# 2. Exception "fangen"

In der ersten Übung wurde aufgezeigt, dass der Umgang mit Laufzeitfehlern - wie z.B. der Dision durch 0 - durch das Exception-Handling geregelt werden kann.

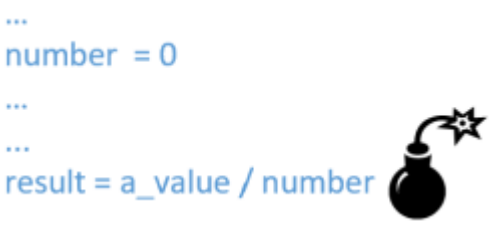



*Abb. 3.1:* Die „Fehlerbombe“
Damit ein Programm geordnet auf eine Ausnahme (Exception) reagieren kann, braucht es ein entsprechendes Konstrukt. Dies wird bei Python durch

```
try:
  # hier folgt der kritische Code, der potentielle Fehlersituationen haben kann.
  # typischerweise im Zusammenhang mit Benutzereingaben.
except:
  # hier folgt die Behandlung des Fehlers
else:
  # hier folgt Code, der nur dann ausgeführt wird, wenn es keine Exception gibt
finally:
  # hier folgt Code der sowohl als auch ausgeführt wird.
```

sichergestellt. Bildlich dargestellt sieht der Sachverhalt wie folgt aus:

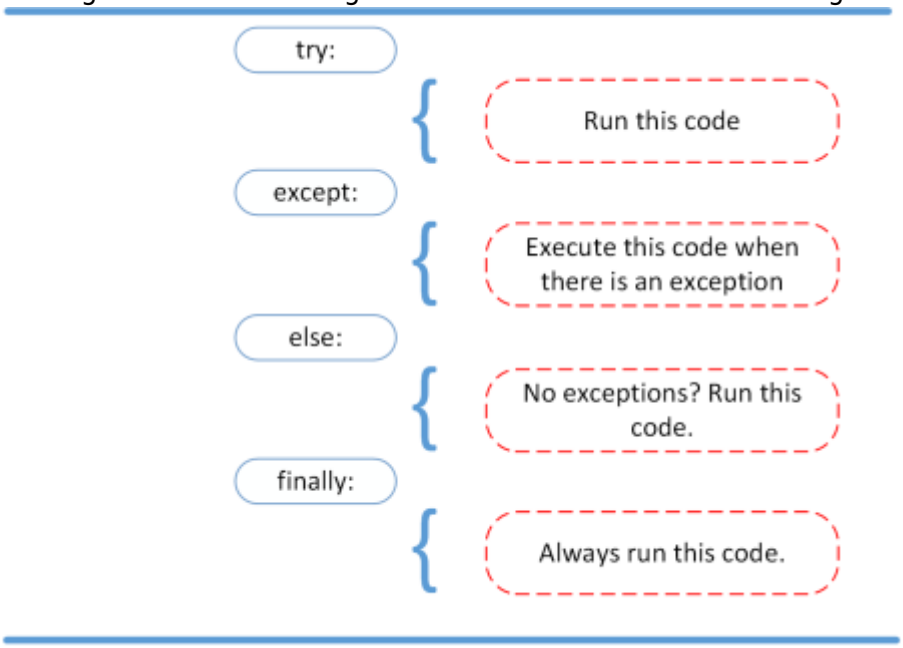



*Abb. 3.2:* try-except Klausel

Können in einem Codeabschnitt mehrere potentielle Exceptions auftreten, so können diese gezielt herausgepickt werden. Dazu wird der `except`-Befehl parametriert.

//Beispiel 3.1: Ein einfache Berechnung //

```
if __name__ == '__main__':

    try:
        x = float(input('Your number: '))
        inverse = 1.0 / x
    except ValueError:
        print('You should have given either an int or a float')
    except ZeroDivisionError:
        print('Divison bei zero')
    else:
        print('well done')
    finally:
        print('There may or may not have been an exception.')
```

Wird im Code aus Beispiel 3.1 ein Buchstanben (z.B. A) eingegeben, resultiert ein `ValueError` und es wird der entsprechende `except` ausgeführt.
Wird 0 eingegeben, wird ein `ZeroDivisionError` geworfen und dann der entsprechende `except`-Pfad ausgeführt.
**Tipp**: Kopieren Sie sich das Code–Schnippsel in Ihre Entwicklungsumgebung und probieren Sie das Verhalten aus.
Welche Fehlerfälle von der jeweiligen Programmiersprache unterstützt werden, ist der Sprachdokumentation zu entnehmen. Typische Problemfälle sind

- I/O-Operationen (z.B. Dateilhandling, DB-Zugriffe usw.)
- Indexfehler bei Array, Listen usw.
- Arithmetische Fehler (z.B. Division mit 0)
- Fehler bei Datentypen und Formaten (sofern eine Typenprüfung stattfindet)
- Speicherfehler
- ...

Was aber passiert bei Situationen, bei denen aus der Anwendungsdomäne heraus zur Laufzeit Fehler auftreten können? Das wird uns im nächsten Kapitel beschäftigen.

© René Probst

From:
https://wiki.bzz.ch/ - **BZZ - Modulwiki**

Permanent link:
**https://wiki.bzz.ch/modul/archiv/m320/learningunits/lu03/theorie/lu4-kapitel_2_v2**

Last update: **2026/08/30 14:09**