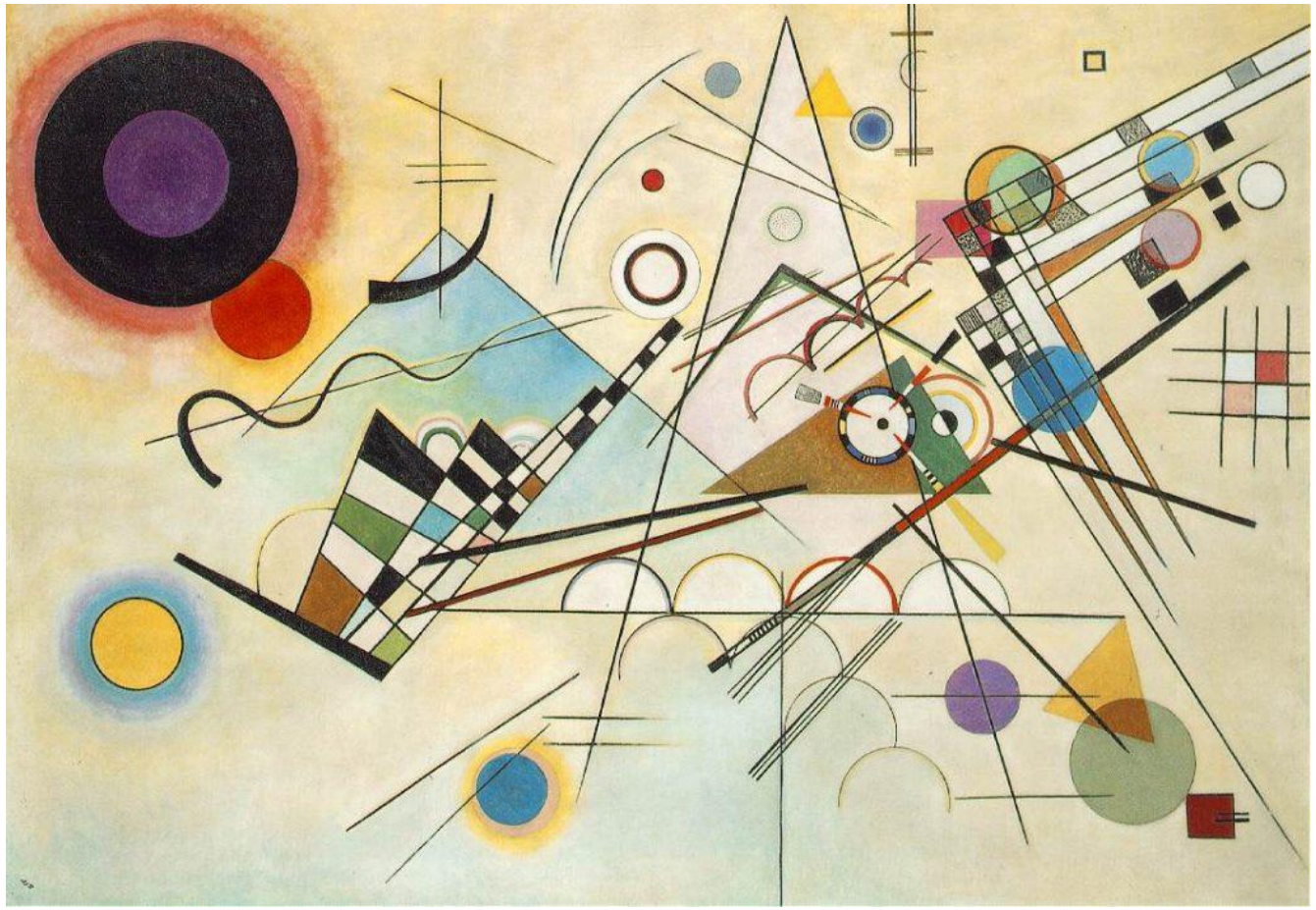
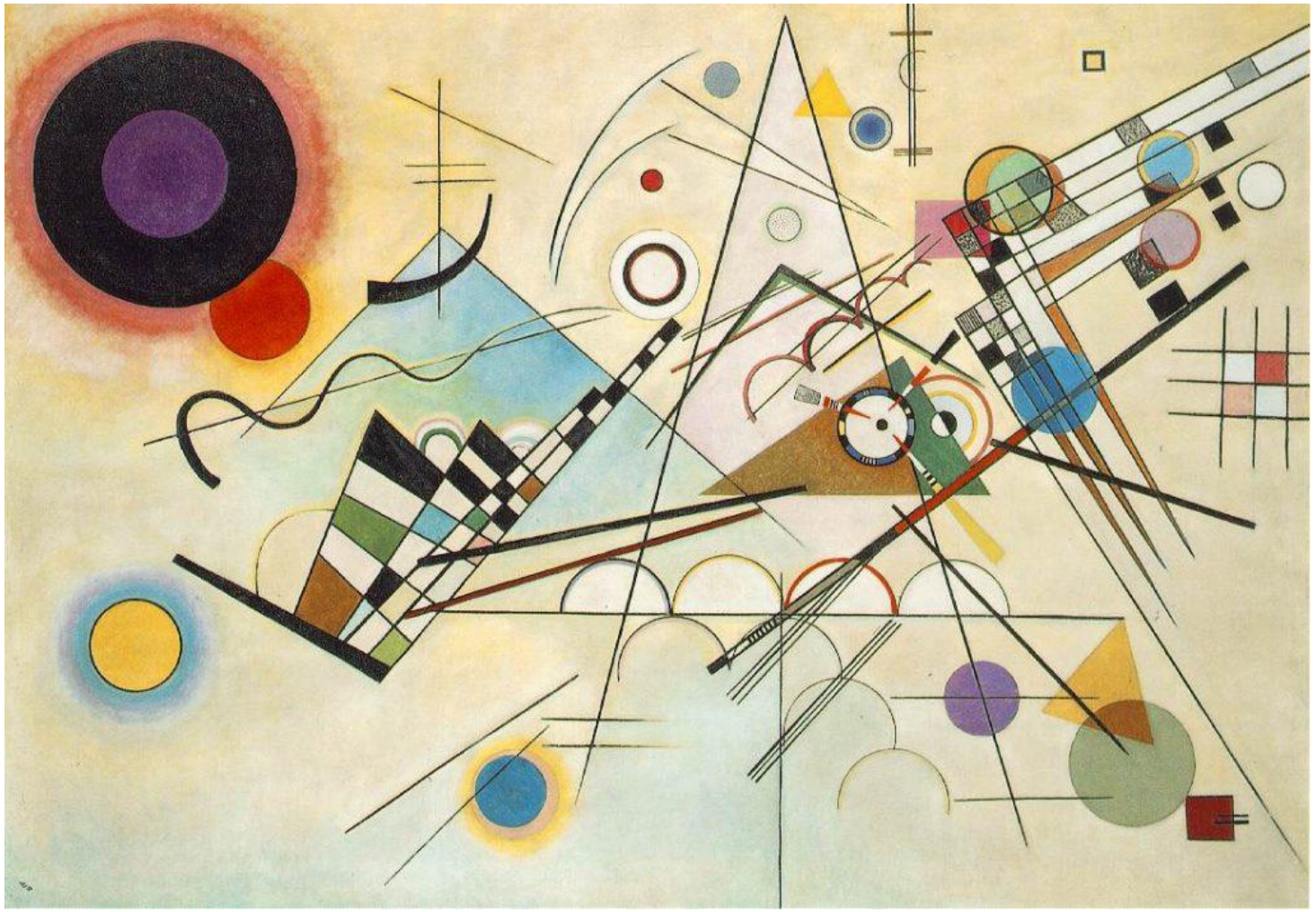


Funktionen

Wassily Kandinsky, Komposition 8, 1929

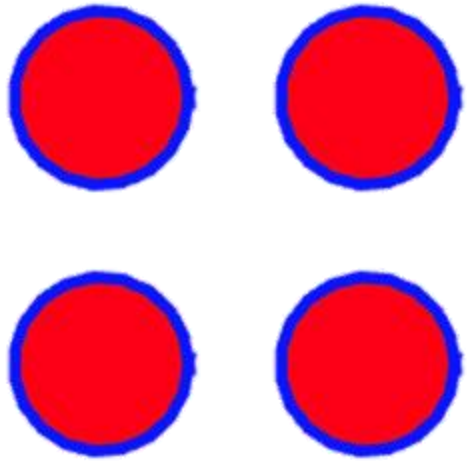


Moderne Kunst mit GTurtle

```
1  from gturtle import *
2
3  turi = Turtle()
4
5  turi.setPenColor("blue") # Stiftfarbe: blau
6  turi.setFillColor("red") # Farbe zum ausfüllen: rot
7  turi.setPenWidth(10) # Stiftdicke: 10
8
9  turi.setPos(-50,50) # Turtle an Koordinate (-50,50) beamen
10 turi.startPath() # starte neuen Pfad
11 turi.leftArc(30,360) # Kreis mit Radius 30px
12 turi.fillPath() # Fläche seit startPath() wird eingefärbt
13
14 turi.hideTurtle() # Turtle nicht anzeigen
```

Auftrag I

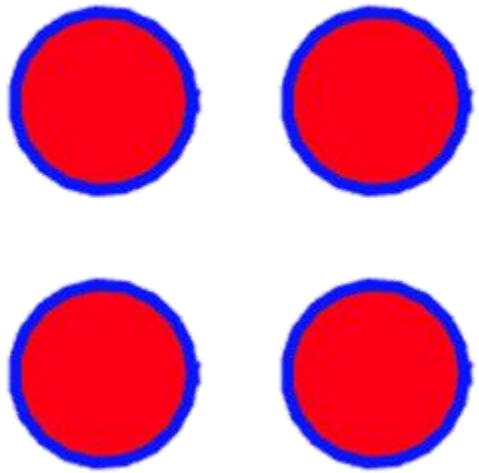
- Reproduziere folgende Figur



```
from gturtle import *

turi = Turtle()

turi.setPenColor("blue") # Stiftfarbe: blau
turi.setFillColor("red") # Farbe zum ausfüllen: rot
turi.setPenWidth(10) # Stiftdicke: 10

turi.setPos(-50,50) # Turtle an Koordinate (-50,50) beamen
turi.startPath() # starte neuen Pfad
turi.leftArc(30,360) # Kreis mit Radius 30px
turi.fillPath() # Fläche seit startPath() wird eingefärbt

turi.hideTurtle() # Turtle nicht anzeigen
```

Auftrag I

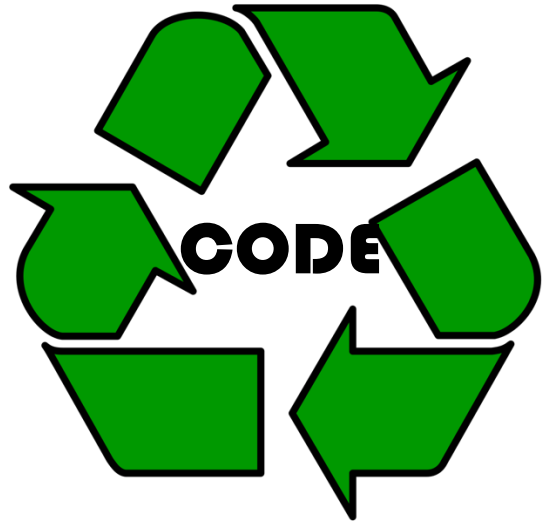
- 'Problem’ mit Code?
- Codewiederholungen: 4x fast identischer Code
- Einziger Unterschied:. x & y Koordinaten
- Lösung: **Funktionen**!



```
1  from gturtle import *
2
3  turi = Turtle()
4
5  turi.setPenColor("blue")
6  turi.setFillColor("red")
7  turi.setPenWidth(10)
8
9  turi.setPos(-50,-50)
10 turi.startPath()
11 turi.leftArc(30,360)
12 turi.fillPath()
13
14 turi.setPos(-50,50)
15 turi.startPath()
16 turi.leftArc(30,360)
17 turi.fillPath()
18
19 turi.setPos(50,-50)
20 turi.startPath()
21 turi.leftArc(30,360)
22 turi.fillPath()
23
24 turi.setPos(50,50)
25 turi.startPath()
26 turi.leftArc(30,360)
27 turi.fillPath()
28
29 turi.hideTurtle()
```

Funktionen

- **Funktionen**, auch **Unterprogramme** genannt, sind *Codeblöcke*, die man *wiederholt aufrufen* kann.
- Vorteil:
  - **Codewiederholungen vermeiden**, da Code ‘recyclen’ kann.
  - Code besser **strukturieren**



Funktionen

Definition von Funktion beginnt immer mit Schlüsselwort keyword **def**

Name der Funktion, selbst passenden Namen wählen

Inputs / Argumente: Variablen für Werte, die der Funktion übergeben werden

Doppelpunkt

```
def circle(x,y):
    turi.setPos(x,y)
    turi.startPath()
    turi.leftArc(30,360)
    turi.fillPath()
```

Codeblock der Funktion, eingerückt

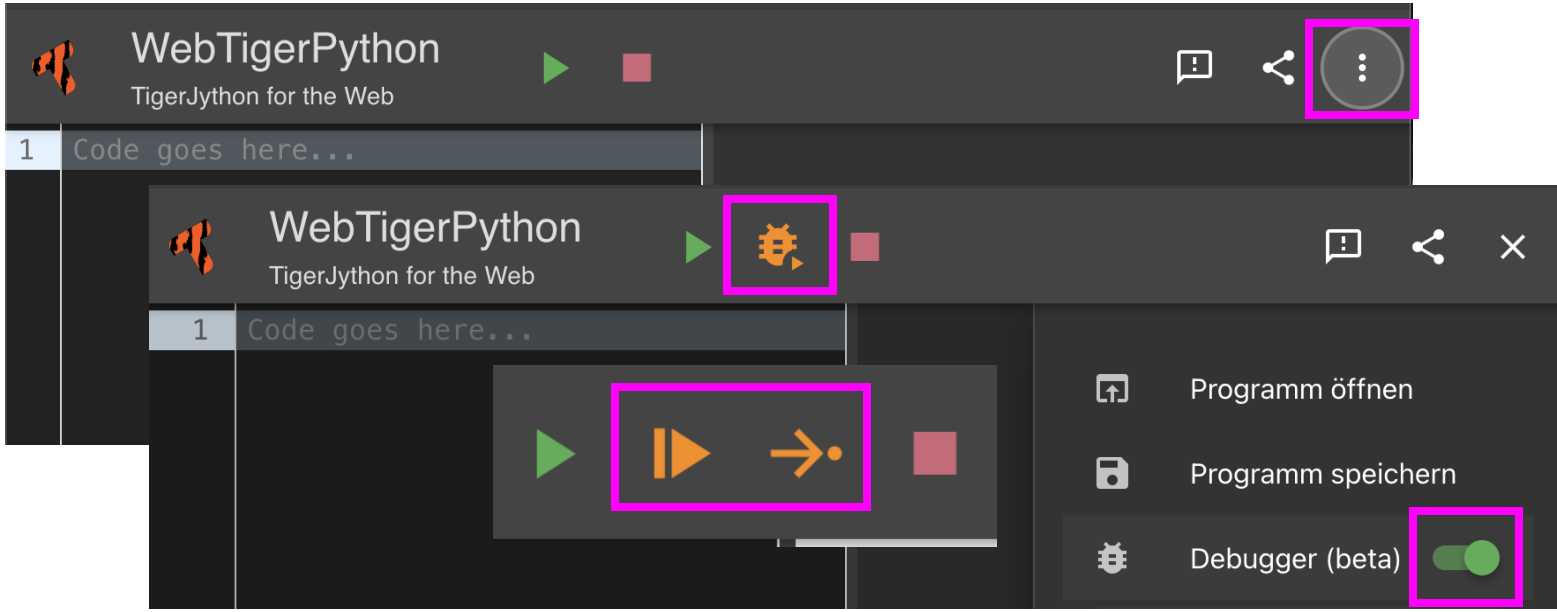
-> Mit **Debugger** Code durchgehen

```
1  from gturtle import *
2
3  turi = Turtle()
4
5  turi.setPenColor("blue")
6  turi.setFillColor("red")
7  turi.setPenWidth(10)
8
9  def circle(x,y):
10     turi.setPos(x,y)
11     turi.startPath()
12     turi.leftArc(30,360)
13     turi.fillPath()
14
15 circle(-50,-50)
16 circle(-50,50)
17 circle(50,-50)
18 circle(50,50)
19
20 turi.hideTurtle()
```

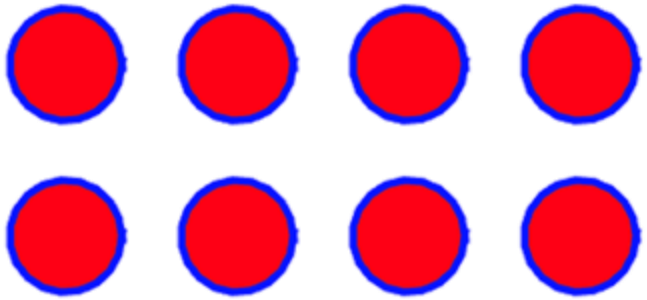
Funktionsaufrufe

Debugger

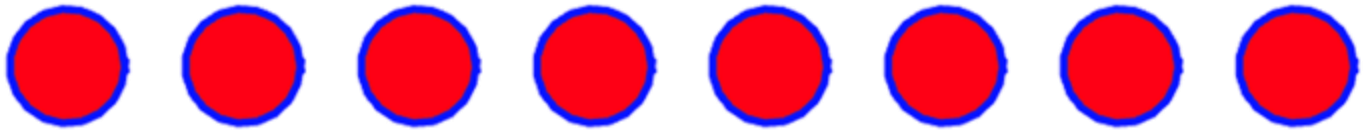
- **Debugger** ist Modus von vielen Editoren, in dem man Code …
- … **Zeile für Zeile ausführen** kann
- Vorteil: Genau nachvollziehen, wo was passiert
- VSCode: F5
- TigerPython:



Auftrag II



- A) Reproduziere die folgende Form:
- B) Reproduziere die folgende Form. Verwende eine for-Schleife.



```
from gturtle import *

turi = Turtle()

turi.setPenColor("blue")
turi.setFillColor("red")
turi.setPenWidth(10)

def circle(x,y):
    turi.setPos(x,y)
    turi.startPath()
    turi.leftArc(30,360)
    turi.fillPath()

circle(-50,-50)
circle(-50,50)
circle(50,-50)
circle(50,50)

turi.hideTurtle()
```

- C) Reproduziere die Figur. Übergebe der Funktion neben der x- & y-Koordinate auch die Füllfarbe.

- ZSA: Funktionen für andere Formen (Quadrat, Dreieck, ...), Input auch für Grösse usw.