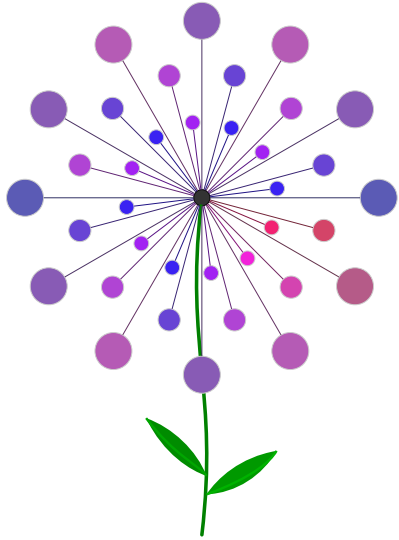


FreeFlower

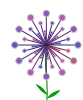


Informatik: Programmieren

Kapitel 04: **Eigene Funktionen definieren**

Lernziele

- Sie können erklären, was eine Funktion in der Programmierung ist.
- Sie können die Vorteile von Funktionen erläutern (z.B. Wiederverwendbarkeit, Modularität).
- Sie können die Syntax zur Definition einer eigenen Funktion beschreiben.
- Sie können eine selbst definierte Funktion aufrufen.



Eigene Funktionen definieren

```
import turtle as t

# zeichne ein Quadrat
for _ in range(4):
    t.fd(100)  # Funktion
    t.rt(90)   # Funktion

t.done()
```



Eigene Funktionen definieren

```
import turtle as t

# zeichne ein Quadrat
for _ in range(4):
    t.fd(100)  # Funktion
    t.rt(90)   # Funktion

t.done()
```

Neue Funktion für alles?



Eigene Funktionen definieren

```
import turtle as t

def quadrat():
    for _ in range(4):
        t.fd(100)
        t.rt(90)

quadrat()
t.rt(45)
quadrat()

t.done()
```

Funktion wird `def`iniert 📄

- Definition = `def` = Funktion
- Ähnlich einem Kochrezept 📄
  - Beschreibung der auszuführenden Schritte
  - Wird erst ausgeführt, wenn aufgerufen! 👩‍🍳



Eigene Funktionen definieren

```
import turtle as t

def quadrat():
    for _ in range(4):
        t.fd(100)
        t.rt(90)

quadrat()
t.rt(45)
quadrat()

t.done()
```

Funktion wird `def`iniert 📄

Aufruf 👨‍🍳

Aufruf 👨‍🍳

- Definition = `def` = Funktion
- Ähnlich einem Kochrezept 📄
  - Beschreibung der auszuführenden Schritte
  - Wird erst ausgeführt, wenn aufgerufen! 👨‍🍳

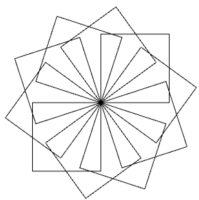


Viereck-Blume

```
import turtle as t

for _ in range(10):
    for _ in range(4):
        t.fd(100)
        t.rt(360 / 4)
    t.rt(360 / 10)

t.done()
```



```
import turtle as t

def viereck():
    for _ in range(4):
        t.fd(100)
        t.rt(360 / 4)

def viereck_blume():
    for _ in range(10):
        viereck()
        t.rt(360 / 10)

viereck_blume()

t.done()
```



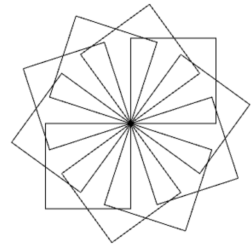
Neue Funktionen definieren

```
import turtle as t

def viereck():
    for _ in range(4):
        t.fd(100)
        t.rt(360 / 4)

def viereck_blume():
    for _ in range(10):
        viereck()
        t.rt(360 / 10)

viereck_blume()
```



- **Modularität**: Zerlegung des Programms in kleinere Stücke
- **Übersichtlicher**!



Neue Funktionen definieren

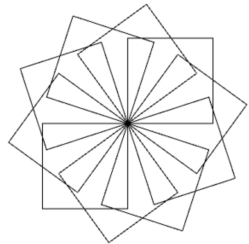
```
import turtle as t

def viereck():
    for _ in range(4):
        t.fd(100)
        t.rt(360 / 4)

def viereck_blume():
    for _ in range(10):
        viereck()
        t.rt(36)

viereck_blume()
```

Aufruf



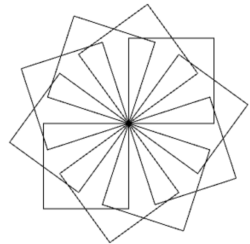
- **Modularität**: Zerlegung des Programms in kleinere Stücke
- **Übersichtlicher**!

FreeFlower

Neue Funktionen definieren

```
import turtle as t

def viereck():
    for _ in range(4):
        t.fd(100)
        t.rt(3
                    Aufruf
def viereck_blume():
    for _ in range(10):
        viereck()
        t.rt(3
                    Aufruf
viereck_blume()
```



- **Modularität**: Zerlegung des Programms in kleinere Stücke
- **Übersichtlicher**!



Funktionen & Schleifen

```
import turtle as t

for _ in range(4):
    for _ in range(9):
        t.fd(4)
        t.lt(10)
    t.fd(100)

t.done()
```



```
import turtle as t

def viertelkreis():
    # genauer: ein 1 / 4 eines
    regelmässigen 36-Ecks
    for _ in range(9):
        t.fd(4)
        t.lt(10)

def abgerundetes_quadrat():
    for _ in range(4):
        viertelkreis()
        t.fd(100)

abgerundetes_quadrat()

t.done()
```

Flower

Funktionen & Schleifen

```
import turtle as t

for _ in range(4):
    for _ in range(9):
        t.fd(4)
        t.lt(10)
    t.fd(100)

t.done()
```



```
import turtle as t

def viertelkreis():
    # genauer: ein 1 / 4 eines
    regelmässigen 36-Ecks
    for _ in range(9):
        t.fd(4)
        t.lt(10)

def abgerundetes_quadrat():
    for _ in range(4):
        viertelkreis()
        t.fd(100)

abgerundetes_quadrat()

t.done()
```

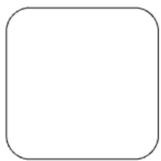
Flower

Funktionen & Schleifen

```
import turtle as t

for _ in range(4):
    for _ in range(9):
        t.fd(4)
        t.lt(10)
    t.fd(100)

t.done()
```



```
import turtle as t

def viertelkreis():
    # genauer: ein 1 / 4 eines
    regelmässigen 36-Ecks
    for _ in range(9):
        t.fd(4)
        t.lt(10)

def abgerundetes_quadrat():
    for _ in range(4):
        viertelkreis()
        t.fd(100)

abgerundetes_quadrat()

t.done()
```

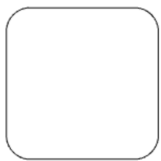
Flower

Funktionen & Schleifen

```
import turtle as t

for _ in range(4):
    for _ in range(9):
        t.fd(4)
        t.lt(10)
    t.fd(100)

t.done()
```



```
import turtle as t

def viertelkreis():
    # genauer: ein 1 / 4 eines
    regelmässigen 36-Ecks
    for _ in range(9):
        t.fd(4)
        t.lt(10)

def abgerundetes_quadrat():
    for _ in range(4):
        viertelkreis()
        t.fd(100)

abgerundetes_quadrat()

t.done()
```

Übersichtlichkeit

- Innere Schleifen → Funktionen

Zusammenfassung

Nutzen von **Funktionen**

- Problem in **Teilprobleme** gliedern



Zusammenfassung

Nutzen von **Funktionen**

- Problem in **Teilprobleme** gliedern
- **Übersichtlichkeit** erhöhen



Zusammenfassung

Nutzen von **Funktionen**

- Problem in **Teilprobleme** gliedern
- **Übersichtlichkeit** erhöhen
- Im „Hauptprogramm“ Funktionen zusammensetzen



Auftrag

- Code in VS Code schreiben und auf Moodle abgeben
- „Aufgaben Funktionen 1“ auf Moodle
- Zeit: 25 Minuten
- Danach: Gemeinsame Besprechung der Aufgabe mit den Pfeilen

