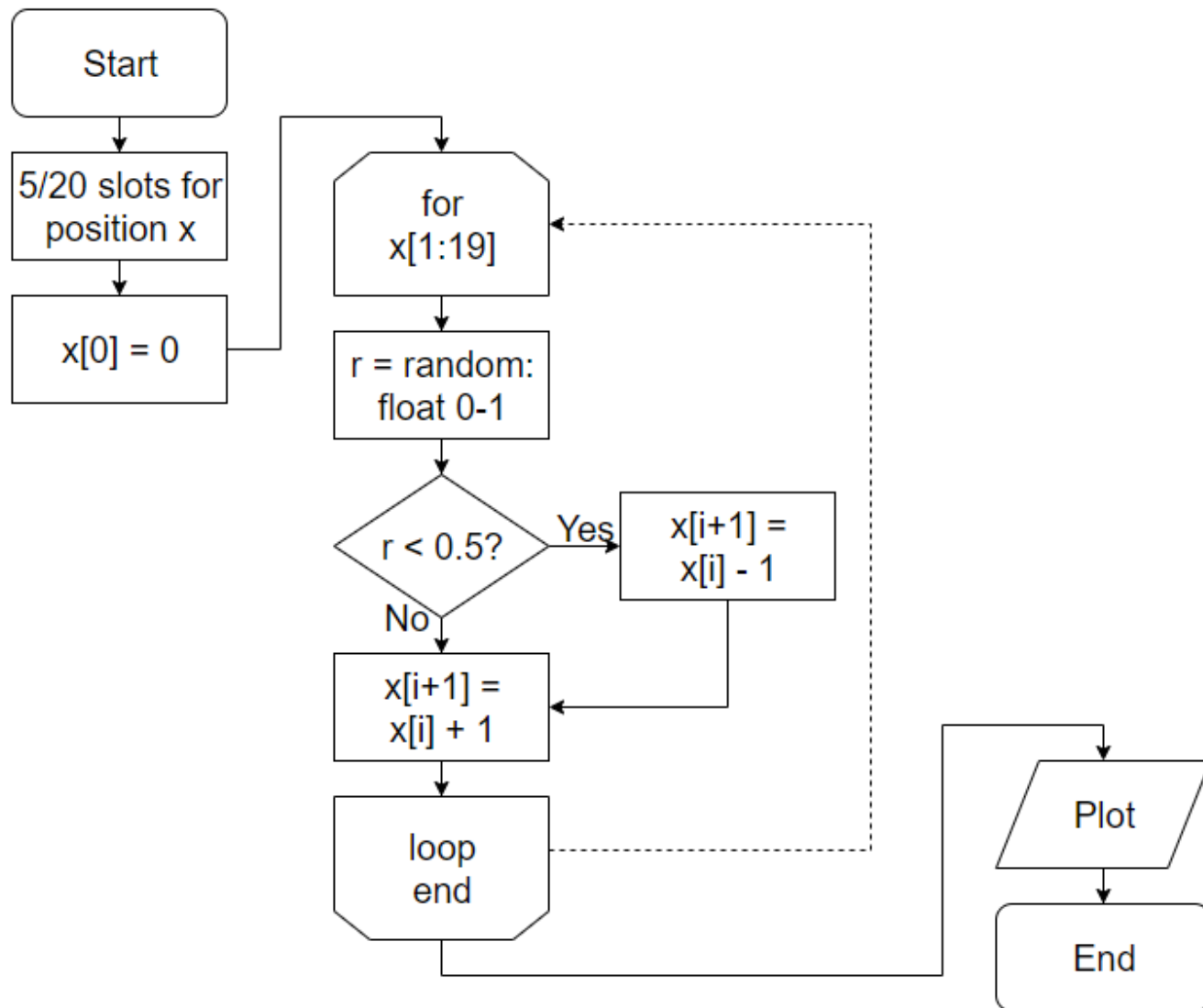


Coding seminar

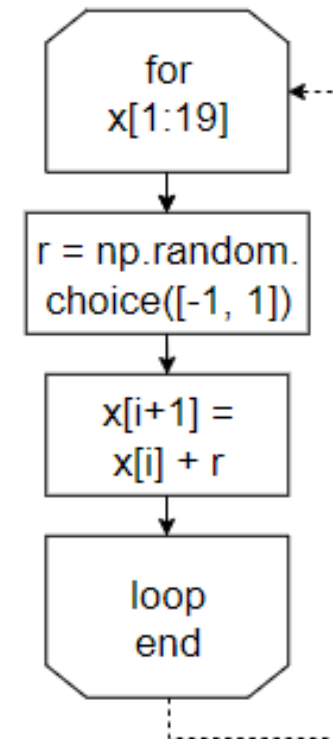
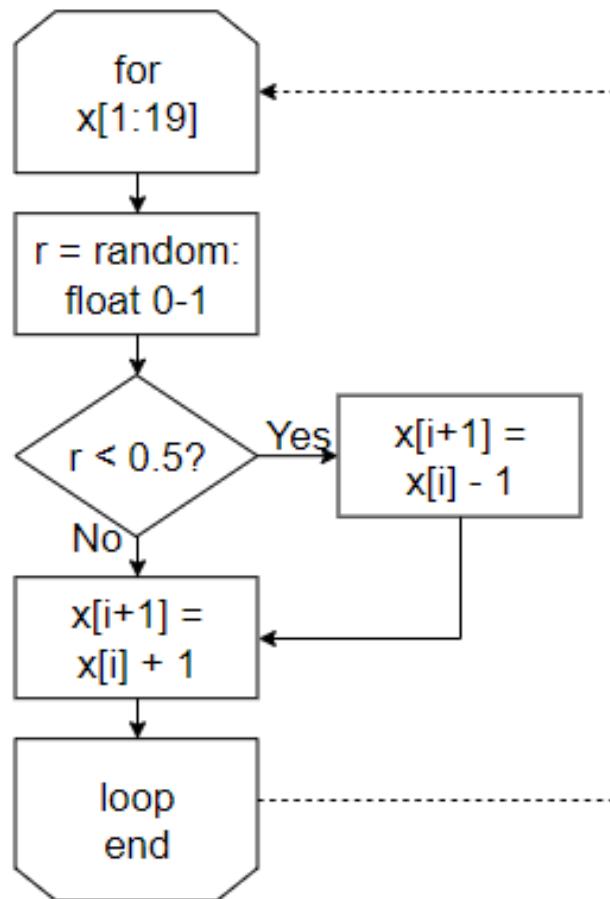
Lesson 3: Handling real-world data

Ikue Hirata, PhD

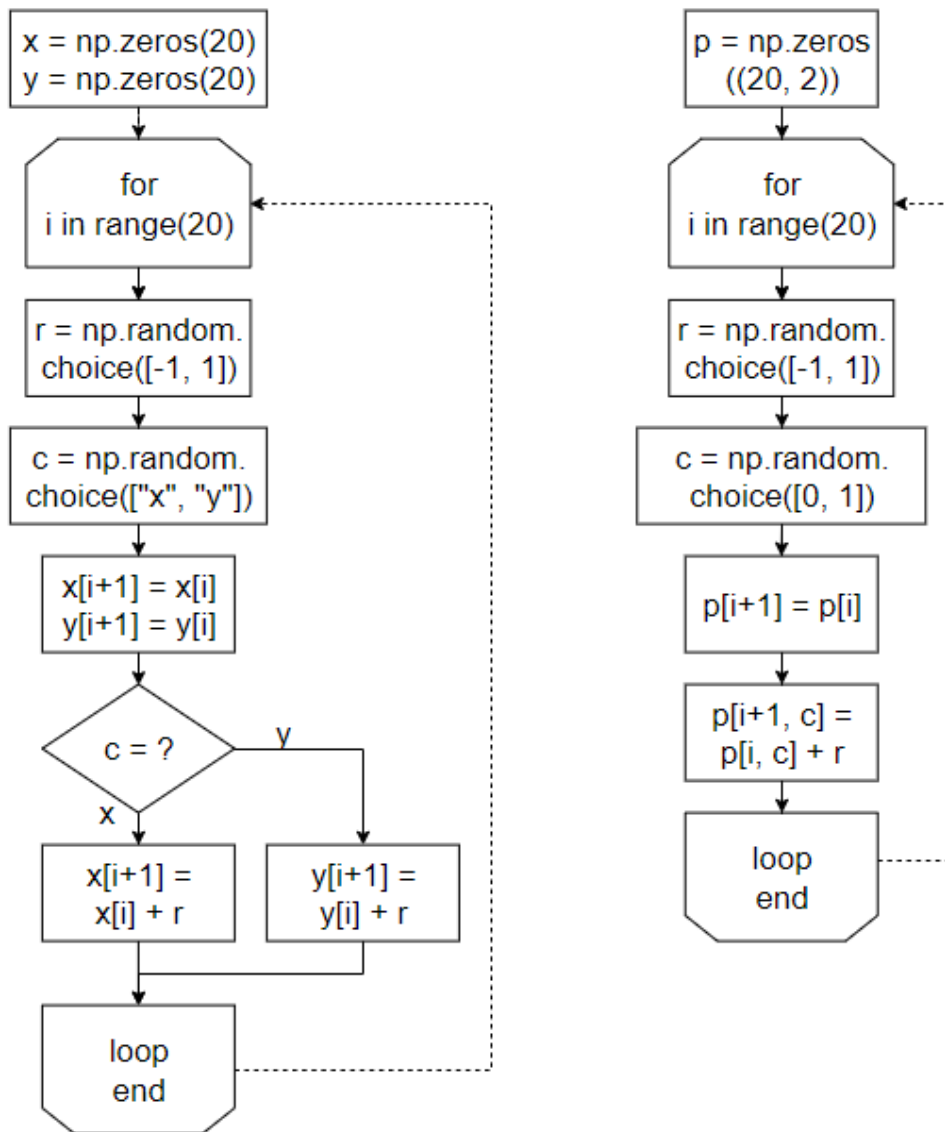
Exercise model answer: random walk



Make it simple



2-D random walk



Contents

Modules: More about Numpy


















Plotting: More about Matplotlib

Editors

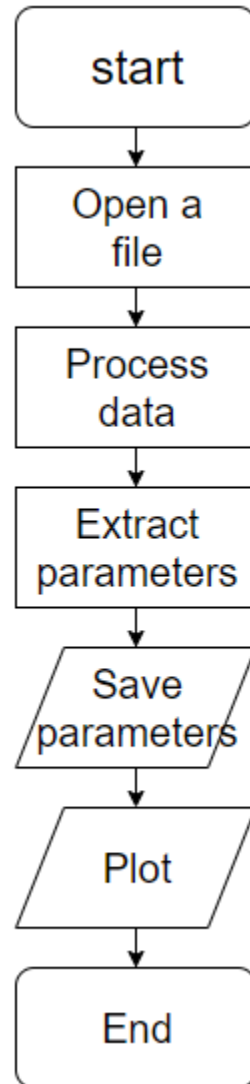
Debugger/debugging

UNIX-based commands for mass-process

Extensions ready?

 firstplot.png	2020/01/15 15:08
 Lesson1.ipynb	2020/01/13 16:46
 Lesson1_Exercise0_modelanswer.ipynb	2020/01/13 17:23
 Lesson1_Exercise1.ipynb	2020/01/08 16:15
 Lesson1_Exercise1_modelanswer.ipynb	2020/01/13 17:29
 Lesson1_Exercise2.ipynb	2020/01/08 15:03
 Lesson1_Exercise2_modelanswer.ipynb	2020/01/15 14:59
 Lesson1_ppt.pdf	2020/01/13 16:44
 Lesson2.ipynb	2020/01/20 0:10
 Lesson2_Exercise1.ipynb	2020/01/01 21:49
 Lesson2_Exercise1_modelanswer.ipynb	2020/01/13 11:09
 Lesson2_Exercise2.ipynb	2020/01/02 12:48
 Lesson2_Exercise2_modelanswer.ipynb	2020/01/13 12:57
 Lesson2_ppt.pdf	2020/01/20 0:13
 Lesson3.ipynb	2020/01/20 0:07
 LICENSE	2019/12/23 14:58
 README.md	2020/01/15 15:18

Data analysis process



Download demo files

Branch: master ▾

New pull request

Create new file

Upload files

Find file

Clone or download ▾

ikuehirata typo corrected

Latest commit 2781394 3 days ago

Lesson3-data	file moved	3 days ago
exercises_and_model_answers	file moved	3 days ago
LICENSE	Initial commit	last month
Lesson1.ipynb	typo corrected	14 days ago
Lesson1_ppt.pdf	typo corrected	14 days ago
Lesson2.ipynb	exercise 2-0 update	7 days ago
Lesson2_ppt.pdf	import update	7 days ago
Lesson3.ipynb	typo corrected	3 days ago
Lesson3_ppt.pdf	ppt added	5 days ago
README.md	demo file update	6 days ago
data0.csv	demo file update	6 days ago

README.md

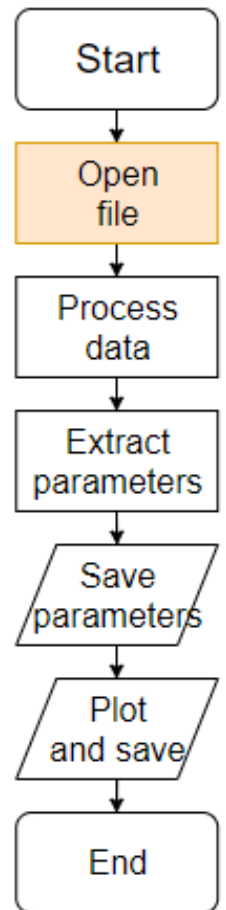
Sample data

```
Programmer's Notepad - [data1.csv]
File Edit Search View Tools Window Help
data1.csv
# Measurement mode = ZTD
# Osc Level = +1.00000E-01
# DC Bias State = 0
# DC Bias Level = +0.00000E+00
# Averaging = MED +1
# Sweep Mode = SEQ
# Open Correction = 0
# Frequency Z phi
2.0000000000000000e+01 1.0074800000000000e+06 -8.5748000000000000e+01 0.0000000000000000e+00 0.0000000000000000e+00
2.0000000000000000e+01 1.0200500000000000e+06 -8.7277799999999999e+01 0.0000000000000000e+00 0.0000000000000000e+00
2.0000000000000000e+01 1.0227400000000000e+06 -8.9246799999999999e+01 0.0000000000000000e+00 0.0000000000000000e+00
2.0000000000000000e+01 1.0243300000000000e+06 -8.9246300000000000e+01 0.0000000000000000e+00 0.0000000000000000e+00
2.0000000000000000e+01 1.0519300000000000e+06 -8.7950999999999999e+01 0.0000000000000000e+00 0.0000000000000000e+00
2.0000000000000000e+01 1.0834900000000000e+06 -8.7283199999999999e+01 0.0000000000000000e+00 0.0000000000000000e+00
2.0000000000000000e+01 1.0720800000000000e+06 -8.7074399999999999e+01 0.0000000000000000e+00 0.0000000000000000e+00
2.0989399999999999e+01 1.0212600000000000e+06 -8.5559899999999999e+01 0.0000000000000000e+00 0.0000000000000000e+00
2.3334599999999999e+01 9.0411400000000000e+05 -8.8224800000000000e+01 0.0000000000000000e+00 0.0000000000000000e+00
2.5941800000000000e+01 7.9983300000000000e+05 -8.7057500000000000e+01 0.0000000000000000e+00 0.0000000000000000e+00
2.8840299999999999e+01 7.2794700000000000e+05 -8.7242900000000000e+01 0.0000000000000000e+00 0.0000000000000000e+00
3.2062699999999999e+01 6.7304800000000000e+05 0.0000000000000000e+00 0.0000000000000000e+00 0.0000000000000000e+00
3.5645099999999999e+01 5.5381000000000000e+05 0.0000000000000000e+00 0.0000000000000000e+00 0.0000000000000000e+00
3.9627800000000000e+01 5.3934700000000000e+05 0.0000000000000000e+00 0.0000000000000000e+00 0.0000000000000000e+00
4.4055500000000000e+01 5.0011700000000000e+05 0.0000000000000000e+00 0.0000000000000000e+00 0.0000000000000000e+00
4.8977899999999999e+01 4.0046700000000000e+05 -8.6866799999999999e+01 0.0000000000000000e+00 0.0000000000000000e+00
5.4450299999999999e+01 3.7458900000000000e+05 -8.6524500000000000e+01 0.0000000000000000e+00 0.0000000000000000e+00
6.0534100000000000e+01 3.5306700000000000e+05 -8.7066699999999999e+01 0.0000000000000000e+00 0.0000000000000000e+00
6.7297700000000000e+01 3.2024800000000000e+05 -8.8080100000000000e+01 0.0000000000000000e+00 0.0000000000000000e+00
7.4816999999999999e+01 2.8903000000000000e+05 -8.7074200000000000e+01 0.0000000000000000e+00 0.0000000000000000e+00
8.3176400000000000e+01 2.6151800000000000e+05 -8.7703599999999999e+01 0.0000000000000000e+00 0.0000000000000000e+00
9.2469800000000000e+01 2.3105100000000000e+05 -8.5886700000000000e+01 0.0000000000000000e+00 0.0000000000000000e+00
1.0280200000000000e+02 2.1358500000000000e+05 -8.5965699999999999e+01 0.0000000000000000e+00 0.0000000000000000e+00
1.1428799999999999e+02 1.8976300000000000e+05 -8.6715100000000000e+01 0.0000000000000000e+00 0.0000000000000000e+00
1.2705700000000000e+02 1.7150800000000000e+05 -8.6708400000000000e+01 0.0000000000000000e+00 0.0000000000000000e+00
```

Array data – csv/tsv

comma/tab separated values

	A	B	C	D	E	F
1	# Measurement mode = ZTD					
2	# Osc Level = +1.00000E-01					
3	# DC Bias State = 0					
4	# DC Bias Level = +0.00000E+00					
5	# Averaging = MED	1				
6	# Sweep Mode = SEQ					
7	# Open Correction = 0					
8	# Frequency	Z	phi			
9	2.00E+01	1.01E+06	-8.57E+01	0.00E+00	0.00E+00	
10	2.00E+01	1.02E+06	-8.73E+01	0.00E+00	0.00E+00	
11	2.00E+01	1.02E+06	-8.92E+01	0.00E+00	0.00E+00	
12	2.00E+01	1.02E+06	-8.92E+01	0.00E+00	0.00E+00	
13	2.00E+01	1.05E+06	-8.80E+01	0.00E+00	0.00E+00	
14	2.00E+01	1.08E+06	-8.73E+01	0.00E+00	0.00E+00	
15	2.00E+01	1.07E+06	-8.71E+01	0.00E+00	0.00E+00	
16	2.10E+01	1.02E+06	-8.56E+01	0.00E+00	0.00E+00	
17	2.33E+01	9.04E+05	-8.82E+01	0.00E+00	0.00E+00	
18	2.59E+01	8.00E+05	-8.71E+01	0.00E+00	0.00E+00	
19	2.88E+01	7.28E+05	-8.72E+01	0.00E+00	0.00E+00	
20	3.21E+01	6.73E+05	-9.05E+01	0.00E+00	0.00E+00	
21	3.56E+01	5.54E+05	-8.54E+01	0.00E+00	0.00E+00	
22	3.96E+01	5.39E+05	-8.75E+01	0.00E+00	0.00E+00	
23	4.41E+01	5.00E+05	-9.02E+01	0.00E+00	0.00E+00	
24	4.90E+01	4.00E+05	-8.69E+01	0.00E+00	0.00E+00	
25	5.45E+01	3.75E+05	-8.65E+01	0.00E+00	0.00E+00	
26	6.05E+01	3.53E+05	-8.71E+01	0.00E+00	0.00E+00	



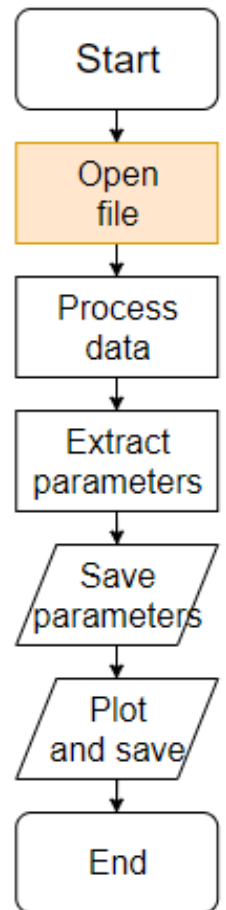
Read data into array

File Edit View Insert Cell Kernel Navigate Widgets Help

In [14]:

```
1 import numpy as np
2 data = np.loadtxt("data0.csv", delimiter='\t')
3 print(data)
```

```
[[ 2.000000e+01  9.04909e+05 -8.85148e+01
  0.000000e+00  0.000000e+00]
 [ 2.000000e+01  9.54938e+05 -8.75136e+01
  0.000000e+00  0.000000e+00]
 [ 2.000000e+01  9.34201e+05 -8.60863e+01
  0.000000e+00  0.000000e+00]
 [ 2.000000e+01  9.25598e+05 -8.47966e+01
  0.000000e+00  0.000000e+00]
 [ 2.000000e+01  9.23228e+05 -8.54492e+01
  0.000000e+00  0.000000e+00]]
```



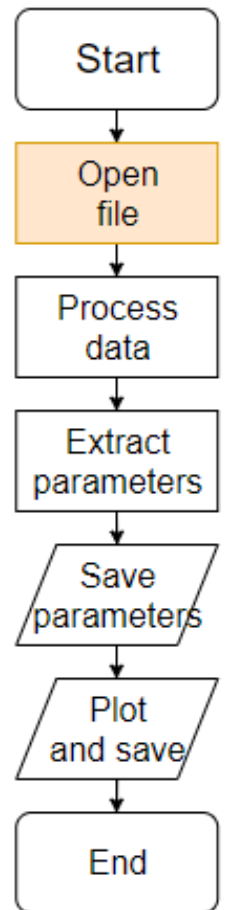
Read data into array

File Edit View Insert Cell Kernel Navigate Widgets Help

In [2]:

```
1 data = np.loadtxt("data1.csv", delimiter="\t",  
2 usecols=[0,1,2])  
3 print(data)
```

```
[[ 2.000000e+01  1.00748e+06 -8.57480e+01]  
 [ 2.000000e+01  1.02005e+06 -8.72778e+01]  
 [ 2.000000e+01  1.02274e+06 -8.92468e+01]  
 [ 2.000000e+01  1.02433e+06 -8.92463e+01]  
 [ 2.000000e+01  1.05193e+06 -8.79510e+01]  
 [ 2.000000e+01  1.08349e+06 -8.72832e+01]  
 [ 2.000000e+01  1.07208e+06 -8.70744e+01]  
 [ 2.09894e+01  1.02126e+06 -8.55599e+01]  
 [ 2.33346e+01  9.04114e+05 -8.82248e+01]  
 [ 2.59418e+01  7.99833e+05 -8.70575e+01]  
 [ 2.88403e+01  7.27047e+05 -8.72420e+01]
```



For more options...

 SciPy.org 

SciPy.org Docs NumPy v1.17 Manual NumPy Reference Routines Array creation routines

index next previous

numpy.loadtxt

`numpy.loadtxt(fname, dtype=<class 'float'>, comments='#', delimiter=None, converters=None, skiprows=0, usecols=None, unpack=False, ndmin=0, encoding='bytes', max_rows=None)` [\[source\]](#)

Load data from a text file.

Each row in the text file must have the same number of values.

Parameters: `fname` : file, str, or pathlib.Path

File, filename, or generator to read. If the filename extension is `.gz` or `.bz2`, the file is first decompressed. Note that generators should return byte strings for Python 3k.

`dtype` : data-type, optional

Data-type of the resulting array; default: float. If this is a structured data-type, the resulting array will be 1-dimensional, and each row will be interpreted as an element of the array. In this case, the number of columns used must match the number of fields in the data-type.

`comments` : str or sequence of str, optional

The characters or list of characters used to indicate the start of a comment. None implies no comments. For backwards compatibility, byte strings will be decoded as 'latin1'. The default is '#'.

`delimiter` : str, optional

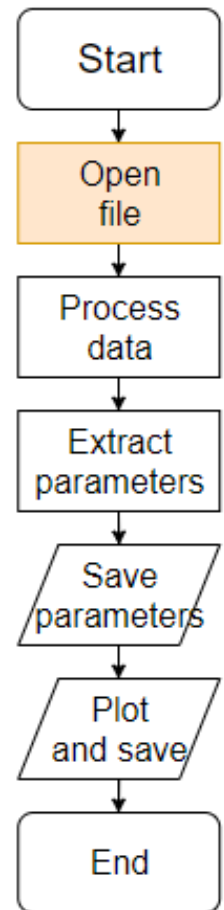
The string used to separate values. For backwards compatibility, byte strings will be decoded as 'latin1'. The default is whitespace.

`converters` : dict, optional

Previous topic
[numpy.fromstring](#)

Next topic
[numpy.core.records.array](#)

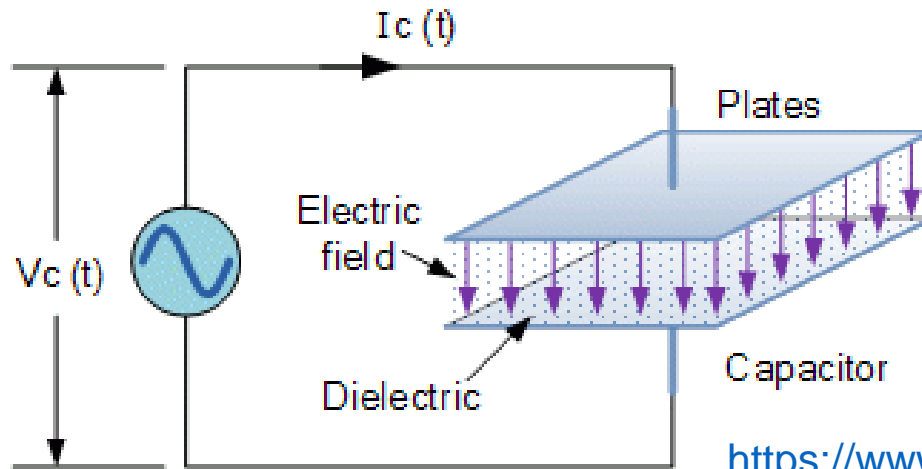
Quick search



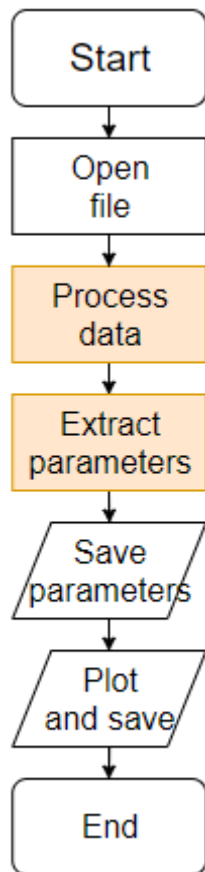
<https://docs.scipy.org/doc/numpy/reference/generated/numpy.loadtxt.html>

What is this?

Complex impedance of a capacitor



https://www.electronicstutorials.ws/filter/filter_1.html



$$Z = \frac{1}{j\omega C}, \omega = 2\pi f, \phi = \arg Z$$

We want to get this value

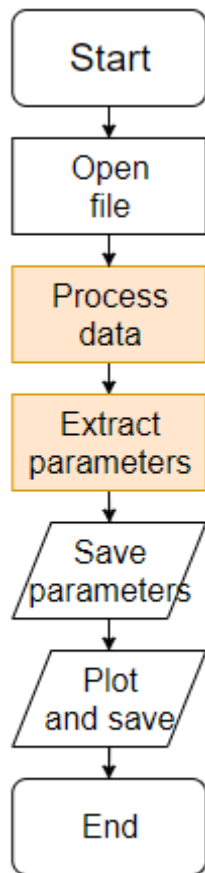
j as imaginary number

Which value is which?

```
3 print(data)
```

f	Z	ϕ
[2.000000e+01	1.00748e+06	-8.57480e+01]
[2.000000e+01	1.02005e+06	-8.72778e+01]
[2.000000e+01	1.02274e+06	-8.92468e+01]
[2.000000e+01	1.02433e+06	-8.92463e+01]
[2.000000e+01	1.05193e+06	-8.79510e+01]
[2.000000e+01	1.08349e+06	-8.72832e+01]
[2.000000e+01	1.07208e+06	-8.70744e+01]
[2.09894e+01	1.02126e+06	-8.55599e+01]
[2.33346e+01	9.04114e+05	-8.82248e+01]
[2.59418e+01	7.99833e+05	-8.70575e+01]
[2.88103e+01	7.27017e+05	-8.72120e+01]

$$Z = \frac{1}{j\omega C}, \omega = 2\pi f, \phi = \arg Z$$



Fitting function

numpy.polynomial.polynomial.polyfit

numpy.polynomial.polynomial.polyfit(x, y, deg, rcond=None, full=False, w=None)
Least-squares fit of a polynomial to data

Return the coefficients of a polynomial of degree *deg* that is the least squares fit to the data values *y* given at points *x*. If *y* is 1-D the returned coefficients will also be 1-D. If *y* is 2-D multiple fits are done, one for each column of *y*, and the resulting coefficients are stored in the corresponding columns of a 2-D return. The fitted polynomial(s) are in the form

$$p(x) = c_0 + c_1 * x + \dots + c_n * x^n,$$

where *n* is *deg*.

Parameters: *x*: array_like, shape (*M*,)

x-coordinates of the *M* sample (data) points (*x*[*i*], *y*[*i*]).

y: array_like, shape (*M*,) or (*M*, *K*)

y-coordinates of the sample points. Several sets of sample points sharing the same x-coordinates can be (independently) fit with one call to [polyfit](#) by passing in for *y* a 2-D array that contains one data set per column.

deg: int or 1-D array_like

Degree(s) of the fitting polynomial(s). If *deg* is a single integer all terms up to and including the *deg*th term are included in the fit. For NumPy versions $\geq 1.11.0$ a list of integers specifying the degrees of the terms to include may be used instead.

rcond: float, optional

Relative condition number of the fit. Singular values smaller than *rcond*, relative to the largest singular value, will be ignored. The default value is $\text{len}(x) * \text{eps}$, where *eps* is the relative precision of the platform's float type, about $2e-16$ in most cases.

full: bool, optional

Switch determining the nature of the return value. When *False* (the default) just the coefficients are returned; when *True*, diagnostic information from the singular value decomposition (used to solve the fit's matrix equation) is also

[\[source\]](#)

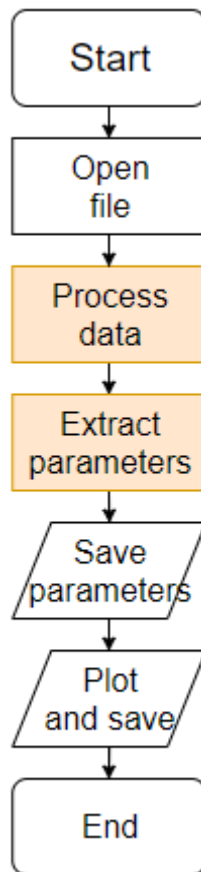
Previous topic

[numpy.polynomial.polynomial.p](#)

Next topic

[numpy.polynomial.polynomial.p](#)

Quick search



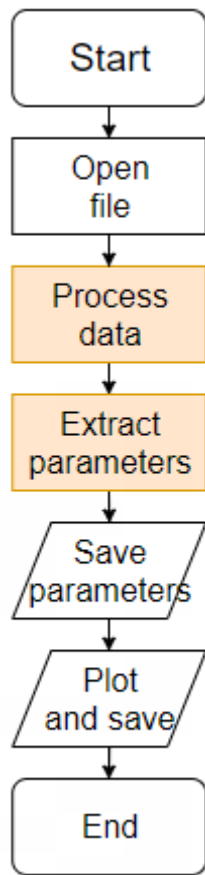
<https://docs.scipy.org/doc/numpy/reference/generated/numpy.polynomial.polynomial.polyfit.html>

Small trick

$$Z = \frac{1}{j \cdot 2\pi f C}$$

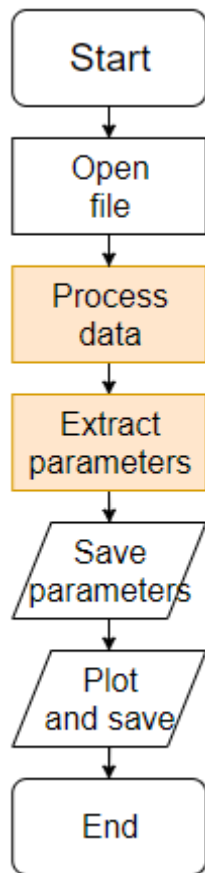
$$\log Z = -\log(2\pi f C)$$

$$\log Z = -\log(2\pi C) - \log f$$



Fitting 1 – Exercise 0-1

$$y = -0.96x + 16.6$$



Save parameter

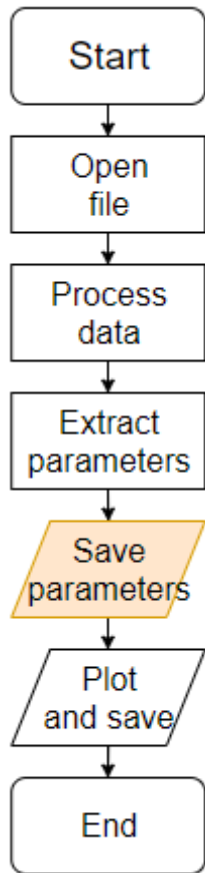
In [30]:

```
logz = np.log(data[:,1])  
logf = np.log(data[:,0])  
from numpy.polynomial import polynomial as P  
p = P.polyfit(logf, logz, 1)  
print(p)
```

```
[16.60429901 -0.96116536]
```

In [31]:

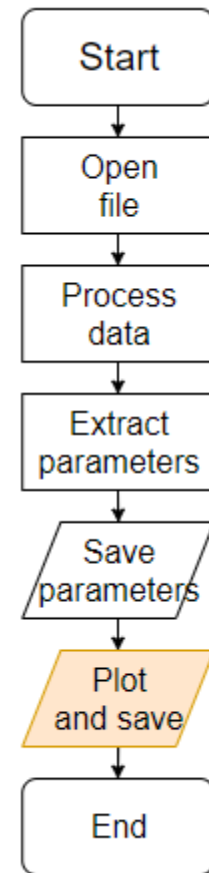
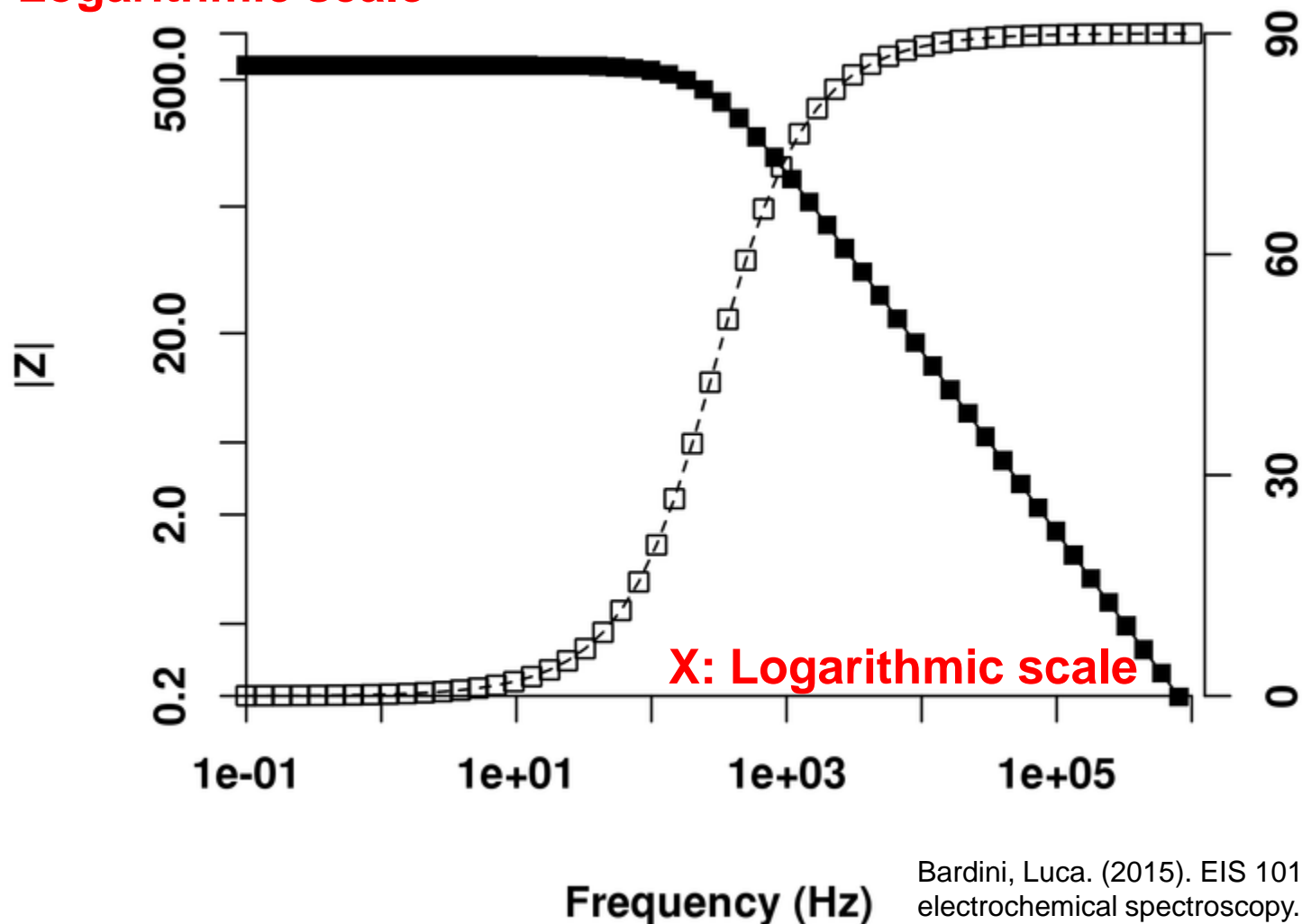
```
np.savetxt("parameters.csv", p, delimiter="\t")
```



Bode plot

Y1: Logarithmic scale

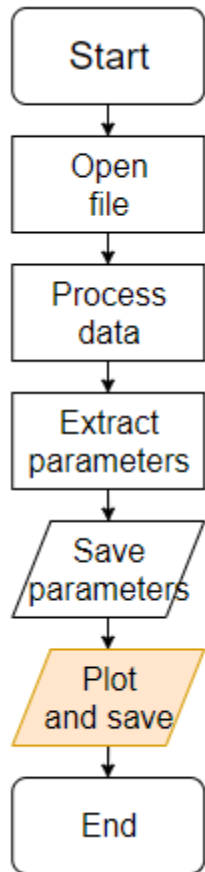
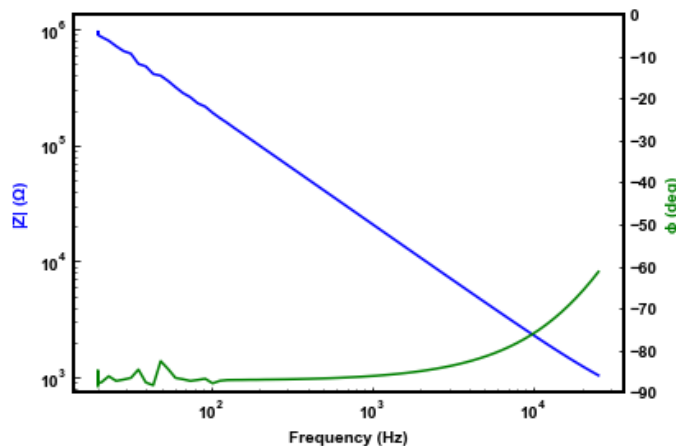
Y2: Linear scale –
I prefer ϕ not inverted



Bardini, Luca. (2015). EIS 101, an introduction to electrochemical spectroscopy. What was a website is now available as a self-contained PDF.

Plotting – Exercise 0-2

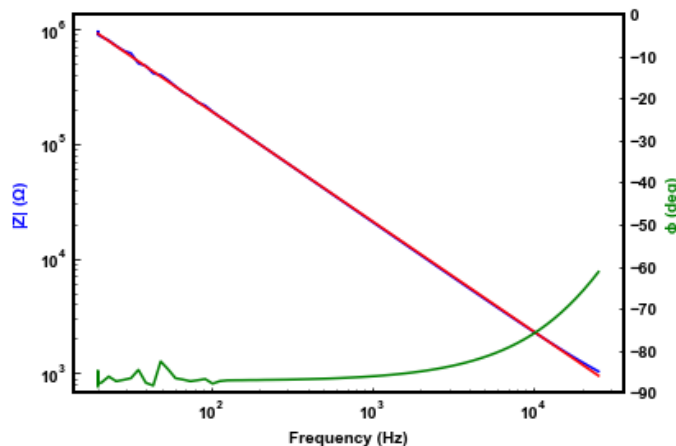
```
ax.set_xlabel("Frequency (Hz)")  
ax.set_ylabel("|Z| ( $\Omega$ )", color="b")  
ax2.set_ylabel(" $\phi$  (deg)", color="g")  
ax2.set_ylim((-90, 0))  
plt.show()
```



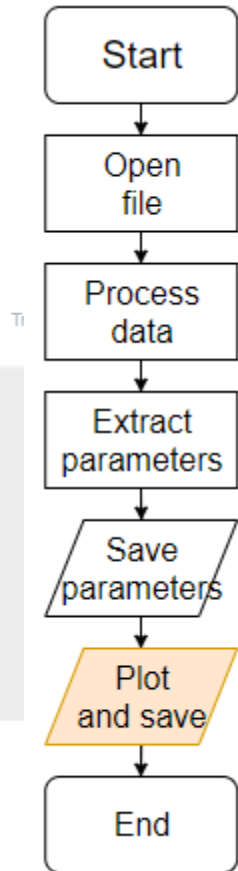
Add fit line

File Edit View Insert Cell Kernel Navigate Widgets Help

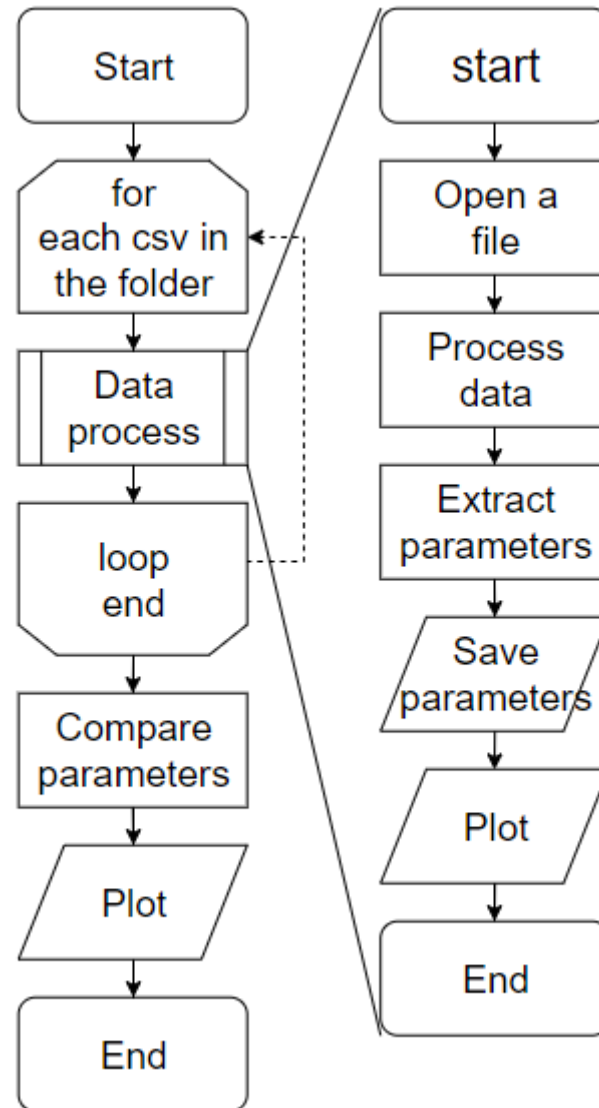
```
fitz = np.exp(p[0] + p[1] * logf)
ax.plot(data[:,0], fitz, color="r")
plt.show()
plt.savefig("impedancefit.png")
```



<Figure size 432x288 with 0 Axes>



That's not all!



Download the code

File Edit View Insert Cell Kernel Navigate Widgets Help

Trusted Python 3

- New Notebook
- Open...
- Make a Copy...
- Save as...
- Rename...
- Save and Checkpoint
- Revert to Checkpoint
- Print Preview
- Download as
- Trusted Notebook
- Close and Halt

- AsciiDoc (.asciidoc)
- HTML (.html)
- HTML with toc (.html)
- LaTeX (.tex)
- Markdown (.md)
- Notebook (.ipynb)
- PDF via LaTeX (.pdf)
- reST (.rst)
- Python (.py)
- Revealjs slides (.slides.html)

```
fitz = np.exp(p[0] + p[1] * logf)
ax.plot(data[:,0], fitz, color="r")
plt.show()
plt.savefig("impedancefit.png")
```

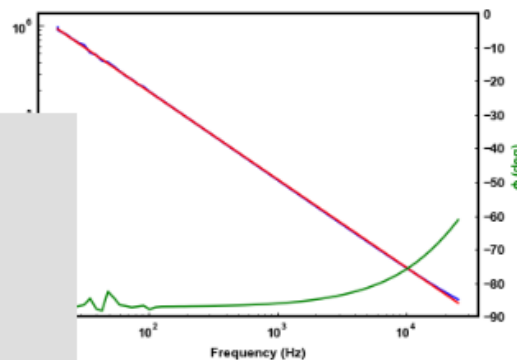
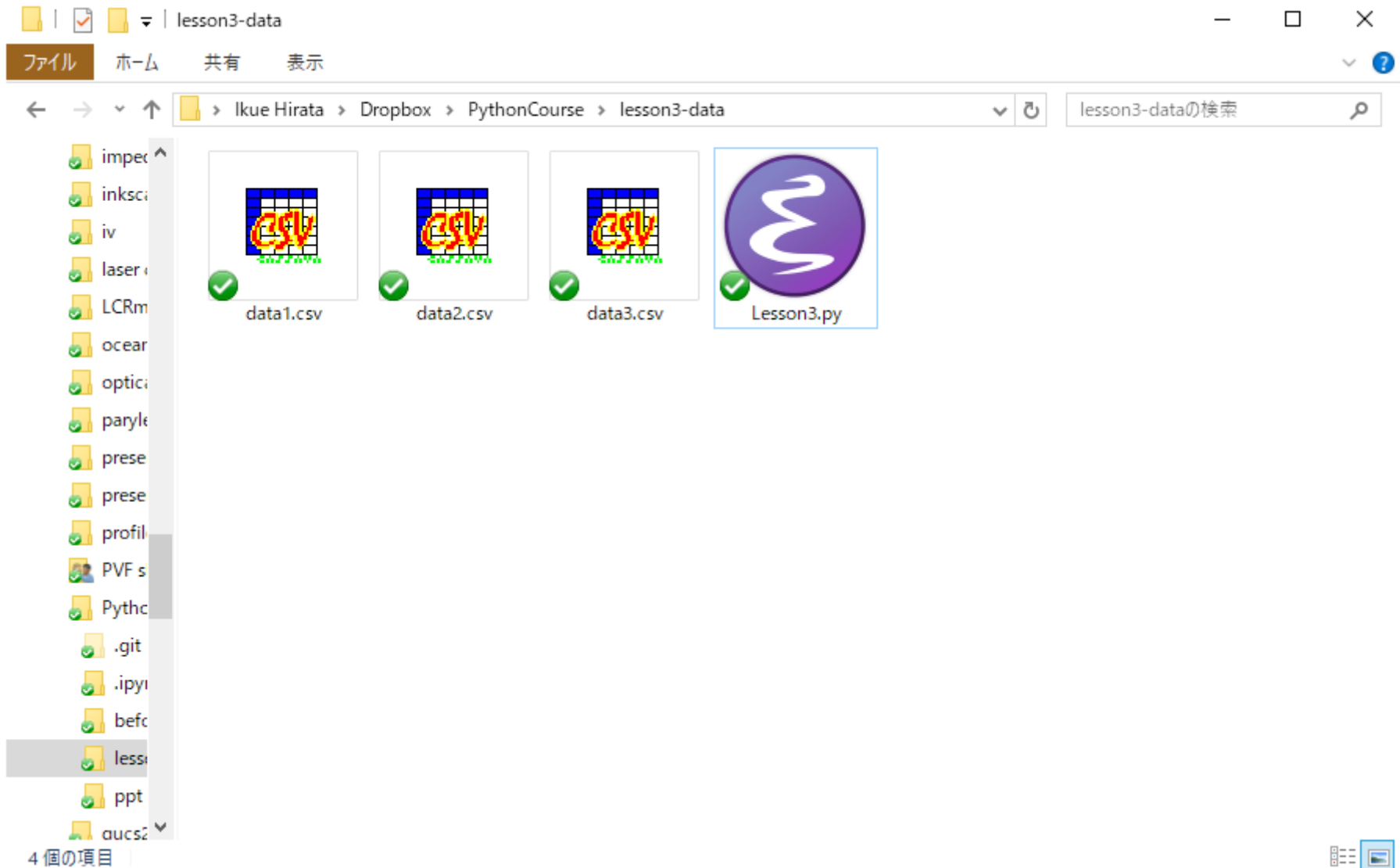
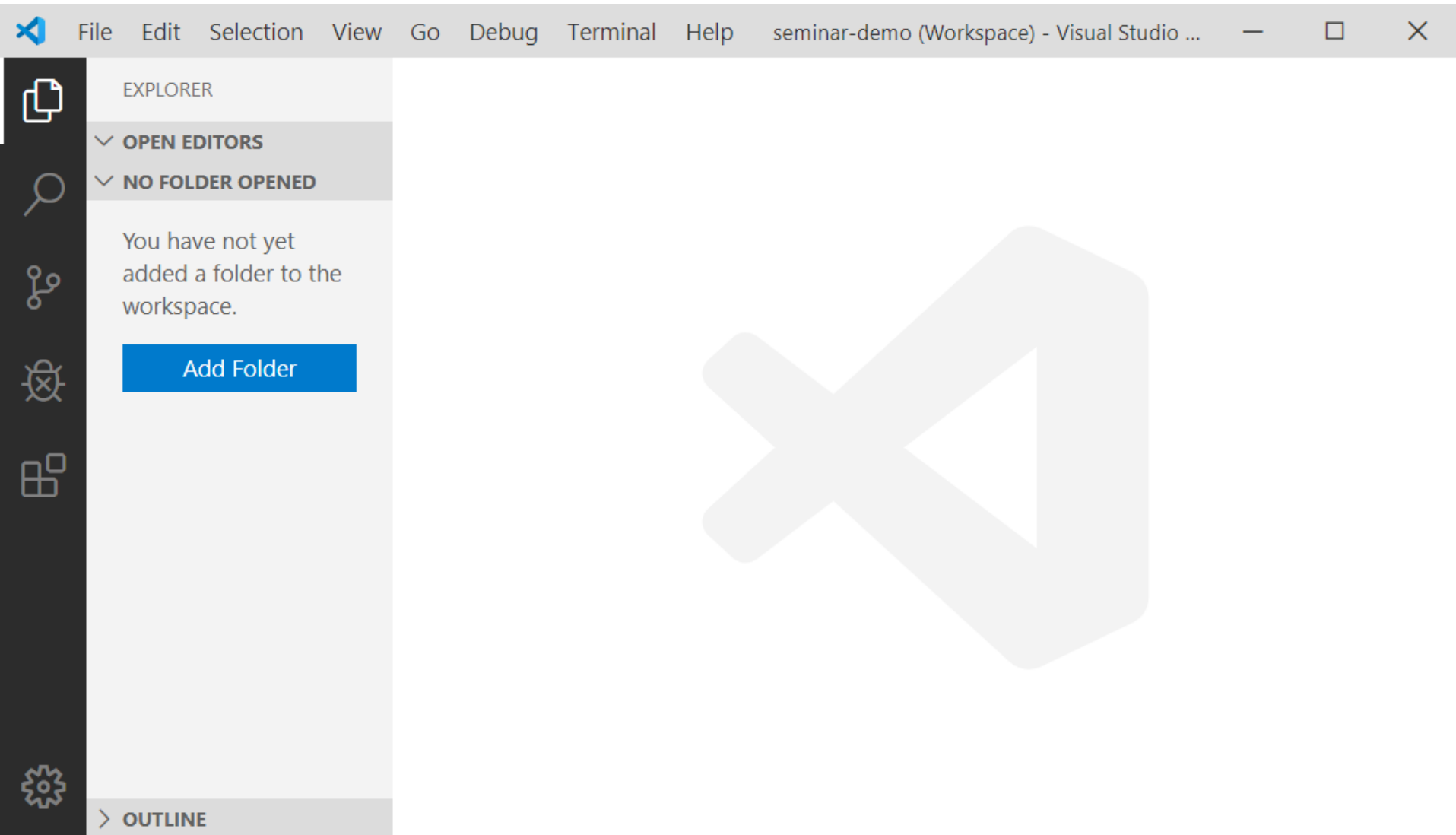


Figure size 432x288 with 0 Axes>

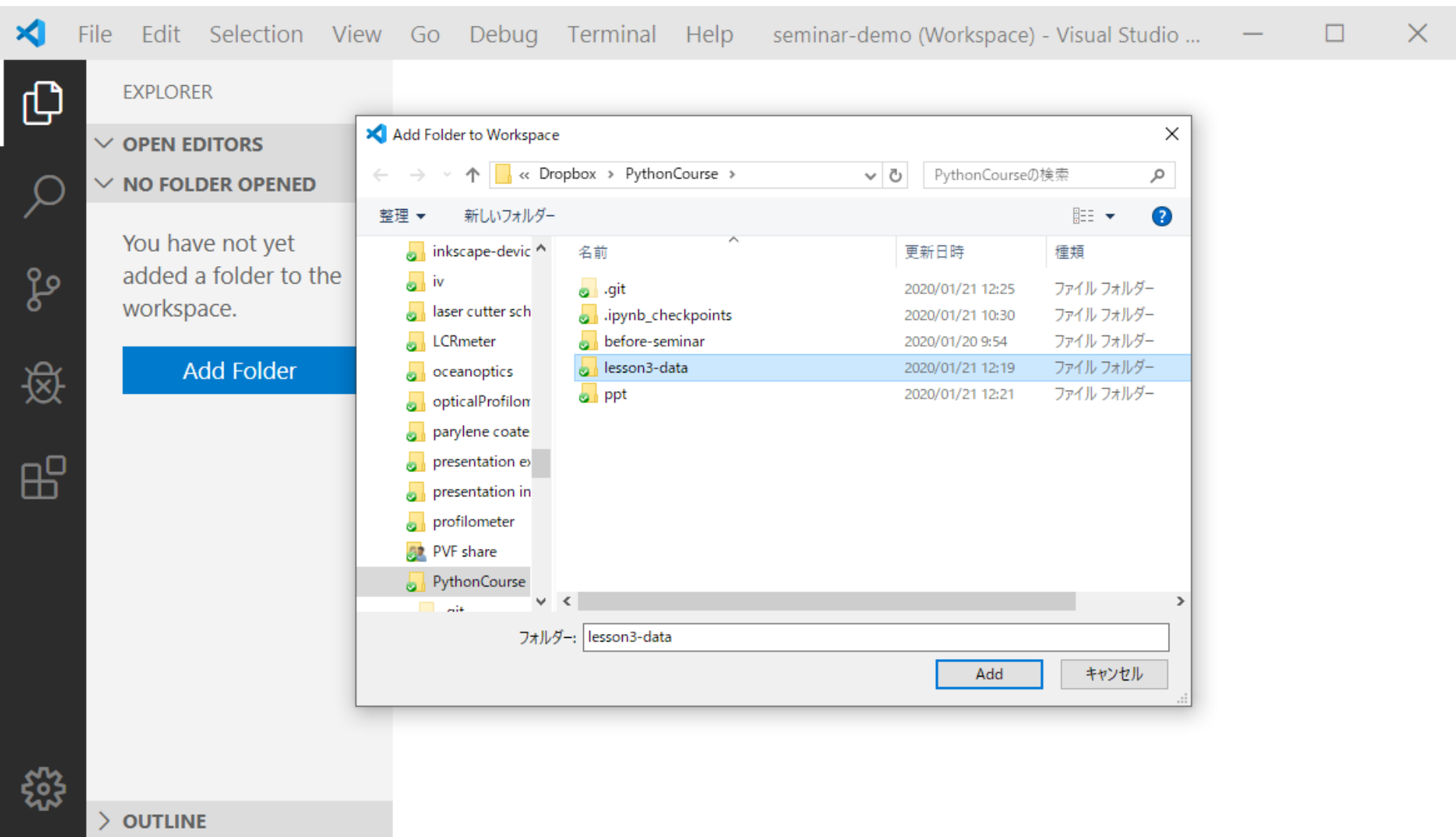
Store it where other data are



Open Visual Studio Code



Add folder – workspace



Open your code

File Edit Selection View Go Debug ... Lesson3.py - seminar-demo (Workspace) - Visual ...

EXPLORER

OPEN EDITOR... 1 UNSAVED

- Lesson3.py 3, U

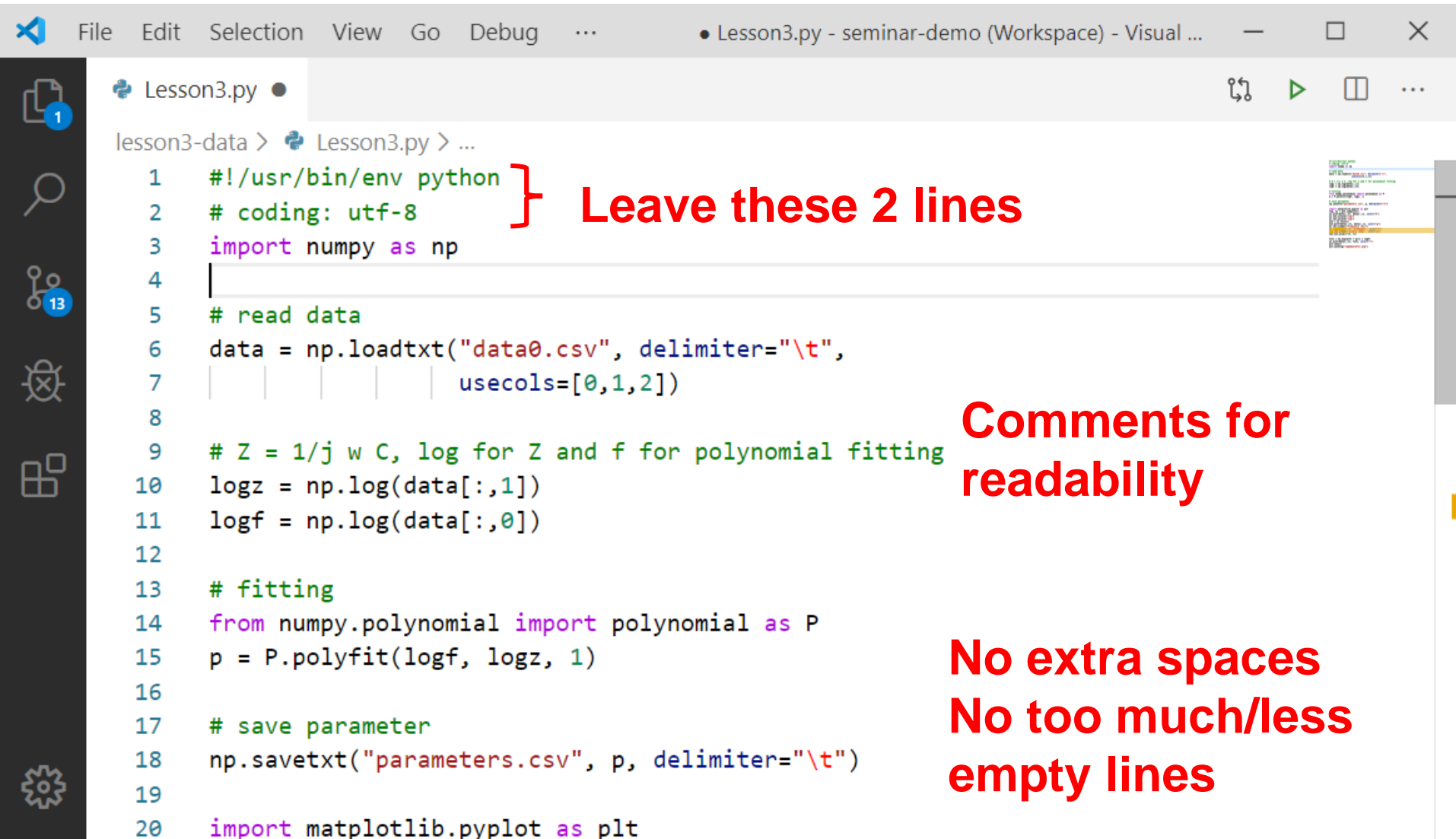
SEMINAR-DEMO (WORKSPACE)

- lesson3-data
 - data1.csv U
 - data2.csv U
 - data3.csv U
 - Lesson3.py 3, U

lesson3-data > Lesson3.py > ...

```
12
13
14 import numpy as np
15 data = np.loadtxt("data0.csv", delimiter='\t')
16 print(data)
17
18
19 # Sometimes the data have many unnecessary parts - you can c
20
21 Run Cell | Run Above | Debug cell
22 # In[3]:
23
24 data = np.loadtxt("data0.csv", delimiter="\t",
25                  usecols=[0,1,2])
26 print(data)
27
28
29 # ### Data processing
30 # Now we have 3 columns. `data[:,0]` is the frequency $f$.
```

Code aesthetics – readability



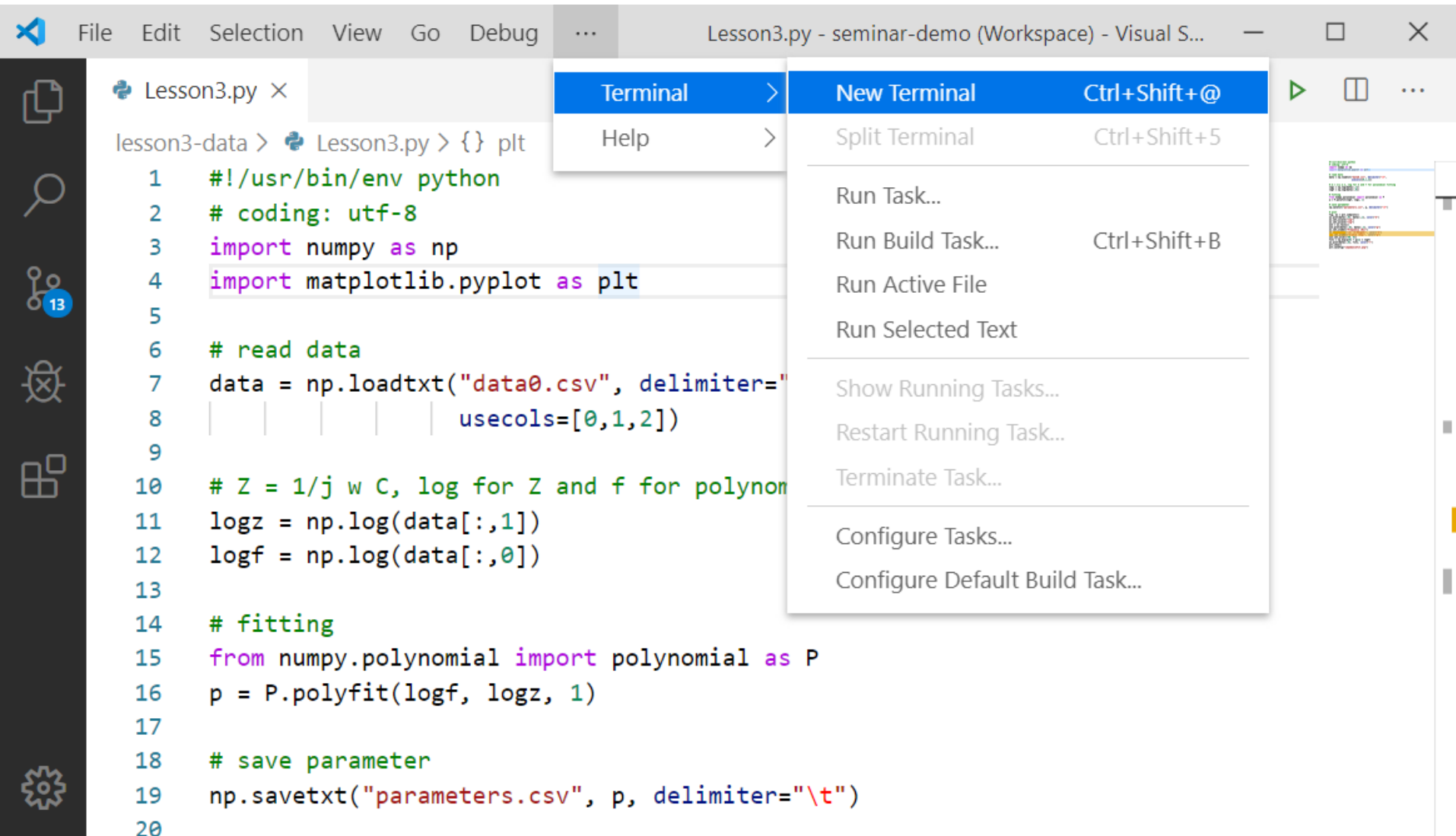
```
Lesson3.py
lesson3-data > Lesson3.py > ...
1  #!/usr/bin/env python
2  # coding: utf-8
3  import numpy as np
4  |
5  # read data
6  data = np.loadtxt("data0.csv", delimiter="\t",
7  | | | | | usecols=[0,1,2])
8
9  # Z = 1/j w C, log for Z and f for polynomial fitting
10 logz = np.log(data[:,1])
11 logf = np.log(data[:,0])
12
13 # fitting
14 from numpy.polynomial import polynomial as P
15 p = P.polyfit(logf, logz, 1)
16
17 # save parameter
18 np.savetxt("parameters.csv", p, delimiter="\t")
19
20 import matplotlib.pyplot as plt
```

Leave these 2 lines

Comments for readability

No extra spaces
No too much/less empty lines

Open Terminal



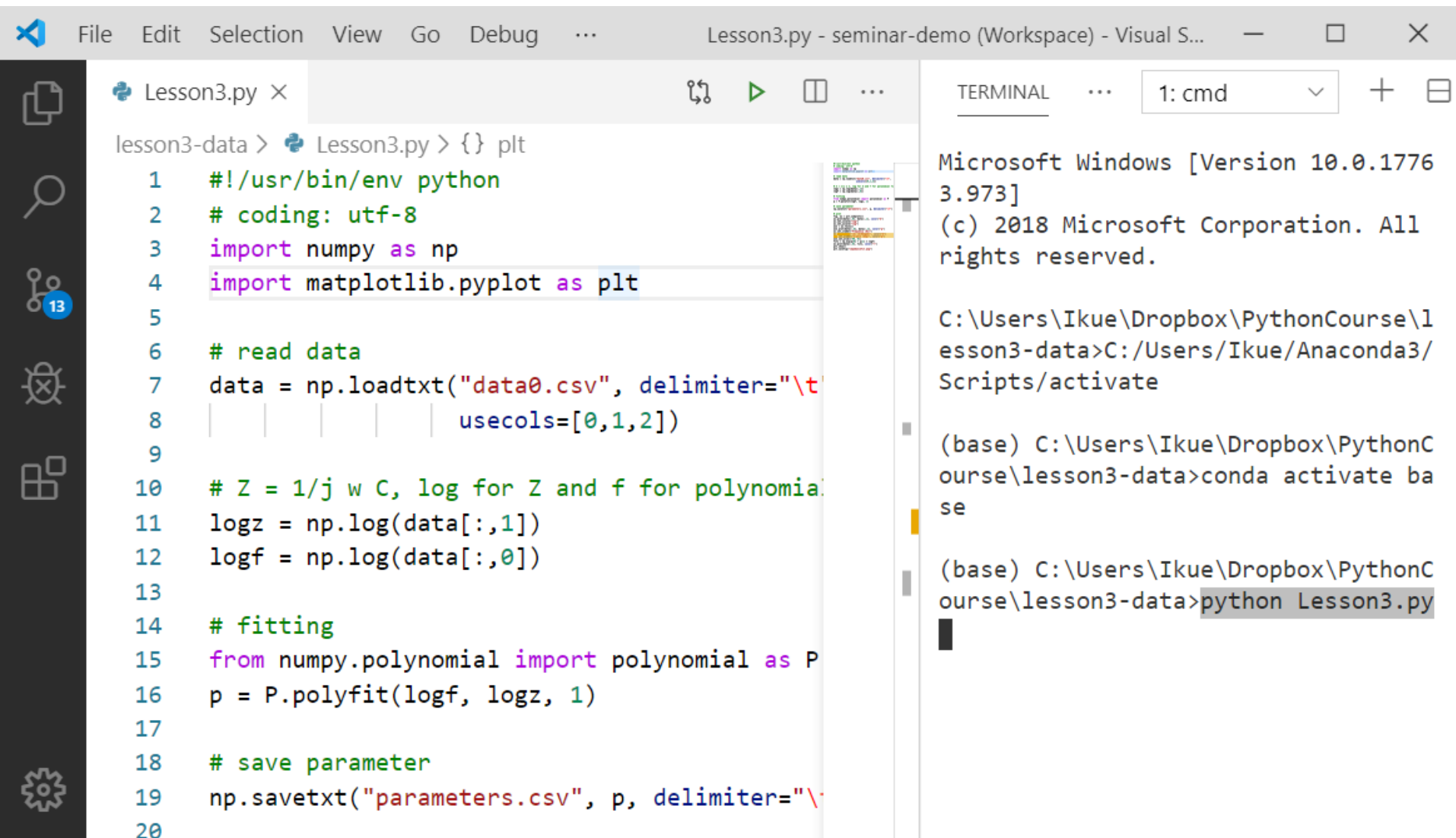
The screenshot shows the Visual Studio Code interface with the 'Terminal' menu open. The menu options are:

- Terminal > New Terminal (Ctrl+Shift+@)
- Terminal > Split Terminal (Ctrl+Shift+5)
- Run Task...
- Run Build Task... (Ctrl+Shift+B)
- Run Active File
- Run Selected Text
- Show Running Tasks...
- Restart Running Task...
- Terminate Task...
- Configure Tasks...
- Configure Default Build Task...

The background shows a Python script named 'Lesson3.py' with the following code:

```
1  #!/usr/bin/env python
2  # coding: utf-8
3  import numpy as np
4  import matplotlib.pyplot as plt
5
6  # read data
7  data = np.loadtxt("data0.csv", delimiter="
8  | | | | | usecols=[0,1,2])
9
10 # Z = 1/j w C, log for Z and f for polynomial
11 logz = np.log(data[:,1])
12 logf = np.log(data[:,0])
13
14 # fitting
15 from numpy.polynomial import polynomial as P
16 p = P.polyfit(logf, logz, 1)
17
18 # save parameter
19 np.savetxt("parameters.csv", p, delimiter="\t")
20
```

Run the code



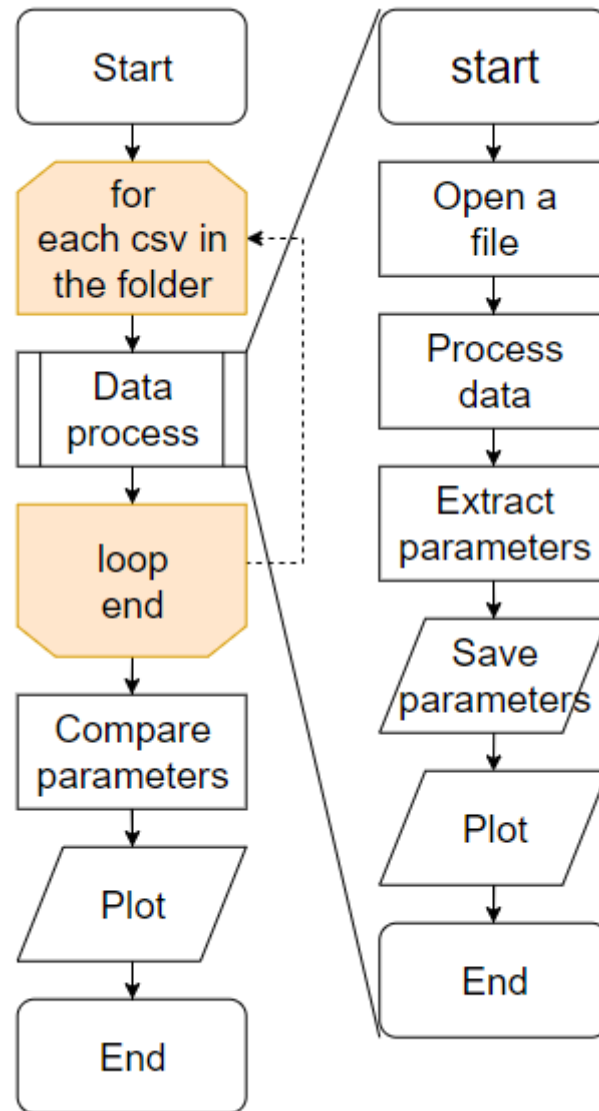
The image shows a Visual Studio Code window with a Python file named Lesson3.py. The code in the file uses NumPy and Matplotlib to load data, calculate logarithms, fit a polynomial, and save the parameters. To the right, a terminal window shows the command prompt output, including the Windows version, copyright notice, and the successful execution of the script.

```
File Edit Selection View Go Debug ... Lesson3.py - seminar-demo (Workspace) - Visual S...  
Lesson3.py X  
lesson3-data > Lesson3.py > {} plt  
1  #!/usr/bin/env python  
2  # coding: utf-8  
3  import numpy as np  
4  import matplotlib.pyplot as plt  
5  
6  # read data  
7  data = np.loadtxt("data0.csv", delimiter="\t",  
8  | | | | | usecols=[0,1,2])  
9  
10 # Z = 1/j w C, log for Z and f for polynomial  
11 logz = np.log(data[:,1])  
12 logf = np.log(data[:,0])  
13  
14 # fitting  
15 from numpy.polynomial import polynomial as P  
16 p = P.polyfit(logf, logz, 1)  
17  
18 # save parameter  
19 np.savetxt("parameters.csv", p, delimiter="\t")  
20
```

TERMINAL 1: cmd

```
Microsoft Windows [Version 10.0.17763.973]  
(c) 2018 Microsoft Corporation. All rights reserved.  
  
C:\Users\Ikue\Dropbox\PythonCourse\lesson3-data>C:/Users/Ikue/Anaconda3/Scripts/activate  
  
(base) C:\Users\Ikue\Dropbox\PythonCourse\lesson3-data>conda activate base  
  
(base) C:\Users\Ikue\Dropbox\PythonCourse\lesson3-data>python Lesson3.py
```


Getting multiple files



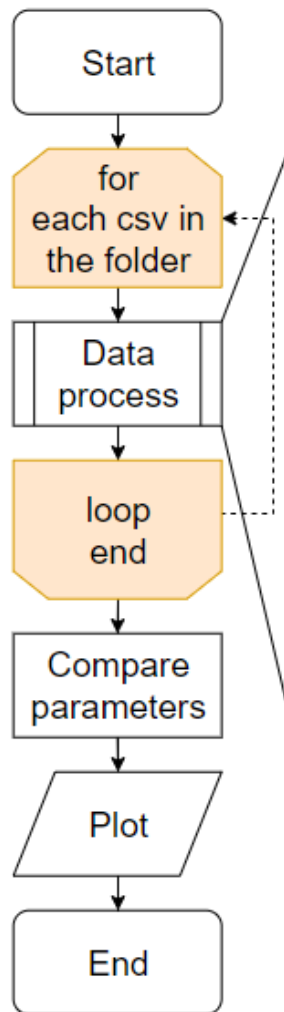
glob

File Edit View Insert Cell Kernel Navigate Widgets Help

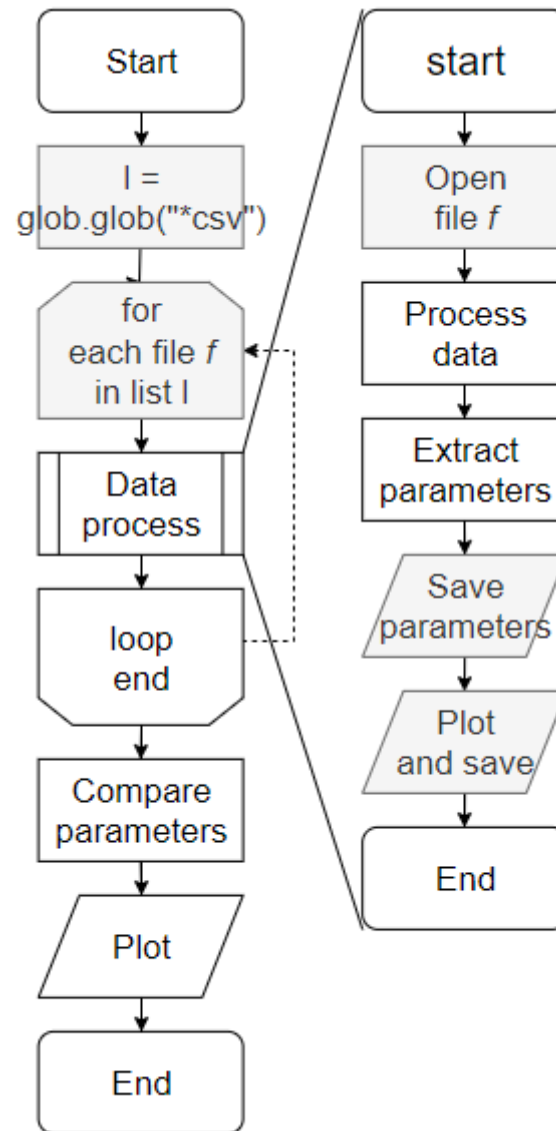
In [1]:

```
import glob
l = glob.glob("*csv")
print(l)
```

```
['data1.csv', 'data2.csv', 'data3.csv']
```



Pay attention to save file name

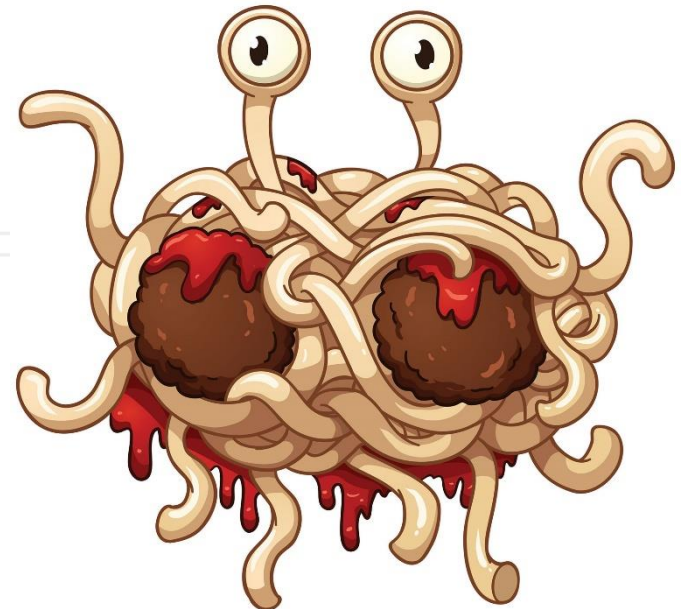


“Spaghetti” code

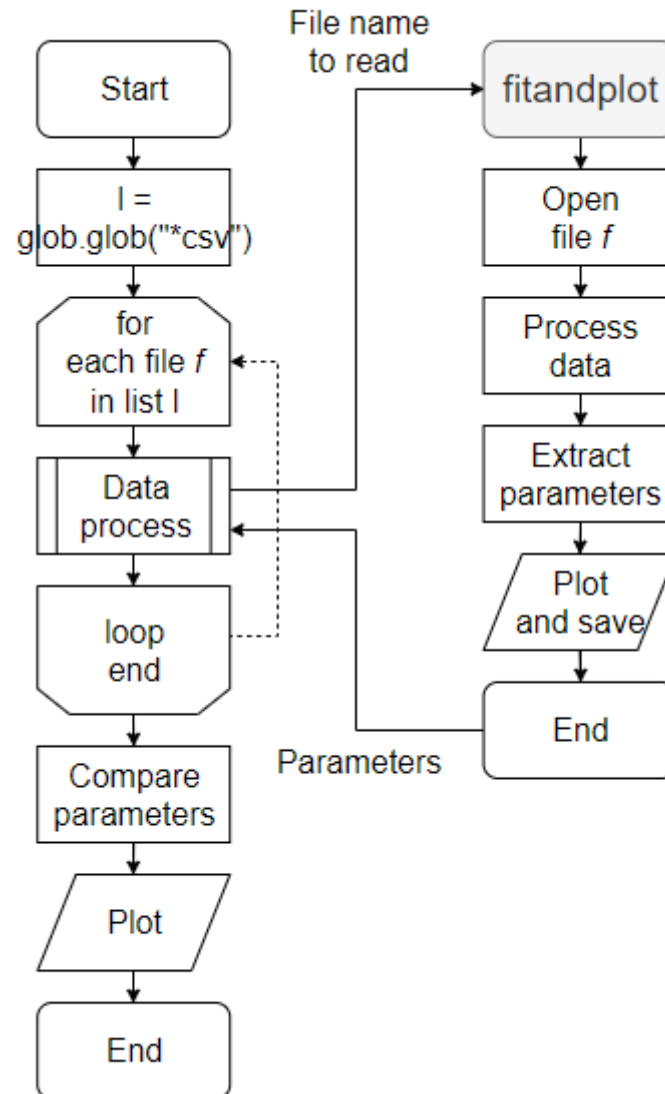
File Edit Selection View Go Debug Terminal Help Lesson3.py - seminar-demo (Workspace) - Visual Studio Code

Lesson3.py x

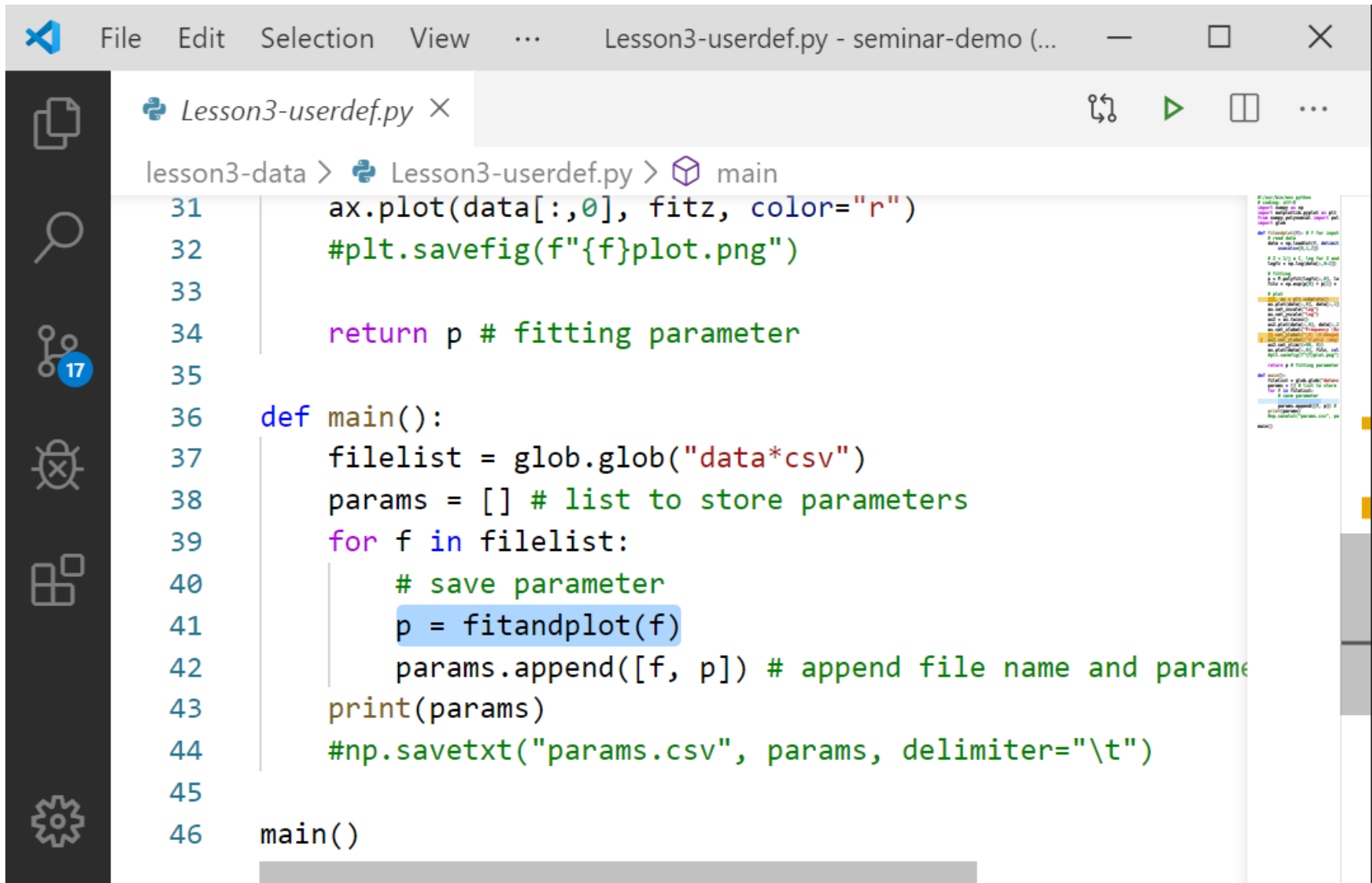
```
lesson3-data > Lesson3.py > ...
4 import matplotlib.pyplot as plt
5 import glob
6
7 filelist = glob.glob("*.csv")
8 params = [] # list to store parameters
9 for f in filelist:
10     # read data
11     data = np.loadtxt(f, delimiter="\t",
12                      usecols=[0,1,2])
13
14     # Z = 1/j w C, log for Z and f for polynomial fitting
15     logz = np.log(data[:,1])
16     logf = np.log(data[:,0])
17
18     # fitting
19     from numpy.polynomial import polynomial as P
20     p = P.polyfit(logf, logz, 1)
21
22     # save parameter
23     np.savetxt(f"{f}.csv", p, delimiter="\t") # save file name using original file
24     params.append([f, p]) # append file name and parameter in the storage
25
26     # plot
27     fig, ax = plt.subplots()
28     ax.plot(data[:,0], data[:,1], color="b")
29     ax.set_xscale("log")
30     ax.set_yscale("log")
31     ax2 = ax.twinx()
32     ax2.plot(data[:,0], data[:,2], color="g")
33     ax.set_xlabel("Frequency (Hz)")
34     ax.set_ylabel("|Z| ($\Omega$)", color="b")
35     ax2.set_ylabel("$\phi$ (deg)", color="g")
36     ax2.set_ylim((-90, 0))
37     fitz = np.exp(p[0] + p[1] * logf)
38     ax.plot(data[:,0], fitz, color="r")
39     plt.show()
40     plt.savefig(f"{f}.png")
41
```



User defined function



Using user defined function

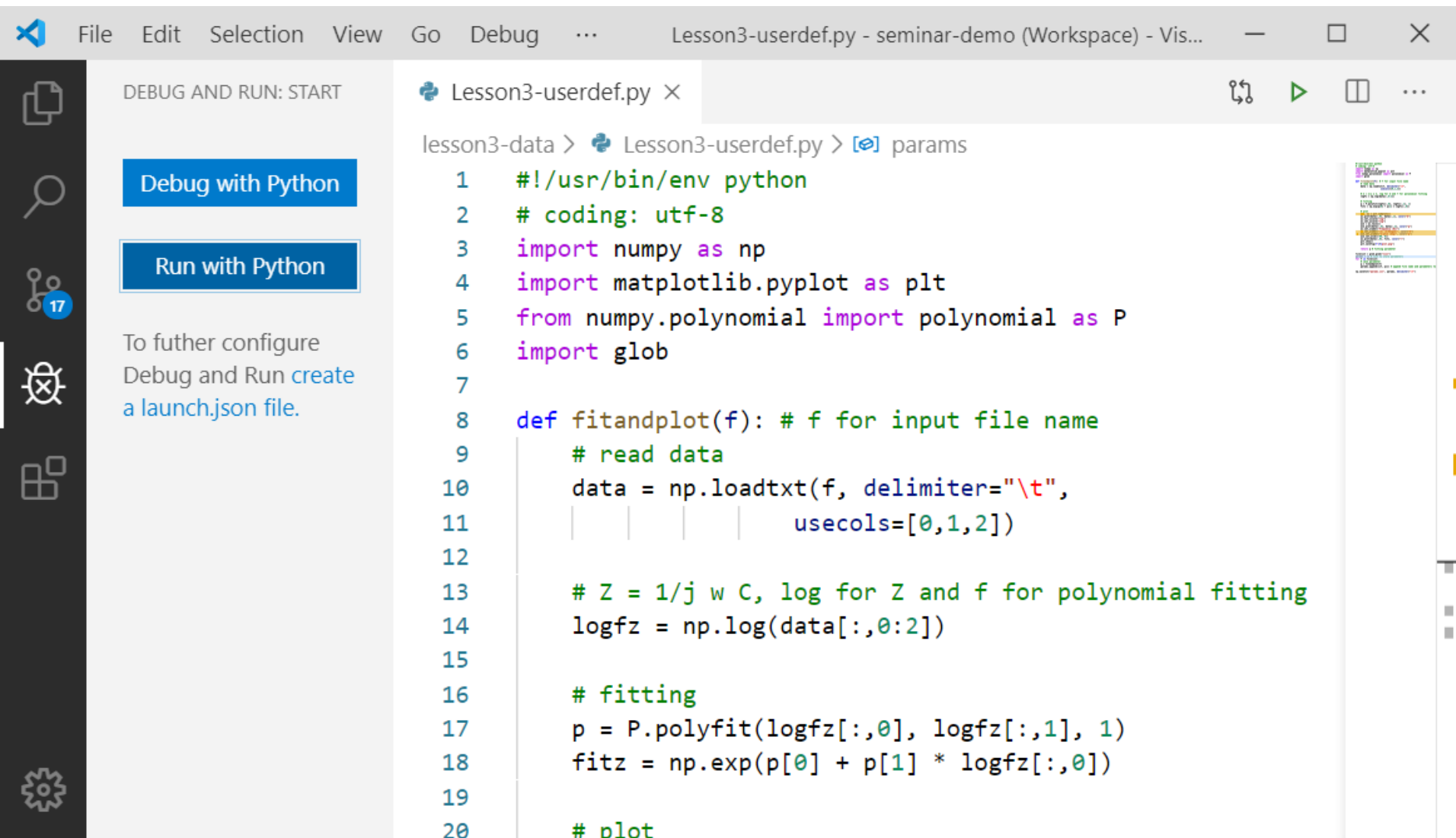


The screenshot shows a code editor window titled "Lesson3-userdef.py - seminar-demo (...)". The editor contains a Python script with the following code:

```
lesson3-data > Lesson3-userdef.py > main
31 ax.plot(data[:,0], fitz, color="r")
32 #plt.savefig(f"{f}plot.png")
33
34 return p # fitting parameter
35
36 def main():
37     filelist = glob.glob("data*csv")
38     params = [] # list to store parameters
39     for f in filelist:
40         # save parameter
41         p = fitandplot(f)
42         params.append([f, p]) # append file name and parameter
43     print(params)
44     #np.savetxt("params.csv", params, delimiter="\t")
45
46 main()
```

The code defines a function `fitandplot(f)` (partially visible on the right) that takes a file name `f` as input, reads data from `data.csv`, fits a function `fitz` to the data, and returns the fitting parameter `p`. The `main()` function uses `glob.glob("data*csv")` to find all CSV files in the `data` directory, iterates over them, calls `fitandplot(f)` to fit the data, and appends the file name and parameter to a list `params`. Finally, it prints the list of parameters.

Debugging



File Edit Selection View Go Debug ... Lesson3-userdef.py - seminar-demo (Workspace) - Vis...

DEBUG AND RUN: START

Debug with Python

Run with Python

To further configure Debug and Run [create a launch.json file](#).

Lesson3-userdef.py X

lesson3-data > Lesson3-userdef.py > [params]

```
1  #!/usr/bin/env python
2  # coding: utf-8
3  import numpy as np
4  import matplotlib.pyplot as plt
5  from numpy.polynomial import polynomial as P
6  import glob
7
8  def fitandplot(f): # f for input file name
9      # read data
10     data = np.loadtxt(f, delimiter="\t",
11                      usecols=[0,1,2])
12
13     # Z = 1/j w C, log for Z and f for polynomial fitting
14     logfz = np.log(data[:,0:2])
15
16     # fitting
17     p = P.polyfit(logfz[:,0], logfz[:,1], 1)
18     fitz = np.exp(p[0] + p[1] * logfz[:,0])
19
20     # plot
```

Breakpoints

The screenshot shows the Visual Studio Code interface with a Python file named `Lesson3-userdef.py` open. The file is located in the workspace `Lesson3-userdef.py - seminar-demo (Workspace) - Vis...`. The left sidebar shows the **DEBUG AND RUN: START** panel with buttons for **Debug with Python** and **Run with Python**. Below these buttons, it says: "To further configure Debug and Run [create a launch.json file](#)."

The main editor displays the code in `Lesson3-userdef.py`. The code is as follows:

```
16 # fitting
17 p = P.polyfit(logfz[
18 fitz = np.exp(p[0] +
19
20 # plot
21 fig, ax = plt.subplo
22 ax.plot(data[:,0], d
23 ax.set_xscale("log")
24 ax.set_yscale("log")
25 ax2 = ax.twinx()
26 ax2.plot(data[:,0],
27 ax.set_xlabel("Freque
28 ax.set_ylabel("|Z| (
29 ax2.set_ylabel("$\ph
30 ax2.set_ylim((-90, 0
31 ax.plot(data[:,0], f
32 plt.show()
33 plt.savefig(f"{f}plo
34
35 return # fitting
```

Two red dots indicate breakpoints: one at line 21 (labeled "Breakpoint") and another at line 28. The code is partially obscured by a tooltip for the `plt.savefig` function.

The right sidebar shows the **TERMINAL** panel with the command prompt `1: Python Del`. The terminal output shows the following commands and their results:

```
Lesson3-data>C:/Users/Ikue/Anaconda3/Scripts/activate
(base) c:\Users\Ikue\Dropbox\PythonCourse\lesson3-data>conda activate ba
se
(base) c:\Users\Ikue\Dropbox\PythonCourse\lesson3-data>cd c:\Users\Ikue\Dropbox\PythonCourse\lesson3-data && cmd /C "set "PYTHONIOENCODING=UTF-8" && set "PYTHONUNBUFFERED=1" && C:\Users\Ikue\Anaconda3\python.exe c:\Users\Ikue\.vscode\extensions\ms-python.python-2020.1.58038\pythonFiles\ptvsd_launcher.py --default --client --host localhost --port 55080 c:\Users\Ikue\Dropbox\PythonCourse\lesson3-data\Lesson3-userdef.py "
```

The terminal output is partially obscured by a tooltip for the `plt.savefig` function.

Run

File Edit Selection View Go Debug ... Lesson3-userdef.py - seminar-demo (Workspace) - Vis...

DEBUG AND ...

VARIABLES

Locals

- > data: array([[2.000000e+0...
- f: 'data1.csv'
- > fitz: array([543273.69168...
- > logfz: array([[2.9957322...
- > p: array([15.26184592, -0...

WATCH

CALL STACK PAUSED ON BREAKPOINT

- fitandplot Lesson3-userde...
- <module> Lesson3-userdef.py

Lesson3-userdef.py > fitandplot

```
16 # fitting
17 p = P.polyfit(
18 fitz = np.exp(
19
20 # plot
21 fig, ax = plt.
22 ax.plot(data[:
23 ax.set_xscale(
24 ax.set_yscale(
25 ax2 = ax.twinx
26 ax2.plot(data[
27 ax.set_xlabel(
28 ax.set_ylabel(
29 ax2.set_ylabel
30 ax2.set_ylim(
31 ax.plot(data[:
32 plt.show()
33 plt.savefig(f"
34
35 return # fit
```

TERMINAL ... 1: Python Del

```
c:\Users\Ikue\Dropbox\PythonCourse\l
esson3-data>C:/Users/Ikue/Anaconda3/
Scripts/activate

(base) c:\Users\Ikue\Dropbox\PythonC
ourse\lesson3-data>conda activate ba
se

(base) c:\Users\Ikue\Dropbox\PythonC
ourse\lesson3-data>cd c:\Users\Ikue\
Dropbox\PythonCourse\lesson3-data &&
cmd /C "set "PYTHONIOENCODING=UTF-8
" && set "PYTHONUNBUFFERED=1" && C:\
Users\Ikue\Anaconda3\python.exe c:\U
sers\Ikue\.vscode\extensions\ms-pyth
on.python-2020.1.58038\pythonFiles\p
tvds_launcher.py --default --client
--host localhost --port 55080 c:\Use
rs\Ikue\Dropbox\PythonCourse\lesson3
-data\Lesson3-userdef.py "
```

Exception handling

File Edit View Insert Cell Kernel Navigate Widgets Help

Trusted Python 3

In [35]:

```
▼ num = [1, 2, 3, 0]
  for n in num:
      print(n, 1/n)
  print("process end")
```

1 1.0

2 0.5

3 0.3333333333333333

ZeroDivisionError

T

Traceback (most recent call last)

<ipython-input-35-89a0c53ab2bc> in <module>

try and except

In [51]:

```
▼ for n in num:  
▼     try:  
        print(n, 1/n)  
▼     except Exception as e:  
        print(f"there's an error: {e}")  
print("process end")
```

1 1.0

2 0.5

3 0.3333333333333333

there's an error: division by zero

process end

Exercise 1

[Code](#)[Issues 0](#)[Pull requests 0](#)[Actions](#)[Projects 0](#)[Wiki](#)[Security](#)[Insights](#)[Settings](#)

Branch: master

[CodingSeminarCMBR](#) / [exercises_and_model_answers](#) / Lesson3_Exercise1.ipynb[Find file](#)[Copy path](#) ikuehirata Lesson 3 added

cda8712 1 hour ago

[1 contributor](#)

62 lines (62 sloc) | 2.15 KB



Raw

Blame

History



Lesson 3 - Exercise 1

1. Using the impedance measurement data .csv provided in [lesson3-data in GitHub](#) fit the values to the formula

$$Z = \frac{1}{j\omega C}$$

where $C = 2\pi f$, j is the imaginary number, and extract C value. The first column of the data is the frequency f , the second the impedance Z , the third the phase $\phi = \arg(Z)$. Plot the experimental value in Bode plot. Add the fit value of $|Z|$.

Hint: use the range range where the experimental fits the theoretical value.

2. The experimental value does not follow the theoretical value explicitly because of some parasite resistances. Capacitor model can be explained as

$$Z = \frac{1}{j\omega C} // R_p + R_s$$

where R_p is parallel resistance, R_s series resistance, operator $//$

$$a // b = \frac{1}{\frac{1}{a} + \frac{1}{b}}$$

, meaning a parallel circuit. Plot the experimental value in Bode plot. Add the fit value of $|Z|$.

Hint: if R_p is huge, it can be ignored.

3. Plot C vs file name to visually compare the extracted C value.

Exercise 2

Plan, code, and run your own program for your own experiments.

To study by yourself: codecademy

Learn Python 2 | Codecademy

codecademy Catalog Pricing Try Pro For Free

Learn Python 2

Learn the basics of the world's fastest growing and most popular programming language used by software engineers, analysts, data scientists, and machine learning engineers alike.

START

*This is outdated Python 2 – print() function is a bit different from Python 3

Overview Syllabus

Join
4,719,900
people who have taken