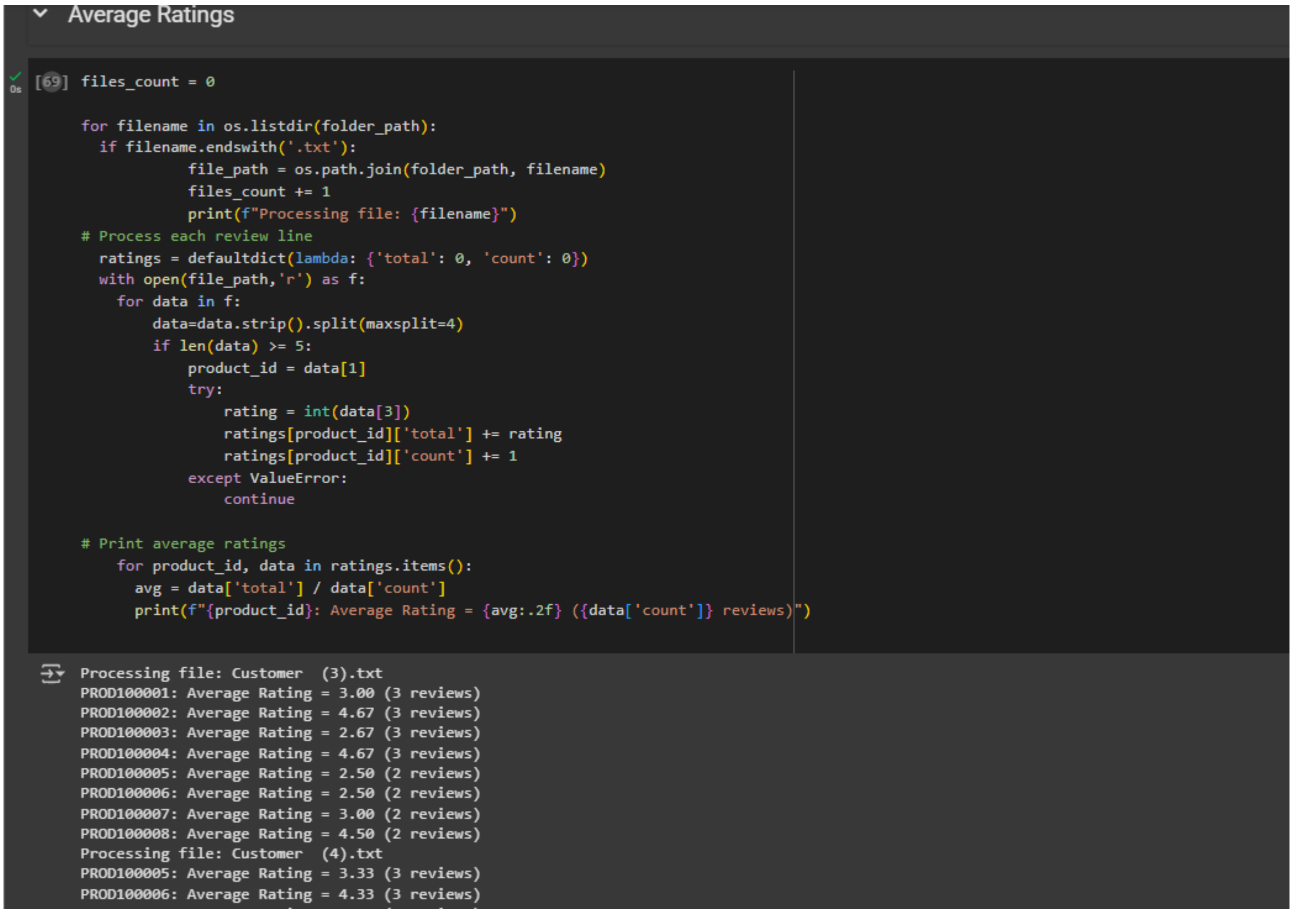
```
['M3N4O5', 'PROD100001', '2024-02-14', '3', 'Pretty', 'standard', 'no', 'real', 'standout', 'features.\n']
['Q6R7S8', 'PROD100002', '2024-02-19', '4', 'Reliable', 'performance', 'and', 'good', 'battery', 'life.\n']
['U9V0W1', 'PROD100003', '2024-02-24', '1', 'Absolute', 'garbage', "don't", 'waste', 'your', 'money.\n']
['Y2Z3A4', 'PROD100004', '2024-02-29', '5', 'A', 'true', 'game-changer', 'for', 'my', 'productivity!\n']
['C5D6E7', 'PROD100005', '2024-03-05', '3', "It's", 'acceptable', 'but', 'nothing', 'to', 'write', 'home', 'about.\n']
['G8H9I0', 'PROD100006', '2024-03-10', '4', 'Solid', 'build', 'feels', 'robust.\n']
['K1L2M3', 'PROD100007', '2024-03-15', '2', 'Connectivity', 'issues', 'make', 'it', 'frustrating', 'to', 'use.\n']
['O4P5Q6', 'PROD100008', '2024-03-20', '5', 'Beyond', 'expectations', 'truly', 'magnificent.\n']
['S7T8U9', 'PROD100001', '2024-03-25', '3', 'Fairly', 'basic', 'but', 'it', 'does', 'function.\n']
['W0X1Y2', 'PROD100002', '2024-03-30', '1', 'Completely', 'fell', 'apart', 'within', 'a', 'month.\n']
['A3B4C5', 'PROD100003', '2024-04-04', '5', 'So', 'easy', 'to', 'use', 'and', 'incredibly', 'effective.\n']
['E6F7G8', 'PROD100004', '2024-04-09', '4', 'Performs', 'well', 'under', 'various', 'conditions.\n']
['I9J0K1', 'PROD100005', '2024-04-14', '2', 'Instructions', 'are', 'poorly', 'translated.\n']
['M2N3O4', 'PROD100006', '2024-04-19', '5', 'Highly', 'recommended', 'for', 'anyone', 'in', 'need.\n']
['Q5R6S7', 'PROD100007', '2024-04-24', '3', 'Decent', 'but', "I've", 'seen', 'better.\n']
['U8V9W0', 'PROD100008', '2024-04-29', '4', 'Good', 'lifespan', 'and', 'consistent', 'output.']
Processing file: Customer  (9).txt
```



Valid and invalid review counts

```python
files_count = 0

for filename in os.listdir(folder_path):
  if filename.endswith('.txt'):
          file_path = os.path.join(folder_path, filename)
          files_count += 1
          print(f"Processing file: {filename}")
  invalid=0
  valid=0
  with open(file_path,'r') as f:
    for data in f:
      data=data.split(maxsplit=4)
      if len(data)==5:
        valid+=1
      else:
        invalid+=1
    print(f"Successfully extracted {valid} valid lines and {invalid} invalid lines.")
```

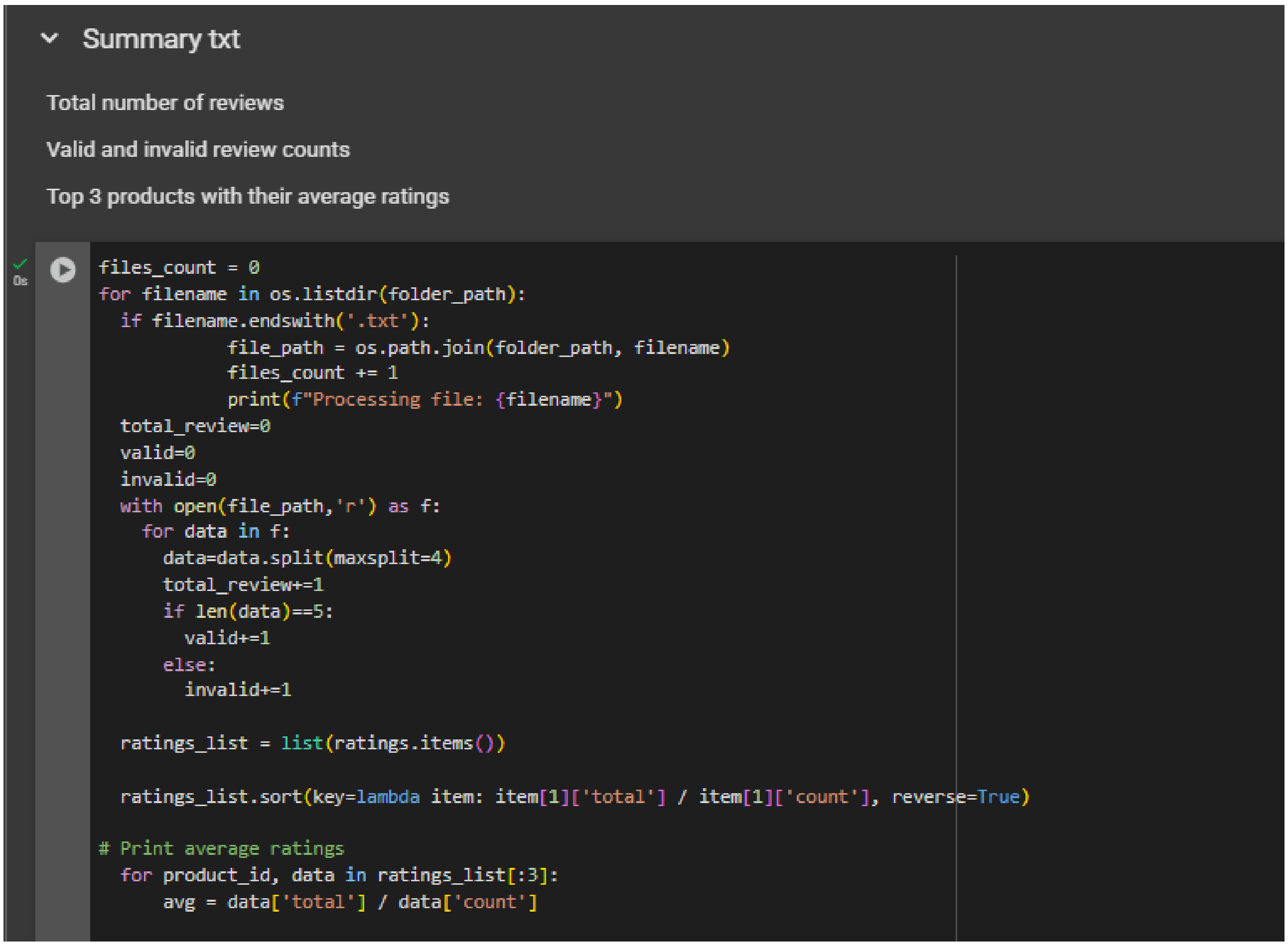
```
Processing file: Customer  (3).txt
Successfully extracted 20 valid lines and 0 invalid lines.
Processing file: Customer  (4).txt
Successfully extracted 20 valid lines and 0 invalid lines.
Processing file: Customer  (5).txt
Successfully extracted 20 valid lines and 0 invalid lines.
Processing file: Customer  (2).txt
Successfully extracted 20 valid lines and 0 invalid lines.
Processing file: Customer  (8).txt
Successfully extracted 20 valid lines and 0 invalid lines.
Processing file: Customer  (1).txt
Successfully extracted 20 valid lines and 0 invalid lines.
Processing file: Customer  (6).txt
Successfully extracted 20 valid lines and 0 invalid lines.
```



Average Ratings

```
files_count = 0

for filename in os.listdir(folder_path):
  if filename.endswith('.txt'):
            file_path = os.path.join(folder_path, filename)
            files_count += 1
            print(f"Processing file: {filename}")
# Process each review line
  ratings = defaultdict(lambda: {'total': 0, 'count': 0})
  with open(file_path,'r') as f:
    for data in f:
        data=data.strip().split(maxsplit=4)
        if len(data) >= 5:
            product_id = data[1]
            try:
                rating = int(data[3])
                ratings[product_id]['total'] += rating
                ratings[product_id]['count'] += 1
            except ValueError:
                continue

# Print average ratings
    for product_id, data in ratings.items():
      avg = data['total'] / data['count']
      print(f"{product_id}: Average Rating = {avg:.2f} ({data['count']} reviews)")
```

```
Processing file: Customer  (3).txt
PROD100001: Average Rating = 3.00 (3 reviews)
PROD100002: Average Rating = 4.67 (3 reviews)
PROD100003: Average Rating = 2.67 (3 reviews)
PROD100004: Average Rating = 4.67 (3 reviews)
PROD100005: Average Rating = 2.50 (2 reviews)
PROD100006: Average Rating = 2.50 (2 reviews)
PROD100007: Average Rating = 3.00 (2 reviews)
PROD100008: Average Rating = 4.50 (2 reviews)
Processing file: Customer  (4).txt
PROD100005: Average Rating = 3.33 (3 reviews)
PROD100006: Average Rating = 4.33 (3 reviews)
```



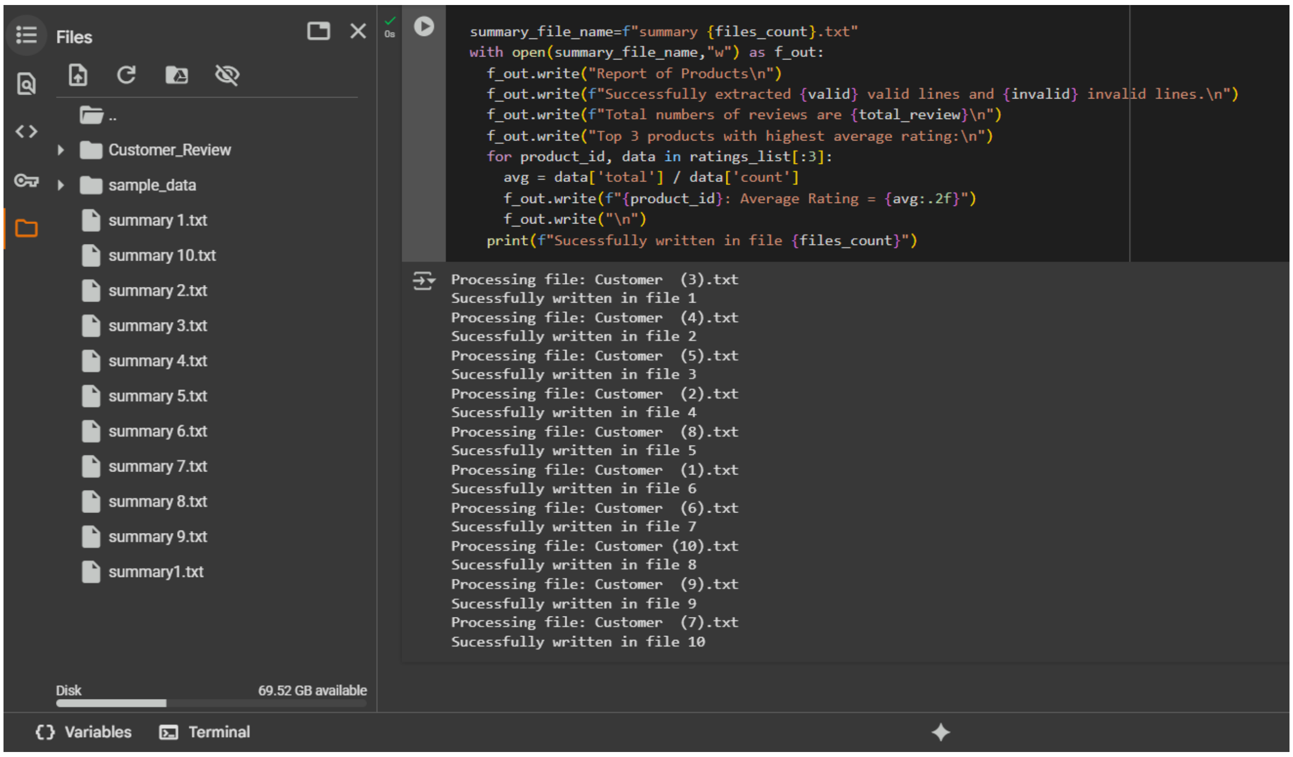
Summary txt

Total number of reviews

Valid and invalid review counts

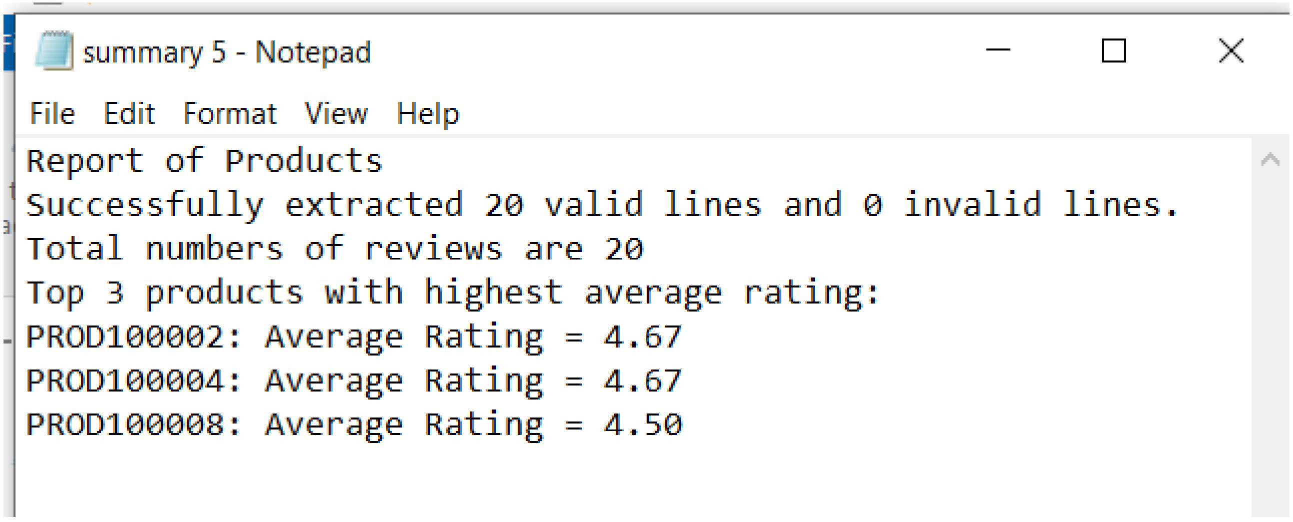
Top 3 products with their average ratings

```
files_count = 0
for filename in os.listdir(folder_path):
  if filename.endswith('.txt'):
            file_path = os.path.join(folder_path, filename)
            files_count += 1
            print(f"Processing file: {filename}")
  total_review=0
  valid=0
  invalid=0
  with open(file_path,'r') as f:
    for data in f:
      data=data.split(maxsplit=4)
      total_review+=1
      if len(data)==5:
        valid+=1
      else:
        invalid+=1

  ratings_list = list(ratings.items())

  ratings_list.sort(key=lambda item: item[1]['total'] / item[1]['count'], reverse=True)

# Print average ratings
  for product_id, data in ratings_list[:3]:
      avg = data['total'] / data['count']
```



```
summary_file_name=f"summary {files_count}.txt"
with open(summary_file_name,"w") as f_out:
  f_out.write("Report of Products\n")
  f_out.write(f"Successfully extracted {valid} valid lines and {invalid} invalid lines.\n")
  f_out.write(f"Total numbers of reviews are {total_review}\n")
  f_out.write("Top 3 products with highest average rating:\n")
  for product_id, data in ratings_list[:3]:
    avg = data['total'] / data['count']
    f_out.write(f"{product_id}: Average Rating = {avg:.2f}")
    f_out.write("\n")
  print(f"Sucessfully written in file {files_count}")
```

```
Processing file: Customer  (3).txt
Sucessfully written in file 1
Processing file: Customer  (4).txt
Sucessfully written in file 2
Processing file: Customer  (5).txt
Sucessfully written in file 3
Processing file: Customer  (2).txt
Sucessfully written in file 4
Processing file: Customer  (8).txt
Sucessfully written in file 5
Processing file: Customer  (1).txt
Sucessfully written in file 6
Processing file: Customer  (6).txt
Sucessfully written in file 7
Processing file: Customer (10).txt
Sucessfully written in file 8
Processing file: Customer  (9).txt
Sucessfully written in file 9
Processing file: Customer  (7).txt
Sucessfully written in file 10
```



```
Report of Products
Successfully extracted 20 valid lines and 0 invalid lines.
Total numbers of reviews are 20
Top 3 products with highest average rating:
PROD100002: Average Rating = 4.67
PROD100004: Average Rating = 4.67
PROD100008: Average Rating = 4.50
```