

Debugging & Matplotlib

Jannusch Bigge

28.11.2023

Matplotlib

Matplotlib is a plotting library.

Matplotlib is a plotting library.
Many different plots are possible.

Matplotlib is a plotting library.

Many different plots are possible.

- Line plots
- Scatter plots
- Bar plots
- 3D plots
- ...

Works great together with other libraries (e.g. numpy).

Works great together with other libraries (e.g. numpy).
Allows you to customize your plots.

Attention

Matplotlib has two modes:

- Implicit mode

Attention

Matplotlib has two modes:

- Implicit mode
 - Global state-based interface mode

Attention

Matplotlib has two modes:

- Implicit mode
 - Global state-based interface mode
 - like MATLAB

Attention

Matplotlib has two modes:

- Implicit mode
 - Global state-based interface mode
 - like MATLAB
 - not recommended (but always used on the internet)

Attention

Matplotlib has two modes:

- Implicit mode

Attention

Matplotlib has two modes:

- Implicit mode
- Explicit mode

Attention

Matplotlib has two modes:

- Implicit mode
- Explicit mode
 - Object-oriented (OO) interface

Attention

Matplotlib has two modes:

- Implicit mode
- Explicit mode
 - Object-oriented (OO) interface
 - Build up the visualization in an instance of *figure.Figure*

Attention

Matplotlib has two modes:

- Implicit mode
- Explicit mode
 - Object-oriented (OO) interface
 - Build up the visualization in an instance of *figure.Figure*
 - **Recommended to use!**

Implicit mode

```
import matplotlib.pyplot as plt  
plt.plot([1, 2, 3, 4], [0, 0.5, 1, 0.2])
```

Implicit mode

```
import matplotlib.pyplot as plt  
plt.plot([1, 2, 3, 4], [0, 0.5, 1, 0.2])
```

Explicit mode

```
import matplotlib.pyplot as plt  
fig = plt.figure()  
ax = fig.subplots()  
ax.plot([1, 2, 3, 4], [0, 0.5, 1, 0.2])
```

Matplotlib - Why explicit?

- More control over your plots

Matplotlib - Why explicit?

- More control over your plots
- If you have to work on an old unreferenced axes

Matplotlib - Why explicit?

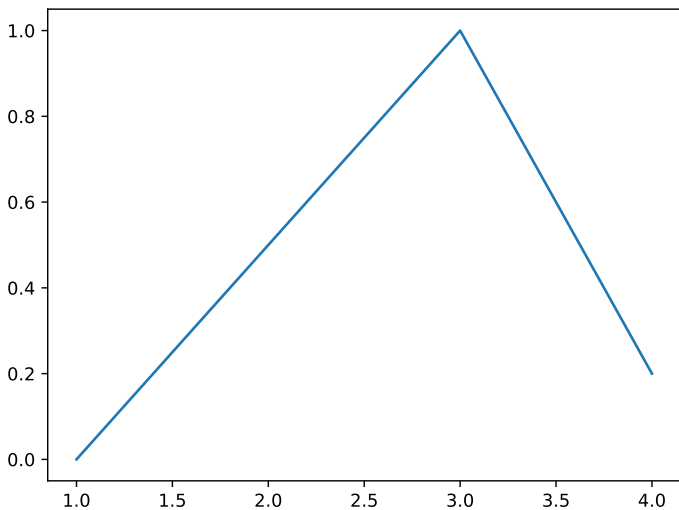
- More control over your plots
- If you have to work on an old unreferenced axes
- Third party often uses explicit mode

Matplotlib - Example

```
import matplotlib.pyplot as plt

fig = plt.figure()
ax = fig.subplots()
ax.plot([1, 2, 3, 4], [0, 0.5, 1, 0.2])
```

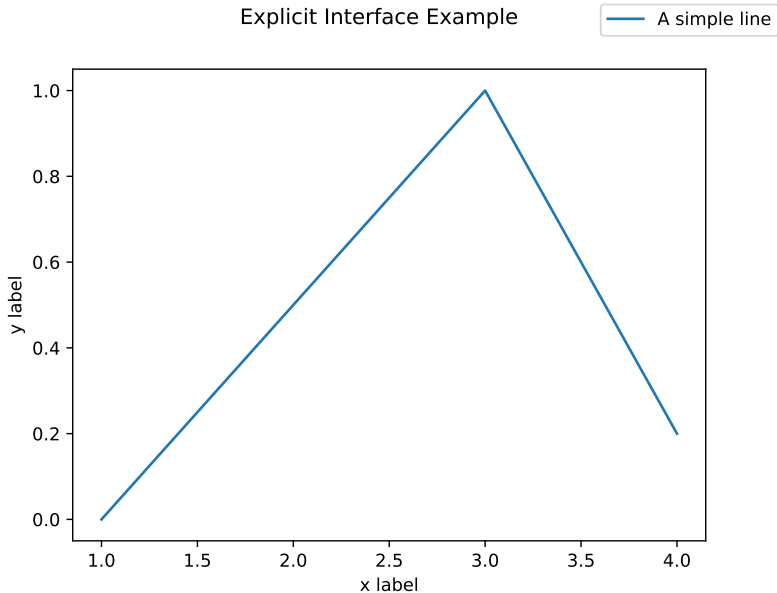
Matplotlib - Example



Matplotlib - Example

```
# Add some text to the figure  
fig.suptitle('Explicit Interface Example')  
ax.set_xlabel('x_label')  
ax.set_ylabel('y_label')
```

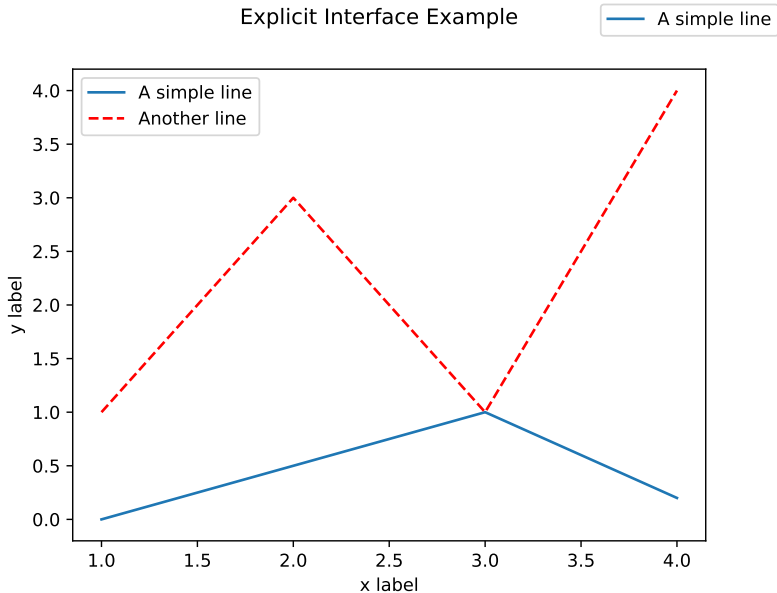

Matplotlib - Example



Matplotlib - Example

```
# Add another line and a second legend  
ax.plot([1, 2, 3, 4], [1, 3, 1, 4], 'r--')  
ax.legend(['A_simple_line', 'Another_line'])  
# And save the figure  
fig.savefig("/workspaces/python_course/slides/week_five/figures/  
    myplot.pdf")
```

Matplotlib - Example



Debugging

- Debugging is the process of finding and fixing errors in your code

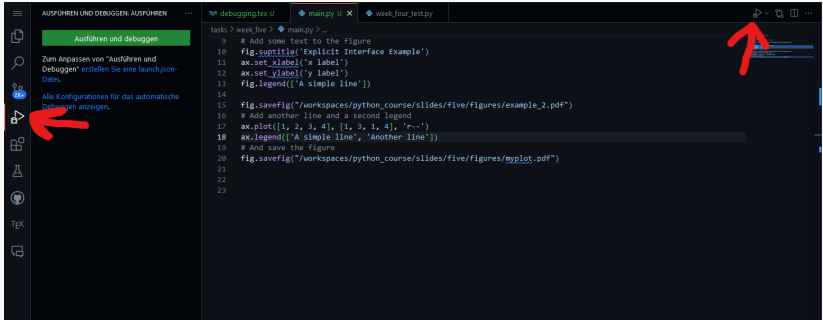
- Debugging is the process of finding and fixing errors in your code
- Special program attached to the running program

- Debugging is the process of finding and fixing errors in your code
- Special program attached to the running program
- Allows you to inspect the program at runtime

Usecases:

- catch errors and show you the code and the state of the code
- set breakpoints and hold the program at a certain point

Debugging - HowTo



Debugging - HowTo

The screenshot shows the VS Code Python IDE interface with several panels and annotations. The main editor displays a Python script with a breakpoint at line 13. The left sidebar contains the 'VARIABLEN' (Variables) panel, the 'AUFRUFLISTE' (Call Stack) panel, and the 'HALTEPUNKTE' (Breakpoints) panel. The bottom panel shows the 'TERMINAL' output.

Annotations (red arrows and numbers) point to the following elements:

- 1) Controls the Debugger: Points to the top toolbar icons.
- 2) Breakpoint and next line: Points to the breakpoint icon and the next line of code.
- 3) vars at the moment of the stop: Points to the 'VARIABLEN' panel.
- 4) vars to watch: Points to the 'AUFRUFLISTE' panel.
- 5) all breakpoints: Points to the 'HALTEPUNKTE' panel.

Code snippet from the main editor:

```
tasks > week_five > main.py > ...
9 # Add some text to the figure
10 fig.suptitle('Explicit Interface Example')
11 ax.set_xlabel('x label')
12 ax.set_ylabel('y label')
13 fig.legend(['A simple line'])
14
15 fig.savefig("/workspaces/python_course/slides/five/figures/example_2.pdf")
16 # Add another line and a second legend
17 ax.plot([1, 2, 3, 4], [1, 3, 1, 4], 'r--')
18 ax.legend(['A simple line', 'Another line'])
19 # And save the figure
20 fig.savefig("/workspaces/python_course/slides/five/figures/myplot.pdf")
21
22
23
```

Terminal output:

```
source /workspaces/python_course/tasks/week_five/venv/bin/activate
@lannusch → /workspaces/python_course (dev) $ source /workspaces/python_course/tasks/week_five/venv/bin/activate
(venv) @lannusch → /workspaces/python_course (dev) $ /usr/bin/env /workspaces/python_course/tasks/week_five/venv/bin/python /home/vscode
599 -- /workspaces/python_course/tasks/week_five/main.py
[]
```

Debugging - Example

Task

- Advent of Code again (now with debugger)
- Try to plot some of the data

Task - Matplotlib

1. Rebuilt the container
2. Create the environment

```
$ cd /workspaces/python_course/tasks/week_five  
$ python3 -m venv venv  
$ source venv/bin/activate
```

3. Install the requirements

```
$ pip install matplotlib
```

4. Set the interpreter in VSCode

Task - Rebuild the container

The screenshot shows the VS Code interface with a LaTeX Beamer presentation open. The Explorer on the left shows the project structure, with the 'week_five' folder highlighted. The file list on the right shows 'debugging.tex' selected. A context menu is open over 'debugging.tex', showing options like 'Rebuild Container' and 'Add Dev Container Configuration Files...'. The presentation content includes a list of steps: 1. Rebuild the container, 2. Create the environment, 3. Install the requirements.

```
slides > week_five > debugging.tex
183
184 \begin{frame}
185 \begin{itemize}
186 \item Add Dev Container Configuration Files...
187 \item Stop Current Codespace
188 \item My Codespaces
189 \item Open in VS Code Desktop
190 \item Open in VS Code Insider's Desktop
191 \item Weiterarbeiten in neuen lokalen Klon
192 \item Continue Working in vscode.dev
193 \item "Visual Studio Code" herunterladen
194 \end{itemize}
195 \end{frame}
196 \begin{frame}
197 \textbf{2. Create the environment}
198 \begin{itemize}
199 \item $ cd /workspaces/python_course/tasks/week_five
200 \item $ python3 -m venv venv
201 \item $ source venv/bin/activate
202 \end{itemize}
203 \end{frame}
204 \begin{frame}
205 \textbf{3. Install the requirements}
206 \begin{itemize}
207 \item $ pip install matplotlib
208 \end{itemize}
209 \end{frame}
210 \end{frame}
```

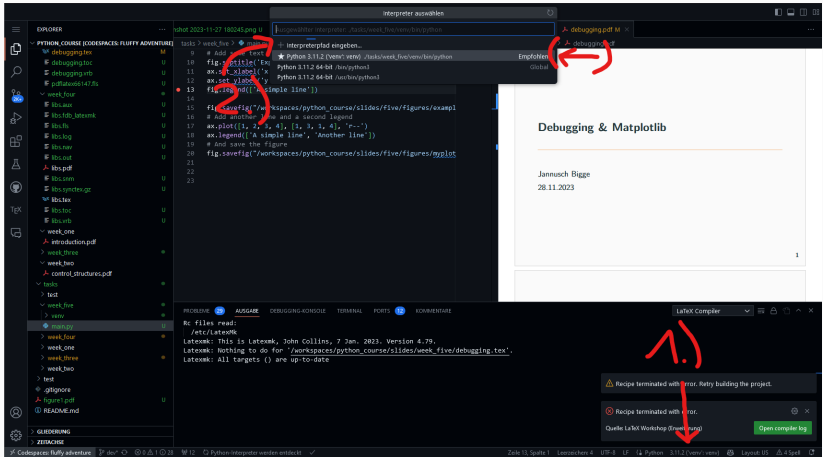
1. Rebuild the container

2. Create the environment

```
$ cd /workspaces/python_course/tasks/week_five
$ python3 -m venv venv
$ source venv/bin/activate
```

3. Install the requirements

Task - Set the interpreter



Maybe you have to navigate to the path (venv/bin/python). Best case VSCode finds it automatically.