

„Is the pipeline literally running from your laptop?“ „Don't be silly, my laptop disconnects far too often to host a service we rely on. It's running on my phone.“

7. Benutzereingabe

Wenn wir die Rolle vom Programmierer zum **Benutzer** eines Programms wechseln, müssen wir die Möglichkeit haben, die Verarbeitung eines Programms zu beeinflussen. Ein Benutzer ist kein Programmierer und soll den Code **nicht** ändern müssen. Dies wird in der Programmierung durch eine Benutzereingabe realisiert. Bei der Turtle-Programmierung können wir uns darunter die Auswahl einer Farbe durch den Benutzer vorstellen. Die Lernziele lauten:

- ☐ Sie erstellen ein Programm, das dem Benutzer die Eingabe eines Floats ermöglicht.
- ☐ Sie erstellen ein Programm, das dem Benutzer die Eingabe eines Strings ermöglicht.
- ☐ Sie erstellen ein Programm, das dem Benutzer die Eingabe eines Integers ermöglicht.

7.1 Die Benutzereingabe von Floats

Mit der Funktion `numinput` des Turtle-Moduls¹ kann der Benutzer zur Eingabe einer **Zahl** aufgefordert werden. Listing 18 zeigt ein Beispiel. In Zeile drei wird der Funktionsaufruf durchgeführt.

```
1 import turtle as t
2
3 a = t.numinput("Quadrat", "Bitte geben Sie die Seitenlänge ein:")
4 for i in range(4):
5     t.forward(a)
6     t.left(90)
7 t.done()
8
```

Listing 18: Wenn wir das Programm ausführen, dann öffnet sich ein Fenster für die Eingabe.

Mit der Funktion `numinput` kann der Benutzer eine **beliebige Zahl** eingeben.

Wichtig! Wenn wir ein Programm mit einem `numinput`-Funktionsaufruf ausführen, dann wird das Programm in der Zeile mit dem `numinput`-Funktionsaufruf **automatisch angehalten**. Es geht erst dann **weiter**, wenn der Benutzer die Eingabe mit „OK“ bestätigt. ■

Die Benutzereingabe können wir dann in einer Variablen speichern. In Listing 18 wird die Eingabe in der Variablen `a` gespeichert. Wenn wir `numinput` verwenden, dann wird **immer ein Float** erzeugt. Auch dann, wenn wir „eine Zahl ohne Punkt“ eingeben.

¹`numinput` ist vermutlich eine Abkürzung für *numerical input*.

7.1.1 Argumente der `numinput`-Funktion

Die `numinput`-Funktion wird typischerweise mit zwei Argumenten aufgerufen:

- Das **erste Argument** bestimmt den **Titel** des Fensters.
- Das **zweite Argument** bestimmt den **Inhalt** des Fensters.

Beide Argumente müssen ein **String** sein.

■ **Beispiel 18** In Abbildung 9 ist das Fenster zum Code aus Listing 18 (Zeile 3) gezeigt. ■

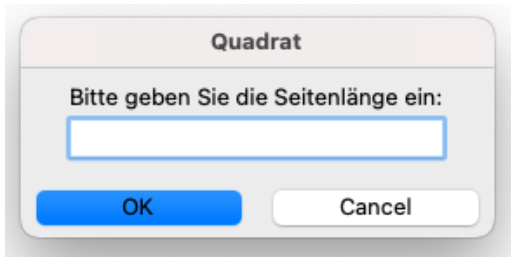


Abbildung 9: Fenster für die Benutzereingabe. Der Screenshot wurde unter macOS gemacht.²

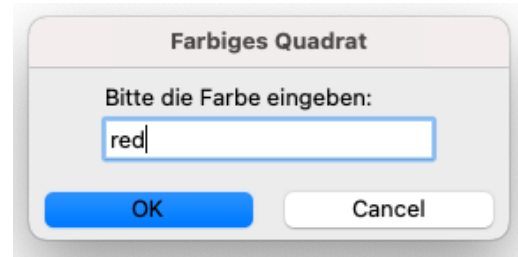


Abbildung 10: Bei einer Texteingabe dürfen die doppelten Anführungszeichen nicht eingegeben werden.³

7.2 Die Benutzereingabe von Strings

Mit der Funktion `textinput` des Turtle-Moduls kann der Benutzer zur Texteingabe aufgefordert werden. Listing 19 zeigt ein Beispiel. In Zeile drei wird der Funktionsaufruf durchgeführt.

```
1 import turtle as t
2
3 farbe = t.textinput("Farbiges Quadrat", "Bitte die Farbe eingeben:")
4 t.pencolor(farbe)
5 for _ in range(4):
6     t.forward(200)
7     t.left(90)
8 t.done()
9
```

Listing 19: Das Konzept für die Benutzereingabe eines Strings ist praktisch identisch mit dem Konzept für die Benutzereingabe eines Floats.

Auch bei `textinput` wird das Programm angehalten und der Benutzer muss mit „OK“ die Eingabe bestätigen. Die Argumente der `textinput` sind identisch zur `numinput`-Funktion.

Wichtig! Wenn wir mit `textinput` etwas eintippen, dann benötigen wir für die Eingabe **keine doppelten Anführungszeichen**. ■

In Listing 19 speichert die Variable `farbe` somit nach der Eingabe einen **String**.

■ **Beispiel 19** In Abbildung 10 ist das Fenster zum Code aus Listing 19 (Zeile 3) gezeigt. Die Texteingabe erfolgt **ohne** doppelte Anführungszeichen. ■

Wir können natürlich auch eine Texteingabe mit der Benutzereingabe für eine Zahl kombinieren. Es ist auch möglich, mehrere Benutzereingaben einzubauen oder eine wiederholte Benutzereingabe in einer Schleife zu verwenden.

²Bildquelle: Screenshot des Programms.

³Bildquelle: Screenshot des Programms.

7.3 Die Benutzereingabe von Integers

Wollen wir den Benutzer nach einer ganzen Zahl fragen, dann benötigen wir die `numinput`-Funktion in Kombination mit der `int`-Funktion.

```

1 import turtle as t
2 import random as r
3
4 # Erste Möglichkeit
5 eingabe = t.numinput("Zufallsquadrat", "Minimale Seitenlänge:")
6 min_laenge = int(eingabe)
7
8 # Zweite Möglichkeit
9 max_laenge = int(t.numinput("Zufallsquadrat", "Maximale Seitenlänge:"))
10
11 seitenlaenge = r.randrange(min_laenge, max_laenge+1)

```

Listing 20: Wir können eine Variable als Zwischenspeicher für die Eingabe verwenden oder die Funktionsaufrufe verschachteln (Auszug aus dem Quellcode).

Die `int`-Funktion konstruiert aus einem Float einen Integer. Bei der Konstruktion werden alle Nachkommastellen ignoriert und nur die **Zahl vor dem Komma** wird zur Erzeugung des Integers verwendet. Es wird **nicht** gerundet.

Wichtig! Auch wenn Sie in der Benutzereingabe einen Float mit einem Punkt eingeben, die `int`-Funktion konstruiert daraus einen Integer und schneidet alle Nachkommastellen „ab“. ■

7.3.1 Verschachtelte Funktionsaufrufe

Wir können zwei Funktionsaufrufe verschachteln, wenn der **erste Funktionsaufruf** einen Wert erzeugt. Wir sagen, die **Funktion gibt einen Wert zurück**. Wir können dann den **Funktionsaufruf als Argument** für den **zweiten Funktionsaufruf** verwenden.

■ **Beispiel 20** Listing 21 zeigt ein paar verschachtelte Funktionsaufrufe.

```

1 import turtle as t
2 import random as r
3
4 t.pencolor(r.choice(["red", "green", "blue"]))
5 t.pensize(r.randrange(1, 11))
6 seitenlaenge = int(t.numinput("Eingabe", "Seitenlänge:"))
7


```

Listing 21: Der innere Funktionsaufruf erzeugt jeweils einen Wert. ■

 **Clean Code** — **Verschachtelte Funktionsaufrufe.** Verschachtelte Funktionsaufrufe mit Vorsicht verwenden! Das Programm **muss korrekt und lesbar** bleiben.

7.3.2 Eingebaute Funktionen

Damit wir die `int`-Funktion verwenden können, benötigen wir **keine import**-Anweisung. Die `int`-Funktion steht jederzeit und überall zur Verfügung. Es ist eine so genannte **eingebaute Funktion** (engl. built-in function). Python stellt diese Funktionen automatisch zur Verfügung. Es gibt ungefähr 70 eingebaute Funktionen.

 Im Kapitel über Schleifen haben wir bereits eine eingebaute Funktion gesehen – die `range`-Funktion. Auch diese Funktion können wir ohne `import`-Anweisung verwenden.