Fachhochschule Nordwestschweiz
Hochschule für Wirtschaft

AACSB ACCREDITED

DevOps

Mit Open Source

Roman Brun

Lokale Umsetzung



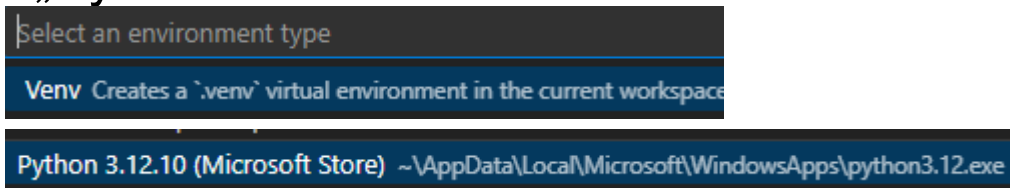
Voraussetzungen

- Grundkenntnisse in Python

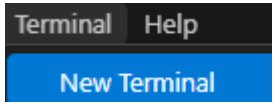
- Python installiert (oder Programmiersprache der Wahl)
- Visual Studio Code als Entwicklungsumgebung IDE (oder Tool der Wahl)

Projekt initialisieren

- Öffne VS Code und erstelle einen neuen Ordner, z. B. primzahl-info.
- Erstelle eine virtuelle Umgebung:
  - Ctrl + Shift + p, wähle „Venv“ aus, wähle die verfügbare „Python“ Version aus ODER
  - Öffne ein Terminal: python -m venv .venv



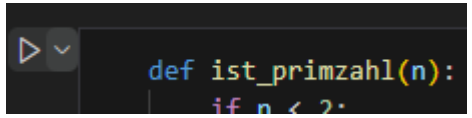
- Umgebung aktivieren
  - Terminal: .\.venv\Scripts\activate # Windows PowerShell
  - source .venv/bin/activate # macOS/Linux



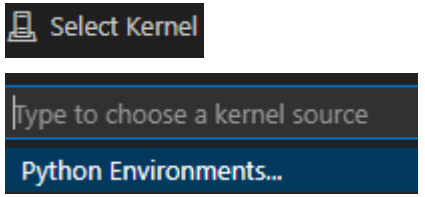
Beispielhaftes Programm erstellen

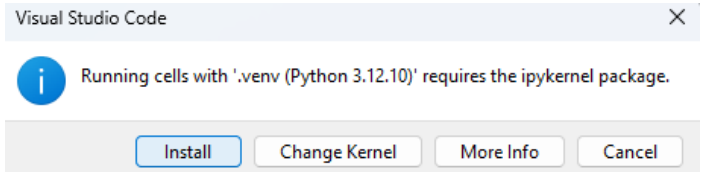
- Programm in Jupyter Notebook erstellen

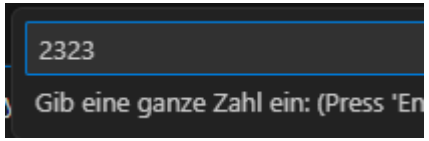
- Starten



  - Kernel auswählen
  - Ggfs. ipykernel installieren
  - Testen








```
2323 ist keine Primzahl. Die nächste Primzahl ist 2333. 🚀
```

Programm lokal über API zur Verfügung stellen (I)

FastAPI ist
- ein modernes, schnelles Web-Framework für Python, das verwendet wird, um REST-APIs bereitzustellen.
- Es generiert automatisch OpenAPI-Dokumentation (`/docs`)
- Die Logik eines Programms wird importiert und API Routen (`@app.post("/primzahl-check")` definiert.
- Es eignet sich für schnelles Protoyping.

main.py

```
39  @app.post("/primzahl-check", response_model=PrimzahlResponse)
```

Lokale API

POST /primzahl-check

Im Beispiel:
- Die Business-Logik (z. B. Berechnung, ob eine Zahl eine Primzahl ist) wird als normale Python-Funktion implementiert.
- FastAPI stellt diese Funktion über einen oder mehrere HTTP-Endpunkte zur Verfügung.
- Ein typischer Endpoint (z. B. `POST /primzahl-check`) nimmt JSON-Daten als Request-Body entgegen, ruft die Business-Logik auf und liefert die Antwort als JSON-Response zurück.
- Die Response könnte so aussehen:

```
{
  "zahl": 3444,
  "ist_prim": false,
  "naechste_prim": 3449
}
```

Programm lokal über API zur Verfügung stellen (II)

- **Uvicorn** ist ein leichter, schneller ASGI-Server, der FastAPI-Anwendungen ausführt
- Ein **ASGI-Server** ist ein **Application Server Gateway Interface** für Python-Webanwendungen. Er ist sozusagen der „Vermittler“ zwischen dem Webserver/Netzwerk und der Python-App.

- Ablauf

Programm lokal über API zur Verfügung stellen (III)

- requirements.txt erstellen

```
.venv\Scripts\activate
pip install -r requirements.txt
```

- Libraries installieren

primzahl-info.ipynb | requirements.txt

```
requirements.txt
1  fastapi==0.115.0
2  uvicorn[standard]==0.30.6
3  pydantic==2.9.2
```

- main.py erstellen (Beispiel auf moodle)
  - Logik des eigenen Programms integrieren

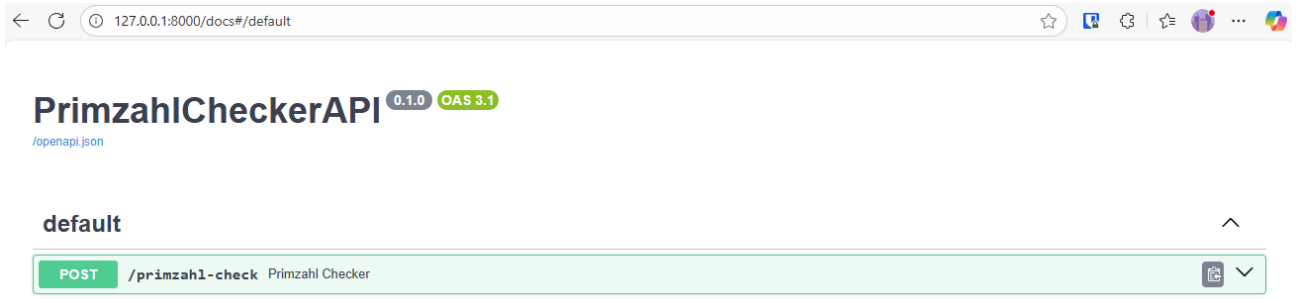
```
Code > main.py > naechste_primzahl
from fastapi import FastAPI           # Importiert die FastAPI-Klasse, um eine Web-API zu erstellen
from pydantic import BaseModel        # Importiert die BaseModel-Klasse für Daten-Validierung und -Serialisierung
from typing import Optional           # Für optionale Felder im Response-Typ

# Hilfsfunktionen für Primzahl-Logik
## DEIN CODE BEGINNT HIER ##

# Definiert das Eingabe-Schema der API: erwartet ein JSON mit dem Feld 'zahl'
class ZahlRequest(BaseModel):
    zahl: int                         # Die zu prüfende ganze Zahl

# Definiert das Ausgabe-Schema der API
class PrimzahlResponse(BaseModel):
    zahl: int                         # Die Eingabezahl
    ist_prim: bool                    # True, wenn Primzahl
    naechste_prim: Optional[int] = None # Nächste Primzahl, falls keine Primzahl eingegeben wurde

# Erstellt eine FastAPI-Applikation mit dem Titel "PrimzahlCheckerAPI"
app = FastAPI(title="PrimzahlCheckerAPI")

# Deklariert einen POST-Endpunkt unter "/primzahl-check"
# response_model sorgt dafür, dass die Antwort dem PrimzahlResponse-Schema entspricht
@app.post("/primzahl-check", response_model=PrimzahlResponse)
async def primzahl_checker(req: ZahlRequest):
    # Prüft, ob die Zahl eine Primzahl ist
    prim = ist_primzahl(req.zahl)

    # Falls nicht, berechnet die Funktion die nächste Primzahl
    next_p = None if prim else naechste_primzahl(req.zahl)

    # Gibt eine Instanz von PrimzahlResponse zurück, die automatisch zu JSON serialisiert wird
    return PrimzahlResponse(
        zahl=req.zahl,
        ist_prim=prim,
        naechste_prim=next_p
    )
```

- Starten

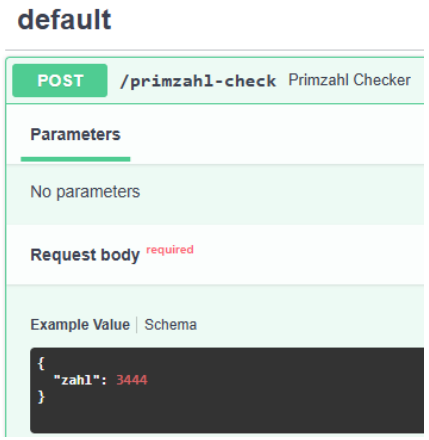
```
(.venv) PS C:\rb\github\primzahl-info> uvicorn Code.main:app --reload
INFO:     Will watch for changes in these directories: ['C:\\rb\\github\\primzahl-info']
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:     Started reloader process [3124] using WatchFiles
INFO:     Started server process [28624]
INFO:     Waiting for application startup.
```

API aufrufen

- API aufrufen: http://127.0.0.1:8000/**docs**



- Try it out



- Testeingabe
- Response

- Terminal

```
INFO:     127.0.0.1:54533 - "GET /docs HTTP/1.1" 200 OK
INFO:     127.0.0.1:54533 - "GET /openapi.json HTTP/1.1" 200 OK
INFO:     127.0.0.1:54534 - "POST /primzahl-check HTTP/1.1" 200 OK
```

Umsetzung mit der Cloud

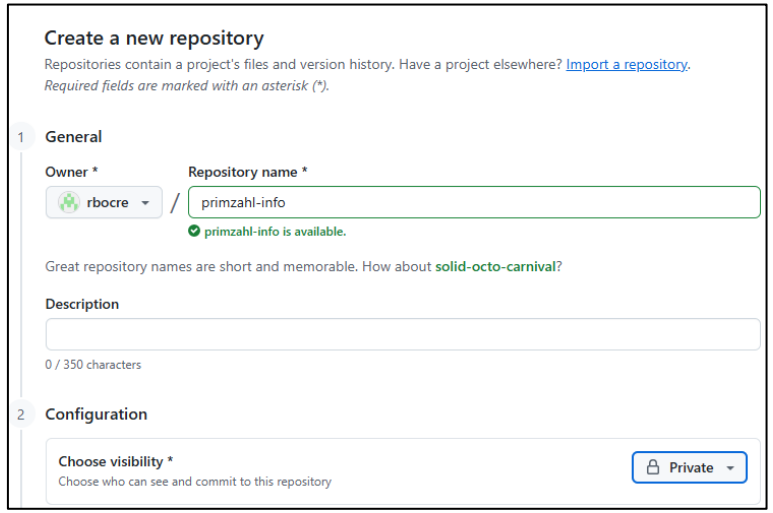
Voraussetzungen

– Grundkenntnisse in Git und GitHub

– Git installiert
– GitHub Account vorhanden
– render.com Account (Einloggen mit dem GitHub Account)

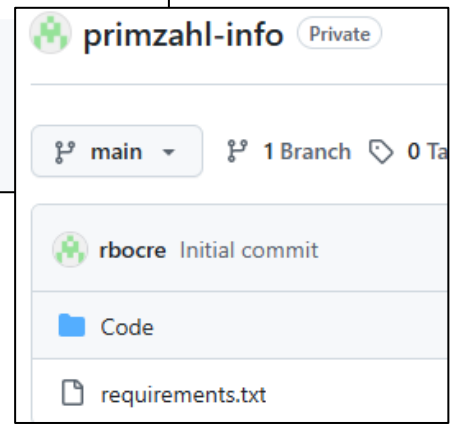
Versionskontrolle mit Git & GitHub

1. Erstelle auf GitHub ein neues Repository mit demselben Namen wie für «DevOps Lokal Umsetzung».



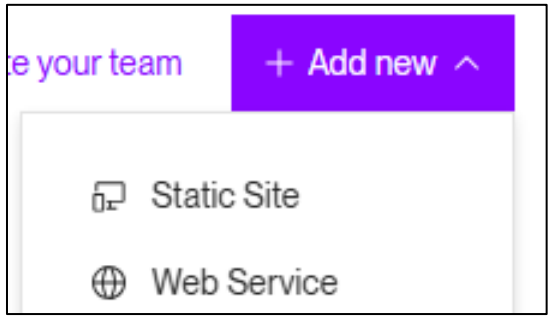
2. Git initialisieren / Lokales Projekt vorbereiten
   - `git init`
   - `echo ".venv/" >> .gitignore`
   - `git add .`
   - `git commit -m "Initial commit"`
   - `git branch -M main`

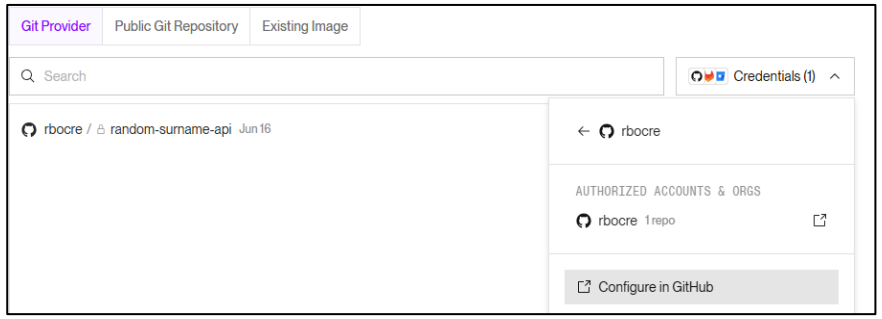
3. Lokales Repo mit GitHub verbinden
   - `git remote add origin https://github.com/rbocre/primzahl-info.git`
   - `git push -u origin main`

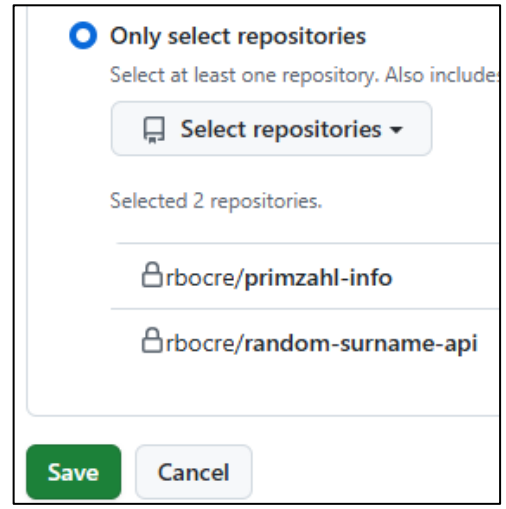


Deployment auf render.com

- Neues Projekt hinzufügen
- Neuer Web Service auf dashboard.render.com
- Zugriff auf Repository in GitHub erlauben







Konfiguration

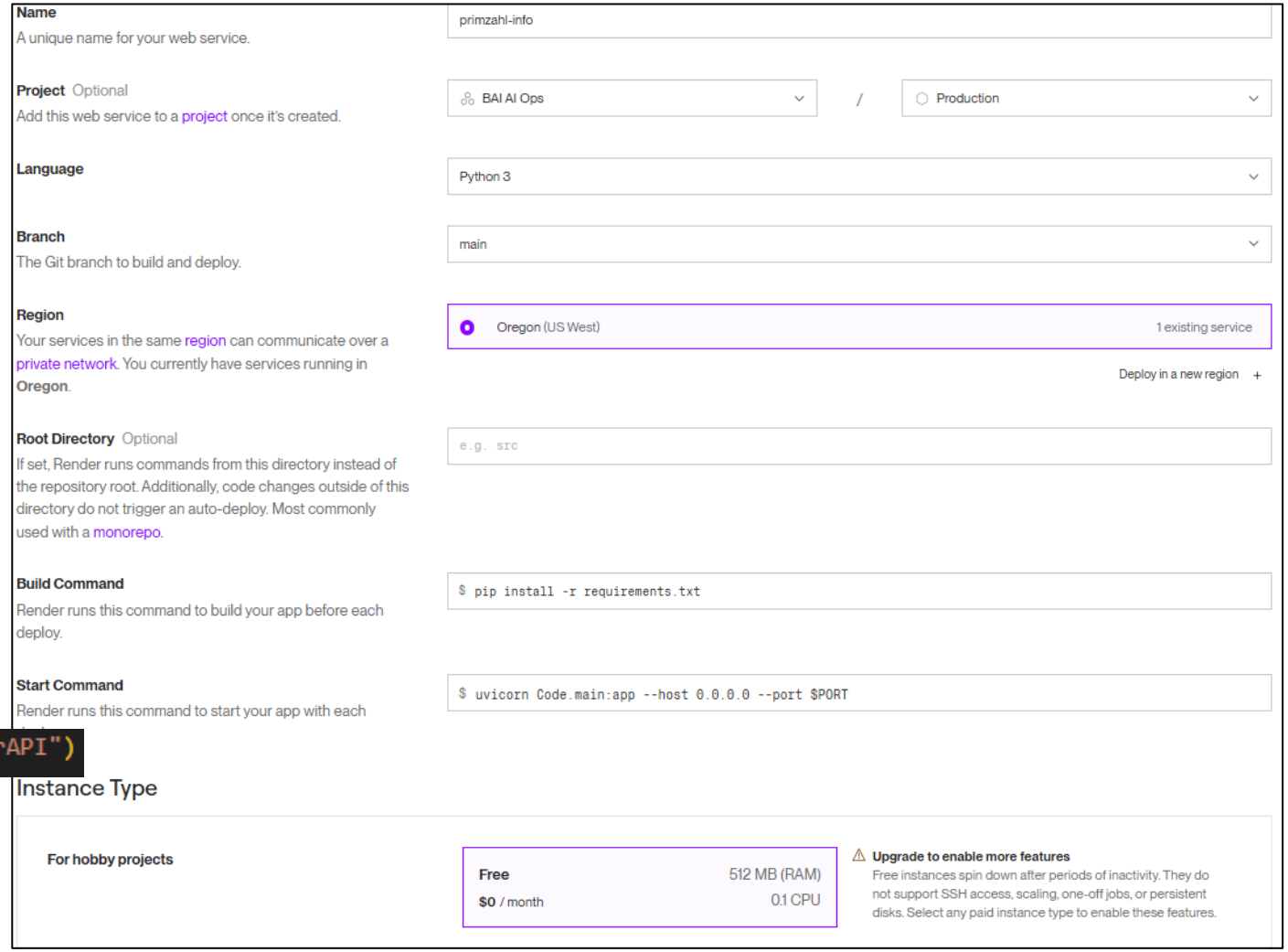
- **Branch**: main
- **Build Command**: pip install -r requirements.txt
- **Start Command**: uvicorn Code.main:app --host 0.0.0.0 --port $PORT

Ordnername

Sofern main.py benannt

In main.py als app benannt

```
app = FastAPI(title="PrimzahlCheckerAPI")
```



Build überprüfen

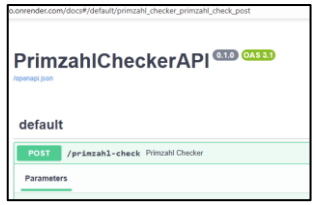
```
Aug 30 09:09:14 PM  ==> Uploading build...
Aug 30 09:09:20 PM  ==> Uploaded in 4.8s. Compression took 1.5s
Aug 30 09:09:20 PM  ==> Build successful 🎉
Aug 30 09:09:23 PM  ==> Deploying...
Aug 30 09:09:36 PM  ==> Running 'uvicorn Code.main:app --host 0.0.0.0 --port $PORT'
Aug 30 09:09:43 PM  INFO:     Started server process [55]
Aug 30 09:09:43 PM  INFO:     Waiting for application startup.
Aug 30 09:09:43 PM  INFO:     Application startup complete.
Aug 30 09:09:43 PM  INFO:     Uvicorn running on http://0.0.0.0:10000 (Press CTRL+C to quit)
Aug 30 09:09:43 PM  INFO:     127.0.0.1:58518 - "HEAD / HTTP/1.1" 404 Not Found
Aug 30 09:09:43 PM  ==> Your service is live 🎉
Aug 30 09:09:43 PM  ==>
Aug 30 09:09:43 PM  ==> ///////////////////////////////////////////////////////////
Aug 30 09:09:43 PM  ==>
Aug 30 09:09:43 PM  ==> Available at your primary URL https://primzahl-info.onrender.com
Aug 30 09:09:43 PM  ==>
Aug 30 09:09:43 PM  ==> ///////////////////////////////////////////////////////////
```

API abrufen

- Bspw. über Jupyter Notebook

```
test_API.ipynb
+ Code  + Markdown | Run All

pip install requests
[2]  ✓ 5.8s
```

- oder Swagger
  - https://primzahl-info.onrender.com/docs




```
import requests

url = "https://primzahl-info.onrender.com/primzahl-check"
resp = requests.post(url, json={"zahl": 3444})
data = resp.json()
print(f"{data['ist_prim']} {data['naechste_prim']} ")

[3]  ✓ 0.6s
...  False 3449
```

```
@app.post("/primzahl-check", r
```

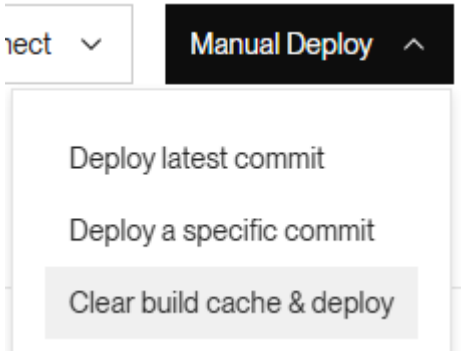
```
return PrimzahlResponse(
    zahl=req.zahl,
    ist_prim=prim,
    naechste_prim=next_p
)
```

- Der Aufruf ist sichtbar in den Logs.

```
Aug 30 09:17:28 PM  9qr6f  INFO:  80.219.19.199:0 - "POST /primzahl-check HTTP/1.1" 200 OK
```

Code anpassen & deployen

- Code in VS Code anpassen
- Stage, Commit. Danach auf GitHub pushen
- Re-Try Web-Service (render.com)



Coaching

Time Slots

Bei Bedarf an Coaching…

- Eine Termineinladung an roman.brun@fhnw.ch
- Zu einem der Timeslots gemäss Tabelle zustellen

- Am Zeitpunkt ins Teams E-AI Operations HS25_M365 einloggen, welches auch sonst für die Vorlesung verwendet wird.

- Vorbereitung für dich auf den Termin:
  - Was wurde erreicht?
  - Was Variante wurden versucht?
  - Wo komme ich nicht weiter?

7. Oktober 2025

Von	Bis
8:00	8:15
8:15	8:30
8:30	8:45
8:45	9:00
9:00	9:15
9:15	9:30
9:30	9:45
9:45	10:00