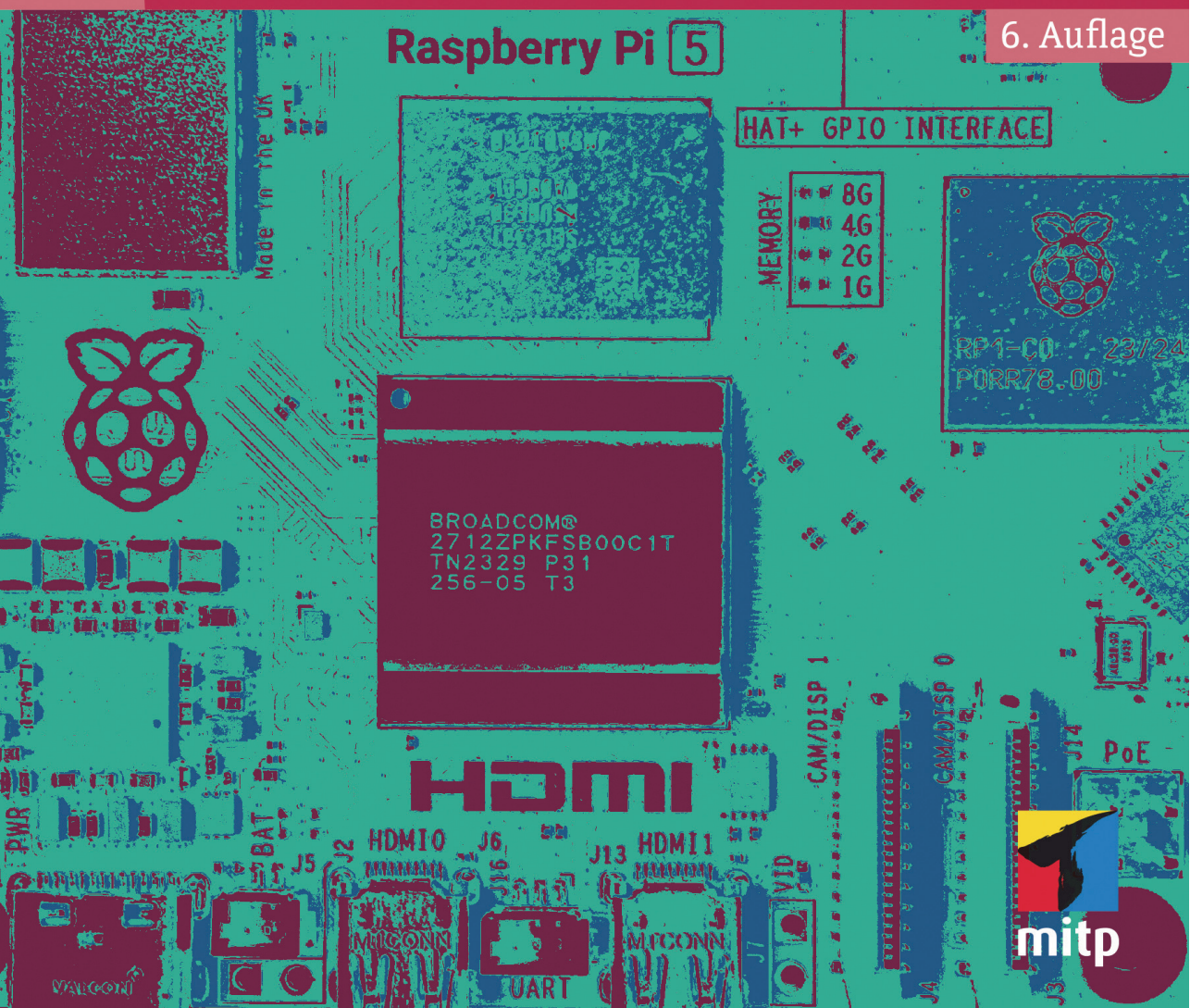
Michael Weigend

Raspberry Pi programmieren mit Python

6. Auflage



mitp

Michael Weigend

Raspberry Pi programmieren mit Python

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

ISBN 978-3-7475-0830-5

6. Auflage 2024

www.mitp.de
E-Mail: mitp-verlag@sigloch.de
Telefon: +49 7953 / 7189 - 079
Telefax: +49 7953 / 7189 - 082

Lektorat: Janina Vervost
Sprachkorrektorat: Petra Heubach-Erdmann, Christine Hoffmeister
Covergestaltung: Christian Kalkert
Bildnachweis: © Michael Weigend
Satz: III-satz, Kiel, www.drei-satz.de

Inhaltsverzeichnis

Einleitung

Der Raspberry Pi – kurz RPi – ist ein preiswerter, kreditkartengroßer Computer, der fast keinen Strom verbraucht, eine SD-Karte als Peripheriespeicher verwendet und an einen hochauflösenden Monitor angeschlossen werden kann. Mittlerweile gibt es ihn in vielen Varianten. Das neuste Modell ist der Raspberry Pi 5, der nicht mehr ganz so billig, aber sehr leistungsfähig ist. Es gibt aber eine preiswerte Alternative, den Raspberry Pi Zero W. Alle Projekte in diesem Buch funktionieren auf beiden Modellen und auf älteren RPis ab Modell 3.

Der RPi beflügelt die Fantasie von Bastlern, professionellen Technikern und Wissenschaftlern. In Kombination mit der Programmiersprache Python bietet er eine wunderbare Umgebung zur Realisierung technischer Ideen.

Dieses Buch erklärt alles von Grund auf. Es werden keine Vorkenntnisse zu Linux, zur Programmierung und zur Hardware des Raspberry Pi vorausgesetzt. Im Anhang finden Sie Hinweise zur Hardware und eine Schritt-für-Schritt-Anleitung zur Installation des Betriebssystems.

Was macht man mit dem Raspberry Pi?

Dieses Buch ist eine Einführung in die Programmiersprache Python auf dem Raspberry Pi. Doch die Beschäftigung mit dem Raspberry Pi ist oft nicht nur reine Programmierung. Ziel eines typischen RPi-Projekts ist der Prototyp einer kompletten Maschine – Hardware und Software. Der RPi legt Technik, die sonst versteckt ist, offen. Auf dem Markt gibt es eine zunehmende Zahl elektronischer Bauteile, die man mit dem RPi verbinden kann. Zudem gibt es immer mehr Firmen, die das benötigte Material im Internet anbieten. Bestellung und Lieferung der oft sehr speziellen Bauteile sind heute kein Problem.

Dieses Buch will eine Idee vom Charme der Programmiersprache Python vermitteln. Darüber hinaus soll es inspirieren, das gelernte Programmierwissen in konkrete Projekte einfließen zu lassen. Damit die Beschreibung von Hardwaretechnik und spezieller Schnittstellen nicht ins Uferlose wächst, gehe ich von fünf allgemeinen Hardwarekonfigurationen aus.

Interaktives Exponat

Auf dem Raspberry Pi läuft ein interaktives Programm mit grafischer Benutzungsoberfläche. Das kleine Gerät ist hinter einen großen Touchscreen geklebt und nicht zu sehen. Der RPi startet das Programm automatisch beim Einschalten. Eine solche Anordnung kann ein interaktives Exponat einer Ausstellung oder ein Auskunftssystem im Foyer eines öffentlichen Gebäudes sein. Da der RPi Grafik in HD-Qualität unterstützt, ist er für diesen Zweck hervorragend geeignet. Projekte dieser Art sind Maschinen, die Bilder und Texte automatisch erzeugen (Kapitel 5), digitale Karteikästen, Staumelder, die Informationen aus dem Internet auswerten und einen Überblick über die aktuelle Verkehrslage geben (Kapitel 6), digitale Bilderrahmen und Kalender (Kapitel 7) oder Animationen, die ihr Verhalten ändern, wenn sich jemand dem Bildschirm nähert (Kapitel 9).

System mit speziellen Eingabegeräten

Bei diesem Typ ist der RPi mit Sensoren oder einer Kamera verbunden. Auf dem Computer läuft ein Programm, das auf Signale dieser Sensoren reagiert. Das kann z.B. ein Spiel sein, bei dem Objekte auf dem Bildschirm über Fußschalter gesteuert werden. Die Sensoren kann man sich mit wenigen Elektronikbauteilen (Kabeln, Widerständen, Thermoelementen, AD-Wandlern) und Alltagsmaterialien (Pappe, Alufolie, Schaumgummi) zusammenbauen. Schon in den ersten beiden Kapiteln finden Sie einfache Beispiele für Programme, die Signale externer Schalter verarbeiten und LEDs oder Relais ansteuern: z.B. Zähler, Alarmanlagen. Komplexere Projekte mit einer grafischen Oberfläche sind z.B. ein Pong-Spiel, bei dem ein Schläger auf dem Bildschirm mit einer selbst gebauten Konsole bewegt wird (Kapitel 8) oder eine Simulation, bei der Sie ein Auto mit einem Potentiometer steuern. Ein ganz besonderes Eingabegerät ist das Kameramodul des Raspberry Pi. In Kapitel 10 werden Projekte vorgestellt, bei denen das Livebild der Kamera auf dem Bildschirm dargestellt und ausgewertet wird. Bewegungen werden erkannt und sogar die Bewegungsrichtung eines Objekts erfasst. Für den RPi gibt es eine Reihe von speziellen Sensoren (Kohlendioxid, Alkohol, Licht), mit denen man Geräte wie Spektralphotometer oder naturwissenschaftliche Experimente entwickeln kann, die automatisch oder halbautomatisch durchgeführt und ausgewertet werden (Kapitel 9).

System mit speziellen Ausgabegeräten

Viele Computerprogramme verwenden den Monitor als Ausgabegerät für Texte oder Bilder. Typische RPi-Projekte experimentieren aber auch mit anderen Ausgabegeräten. Zum Beispiel kann man die aktuelle Uhrzeit oder Temperatur auch auf einem OLED-Display anzeigen (Kapitel 12) oder über eine Bluetooth-Box als gesprochenen Text ausgeben (Kapitel 9). Mithilfe von MOSFET-Chips können Sie farbige LED-Strips ansteuern (Kapitel 1).

Mobiles Gerät

Der RPi ist klein und braucht wenig Strom. Er ist deshalb sehr gut für mobile Geräte und autonome Roboter geeignet. Bei den Projekten in diesem Buch steht die Programmierung im Vordergrund. Die Hardware ist möglichst einfach und enthält Bauteile (Steckplatinen, LEDs, Widerstände, Thermoelement, Ultraschallsensor, Kamera, OLED-Display), die man auch noch für andere Vorhaben verwenden kann. Typische Anwendungen sind mobile Messgeräte, die Messwerte (z.B. die Temperatur) speichern, oder ein Suchgerät, das in der Lage ist, die heißeste Stelle im Raum zu finden (Kapitel 9). In Kapitel 11 finden Sie ein Beispiel für eine mobile Webcam. Eine spezielle Erweiterung des RPi ist die HAT-Technik. In Kapitel 12 wird das Sense-HAT vorgestellt, eine Zusatzplatine, die man auf den RPi aufschraubt und die einige Sensoren und eine LED-Matrix enthält.

Server für spezielle Aufgaben

Der RPi kostet wenig und benötigt eine elektrische Leistung von nur 3,5 Watt (Modell B). Damit ist er der ideale Server, der permanent arbeitet und ständig bereit ist, Anfragen über das Internet oder Intranet zu beantworten (HTTP-Server). Ein solcher Server braucht weder Tastatur noch Monitor. Er kann über eine VNC-Verbindung von einem anderen Rechner aus gesteuert werden. In Kapitel 11 finden Sie Beispiele für serverbasierte Projekte, darunter eine Webcam.

Warum überhaupt Python auf dem RPi?

Ursprünglich sollte der RPi mit einem fest eingebauten Interpreter für Python-Programme ausgestattet werden (Pi steht für *Python interpreter*). Aber letztlich ist das Design doch flexibler geworden. Betriebssystem und Programmiersprachen können nach Wunsch installiert werden. Python ist besonders leicht zu lernen und erlaubt aber dennoch die Entwicklung komplexer Programme. Ein Vorteil für experimentelle Projekte mit dem Raspberry Pi ist, dass ein reicher Schatz freier Softwarepakete zur Verfügung steht, die man aus dem öffentlichen Python Package Index (PyPI) herunterladen und installieren kann.

Zum Aufbau dieses Buchs

In den Kapiteln werden Schritt für Schritt die wesentlichen Elemente der Python-Programmierung eingeführt. Ab Kapitel 2 werden kleine in sich abgeschlossene Projekte beschrieben, die praktische Anwendungsmöglichkeiten der zuvor eingeführten Techniken illustrieren. Dabei spielen in den ersten Kapiteln Peripheriegeräte noch keine Rolle. Der RPi wird in einer Standard-Hardwarekonfiguration mit Tastatur, Maus und Monitor benutzt wie ein normaler Computer. Die Elemente der Programmiersprache Python werden Schritt für Schritt eingeführt, von den

elementaren Grundlagen bis zu fortgeschrittenen Techniken der objektorientierten Programmierung. Die spannenderen Projekte kommen weiter hinten. Haben Sie also zu Beginn etwas Geduld.

In Kapitel 5 werden grafische Benutzungsoberflächen eingeführt. Die meisten Projekte sind nun Anwendungsprogramme, die nicht mehr auf eine Tastatur angewiesen sind und leicht zu interaktiven Exponaten für Ausstellungen oder Ähnlichem weiterentwickelt werden können. Etwa ab der Mitte des Buchs enthält jedes Kapitel Anregungen und Beispiele für Projekte mit speziellen Hardware-Komponenten, wie Kamera, Temperatur-Sensoren, AD-Wandlern, Schaltern und LEDs. Am Ende jedes Kapitels finden Sie Aufgaben mit Lösungen zum Download, mit denen Sie Ihr Wissen festigen, erweitern und vertiefen können.

Speziellere Informationen zum Betriebssystem (Installation, Autostart) und zur Hardware des RPi (GPIO) finden Sie in den Anhängen.

Lösungen und Programmcode zum Download

Unter der Webadresse `http://www.mitp.de/0829` können Sie die Listings aus dem Buch sowie die Lösungen zu den Aufgaben kostenlos downloaden.

Außerdem finden Sie die Lösungen im Buch hinter dem Stichwortverzeichnis.

Kapitel 1

Begegnung mit Python

In diesem Kapitel verwenden Sie Python im interaktiven Modus. Sie geben in der Kommandozeile der IDLE-Shell einzelne Befehle ein, die der Python-Interpreter sofort ausführt. Sie lernen Operatoren, Datentypen, die Verwendung von Funktionen und den Aufbau von Termen kennen. Dabei bekommen Sie einen ersten Eindruck vom Charme und der Mächtigkeit der Programmiersprache Python. Ich gehe davon aus, dass Sie bereits ein fertig konfiguriertes Computersystem besitzen, bestehend aus SD-Karte, Tastatur, Netzteil, Monitor und natürlich – als Herzstück – den Raspberry Pi, der meist als Raspi oder RPi abgekürzt wird. Auf der SD-Speicherkarte ist als Betriebssystem Raspberry Pi OS installiert. Das neuste und leistungsstärkste Modell des Minicomputers ist der Raspberry Pi 5. Alle Projekte in diesem Buch funktionieren aber auch auf den älteren Modellen 3 und 4 und auch auf dem besonders preiswerten Raspberry Pi Zero WH. Beachten Sie, dass der kleine Raspberry Pi Zero sehr langsam ist. Haben Sie mit ihm Geduld und halten Sie nur möglichst wenige Fenster geöffnet.

Falls Sie noch nicht so weit sind, können Sie in Anhang A nachlesen, welche Komponenten Sie benötigen und wie Sie bei der Einrichtung Ihres RPi-Systems vorgehen.

1.1 Was ist Python?

Python ist eine Programmiersprache, die so gestaltet wurde, dass sie leicht zu erlernen ist und besonders gut lesbare Programmtexte ermöglicht. Ihre Entwicklung wurde 1989 von Guido van Rossum am Centrum voor Wiskunde en Informatica (CWI) in Amsterdam begonnen und wird nun durch die nichtkommerzielle Organisation »Python Software Foundation« (PSF, `http://www.python.org/psf`) koordiniert. Das Logo der Programmiersprache ist eine stilisierte Schlange. Dennoch leitet sich der Name nicht von diesem Schuppenkriechtier ab, sondern soll an die britische Comedy-Gruppe Monty Python erinnern.

Ein Python-Programm – man bezeichnet es als Skript – wird von einem Interpreter ausgeführt und läuft auf den gängigen Systemplattformen (Unix, Windows, Mac OS). Ein Programm, das auf Ihrem Raspberry Pi funktioniert, läuft in der Regel auch auf einem Windows-Rechner. Python ist kostenlos und kompatibel mit der GNU General Public License (GPL).

Python ist objektorientiert, unterstützt aber auch andere Programmierparadigmen (z.B. funktionale und imperative Programmierung). Python ist eine universelle Programmiersprache mit vielen Einsatzmöglichkeiten. Es wird in der Wissenschaft und Technik verwendet (z.B. im Deutschen Zentrum für Luft- und Raumfahrt), aber auch für visuell-kreative Projekte (z.B. bei Disney zur Entwicklung von Computerspielen). Python hat gegenüber älteren Programmiersprachen drei Vorteile:

- Python ist leicht zu erlernen und hat eine niedrige »Eingangsschwelle«. Ohne theoretische Vorkenntnisse kann man sofort die ersten Programme schreiben. Im interaktiven Modus kann man einzelne Befehle eingeben und ihre Wirkung beobachten. Es gibt nur wenige Schlüsselwörter, die man lernen muss. Gewohnte Schreibweisen aus der Mathematik können verwendet werden, z.B. mehrfache Vergleiche wie $0 < a < 10$.
- Python-Programme sind kurz und gut verständlich. Computerprogramme werden von Maschinen ausgeführt, aber sie werden für Menschen geschrieben. Software wird meist im Team entwickelt. Programmtext muss für jedermann gut lesbar sein, damit er verändert, erweitert und verbessert werden kann. Der berühmte amerikanische Informatiker Donald Knuth hat deshalb schon vor drei Jahrzehnten vorgeschlagen, Programme als Literatur zu betrachten, so wie Romane und Theaterstücke. Nicht nur Korrektheit und Effizienz, auch die Lesbarkeit ist ein Qualitätsmerkmal.
- Programme können mit Python nachweislich in kürzerer Zeit entwickelt werden als mit anderen Programmiersprachen. Das macht Python nicht nur für die Software-Industrie interessant; auch Universitäten verwenden immer häufiger Python, weil so weniger Zeit für den Lehrstoff benötigt wird.

1.2 IDLE

Zur Standardinstallation von Python gehört eine integrierte Entwicklungsumgebung namens IDLE. Der Name erinnert an den englischen Schauspieler, Autor und Komponisten Eric Idle, ein Mitglied der Comedy-Gruppe Monty Python.

IDLE ist nicht (mehr) in der Standarddistribution von Raspberry Pi OS enthalten. Sie können es aber in wenigen Schritten installieren:

- Klicken Sie oben links auf den Startbutton mit der Himbeere, dann auf EINSTELLUNGEN und dann auf ADD/REMOVE SOFTWARE (Software hinzufügen oder entfernen). Es erscheint ein Fenster wie in Abbildung 1.1.
- Geben Sie oben links als Suchbegriff `Idle` ein und drücken Sie ↵. Es erscheint eine Liste von Software-Angeboten.
- Rollen Sie etwas herunter und wählen Sie eine IDLE-Version, indem Sie die Checkbox vor dem Eintrag anklicken.

- Klicken Sie unten rechts auf OK.
- Zur Sicherheit müssen Sie im Authentifizierungsfenster Ihr Passwort eingeben.
- Dann wird IDLE installiert.

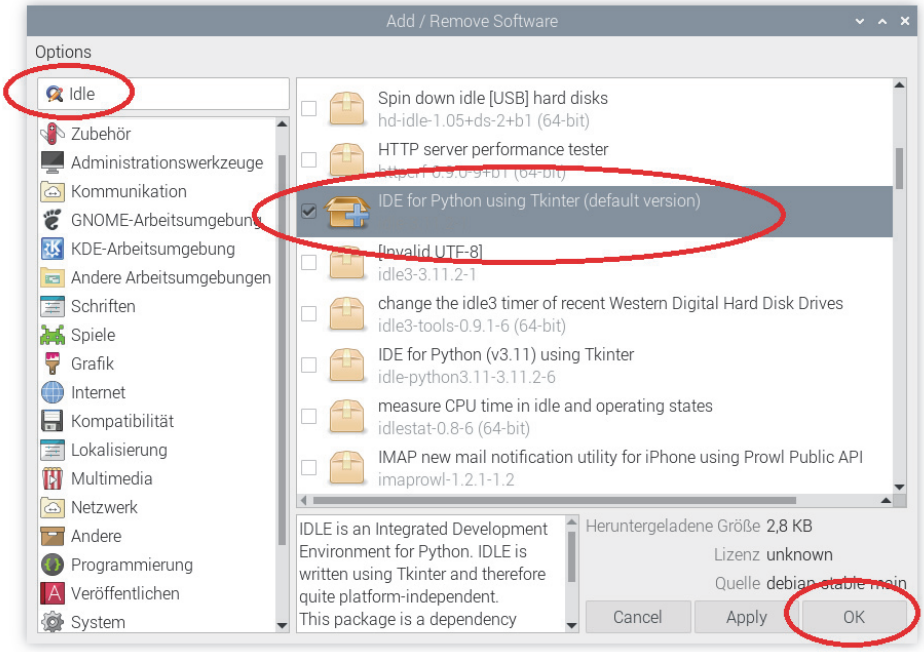


Abb. 1.1: IDLE installieren

Klicken Sie oben links auf den Startbutton mit der Himbeere und öffnen Sie das Untermenü ENTWICKLUNG. Sie sehen, dass das Menü einen neuen Eintrag enthält, z.B.: IDLE (USING PYTHON 3.11). Wenn Sie diesen Eintrag anklicken, öffnet sich die Entwicklungsumgebung. Beachten Sie, dass die Python-Version von Ihrem Raspberry-Pi-Modell abhängt. Auf dem kleinen RPi Zero kann eine ältere Version installiert sein als auf dem RPi 5.

IDLE besteht im Wesentlichen aus drei Komponenten:

- **Die IDLE-Shell.** Wenn Sie IDLE starten, öffnet sich zuerst das IDLE-Shell-Fenster. Die Shell ist eine Anwendung, mit der Sie direkt mit dem Python-Interpreter kommunizieren können: Sie können auf der Kommandozeile einzelne Python-Anweisungen eingeben und ausführen lassen. Ein Python-Programm, das eine Bildschirmausgabe liefert, gibt diese in einem Shell-Fenster aus.

- **Der Programmeditor.** Das ist eine Art Textverarbeitungsprogramm zum Schreiben von Programmen. Sie starten den Programmeditor vom Shell-Fenster aus (FILE|NEW FILE).
- **Der Debugger.** Er dient dazu, den Lauf eines Programms zu kontrollieren und zu überwachen, um logische Fehler zu finden.

Neben IDLE gibt es natürlich noch viele andere Entwicklungsumgebungen für Python. Auf dem RPi sind standardmäßig noch GEANY und THONNY installiert.

1.2.1 IDLE-Shell

Anweisungen sind die Bausteine von Computerprogrammen. Mit der IDLE-Shell können Sie einzelne Python-Anweisungen ausprobieren. Man spricht auch vom interaktiven Modus, weil Sie mit dem Python-Interpreter eine Art Dialog führen: Sie geben über die Tastatur einen Befehl ein – der Interpreter führt ihn aus und liefert eine Antwort.

Öffnen Sie das Shell-Fenster der Python-3-Version auf Ihrem Rechner (STARTBUTTON|ENTWICKLUNG|IDLE). Da Sie ständig mit IDLE arbeiten werden, lohnt es sich das Programmicon auf den Bildschirm zu bringen. Das geht so: Sie klicken mit der rechten Maustaste auf das Icon PYTHON 3 und wählen im Kontextmenü den Befehl ZUR ARBEITSFLÄCHE HINZUFÜGEN. Die IDLE-Shell meldet sich immer mit einer kurzen Information über die Version und einigen weiteren Hinweisen.

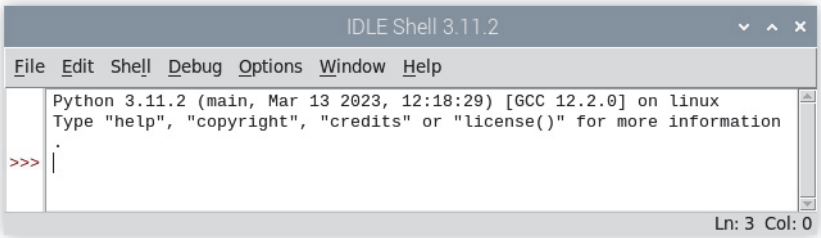


Abb. 1.2: Die IDLE-Shell

Die unterste Zeile beginnt mit einem Promptstring aus drei spitzen Klammern >>> als Eingabeaufforderung. Das ist die Kommandozeile. Hinter dem Prompt können Sie eine Anweisung eingeben. Wenn Sie die Taste ↵ drücken, wird der Befehl ausgeführt. In den nächsten Zeilen kommt entweder eine Fehlermeldung, ein Ergebnis oder manchmal auch *keine* Systemantwort. Probieren Sie aus:

```
>>> 2+2
4
```

Hier ist die Anweisung ein mathematischer Term. Wenn Sie ↵ drücken, wird der Term ausgewertet (also die Rechnung ausgeführt) und in der nächsten Zeile (ohne Prompt) das Ergebnis dargestellt.

```
>>> 2 +
SyntaxError: invalid syntax
```

Jetzt haben Sie einen ungültigen Term eingegeben (der zweite Summand fehlt). Dann folgt eine Fehlermeldung.

```
>>> a = 1257002
>>>
```

Eine solche Anweisung nennt man eine Zuweisung. Der Variablen `a` wird der Wert `1257002` zugewiesen. Dabei ändert sich zwar der Zustand des Python-Laufzeitsystems (es merkt sich eine Zahl), aber es wird nichts ausgegeben. Sie sehen in der nächsten Zeile sofort wieder der Prompt. Die gespeicherte Zahl können Sie wieder zum Vorschein bringen, indem Sie den Variablennamen eingeben:

```
>>> a
1257002
```

1.2.2 Hotkeys

Um effizient mit der IDLE-Shell arbeiten zu können, sollten Sie einige Tastenkombinationen (Hotkeys) beherrschen.

Manchmal möchten Sie ein früheres Kommando ein zweites Mal verwenden – vielleicht mit kleinen Abänderungen. Dann benutzen Sie die Tastenkombination Alt+P. Beispiel:

```
>>> 1 + 2*3 + 4
11
>>>
```

Wenn Sie jetzt *einmal* die Tastenkombination Alt+P betätigen, erscheint hinter dem letzten Prompt wieder das vorige Kommando (*previous*):

```
>>> 1 + 2*3 + 4
```

Wenn Sie nochmals diesen Hotkey drücken, verschwindet der Term wieder und es erscheint das *vorletzte* Kommando, beim nächsten Mal das vorvorletzte und so wei-

ter. Die Shell merkt sich alle Ihre Eingaben in einer Folge, die man *History* nennt. Mit Alt+P und Alt+N können Sie in der History rückwärts- und vorwärtsgehen (Abbildung 1.3).

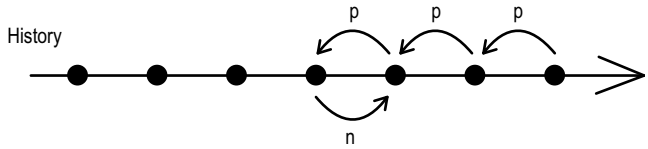


Abb. 1.3: Navigieren in der History mit Alt+P und Alt+N

Eine dritte wichtige Tastenkombination, die Sie sich merken sollten, ist Strg+C. Damit können Sie die Ausführung des gerade laufenden Programms abbrechen, z.B. wenn sie zu lange dauert.

Tastenkombination	Wirkung
Alt+P	Previous Command. Die vorige Anweisung in der History (Liste der bisherigen Anweisungen) erscheint hinter dem Prompt.
Alt+N	Next Command. Die nachfolgende Anweisung in der History erscheint hinter dem Prompt.
Strg+C	Keyboard Interrupt. Der Abbruch eines Programms (z.B. bei einer Endlosschleife) wird erzwungen.

Tabelle 1.1: Wichtige Tastenkombinationen der IDLE-Shell

1.3 Die IDLE-Shell als Taschenrechner

Die IDLE-Shell können Sie wie einen mächtigen und komfortablen Taschenrechner benutzen. Sie geben einen mathematischen Term ein, drücken ↵ und erhalten das Ergebnis in der nächsten Zeile.

1.3.1 Operatoren und Terme

Ein mathematischer Term (Ausdruck) kann aus Zahlen, Operatoren und Klammern aufgebaut werden. Die Schreibweise ist manchmal ein kleines bisschen anders als in der Schulmathematik. Zum Beispiel dürfen Sie beim Multiplizieren den Multiplikationsoperator * niemals weglassen. Das Kommando

```
>>> (1 + 1) (6 - 2)
```

führt zu einer Fehlermeldung. Korrekt ist:

```
>>> (1 + 1) * (6 - 2)
8
```

Es gibt keine langen Bruchstriche. Für Zähler oder Nenner müssen Sie Klammern verwenden:

```
>>> (2 + 3) / 2
2.5
```

Python unterscheidet zwischen der exakten Division / und der ganzzahligen Division //. Die ganzzahlige Division liefert eine ganze Zahl, und zwar den nach unten gerundeten Quotienten. Probieren Sie aus:

```
>>> 3/2
1.5
>>> 3//2
1
```

Die Modulo-Operation % liefert den Rest, der bei einer ganzzahligen Division übrig bleibt. Beispiel: 7 geteilt durch 3 ist 2 Rest 1.

```
>>> 7 // 3
2
>>> 7 % 3
1
```

Zum Potenzieren einer Zahl verwenden Sie den Operator **. Beachten Sie, dass Sie mit Python fast beliebig große Zahlen berechnen können.

```
>>> 2**3
8
>>> 2**-3
0.125
>>> 2**0.5
1.4142135623730951
>>> 137 ** 57
62097244310190258855130481009768710553783221891824568918264378730801621731509707020422858215922309341135893663853254591817
```

Bei Termen mit mehreren Operatoren müssen Sie deren Prioritäten beachten (Tabelle 1.2). Ein Operator mit höherer Priorität wird zuerst ausgewertet. Der Potenzoperator hat die höchste Priorität, Addition und Subtraktion die niedrigste.

```
>>> 2*3**2
18
>>> (2*3)**2
36
```

Operator	Bedeutung
**	Potenz, `x**y` = x^y
*, /, //	Multiplikation, Division und ganzzahlige Division
%	Modulo-Operation. Der Rest einer ganzzahligen Division.
+, -	Addition und Subtraktion

Tabelle 1.2: Arithmetische Operatoren in der Reihenfolge ihrer Priorität

1.3.2 Zahlen

Wer rechnet, verwendet Zahlen. Mit Python können Sie drei Typen von Zahlen verarbeiten:

- Ganze Zahlen (`int`)
- Gleitpunktzahlen (`float`)
- Komplexe Zahlen (`complex`)

Was ist überhaupt eine Zahl? In der Informatik unterscheidet man zwischen dem abstrakten mathematischen Objekt und der Ziffernfolge, die eine Zahl darstellt. Letzteres nennt man auch *Literal*. Für ein und dieselbe Zahl im mathematischen Sinne, sagen wir die `13`, gibt es unterschiedliche Literale, z.B. `13`oder `13.0` oder `13.0000`. Drei Schreibweisen – eine Zahl.

Ganze Zahlen (Typ int)

Literale für ganze Zahlen sind z.B. `12`, `0`, `-3`. Ganze Dezimalzahlen bestehen aus den zehn Ziffern `0`, `1`, `2`, `3`, `4`, `5`, `6`, `7`, `8`, `9`. Es kann ein negatives Vorzeichen vor die erste Ziffer geschrieben werden. Eine ganze Dezimalzahl darf nicht mit einer Null beginnen. Probieren Sie es aus!

```
>>> 023
SyntaxError: invalid token
```

Ganze Zahlen dürfen bei Python beliebig lang sein. Eine Grenze ist nur durch die Speicherkapazität des Rechners gegeben. Probieren Sie es aus:

Ganze Zahlen sind vom Typ `int` (*engl.* integer = unversehrt, ganz). Mit der Funktion `type()` können Sie den Typ eines Literals abfragen:

```
>>> type(13)
<class 'int'>
>>> type(13.0)
<class 'float'>
```

Sie sehen, dass die Literale `13` und `13.0` zu verschiedenen Typen gehören, obwohl sie den gleichen mathematischen Wert darstellen. Was hat es mit dem Begriff `class` auf sich? Der Typ `int` wird durch eine Klasse (engl. *class*) realisiert. Eine Klasse kann man sich als eine Art Bauplan für Objekte eines Typs vorstellen. In der Klasse `int` ist z.B. definiert, wie die Operationen für ganze Zahlen ausgeführt werden. Mehr Informationen zu Klassen finden Sie in Abschnitt 3.8.2.

Binär, oktal und hexadezimal – andere Schreibweisen für ganze Zahlen

Üblicherweise verwenden wir das dezimale Zahlensystem. Es gibt aber auch Binärzahlen (mit nur zwei Ziffern 0 und 1), Oktalzahlen (mit acht verschiedenen Ziffern) und Hexadezimalzahlen (mit 16 Ziffern).

Wie das Dezimalsystem ist auch das *Binärsystem* (Dualsystem, Zweiersystem) ein Stellenwertsystem für Zahlen. Aber anstelle von zehn Ziffern gibt es nur zwei, die Null und die Eins. Jede Zahl lässt sich als Summe von Zweierpotenzen (`1`, `2`, `4`, `8`, `16`, ...) darstellen. Die Binärzahl `10011` hat den Wert

```
1*16 + 0*8 + 0*4 + 1*2 + 1*1 = 16 + 2 + 1 = 19
```

Nun muss man natürlich erkennen können, ob eine Ziffernfolge wie `10011` als Dezimalzahl (zehntausendundelf) oder als Binärzahl gemeint ist. Deshalb beginnen bei Python Literale für Binärzahlen immer mit der Ziffer `0` und dem Buchstaben `b`, also z.B. `0b10011`. Wenn Sie in der IDLE-Shell eine solche Ziffernfolge eingeben und ↵ drücken, erscheint in der nächsten Zeile der Wert als Dezimalzahl:

```
>>> 0b10011
19
```

Mit der Funktion `bin()` können Sie zu einer Dezimalzahl die Binärdarstellung berechnen lassen:

```
>>> bin(19)
'0b10011'
```

Das *Oktalsystem* verwendet die Zahl 8 als Zahlenbasis. Jede Oktalzahl repräsentiert eine Summe aus Achterpotenzen (1, 8, 64, 512, ...). Bei Python beginnen die Literale von Oktalzahlen mit der Ziffer `0` und dem Buchstaben `o oder 0`. Beispiel:

```
>>> 0o210
136
```

Der Dezimalwert berechnet sich so: `2*64 + 1*8 + 0*1 = 128 + 8 = 136`

Im *Hexadezimalsystem* ist 16 die Zahlenbasis. Eine Hexadezimalzahl repräsentiert also eine Summe aus Potenzen der Zahl 16. Die 16 Ziffern werden durch die üblichen Dezimalziffern `0` bis `9` und die sechs ersten Buchstaben des Alphabets dargestellt. Dabei hat `A` den Wert `10`, `B` den Wert `11` usw. Bei Python beginnen Hexadezimalzahlen immer mit den Zeichen `0x` oder `0X`. Das erste Zeichen ist die Ziffer null und nicht der Buchstabe O. Beispiel:

```
>>> 0x10A
266
```

Der Dezimalwert berechnet sich so: `1*256 + 0*16 + 10*1 = 256 + 10 = 266`

Die Tatsache, dass die 16 Ziffern der Hexadezimalzahlen auch Buchstaben enthalten, hat Programmierer zum *Hexspeak* inspiriert. Zahlen, die in einem Programmsystem eine besondere Bedeutung haben (*magical numbers*), werden gerne so gewählt, dass ihre Hexadezimaldarstellung ein sinnvolles Wort ist.

```
>>> 0xCAFE
51966
>>> 0xBADBEEF
195935983
>>> xABAD1DEA
2880249322
```

Gleitkommazahlen (Typ float)

Gleitpunktzahlen oder Gleitkommazahlen (engl. *floating point numbers*) sind Dezimalbrüche. Meist schreibt man eine Gleitkommazahl als eine Folge von Ziffern mit einem einzigen Punkt auf. In der Schulmathematik verwenden wir in Deutschland ein Komma, in Python und allen anderen Programmiersprachen wird die angelsächsische Schreibweise verwendet, bei der ein Punkt an die Stelle des Kommas tritt. Es ist ein bisschen seltsam, von einer Gleitkommazahl zu sprechen und dann einen Punkt zu schreiben. Um diesen Widerspruch zu vermeiden, verwenden viele Leute den Begriff Gleich*punkt*zahl. »Gleitkommazahl« ist übrigens ein Gegenbegriff zu »Festkommazahl«. Eine Festkommazahl ist ein Dezimalbruch mit einer festen Anzahl von Nachkommastellen. Z.B. geben wir Geldbeträge in Euro immer mit zwei Nachkommastellen an. Wir schreiben 3,50 Euro anstelle von 3,5 Euro.

Gültige Python-Gleitkommazahlen sind:

- `3.14` oder `0.2` oder `0.00012`
- `.2` (eine Null vor dem Punkt darf man auch weglassen)
- `5.` (eine Null nach dem Punkt darf man weglassen)

Das Literal 5 ist dagegen keine Gleitkommazahl (Punkt fehlt).

Für Zahlen, die sehr nahe bei 0 liegen oder sehr groß sind, wird die Exponentialschreibweise verwendet. Dabei wird die Zahl als Produkt einer rationalen Zahl `m` (Mantisse) mit einer Zehnerpotenz mit dem Exponenten `e` dargestellt:

```
z = m*10^e
```

Beispiele:

```
123000000 = 123*10^6
0.00012 = 1.2*10^-4
```

Bei Python ist eine Gleitkommazahl in Exponentialschreibweise so aufgebaut: Sie beginnt mit einem Dezimalbruch oder einer ganzen Zahl (ohne Punkt) für die Mantisse. Danach folgt der Buchstabe `e` oder `E`, ein Vorzeichen (+ oder –), das bei positiven Exponenten auch weggelassen werden kann, und schließlich eine ganze Zahl als Exponent.

Gültige Literale sind:

`1.0e-8`	entspricht der Zahl 0.00000001
`2.1E+7`	entspricht der Zahl 21000000
`.2e0`	entspricht der Zahl 0.2
`001e2`	entspricht der Zahl 100 (mehrere führende Nullen sind erlaubt)

Ungültig sind:

`0.1-E7`	(Minuszeichen vor dem E)
`1.2e0.3`	(keine ganze Zahl als Exponent)

Mantisse und Exponent sind immer Dezimalzahlen und niemals Oktal- oder Hexadezimalzahlen.

Gleitkommazahlen sind vom Datentyp `float`. Mit der Funktion `type()` können Sie das nachprüfen:

```
>>> type(1.2)
<class 'float'>
```

Im Unterschied zu ganzen Zahlen (Typ `int`), die immer exakt sind, haben Gleitkommazahlen nur eine begrenzte Genauigkeit. Gibt man längere Ziffernfolgen ein, so werden die letzten Stellen einfach abgetrennt.

```
>>> 1.2345678901234567890
1.2345678901234567
```

1.3.3 Mathematische Funktionen

Wenn Sie die IDLE-Shell als Taschenrechner verwenden wollen, benötigen Sie auch mathematische Funktionen wie Sinus, Kosinus, die Quadratwurzelfunktion oder die Exponentialfunktion. Nun stellt Python eine Reihe von Standardfunktionen zur Verfügung, die gewissermaßen fest in die Sprache eingebaut sind (*built-in functions*). Wir haben schon die Funktion `type()` verwendet, die den Typ eines Objekts (z.B. einer Zahl) zurückgibt. Nur wenige Standardfunktionen haben eine mathematische Bedeutung (Tabelle 1.3). Der Aufruf einer Funktion ist so aufgebaut: Zuerst kommt der Name der Funktion, dahinter folgen in runden Klammern die Argumente. Das sind Objekte, die die Funktion verarbeitet, um daraus einen neuen Wert zu berechnen und zurückzugeben. Statt *Argument* sagt man manchmal auch *Parameter*. Beispiel:

```
>>> abs(-12)
12
```

Hier ist `abs` der Name der Funktion und die Zahl `-12` das Argument. Die Funktion `abs()` liefert den positiven Wert einer Zahl. Die Funktion ist einstellig, das heißt, sie akzeptiert immer nur genau ein Argument. Wenn Sie zwei Argumente angeben, erhalten Sie eine Fehlermeldung.

Es gibt aber auch Funktionen, die man mit einer unterschiedlichen Anzahl von Argumenten aufrufen kann. So liefert `min()` die kleinste Zahl von den Zahlen, die als Argumente übergeben worden sind.

```
>>> min (3, 2)
2
>>> min (2.5, 0, -2, 1)
-2
```

Die Funktion `round()` können Sie mit einem oder zwei Argumenten aufrufen. Das erste Argument ist die Zahl, die gerundet werden soll. Das zweite Argument ist optional und gibt die Anzahl der Nachkommastellen an, auf die gerundet werden soll. Fehlt das zweite Argument, gibt die Funktion eine ganze Zahl zurück.

```
>>> round(1.561)
2
>>> round(1.561, 1)
1.6
```

Funktion	Erklärung
`abs(x)`	Liefert den absoluten (positiven) Wert einer Zahl *x*.
`float(x)`	Liefert zu einer Zahl (oder einem anderen Objekt) eine Gleitkommazahl.
`int(x)`	Liefert zu einer Gleitkommazahl (oder einem anderen Objekt) eine nach unten gerundete ganze Zahl.
`max(x0, ..., xn)`	Liefert die größte Zahl von *x0, ... , xn*.
`min(x0, ..., xn)`	Liefert die kleinste Zahl von *x0, ... , xn*.
`round(x,[n])`	Die Zahl x wird auf *n* Stellen nach dem Komma gerundet und das Ergebnis zurückgegeben.

Tabelle 1.3: Mathematische Standardfunktionen (müssen nicht importiert werden)

Module importieren

Die meisten mathematischen Funktionen gehören nicht zu den Standardfunktionen. Will man sie benutzen, muss man zunächst das Modul `math` importieren. Module sind Sammlungen von Funktionen, Klassen und Konstanten zu einer Thematik. Sie sind so etwas wie Erweiterungen der Grundausstattung. Das Modul `math` enthält z.B. die Konstanten `e` und `pi` und mathematische Funktionen wie die Quadratwurzelfunktion `sqrt()`. Für Python gibt es Tausende von Modulen. Die wichtigsten gehören zum Standardpaket von Python und sind auf Ihrem RPi bereits installiert. Speziellere Module müssen zuerst aus dem Internet heruntergeladen werden. Dazu später mehr (z.B. in Abschnitt 9.8.1).

Sie können ein Modul auf verschiedene Weisen importieren. Beispiele:

```
>>> import math
>>> import math as m
>>> from math import *
>>> from math import pi, sqrt
```

Mit dem Befehl

```
>>> import math
```

importieren Sie den Modulnamen. Wenn Sie eine Funktion aufrufen wollen, müssen Sie dem Funktionsnamen noch den Modulnamen voranstellen.

```
>>> math.sqrt(4)
2.0
>>> math.pi
3.141592653589793
```

Sie können ein Modul unter einem anderen Namen importieren. Das ist vor allem dann praktisch, wenn ein Modul einen langen Namen hat, den man im Programmtext nicht immer wieder ausschreiben will.

```
>>> import math as m
>>> m.pi
3.141592653589793
```

Mit dem Befehl

```
>>> from math import *
```

importieren Sie alle Namen des Moduls `math`. Sie können dann die Funktionen und Konstanten ohne vorangestellten Modulnamen verwenden.

```
>>> sqrt(4)
2.0
>>> pi
3.141592653589793
```

Sie können auch gezielt nur die Namen importieren, die Sie auch verwenden wollen. Dann müssen Sie die Namen der benötigten Funktionen und Konstanten auflisten. Beispiel:

```
>>> from math import sqrt
>>> sqrt(4)
2.0
```

Mathematische Funktionen und Konstanten

Wenn Sie die IDLE-Shell als wissenschaftlichen Taschenrechner verwenden wollen, geben Sie einmal die Import-Anweisung

```
>>> from math import *
```