Python Kurs, Martin Reiche 2024
Version vom 5.5.2024

Start-Infos für das Projekt „Taschenrechner“

Willst du einen Taschenrechner programmieren, hast du mehrere Möglichkeiten, z.B. für die Benutzeroberfläche:

1. Programmiere für jeden Knopf einen tkinter Button. Dann kannst du die Verfahren aus dem Projekt „Benzinrechner“ verwenden.

2. Verwende das beiliegende Skript *RechnerStarter.py*. Dort ist die Benutzeroberfläche (GUI = graphical user interface) schon programmiert.

Bei beiden Weg verzweigt es sich dann weiter: Man kann entweder

a) den in Python eingebauten eval-Befehl für die Rechnung nutzen (Probiere einmal `eval('3.1 + 4 * 5')` auf der REPL)

b) oder aber die Auswertung der Eingabe selber vornehmen.

Der schnellste und für den Einstieg empfohlene Weg ist 2a. Dort wird man mit dem ersten der nachfolgend benannten Szenarien anfangen.

Szenarien

Einfaches drauflos-Programmieren kann man einmal probieren, wird aber schnell unübersichtlich. Besser ist, die typischen Abläufe bei der Benutzung vorab schriftlich festzuhalten. Bei den folgenden Szenarien bedeutet:

- *num* = Number: Eingabe einer Zahl inkl. Dezimalpunkt und Vorzeichen
- *op* = Operator: + - * /
- *eq* = Equals: Ist-Gleich Taste
- Zahl in Klammern: Die Ergebnisanzeige im Display

Unabhängige Rechnungen

Eingabeschema: *num op num eq num op num eq*

Das Ergebnis der ersten Rechnung wird angezeigt, die zweite Rechnung erfolgt unabhängig.
Beispiel: 12 * 10 = (120) 41 + 91 = (132)

Aufbauende Rechnung

Eingabeschema: *num op num eq op num eq op num eq*

Das Ergebnis der ersten Rechnung findet Verwendung in der zweiten.
Beispiel: 7 + 2 = (9) + 3 = (12) * 5 = (60)

Wiederholung der letzten Rechnung

Eingabeschema: *num op num eq eq eq*

Beispiel: 7 + 4 = (11) = (15) = (19)

Verdopplung des ersten Operanden

Eingabeschema: *num op eq*

Beispiel: 7 * = (49) = (343) oder auch 10 – 5 = (5) * = (25) = (125) = (625)

Ketten-Rechnung ohne =

Eingabeschema: *num op num op num op num eq*

Kettenrechnung ohne „Punkt vor Strich“. Beispiel: 18 / 3 + (6) 4 * (10) 5 = (50)

Punkt vor Strich Rechnung

Eingabeschema: *num op num op num op num eq*

Beispiel: 3 + 4 * 5 = (23)

Design

Das einfachste Vorgehen im Verfahren a besteht darin, alle Eingaben zu sammeln, anzuzeigen und dann dem eval-Befehl zu übergeben, wenn die =-Taste gedrückt wird.

Will man vermeiden, dass die Operatoren im Display angezeigt werden, empfiehlt es sich, eine zweite globale String-Variable namens *expression* zu definieren, in der man den später mit *eval* auszuwertenden Ausdruck aufbaut.

Die anderen Szenarien erfordern eine aufwändigeren Ansatz. Man muss z.B. beim Drücken der = Taste nämlich unterscheiden, ob überhaupt Operanden und ein Operator eingegeben wurde. Falls nicht, sollte die Taste ignoriert werden. Dieser Ansatz gelingt mit der Einführung einer Zustandsvariablen. Dieser wird sich z.B. ändern, sobald ein Operator eingegeben wurde. Eine neue Ziffer startet dann den zweiten Operanden.

Mehr zu diesem Thema gibt es auf Nachfrage. Prinzipiell muss das Programm nämlich für jeden Tastendruck zu jeder Zeit richtig reagieren! Dies zu analysieren ist aufwändig, aber machbar, wie die käuflichen Taschenrechner beweisen. So muss man auch dafür sorgen, dass Benutzer-Fehler, wie z.B. die doppelte Eingabe des Dezimaltrenners abgefangen werden.