

# Python / Taschenrechner ⚡

## 1. Was ist Python?

Python ist eine gut lesbare Programmiersprache. Sie hat einen knappen Programmierstil. Sie ist aktuell die zweit häufigste Programmiersprache. Quelle: <https://www.heise.de/developer/meldung/Python-schreibt-Geschichte-Platz-2-im-Programmiersprachen-Ranking-4673726.html>

## 3. Beispielquellcode zum Ergänzen

```
# Import von Bibliotheken
from tkinter import *

# Initialisierung des Fensters
Fenster = Tk()

#Titel des Fensters
Fenster.title("Taschenrechner")

# Label erstellen
L1 = Label(Fenster, font="Verdana 16 bold", text="Taschenrechner")
# Positionierung im Fenster
# column = Spalte
# row = Reihe
# columnspan = Breite
L1.grid(column=1, row=1, columnspan=2)

# Zahl1
L2 = Label(Fenster, text="Zahl 1:")
L2.grid(column=1, row=2)
E1 = Entry(Fenster)
E1.grid(column=2, row=2)

# Ergebnis
L3 = Label(Fenster, text="Ergebnis:")
L3.grid(column=1, row=4)
E3 = Entry(Fenster)
E3.grid(column=2, row=4)

# Funktionen:

# Addition
def addition():
    # Feld leeren
    E3.delete(0, END)
    # Variablen zahl1 und zahl2 auslesen
    zahl1 = int(E1.get())
    zahl2 = int(E2.get())
    # Berechnung
    ergebnis = zahl1 + zahl2
    # Ausgabe
    E3.insert(0, ergebnis)

# Buttons
B1 = Button(Fenster, text="+", width=25, command = addition)
B1.grid(column=1, row=5)

Fenster.mainloop()
```

## 4. Kommentare

Alle Zeilen, die mit einem # anfangen gelten als Kommentar und werden nicht als "Befehl" interpretiert. Durch Kommentare kann man ein Programm kommentieren.

Beispiel:

```
# Ich bin ein Kommentar.
```

## 5. Bibliotheken

Das Programm braucht für die Darstellung des grafischen Fensters die Bibliothek tkinter.

```
# Import von Bibliotheken
from tkinter import *
```

# Python / Taschenrechner ⚡

## 6. Initialisierung des Fensters

Durch folgenden Zeile wird das **Fenster** initialisiert.

```
# Initialisierung des Fensters
Fenster = Tk()
```

## 7. Titel des Fensters

Durch folgende Zeile wird die Überschrift des Fensters gesetzt.

```
# Titel des Fensters
Fenster.title("Taschenrechner")
```

**Aufgabe:** Ändere den Titel des Fensters in "Taschenrechner von MEINNAME".

## 8. Label und Formatierung

Das Programm braucht für die Darstellung des grafischen Fensters die Bibliothek tkinter.

```
# Label erstellen
L1 = Label(Fenster, font="Verdana 16 bold", text="Taschenrechner")
# Positionierung im Fenster
# column = Spalte
# row = Reihe
# columnspan = Breite
L1.grid(column=1, row=1, columnspan=2)
```

Durch `font="Verdana 16 bold"` kann ich die Schriftart und -größe eines Labels einstellen.

**Aufgabe:** Ändere die Schriftgröße auf 20.

Durch `L1.grid( ... )` kann ich ein Fensterelement positionieren. Mit `row` bestimme ich die Reihe, mit `column` die Spalte. Mit `columnspan` definiere ich die Breite eines Elements. Dies brauche ich, wenn ein Element über zwei Spalten geht.

## 9. Label und Texteingabe für Zahl1, Zahl2 und Ergebnis

Durch folgende Zeilen wird ein Label und eine Textfeld für Zahl1 erstellt.

```
# Zahl1
L2 = Label(Fenster, text="Zahl 1:")
L2.grid(column=1, row=2)
E1 = Entry(Fenster)
E1.grid(column=2, row=2)
```

Mit `Entry(Fenster)` erstelle ich eine Texteingabe.

## 10. Buttons

Mit den Buttons kann ich mit dem Programm interagieren.

```
# Buttons
B1 = Button(Fenster, text="+", width=25, command = addition)
B1.grid(column=1, row=5)
```

Der Button hat mehrere Attribute. Diese werden jeweils durch ein Komma getrennt.

Der Button wird dem Fenster zugeordnet: `B1 = Button(Fenster,`

Erhält als Text "+": `text="+",`

Hat eine Breite von 25 Pixeln: `width=25,`

Beim Drücken wird die Funktion `__addition__` aufgerufen: `command = addition`

## 11. Funktion Addition

Für das Berechnen der Addition erhält das Programm eine Funktion(ein Unterprogramm).

```
# Funktionen:
# Addition
def addition():
    # Feld leeren
    E3.delete(0, END)
    # Variablen zahl1 und zahl2 auslesen
    zahl1 = int(E1.get())
    zahl2 = int(E2.get())
```

## Python / Taschenrechner ⚡

```
# Berechnung
ergebnis = zahl1 + zahl2
# Ausgabe
E3.insert(0, ergebnis)
```

Eine Funktion wird durch folgende Zeile erstellt: `def addition():`

Es ist wichtig die `()` und den `:` hinter den Funktionsnamen zu schreiben.

### **12. Aufgabenstellung**

Programmiere einen Taschenrechner, der neben der Addition auch `-`, `*` und geteilt kann.