

# HeFDI-Forschungsdatentag 24. Juni 2022

Track	Session	Referent*in(nen)
1. Datenmodellierung & Coding	1.2. Einstieg in die Programmierung, mit, insbesondere, Python	David Wallace (Universitäts- und Landesbibliothek Darmstadt)



## Abstract:

Das Warum und Wie eines Einstiegs ins Programmieren wird hier generell besprochen. Es gibt eine Vielfalt an Anwendungsmöglichkeiten und Technologien, um ins Programmieren einzusteigen. In dieser Session wird aber ein Beispiel der tabellierten Datenauswertung in Python – mithilfe eines interaktiven Interpreters – vorgetragen.

## Zu den Forschungsdatentagen:

Die HeFDI-Forschungsdatentage ermöglichen es Forschenden, Lehrenden und allen weiteren Interessierten, Themen und Angebote des Forschungsdatenmanagements (FDM) kennenzulernen und zu erproben. Zudem stellen verschiedene hessische Infrastruktur-Einrichtungen und Zentren ihre Dienste und Angebote vor. Die Forschungsdatentage sind ein Angebot der Landesinitiative HeFDI - Hessische Forschungsdateninfrastrukturen, welche vom Hessischen Ministerium für Wissenschaft und Kunst (HMWK) finanziert wird.

DOI: <https://doi.org/10.5281/zenodo.6984248>; Lizenzinformation: Creative Commons Attribution 4.0 International ([CC BY 4.0](#))



# Einstieg in die Programmierung, mit, insbesondere, Python

David Wallace (ULB, TU Darmstadt)

[david.wallace@tu-darmstadt.de](mailto:david.wallace@tu-darmstadt.de)

[github.com/MyPyDavid](https://github.com/MyPyDavid)

2022-06-24 Forschungsdatentag

Beispieldatensatz zur Präsentation: <https://github.com/MyPyDavid/HeFDI-Forschungsdatentag-2022>



# Themen der Sessions<sup>1</sup>

1. Kurzvorstellung
2. Warum überhaupt Programmieren lernen? <sup>1</sup>
3. Warum Python?
4. Was macht den Einstieg so schwierig? <sup>1</sup>
5. Python und der Interpreter
6. Lernpfade und Strategien<sup>1</sup>
7. Fragen und Erfahrungsaustausch<sup>1</sup>

[1] Vielen Dank an Andreas Schieberle von der Hochschule Darmstadt für die Themen und Inhalte der Session.



# Kurzvorstellung : an der ULB

Projektmitarbeiter im Team Forschungsdaten-Services in der Abteilung Informationstechnologie, Forschung und Entwicklung (ITF&E) an der ULB.

„Das Team entwickelt die bestehenden Services weiter und entwickelt neue, auch im Verbund auf Landesebene (HeFDI) und Bundesebene (NFDI).“



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT



Universitäts- und  
Landesbibliothek  
Darmstadt



# Kurzvorstellung: Arbeitsschwerpunkt

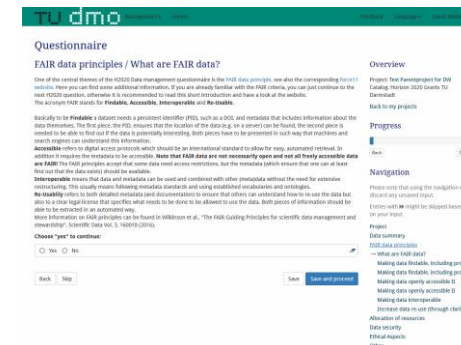
Web-Entwicklung mit



Open Source Software



Webseite oder Instanz



Projekt Management, Code  
Versionierung und Deployment



„Mit dem Research Data Management  
Organiser (RDMO) können Institutionen und  
Forschende das  
Forschungsdatenmanagement ihre Projekte  
strukturiert planen und durchführen.“

# Warum Programmieren lernen?

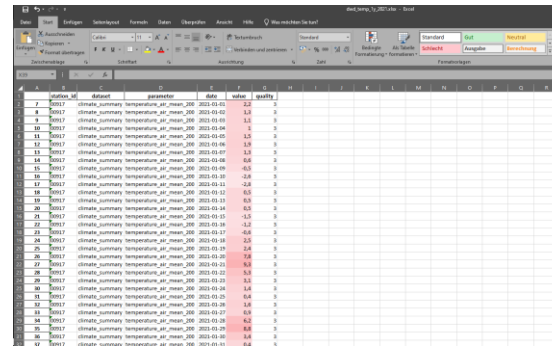
Manchmal braucht man **maßgeschneiderte Werkzeuge** für die **Datenverarbeitung** oder für den **Datenverarbeitungsprozess**.

Mit einer Programmiersprache kann man, unter anderem, die Notwendigkeiten **umsetzen** und die Werkzeuge **erzeugen**.



# Werkzeugen für Datenverarbeitung

Meistgenutzte Werkzeugen sind zB. Excel Spreadsheets und ähnliches.



Station_id	parameter	date	value	quality
1	climate_journary temperature_at_mess_200	2023-01-01	1.5	1
2	climate_journary temperature_at_mess_200	2023-01-02	1.5	1
3	climate_journary temperature_at_mess_200	2023-01-03	1.5	1
4	climate_journary temperature_at_mess_200	2023-01-04	1.5	1
5	climate_journary temperature_at_mess_200	2023-01-05	1.5	1
6	climate_journary temperature_at_mess_200	2023-01-06	1.5	1
7	climate_journary temperature_at_mess_200	2023-01-07	1.5	1
8	climate_journary temperature_at_mess_200	2023-01-08	1.5	1
9	climate_journary temperature_at_mess_200	2023-01-09	1.5	1
10	climate_journary temperature_at_mess_200	2023-01-10	1.5	1
11	climate_journary temperature_at_mess_200	2023-01-11	1.5	1
12	climate_journary temperature_at_mess_200	2023-01-12	1.5	1
13	climate_journary temperature_at_mess_200	2023-01-13	1.5	1
14	climate_journary temperature_at_mess_200	2023-01-14	1.5	1
15	climate_journary temperature_at_mess_200	2023-01-15	1.5	1
16	climate_journary temperature_at_mess_200	2023-01-16	1.5	1
17	climate_journary temperature_at_mess_200	2023-01-17	1.5	1
18	climate_journary temperature_at_mess_200	2023-01-18	1.5	1
19	climate_journary temperature_at_mess_200	2023-01-19	1.5	1
20	climate_journary temperature_at_mess_200	2023-01-20	1.5	1
21	climate_journary temperature_at_mess_200	2023-01-21	1.5	1
22	climate_journary temperature_at_mess_200	2023-01-22	1.5	1
23	climate_journary temperature_at_mess_200	2023-01-23	1.5	1
24	climate_journary temperature_at_mess_200	2023-01-24	1.5	1
25	climate_journary temperature_at_mess_200	2023-01-25	1.5	1
26	climate_journary temperature_at_mess_200	2023-01-26	1.5	1
27	climate_journary temperature_at_mess_200	2023-01-27	1.5	1
28	climate_journary temperature_at_mess_200	2023-01-28	1.5	1
29	climate_journary temperature_at_mess_200	2023-01-29	1.5	1
30	climate_journary temperature_at_mess_200	2023-01-30	1.5	1
31	climate_journary temperature_at_mess_200	2023-01-31	1.5	1
32	climate_journary temperature_at_mess_200	2023-02-01	1.5	1
33	climate_journary temperature_at_mess_200	2023-02-02	1.5	1
34	climate_journary temperature_at_mess_200	2023-02-03	1.5	1
35	climate_journary temperature_at_mess_200	2023-02-04	1.5	1
36	climate_journary temperature_at_mess_200	2023-02-05	1.5	1
37	climate_journary temperature_at_mess_200	2023-02-06	1.5	1
38	climate_journary temperature_at_mess_200	2023-02-07	1.5	1
39	climate_journary temperature_at_mess_200	2023-02-08	1.5	1
40	climate_journary temperature_at_mess_200	2023-02-09	1.5	1
41	climate_journary temperature_at_mess_200	2023-02-10	1.5	1
42	climate_journary temperature_at_mess_200	2023-02-11	1.5	1
43	climate_journary temperature_at_mess_200	2023-02-12	1.5	1
44	climate_journary temperature_at_mess_200	2023-02-13	1.5	1
45	climate_journary temperature_at_mess_200	2023-02-14	1.5	1
46	climate_journary temperature_at_mess_200	2023-02-15	1.5	1
47	climate_journary temperature_at_mess_200	2023-02-16	1.5	1
48	climate_journary temperature_at_mess_200	2023-02-17	1.5	1
49	climate_journary temperature_at_mess_200	2023-02-18	1.5	1
50	climate_journary temperature_at_mess_200	2023-02-19	1.5	1
51	climate_journary temperature_at_mess_200	2023-02-20	1.5	1
52	climate_journary temperature_at_mess_200	2023-02-21	1.5	1
53	climate_journary temperature_at_mess_200	2023-02-22	1.5	1
54	climate_journary temperature_at_mess_200	2023-02-23	1.5	1
55	climate_journary temperature_at_mess_200	2023-02-24	1.5	1
56	climate_journary temperature_at_mess_200	2023-02-25	1.5	1
57	climate_journary temperature_at_mess_200	2023-02-26	1.5	1
58	climate_journary temperature_at_mess_200	2023-02-27	1.5	1
59	climate_journary temperature_at_mess_200	2023-02-28	1.5	1
60	climate_journary temperature_at_mess_200	2023-02-29	1.5	1
61	climate_journary temperature_at_mess_200	2023-03-01	1.5	1

**Grafische Oberfläche (GUI)**

Inhalt einer Zelle:  
**Daten** mit “Datentypen”  
wie Zahlen, Text, ..

**Verarbeitungsmethoden:**  
Editieren, Selektieren,  
Formeln wie SUM,  
Diagramme, etc..



# Wo stoßen Spreadsheets an ihre Grenzen?

- Datenquelle/eingabe ohne Validierung
- „Kreative“ Gestaltung
- Farbe als Informationselement
- Automatisierung und Weiterverarbeitung
- Datums- und Zeitangaben
- Komplexere Datentypen
- Versionierung und Nachvollziehbarkeit
- Mehrbenutzerfähigkeit





# Grenzen verlegen mit Programmieren?

Mit Kenntnissen in einer Programmiersprache können Sie:

- Manuelle, repetitive Arbeitsabläufe **automatisieren**
- Größere Datenmengen **verarbeiten**, transformieren, validieren und aggregieren
- Arbeitsabläufe transparent und **nachvollziehbar dokumentieren**
- Und vieles mehr ...

Bonus:

- Wachsendes **Verständnis** für Datenmodelle, IT-Systeme und die Menschen die hauptberuflich damit arbeiten
- **Jobs**, bei viele Projekte mangelt es an Entwickler



# Vorteile im Forschungsdatenmanagement

- Einbindung an den FAIR Data Prinzipien
  - Daten und Metadaten findbar machen, maschinenlesbare Dokumente erzeugen.
  - Integrierbar mit anderen Datensätze oder Werkzeugen
  - Automatisierung macht ihre wissenschaftliche Arbeit reproduzierbar und wieder verwendbar
- Code macht Ihr Vorgehen (Workflow) transparent und nachvollziehbar für andere (und einen selbst).
- Manuelle Datenbearbeitung wird reduziert: mehr Zeit für Methodik.
- Versionierung und Vermeidung von Datenverlust.



# Warum Python?



- Universelle Programmiersprache für alle gängigen Betriebssysteme
- Gut lesbare Syntax, einsteigerfreundlich und übersichtlich
- Große Verbreitung in Wissenschaft, Industrie und Open Source Communities
- Umfangreiche Sammlung an ausgereiften Softwarepaketen
- Anwendungsgebiete in der Wissenschaft:
  - Numerische Berechnungen und Statistik
  - Visualisierung
  - Natural Language Processing
  - Maschinelles Lernen und Deep Learning



# Anwendungsfelder für den Einstieg

## **Scripting**

Einfache Computerprogramme zur Datenverarbeitung, Workflow, Schritt-für-Schritt

## **Visualisierung**

Erstellung von Grafiken und Diagrammen aus Daten

## **Entdeckungsreise**

Browsen über GitHub Projekte, Tutorials, Blogs oder Guides folgen.

## **Webentwicklung**

Kleine Web-Anwendungen mit Datenbank und Benutzeroberfläche

## **Web Scraping**

Daten von Webseiten extrahieren, speichern und verarbeiten

## **Bots**

Kleine Programme, die auf Mitteilungen reagieren und mit denen man interagieren kann (z.B. Chat, Twitter)



# Anwendungsfelder für den Einstieg: Python Pakete

## Scripting

[pandas](#), [pathlib](#), [python-dotenv](#)

## Visualisierung

[matplotlib](#), [seaborn](#), [Altair](#), [Dash](#)

## Entdeckungsreise

[PyPI](#), [GitHub](#), [Awesome Python](#)

## Webentwicklung

[Django](#), [Flask](#), [FastAPI](#)

## Web Scraping

[Beautiful Soup](#) , [scrapy](#)

## Bots

[Tweepy](#)

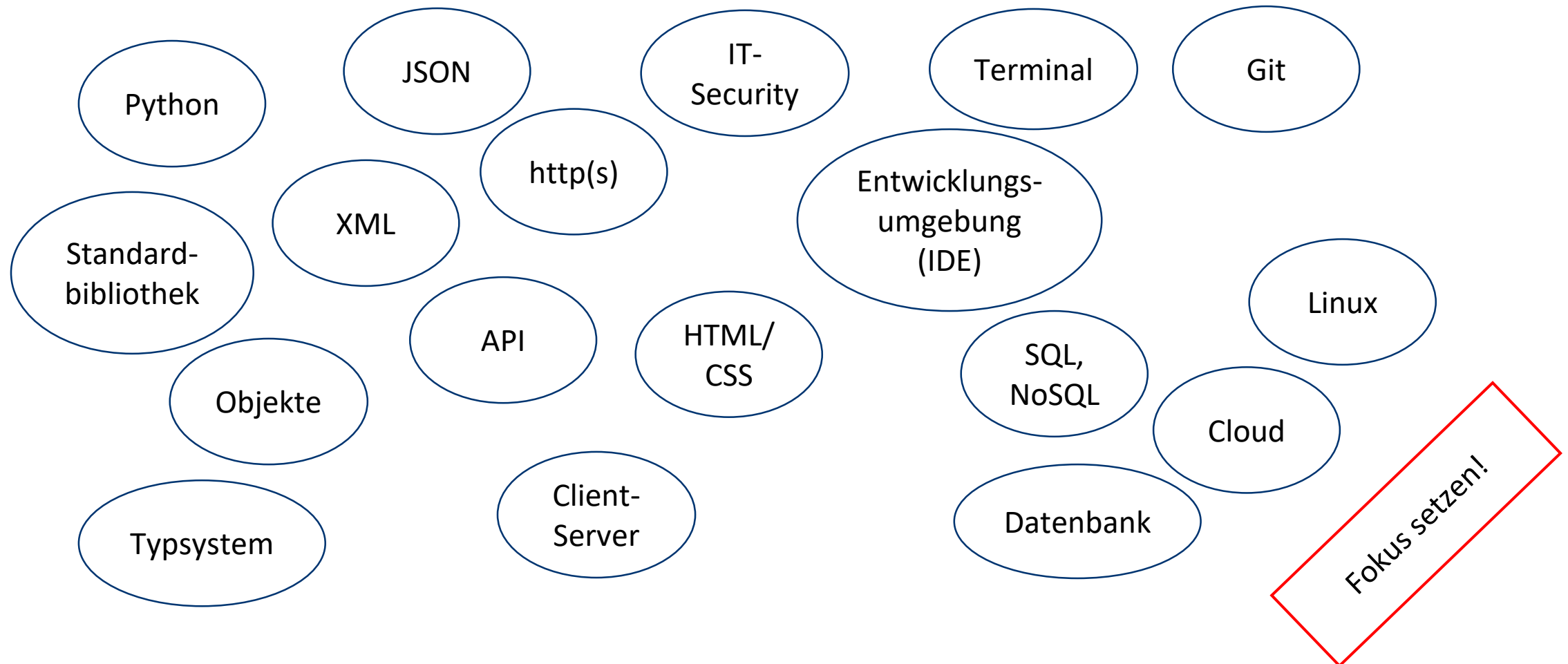


# Was macht den Einstieg so schwierig?

- Eine Programmiersprache zu lernen ist einfach, Programmieren lernen ist schwer! (Lernkurve)
- Das Übersetzen des mentalen Modells in ein Programm benötigt viel Übung
- Programmiersprachen und Werkzeuge sind für professionelle Softwareentwicklung gemacht (mehr CLI, weniger GUI)
- Lernressourcen von unterschiedlicher Qualität, Tiefe, Breite und oft ohne konkreten Anwendungsbezug
- Sehr heterogene Communities in Online-Foren z.B. auf Stackoverflow, Reddit
- Vielzahl an ineinandergreifenden Technologien. Standards, Frameworks, APIs



# Vielzahl an ineinandergreifenden Technologien



# Wo soll ich anfangen?

- Persönliches Lernziel definieren
  - Konkrete Anwendung im Berufskontext oder Hobby Projekt?
  - Auf welchen Vorkenntnissen kann ich aufbauen?
  - Was motiviert mich?
- Geeignete Lernressourcen für den eigenen Lerntyp finden
  - Bücher?
  - Videos?
  - Strukturierte Kurse?
  - Hands-on?

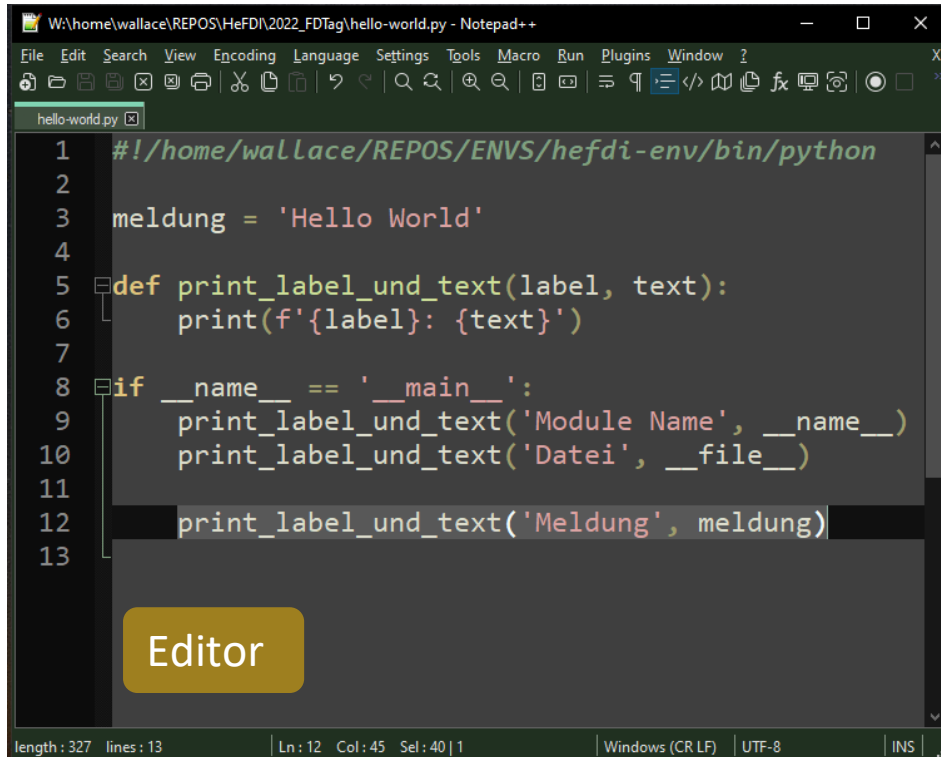




# Python und der Interpreter: Anfang

Eine .py Datei

Notepad++



The screenshot shows the Notepad++ text editor with a file named 'hello-world.py' open. The code is as follows:

```
1 #!/home/wallace/REPOS/ENV/HEFDI/bin/python
2
3 meldung = 'Hello World'
4
5 def print_label_und_text(label, text):
6     print(f'{label}: {text}')
7
8 if __name__ == '__main__':
9     print_label_und_text('Module Name', __name__)
10    print_label_und_text('Datei', __file__)
11
12    print_label_und_text('Meldung', meldung)
13
```

A yellow box with the word 'Editor' is overlaid on the bottom left of the code area. The status bar at the bottom indicates 'length: 327 lines: 13', 'Ln: 12 Col: 45 Sel: 40 | 1', 'Windows (CR LF)', 'UTF-8', and 'INS'.

Wie kann ich coden wenn da nur Text steht?

Für Python gibt es eine interaktive Computerprogrammierungsumgebung (Interpreter) womit man einfach den Code “ausprobieren” kann.

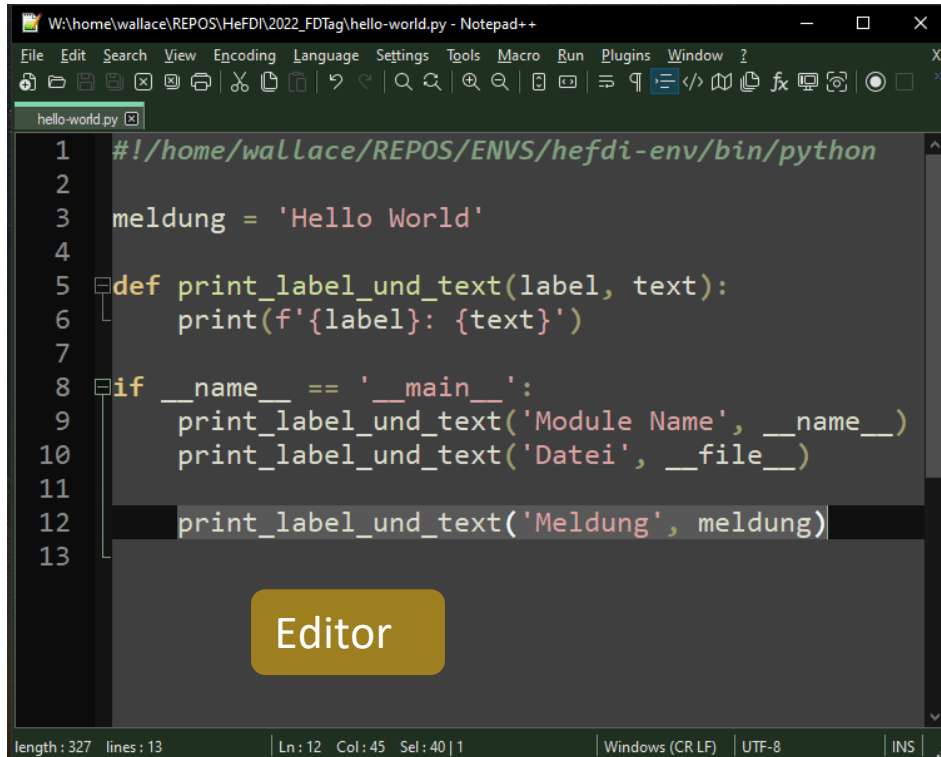
Damit bekommt man eine “Read-Eval-Print-Schleife” als herangehensweise.



# Python und der Interpreter: Entwicklungsumgebung (IDE)

Grundlage für eine IDE

## Notepad++

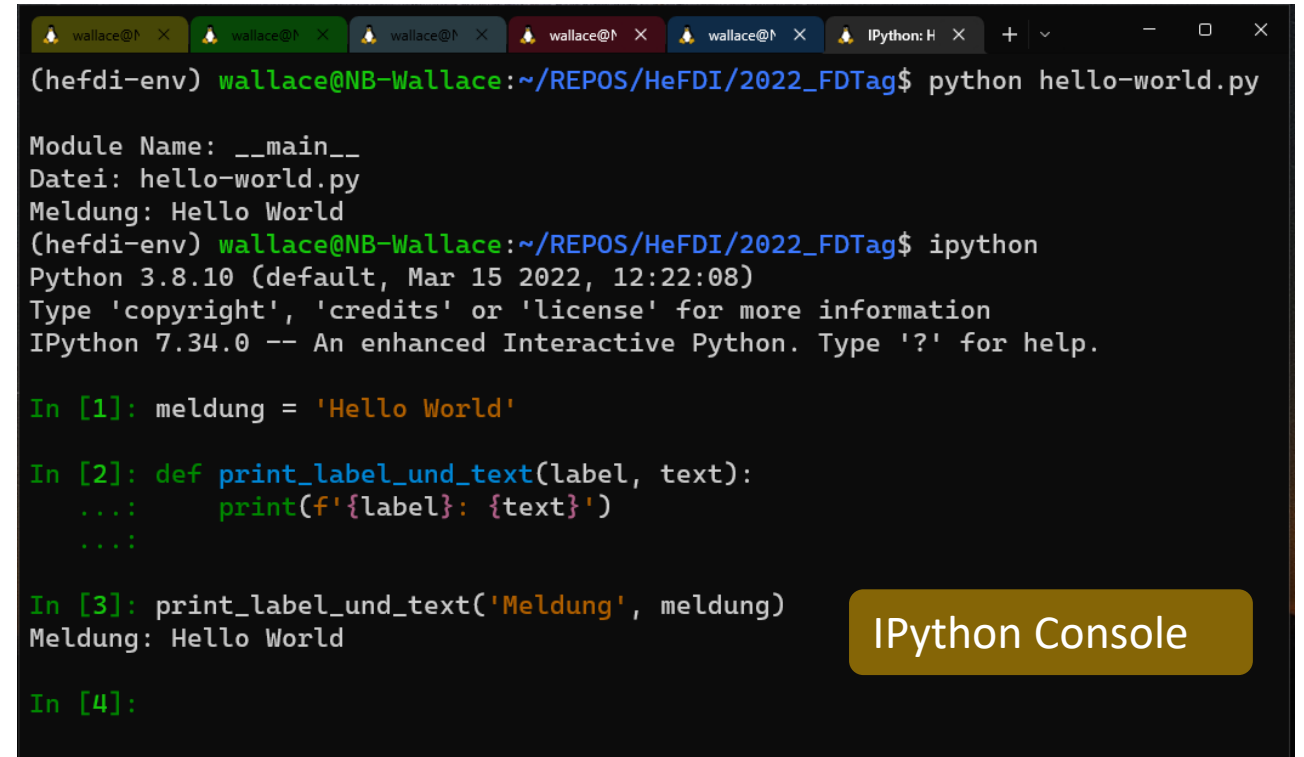


The screenshot shows the Notepad++ text editor with a file named 'hello-world.py'. The code is as follows:

```
1 #!/home/wallace/REPOS/ENV/HEFDI/bin/python
2
3 meldung = 'Hello World'
4
5 def print_label_und_text(label, text):
6     print(f'{label}: {text}')
7
8 if __name__ == '__main__':
9     print_label_und_text('Module Name', __name__)
10    print_label_und_text('Datei', __file__)
11
12    print_label_und_text('Meldung', meldung)
13
```

A yellow button labeled 'Editor' is positioned at the bottom center of the editor window.

## Interpreter (IPython Console) oder Shell



The screenshot shows the IPython console with the following output:

```
(hefdi-env) wallace@NB-Wallace:~/REPOS/HeFDI/2022_FDTag$ python hello-world.py
Module Name: __main__
Datei: hello-world.py
Meldung: Hello World
(hefdi-env) wallace@NB-Wallace:~/REPOS/HeFDI/2022_FDTag$ ipython
Python 3.8.10 (default, Mar 15 2022, 12:22:08)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.34.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: meldung = 'Hello World'

In [2]: def print_label_und_text(label, text):
...:     print(f'{label}: {text}')
...:

In [3]: print_label_und_text('Meldung', meldung)
Meldung: Hello World

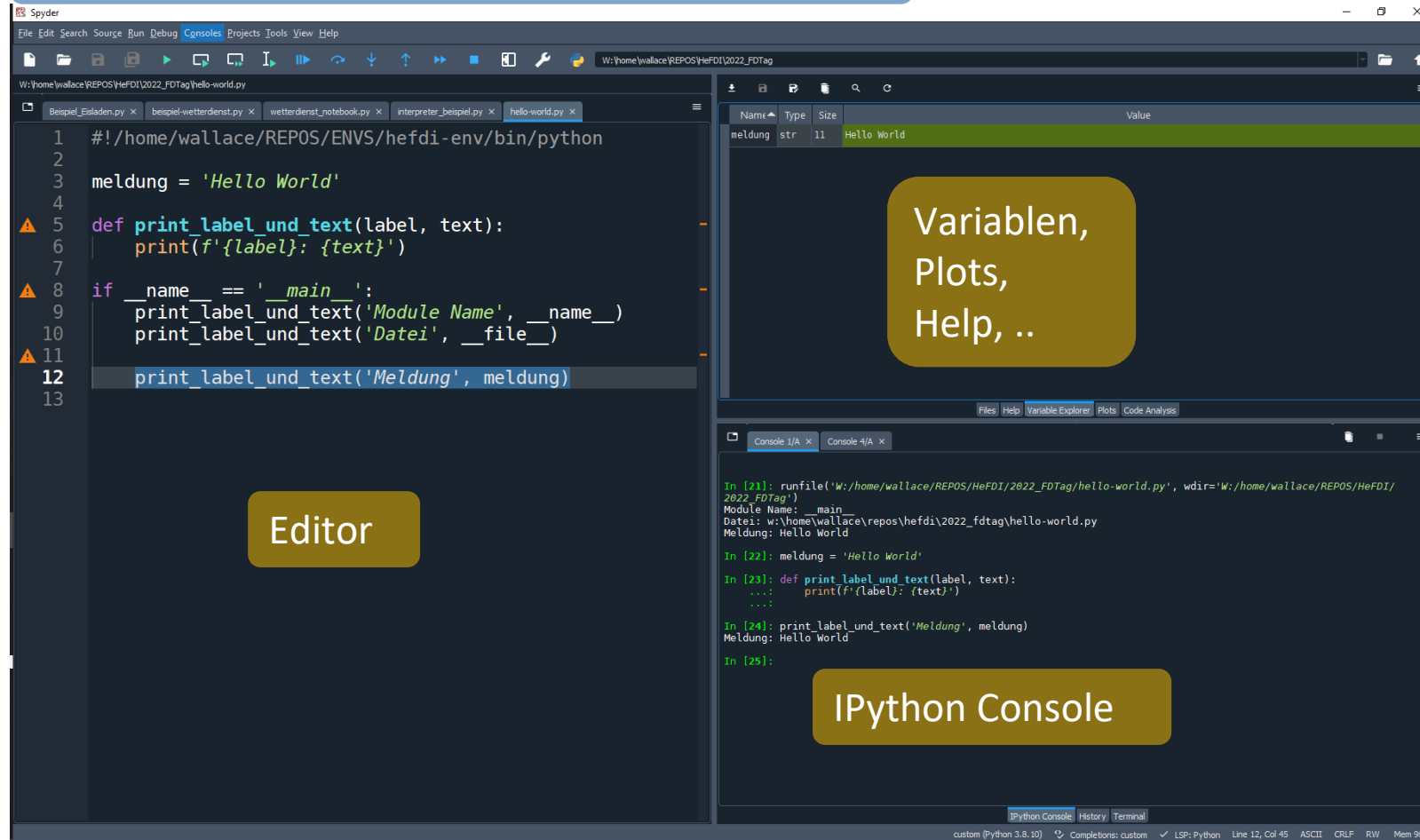
In [4]:
```

A yellow button labeled 'IPython Console' is positioned at the bottom right of the console window.



# Python und der Interpreter: Entwicklungsumgebung

IDE für Wissenschaftlicher und Forschungsdaten



## Spyder

Open source

Für und mit Python

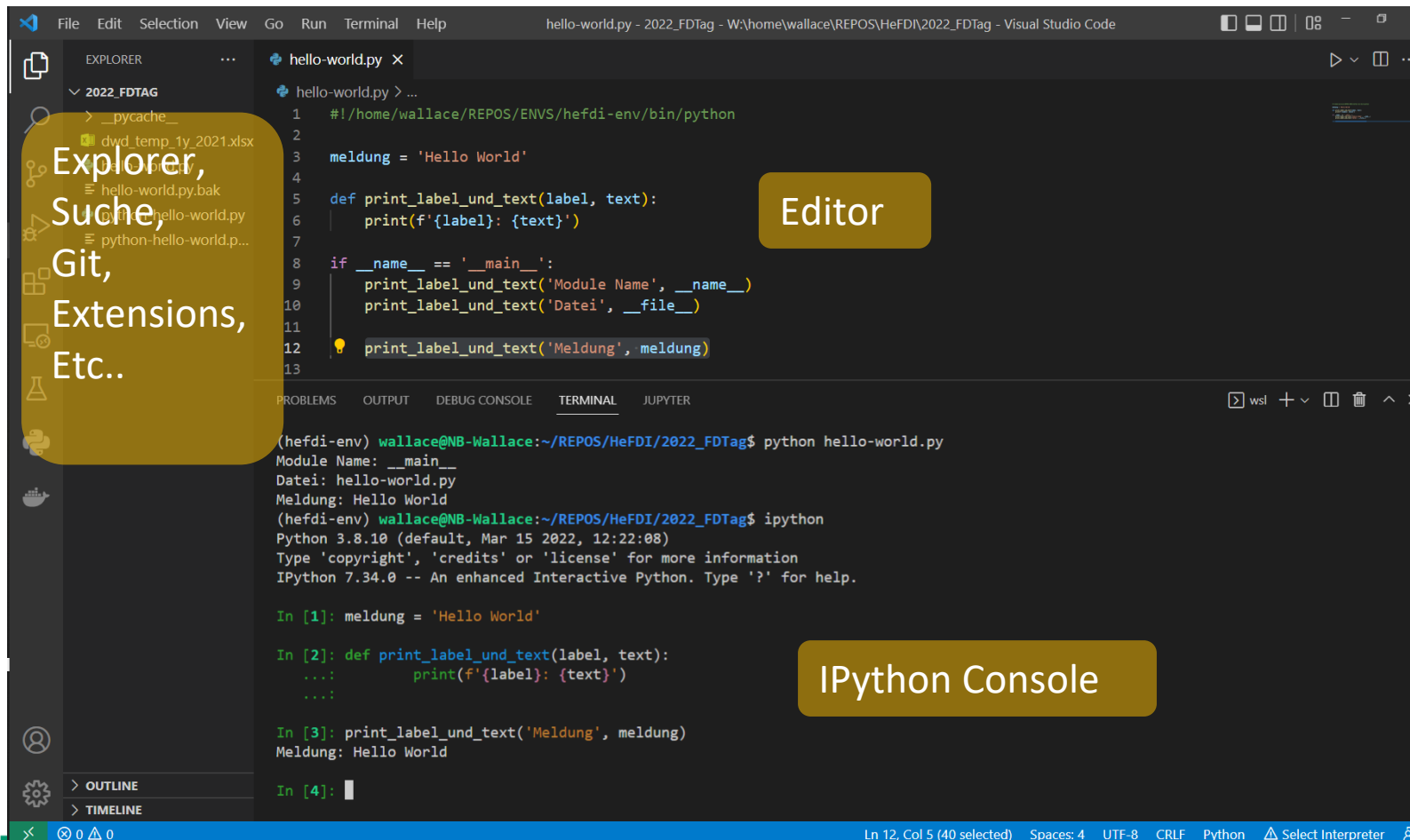
Plots und Variablen  
sind direkt sichtbar

Passend für die  
Arbeit mit FD



# Python und der Interpreter: Entwicklungsumgebung

IDE für die Webentwicklung oder Software Engineering



## Visual Studio Code

Open source aber  
Produkt von Microsoft

Unterstützung für  
viele andere Sprachen

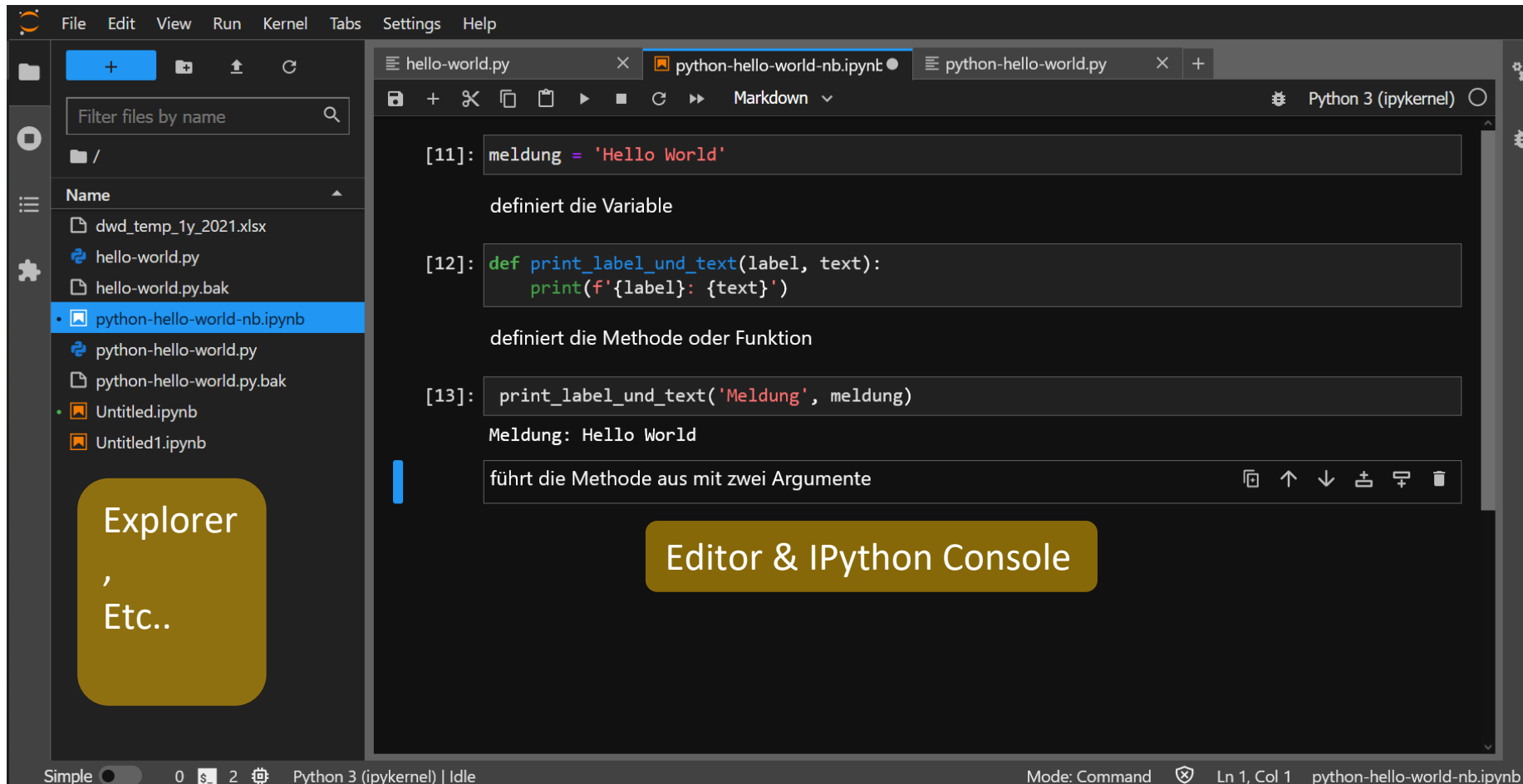
GUI für Git

Viele Extensions

Passend für  
complexere Projekte

# Python und der Interpreter: Entwicklungsumgebung

IDE in dem Browser für Wissenschaftlicher und Forschungsdaten



## JupyterLab oder Notebook

Open source  
Integriert Editor und Console  
Unterstützung für mehrere Sprachen  
Läuft im Browser, wurde als Webservice angeboten

# Live Coding Beispiele?

Mitschreiben auf zum Beispiel: <https://jupyter.org/try-jupyter/lab/>



# Live Coding Beispiele?

## Datentypen Beispiel

```
[2]: a = '1'  
     b = '2'
```

```
[18]: c = a + b  
      c_int = int(a) + int(b)  
      print(f'c = a + b = {c}')
```

```
      print(f'c_int = int(a) + int(b) = {c_int}')
```

c = a + b = 12  
c\_int = int(a) + int(b) = 3

```
[19]: try:  
      d = c + 5  
      except TypeError as exc:  
          print(f'Exception gefangen: {exc}')          d = c + str(5)  
  
      print(f'd = {d}')
```

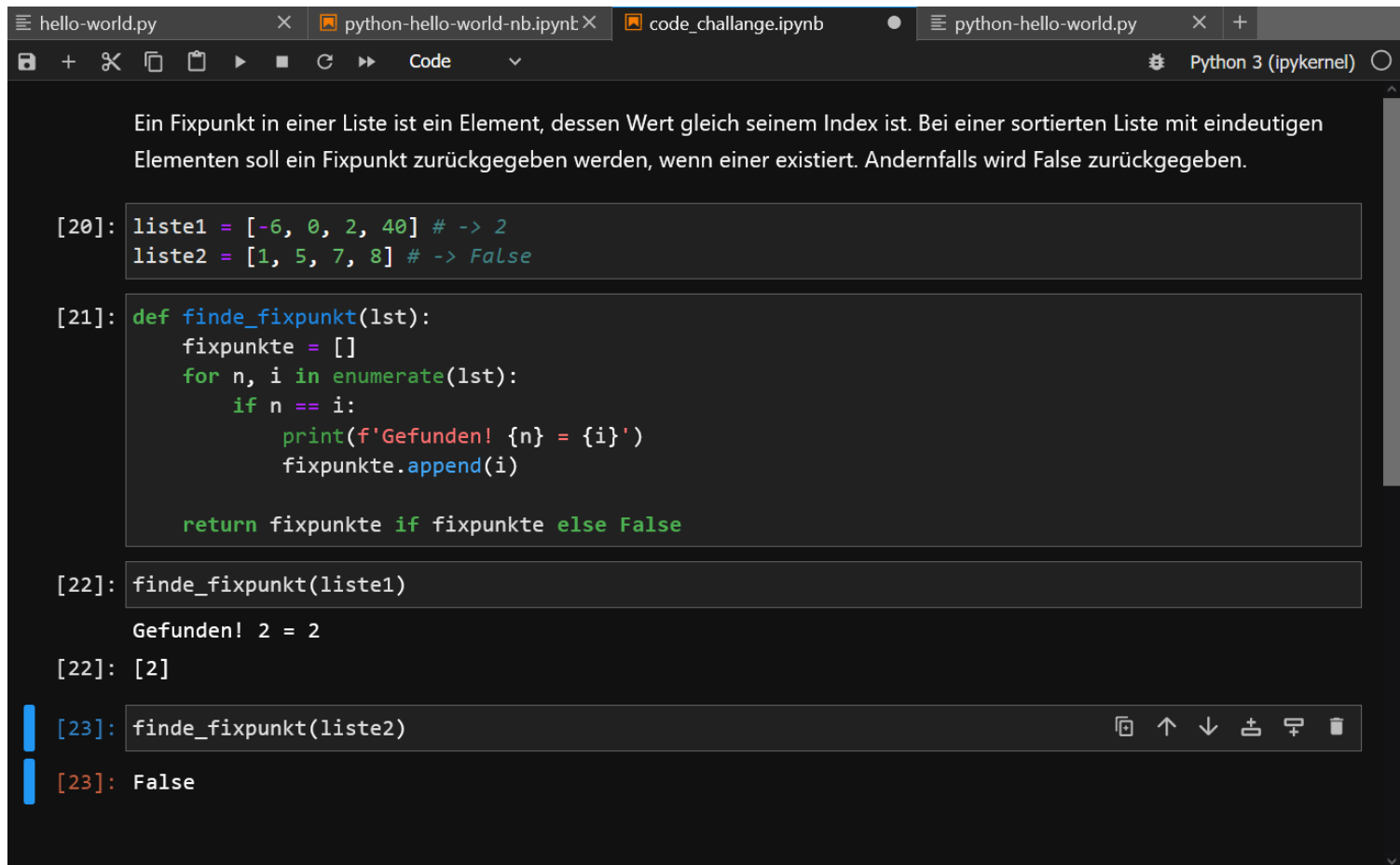
Exception gefangen: can only concatenate str (not "int") to str  
d = 125

```
[ ]:
```



# Live Coding Beispiele?

## Code Challenges: Beispiel Lösung



The screenshot shows a Jupyter Notebook with three tabs: 'hello-world.py', 'python-hello-world-nb.ipynb', and 'code\_challenge.ipynb'. The active tab is 'code\_challenge.ipynb'. The notebook contains a text description of a 'Fixpunkt' problem, followed by three code cells. The first cell defines two lists, 'liste1' and 'liste2'. The second cell defines a function 'finde\_fixpunkt' that iterates through a list to find elements where the index equals the value. The third cell calls this function on 'liste1', resulting in the output 'Gefunden! 2 = 2'. The fourth cell calls the function on 'liste2', resulting in the output 'False'.

```
Ein Fixpunkt in einer Liste ist ein Element, dessen Wert gleich seinem Index ist. Bei einer sortierten Liste mit eindeutigen Elementen soll ein Fixpunkt zurückgegeben werden, wenn einer existiert. Andernfalls wird False zurückgegeben.
```

```
[20]: liste1 = [-6, 0, 2, 40] # -> 2
      liste2 = [1, 5, 7, 8] # -> False
```

```
[21]: def finde_fixpunkt(lst):
      fixpunkte = []
      for n, i in enumerate(lst):
          if n == i:
              print(f'Gefunden! {n} = {i}')
              fixpunkte.append(i)

      return fixpunkte if fixpunkte else False
```

```
[22]: finde_fixpunkt(liste1)
      Gefunden! 2 = 2
      [22]: [2]
```

```
[23]: finde_fixpunkt(liste2)
      [23]: False
```





# Lernpfade und Strategien

- Persönliches Lernziel definieren
- Basiskenntnisse aufbauen
  - Datentypen, Variablen, Funktionen
  - Schleifen und Verzweigungen
  - Klassen und Objekte
  - Verwendung der Standardbibliotheken
  - Import- und Exportformate (csv, json, ggf. xml)
- Fortgeschrittene Themen
  - Funktionen höherer Ordnung, lambda Funktionen, Dekoratoren
  - List- und Dict-Comprehension, map, reduce, filter

## Ressourcen

Youtube  
freecodecamp  
Coursera  
edX  
Udemy  
Bücher/eBooks



# Lernpfade und Strategien

## Projektbasiertes Lernen

- Definieren Sie für sich selbst ein motivierendes Projekt z.B.:
  - Webanwendung (To-Do Liste, Eigene Profilseite, Online-Shop)
  - Web Scraping, Suchmaschine
  - Natural Language Processing, Sentiment Analyse
  - Datenanalyse, Statistik
- Keine eigenen Daten? Schauen Sie mal auf Kaggle->
  - [www.kaggle.com](https://www.kaggle.com)

## Ressourcen

Youtube  
Stackoverflow  
Reddit  
#100DaysOfCode  
Github  
Real Python  
Medium Tutorials



# Lernpfade und Strategien

## Code Challenges

- Kleine Herausforderungen mit unterschiedlichem Schwierigkeitsgrad
- Trainiert die Problemlösungskompetenz und algorithmisches Denken

## Ressourcen

Codewars

Daily Coding  
Problems

Project Euler



# Fragen und Erfahrungsaustausch

Danke für Ihre Aufmerksamkeit

