BIOS-584 Python Programming (Non-Bios Student)

Week 09

Instructor: Tianwen Ma, Ph.D.

Department of Biostatistics and Bioinformatics,
Rollins School of Public Health,
Emory University

Lecture Overview

- Pandas Tutorial 9 (week-09-pandas.ipynb)
- Debugging in PyCharm

Pandas

- A data manipulation package in Python for tabular data
    - In the form of rows and columns, known as DataFrames
- Functionality includes
    - Data transformation
    - Summary Statistics
    - Data merging
    - Integration with other data science packages, including NumPy, Matplotlib, Seaborn, Plotly, and Scikit-learn

What is Pandas used for?

- Import datasets from databases, spreadsheets, comma-separated values (CSV) files, etc.
- Clean datasets, i.e., handling missing values
- Tidy datasets by reshaping the structure into a suitable format prior to analysis.
- Aggregate data by calculating summary statistics.
- Visualize datasets and uncover hidden patterns.

Pandas Tutorial

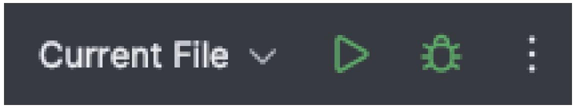
- I will go over the Jupyter notebook today.

Debugging in PyCharm

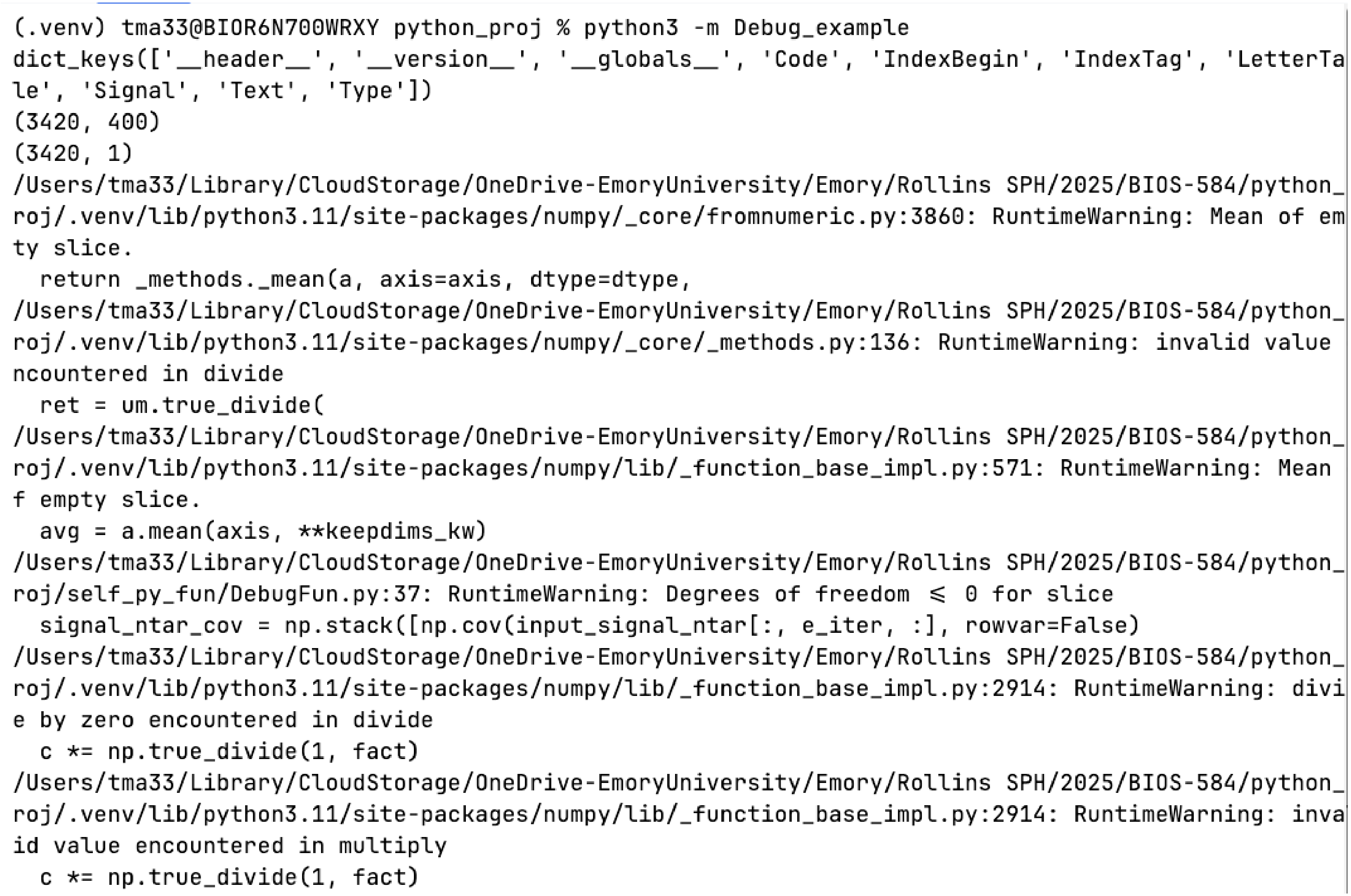
- https://www.jetbrains.com/help/pycharm/debugging-your-first-python-application.html#summary
- Find Debug_example.py
- Change the parent_dir to your own.

Overall Step

- Find unexpected output
- Use print() function
- Set checkpoint
- Start debugging mode



- Find the bug and resolve it
- Exit debugging mode and rerun the program



```
(.venv) tma33@BIOR6N700WRXY python_proj % python3 -m Debug_example
dict_keys(['__header__', '__version__', '__globals__', 'Code', 'IndexBegin', 'IndexTag', 'LetterTa
le', 'Signal', 'Text', 'Type'])
(3420, 400)
(3420, 1)
/Users/tma33/Library/CloudStorage/OneDrive-EmoryUniversity/Emory/Rollins SPH/2025/BIOS-584/python_
roj/.venv/lib/python3.11/site-packages/numpy/_core/fromnumeric.py:3860: RuntimeWarning: Mean of em
ty slice.
  return _methods._mean(a, axis=axis, dtype=dtype,
/Users/tma33/Library/CloudStorage/OneDrive-EmoryUniversity/Emory/Rollins SPH/2025/BIOS-584/python_
roj/.venv/lib/python3.11/site-packages/numpy/_core/_methods.py:136: RuntimeWarning: invalid value 
ncountered in divide
  ret = um.true_divide(
/Users/tma33/Library/CloudStorage/OneDrive-EmoryUniversity/Emory/Rollins SPH/2025/BIOS-584/python_
roj/.venv/lib/python3.11/site-packages/numpy/lib/_function_base_impl.py:571: RuntimeWarning: Mean 
f empty slice.
  avg = a.mean(axis, **keepdims_kw)
/Users/tma33/Library/CloudStorage/OneDrive-EmoryUniversity/Emory/Rollins SPH/2025/BIOS-584/python_
roj/self_py_fun/DebugFun.py:37: RuntimeWarning: Degrees of freedom <= 0 for slice
  signal_ntar_cov = np.stack([np.cov(input_signal_ntar[:, e_iter, :], rowvar=False)
/Users/tma33/Library/CloudStorage/OneDrive-EmoryUniversity/Emory/Rollins SPH/2025/BIOS-584/python_
roj/.venv/lib/python3.11/site-packages/numpy/lib/_function_base_impl.py:2914: RuntimeWarning: divi
e by zero encountered in divide
  c *= np.true_divide(1, fact)
/Users/tma33/Library/CloudStorage/OneDrive-EmoryUniversity/Emory/Rollins SPH/2025/BIOS-584/python_
roj/.venv/lib/python3.11/site-packages/numpy/lib/_function_base_impl.py:2914: RuntimeWarning: inva
id value encountered in multiply
  c *= np.true_divide(1, fact)
```

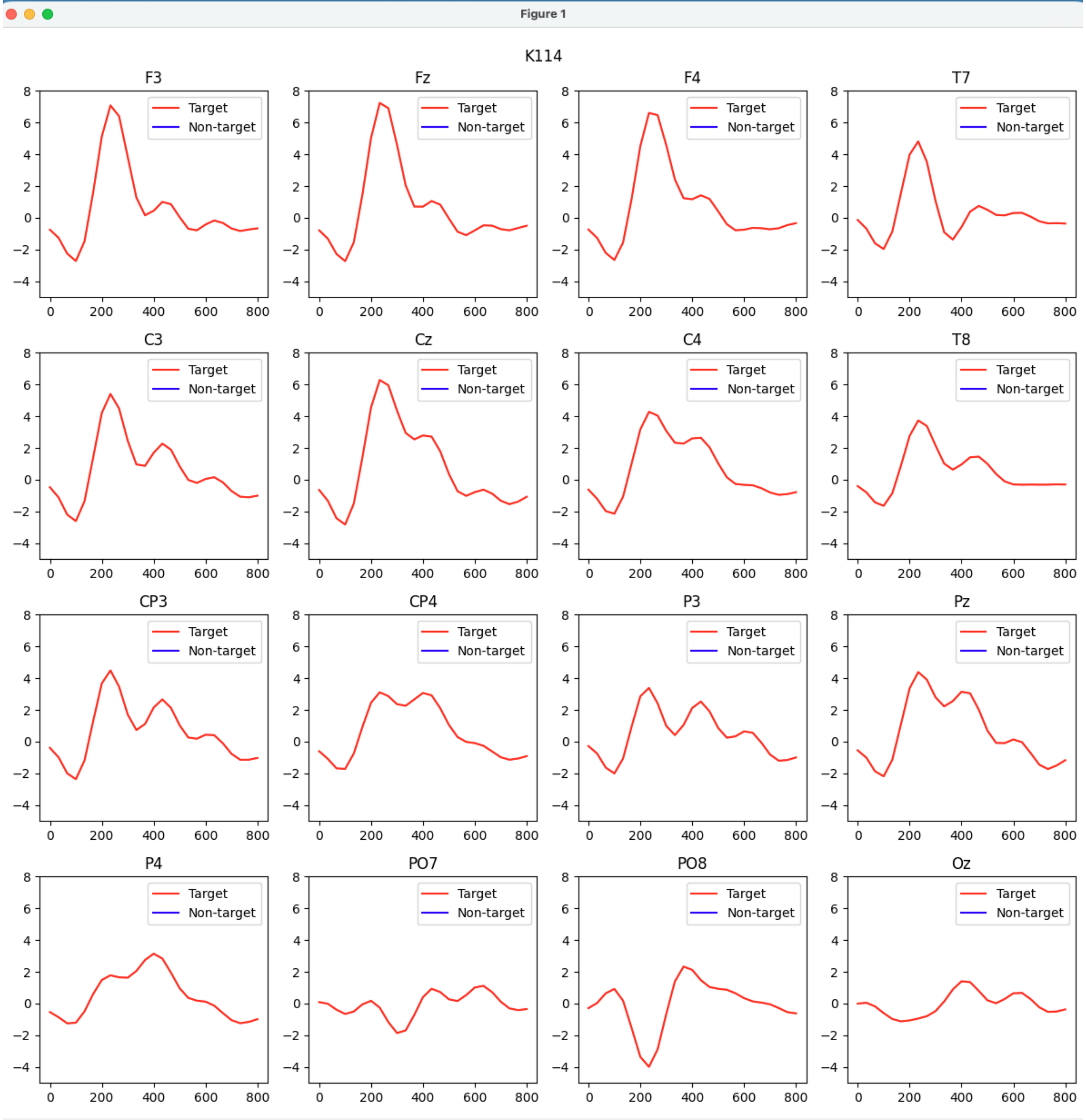
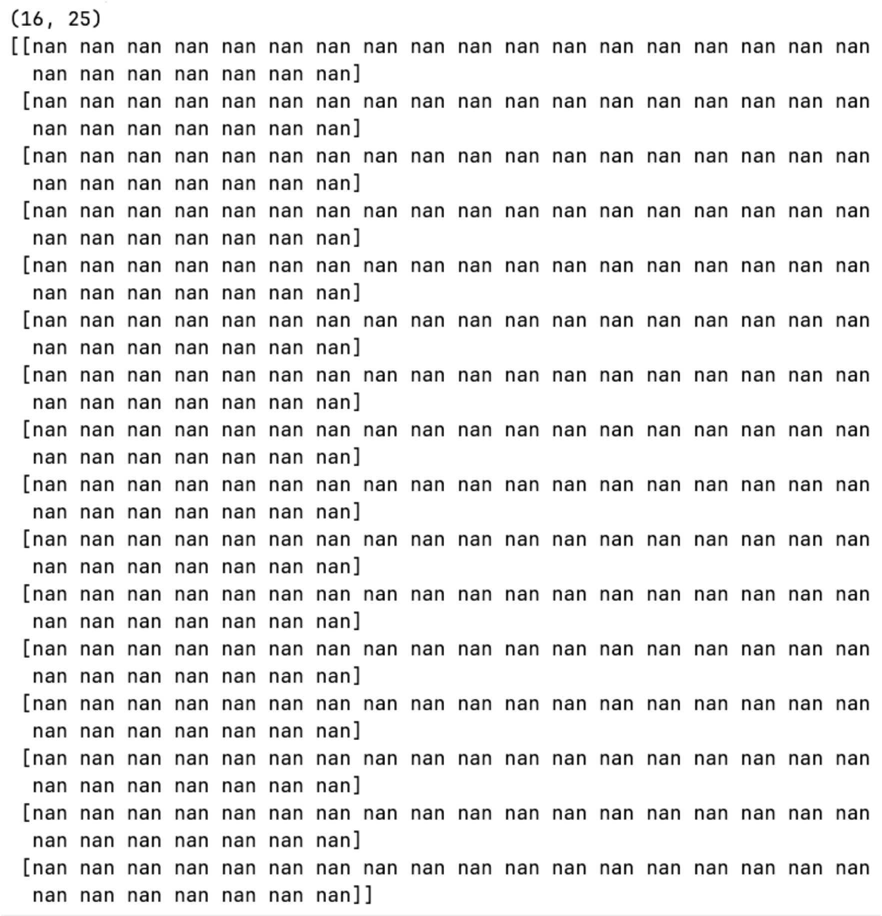


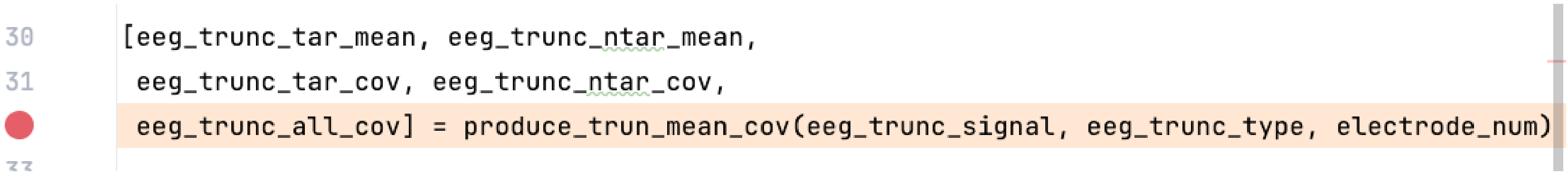
Figure 1
K114
F3
Fz
F4
T7
C3
Cz
C4
T8
CP3
CP4
P3
Pz
P4
PO7
PO8
Oz
Target
Non-target
8
6
4
2
0
−2
−4
0
200
400
600
800

Print output



```
(16, 25)
[[nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
  nan nan nan nan nan nan nan]
 [nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
  nan nan nan nan nan nan nan]
 [nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
  nan nan nan nan nan nan nan]
 [nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
  nan nan nan nan nan nan nan]
 [nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
  nan nan nan nan nan nan nan]
 [nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
  nan nan nan nan nan nan nan]
 [nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
  nan nan nan nan nan nan nan]
 [nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
  nan nan nan nan nan nan nan]
 [nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
  nan nan nan nan nan nan nan]
 [nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
  nan nan nan nan nan nan nan]
 [nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
  nan nan nan nan nan nan nan]
 [nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
  nan nan nan nan nan nan nan]
 [nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
  nan nan nan nan nan nan nan]
 [nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
  nan nan nan nan nan nan nan]
 [nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
  nan nan nan nan nan nan nan]
 [nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
  nan nan nan nan nan nan nan]]
```

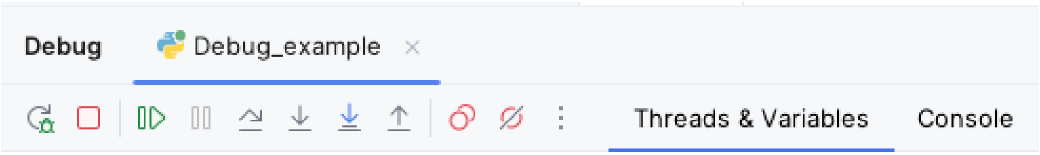
Set Checkpoint



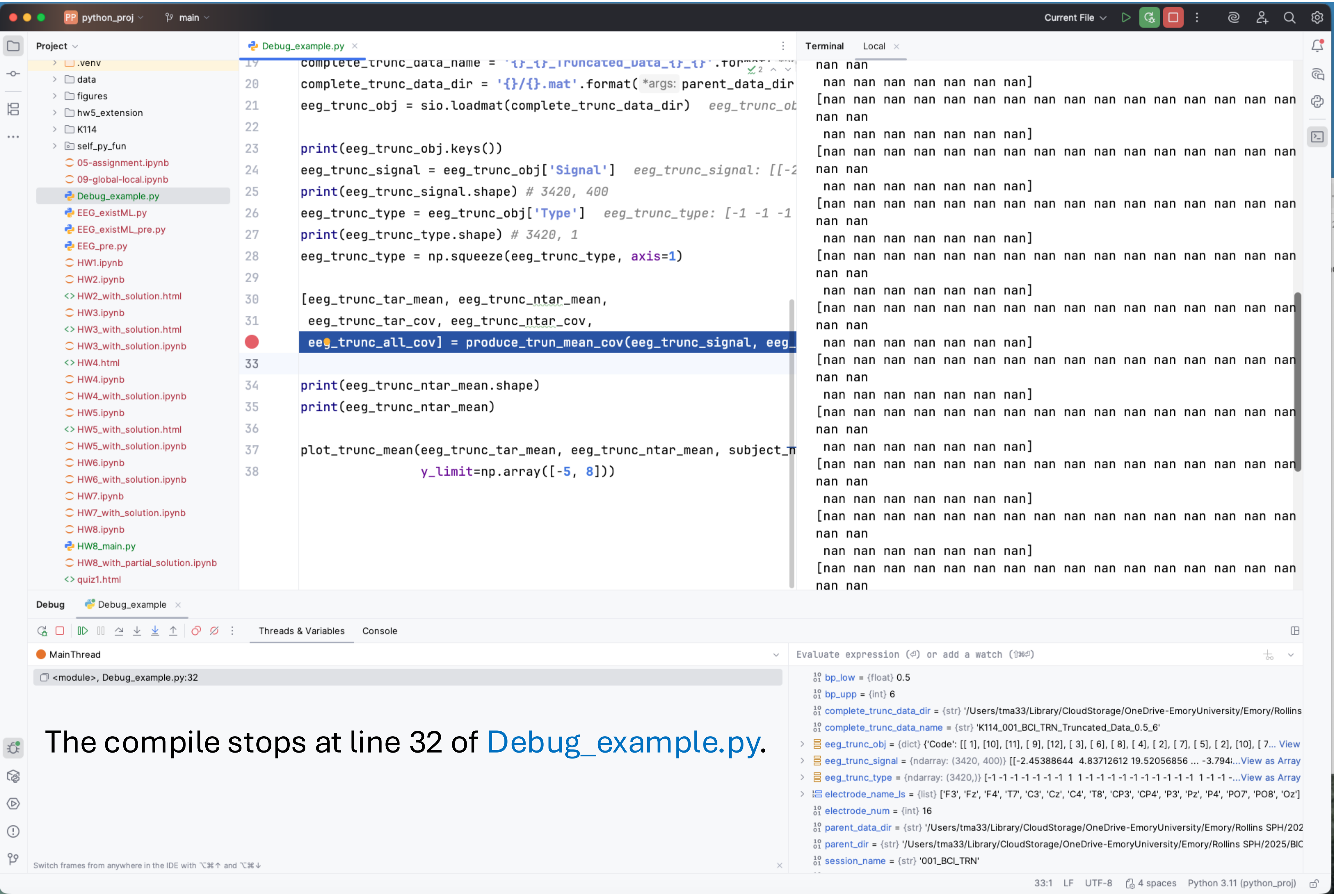
```
[eeg_trunc_tar_mean, eeg_trunc_ntar_mean,
 eeg_trunc_tar_cov, eeg_trunc_ntar_cov,
 eeg_trunc_all_cov] = produce_trun_mean_cov(eeg_trunc_signal, eeg_trunc_type, electrode_num)
```

Start Debugging Mode

- Two ways to start debugging mode
- Use the following icons to step into the function of question



- The values of variables will appear on the RHS of the interface.
- You can evaluate your own expression to confirm by click the vertical three dots ⋮



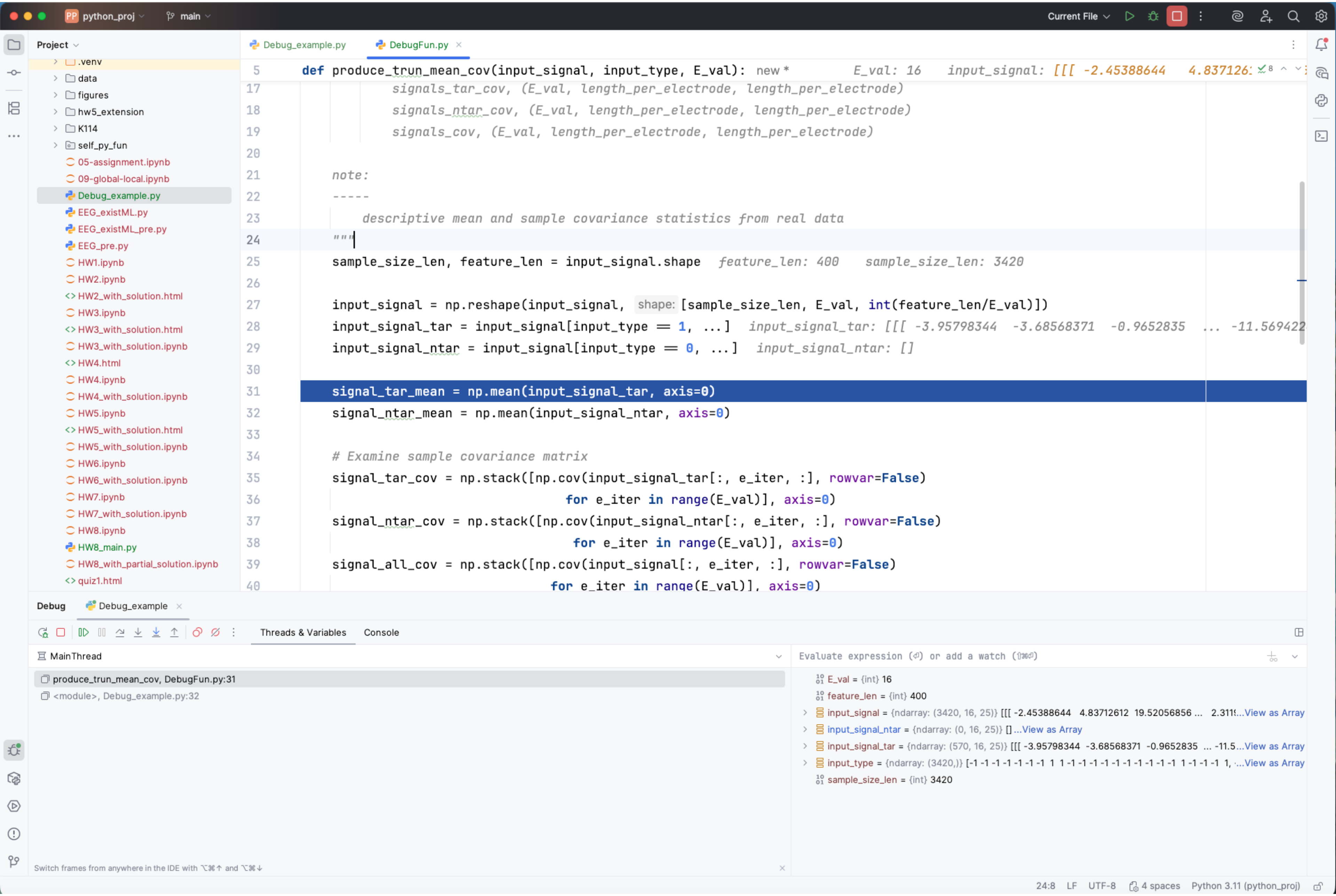
```
19   complete_trunc_data_name = '{}_{}_Truncated_Data_{}_{}'.format
20   complete_trunc_data_dir = '{}/{}.mat'.format( *args: parent_data_dir
21   eeg_trunc_obj = sio.loadmat(complete_trunc_data_dir)   eeg_trunc_ob
22
23   print(eeg_trunc_obj.keys())
24   eeg_trunc_signal = eeg_trunc_obj['Signal']   eeg_trunc_signal: [[-2
25   print(eeg_trunc_signal.shape) # 3420, 400
26   eeg_trunc_type = eeg_trunc_obj['Type']   eeg_trunc_type: [-1 -1 -1
27   print(eeg_trunc_type.shape) # 3420, 1
28   eeg_trunc_type = np.squeeze(eeg_trunc_type, axis=1)
29
30   [eeg_trunc_tar_mean, eeg_trunc_ntar_mean,
31    eeg_trunc_tar_cov, eeg_trunc_ntar_cov,
32    eeg_trunc_all_cov] = produce_trun_mean_cov(eeg_trunc_signal, eeg_
33
34   print(eeg_trunc_ntar_mean.shape)
35   print(eeg_trunc_ntar_mean)
36
37   plot_trunc_mean(eeg_trunc_tar_mean, eeg_trunc_ntar_mean, subject_n
38                   y_limit=np.array([-5, 8]))
```

```
nan nan
 nan nan nan nan nan nan nan]
[nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
nan nan
 nan nan nan nan nan nan nan]
[nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
nan nan
 nan nan nan nan nan nan nan]
[nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
nan nan
 nan nan nan nan nan nan nan]
[nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
nan nan
 nan nan nan nan nan nan nan]
[nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
nan nan
 nan nan nan nan nan nan nan]
[nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
nan nan
 nan nan nan nan nan nan nan]
[nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
nan nan
 nan nan nan nan nan nan nan]
[nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
nan nan
 nan nan nan nan nan nan nan]
[nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
nan nan
 nan nan nan nan nan nan nan]
[nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
nan nan
```

Evaluate expression (⏎) or add a watch (⇧⌘⏎)

```
bp_low = {float} 0.5
bp_upp = {int} 6
complete_trunc_data_dir = {str} '/Users/tma33/Library/CloudStorage/OneDrive-EmoryUniversity/Emory/Rollins
complete_trunc_data_name = {str} 'K114_001_BCI_TRN_Truncated_Data_0.5_6'
> eeg_trunc_obj = {dict} {'Code': [[ 1], [10], [11], [ 9], [12], [ 3], [ 6], [ 8], [ 4], [ 2], [ 7], [ 5], [ 2], [10], [ 7... View
> eeg_trunc_signal = {ndarray: (3420, 400)} [[-2.45388644  4.83712612 19.52056856 ... -3.794...View as Array
> eeg_trunc_type = {ndarray: (3420,)} [-1 -1 -1 -1 -1 -1 -1  1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1 -1 -1 -...View as Array
> electrode_name_ls = {list} ['F3', 'Fz', 'F4', 'T7', 'C3', 'Cz', 'C4', 'T8', 'CP3', 'CP4', 'P3', 'Pz', 'P4', 'PO7', 'PO8', 'Oz']
electrode_num = {int} 16
parent_data_dir = {str} '/Users/tma33/Library/CloudStorage/OneDrive-EmoryUniversity/Emory/Rollins SPH/202
parent_dir = {str} '/Users/tma33/Library/CloudStorage/OneDrive-EmoryUniversity/Emory/Rollins SPH/2025/BIC
session_name = {str} '001_BCI_TRN'
```

The compile stops at line 32 of Debug_example.py.



```
def produce_trun_mean_cov(input_signal, input_type, E_val):  new *      E_val: 16    input_signal: [[[ -2.45388644    4.837126:
            signals_tar_cov, (E_val, length_per_electrode, length_per_electrode)
            signals_ntar_cov, (E_val, length_per_electrode, length_per_electrode)
            signals_cov, (E_val, length_per_electrode, length_per_electrode)

    note:
    -----
        descriptive mean and sample covariance statistics from real data
    """
    sample_size_len, feature_len = input_signal.shape   feature_len: 400    sample_size_len: 3420

    input_signal = np.reshape(input_signal,  shape: [sample_size_len, E_val, int(feature_len/E_val)])
    input_signal_tar = input_signal[input_type == 1, ...]  input_signal_tar: [[[ -3.95798344  -3.68568371  -0.9652835   ... -11.569422
    input_signal_ntar = input_signal[input_type == 0, ...]  input_signal_ntar: []

    signal_tar_mean = np.mean(input_signal_tar, axis=0)
    signal_ntar_mean = np.mean(input_signal_ntar, axis=0)

    # Examine sample covariance matrix
    signal_tar_cov = np.stack([np.cov(input_signal_tar[:, e_iter, :], rowvar=False)
                               for e_iter in range(E_val)], axis=0)
    signal_ntar_cov = np.stack([np.cov(input_signal_ntar[:, e_iter, :], rowvar=False)
                                for e_iter in range(E_val)], axis=0)
    signal_all_cov = np.stack([np.cov(input_signal[:, e_iter, :], rowvar=False)
                              for e_iter in range(E_val)], axis=0)
```

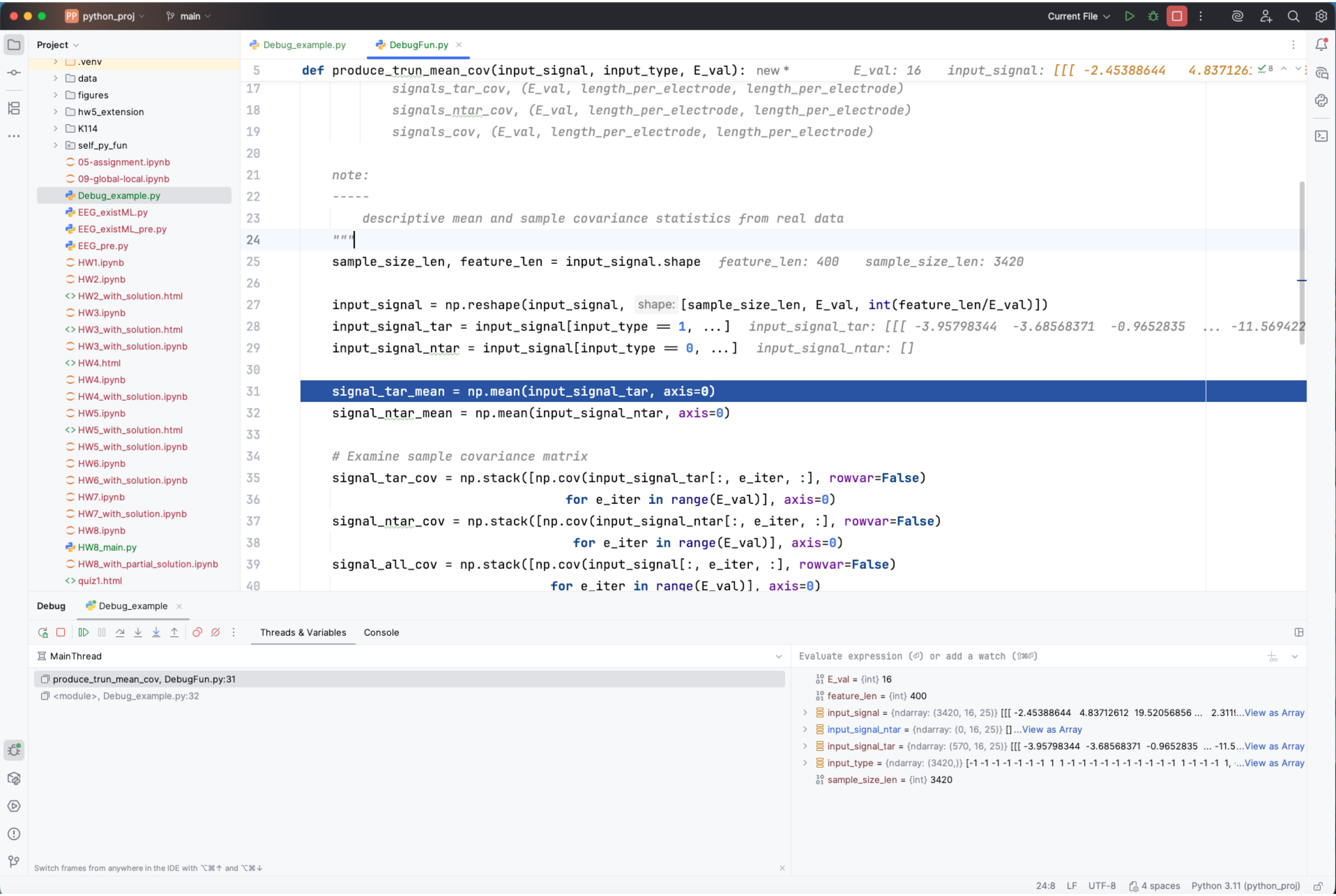
```
produce_trun_mean_cov, DebugFun.py:31
<module>, Debug_example.py:32
```

```
E_val = {int} 16
feature_len = {int} 400
input_signal = {ndarray: (3420, 16, 25)} [[[ -2.45388644   4.83712612  19.52056856 ...   2.311...View as Array
input_signal_ntar = {ndarray: (0, 16, 25)} []...View as Array
input_signal_tar = {ndarray: (570, 16, 25)} [[[ -3.95798344  -3.68568371  -0.9652835  ... -11.5...View as Array
input_type = {ndarray: (3420,)} [-1 -1 -1 -1 -1 -1 -1  1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1 -1 -1 -1  1, ...View as Array
sample_size_len = {int} 3420
```

Switch frames from anywhere in the IDE with ⌥⌘↑ and ⌥⌘↓

24:8   LF   UTF-8   4 spaces   Python 3.11 (python_proj)

After stepping into the function, we are in DebugFun.py. You can review your output line by line or jump to a certain line (another checkpoint is required).



```
def produce_trun_mean_cov(input_signal, input_type, E_val):  new *     E_val: 16    input_signal: [[[ -2.45388644    4.837126
            signals_tar_cov, (E_val, length_per_electrode, length_per_electrode)
            signals_ntar_cov, (E_val, length_per_electrode, length_per_electrode)
            signals_cov, (E_val, length_per_electrode, length_per_electrode)

    note:
    -----
        descriptive mean and sample covariance statistics from real data
    """
    sample_size_len, feature_len = input_signal.shape   feature_len: 400    sample_size_len: 3420

    input_signal = np.reshape(input_signal,  shape: [sample_size_len, E_val, int(feature_len/E_val)])
    input_signal_tar = input_signal[input_type == 1, ...]   input_signal_tar: [[[ -3.95798344  -3.68568371  -0.9652835   ... -11.569422
    input_signal_ntar = input_signal[input_type == 0, ...]   input_signal_ntar: []

    signal_tar_mean = np.mean(input_signal_tar, axis=0)
    signal_ntar_mean = np.mean(input_signal_ntar, axis=0)

    # Examine sample covariance matrix
    signal_tar_cov = np.stack([np.cov(input_signal_tar[:, e_iter, :], rowvar=False)
                               for e_iter in range(E_val)], axis=0)
    signal_ntar_cov = np.stack([np.cov(input_signal_ntar[:, e_iter, :], rowvar=False)
                                for e_iter in range(E_val)], axis=0)
    signal_all_cov = np.stack([np.cov(input_signal[:, e_iter, :], rowvar=False)
                              for e_iter in range(E_val)], axis=0)
```

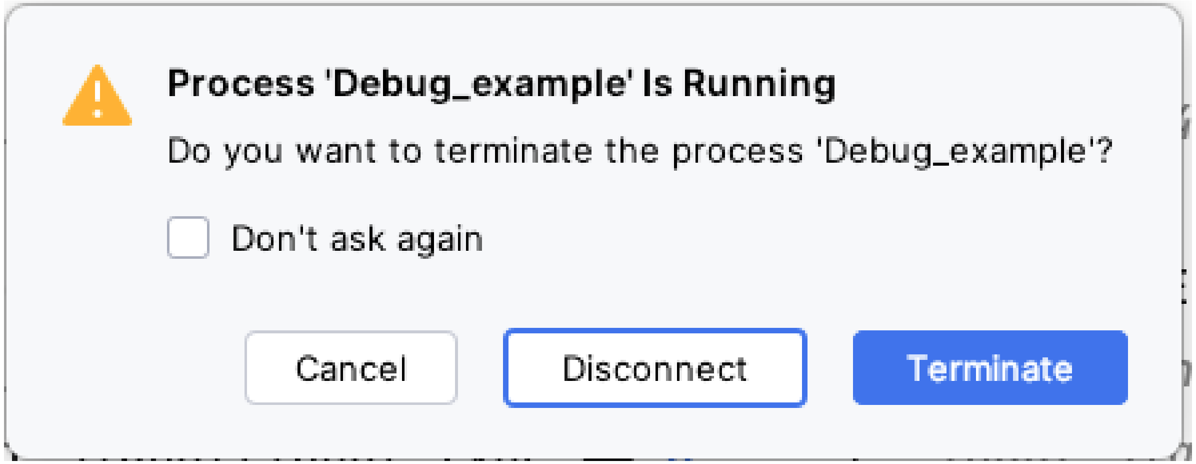
```
produce_trun_mean_cov, DebugFun.py:31
<module>, Debug_example.py:32
```

```
E_val = {int} 16
feature_len = {int} 400
input_signal = {ndarray: (3420, 16, 25)} [[[ -2.45388644   4.83712612  19.52056856 ...   2.311...View as Array
input_signal_ntar = {ndarray: (0, 16, 25)} []...View as Array
input_signal_tar = {ndarray: (570, 16, 25)} [[[ -3.95798344  -3.68568371  -0.9652835  ... -11.5...View as Array
input_type = {ndarray: (3420,)} [-1 -1 -1 -1 -1 -1 -1  1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1 -1 -1 -1  1, ...View as Array
sample_size_len = {int} 3420
```

Given the current output, do you know where the bug is and how to resolve it?

Remove Checkpoint

- Close Debug_example.py and confirm to terminate the debugging process



Process 'Debug_example' Is Running

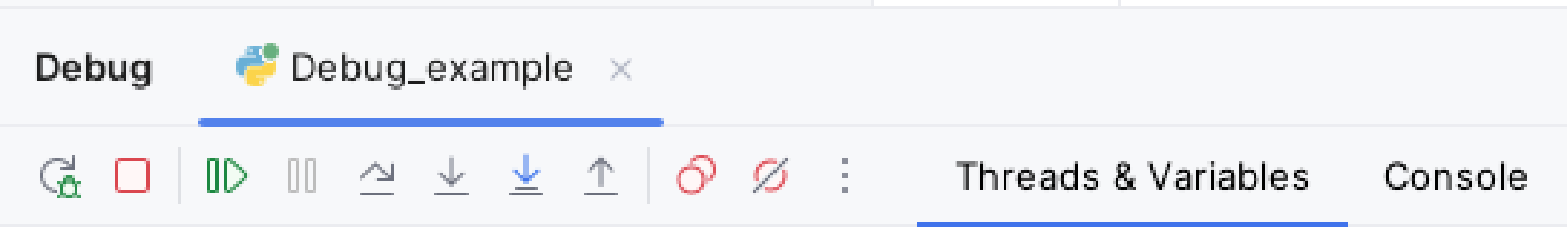
Do you want to terminate the process 'Debug_example'?

□ Don't ask again

- Remove checkpoint
- Rerun the program to confirm

Other Useful Icons

- The left-most one: Re-enter debug mode
- The 2nd on the left: Stop


Debug
Debug_example
Threads & Variables
Console