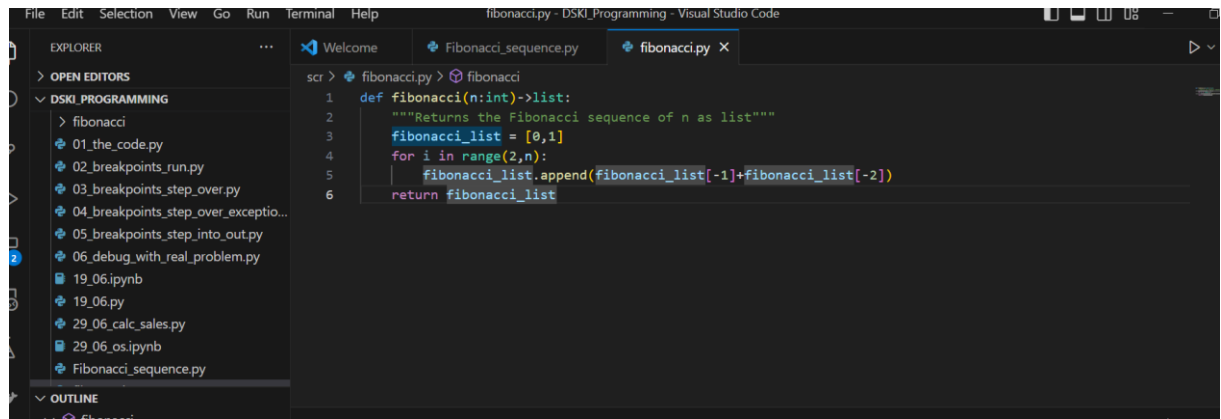


Eigene Module erstellen und in eine virtuelle Umgebung einladen (Windows)

1) Eigenes Modul erstellen I



```
1 def fibonacci(n:int)->list:
2     """Returns the Fibonacci sequence of n as list"""
3     fibonacci_list = [0,1]
4     for i in range(2,n):
5         fibonacci_list.append(fibonacci_list[-1]+fibonacci_list[-2])
6     return fibonacci_list
```

In Visual Studio Code (VSC) wurde die Beispieldatei fibonacci.py erstellt. In dieser Datei wird die Funktion fibonacci definiert. Die Funktion gibt die Fibonacci-Folge bis zum n-ten Element als Liste zurück.

2) Eigenes Modul erstellen II

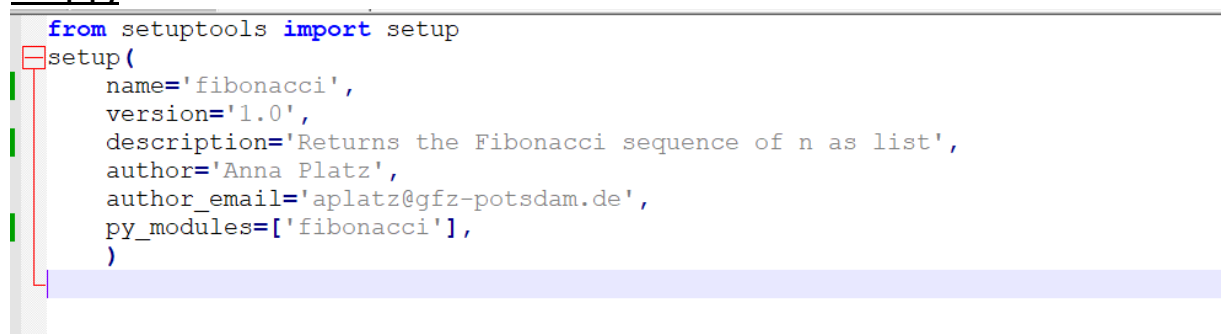
scr > fibonacci

fibonacci durchsuchen

Name	Änderungsdatum	Typ
 fibonacci	06.07.2023 08:13	Python-Quelle
 readme	23.06.2023 09:31	Textdokument
 setup	06.07.2023 08:15	Python-Quelle

Die Datei fibonacci.py wird in einen Unterordner gespeichert (hier mit den Namen fibonacci). Außerdem müssen in dem Ordner noch zwei weitere Dateien erstellt werden: readme.txt (kann erst einmal leer sein) und setup.py.

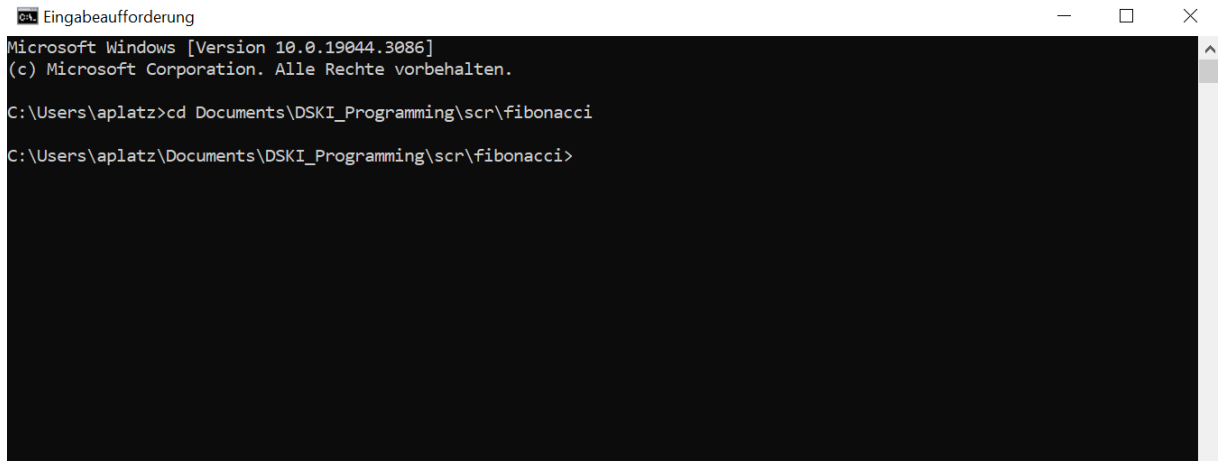
3) Setup.py



```
from setuptools import setup
setup(
    name='fibonacci',
    version='1.0',
    description='Returns the Fibonacci sequence of n as list',
    author='Anna Platz',
    author_email='aplatz@gfz-potsdam.de',
    py_modules=['fibonacci'],
)
```

Beispielcode für die Datei setup.py. Wichtig sind vor allem der Name und die geladenen Python-Module (in unserem Fall nur die Datei fibonacci.py).

4) Eigenes Modul erstellen III



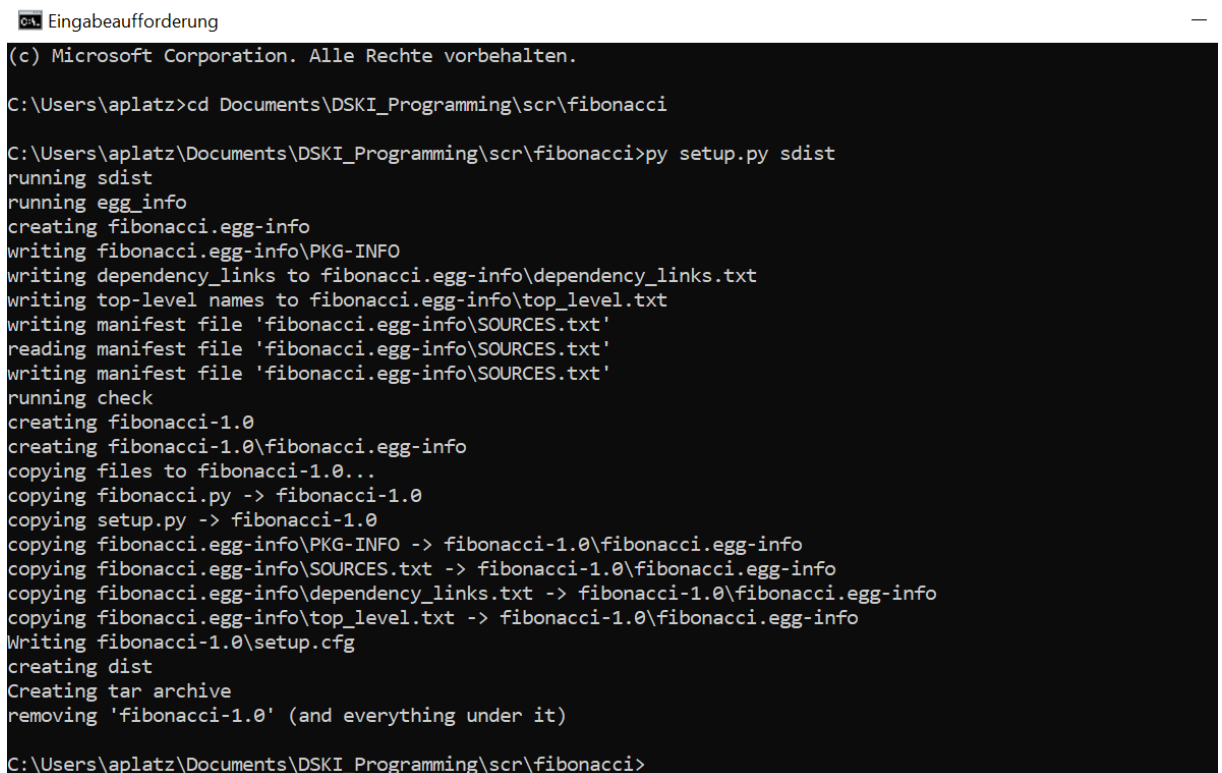
```
Microsoft Windows [Version 10.0.19044.3086]
(c) Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\aplatz>cd Documents\DSKI_Programming\scr\fibonacci

C:\Users\aplatz\Documents\DSKI_Programming\scr\fibonacci>
```

Wenn alle Dateien erstellt wurden, öffnen wir eine Windows-Eingabeaufforderung (CMD). Wir gehen mit `cd ...` in den erstellten Unterordner, in dem unsere Dateien (fibonacci.py, setup.py und readme.txt) liegen.

5) Eigenes Modul erstellen IV



```
(c) Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\aplatz>cd Documents\DSKI_Programming\scr\fibonacci

C:\Users\aplatz\Documents\DSKI_Programming\scr\fibonacci>py setup.py sdist
running sdist
running egg_info
creating fibonacci.egg-info
writing fibonacci.egg-info\PKG-INFO
writing dependency_links to fibonacci.egg-info\dependency_links.txt
writing top-level names to fibonacci.egg-info\top_level.txt
writing manifest file 'fibonacci.egg-info\SOURCES.txt'
reading manifest file 'fibonacci.egg-info\SOURCES.txt'
writing manifest file 'fibonacci.egg-info\SOURCES.txt'
running check
creating fibonacci-1.0
creating fibonacci-1.0\fibonacci.egg-info
copying files to fibonacci-1.0...
copying fibonacci.py -> fibonacci-1.0
copying setup.py -> fibonacci-1.0
copying fibonacci.egg-info\PKG-INFO -> fibonacci-1.0\fibonacci.egg-info
copying fibonacci.egg-info\SOURCES.txt -> fibonacci-1.0\fibonacci.egg-info
copying fibonacci.egg-info\dependency_links.txt -> fibonacci-1.0\fibonacci.egg-info
copying fibonacci.egg-info\top_level.txt -> fibonacci-1.0\fibonacci.egg-info
Writing fibonacci-1.0\setup.cfg
creating dist
Creating tar archive
removing 'fibonacci-1.0' (and everything under it)

C:\Users\aplatz\Documents\DSKI_Programming\scr\fibonacci>
```

Wir tippen den Befehl `py setup.py sdist` ein. Damit wird das Modul erstellt, siehe den neuen Ordner dist.

« scr » fibonacci

fibonacci durchsuchen

Name	Änderungsdatum	Typ
dist	06.07.2023 08:19	Dateiordner
fibonacci.egg-info	06.07.2023 08:19	Dateiordner
fibonacci	06.07.2023 08:13	Python-Quelldat
readme	23.06.2023 09:31	Textdokument
setup	06.07.2023 08:15	Python-Quelldat

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Verzeichnis: C:\Users\aplatz\Documents\DSKI_Programming\scr\fibonacci\dist

Mode                LastWriteTime         Length Name
-----
-a----             06.07.2023      08:19           893 fibonacci-1.0.tar.gz

(venv_dski) PS C:\Users\aplatz\Documents\DSKI_Programming\scr\fibonacci\dist> py -m pip install fibonacci
-1.0.tar.gz
Processing c:\users\aplatz\documents\dski_programming\scr\fibonacci\dist\fibonacci-1.0.tar.gz
Installing build dependencies ... done
Getting requirements to build wheel ... done
Preparing metadata (pyproject.toml) ... done
Building wheels for collected packages: fibonacci
  Building wheel for fibonacci (pyproject.toml) ... done
  Created wheel for fibonacci: filename=fibonacci-1.0-py3-none-any.whl size=1299 sha256=a92455cf10ddd40b9914d88
97a93d44801e05698b84be83df8676f72d3e5ba16
  Stored in directory: c:\users\aplatz\appdata\local\pip\cache\wheels\af\80\3f\ef04ba83e8511d112c38580e1b712f0d7
9846a57bcc8c2a3205
Successfully built fibonacci
Installing collected packages: fibonacci
Successfully installed fibonacci-1.0

(venv_dski) PS C:\Users\aplatz\Documents\DSKI_Programming\scr\fibonacci\dist>

```

Name	Änderungsdatum	Typ
executing-1.2.0.dist-info	19.06.2023 10:45	Dateiordner
fastjsonschema	19.06.2023 10:45	Dateiordner
fastjsonschema-2.17.1.dist-info	19.06.2023 10:45	Dateiordner
fibonacci-1.0.dist-info	06.07.2023 08:23	Dateiordner
fontTools	19.06.2023 10:46	Dateiordner
fonttools-4.40.0.dist-info	19.06.2023 10:46	Dateiordner

Danach kann das Modul beliebig in der virtuellen Umgebung genutzt werden für neue Programme oder direkt im Terminal, s.h. Beispiel.

```
(venv_dski) PS C:\Users\aplatz\Documents\DSKI_Programming> python
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun 7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import fibonacci as fib
>>> fib.fibonacci(10)
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
>>> █
```

Im Beispiel wird Python in der aktuellen virtuellen Umgebung direkt im Terminal aktiviert durch den Aufruf **python**. Danach wird das neu installierte Modul fibonacci als fib importiert. Danach wird mit Hilfe des Aufrufs fib.fibonacci(10) die Fibonacci-Folge bis zur 10. Stelle ausgegeben.