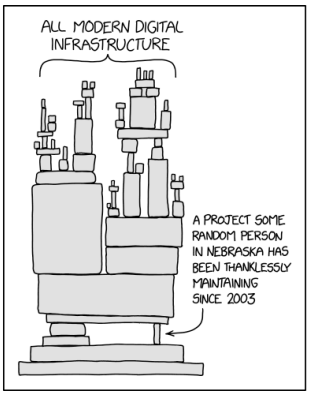
DEPENDENCY https://xkcd.com/2347/



Someday ImageMagick will finally break for good and we'll have a long period of scrambling as we try to reassemble civilization from the rubble.

5. Import und Funktionsaufruf

Bisher haben wir beim Programmieren stets von Befehlen gesprochen. Wir haben Befehle verwendet, um die Turtle zu bewegen und zu drehen. In diesem Abschnitt geht es darum, etwas genauer hinzuschauen. Wir können Befehle in unterschiedliche Kategorie einteilen. Wir befassen uns nun mit den ersten Kategorien, analysieren die bisherigen Befehle und führen neue Fachbegriffe ein. Folgende Ziele erreichen Sie nach diesem Kapitel:

- ☐ Sie erklären die neuen Fachbegriffe und nennen dazu Beispiele.
- ☐ Sie verknüpfen die Fachbegriffe mit einem gegebenen Python-Programm.

5.1 Befehle analysieren

Wir besprechen die neuen Fachbegriffe mithilfe von Listing 5.1. Es ist das bereits bekannte Programm für ein Quadrat der Seitenlänge 100. Das Programm besteht aus zehn Befehlen, die wir zwei Kategorien zuordnen können:

- `import`-Anweisung
- Funktionsaufruf

Woher weiss ich, zu welcher Kategorie ein Befehl gehört? Dies müssen Sie in der Beschreibung zur Programmiersprache nachlesen. Wir nennen diese Beschreibung, **Dokumentation**. Die Dokumentation von Python ist unter `https://docs.python.org/` abrufbar. Wir fassen in diesen Unterlagen für ausgewählte Befehle das wichtigste aus der Dokumentation zusammen.

```
1  import turtle
2
3  turtle.forward(100)
4  turtle.left(90)
5  turtle.forward(100)
6  turtle.left(90)
7  turtle.forward(100)
8  turtle.left(90)
9  turtle.forward(100)
10 turtle.left(90)
11 turtle.done()
```

Listing 5.1: Python-Programm für ein Quadrat.

5.2 `import`-Anweisung

Der Befehl in der ersten Zeile aus Listing 5.1 (`import turtle`) gehört zur Kategorie „**`import`-Anweisung**“.
Möchten wir bestehende Programme in unserem eigenen Python-Programm verwenden, dann können wir die entsprechenden Programme hinzufügen. Dadurch erhalten wir Zugriff auf bereits existierende Befehle. Die `import`-Anweisung macht genau dies. Die bestehenden Programme werden **Module** genannt. Software-Entwickler können Module erstellen und diese zur Verfügung stellen. Jedes Modul besitzt dabei einen **Namen**. Diesen Namen geben wir bei der `import`-Anweisung an. Damit sucht Python nach dem Modul und fügt die darin enthaltenen Programmierelemente dem Programm hinzu. Die `import`-Anweisung lautet allgemein:

```
import modulname
```

Wir müssen `modulname` dann in unserem Programm durch den gewünschten Modulnamen ersetzen.

Wichtig! Achten Sie darauf, dass zwischen `import` und dem Modulnamen zwingend ein Leerzeichen eingefügt werden muss. ■

Es ist die Aufgabe des Programmierers sicherzustellen, dass der Modulname existiert.

■ **Beispiel 5.1** Unsere bisherigen Programme verwenden das Turtle-Modul. Mit `import turtle` erhalten wir die Befehle, um die Turtle zu bewegen. Es gibt noch weitere Module, die wir einsetzen werden. Wir können etwa mit `import random` Befehle verwenden, um zufällige Zahlen zu erzeugen oder mit `import math` erhalten wir Zugriff auf Befehle für mathematische Berechnungen (zum Beispiel Quadratwurzel ermitteln). ■

5.2.1 Worin liegt der Vorteil?

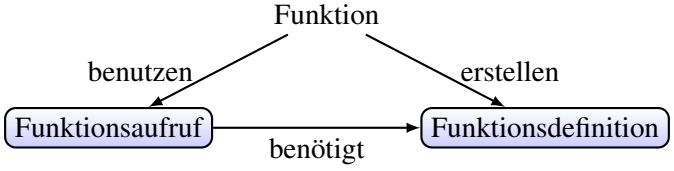
Häufig verwendete Programmierelemente müssen nicht selbst erstellt werden. Wir können auf bestehende Module zurückgreifen und reduzieren dadurch den Programmieraufwand.

5.3 Funktionsaufruf

Die Befehle in den Zeilen drei bis zehn und der Befehl in Zeile zwölf aus Listing 5.1 gehören alle zur Kategorie „**Funktionsaufruf**“. Jeder dieser Befehle stellt einen Funktionsaufruf (eng. function call) dar.
Wir können eine existierende Funktion ausführen, in dem wir einen **Funktionsaufruf** verwenden. Ein Funktionsaufruf führt Code aus, der unter einem Namen zusammengefasst ist. Wir erkennen einen Funktionsaufruf wie folgt:

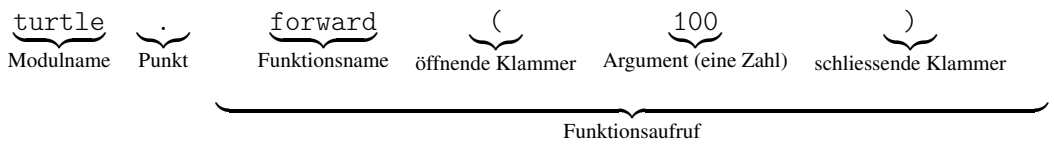
```
funktionsname(argumente)
```

Ein Argument ist unter anderem eine Zahl oder ein Text, den wir beim Funktionsaufruf übergeben. Es gibt auch Funktionen, die kein Argument oder mehrere Argumente verlangen.
Vorerst „bedienen“ wir uns an bereits existierenden Funktionen. Wir werden später sehen, wie wir **selbst** Funktionen erstellen können (Funktionsdefinition).



Schauen wir uns nun einen bekannten Funktionsaufruf im Detail an.

■ **Beispiel 5.2** Wir rufen die Funktion `forward` mit dem Argument `100` auf. Da die Funktion aus dem Turtle-Modul stammt, müssen wir den Modulnamen vor den Funktionsaufruf notieren. Modulname und Funktionsaufruf werden durch einen **Punkt** miteinander verbunden.



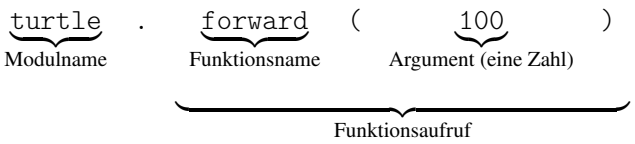
■

5.4 Aufgaben

In den folgenden Aufgaben repetieren Sie die Fachbegriffe und setzen sich mit der Python-Dokumentation auseinander.

5.4.1 Aufgabe 1

Analysieren Sie alle bisherigen Funktionsaufrufe. Beispiel:




```
turtle.backward(50)
```

```
turtle.left(90)
```

```
turtle.right(45)
```

```
turtle.pu()
```

```
turtle.pd()
```

```
turtle.done()
```

5.4.2 Aufgabe 2

1. Wer hat die Idee mit den Turtle-Grafiken erfunden und wann? Sie finden die Antwort in der Dokumentation für das Turtle-Modul. Starten Sie unter `https://docs.python.org/3/` und suchen Sie die Webseite des Turtle-Moduls.

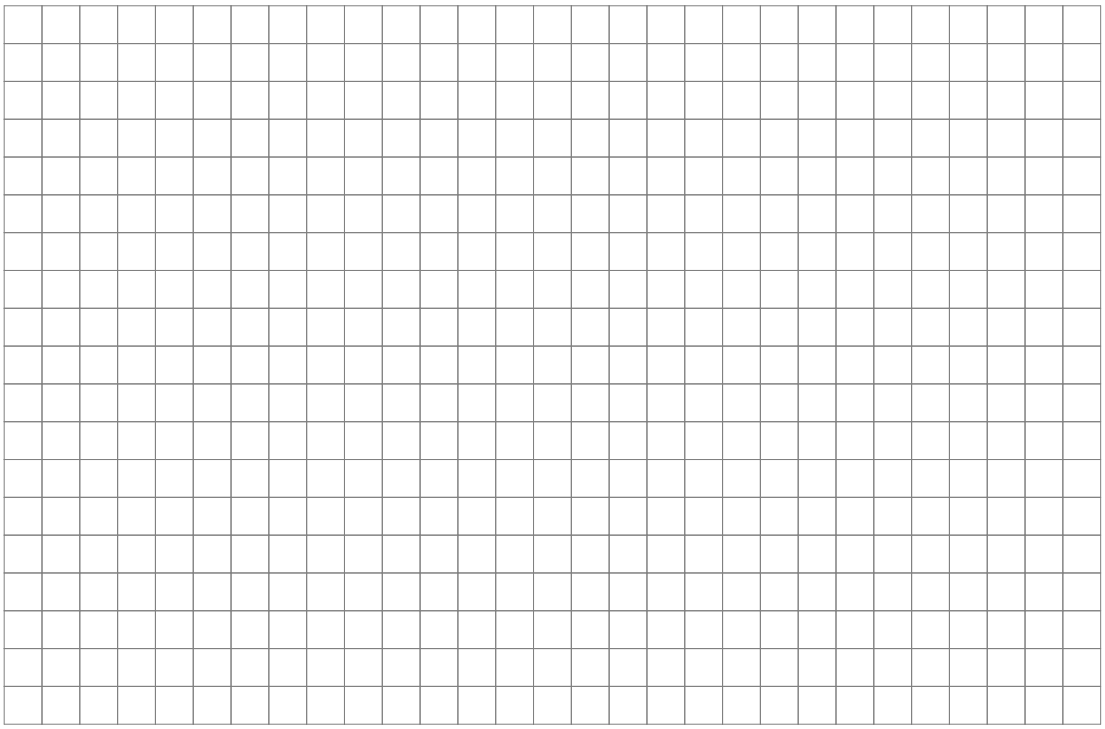
2. Welche Funktionsnamen können Sie für `forward`, `backward`, `left`, `right`, `penup` und `pendown` auch verwenden? Benutzen Sie die Dokumentation.

3. Mit welchem Funktionsaufruf können Sie die Turtle „unsichtbar“ machen? Welcher Vorteil ergibt sich dadurch? Wie können Sie die Turtle wieder anzeigen?

4. Mit welchem Funktionsaufruf können Sie die Zeichengeschwindigkeit der Turtle beeinflussen? Welche Stufen gibt es?

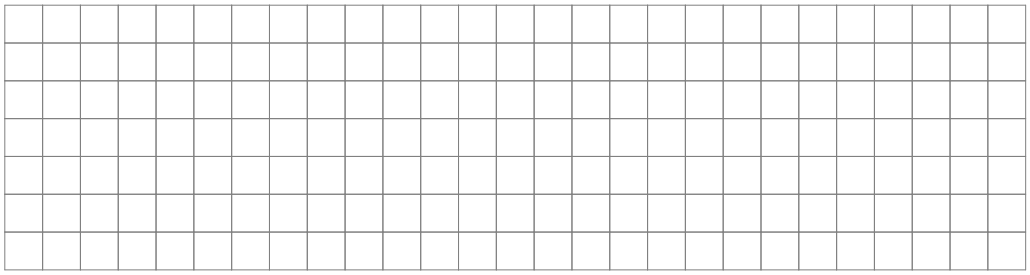
5.4.3 Aufgabe 3

Erstellen Sie handschriftlich ein Python-Programm, welches ein Quadrat zeichnet. Die Turtle soll nach der Hälfte des Quadrats unsichtbar werden. Stellen Sie die Zeichengeschwindigkeit der Turtle auf das Minimum (möglichst langsam zeichnen).

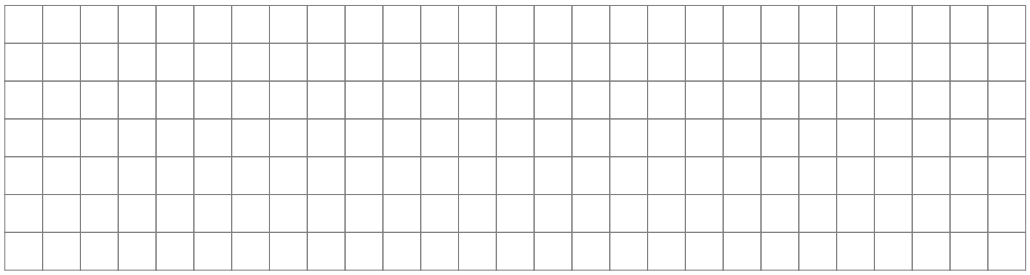


5.4.4 Aufgabe 4

1. Wie berechnen wir mit Python die Quadratwurzel von 42? Finden Sie die passende Funktion in der Dokumentation des Moduls `math`. Notieren Sie den Code hier.



2. Wie erzeugen wir mit Python eine Zufallszahl zwischen 1 und 100? Finden Sie die passende Funktion in der Dokumentation des Moduls `random`. Notieren Sie den Code hier.



3. Wie prüfen wir mit Python, ob ein Jahr (z.B. 2017) ein Schaltjahr war? Finden Sie die passende Funktion in der Dokumentation des Moduls `calendar`. Notieren Sie den Code hier.