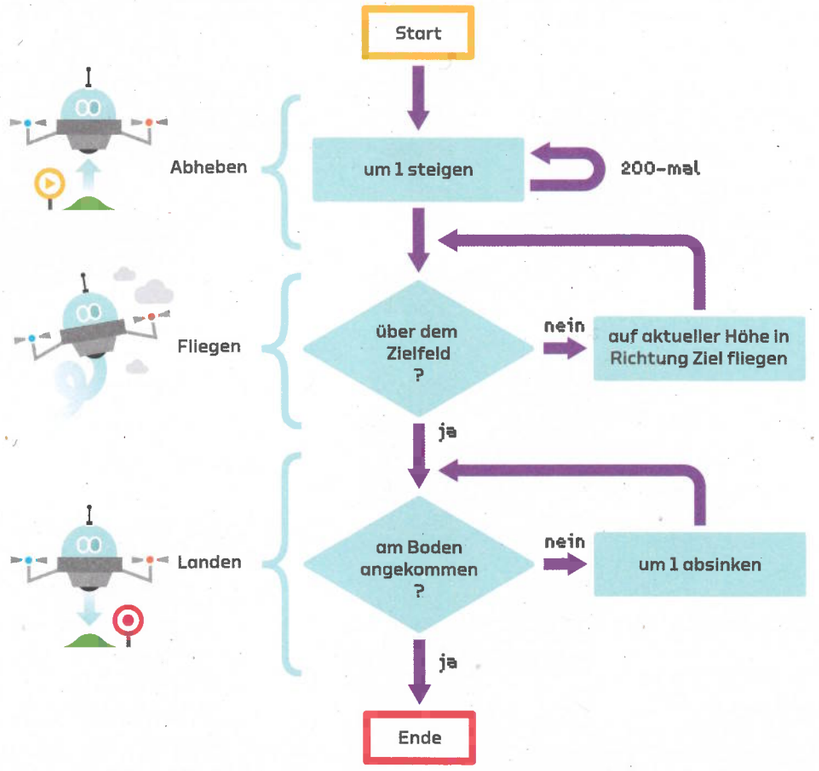
Zum Ziel und zurück

Die Drohne in unserem Modell soll nun automatisch vom Startfeld zum Zielfeld fliegen. Unser Algorithmus im Ablaufdiagramm beschreibt genau, wie sie vorgehen soll.

311 *Übersetze den Algorithmus aus dem Ablaufdiagramm in Programmbefehle für die Drohne. Schreibe das Programm für alle drei Phasen: Abheben, Fliegen, Landen.*



Tipp

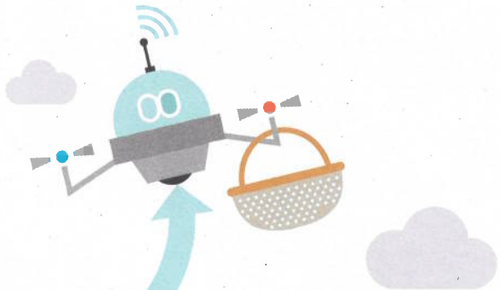
Um zu prüfen, ob sich die Drohne bereits über dem Zielfeld befindet, kann ein Vergleich verwendet werden:

312 *Verwende verschiedene Landschaften, um dein Drohnenprogramm in unterschiedlichen Situationen zu testen.*

313 *Erweitere dein Drohnenprogramm. Die Drohne soll nun am Zielfeld kurz warten und wieder zum Startfeld zurückfliegen.*

Tipp

Du kannst das Programm vom Hinflug kopieren und für den Rückflug anpassen.



Aus: einfach INFORMATIK - Programmieren, Klett und Balmer Verlag, 2018, S. 98/99

Zusammenfassung

Beim Programmieren braucht man die Möglichkeit, abhängig von der Situation verschiedene Tätigkeiten auszuüben oder eine Suche abzubrechen. Die Situation wird durch die Werte der Variablen beschrieben. Der Computer muss also abhängig von seinen Variablen-Werten **Entscheidungen** treffen können. Das ermöglichen die neuen Anweisungen `if`, `else`, `break` und `while`.
Mit `if` lassen wir den Computer eine **Bedingung** prüfen und entscheiden, ob er den Körper der `if`-Anweisung ausführen soll oder nicht. Mit `if` und `else` zusammen entscheidet sich der Computer für eine von zwei möglichen Tätigkeiten. Mit `break` kannst du die Schleife sofort abbrechen. Der Computer springt damit aus der Schleife und führt die Anweisungen aus, die nach der Schleife stehen. Mit `while` hast du eine Schleife, für die du nicht im Voraus bestimmen musst, wie viele Male sie durchlaufen werden soll. Sie wird so lange wiederholt, wie die angegebene Bedingung erfüllt ist. Wenn die Bedingung der `while`-Schleife immer erfüllt ist, dann arbeitet die Schleife unendlich lange. Das ist das erste Mal, dass du Programme schreiben kannst, die im Prinzip ewig laufen können.
Die Bedingungen in `if`- und `while`-Anweisungen dürfen **komplexe Bedingungen** sein. Mit `and` (und), `or` (oder) und `not` (nicht) kannst du einfache Bedingungen zu komplexen Bedingungen verknüpfen.

In diesem Kapitel hast du folgende Befehle gelernt:

if Bedingung: Körper	Die Anweisungen im Körper zu `if` werden nur dann ausgeführt, wenn die Bedingung erfüllt ist. Wenn du als Bedingung angibst, dass eine Variable einen bestimmten Wert haben soll, dann verwendest du ein doppeltes Gleichheitszeichen ==. Zwei Bedingungen kannst du mit `and` oder `or` verknüpfen.

if Bedingung:
  Körper 1
else:
  Körper 2
	Der Computer führt entweder die Anweisungen im Körper 1 oder die Anweisungen im Körper 2 aus. Wenn die Bedingung hinter `if` erfüllt ist, dann wird der Körper 1 ausgeführt, andernfalls der Körper 2.

break	Die Schleife sofort abbrechen und den nächsten Befehl ausführen, der nach der Schleife steht.
while Bedingung: Körper	Solange die Bedingung hinter `while` erfüllt ist, wird der Körper der `while`-Schleife wiederholt. Nach jeder Wiederholung des Schleifenkörpers prüft der Computer, ob die Bedingung erfüllt ist. Wenn ja, dann wird die Schleife noch einmal wiederholt.

Teste dich selbst

Konzepte und Befehle

1 Wie kannst du erreichen, dass ein Programm eine Tätigkeit nur unter gewissen Bedingungen ausführt?
2 Wie kannst du einem Programm die Möglichkeit geben, sich für eine von zwei möglichen Alternativen zu entscheiden?
3 Kann ein Programm abhängig von seinen Variablen-Werten eine von vielen unterschiedlichen Tätigkeiten auswählen? Falls ja, zeige mit einem Beispiel, wie es geht.
4 Wie kannst du eine `repeat`-Schleife vorzeitig abbrechen?
5 Welche Struktur hat eine `while`-Schleife?
6 Was passiert, wenn die Bedingung hinter `while` erfüllt ist? Was passiert, wenn sie nicht erfüllt ist? Was hat das mit der Bedeutung des englischen Wortes «while» zu tun?
7 Was macht der Computer, wenn er die Zeile `if 2*x-7 == y*y+3:` abarbeitet? Wie stellt er fest, ob diese Bedingung für die aktuellen Werte von **x** und **y** erfüllt ist?
8 Wann ist die zusammengesetzte Bedingung `A and not B` erfüllt? Müssen A und B dazu beide erfüllt sein?
9 Ist die zusammengesetzte Bedingung `x > 0 or x < 9` auch erfüllt, wenn der Wert von **x** gleich 3 ist?
10 Was ist die Negation der Bedingung `x == 4`?

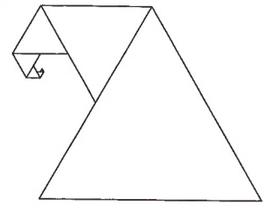
Aufgaben

1 Das folgende Programm fordert dich auf, eine Zahl einzugeben. Was macht das Programm mit dieser Zahl? Was gibt das Programm zum Beispiel aus, wenn du die Zahl 7 eingibst? Stelle zuerst eine Vermutung auf, bevor du es mit dem Computer ausprobierst.

```
1  def rechne(x):
2      summe = 0
3      while x > 0:
4          summe += 3
5          x -= 1
6      print summe
7
8  zahl = input("Gib eine Zahl ein:")
9  if zahl > 0:
10     rechne(zahl)
```

2 Du hast drei Zahlen **a**, **b**, **c**. Schreibe ein Programm, das prüft, ob alle drei Zahlen unterschiedlich sind. Wenn ja, gibt es aus: «Alle drei Zahlen sind unterschiedlich.» Wenn nein, gibt das Programm aus: «Es gibt mindestens zwei gleiche Zahlen.»

3 Entwickle ein Programm, das die Figur in der Abbildung zeichnet. Du hast diese Figur bereits in Aufgabe 3 des «Teste dich selbst» von Kapitel 5 gesehen. Starte mit dem grössten Dreieck. Das nächstkleinere Dreieck hat jeweils eine halb so lange Seitenlänge. Dein Programm soll so lange Dreiecke zeichnen, bis die Seitenlänge des Dreiecks kleiner wird als 3.



4 Entwickle ein Programm, bei dem du zuerst mit `input()` eine Zahl **n** eingeben kannst. Das Programm gibt dir dann alle Primzahlen aus, die kleiner oder gleich **n** sind.