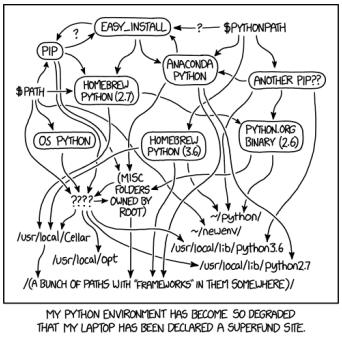
PYTHON ENVIRONMENT https://xkcd.com/1987/



The Python environmental protection agency wants to seal it in a cement chamber, with pictorial messages to future civilizations warning them about the danger of using sudo to install random Python packages.

9. Parameter

Wir kennen bereits Funktionen, die beim Funktionsaufruf ein Argument benötigen.

■ **Beispiel 24** Die Funktion `forward` wird mit einem Argument aufgerufen. Das Argument ist ein Integer. Bei `forward(100)` ist das Argument der Integer `100`. ■

Damit wir beim Funktionsaufruf ein Argument übergeben können, müssen wir bei der Funktionsdefinition einen **Parameter** definieren. Dies schauen wir uns nun an. Die Lernziele lauten:

- ☐ Sie erklären das Prinzip der Parameter anhand eines Beispiels.
- ☐ Sie erstellen Ihre eigene Funktion mit einem oder mehreren Parametern.
- ☐ Sie rufen eine eigene Funktion mit einem oder mehreren Parametern auf.
- ☐ Sie verwenden Funktionen mit Parametern, um Code bei einem Mausereignis auszuführen.

9.1 Vier Quadrate

Wir können mit dem Programm aus Listing 27 die Figur aus Abbildung 13 zeichnen.

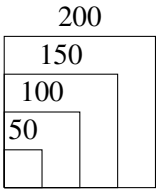


Abbildung 13: Das Programm aus Listing 27 zeichnet vier unterschiedlich grosse Quadrate.

Wir können die Seitenlänge der Quadrate auf einfache Weise anpassen, indem wir bei der Funktionsdefinition einen Parameter notieren (Zeile 4). Ein Parameter ist eine spezielle Variable. Beim Funktionsaufruf können wir dann ein Argument übergeben (Zeile 10, Zeile 11 und Zeile 12).

```
1  import turtle as t
2
3
4  def quadrat(seitenlaenge):
5      for _ in range(4):
6          t.fd(seitenlaenge)
7          t.lt(90)
8
9
10 def quadrate():
11     quadrat(50)
12     quadrat(100)
13     quadrat(150)
14     quadrat(200)
15
16
17 quadrate()
18 t.done()
19
```

Listing 27: `vier_quadrate.py`

9.2 Parameter definieren

Wir können in der Funktionsdefinition **zwischen** den Klammern Parameter definieren.

Definition 13 — Parameter. Ein Parameter ist eine spezielle **Variable** der Funktion. Die Variable steht nur innerhalb der Funktionsdefinition zur Verfügung. Ein Parameter ist also eine **lokale Variable**. Wir definieren einen oder mehrere Parameter bei der Funktionsdefinition **zwischen** den runden Klammern. **Mehrere Parameter** werden durch **Kommata** getrennt.

■ **Beispiel 25** In Listing 27 wird in Zeile vier bei der Funktionsdefinition ein Parameter verwendet. Der Parametername lautet `seitenlaenge`. ■

Wichtig! Beim Funktionsaufruf muss für **jeden** Parameter ein **Argument** vorhanden sein. Man muss jedem Parameter einen Wert übergeben. ■

9.2.1 Wie wählen wir Parameternamen?

Im Rahmen der üblichen Regeln sind wir bei der Wahl des Parameternamens grundsätzlich frei. Es wird jedoch empfohlen, den Parameternamen so zu wählen, dass allein aus dem Parameternamen ersichtlich ist, wofür der Parameter verwendet wird.

Clean Code — Sinnvolle Parameternamen. Wir wählen die **Parameternamen** so, dass wir sofort verstehen, wofür der Parameter verwendet wird.

Wichtig! Hat eine Funktionsdefinition mehrere Parameter, so muss jeder Parametername eindeutig sein. ■

Clean Code — Snake Case für Parameternamen. Da ein Parameter eine lokale **Variable** darstellt, verwenden wir auch für **Parameternamen** die Snake Case-Notation.

9.3 Variablen beim Funktionsaufruf

Wir können beim Funktionsaufruf auch eine **Variable** verwenden. Listing 28 zeigt, wie wir die Figur aus Abbildung 13 mit einer for-Schleife zeichnen können.

```
1  import turtle as t
2
3
4  def quadrat(seitenlaenge):
5      for _ in range(4):
6          t.fd(seitenlaenge)
7          t.lt(90)
8
9
10 def quadrate():
11     a = 50
12     for _ in range(4):
13         quadrat(a)
14         a = a + 50
15
16
17 quadrate()
18 t.done()
19
```

Listing 28: Variable beim Funktionsaufruf (`vier_quadrate_variante.py`).

9.4 Mehrere Parameter

Wir können beliebig viele Parameter für eine Funktion definieren.

Wichtig! Beim Funktionsaufruf spielt die **Reihenfolge der Parameter** eine Rolle: Die Argumente müssen in der gleichen Reihenfolge wie die Parameter angegeben werden. ■

■ **Beispiel 26** Listing 29 zeigt ein Beispiel mit zwei Parametern. Der erste Parameter der Funktion `quadrat` bestimmt die Seitenlänge des Quadrats. Der zweite Parameter bestimmt die Linienfarbe.

```
1  import␣turtle␣as␣t
2  import␣random␣as␣r
3
4
5  def␣quadrat(seitenlänge,␣farbe):
6  ␣␣␣␣t.pencolor(farbe)
7  ␣␣␣␣for␣_␣in␣range(4):
8  ␣␣␣␣␣␣␣␣t.fd(seitenlänge)
9  ␣␣␣␣␣␣␣␣t.lt(90)
10
11
12 def␣quadrate():
13 ␣␣␣␣for␣_␣in␣range(4):
14 ␣␣␣␣␣␣␣␣farbe␣=␣r.choice(["red",␣"green",␣"blue"])
15 ␣␣␣␣␣␣␣␣quadrat(50,␣farbe)
16 ␣␣␣␣␣␣␣␣t.pu()
17 ␣␣␣␣␣␣␣␣t.fd(125)
18 ␣␣␣␣␣␣␣␣t.pd()
19
20
21 quadrate()
22 t.done()
23
```

Listing 29: Funktion mit zwei Parametern (`vier_bunte_quadrate.py`).

■

Es dürfen auch mehrere Funktionen definiert werden, welche mehrere Parameter besitzen.

Clean Code — Leerzeichen 7. Mehrere **Parameter** werden durch ein **Komma** getrennt. **Nach** einem Komma notieren wir ein **Leerzeichen**.

9.5 Mausereignisse verarbeiten

Mit der Funktion `onscreenclick` registrieren wir eine **Funktion**, die beim Klicken der Maustaste einmalig ausgeführt wird. Wird die Maustaste zweimal hintereinander gedrückt, wird die Funktion zweimal hintereinander ausgeführt. Die Funktion `onscreenclick` benötigt nur ein Argument beim Funktionsaufruf. Es muss der **Funktionsname** übergeben werden, damit beim Mausklick die richtige Funktion aufgerufen wird.

Wichtig! Das Argument ist **nur der Funktionsname** - keine runden Klammern! ■

Damit die Mausereignisse verarbeitet werden, müssen wir **nach der Registrierung einmal** die `listen`-Funktion aufrufen. Bei der **Funktionsdefinition** zur Verarbeitung des Mausklicks müssen

zwei Parameter notiert werden. Der erste Parameter speichert die **x-Koordinate** und der zweite Parameter die **y-Koordinate** des Mausklicks. Diese können wir dann in der Funktion verwenden.

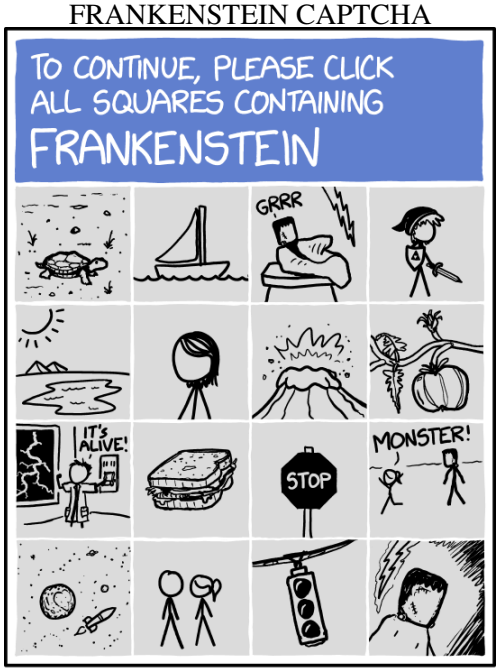
■ **Beispiel 27** Listing 30 zeigt ein Beispiel, wie wir für jeden Mausklick ein Quadrat zeichnen können. Die Parameter `x` und `y` speichern die Koordinaten des Mausklicks. Wir bewegen dann die Turtle zu dieser Position und zeichnen dort das Quadrat mit der Seitenlänge 50.

```
import turtle as t


def quadrat_50(x, y):
    t.pu()
    t.goto(x, y)
    t.pd()
    for _ in range(4):
        t.fd(50)
        t.lt(90)


t.onscreenclick(quadrat_50)
t.listen()
t.done()
```

Listing 30: Bei jedem Mausklick wird die Funktion einmal ausgeführt. (`quadrate.py`).

■

FRANKENSTEIN CAPTCHA



OH NO.

The distinction between a ship and a boat is a line drawn in water.[1]

[1] https://xkcd.com/2604/