6. Variablen

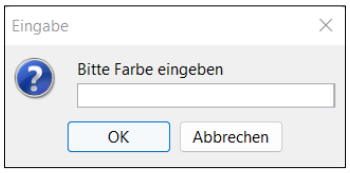
Lernziele:

- Du lernst, was Variablen sind und wofür man sie im Programm nutzen kann.
- Du lernst, wie man Werte während der Ausführung eines Programms speichern und ändern kann.

6.1 Eingabe und Speicherung von Werten

Bisher haben wir Werte wie Seitenlängen, Farbe etc. im Programm einem Befehl fest als Parameter mitgegeben. Nach dem Programmstart hatten wir keinen Einfluss mehr auf diese Werte. Wir möchten nun Werte während der Ausführung des Programms eingeben und so die Ausgabe dynamisch beeinflussen.

Mit `input("Bitte Farbe eingeben")` erzeugt man ein Eingabefenster in das Zahlen und Text eingegeben werden können. In diesem Beispiel soll eine Farbe eingegeben werden:



Damit der eingegebene Wert im Programm weiterverwendet werden kann, soll er **abgespeichert** werden. Dazu wählt man für den Speicherplatz einen Namen (z.B. `meineFarbe`) und weist ihm den eingegeben Wert mit dem Gleichheitszeichen = zu:

```
meineFarbe=input("Bitte Farbe eingeben")
```

Beispiel 6.1

Im folgenden Programm können Seitenlänge und Farbe eines Quadrats eingegeben werden.

```
1  def quadrat(seite, farbe):
2      setPenColor(farbe)
3      repeat 4:
4          fd(seite)
5          rt(90)
6  ##########################
7  from gturtle import*
8  makeTurtle()
9
10 meineSeite=input("Bitte Seitenlänge eingeben")
11 meineFarbe=input("Bitte Farbe eingeben")
12
13 quadrat(meineSeite,meineFarbe)
```

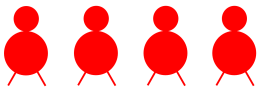
Begriffe

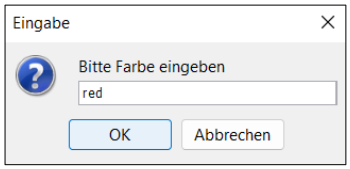
Eine **Variable** ist ein Speicherplatz, dem man einen Namen gibt. Im Speicherplatz kann genau **ein Wert** gespeichert werden. Mit dem Gleichheitszeichen (=) definiert man eine Variable und weist ihr einen Wert zu. Der Name einer Variable kann beliebig gewählt werden mit Ausnahme von Befehlswörtern (z.B. `repeat`, `forward`, `print` etc.), Sonderzeichen und Zahlen.

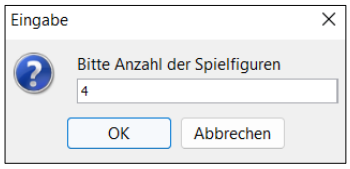
Aufgabe 6.1

Für ein Spiel sollen Spielfiguren nebeneinander gezeichnet werden. Die gewünschte Farbe und die Anzahl Spielfiguren sollen eingegeben werden können gemäss abgebildeten Eingabefenstern. Ergänze das Hauptprogramm mit den entsprechenden Anweisungen.

```
1  from gturtle import*
2
3  def spielfigur(farbe):
4      setPenColor(farbe)
5      setPenWidth(2)
6      dot(30)
7      fd(25)
8      dot(22)
9      bk(25)
10     rt(30)
11     bk(30)
12     fd(30)
13     lt(60)
14     bk(30)
15     fd(30)
16     rt(30)
17 ###################
18 makeTurtle()
19 ht()
20
21 ...
22 ...
```

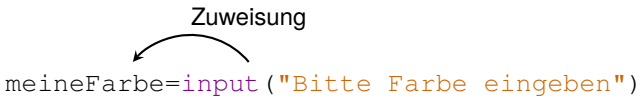






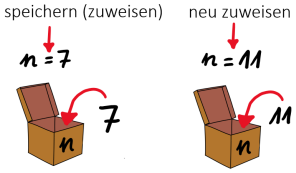
6.2 Eigenschaften von Variablen

1. Das **Gleichheitszeichen** hat die Bedeutung der Zuweisung (Speicherung) und nicht wie in der Mathematik eines Vergleichsoperators:



2. Eine Variable kannst du dir auch als Box vorstellen, die einen Namen trägt und in die man einen Wert legt. Dieser kann während des Programmablaufs beliebig ändern, in dem du der Variable einfach einen neuen Wert zuweist(überschreibst).

```
1 n=7
2 n=11
```



3. Weist man zwei Variablennamen einander zu (Zeile 3), so wird der Wert der rechten Variablen der linken zugeordnet. Im Beispiel rechts ist am Schluss unter `name1` wie auch `name2` den Wert `"Peter"` gespeichert.

```
1 name1="Jana"
2 name2="Peter"
3 name1=name2
```

4. Einer Variablen **muss vor ihrer ersten Verwendung** im Programm ein Wert zugewiesen werden!

```
1  from gturtle import*
2  makeTurtle()
3
4  # Einer Variable muss vor erstem Gebrauch ein Wert zugewiesen werden.
5  farbe="red"
6  setPenColor(farbe)
7
8  # Fehler: seite nicht bekannt
9  forward(seite)
10 seite=100
```

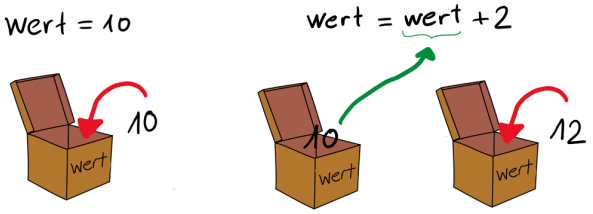
6.3 Variablenwerte bei der Ausführung ändern

Mit Hilfe von Variablen können wir im Programmablauf Werte automatisch ändern. Mit dem Ausdruck

```
wert+=2
```

wird zur Variable `wert` die Zahl `2` **addiert** und das Resultat wieder in `wert` gespeichert.

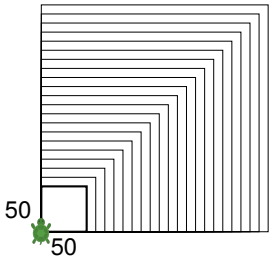
Es ist eine abgekürzte Schreibweise für `wert=wert+2`. Man kann sich das wieder mit einer Box vorstellen. Zuerst legen wir den Wert `10` in eine Box und geben ihr den Namen `wert`. Anschliessend wird `10` wird aus der Box genommen, dazu `2` addiert und das Resultat `12` wieder in die Box abgelegt.



Beispiel 6.2

Das folgende Programm zeichnet ein Muster aus Quadraten, deren Seitenlänge jeweils um 10 Pixel zunehmen, beginnend bei einer Seitenlänge von 50.

```
1  from gturtle import*
2
3  def quadrat(seite):
4      repeat 4:
5          fd(seite)
6          rt(90)
7
8  ##################
9  makeTurtle()
10 ht()
11
12 seite=50
13 repeat 20:
14     quadrat(seite)
15     seite+=10
```

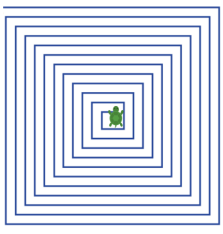


Folgende Grundoperationen sind mit dieser Notation ebenfalls möglich:

`x += Zahlenwert`	Addiert den Zahlenwert zum aktuellen Wert der Variable `x`
`x -= Zahlenwert`	Subtrahiert den Zahlenwert vom aktuellen Wert der Variable `x`
`x *= Zahlenwert`	Multipliziert den aktuellen Wert der Variable `x` mit dem Zahlenwert.
`x /= Zahlenwert`	Dividiert den aktuellen Wert der Variable `x` durch den Zahlenwert.

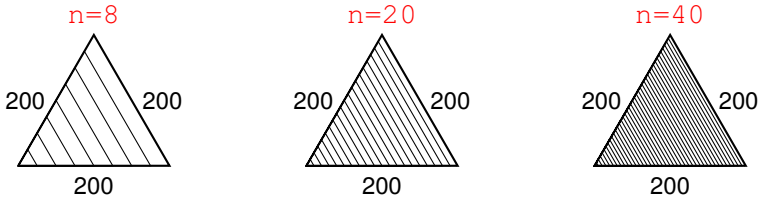
Aufgabe 6.2

Schreibe ein Programm, welches eine Spirale aus 42 Strecken zeichnet, bei der die Seitenlänge um jeweils 3 Pixel zunimmt. Die kleinste Startseite beträgt 10 Pixel.



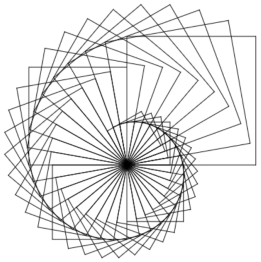
Aufgabe 6.3

Ein Programm zeichnet eine Figur, die aus `n` gleichseitigen Dreiecken besteht, deren Seitenlängen linear abnehmen. Das grösste Dreieck hat immer die Seitenlänge 200 und `n` soll eingegeben werden können.



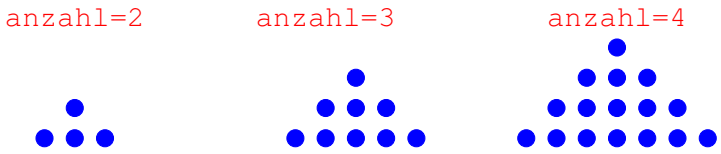
Aufgabe 6.4

Schreibe ein Programm für eine Figur aus vierzig Quadraten deren Seitenlängen um 3% abnehmen und um 10° verdreht gezeichnet werden. Die Seitenlänge des grössten Quadrats beträgt 200.



Aufgabe 6.5

Erstelle einen Befehl `punkte(n)`, der eine vertikale Anordnung von `n` Punkten zeichnet. Beim Programmstart soll `anzahl` (die Anzahl der Punkte der höchsten Punktspalte) eingegeben werden können. Daraus soll dann die folgende Grafik gezeichnet werden.



Aufgabe 6.6

Schreibe ein Programm, welches die abgebildete Europaflagge zeichnet. Mit dem Befehl `heading(0)` kann die Turtle senkrecht (in Richtung Norden) ausrichtet werden.

Projektaufgabe - Analoguhr

Bearbeite die **Projektaufgabe Analoguhr** (siehe Webseite → Weitere Kapitel/Projektaufgaben), in der eine animierte Uhr entwickelt werden soll, welche die aktuelle Uhrzeit anzeigt.

6.4 Ausgabe und Zahlenfolgen

Mit Hilfe von `print(wert)` lassen sich Werte ausgeben. Neben Zahlen und Strings (mit Hochkommas) kann in den Klammern eine Variable übergeben werden, die stets durch ihren Wert ersetzt und ausgegeben wird.

```
1 print(3)          #Ausgabe: 3
2 print("Wort")     #Ausgabe: Wort
3 zahl=4
4 print(zahl)       #Ausgabe: 4
```

In den Klammern kann man auch Berechnungen einfügen, die ausgeführt und ausgegeben werden.

```
1 a=4
2 b=10
3 print(a+b+2)   #Ausgabe: 16
```

Möchte man Werte verschiedener Datentypen ausgeben (z.B. einen Text mit einer Variablen), so müssen diese mit einem Komma getrennt werden.

```
1 x=12
2 print("Die Zahl",x,"ist gerade.")   # Ausgabe: Die Zahl 12 ist gerade.
```

Aufgabe 6.7

Was ist die Ausgabe der folgenden Programme?

```
1 def berechne(x,y):
2     a=2*x
3     b=a+3*y
4     b+=a
5     print(b)
6
7 berechne(2,3)
```

(a)

```
1 x=1
2 zahl=0
3 repeat 4:
4     zahl*=10
5     zahl+=x
6     print(zahl)
```

(b)

Beispiel 6.3

Der Befehl `gerade(n)` gibt die ersten `n` geraden Zahlen aus:

```
1 def gerade(n):
2     x=1
3     repeat n:
4         print(2*x)
5         x+=1
6
7 n=input("Bitte ein n eingeben")
8 gerade(n)
```

Aufgabe 6.8

Ergänze das Programm aus Beispiel 6.3 mit einem Befehl `ungerade(n)`, der die ersten `n` ungeraden Zahlen ausgibt. (z.B. `ungerade(n)` gibt 1, 3, 5, 7 aus)

Aufgabe 6.9

Schreibe einen Befehl `quadratzahl(n)`, der alle Quadratzahlen der ersten `n` natürlichen Zahlen ausgibt. `n` soll eingegeben werden können. (z.B. `quadratzahl(5)` gibt 1, 4, 9, 16, 25 aus)

Aufgabe 6.10

Schreibe einen Befehl `sum(n)`, der die Summe der ersten `n` natürlichen Zahlen berechnet und ausgibt. (`sum(4)` gibt 10 aus, weil 1+2+3+4=10)