

Happy little turtles.

2. Turtle-Grundlagen

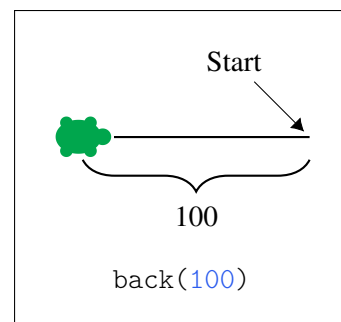
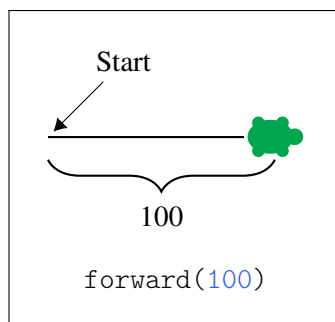
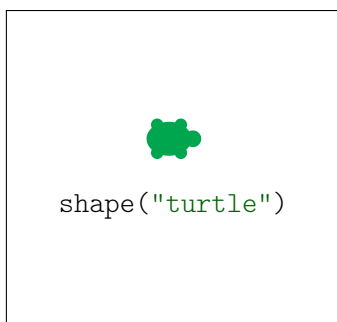
Am Ende dieses Kapitels werden Sie die grundlegenden Befehle der Turtle-Programmierung anwenden können. Die Lernziele sind:

- ☐ Die Turtle eine Anzahl Schritte vorwärts bzw. rückwärts bewegen.
- ☐ Die Turtle um einen beliebigen Winkel (in Grad) nach rechts bzw. links drehen.
- ☐ Die Turtle in den „Wandermodus“ bzw. „Zeichenmodus“ versetzen.
- ☐ Das Icon (kleines, ausgefülltes Dreieck) anpassen.
- ☐ Sie analysieren ein gegebenes Programm und erklären „was das Programm macht“.
- ☐ Sie erstellen ein Programm, welches eine gegebene geometrische Figur zeichnet.
- ☐ Sie erkennen Programmierfehler und korrigieren diese selbstständig.
- ☐ Sie wenden die Regeln für einen guten Programmierstil an.

2.1 Erste Schritte mit der Turtle

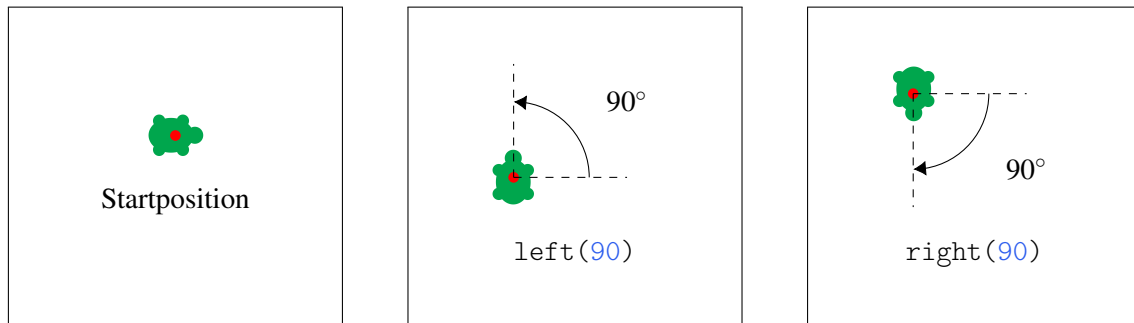
Wir besprechen nun die grundlegenden Konzepte.

- H** Die Turtle befindet sich zu Beginn in der Mitte des Fensters und schaut nach **rechts**. Die Turtle befindet sich zu Beginn immer im **Zeichenmodus**. Wenn sich die Turtle im Zeichenmodus befindet, hat sie einen Stift und zeichnet ihren Weg. Dies geschieht nur, wenn sich die Turtle vorwärts oder rückwärts bewegt.



Mit `shape("turtle")` können wir das **Icon** in eine Turtle ändern. Dies hat keinen Einfluss auf die Bewegungsabläufe der Turtle. Es wird lediglich das Icon ausgetauscht. Das Icon ändert sich nur dann, wenn wir es in das Programm einbauen. Das Icon ändert sich erst, nachdem die entsprechende Code-Zeile abgearbeitet wurde. Mit dem Befehl `forward(distance)` bewegt sich die Turtle um

distance Schritte vorwärts. Dabei läuft sie in **Blickrichtung**. Der Befehl `back(distance)` bewirkt das Gegenteil. Die Turtle bewegt sich um distance Schritte rückwärts. Sie läuft **entgegen** der **Blickrichtung**. Die Anzahl der „Schritte“ können wir frei wählen.



Mit `right(angle)` können wir die Turtle um angle Grad nach **rechts** drehen. Den Befehl `left(angle)` können wir gleichermassen für eine Linksdrehung verwenden. Die Turtle dreht sich dabei um angle Grad nach **links**. Die Drehung der Turtle bezieht sich **immer** auf die aktuelle Blickrichtung der Turtle.¹ Den Winkel können wir frei wählen. Die Drehung bewegt die Turtle weder vorwärts noch rückwärts - es wird **nichts** gezeichnet. Prinzipiell können die Befehle zum Bewegen und Drehen beliebig oft und mit unterschiedlichen Zahlen in einem Programm verwendet werden.

■ **Beispiel 2** Das Listing 2 zeigt, wie wir die Turtle mit mehreren Befehlen mehrfach bewegen und drehen können. In Abbildung 4 ist das Ergebnis grafisch dargestellt.

```

1  import turtle
2
3  turtle.shape("turtle")
4  turtle.left(90)
5  turtle.forward(50)
6  turtle.right(90)
7  turtle.forward(100)
8  turtle.right(135)
9  turtle.forward(100)
10 turtle.done()
11

```

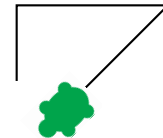


Abbildung 4: Resultat, wenn wir das Programm aus Listing 2 ausführen.

Listing 2: Beispielprogramm, welches die Figur aus Abbildung 4 zeichnet (`bsp_1.py`).

■

2.2 Programmausführung und Clean Code

Zuerst erklären wir, wie ein Python-Programm vom Computer ausgeführt wird, und gehen dann auf den Programmierstil ein.


Definition 5 — Programmausführung. Ein Python-Programm wird von **oben** nach **unten** ausgeführt. Der Computer liest Zeile für Zeile des Programms und führt nacheinander die entsprechenden Befehle aus. Bei einer Leerzeile passiert „nichts“. Es gibt Befehle, die Code-Zeilen überspringen oder zu einer vorherigen Code-Zeile zurückgehen.

¹Manchmal hilft folgender „Trick“: Versetzen Sie sich in die Turtle und führen Sie dann die Drehung aus.

Python führt also ein Programm immer ab Zeile 1 aus. Damit Programme übersichtlich und leserlich bleiben, werden bei der Software-Entwicklung bestimmte Regeln vereinbart. Typischerweise halten sich **alle** in einem Team an diese Regeln. Wenn wir gegenseitig Code austauschen und betrachten, finden sich so alle auf Anhieb zurecht. Ausserdem sollen die Regeln aufzeigen, wie wir unseren Programmierstil verbessern und dadurch verständlichere Programme erstellen können. Wir fassen diese Regeln unter dem Begriff **Clean Code** zusammen.² Die ersten Regeln sind:

 **Clean Code** — **Ein Befehl pro Zeile.** Wir notieren **einen** Befehl pro Zeile.

 **Clean Code** — **import-Befehle an den Anfang der Python-Datei.** Alle **import**-Befehle werden in den ersten Zeilen der Python-Datei notiert. Nach dem letzten **import**-Befehl fügen wir eine **Leerzeile** ein.

 **Clean Code** — **Leerzeile am Ende.** Wir notieren am **Ende des Programms** eine **Leerzeile**. Jede Python-Datei enthält also als letzte Code-Zeile eine leere Zeile.

2.3 Wandermodus

Mit `penup()` bringen wir die Turtle in den **Wandermodus**. Im Wandermodus bewegt sich die Turtle, **ohne** mit dem Stift zu zeichnen. Der Befehl `pendown()` bringt die Turtle in den **Zeichenmodus**. Im Zeichenmodus hat die Turtle einen Stift und zeichnet.

■ **Beispiel 3** Listing 3 zeigt, wie wir den Wander- und Zeichenmodus verwenden.

```
1  import turtle
2
3  turtle.shape("turtle")
4  turtle.forward(50)
5  turtle.penup()
6  turtle.forward(50)
7  turtle.pendown()
8  turtle.forward(50)
9  turtle.done()
10
```



Abbildung 5: Resultat, wenn wir das Programm aus Listing 3 ausführen.

Listing 3: Beispielprogramm, welches die Figur aus Abbildung 5 zeichnet (`bsp_2.py`).

²Wikipedia fasst den Begriff „perfekt“ zusammen: https://de.wikipedia.org/wiki/Clean_Code