Taschenrechner-Projekt mit Python GUI und Flask-Server

Dieses Projekt besteht aus einem einfachen Taschenrechner mit einer grafischen Benutzeroberfläche (GUI), der mit Python und `Tkinter` entwickelt wurde. Die arithmetischen Berechnungen werden von einem Flask-basierten Server durchgeführt, der als REST-API dient.

Projektübersicht

- **Client (Tkinter GUI)**: Die GUI nimmt zwei Zahlen als Eingabe und sendet Anfragen an den Server, um die Berechnungen (Addition, Subtraktion, Multiplikation und Division) durchzuführen.
- **Server (Flask API)**: Der Server verarbeitet die Berechnungen basierend auf den von der GUI gesendeten Anfragen und gibt die Ergebnisse zurück.

Voraussetzungen

Vorerst muss vorerst Folgendes auf dem System installiert sein:

1. **Python 3.12**: Du kannst Python von python.org herunterladen und installieren.
2. **Flask**: Flask ist das Framework, das für den Server verwendet wird. Installiere es mit pip:

```
pip install Flask
```

Schritt 1: Flask-Server einrichten und starten

Code für den Server (`server.py`):

```python
from flask import Flask, request, jsonify

app = Flask(__name__)

@app.route('/add')
def addieren():
    zahl1 = float(request.args.get('zahl1'))
    zahl2 = float(request.args.get('zahl2'))
    result = zahl1 + zahl2
    return jsonify({'result': result})

@app.route('/sub')
def subtrahieren():
    zahl1 = float(request.args.get('zahl1'))
    zahl2 = float(request.args.get('zahl2'))
    result = zahl1 - zahl2
    return jsonify({'result': result})

@app.route('/mul')
def multiplizieren():
    zahl1 = float(request.args.get('zahl1'))
    zahl2 = float(request.args.get('zahl2'))
    result = zahl1 * zahl2
    return jsonify({'result': result})

@app.route('/div')
def dividieren():
    zahl1 = float(request.args.get('zahl1'))
    zahl2 = float(request.args.get('zahl2'))
    if zahl2 != 0:
        result = zahl1 / zahl2
    else:
        result = 'Division durch 0 ist nicht erlaubt'
    return jsonify({'result': result})

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

Erklärung des Server-Codes:

- **Imports**: Die Bibliotheken `Flask`, `request`, und `jsonify` werden importiert. `Flask` ist das Framework für den Webserver, `request` wird verwendet, um die Anfrageparameter zu erhalten, und `jsonify` wird verwendet, um JSON-Antworten zu erstellen.
- **Erstellen der Flask-Anwendung**: `app = Flask(__name__)` erstellt eine Flask-Anwendung.

- **Routen-Definition**: Jede Funktion (`addieren` , `subtrahieren` , `multiplizieren` , `dividieren`) definiert eine Route, die auf eine bestimmte URL-Anfrage reagiert und die entsprechenden Berechnungen durchführt. Die Ergebnisse werden als JSON zurückgegeben.
- **Starten des Servers**: `app.run(host='0.0.0.0', port=5000)` startet den Server auf `localhost` an Port 5000.

Schritte zum Starten des Servers:

1. Speichere den obigen Code in einer Datei namens `server.py` .
2. Öffne ein Terminal oder eine Eingabeaufforderung.
3. Navigiere zu dem Verzeichnis, in dem die Datei `server.py` gespeichert ist.
4. Starte den Server mit dem folgenden Befehl:

```
python server.py
```

1. Der Server wird auf `localhost:5000` laufen und Anfragen für Berechnungen entgegennehmen.

Testen des Servers:

Um sicherzustellen, dass der Server läuft, kannst du in deinem Webbrowser die folgende URL aufrufen:

```
http://localhost:5000/add?zahl1=2&zahl2=3
```

Wenn alles richtig funktioniert, sollte eine JSON ausgabe ersichtlich sein:

```
{"result": 5}
```

Schritt 2: Tkinter GUI-Client einrichten und starten

Code für den Client (`tr_mit_gui_as.py`):

```
import tkinter as tk
import requests

# Funktion, um das Ergebnis vom Server zu bekommen
def get_result(operation, zahl1, zahl2):
    base_url = 'http://localhost:5000'
    urls = {
        'addieren': '/add',
        'subtrahieren': '/sub',
        'multiplizieren': '/mul',
        'dividieren': '/div'
    }

    url = base_url + urls.get(operation, '/add')
    params = {'zahl1': zahl1, 'zahl2': zahl2}

    try:
        response = requests.get(url, params=params)
        response.raise_for_status()
        result = response.json()
        return result.get('result', 'Fehler beim Abrufen des Ergebnisses')
    except requests.exceptions.ConnectionError:
        return "Server ist nicht erreichbar. Bitte sicherstellen, dass der Server
läuft."
    except requests.exceptions.RequestException as err:
        return f"Fehler: {err}"

# Funktionen für die Berechnungen
def addieren():
    try:
        zahl1 = float(entry_1.get())
        zahl2 = float(entry_2.get())
        result = get_result('addieren', zahl1, zahl2)
        solution_entry.delete(0, tk.END)
        solution_entry.insert(0, f"{result}")
    except ValueError:
        solution_entry.delete(0, tk.END)
        solution_entry.insert(0, "Ungültige Eingabe. Bitte Zahlen eingeben.")

# Ähnliche Funktionen für Subtrahieren, Multiplizieren, Dividieren...

# GUI Setup
root = tk.Tk()
root.title("Einfacher Taschenrechner")

tk.Label(root, text="Erste Zahl:").grid(row=0, column=0)
entry_1 = tk.Entry(root)
```

```
entry_1.grid(row=0, column=1)

tk.Label(root, text="Zweite Zahl:").grid(row=1, column=0)
entry_2 = tk.Entry(root)
entry_2.grid(row=1, column=1)

tk.Label(root, text="Solution:").grid(row=2, column=0)
solution_entry = tk.Entry(root)
solution_entry.grid(row=2, column=1)

tk.Button(root, text="+", command=addieren).grid(row=3, column=0)

# Weitere Buttons für -, *, /

root.mainloop()
```

Erklärung des Client-Codes:

- **Imports**: `tkinter` wird für die GUI verwendet und `requests` für die HTTP-Anfragen an den Server.
- **`get_result`-Funktion**: Diese Funktion sendet eine GET-Anfrage an den Server mit den gewählten Operationen und den zwei Zahlen. Sie verarbeitet die Antwort und gibt das Ergebnis zurück. Bei Fehlern wird eine entsprechende Fehlermeldung angezeigt.
- **Berechnungsfunktionen**: Funktionen wie `addieren`, `subtrahieren`, `multiplizieren` und `dividieren` sammeln die Eingaben des Benutzers, rufen `get_result` auf und zeigen das Ergebnis in der GUI an.
- **GUI Setup**: Der Code erstellt das GUI-Fenster mit Labels, Eingabefeldern und Buttons. Jede Schaltfläche ruft die entsprechende Berechnungsfunktion auf.
- **Starten der GUI**: `root.mainloop()` startet die Tkinter-Event-Schleife, die das GUI-Fenster anzeigt und auf Benutzereingaben wartet

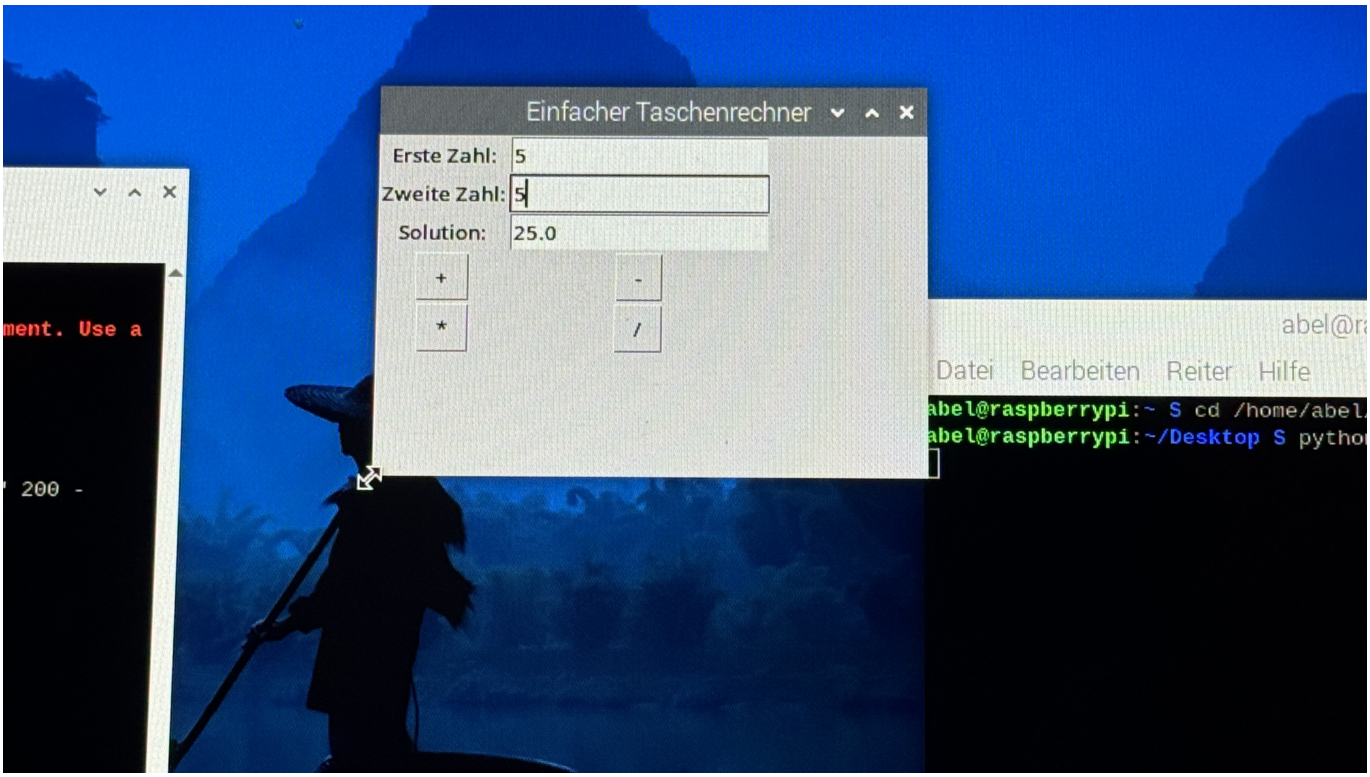
Schritte zum Starten der GUI:

1. Speichere den oben stehenden Code in einer Datei namens `tr_mit_gui_as.py`.
2. Stelle sicher, dass der Flask-Server bereits läuft (siehe Schritt 1).
3. Öffne ein weiteres Terminal oder eine neue Eingabeaufforderung.
4. Navigiere zu dem Verzeichnis, in dem `tr_mit_gui_as.py` gespeichert ist.

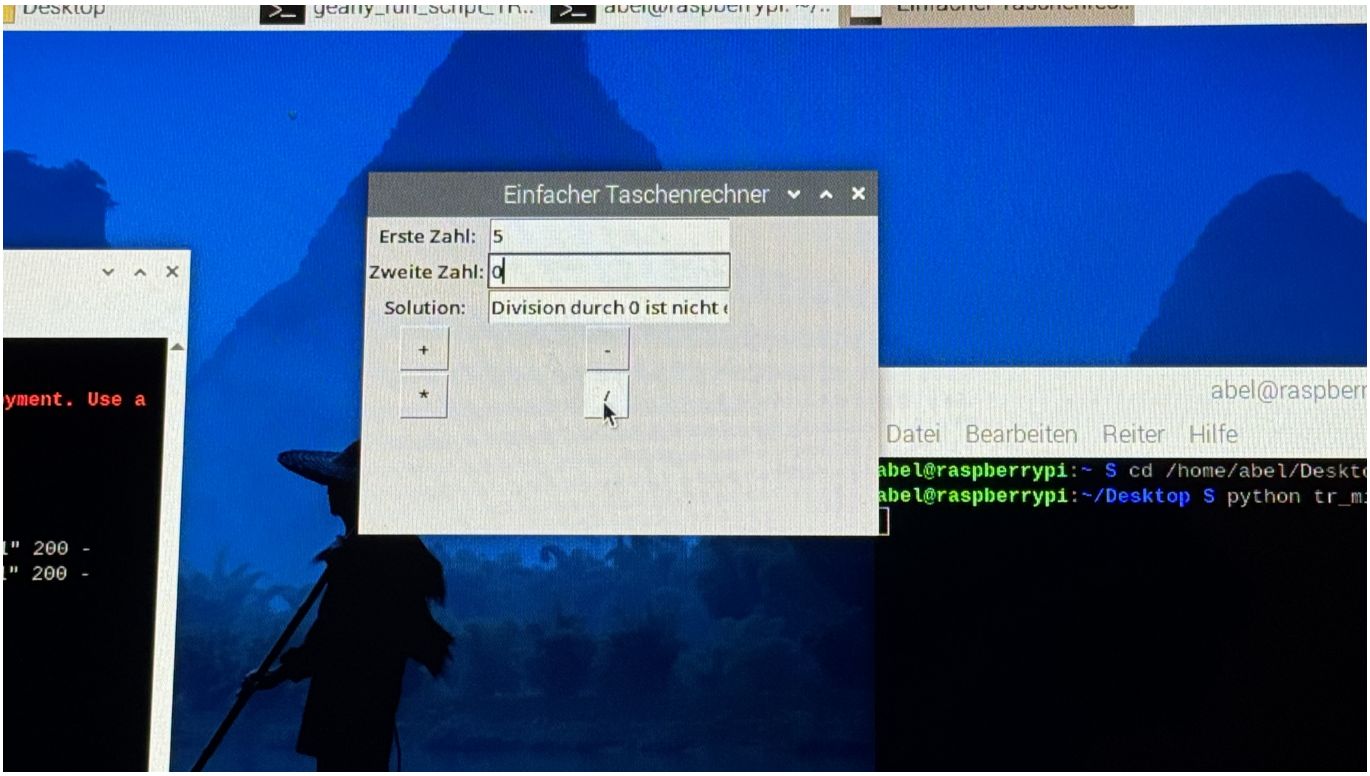
5. Starte die Tkinter GUI mit dem folgenden Befehl:

```
python tr_mit_gui_as.py
```

1. Ein Fenster mit der Taschenrechner-GUI wird angezeigt. Gib zwei Zahlen ein und klicke auf die `*`-Schaltfläche, um eine Multiplikation durchzuführen. Das Ergebnis wird im `Solution`-Feld angezeigt.



2. Hier ein Beispiel wenn man durch 0 dividiert. Es erscheint eine Fehlermeldung:


Einfacher Taschenrechner
Erste Zahl: 5
Zweite Zahl: 0
Solution: Division durch 0 ist nicht
+
-
*
/
Datei Bearbeiten Reiter Hilfe
abel@raspberrypi:~ $ cd /home/abel/Deskt
abel@raspberrypi:~/Desktop $ python tr_m
yment. Use a
" 200 -
" 200 -