

# Bruchrechnen mit Python

*Ein Hackathon*

Polu Tartakower

*Kiel, September 2025*

# Inhaltsverzeichnis

|   |           |
|---|-----------|
| <b>1 Übersicht .....</b>                                    | <b>4</b>  |
| 1.1 Die Aufgabenstellung .....                              | 4         |
| 1.2 Notwendige Programmiererfahrung .....                   | 4         |
| 1.3 Mathematische Vorkenntnisse .....                       | 5         |
| 1.4 Eingesetzte Technologien .....                          | 5         |
| 1.5 Methoden der Softwareentwicklung .....                  | 5         |
| <b>2 Die Entwicklungsumgebung .....</b>                     | <b>7</b>  |
| 2.1 Installationsquellen und Lizenzbedingungen .....        | 7         |
| 2.2 Das SDK für Python .....                                | 8         |
| 2.3 Die integrierte Entwicklungsumgebung (IDE) ...          | 8         |
| 2.3.1 VS Code .....   | 9         |
| 2.4 Die Projektstruktur .....                               | 9         |
| <b>3 Die Integrierte Entwicklungsumgebung VS Code .....</b> | <b>10</b> |
| 3.1 Installation von VS Code .....                          | 10        |
| 3.2 Deutsche Sprachunterstützung .....                      | 10        |
| 3.3 Ein Profil für Python-Programmierung anlegen .          | 10        |
| 3.4 Sprachunterstützungen für Python .....                  | 11        |
| 3.5 Die Befehlspalette .....                                | 11        |
| 3.6 Einstellungen von VS Code und den Plugins ...           | 11        |
| <b>4 Einrichten des Entwicklungsprojektes .....</b>         | <b>12</b> |
| 4.1 Der Hauptordner des Projektes .....                     | 12        |
| 4.2 Der Arbeitsbereich in VS Code .....                     | 12        |
| 4.3 Die Ordner für den Quellcode .....                      | 12        |
| 4.4 Die Pfadangabe für die Python-Module .....              | 12        |
| 4.5 Die virtuelle Entwicklungsumgebung .....                | 13        |

|   |           |
|---|-----------|
| 4.6 Konfiguration der Softwaretest und Installation<br>des Testframeworks ..... | 13        |
| 4.7 Installation des GUI-Frameworks NiceGui .....                               | 14        |
| <b>5 Die Softwarearchitektur .....</b>  | <b>15</b> |
| 5.1 Die Problemstellung .....   | 15        |
| 5.2 Die Lösung .....  | 15        |
| 5.3 Unterteilung in Module .....  | 15        |
| <b>6 Eine Funktion in Python: Programmierung und<br/>Testen .....</b>           | <b>16</b> |
| 6.1 Die Funktion zur Berechnung des größten<br>gemeinsamen Teilers .....        | 16        |
| 6.2 Der Unittest .....  | 16        |

# 1 Übersicht

## 1.1 Die Aufgabenstellung

Es soll eine Webanwendung programmiert werden, mit der beispielsweise folgende Bruchrechenoperation berechnet werden kann:

$$\frac{1}{2} \cdot \frac{2}{3}$$

Die Ergebniszeile soll so aussehen:

$$\frac{1}{2} \cdot \frac{2}{3} = \frac{2}{6} = \frac{1}{3}$$

Das soll auf der Oberfläche so gestaltet werden:

- Auf der Webseite gibt es je ein Eingabefeld für den Nenner und für den Zähler der beiden Operanden.
- Die Rechenoperation kann aus einer Dropdown-Liste ausgewählt werden, welche die vier Grundrechenarten enthält.
- Das erste Gleichheitszeichen ist ein Button, dessen Betätigung die Be-rechnung ausführt.
- Nach dem ersten Gleichheitszeichen wird das Ergebnis ungekürzt darge-stellt. Das gekürzte Ergebnis folgt nach dem zweiten Gleichheitszeichen.

## 1.2 Notwendige Programmiererfahrung

Du solltest erste Erfahrungen mit der Programmierung besitzen, konkret folgende Sprachkonstrukte bereits angewendet haben:

- Variable und Konstante
- Wertzuweisungen
- Bedingte Anweisungen
- Schleifen

Falls du noch nie programmiert hast, wird der Hackathon vermutlich zu schwierig für dich sein. Dann solltest du dich vorher unbedingt mit den grundlegenden Sprachelementen von Python vertraut machen.

## **1.3 Mathematische Vorkenntnisse**

Außerdem musst du ein grundlegendes Wissen über das Rechnen mit Brüchen besitzen:

- Was ist ein Bruch? Woraus besteht er?
- Welche Bedeutung besitzt der Bruchstrich?
- Was ist ein gekürzter Bruch? Wie wird ein Bruch gekürzt?
- Wie wird ein Bruch erweitert?
- Wie werden die Grundrechenoperationen mit Brüchen durchgeführt?

Wenn du noch nie mit Brüchen gerechnet hast oder dein Wissen nicht mehr frisch ist, muss du unbedingt zuvor das Rechnen mit Brüchen lernen oder wiederholen.

## **1.4 Eingesetzte Technologien**

Für die Programmierung verwenden wir die Programmiersprache Python. Wenn du bisher nur mit einer anderen Sprache programmiert hast, beispielsweise mit Scratch, wirst du dich schnell in Python zurecht finden.

Für die Programmierung der Web-Oberfläche verwenden wir das GUI-Framework NiceGui. Du musst also keine Vorkenntnisse in HTML, CSS oder JavaScript besitzen, sie helfen aber auf jeden Fall.

## **1.5 Methoden der Softwareentwicklung**

Im Rahmen des Hackathons soll nicht nur programmiert werden, sondern auch verschiedene moderne Methoden, wie in der Softwareentwicklung

heutzutage gearbeitet wird, angewendet werden. Dazu sollst du im Team zusammen mit anderen folgende Aufgaben bearbeiten:

- Erstellung einer Entwicklungsumgebung (Abschnitt 2)
- Entwurf einer Softwarearchitektur
- Programmierung einer Funktion zur Berechnung des größten gemeinsamen Teilers (ggT)
- Programmierung eines Unittests für die ggT-Funktion
- Objektorientierte Analyse und Design für einen Bruch
- Implementierung der Klasse Bruch
- Programmierung der Unittests für die Klasse Bruch
- Programmierung der Benutzeroberfläche

## 2 Die Entwicklungsumgebung

Unsere Entwicklungsumgebung umfasst folgende Komponenten:

- das Software Developer Kit (SDK), also die Programmiersprache Python einschließlich des Interpreters und vieler Bibliotheken
- eine integrierte Entwicklungsumgebung (**IDE**) zum Editieren von Quellcode
- eine einheitliche Ordnerstruktur im Dateisystem für Softwareentwicklungsprojekte
- eine virtuelle Umgebung
- ein Testframework
- weitere Bibliotheken von Drittanbietern

Wer bereits Erfahrung mit den Werkzeugen für die Softwareentwicklung hat, kann folgende Komponenten zusätzlich integrieren:

- eine Software zur Verwaltung des Python-Projektes
- eine lokale Versionsverwaltung
- einen Account für eine cloudestützte Versionsverwaltung

### 2.1 Installationsquellen und Lizenzbedingungen

Wir werden verschiedene Tools zur Unterstützung der Programmierung herunterladen und installieren. Dabei sind zwei Aspekte sehr, sehr wichtig:

#### **Verwende nur seriöse Installationsquellen!**

Die Installation von Software auf dem Computer birgt immer das Risiko, den Computer mit Schadsoftware zu infizieren. Das muss unbedingt vermieden werden!

Die sichersten Installationsquellen bilden entweder die Paketverwaltung des Betriebssystems oder die Webseite des Herstellers.

#### **Halte die Lizenzbedingungen der Software ein!**

Jede Software enthält eine Lizenz, deren Bedingungen bei Nutzung unbedingt eingehalten werden müssen. Andernfalls begeht man als Nutzer eine Straftat! Das gilt auch für freie und kostenlose Software! Informiere dich über die jeweiligen Lizenzen und halten deren Bedingungen unbedingt ein.

## 2.2 Das SDK für Python

In einer aktuellen Linux-Installation sollte in der Regel eine aktuelle Version von Python bereits installiert sein. Unter MS Windows kann eine aktuelle Version von Python über den Microsoft Shop installiert werden.

Aufgaben:

- Recherchiere, welche Versionen von Python als aktuell gelten.
- Finde heraus, wie auf einem Computer geprüft werden kann, ob Python installiert ist und gegebenenfalls welche Version vorliegt.

## 2.3 Die integrierte Entwicklungsumgebung (IDE)

Eine integrierte Entwicklungsumgebung (Integrated Developer Environment, **IDE**) enthält als Kernkomponente einen Editor zum Erstellen und Bearbeiten von Quellcode. Dieser Codeeditor unterstützt die Entwicklerin durch farbige Markierung des Quelltextes und laufende Prüfungen beim Programmieren.

Darüber hinaus besteht eine IDE aus weiteren zahlreichen Hilfsprogrammen für die Softwareentwicklung, z.B. zur Bedienung der Versionsverwaltung oder zur Ausführung der Testprogramme. Alle Funktionalitäten sind benutzerfreundlich innerhalb einer Bedienoberfläche zusammen gefasst.

**Tipp:** Verwende zum Programmieren unbedingt eine für die jeweilige Programmiersprache optimierte IDE.

Für Python existieren zwei sehr gut geeignete IDE:

- Microsoft Visual Code (VS Code)

- JetBrains Pycharm

Bei diesem Hackathon verwenden wir VS Code. Falls du bereits Erfahrung mit Pycharm besitzt, darfst du natürlich weiterhin Pycharm verwenden.

### **2.3.1 VS Code**

Informiere dich über VS Code. Aufgaben:

- Woher kannst du VS Code beziehen?
- Wie viel kostet es?
- VS Code allein ist gar keine IDE, sondern nur ein Code-Editor. Wie wird VS Code zur vollwertigen IDE?

Die Installation und Konfiguration von VS Code behandeln wir im nächsten Kapitel. (Abschnitt 3)

## **2.4 Die Projektstruktur**

Der Aufbau der Projektstruktur wird im übernächsten Kapitel beschrieben. (Abschnitt 4)

## **3 Die integrierte Entwicklungsumgebung VS Code**

Das Tool Microsoft Visual Studio Code (**VS Code**) ist streng genommen nur ein Code-Editor, der allerdings durch die Integration von verschiedenen Plugins zu einer integrierten Entwicklungsumgebung (**IDE**) wird.

### **3.1 Installation von VS Code**

VS Code kann entweder über die Paketverwaltung des Betriebssystems oder die Herstellerseite bezogen werden. Aus der Webseite finden sich auch umfangreiche Dokumentationen, insbesondere zur Installation.

### **3.2 Deutsche Sprachunterstützung**

Wir werden in diesem Text die deutschen Bezeichnungen benutzen. Daher empfehlen wir die Installation des Plugins *German Language Pack* von Microsoft.

Wer bereits größere Erfahrung hat, darf natürlich mit den englischen Bezeichnungen arbeiten.

### **3.3 Ein Profil für Python-Programmierung anlegen**

Wer mit verschiedenen Programmiersprachen in VS Code arbeiten möchte, benötigt jeweils spezielle Plugins für diese Programmiersprachen. Um die unterschiedlichen Plugins voneinander getrennt zu halten, empfiehlt sich die Einrichtung eines Profils für jeweils jede Programmiersprache.

Im Navigator links unten über das Zahnrad das Kontextmenü öffnen, dort *Profil* auswählen.

## **3.4 Sprachunterstützungen für Python**

Falls du ein Profil für die Programmiersprache Python angelegt hast, aktiviere dieses.

Installiere das Plugin *Python* von Microsoft.

Dadurch werden zwei weitere Plugins installiert, die wir benötigen.

- *Pylance*
- *Python Debugger*

Falls zusätzlich das Plugin *Python Environments* installiert wird, diese bitte wieder löschen! Es ist von sehr schlechter Qualität und schadet nur.

## **3.5 Die Befehlspalette**

Wichtige Funktionen von VS Code und der Python-Plugins werden über die Befehlspalette ausgeführt. Diese kann auf zwei Wegen aufgerufen werden:

- Menü *Anzeigen*, erster Eintrag *Befehlspalette...*
- Tastenkombination STRG+UMSCHALT+P

## **3.6 Einstellungen von VS Code und den Plugins**

VS Code und die Plugins lassen sich umfangreich konfigurieren. Das geschieht über die Einstellungen, die auf folgenden Wegen aufgerufen werden können:

- Im Navigator links unten über das Zahnrad das Kontextmenü öffnen.
- Tastenkombination STRG+,

## 4 Einrichten des Entwicklungsprojektes

In diesem Kapitel werden wir die Infrastruktur für unser Entwicklungsprojekt einrichten. Das sollte überwiegend in VS Code durchgeführt werden, damit alle Komponenten reibungslos zusammen arbeiten.

### 4.1 Der Hauptordner des Projektes

Lege im Dateisystem unter dem Ordner *Programmierprojekte* einen neuen Ordner *Bruchrechnen* an. Dieser neue Ordner wird der Hauptordner des Projektes sein.

Ab jetzt werden alle Einstellungen in VS Code vorgenommen.

### 4.2 Der Arbeitsbereich in VS Code

- Starte VS Code.
- Öffne dort den Ordner *Bruchrechnen*.
- Lege einen Arbeitsbereich an: Im Menüpunkt *Datei/Arbeitsbereich speichern unter...* den Vorschlag annehmen.
- Der Arbeitsbereich lässt sich zukünftig über die Datei *Bruchrechnen.code-workspace* öffnen und schließen.

### 4.3 Die Ordner für den Quellcode

Lege im Ordner *Bruchrechnen* folgende Unterordner an.

- *src*: Hier speichern wir die Quellcode-Dateien für funktionalen Code.
- *test*: Hier werden die Unitests gespeichert.

### 4.4 Die Pfadangabe für die Python-Module

Damit sich alle Python-Module untereinander kennen, gibt es die Umgebungsvariable PYTHONPATH, die wir ergänzen müssen.

- Lege im Hauptordner die Datei `.env` an. Beachte: Der Name beginnt mit einem Punkt.
- Schreibe folgende Zeile in diese Datei: `PYTHONPATH=$PYTHONPATH:../../src:/test`

## 4.5 Die virtuelle Entwicklungsumgebung

Dieser Schritt sollte unbedingt in VS Code erfolgen! Die Einrichtung einer virtuellen Umgebung ohne VS Code kann große Probleme für unser Vorhaben erzeugen.

- Öffne die Befehlspalette.
- Wähle *Python: Umgebung erstellen*.
- Wähle *Venv* aus.
- Wähle den Interpreter `/bin/python3` aus.

Im Hintergrund wird eine virtuelle Umgebung angelegt. Dass es geklappt hat, erkennst du an:

- Im Hauptordner gibt es einen neuen Unterordner `.venv`.
- Wenn du ein neues Terminal startest, dann ist die virtuelle Umgebung aktiviert. Der Pfadname des Terminalprompts beginnt mit `(.venv)`

## 4.6 Konfiguration der Softwaretest und Installation des Testframeworks

- Öffne die Befehlspalette.
- Wähle *Python: Tests konfigurieren*.
- Wähle *pytest*.
- Wähle *. Root directory*.

Im Hintergrund wird das Testframework Pytest installiert. Prüfe:

- Starte ein Terminal.
- Der Befehl `pip list` sollte `pytest` auflisten.

- Drücke im Navigator auf der linken Seite auf den Erlenmeyerkolben.
  - Zeigt dieser den Hauptordner an? Dann sollte alles in Ordnung sein.
  - Zeigt der Navigator einen Fehler? Dieser sollte nach dem Aktualisieren verschwinden.

## 4.7 Installation des GUI-Frameworks NiceGui

- Öffne ein Terminal.
- Ist die virtuelle Umgebung aktiviert? Andernfalls bitte aktivieren:  
`source .venv/bin/activate`
- Der Befehl `python -m pip install nicegui` installiert das GUI-Framework.

# **5 Die Softwarearchitektur**

## **5.1 Die Problemstellung**

Ein einziges Programm mit vielen Zeilen Code ist unübersichtlich und schwer zu pflegen. Fehler sind schwer zu finden und Änderungen können nur aufgenommen vorgenommen werden.

## **5.2 Die Lösung**

Wir teilen das Programm in verschiedene Einzelmodule auf. Jedes Modul nimmt eine Aufgabe wahr. So wird das Gesamtprogramm viel übersichtlicher und kann besser gewartet werden.

Die Unterteilung in Module und deren Zusammenspiel nennt man **Softwarearchitektur**.

## **5.3 Unterteilung in Module**

# **6 Eine Funktion in Python: Programmierung und Testen**

## **6.1 Die Funktion zur Berechnung des größten gemeinsamen Teilers**

Für das Kürzen von Brüchen benötigen wir die Berechnung des größten gemeinsamen Teilers zweier Ganzzahlen.

### **Aufgabe:**

- Lege eine Python-Datei mit dem Namen *mathehelper.py* an.
- Programmiere darin eine Funktion, die zwei Ganzzahlen als Parameter entgegen nimmt und den zugehörigen ggT berechnet.
- Tipp: Verwende Type-Hints für die Deklaration der Parameter und für den Rückgabetyp der Funktion.
- Tipp: Zur Berechnung des ggT gibt es den sogenannten Euklidischen Algorithmus in einer iterativen, schnellen Variante.

## **6.2 Der Unittest**