



Tag 4

Debuggen & Module / Bibliotheken

10. März 2023

Ablauf

- Rückblick dritter Tag
- Übung besprechen
- Zwischenprüfung (60 Minuten)
- Debuggen und loggen
- Eigene Module erstellen

Rückblick

Dritter Tag

Debuggen

- Nicht nur Bugs finden



- Auch anderer Code verstehen
- Logging

Logging

- Konfigurierbar, daher print() vorzuziehen.
 - z. B. Level DEBUG zu erstellen, danach WARNING für den produktiven Code
 - So können die loggenden Einträge belassen werden und bei Bedarf wieder aktiviert werden.
 - Neben Level kann auch Format, log to file, uvm. [konfiguriert](#) werden.
- Siehe *c1_logging_demo.py*
- **Logging.basicConfig() mit level=DEBUG** nicht vergessen!
- Levels:

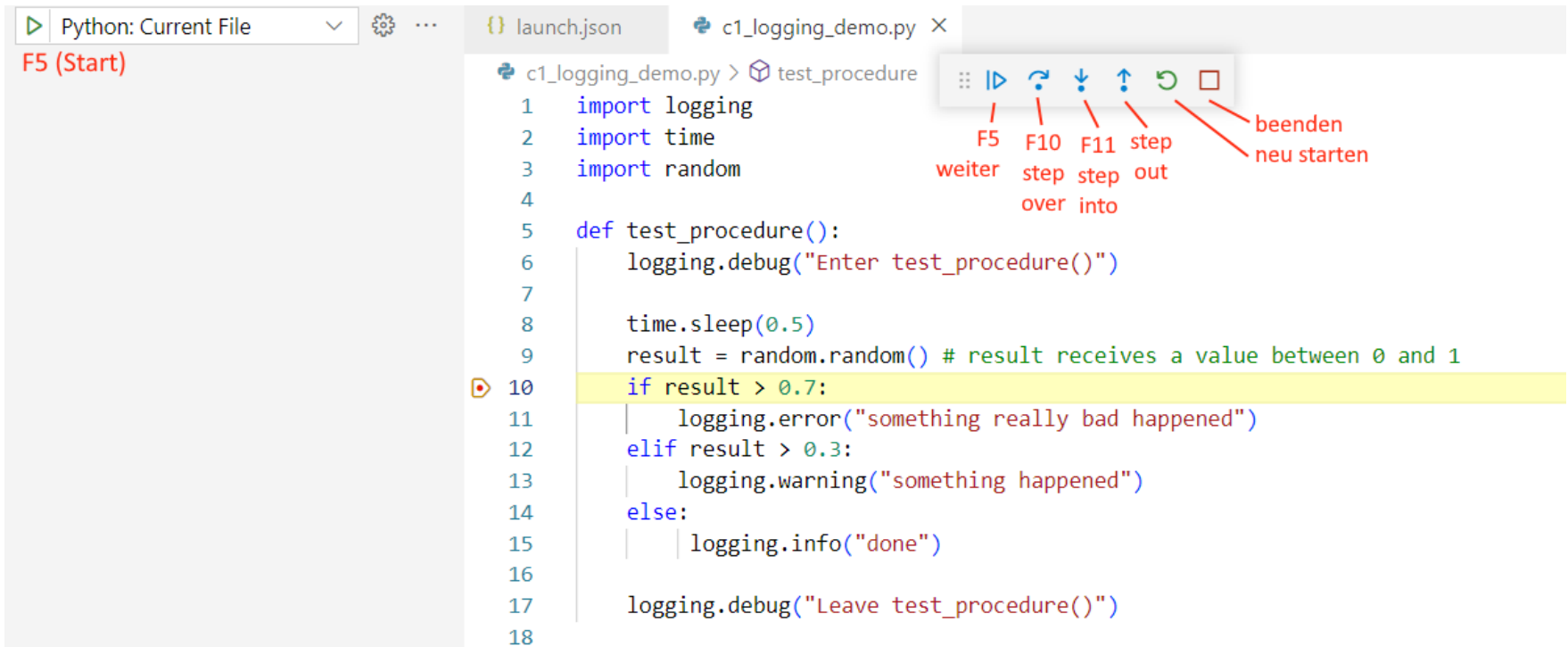
CRITICAL	sehr schlimm
ERROR	schlimm
WARNING	potentiell schlimm
INFO	eventuell interessant im produktiven Umfeld
DEBUG	eventuell interessant während der Entwicklung
- Die Wahl des richtigen Loglevels ist sehr schwierig, daher hier nur diese saloppen Vorgaben.
 - Zu niedrig → log wird geflutet, Wichtiges geht im rauschen unter
 - Zu hoch → potentiell hilfreiche Nachricht zur Entstehung des Fehlers nicht geloggt

Debuggen

- Debuggen ist der kunstvolle Einsatz von **Breakpoints** und **Einzelschrittausführung**
- Schnell müssen auch externe Signale, Quellen und Ressourcen miteinbezogen werden
- Breakpoints können auch «bedingt» sein:
 - **Ausdruck**
 - Trefferanzahl
 - Protokollnachricht («temporäres Logging»)



Einzelsschritte



The screenshot shows a Python IDE interface. On the left, a sidebar displays 'Python: Current File' and 'F5 (Start)'. The main editor shows a file named 'c1_logging_demo.py' with the following code:

```
1 import logging
2 import time
3 import random
4
5 def test_procedure():
6     logging.debug("Enter test_procedure()")
7
8     time.sleep(0.5)
9     result = random.random() # result receives a value between 0 and 1
10    if result > 0.7:
11        logging.error("something really bad happened")
12    elif result > 0.3:
13        logging.warning("something happened")
14    else:
15        logging.info("done")
16
17    logging.debug("Leave test_procedure()")
18
```

A toolbar is visible above the code editor, containing icons for various debugging actions. Red arrows point from German labels to these icons:

- F5: weiter (continue)
- F10: step over (step over)
- F11: step into (step into)
- step out (step out)
- beenden (stop)
- neu starten (restart)

Step over: Die Methode wird als Ganzes ausgeführt

Step into: Es wird in die Methode gesprungen

Step out: Die aktuelle Methode wird bis zum Ende ausgeführt

import – Module verwenden

```
import math  
square_root = math.sqrt(49)
```

Einfacher, direkter import des gesamten Moduls, hier math.

Alle Verwendungen müssen mit dem Modulnamen beginnen:

```
math.sqrt()  
math.pi
```


import – Module verwenden

```
import math as m  
sinus = m.sin(3.14)
```

Import mit aliasing (<umbenennen>)

Alle Verwendungen müssen mit dem Alias beginnen:

```
m.sqrt()  
m.pi
```

import – Module verwenden

```
from math import pi, cos  
cosine = cos(pi)
```

Import von einzelnen Funktionen,
Variablen oder Klassen

Kein Präfix mehr notwendig

```
cos()  
pi
```

Bequem

Standard bei Klassen

Eigene Module erstellen

- Einzelne Datei
- Mehrere Dateien einzeln
- Mehrere Dateien zusammen
(`__init__.py`, nicht behandelt hier)
- Die ersten zwei Ansätze brauchen keinen speziellen Syntax

Eigenes Modul

a2_my_module.py:

```
the_answer = 42
```

```
def my_adding_function(a, b):  
    return a + b
```

Verwendung

a3_import_of_my_module.py:

```
import a2_my_module
```

```
print(f"variable: {a2_my_module.the_answer}")
```

```
print(f"function: {a2_my_module.my_adding_function(1, 1)}")
```

Fragen?

Vorschau Übung

Kommando-Interpreter

- 1 - ask for an amount of days and calculate and print the date from now (now + days)
- 2 - ask for an amount of days and calculate and print when that date was (now - days)
- 3 - ask for a date and an amount of days to calculate and print a second date
- 4 - ask for two dates and calculate and print the time delta in days

- 10 - print current directory
- 11 - print items of current directory ('ls'/ 'dir')
- 12 - move one directory up ('cd ..')
- 13 - ask for a sub-directory name and go into ('cd xyz')

- 30 - print computer name
- 31 - print operating system
- 32 - print platform info
- 33 - print Python version

- ? - print this instructions
- e - exit program

Gruppenarbeit

- Gruppe 1
datetime w3schools docs.python
- Gruppe 2
os (w3schools) docs.python
- Gruppe 3
platform docs.python
math w3schools docs.python

Ziele

- Kurzvortrag, 5-10 Minuten
 - Überblick
 - Vorstellung einzelne Funktionen
- Handout (Pdf, Word, Jupyter) 1-2 Seiten