eds(Theory Activity No. 1)

NAME: SIDDHANT BALASO SHINDE

ROLL no: CS1-77

BATCH: C14

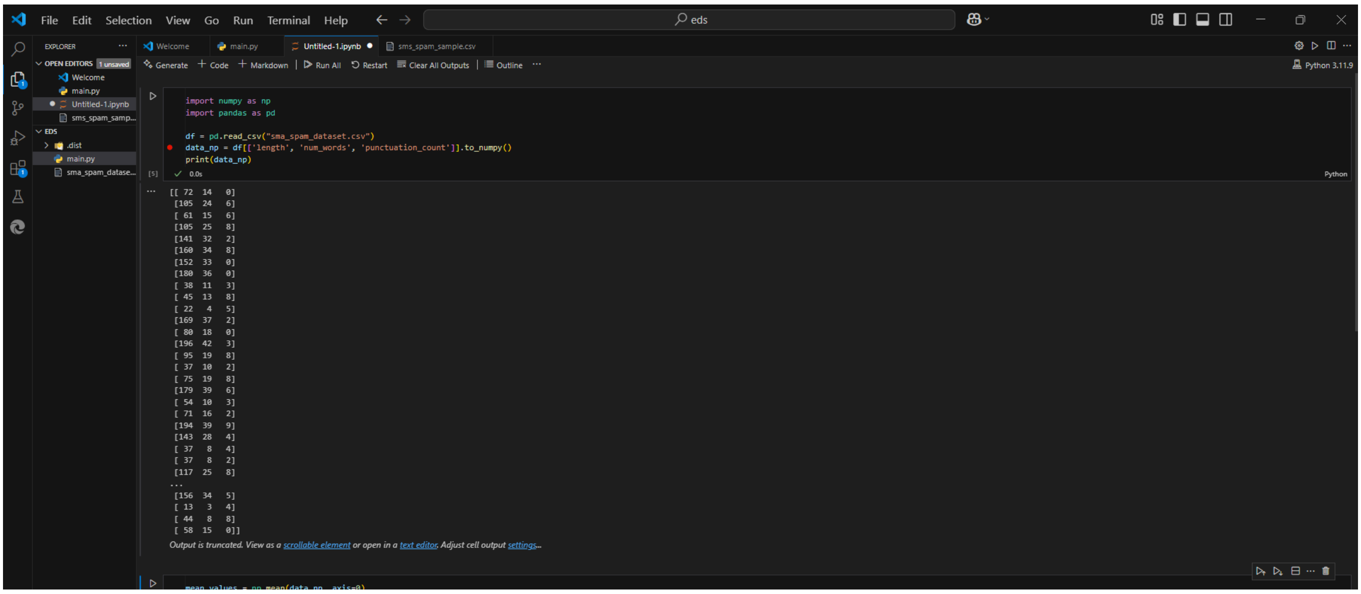
PRN: 202401040250

dataset: SMS SPAM COLLECTION

numpy:

1. Convert DataFrame columns to NumPy array

solution:



```
import numpy as np
import pandas as pd

df = pd.read_csv("sma_spam_dataset.csv")
data_np = df[['length', 'num_words', 'punctuation_count']].to_numpy()
print(data_np)
```

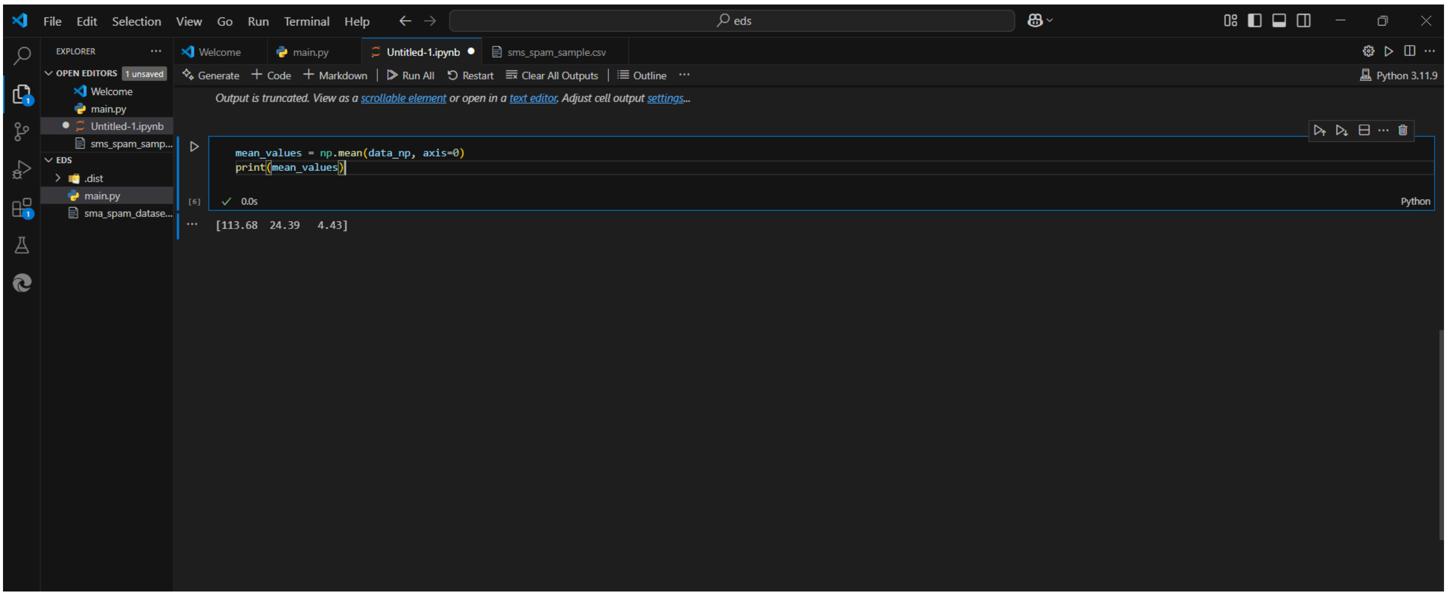
[5] ✓ 0.0s

```
[[ 72  14   0]
 [105  24   6]
 [ 61  15   6]
 [105  25   8]
 [141  32   2]
 [160  34   8]
 [152  33   0]
 [180  36   0]
 [ 38  11   3]
 [ 45  13   8]
 [ 22   4   5]
 [169  37   2]
 [ 80  18   0]
 [196  42   3]
 [ 95  19   8]
 [ 37  10   2]
 [ 75  19   8]
 [179  39   6]
 [ 54  10   3]
 [ 71  16   2]
 [194  39   9]
 [143  28   4]
 [ 37   8   4]
 [ 37   8   2]
 [117  25   8]
...
 [156  34   5]
 [ 13   3   4]
 [ 44   8   8]
 [ 58  15   0]]
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

```
mean_values = np.mean(data_np, axis=0)
```

2. Calculate mean of each numeric column



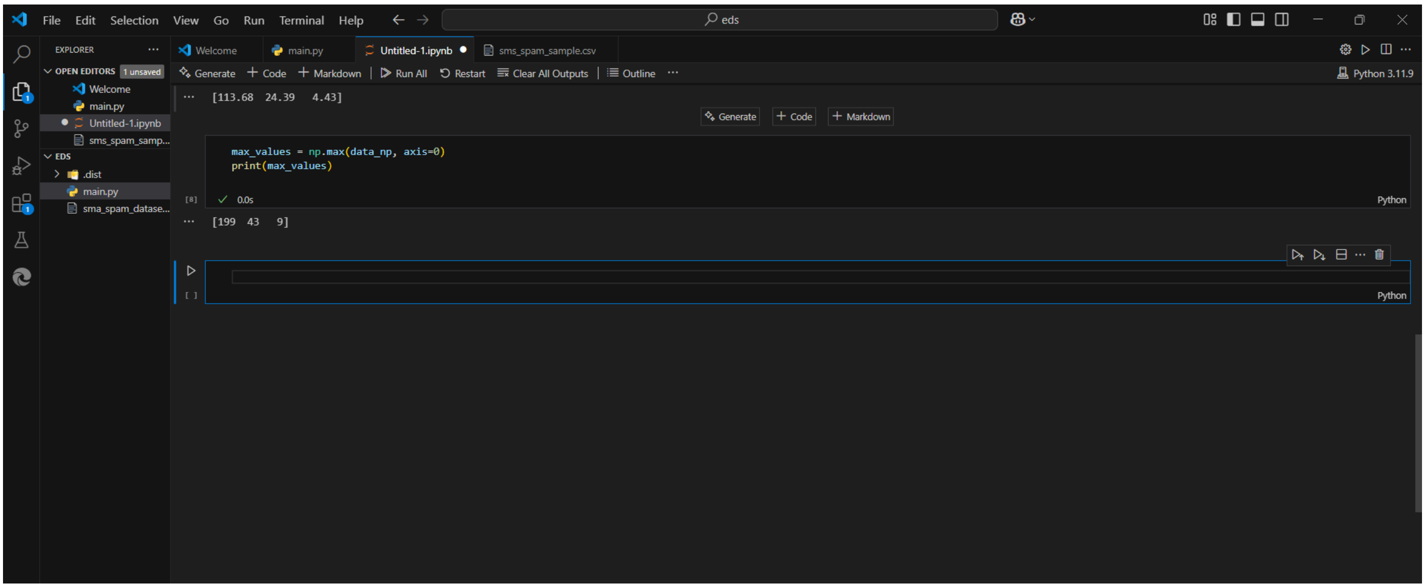
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

```
mean_values = np.mean(data_np, axis=0)
print(mean_values)
```

```
[113.68  24.39   4.43]
```

3. Find max values in each column

solution:



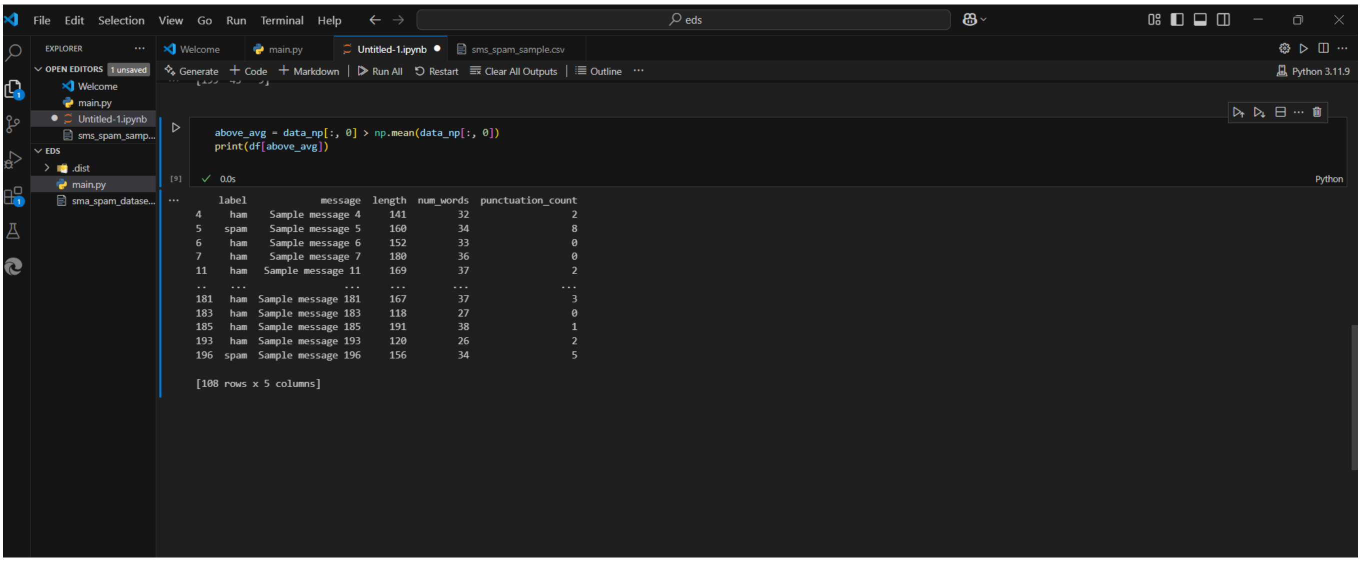
```
[113.68  24.39   4.43]
```

```
max_values = np.max(data_np, axis=0)
print(max_values)
```

```
[199  43   9]
```

4. Find rows where message length is above average

solution:



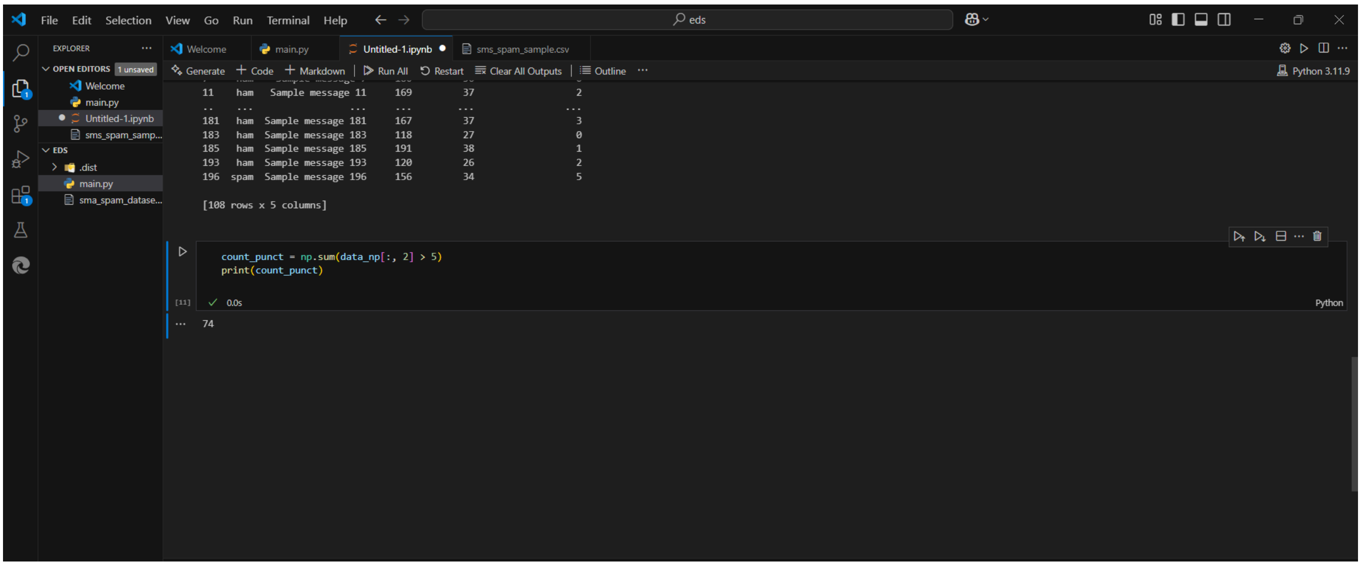
```
above_avg = data_np[:, 0] > np.mean(data_np[:, 0])
print(df[above_avg])
```

```
    label            message  length  num_words  punctuation_count
4     ham   Sample message 4     141         32                  2
5    spam   Sample message 5     160         34                  8
6     ham   Sample message 6     152         33                  0
7     ham   Sample message 7     180         36                  0
11    ham  Sample message 11     169         37                  2
..    ...                ...     ...        ...                ...
181   ham Sample message 181     167         37                  3
183   ham Sample message 183     118         27                  0
185   ham Sample message 185     191         38                  1
193   ham Sample message 193     120         26                  2
196  spam Sample message 196     156         34                  5

[108 rows x 5 columns]
```

5. Count messages with more than 5 punctuation marks

solution:



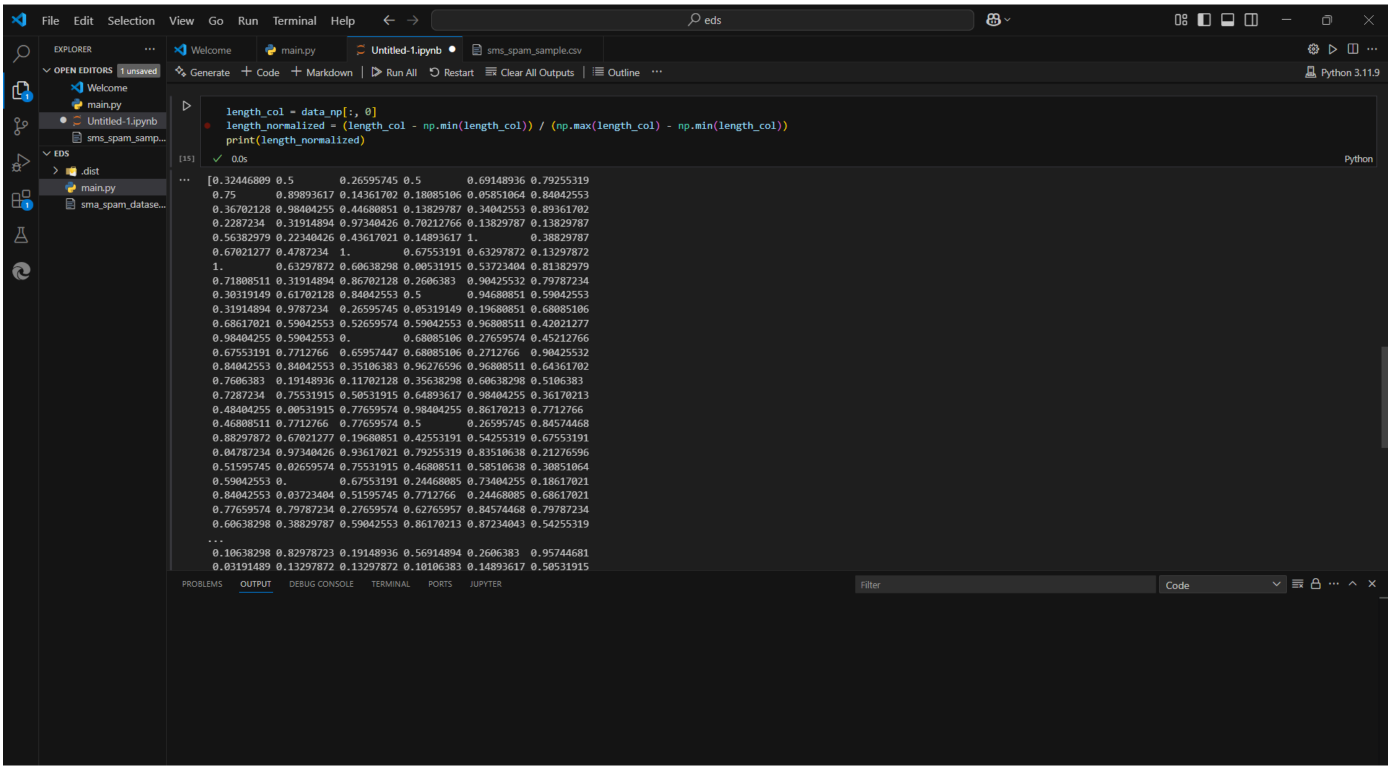
```
11    ham  Sample message 11     169         37                  2
..    ...                ...     ...        ...                ...
181   ham Sample message 181     167         37                  3
183   ham Sample message 183     118         27                  0
185   ham Sample message 185     191         38                  1
193   ham Sample message 193     120         26                  2
196  spam Sample message 196     156         34                  5

[108 rows x 5 columns]
```

```
count_punct = np.sum(data_np[:, 2] > 5)
print(count_punct)
```

```
74
```

6. Normalize the 'length' column



```
length_col = data_np[:, 0]
length_normalized = (length_col - np.min(length_col)) / (np.max(length_col) - np.min(length_col))
print(length_normalized)
```

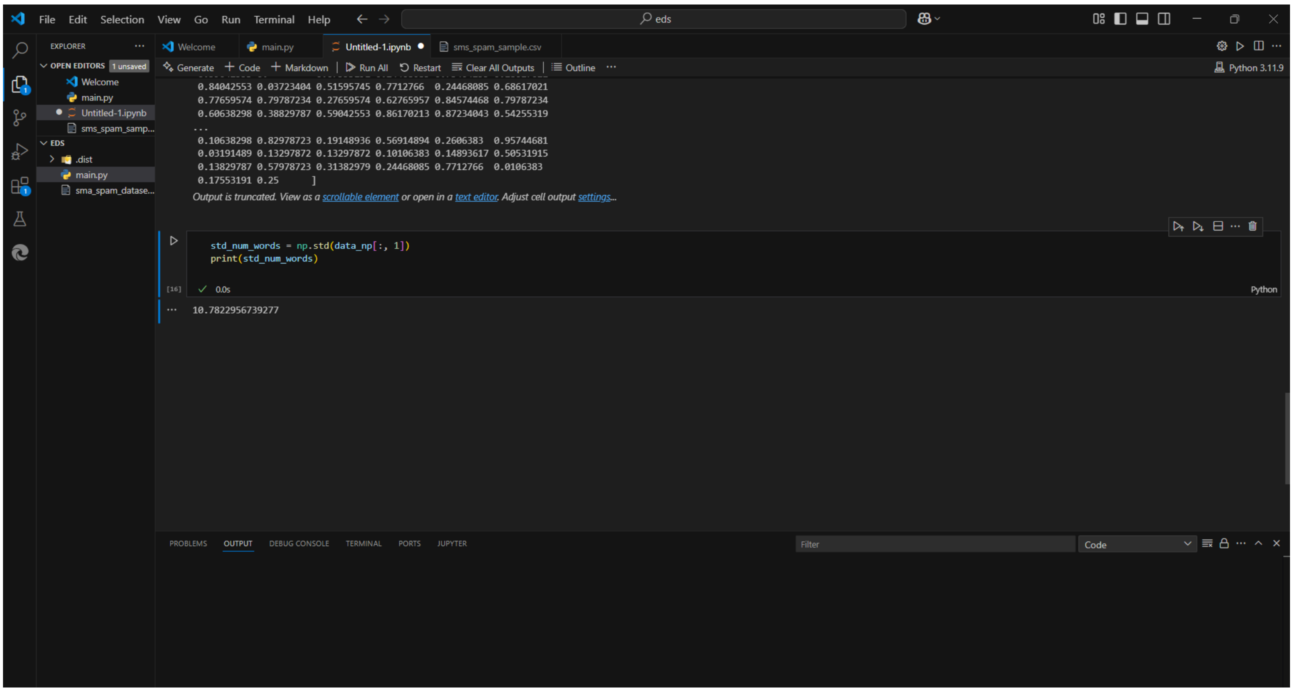
[15] ✓ 0.0s

```
[0.32446809 0.5        0.26595745 0.5        0.69148936 0.79255319
 0.75       0.89893617 0.14361702 0.18085106 0.05851064 0.84042553
 0.36702128 0.98404255 0.44680851 0.13829787 0.34042553 0.89361702
 0.2287234  0.31914894 0.97340426 0.70212766 0.13829787 0.13829787
 0.56382979 0.22340426 0.43617021 0.14893617 1.         0.38829787
 0.67021277 0.4787234  1.         0.67553191 0.63297872 0.13297872
 1.         0.63297872 0.60638298 0.00531915 0.53723404 0.81382979
 0.71808511 0.31914894 0.86702128 0.2606383  0.90425532 0.79787234
 0.30319149 0.61702128 0.84042553 0.5        0.94680851 0.59042553
 0.31914894 0.9787234  0.26595745 0.05319149 0.19680851 0.68085106
 0.68617021 0.59042553 0.52659574 0.59042553 0.96808511 0.42021277
 0.98404255 0.59042553 0.         0.68085106 0.27659574 0.45212766
 0.67553191 0.7712766  0.65957447 0.68085106 0.2712766  0.90425532
 0.84042553 0.84042553 0.35106383 0.96276596 0.96808511 0.64361702
 0.7606383  0.19148936 0.11702128 0.35638298 0.60638298 0.5106383
 0.7287234  0.75531915 0.50531915 0.64893617 0.98404255 0.36170213
 0.48404255 0.00531915 0.77659574 0.98404255 0.86170213 0.7712766
 0.46808511 0.7712766  0.77659574 0.5        0.26595745 0.84574468
 0.88297872 0.67021277 0.19680851 0.42553191 0.54255319 0.67553191
 0.04787234 0.97340426 0.93617021 0.79255319 0.83510638 0.21276596
 0.51595745 0.02659574 0.75531915 0.46808511 0.58510638 0.30851064
 0.59042553 0.         0.67553191 0.24468085 0.73404255 0.18617021
 0.84042553 0.03723404 0.51595745 0.7712766  0.24468085 0.68617021
 0.77659574 0.79787234 0.27659574 0.62765957 0.84574468 0.79787234
 0.60638298 0.38829787 0.59042553 0.86170213 0.87234043 0.54255319
...
 0.10638298 0.82978723 0.19148936 0.56914894 0.2606383  0.95744681
 0.03191489 0.13297872 0.13297872 0.10106383 0.14893617 0.50531915
```

7. Compute standard deviation of 'num_words'

.

solution:



```
 0.84042553 0.03723404 0.51595745 0.7712766  0.24468085 0.68617021
 0.77659574 0.79787234 0.27659574 0.62765957 0.84574468 0.79787234
 0.60638298 0.38829787 0.59042553 0.86170213 0.87234043 0.54255319
...
 0.10638298 0.82978723 0.19148936 0.56914894 0.2606383  0.95744681
 0.03191489 0.13297872 0.13297872 0.10106383 0.14893617 0.50531915
 0.13829787 0.57978723 0.31382979 0.24468085 0.7712766  0.0106383
 0.17553191 0.25      ]
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

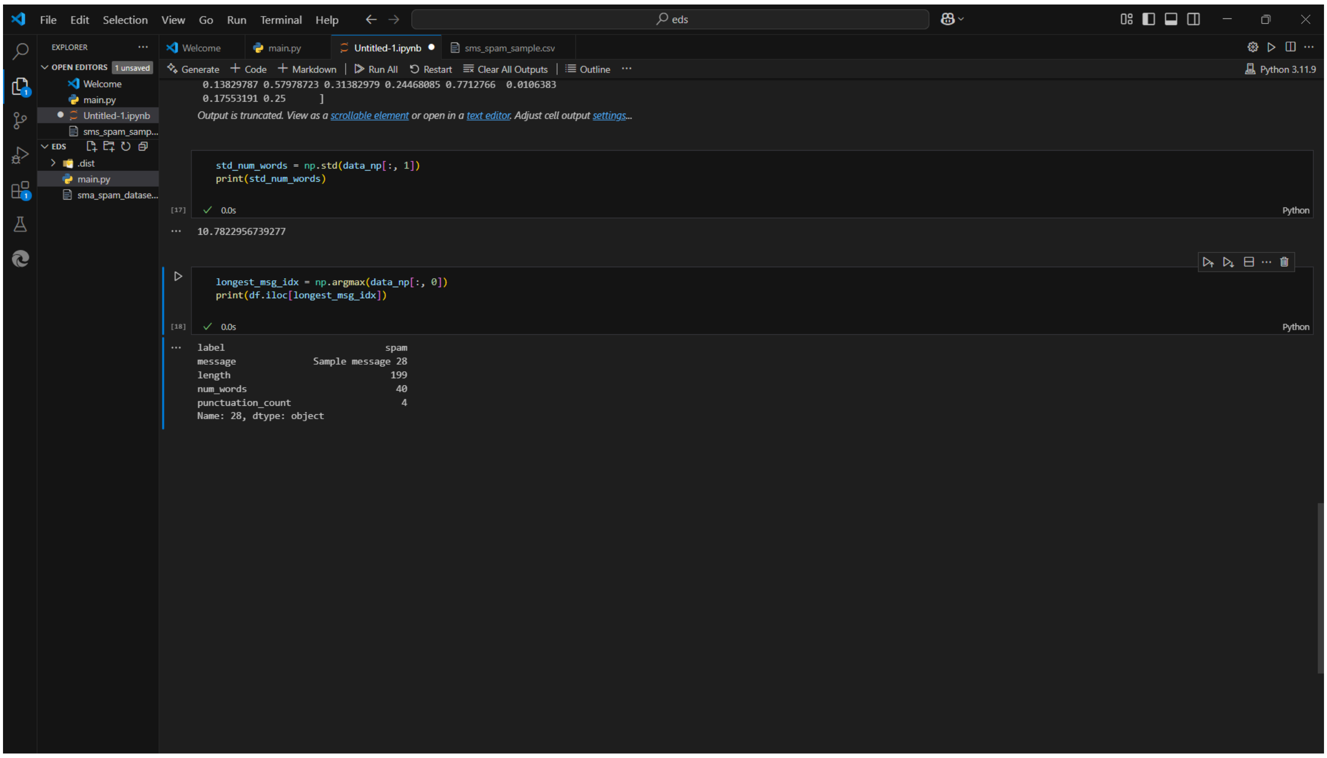
```
std_num_words = np.std(data_np[:, 1])
print(std_num_words)
```

[16] ✓ 0.0s

```
10.7822956739277
```

8. Find index of longest message.

solution:



```
 0.13829787 0.57978723 0.31382979 0.24468085 0.7712766  0.0106383
 0.17553191 0.25      ]
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

```
std_num_words = np.std(data_np[:, 1])
print(std_num_words)
```

[17] ✓ 0.0s

```
10.7822956739277
```

```
longest_msg_idx = np.argmax(data_np[:, 0])
print(df.iloc[longest_msg_idx])
```

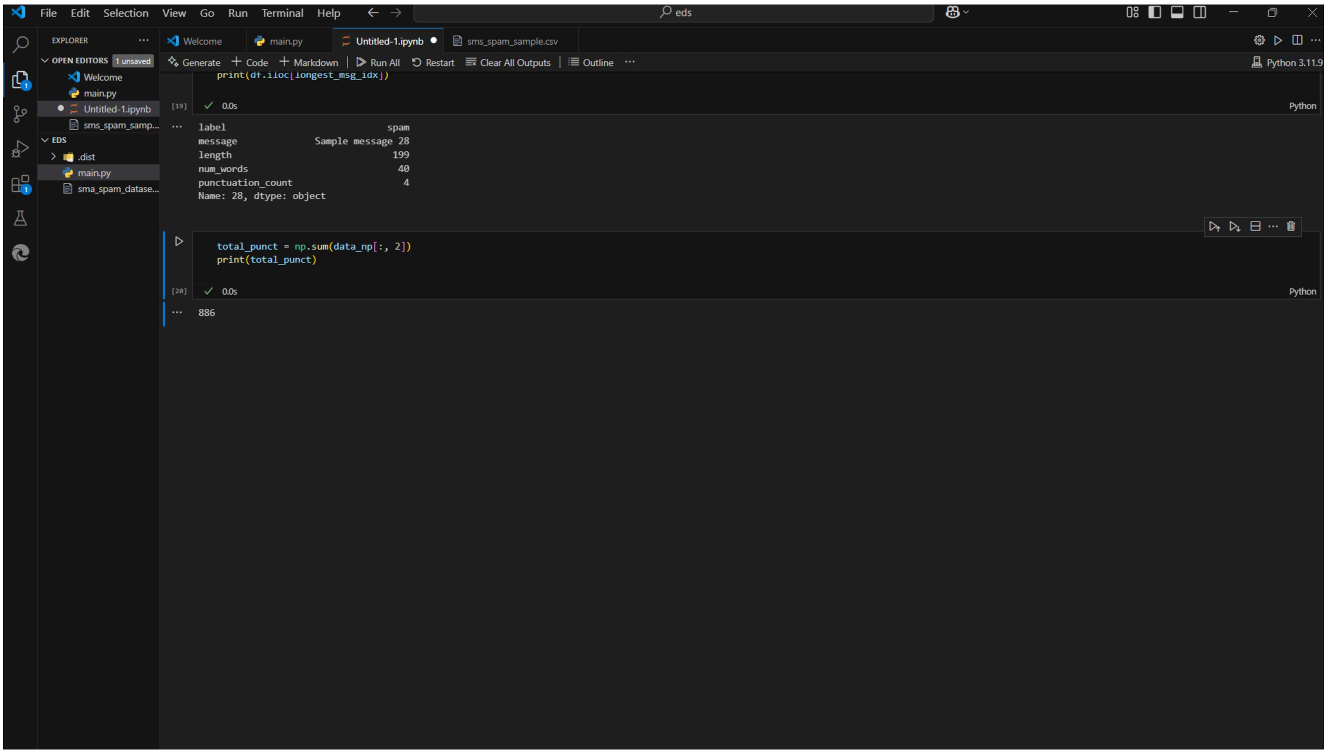
[18] ✓ 0.0s

```
label                           spam
message            Sample message 28
length                           199
num_words                         40
punctuation_count                  4
Name: 28, dtype: object
```

9. Sum of punctuation across all messages

.

solution:



```
print(df.iloc[longest_msg_idx])
```

[19] ✓ 0.0s

```
label                           spam
message            Sample message 28
length                           199
num_words                         40
punctuation_count                  4
Name: 28, dtype: object
```

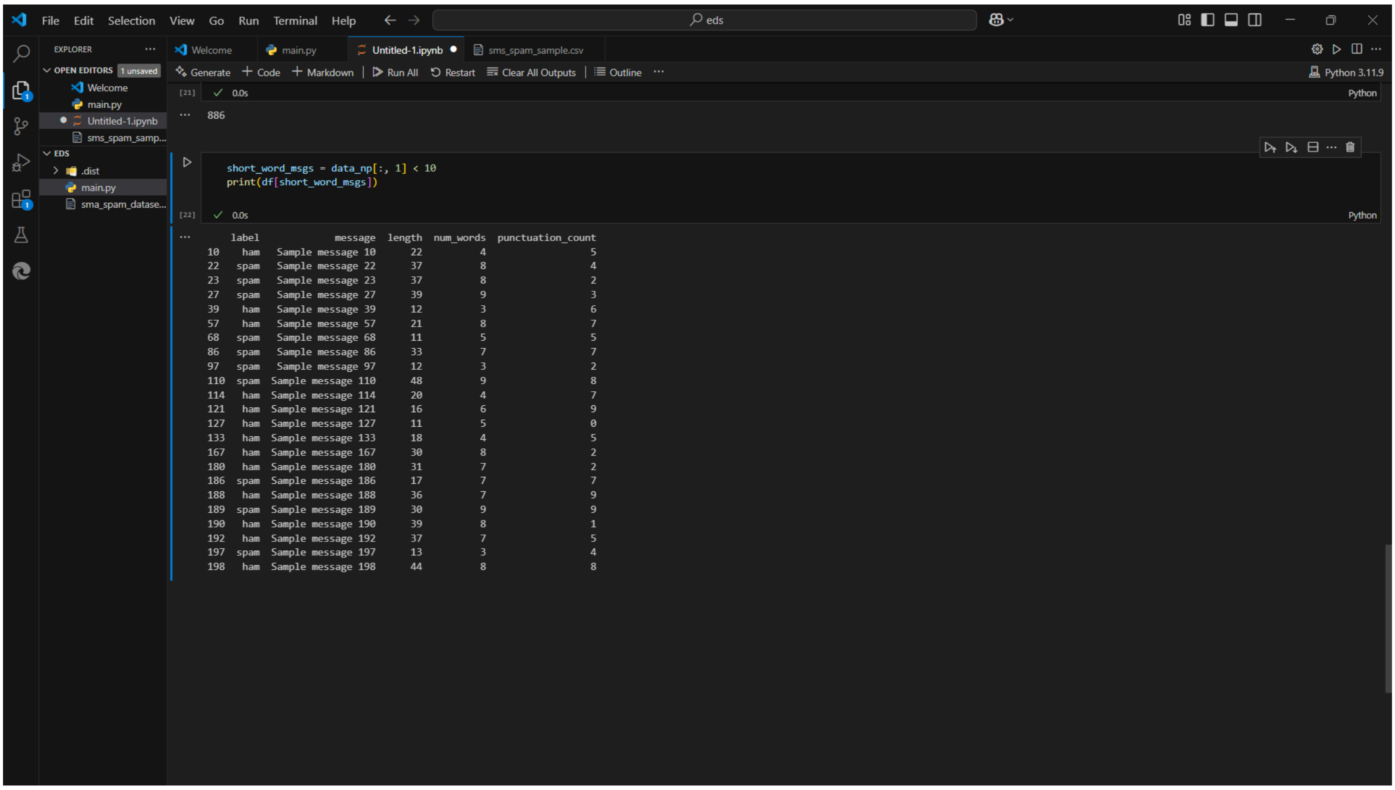
```
total_punct = np.sum(data_np[:, 2])
print(total_punct)
```

[20] ✓ 0.0s

```
886
```

10. Messages where number of words is less than 10

solution:



```
886
```

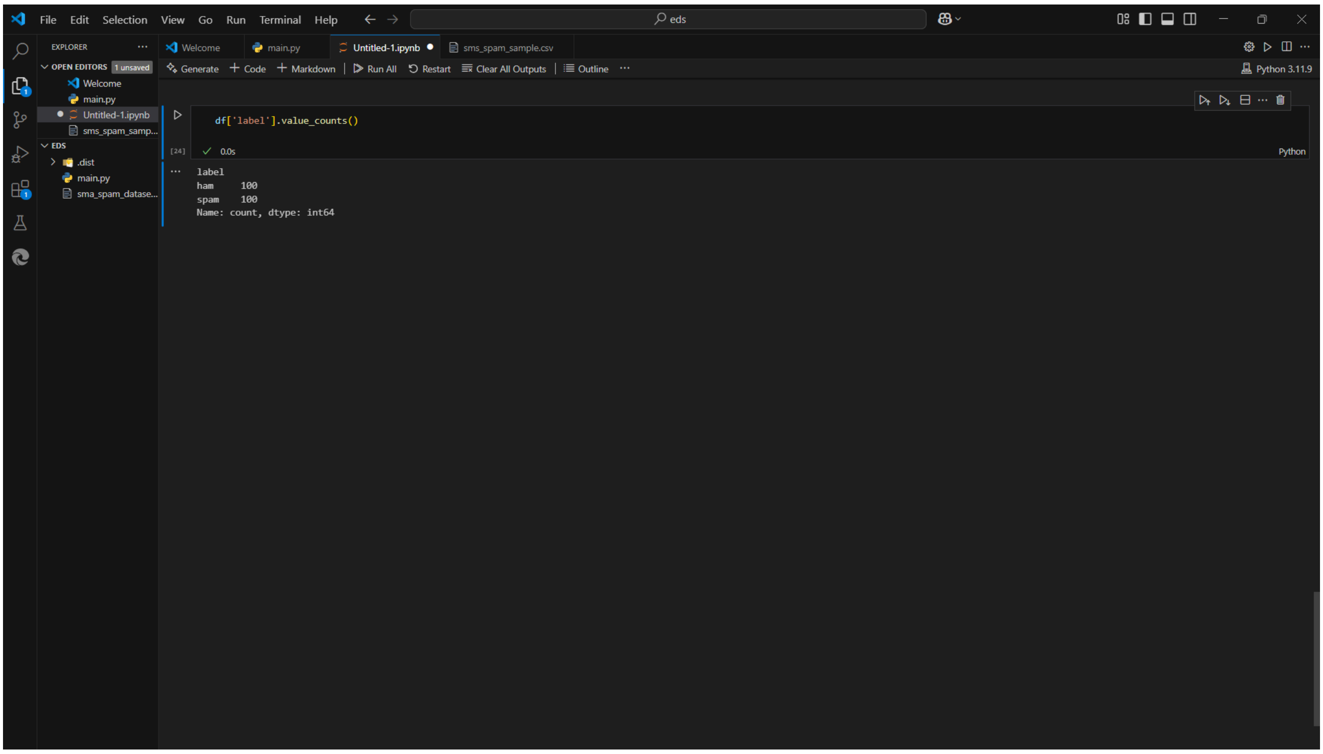
```
short_word_msgs = data_np[:, 1] < 10
print(df[short_word_msgs])
```

```
    label          message  length  num_words  punctuation_count
10    ham   Sample message 10      22          4                  5
22   spam   Sample message 22      37          8                  4
23   spam   Sample message 23      37          8                  2
27   spam   Sample message 27      39          9                  3
39    ham   Sample message 39      12          3                  6
57    ham   Sample message 57      21          8                  7
68   spam   Sample message 68      11          5                  5
86   spam   Sample message 86      33          7                  7
97   spam   Sample message 97      12          3                  2
110  spam  Sample message 110      48          9                  8
114   ham  Sample message 114      20          4                  7
121   ham  Sample message 121      16          6                  9
127   ham  Sample message 127      11          5                  0
133   ham  Sample message 133      18          4                  5
167   ham  Sample message 167      30          8                  2
180   ham  Sample message 180      31          7                  2
186  spam  Sample message 186      17          7                  7
188   ham  Sample message 188      36          7                  9
189  spam  Sample message 189      30          9                  9
190   ham  Sample message 190      39          8                  1
192   ham  Sample message 192      37          7                  5
197  spam  Sample message 197      13          3                  4
198   ham  Sample message 198      44          8                  8
```

pandas:

1. How many spam and ham messages are there?

solution:

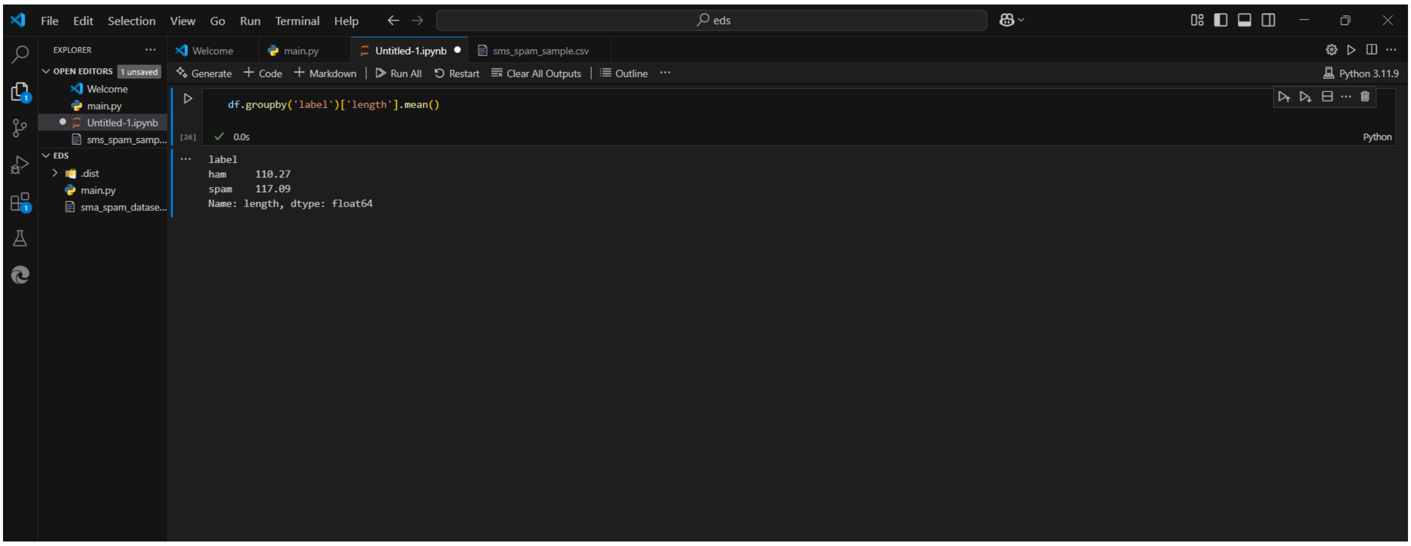


```
df['label'].value_counts()
```

```
label
ham     100
spam    100
Name: count, dtype: int64
```

2. What is the average length of spam and ham messages?.

solution:

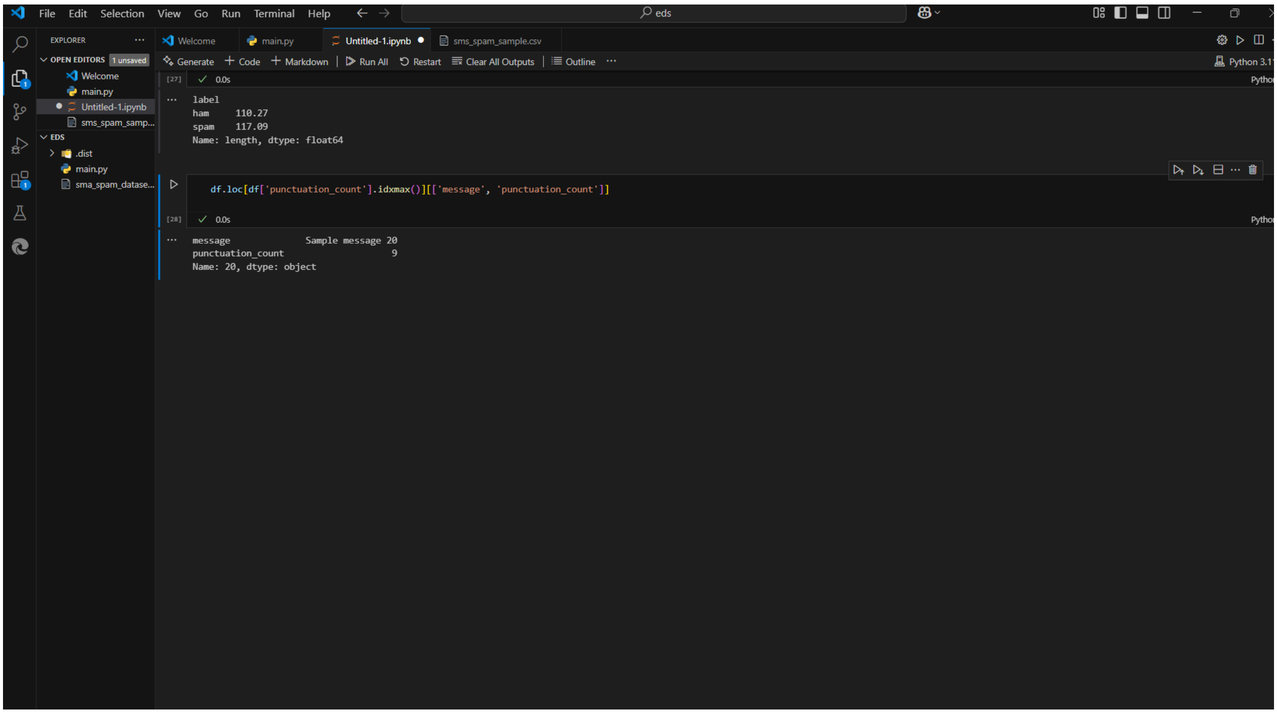


```
df.groupby('label')['length'].mean()
```

```
label
ham     110.27
spam    117.09
Name: length, dtype: float64
```

3. Which message has the most punctuation?

solution:



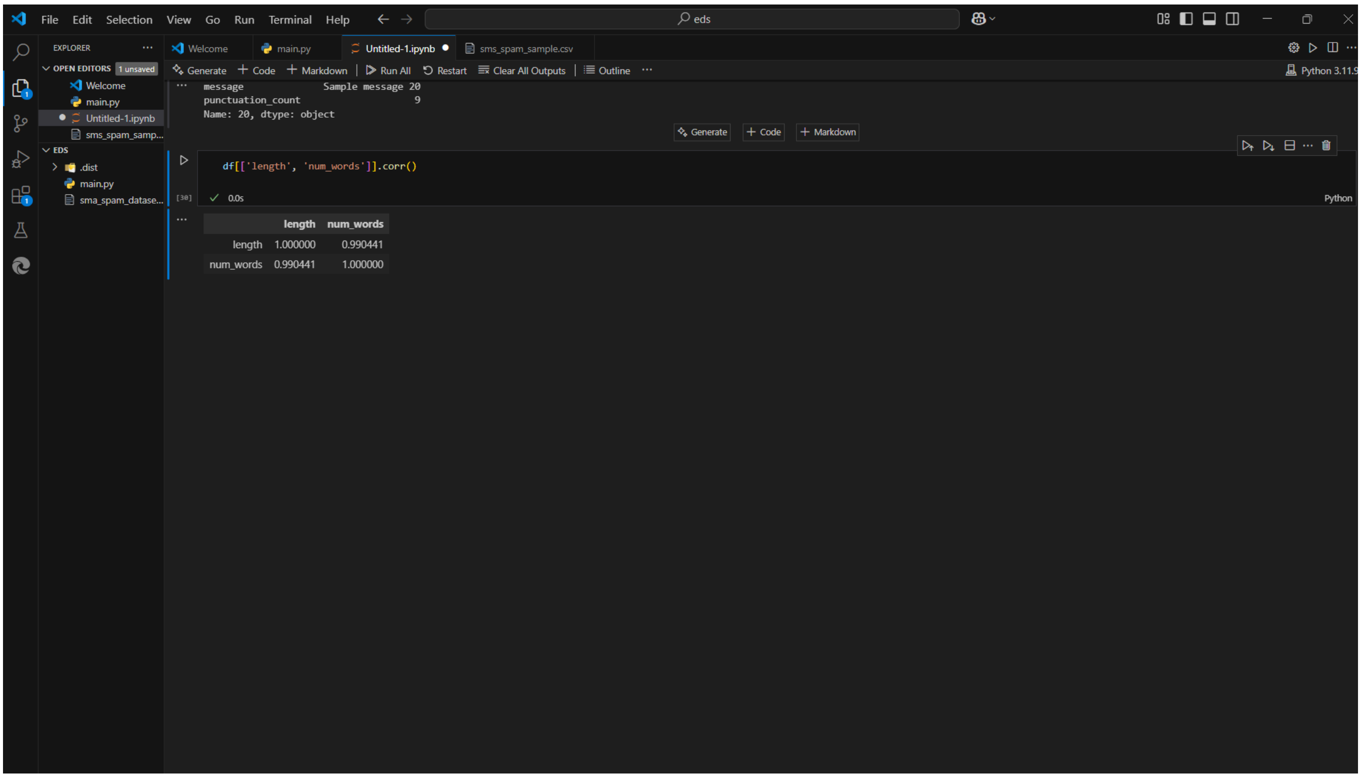
```
label
ham     110.27
spam    117.09
Name: length, dtype: float64
```

```
df.loc[df['punctuation_count'].idxmax()][['message', 'punctuation_count']]
```

```
message              Sample message 20
punctuation_count                    9
Name: 20, dtype: object
```

4. What is the correlation between message length and number of words?

solution:



```
message                 Sample message 20
punctuation_count                       9
Name: 20, dtype: object
```

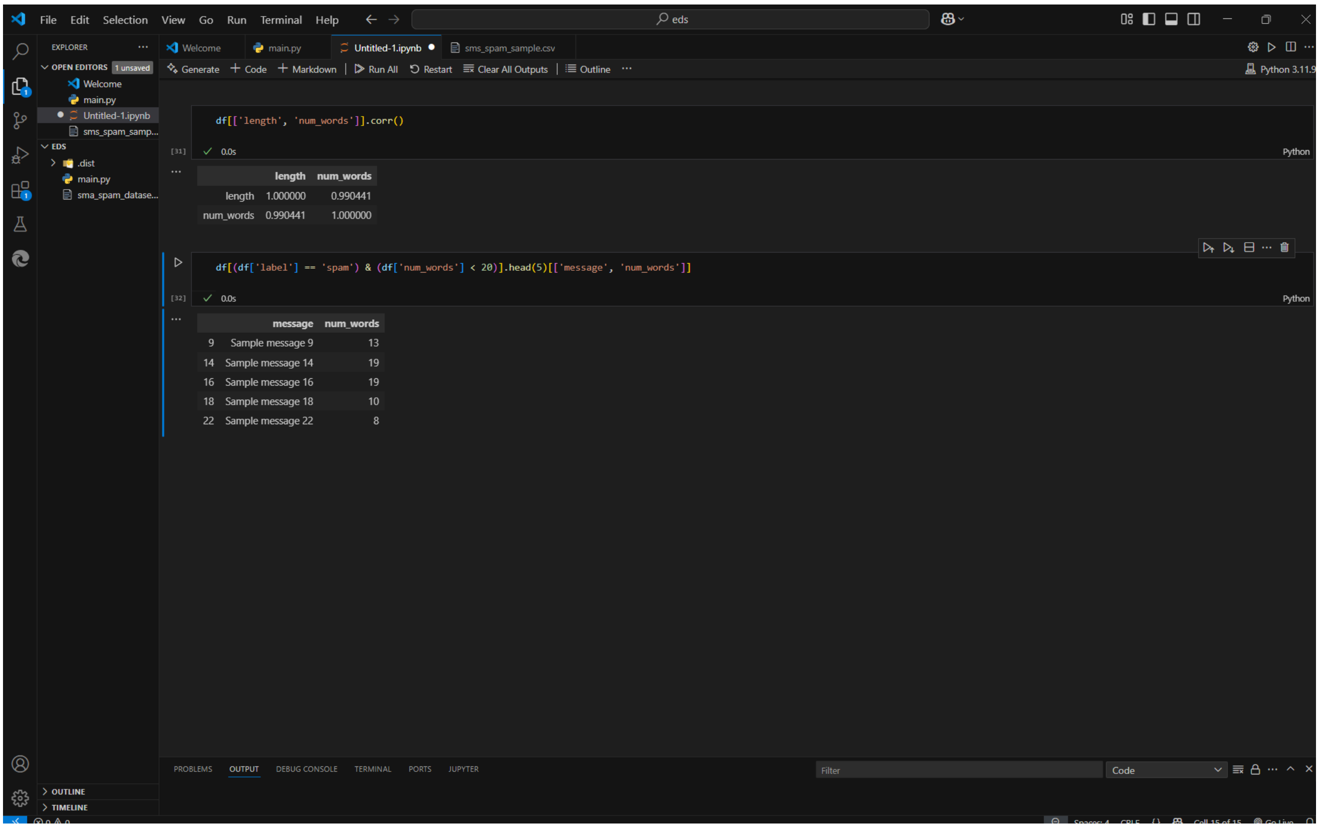
```
df[['length', 'num_words']].corr()
```

[30] ✓ 0.0s

	length	num_words
length	1.000000	0.990441
num_words	0.990441	1.000000

5. What are the first 5 spam messages with less than 20 words?

solution:



```
df[['length', 'num_words']].corr()
```

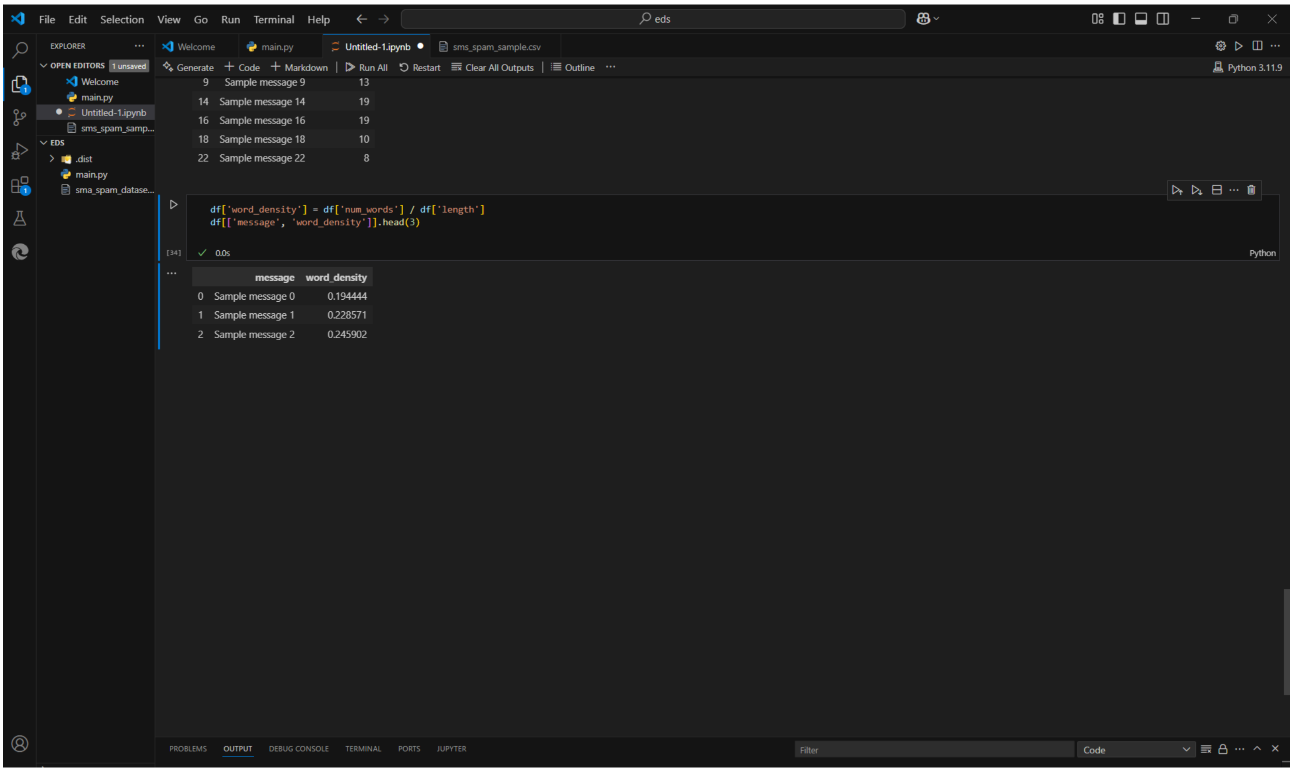
```
           length  num_words
length     1.000000  0.990441
num_words  0.990441  1.000000
```

```
df[(df['label'] == 'spam') & (df['num_words'] < 20)].head(5)[['message', 'num_words']]
```

```
             message  num_words
9    Sample message 9         13
14  Sample message 14         19
16  Sample message 16         19
18  Sample message 18         10
22  Sample message 22          8
```

6. Add a column for word density (words per character) and show the top 3 rows

solution:



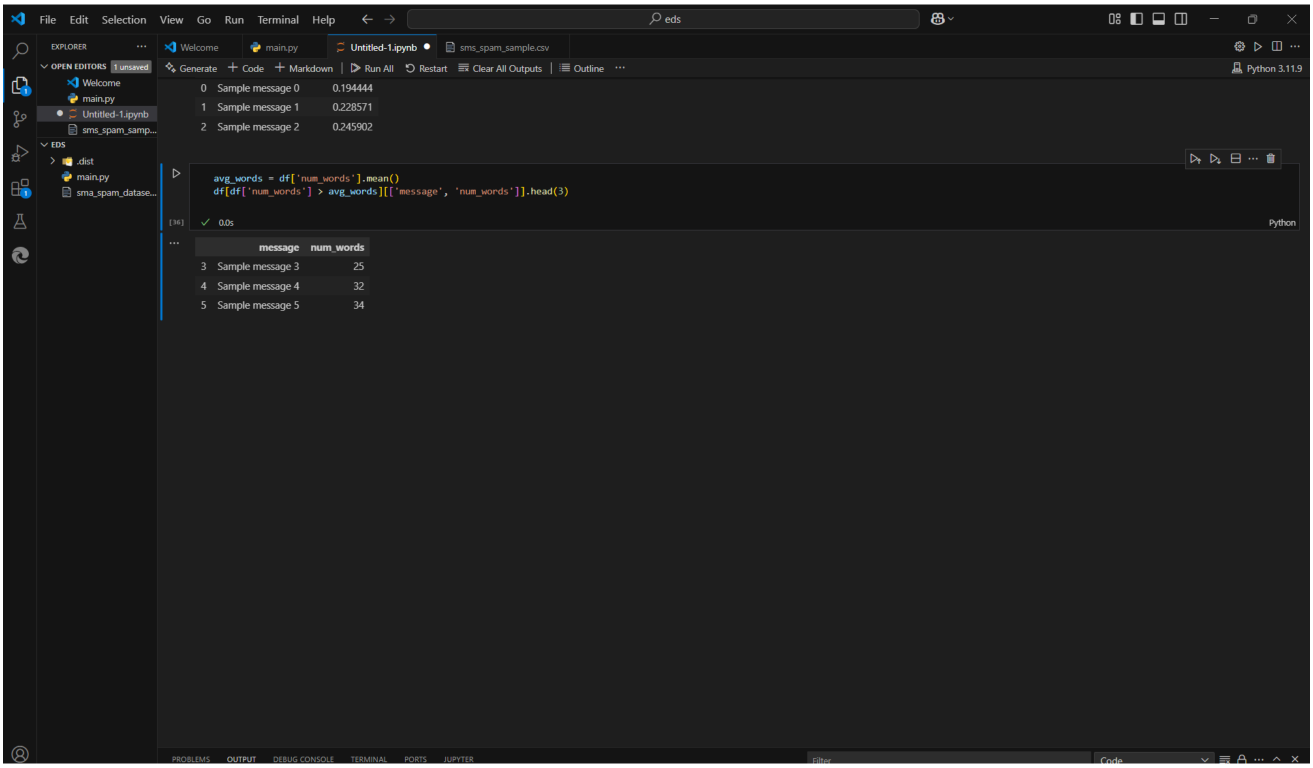
```
9    Sample message 9         13
14  Sample message 14         19
16  Sample message 16         19
18  Sample message 18         10
22  Sample message 22          8
```

```
df['word_density'] = df['num_words'] / df['length']
df[['message', 'word_density']].head(3)
```

```
            message  word_density
0  Sample message 0      0.194444
1  Sample message 1      0.228571
2  Sample message 2      0.245902
```

7. What are the messages with word counts above the average?

solution:



```
0  Sample message 0   0.194444
1  Sample message 1   0.228571
2  Sample message 2   0.245902
```

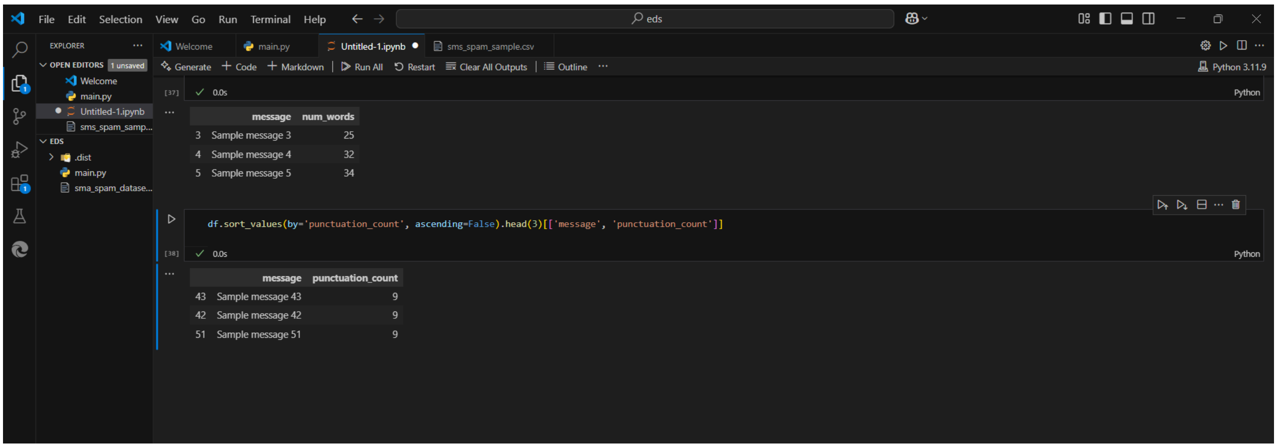
```python
avg_words = df['num_words'].mean()
df[df['num_words'] > avg_words][['message', 'num_words']].head(3)
```

```
[36]  ✓ 0.0s
```

```
            message  num_words
3  Sample message 3         25
4  Sample message 4         32
5  Sample message 5         34
```

8. Sort the messages by punctuation count and show the top 3

solution:



```
[37]  ✓ 0.0s
```

```
            message  num_words
3  Sample message 3         25
4  Sample message 4         32
5  Sample message 5         34
```

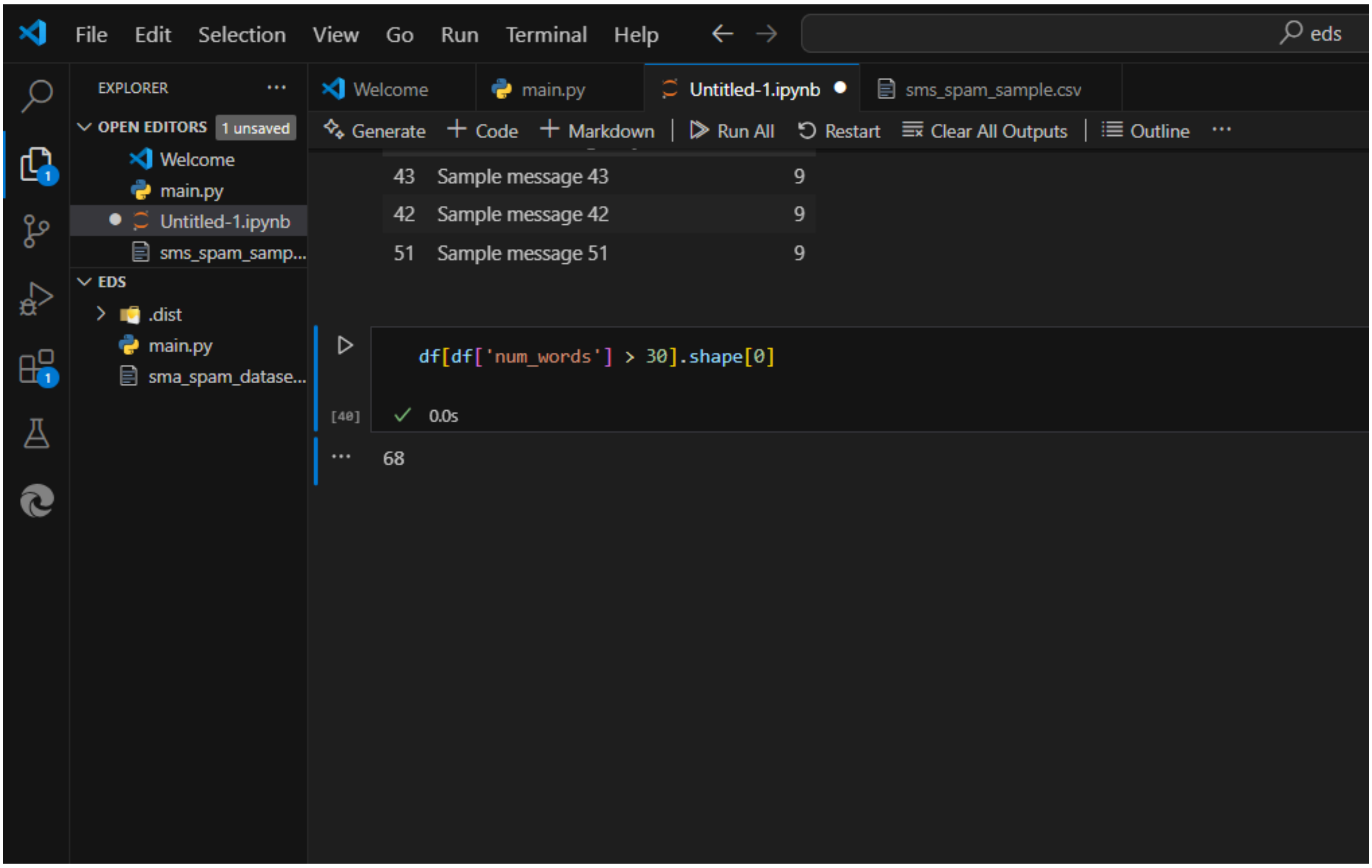
```python
df.sort_values(by='punctuation_count', ascending=False).head(3)[['message', 'punctuation_count']]
```

```
[38]  ✓ 0.0s
```

```
              message  punctuation_count
43  Sample message 43                  9
42  Sample message 42                  9
51  Sample message 51                  9
```

9. Count how many messages have more than 30 words

solution:



```
43  Sample message 43   9
42  Sample message 42   9
51  Sample message 51   9
```

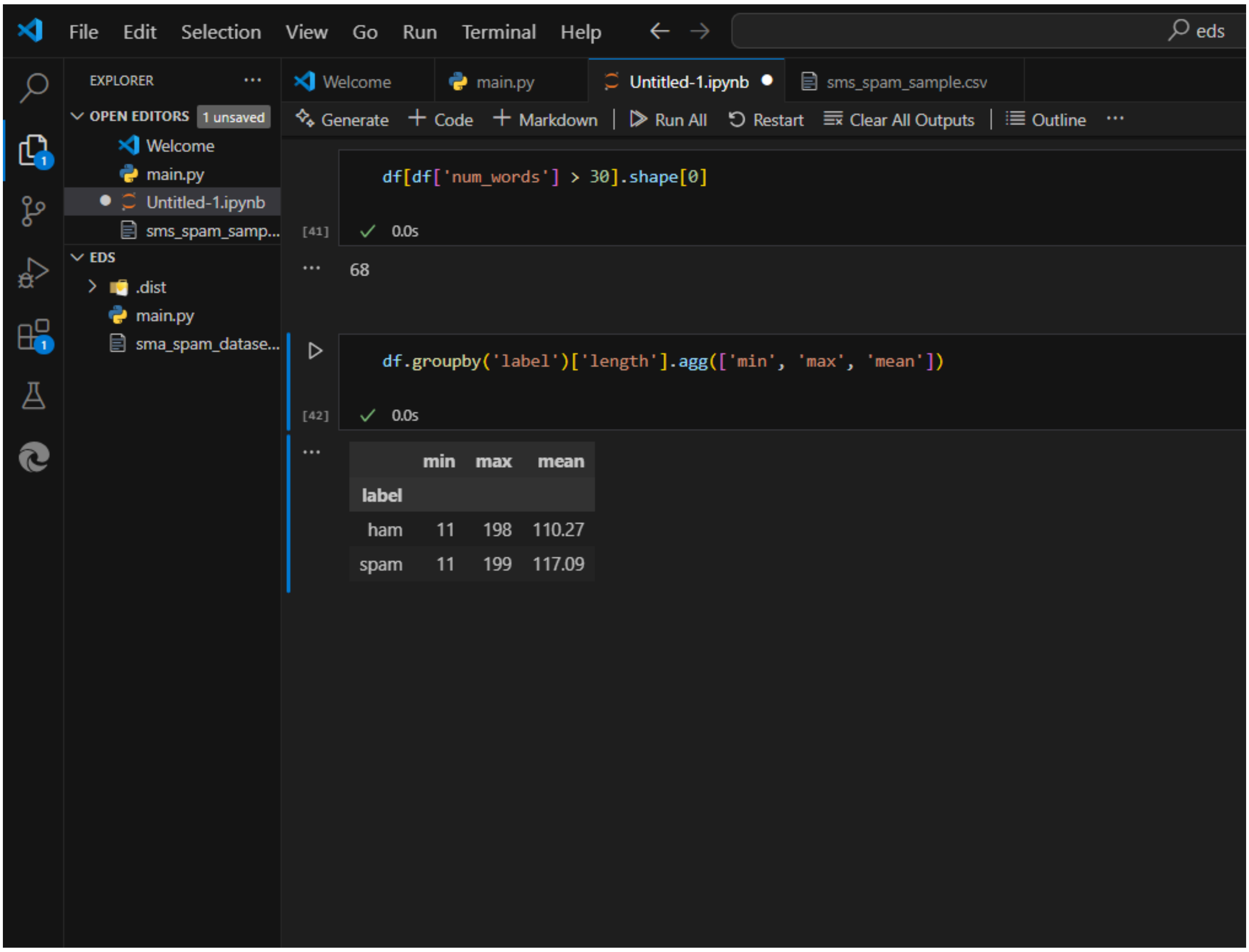
```
df[df['num_words'] > 30].shape[0]
```

[40] ✓ 0.0s

```
68
```

10. Group by label and get max, min, mean of length

solution:



```
df[df['num_words'] > 30].shape[0]
```

[41] ✓ 0.0s

```
68
```

```
df.groupby('label')['length'].agg(['min', 'max', 'mean'])
```

[42] ✓ 0.0s

	min	max	mean
label			
ham	11	198	110.27
spam	11	199	117.09

---end---