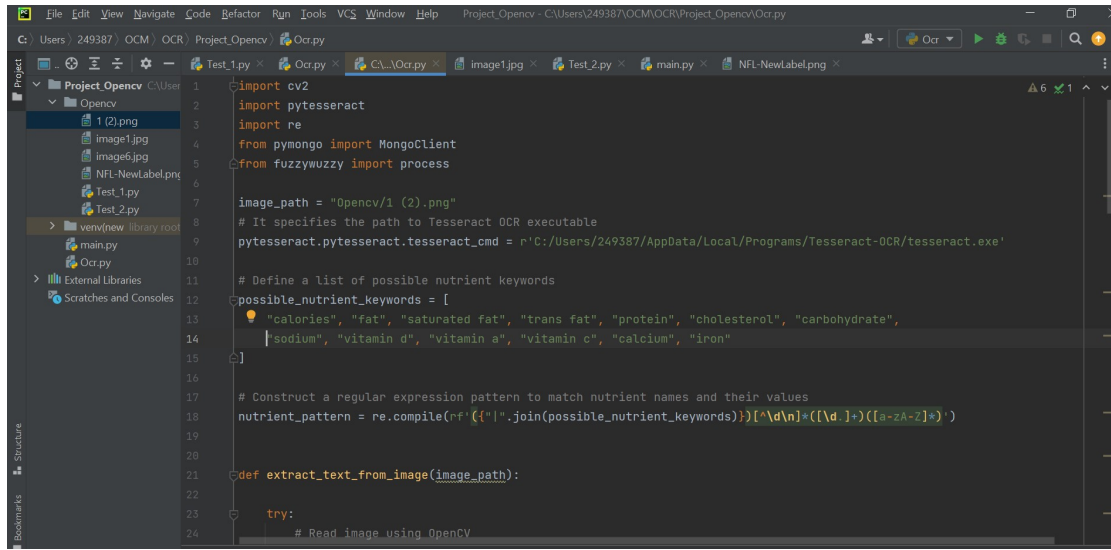


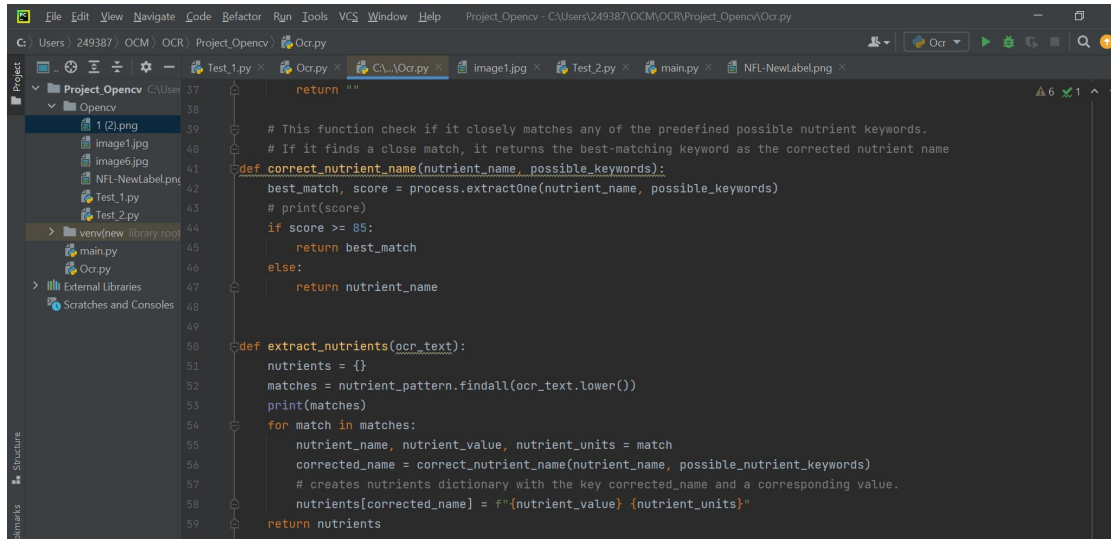
# Python Developer – Initial Screening for UST (SEMICON) – Oct 2023

## CODE:



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help Project_Opencv - C:\Users\249387\OCM\OCR\Project_Opencv\Ocr.py
C:\Users\249387\OCM\OCR\Project_Opencv\Ocr.py
Project_Opencv
  Opencv
    1 (2).png
    image1.jpg
    image6.jpg
    NFL-NewLabel.png
    Test_1.py
    Test_2.py
  venv\new_library_root
  main.py
  Ocr.py
  External Libraries
  Scratches and Consoles
Structure
Bookmarks

1 import cv2
2 import pytesseract
3 import re
4 from pymongo import MongoClient
5 from fuzzywuzzy import process
6
7 image_path = "Opencv/1 (2).png"
8 # It specifies the path to Tesseract OCR executable
9 pytesseract.pytesseract.tesseract_cmd = r'C:\Users\249387\AppData\Local\Programs\Tesseract-OCR\tesseract.exe'
10
11 # Define a list of possible nutrient keywords
12 possible_nutrient_keywords = [
13     "calories", "fat", "saturated fat", "trans fat", "protein", "cholesterol", "carbohydrate",
14     "sodium", "vitamin d", "vitamin a", "vitamin c", "calcium", "iron"
15 ]
16
17 # Construct a regular expression pattern to match nutrient names and their values
18 nutrient_pattern = re.compile(rf'({"|".join(possible_nutrient_keywords)})[\d\n]*([d.]+)([a-zA-Z]*)')
19
20
21 def extract_text_from_image(image_path):
22
23     try:
24         # Read image using OpenCV
```



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help Project_Opencv - C:\Users\249387\OCM\OCR\Project_Opencv\Ocr.py
C:\Users\249387\OCM\OCR\Project_Opencv\Ocr.py
Project_Opencv
  Opencv
    1 (2).png
    image1.jpg
    image6.jpg
    NFL-NewLabel.png
    Test_1.py
    Test_2.py
  venv\new_library_root
  main.py
  Ocr.py
  External Libraries
  Scratches and Consoles
Structure
Bookmarks

37 return ""
38
39 # This function check if it closely matches any of the predefined possible nutrient keywords.
40 # If it finds a close match, it returns the best-matching keyword as the corrected nutrient name
41 def correct_nutrient_name(nutrient_name, possible_keywords):
42     best_match, score = process.extractOne(nutrient_name, possible_keywords)
43     # print(score)
44     if score >= 85:
45         return best_match
46     else:
47         return nutrient_name
48
49
50 def extract_nutrients(ocr_text):
51     nutrients = {}
52     matches = nutrient_pattern.findall(ocr_text.lower())
53     print(matches)
54     for match in matches:
55         nutrient_name, nutrient_value, nutrient_units = match
56         corrected_name = correct_nutrient_name(nutrient_name, possible_nutrient_keywords)
57         # creates nutrients dictionary with the key corrected_name and a corresponding value.
58         nutrients[corrected_name] = f"{nutrient_value} {nutrient_units}"
59     return nutrients
```

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help Project_Opencv - C:\Users\249387\OCM\OCR\Project_Opencv\Ocr.py
G:\ Users\ 249387\ OCM\ OCR\ Project_Opencv\ Ocr.py
Test_1.py x Ocr.py x C:\Users\249387\OCM\OCR\Project_Opencv\Ocr.py x image1.jpg x Test_2.py x main.py x NFL-NewLabel.png x
Project
  Project_Opencv
    OpenCV
      1 (2).png
      image1.jpg
      image6.jpg
      NFL-NewLabel.png
      Test_1.py
      Test_2.py
    venv\new_library_root
      main.py
      Ocr.py
    External Libraries
    Scratches and Consoles
Structure
16
17 # Construct a regular expression pattern to match nutrient names and their values
18 nutrient_pattern = re.compile(r'({}|'.join(possible_nutrient_keywords))['^\\d\\n']*([\\d.1+]*([a-zA-Z]+)?)')
19
20
21 def extract_text_from_image(image_path):
22
23     try:
24         # Read image using OpenCV
25         img = cv2.imread(image_path)
26         # Converted the input image from a colored image to a grayscale image
27         img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
28         # Applies a binary threshold to the grayscale image.
29         thresh, image_th = cv2.threshold(img_gray, thresh: 150, maxval: 180, cv2.THRESH_BINARY)
30         # print(thresh)
31         # cv2.imshow("Window Title", image_th)
32         # cv2.waitKey(0)
33         text = pytesseract.image_to_string(image_th)
34         return text
35     except Exception as e:
36         print(f"Error extracting text: {str(e)}")
37         return ""
38
```

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help Project_Opencv - C:\Users\249387\OCM\OCR\Project_Opencv\Ocr.py
G:\ Users\ 249387\ OCM\ OCR\ Project_Opencv\ Ocr.py
Test_1.py x Ocr.py x C:\Users\249387\OCM\OCR\Project_Opencv\Ocr.py x image1.jpg x Test_2.py x main.py x NFL-NewLabel.png x
Project
  Project_Opencv
    OpenCV
      1 (2).png
      image1.jpg
      image6.jpg
      NFL-NewLabel.png
      Test_1.py
      Test_2.py
    venv\new_library_root
      main.py
      Ocr.py
    External Libraries
    Scratches and Consoles
Structure
58
59 nutrients[corrected_name] = f"{nutrient_value} {nutrient_units}"
60 return nutrients
61
62 # Upload nutrient information to MongoDB
63
64 def upload_to_mongodb(nutrient_info):
65     try:
66         client = MongoClient(f'mongodb+srv://shyma:z3iIfgcL3MpIzwQD@cluster0.xdxdaba.mongodb.net/?retryWrites=true&w=majority')
67         db = client['semicon']
68         collection = db['Data']
69         collection.insert_one(nutrient_info)
70         print("Data uploaded to MongoDB successfully.")
71     except Exception as e:
72         print(f"Error uploading data to MongoDB: {str(e)}")
73
74 # Main
75 if __name__ == "__main__":
76     ocr_text = extract_text_from_image(image_path)
77     print(ocr_text)
78     extracted_nutrients = extract_nutrients(ocr_text)
79
```

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help Project_Opencv - C:\Users\249387\OCM\OCR\Project_Opencv\Ocr.py
G:\ Users\ 249387\ OCM\ OCR\ Project_Opencv\ Ocr.py
Test_1.py x Ocr.py x C:\Users\249387\OCM\OCR\Project_Opencv\Ocr.py x image1.jpg x Test_2.py x main.py x NFL-NewLabel.png x
Project
  Project_Opencv
    OpenCV
      1 (2).png
      image1.jpg
      image6.jpg
      NFL-NewLabel.png
      Test_1.py
      Test_2.py
    venv\new_library_root
      main.py
      Ocr.py
    External Libraries
    Scratches and Consoles
Structure
67
68 collection.insert_one(nutrient_info)
69 print("Data uploaded to MongoDB successfully.")
70
71 except Exception as e:
72     print(f"Error uploading data to MongoDB: {str(e)}")
73
74 # Main
75 if __name__ == "__main__":
76     ocr_text = extract_text_from_image(image_path)
77     print(ocr_text)
78     extracted_nutrients = extract_nutrients(ocr_text)
79
80 if extracted_nutrients:
81     print("Extracted Nutrient Information:")
82     for nutrient, value in extracted_nutrients.items():
83         print(f"{nutrient.capitalize()}: {value}")
84     # Upload the nutrient information to MongoDB
85     upload_to_mongodb(extracted_nutrients)
86 else:
87     print("No nutrient information found in the image.")

```

# Mongo DB Result:

MongoDB Compass - Shyma/semicon.Data  
Connect Edit View Collection Help

Shyma ...

Documents  
semicon.Data

semicon.Data 19 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Indexes Validation

Filter ⓘ Type a query: { field: 'value' } Explain Reset Find ⌕ Options ▶

ADD DATA EXPORT DATA

1 - 4 of 4

```
{
  "_id": ObjectId("652410d86c93c792b17f852a"),
  "calories": "230",
  "fat": "8 g",
  "saturated fat": "1 g",
  "trans fat": "0 g",
  "cholesterol": "0 mg",
  "sodium": "160 mg",
  "carbohydrate": "379",
  "protein": "3 g",
  "vitamin d": "2 mcg",
  "calcium": "260 mg",
  "iron": "8 mg"
}
```

MongoDB Compass - Shyma/semicon.Data  
Connect Edit View Collection Help

Shyma ...

Documents  
semicon.Data

semicon.Data 19 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Indexes Validation

Filter ⓘ Type a query: { field: 'value' } Explain Reset Find ⌕ Options ▶

ADD DATA EXPORT DATA

1 - 4 of 4

```
{
  "_id": ObjectId("652417d9dc57dd18cf1ec8ab"),
  "calories": "230",
  "fat": "8 g",
  "saturated fat": "1 g",
  "trans fat": "0 g",
  "cholesterol": "0 mg",
  "sodium": "160 mg",
  "protein": "3 g",
  "vitamin d": "2 mcg",
  "calcium": "260 mg"
}
```

```
{
  "_id": ObjectId("652418c52659de57d763da8a"),
  "fat": "0 g",
  "cholesterol": "0",
  "sodium": "0",
  "carbohydrate": "175"
}
```

MongoDB Compass - Shyma/semicon.Data  
Connect Edit View Collection Help

Shyma ...

Documents  
semicon.Data

semicon.Data 19 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Indexes Validation

Filter ⓘ Type a query: { field: 'value' } Explain Reset Find ⌕ Options ▶

ADD DATA EXPORT DATA

1 - 4 of 4

```
{
  "cholesterol": "0",
  "sodium": "0",
  "carbohydrate": "175"
}
```

```
{
  "_id": ObjectId("65241a537ae5ba81a581c88b"),
  "fat": "9 g",
  "saturated fat": "4.59",
  "trans fat": "0 g",
  "cholesterol": "35 mg",
  "sodium": "850 mg",
  "carbohydrate": "34 g",
  "protein": "159",
  "vitamin d": "9",
  "calcium": "320 mg",
  "iron": "1.6 mg"
}
```

## Input:

### Nutrition Facts

8 servings per container

**Serving size** 2/3 cup (55g)

**Amount per serving**

**Calories** 230

**% Daily Value\***

**Total Fat** 8g **10%**

Saturated Fat 1g **5%**

*Trans* Fat 0g

**Cholesterol** 0mg **0%**

**Sodium** 160mg **7%**

**Total Carbohydrate** 37g **13%**

Dietary Fiber 4g **14%**

Total Sugars 12g

Includes 10g Added Sugars **20%**

**Protein** 3g

Vitamin D 2mcg 10%

Calcium 260mg 20%

Iron 8mg 45%

Potassium 240mg 6%

\* The % Daily Value (DV) tells you how much a nutrient in a serving of food contributes to a daily diet. 2,000 calories a day is used for general nutrition advice.

## Output:

```
Extracted Nutrient Information:
Calories: 230
Fat: 8 g
Saturated fat: 1 g
Trans fat: 0 g
Cholesterol: 0 mg
Sodium: 160 mg
Carbohydrate: 37g
Protein: 3 g
Vitamin d: 2 meg
Calcium: 260 mg
Iron: 8 mg
Data uploaded to MongoDB successfully.

Process finished with exit code 0
|
```