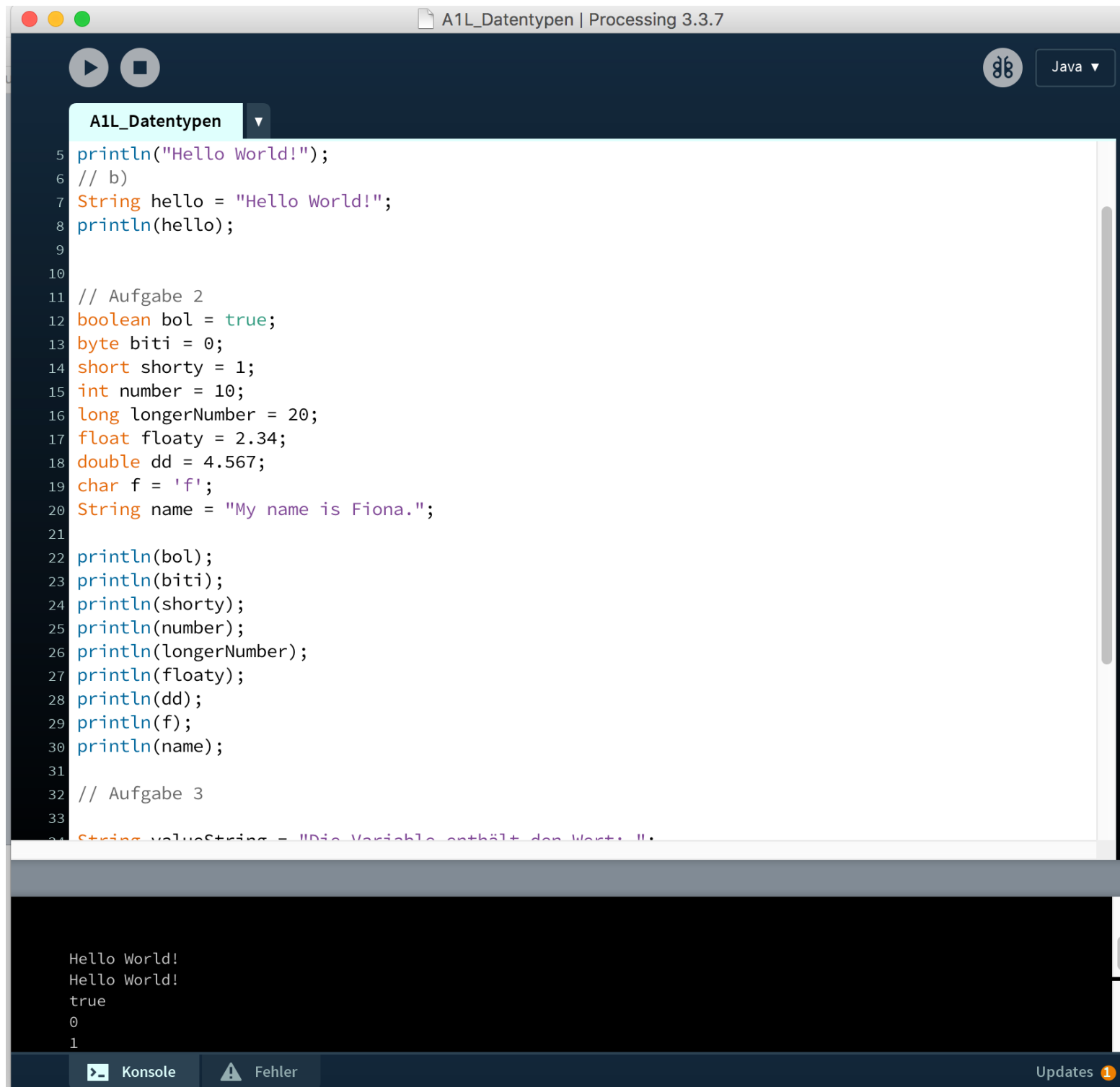


Processing IDE



```
A1L_Datentypen | Processing 3.3.7

println("Hello World!");
// b)
String hello = "Hello World!";
println(hello);

// Aufgabe 2
boolean bol = true;
byte biti = 0;
short shorty = 1;
int number = 10;
long longerNumber = 20;
float floaty = 2.34;
double dd = 4.567;
char f = 'f';
String name = "My name is Fiona.";

println(bol);
println(biti);
println(shorty);
println(number);
println(longerNumber);
println(floaty);
println(dd);
println(f);
println(name);

// Aufgabe 3
String valueString = "Die Variable enthält den Wert: ";
```

Hello World!
Hello World!
true
0
1

Konsole

Pro Anweisung eine Zeile

Eine Anweisung wird mit einem Semicolon abgeschlossen

Konsole

- **Variable** = Gefäß um Daten zu speichern
- kann in Java nur Daten von einem definiertem Wert aufnehmen

```
int variableName;           // Deklaration. Es wird deklariert welchen
                             // Typ und Namen unser Speichergefäß hat.

//...

variableName = 2;           // später im Code: es kann immer wieder
println(variableName);      // auf die Variable zugegriffen werden.
```

// primitive Datentypen		// Referenztyp
byte	<u>b</u> ;	String <u>str</u> ;
short	<u>s</u> ;	
int	<u>i</u> ;	
long	<u>l</u> ;	
	float <u>f</u> ;	
	double <u>d</u> ;	
	char <u>c</u> ;	
	boolean <u>bol</u> ;	

Beispiel

```
5 // variablen Deklaration
6 // Typ und Name
7 int sumandOne;
8 int sumandTwo;
9
10 // Wert Zuweisung
11 sumandOne = 3;
12 sumandTwo = 4;
13
14 // Deklaration und Initialisierung
15 // Zugriff auf Variablen
16 int sum = sumandOne + sumandTwo;
17
18 // Zugriff auf Variablen
19 println( sum );
```

```
24 // Variablen Deklaration
25 boolean first;
26 boolean second;
27
28 // Wert Zuweisung
29 first = true;
30 second = false;
31
32 // Variablen Deklaration
33 boolean result;
34
35 // Zuweisung und Operation
36 result = first & second;
37
38 println( result );
```

Mit **Operatoren** können Zuweisungen und Berechnungen vorgenommen und Bedingungen formuliert und geprüft werden.

```
int c = a + b;
```

```
int c = a - b;
```

```
int c = a * b;
```

```
int c = a / b;
```

```
int c = a % b;
```

```
a++;
```

```
b--;
```

```
a < b
```

```
a <= b
```

```
a > b
```

```
a >= b
```

```
a == b
```

```
a != b
```

```
!
```

```
&
```

```
^
```

```
|
```

```
&&
```

```
||
```

a	b	a & b a && b	a b a b	a ^ b
true	true	true	true	false
true	false	false	true	true
false	true	false	true	true
false	false	false	false	false

Array = Sammlung von Elementen des gleichen Datentyps

(Kann sich als Sammlung von Variablen desselben Datentyps vorgestellt werden)

```
int[] arrayName;           // Array Deklaration.  
  
arrayName = new int[10];    // Entweder: Erzeugung mit Angabe wie viele Elemente  
                             // die Sammlung enthalten soll  
arrayName = {1, 2, 6, 4, 8}; // Oder: Erzeugung direkt mit Angabe der enthaltenen Elementen
```

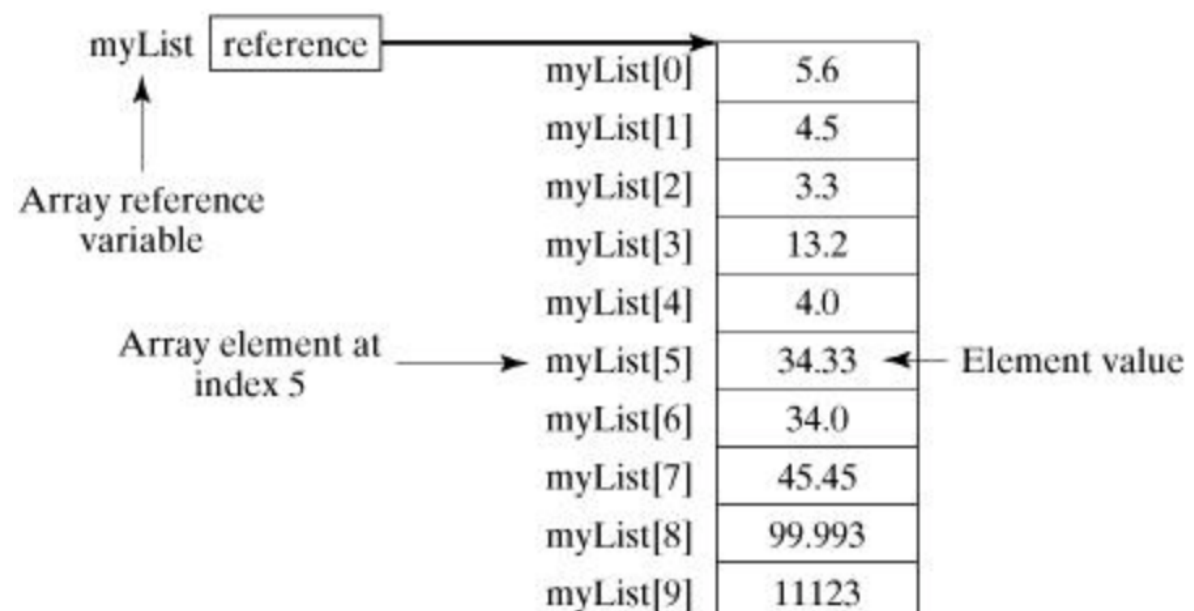
```
int index = 1;  
  
// Array Elementzugriff:  
  
int element = arrayName[index];  
arrayName[index] = 2;
```

Das erste Element erhält man mit index = 0.

```
// Die Anzahl der Elemente im Array  
int arraySize = arrayName.length;
```

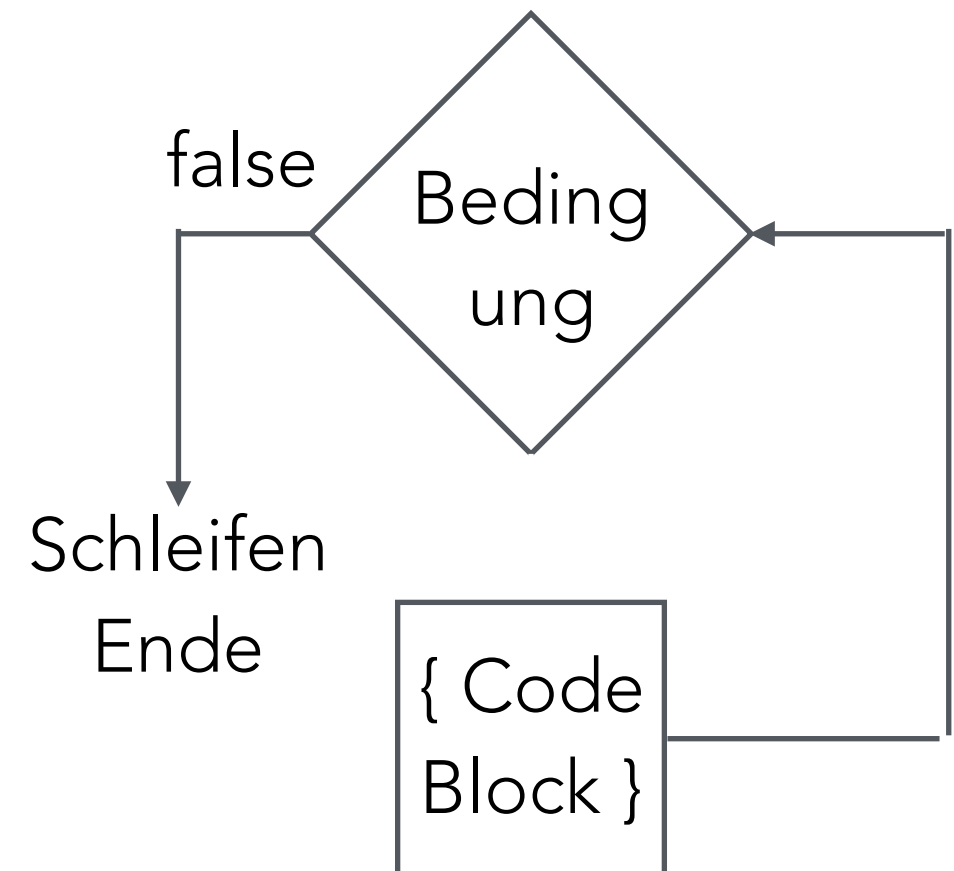
```
double[] myList = new double[10];
```

Following picture represents array myList. Here, myList holds ten double values and the indices are from 0 to 9.



Die **while**-Schleife

```
boolean abbruchsbedingung;  
  
while ( abbruchsbedingung ){  
    // Anweisungs Block  
}  
  
do {  
    // Anweisungs Block  
} while ( abbruchsbedingung );
```



```
int max = 10;  
int sum = 10;  
  
while ( sum < max ){  
    sum = sum + 2;  
}  
println(sum); 10
```

```
sum = 10;  
  
do{  
    sum = sum + 2;  
}while ( sum < max );  
println(sum); 12
```

Die **for**-Schleife

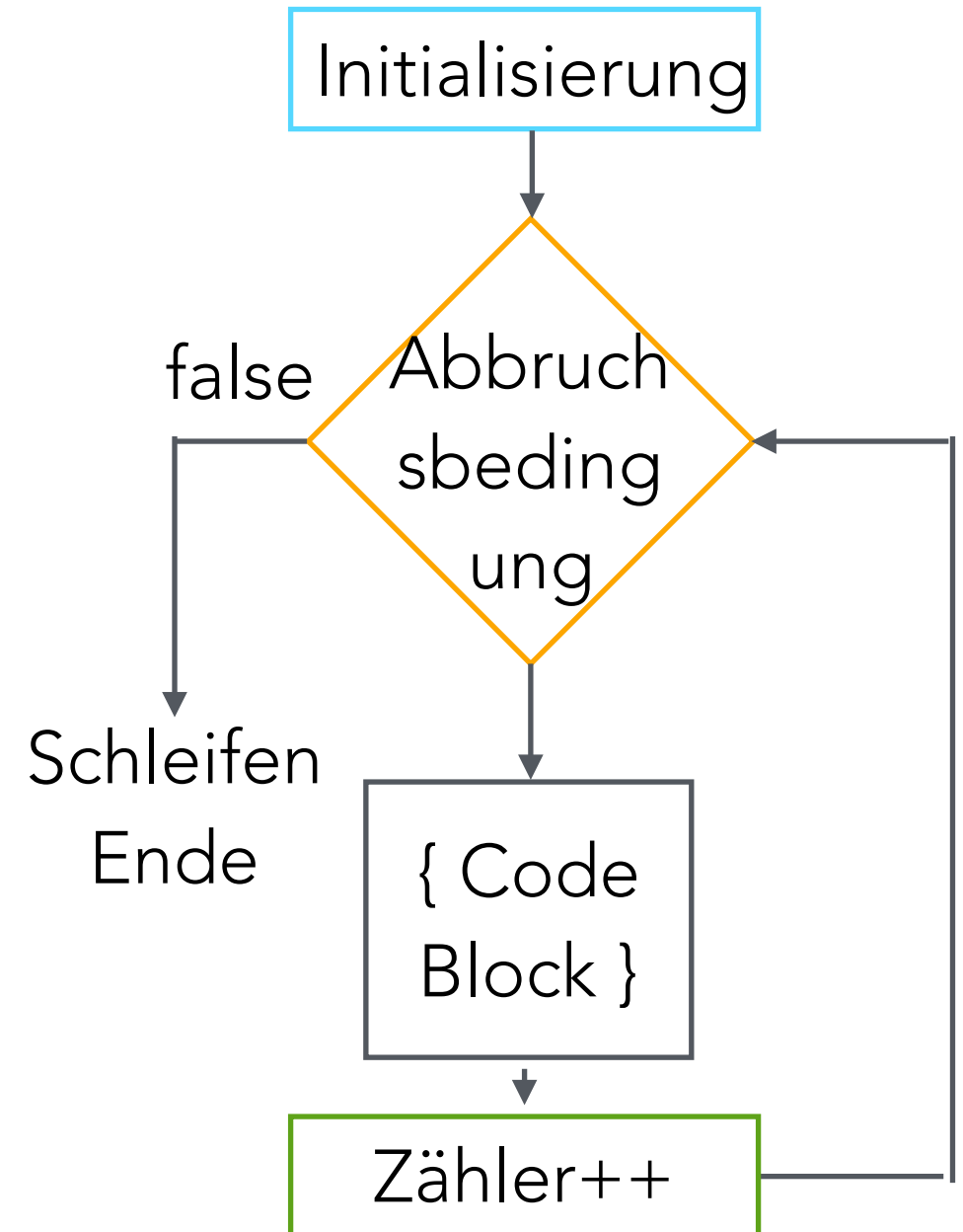
```
int max = 10;
```

```
for( int i = 0; i < max; i++){  
    // Anweisungs Block  
}
```

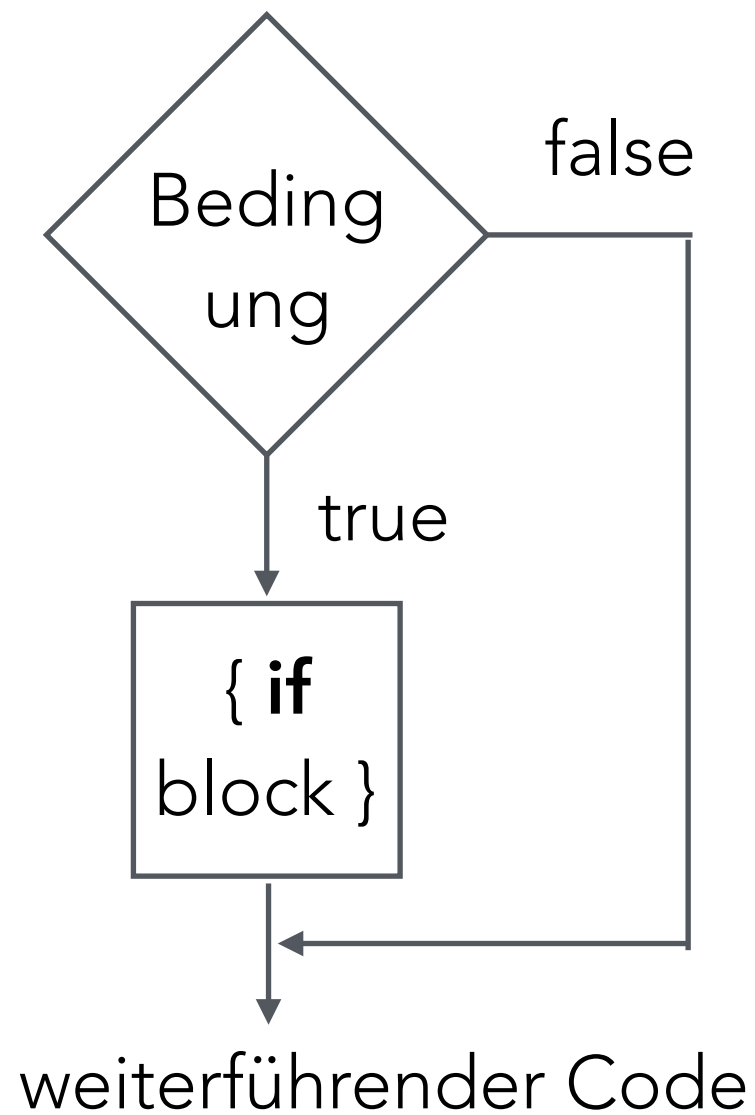
Update der
Zähler Variable

Abbruchsbedingung

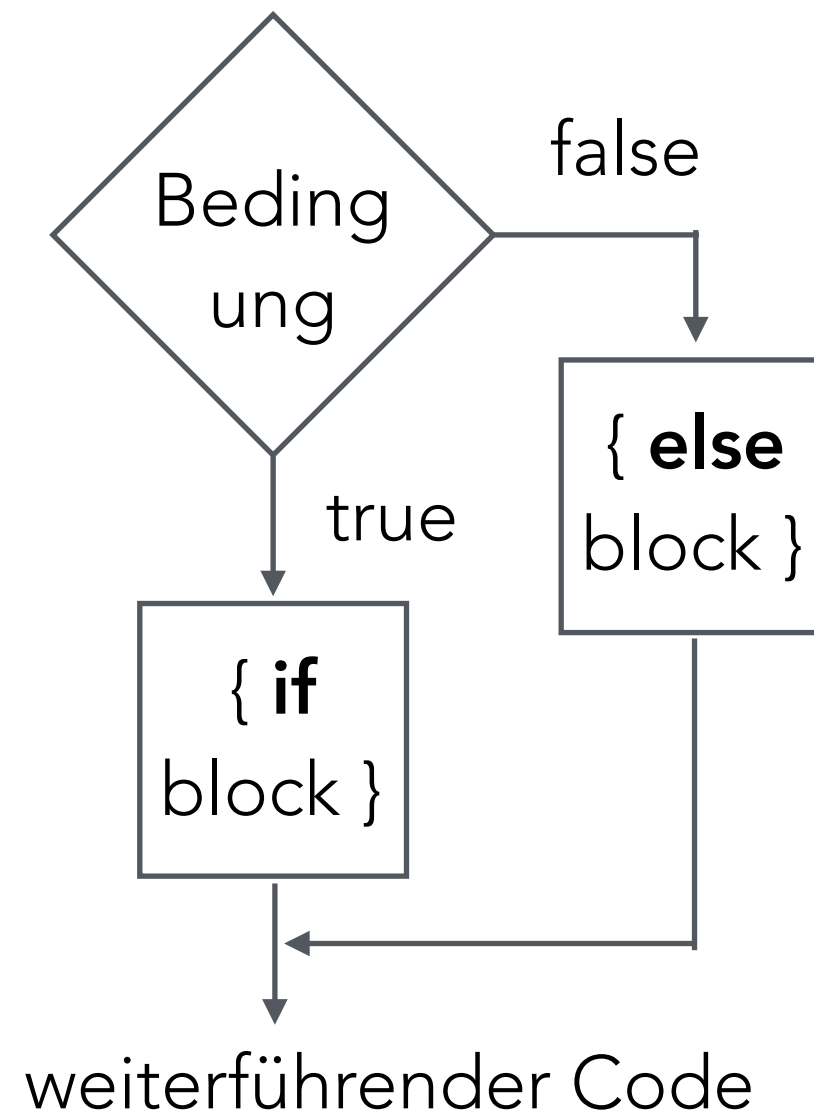
Initialisierung der Zähler
Variable vom Typ int



```
if( a < b ){  
    // if code block  
}
```



```
if( a < b ){  
    // if code block  
}else{  
    // else code block  
}
```

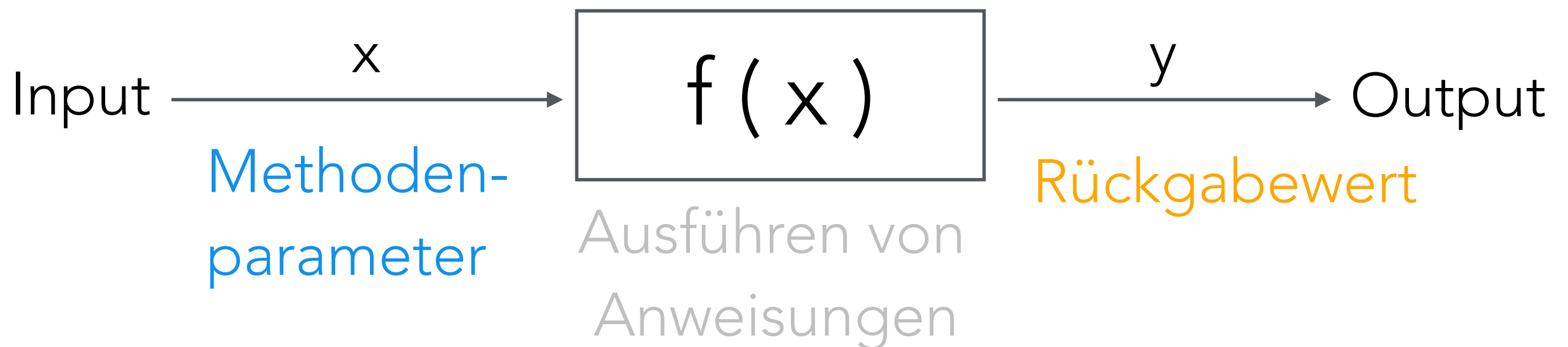


Eine **Methode** kann sich (in etwa) wie eine mathematische Funktion vorgestellt werden

$$y = f(x)$$

Funktionsargument

Resultat Ausführen von Berechnung



Rückgabe Typ
(Datentyp des Outputs)

Methodenname

Methodenparameter
(Input mit Typ und Name)

return Statement

- Definiert welche Variable zurückgegeben wird.
- Muss dem Typ des Rückgabewerts entsprechen.
- Beendet die Methode.

```
int line(int x){  
    int y = 2 * x + 1;  
    return y;  
}
```

Methoden
Code
Block

```
int result = line(2);
```

Methoden Aufruf

