7. Verzweigungen und bedingte Schleifen

Lernziele:

- Du lernst, wie der Computer anhand aktueller Variablenwerte Entscheidungen treffen kann.
- Du lernst einen neuen Schleifentyp kennen, bei dem eine Tätigkeit solange wiederholt wird, bis eine Bedingung erfüllt ist.

7.1 Bedingte Anweisung

Das Vorgehen der Einstiegsübung mit den Zahlenkarten könnte man so formulieren:

> **WENN** sich die gedachte Zahl auf der Karte befindet:
> Dann addiere die erste Zahl dieser Karte zu den anderen ersten Zahlen, deren Karten diese Bedingung erfüllt haben.

oder allgemeiner:

> **WENN** eine *Bedingung* erfüllt ist:
> Führe *Anweisung(en)* aus.

Eine **bedingte Anweisung** besteht aus einer Bedingung und einem Block, der aus einer oder mehreren Anweisungen besteht. In Python hat die `if`-Anweisung die Form:

```
1 if Bedingung:
2     Anweisung1
3     Anweisung2
```

Die if-Anweisung überprüft, ob eine Bedingung wahr (`True`) oder nicht wahr (`False`) ist. Falls sie wahr ist, wird der eingerückte Block (ab Zeile 2) ausgeführt. Falls nicht, wird der eingerückte Block ignoriert und das Programm auf Zeile 4 weitergeführt (hier nicht sichtbar).

Beachte, dass nach der Bedingung ein Doppelpunkt folgt.

Beispiel 7.1

1. Ein Programm überprüft den eingegebenen Wert `x` auf die Bedingung x>0 und gibt eine entsprechende Meldung aus.

```
1 x=input("Bitte eine Zahl eingeben")
2
3 if x>0:
4     print("Die Zahl",x,"ist positiv")
```

2. Ein Programm überprüft, ob eine Zahl gleich 42 ist:

```
1 x=input("Bitte eine Zahl eingeben")
2
3 if x==42:
4     print("Die Zahl",x,"ist einzigartig.")
```

Aufgabe 7.1

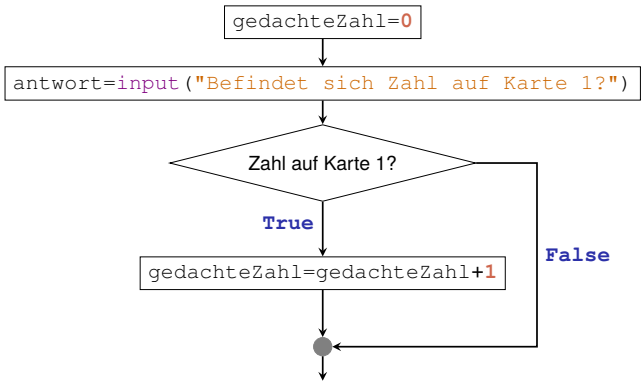
Für den Vergleich `if x==42:` verwenden wir das doppelte Gleichheitszeichen. Warum verwenden wir nicht das einfache Gleichheitszeichen, also `x=42`?

Beispiel 7.2

Abgebildet ist die Überprüfung der ersten Karte des Einführungsbeispiels. Für die zu bestimmende Zahl definieren wir eine Variable `gedachteZahl` und setzen sie gleich 0.

```
1 gedachteZahl=0
2
3 antwort=input("Befindet sich die gedachte Zahl auf der Karte 1? (j/n)")
4 if antwort=="j":
5     gedachteZahl=gedachteZahl+1
```

Mit Hilfe eines **Flussdiagramms** lässt sich der Programmabschnitt des Beispiels 7.2 visualisieren:



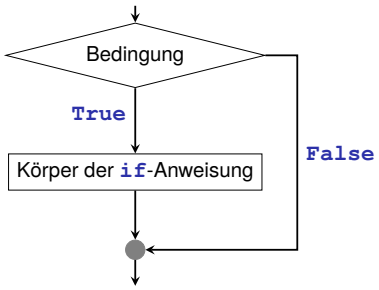
Die zu prüfende Bedingung steht stets in einem Rhombus, von dem zwei Pfeile ausgehen. Einer für den Fall, dass die Bedingung erfüllt ist (`True`) und einer für den Fall, dass die Bedingung nicht erfüllt ist (`False`). Die restlichen Anweisungen stehen in rechteckigen Kästchen.

Aufgabe 7.2

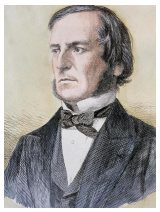
Vervollständige das Programm aus Beispiel 7.2 für alle sieben Karten. Am Schluss soll die gedachte Zahl ausgegeben werden.

Begriffe

Mit der `if`-Anweisung realisieren wir eine **bedingte Anweisung**, mit welcher der Programmablauf durch die Überprüfung einer **Bedingung** gesteuert werden kann. Ist die Bedingung wahr (`True`), so wird der eingerückte Programmblock (**Körper der `if`-Anweisung**) ausgeführt. Ist sie nicht wahr (`False`), wird der eingerückte Block ignoriert und das Programm auf der nicht eingerückten Programmzeile weiter ausgeführt.



Die beiden Wahrheitswerte `True` (wahr) und `False` (nicht wahr) bilden den Datentyp **Boolean**, benannt nach dem Mathematiker/Logiker George Boole (1815-1864). Die beiden Wahrheitswerte werden auch als **Boolesche Variablen** bezeichnet.



Aufgabe 7.3

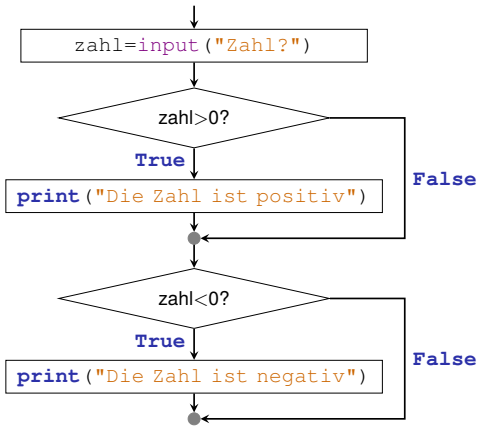
Entwickle ein kleines Quiz zu einem beliebigen Gebiet mit fünf Fragen. Für jede korrekte Antwort gibt es einen Punkt. Am Schluss soll die erreichte Punktzahl ausgegeben werden.

Aufgabe 7.4

a) Schreibe das Programm zum Flussdiagramm (rechts).

b) Ist dieses Programm logisch sinnvoll? Gibt es Verbesserungspotenzial?

c) Welcher „Spezialfall“ fehlt noch?



7.2 Die if-else Verzweigung

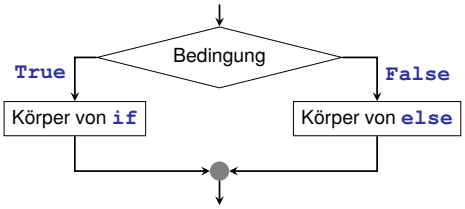
Im Programm der Aufgabe 7.3 prüfen wir, ob zahl>0 und ob zahl<0. Wenn eine Zahl positiv ist, muss nicht mehr geprüft werden, ob sie negativ ist. Man kann sich also die zweite **if**-Anweisung sparen und stattdessen die **if-else**-Anweisung verwenden!

```
1 zahl=input("Zahl?")
2
3 if zahl>0:
4     print("Die Zahl ist positiv")
5 else:
6     print("Die Zahl ist negativ")
```

Beachte, dass jeweils nach **if** und **else** Doppelpunkte eingeführt und die auszuführenden Blöcke eingerückt werden müssen.

Begriffe

Mit der strukturierten Anweisung **if-else** können wir ein Programm in zwei Richtungen verzweigen. Der **else** -Zweig wird nur dann ausgeführt, wenn die Bedingung nicht wahr (**False**) ist.



Aufgabe 7.5

Schreibe ein Programm zur Eingabe deines Vornamens. Falls dein Vorname (z.B.`"Anna"`) eingegeben wird, soll eine Begrüssung `Hallo Anna` ausgegeben werden, andernfalls `Hallo Unbekannte(r)`.

Aufgabe 7.6

Ergänze im Programm zu Aufgabe 7.4 den fehlenden Fall der Zahl 0. In diesem Fall soll die Meldung: „Die Zahl ist 0“ ausgegeben werden. Verwende die **if-else**-Anweisung.

Beachte: Ist die Zahl gleich 0, müssen die beiden anderen Fälle nicht mehr geprüft werden!

Aufgabe 7.7

Was ist die Ausgabe, wenn a=10 eingegeben wird?

```
a=input("Bitte Zahl eingeben")
b=4
if a>0:
    if a<3:
        b=b*3
        a=a+b
    else:
        b=b/2
        a=b-a
else:
    b=b+2
    a=b*a

print(a,b)
```

7.3 Tastatursteuerung

In diesem Abschnitt lernst du, wie man Befehle mit Hilfe der Tastatur ausführen lassen kann. Dazu muss der Computer wissen, welche Taste gedrückt wurde und was dann passieren soll.

Jede Taste besitzt einen Zahlencode (Dezimalzahl), den du mit dem Befehl `getKeyCode()` beim Betätigen der Taste ermitteln kannst:

Beispiel 7.3

Das Beispielprogramm gibt im Sekundentakt eine Zahl aus.

Beim Drücken von F wird die Zahl 70 ausgegeben.
Wird keine Taste gedrückt, so wird 0 ausgegeben.

```
from gturtle import*
makeTurtle()

repeat:
    key=getKeyCode()
    delay(1000)
    print(key)
```

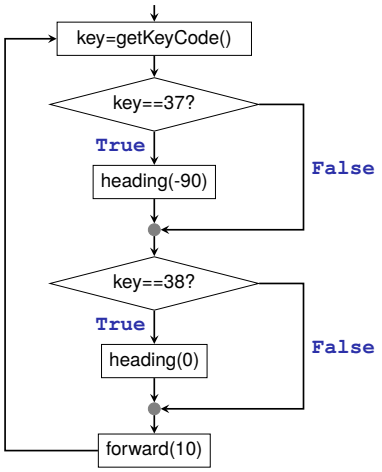
Wir bauen eine kleine Turtlesteuerung und benutzen den Befehl `heading(winkel)`, mit dem sich die Turtle wie folgt ausrichten lässt:

`heading(0)` | `heading(-90)` | `heading(90)` | `heading(180)`

Beispiel 7.4

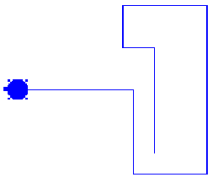
Im Beispiel benutzen wir die Pfeiltasten ↑ und ← um die Turtle, die sich kontinuierlich vorwärts bewegt, nach links und nach oben auszurichten.

```
from gturtle import*
makeTurtle()

repeat:
    key=getKeyCode()
    if key==37:
        heading(-90)
    if key==38:
        heading(0)
    forward(10)
```



Aufgabe 7.8

a) Erweitere das Programm aus Beispiel 7.4 so, dass die Turtle auch nach rechts und nach unten ausgerichtet werden kann. Falls du eine Tastatur ohne Pfeiltasten hast, kannst du andere verwenden.



b) Ergänze die Möglichkeit, dass sich durch das Betätigen der Taste B die Turtle stoppt bzw. mit der Taste F die Turtle vorwärts bewegt. Die Turtle soll sich zu Beginn vorwärts bewegen.

7.4 Die if-elif Verzweigung

In Beispiel 7.4 wird jede der Bedingungen geprüft, auch wenn die vorangegangene bereits erfüllt war.

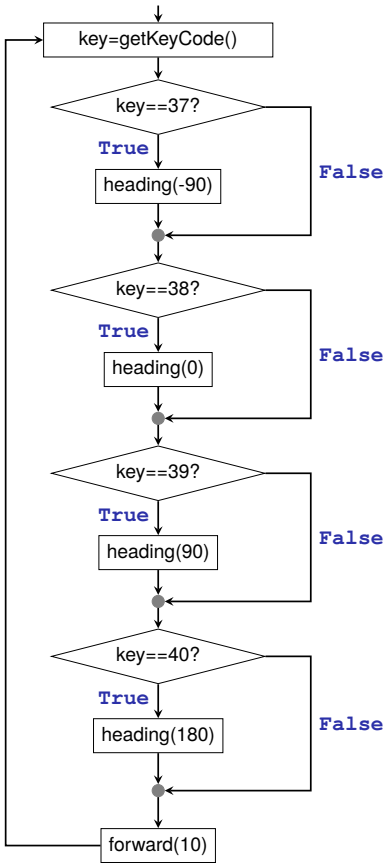
Mit `if-elif`-Anweisung kann man dies besser umsetzen.

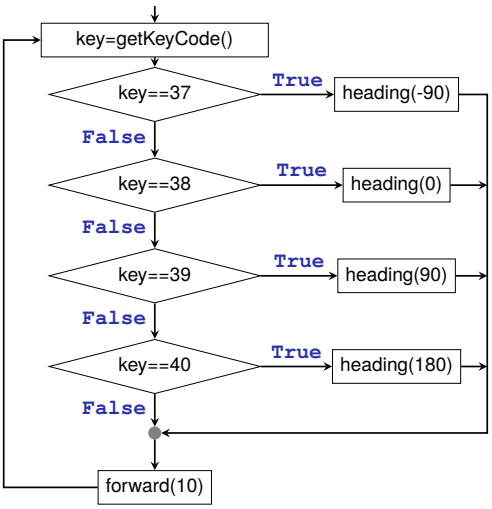
Das Programm mit dem zugehörigen Flussdiagramm (unten rechts) zeigen die Anwendung von `if-elif`. Sobald eine der Bedinungen wahr ist, prüft das Programm die restlichen Bedingungen nicht mehr.

```
1  from gturtle import*
2  makeTurtle("blue")
3
4  repeat:
5      key=getKeyCode()
6      if key==37:
7          heading(-90)
8      elif key==38:
9          heading(0)
10     elif key==39:
11         heading(90)
12     elif key==40:
13         heading(180)
14     forward(10)
```

.

Der Unterschied zwischen der `if`- und `if-elif`-Anweisung:





Begriffe

Die `if-elif`-Anweisung ermöglicht es, ein Programm in beliebig viele Wege zu verzweigen. Der Vorteil liegt bei der Überprüfung der Bedingungen: Sobald eine Bedingung wahr ist, werden die restlichen Bedingungen nicht mehr geprüft.

Projektaufgabe

Bearbeite die **Projektaufgabe- Aliens einfangen** zur Entwicklung eines kleinen Spiels mit Tastatursteuerung (siehe Webseite → Weitere Kapitel/Projektaufgaben).

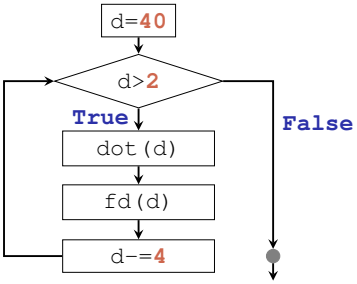
7.5 Eine Schleife abbrechen

Mit der Anweisung `break` kann man die Ausführung einer Schleife abbrechen und verlassen.

Beispiel 7.5

Das Programm zeichnet eine Reihe von kleiner werdenden Kreisen. Die Schleife, und damit auch das Programm, wird abgebrochen, sobald ein Durchmesser eines Kreises kleiner oder gleich 2 ist.

```
1  from gturtle import*
2  makeTurtle()
3  ht()
4
5  d=40
6  repeat:
7      if d>2:
8          dot(d)
9          fd(d)
10         d-=4
11     else:
12         break
```



Aufgabe 7.9

Was passiert bei folgenden Programmen. Was wird ausgegeben?

```
1  i=1
2  repeat:
3      if i<50:
4          print(2*i)
5          i+=1
6      else:
7          break
```

```
1  j=1
2  repeat:
3      if j<50:
4          print(j)
5          j-=1
6      else:
7          break
```

Aufgabe 7.10

Ein Programm soll folgende Brüche miteinander addieren:

$$\frac{1}{2}+\frac{1}{4}+\frac{1}{8}+\frac{1}{16}+\frac{1}{32}\ldots$$

Beginnend mit $\frac{1}{2}$ werden die Brüche jeweils halbiert. Addiere zuerst 10 Brüche, dann 100 und schliesslich 10000. Gibt die Summe mit `print` aus. Welches sind die Resultate?

Projektaufgabe

Bearbeite die **Projektaufgabe - NIM Spiel** (siehe Webseite → Weitere Kapitel/Projektaufgaben) zum kleinen Strategiespiel.

7.6 Die `while`-Schleife

Mit Hilfe der `while`-Anweisung lässt sich das Konstrukt mit `repeat`, `if` und `break` einfacher umsetzen:

Beispiel 7.6

Das Programm aus Beispiel 7.5 (links) lässt sich mit `while` kürzer umsetzen (rechts):

```
from gturtle import*
makeTurtle()
ht()

d=40
repeat:
    if d>2:
        dot(d)
        fd(d)
        d-=4
    else:
        break
```

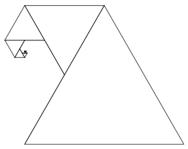
```
from gturtle import*
makeTurtle()
ht()

d=40
while d>2:
        dot(d)
        fd(d)
        d-=4
```

Begriffe

Die `while`-Schleife wiederholt eine Tätigkeit, solange die Bedingung hinter `while` wahr ist. Wenn die Bedingung hinter `while` immer wahr ist, dann läuft die Ausführung der Schliefe unendlich weiter. In diesem Zusammenhang spricht man auch von einer **Endlosschleife**.

Aufgabe 7.11

Erstelle die Grafik aus gleichseitigen Dreiecken, deren Seitenlängen jeweils solange halbiert werden, bis eine Seitenlänge kleiner oder gleich 2 ist. Die Startseite ist 300 Pixel.



Aufgabe 7.12

Was macht das Programm mit der Zahl, die man zu Beginn eingeben muss? Was ist die Ausgabe?
Erkläre allgemein, ohne das Programm abzutippen.

```
def rechne(x):
    summe=0
    while x>0:
        summe+=3
        x-=3
    print(summe)

zahl=input("Gib eine Zahl ein")
if zahl>0:
    rechne(zahl)
```

Beispiel 7.7: Neue Operatoren

Mit dem //-Operator kann man eine Zahl **ganzzahlig** teilen:
Beispiele: a) 7//3=2 b) 13//12=1 c) 3//4=0.

Mit dem %-Operator lässt sich der **Rest** einer Division bestimmen:
Beispiele: a) 7%3=**1** weil 7//3=3 Rest **1** b) 3%4=**3** weil 3//4=0 Rest **3**.

Aufgabe 7.13

Schreibe ein Programm und verwende eine `while`-Schleife und die Operatoren // und %:

a) Nach Eingabe einer Zahl sollen alle Teiler dieser Zahl bestimmt und ausgeben werden.

b) Erweitere das Programm a): Falls die eingegebene Zahl eine Primzahl ist, soll `"Die Zahl ist eine Primzahl"` ausgegeben werden.