Einführung für Lehrkräfte

> **Fassung ohne Lösungen:** Dieses Handbuch enthält keine Musterlösungen. Die Lösungen und die genannten Programmdateien gibt es für Lehrkräfte auf Anfrage per E-Mail an info@picoboy.de.

Dieses Heft erklärt, was Sie vor der ersten Stunde wissen und vorbereiten sollten: das Gerät, die Programmierumgebung PBC+ Studio im Textmodus, die Wahl zwischen Emulator und Gerät, die Einrichtung im Computerraum und die Besonderheiten von MicroPython.

Auf einen Blick

Zielgruppe	ab Klasse 8, ohne Vorkenntnisse
Gerät	PicoBoy Color Plus (PBC+): Farbbildschirm 240 × 280 Pixel, Joystick (fünf Richtungen inkl. Drücken = MITTE), Tasten A und B, drei Status-LEDs, RGB-LED, Lautsprecher, Bewegungssensor, USB-C
Sprache	Python (MicroPython 1.29 auf dem Gerät und im Emulator)
Programmierumgebung	PBC+ Studio im Browser: **pbcstudio.de**, Chrome oder Edge, **Build 110** oder neuer, im **Textmodus**
Ohne Gerät?	Ja. Der eingebaute Emulator führt alle Programme aus; Ausnahmen: Bewegungssensor (liegt immer flach), Dateien bleiben nur bis zum Neuladen der Seite.
Anmeldung	keine. Kein Konto, keine Klassenverwaltung.
Material	26 Arbeitsblätter (AB 01–AB 24, AB 23 in drei Teilen), je ein Lehrermaterial mit Stundenverlauf und typischen Fehlern; das Schülerbuch „Programmieren mit Python“ (22 Kapitel und Zusatzkapitel „Objekte und Klassen“); L 01–L 07

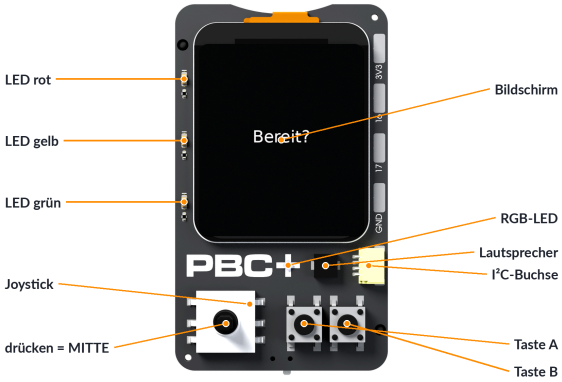
Das Gerät

Der PicoBoy Color Plus ist ein kleiner Computer im Format einer Handkonsole mit einem Mikrocontroller RP2350. Als Firmware läuft MicroPython mit den Zusatzmodulen `pbc` (Display, Tasten, LEDs, Ton, Sensor, Canvas, Sprites) und `turtle`. PBC+ Studio schickt das Programm über das USB-Kabel auf das Gerät und zeigt Ausgaben und Fehler im Terminal.

Das Koordinatensystem des Bildschirms beginnt **oben links** bei (0, 0); y wächst nach **unten**. Die Turtle-Grafik verwendet ein eigenes System mit (0, 0) in der Mitte und y nach oben (Kopiervorlagen A und C in L 07).

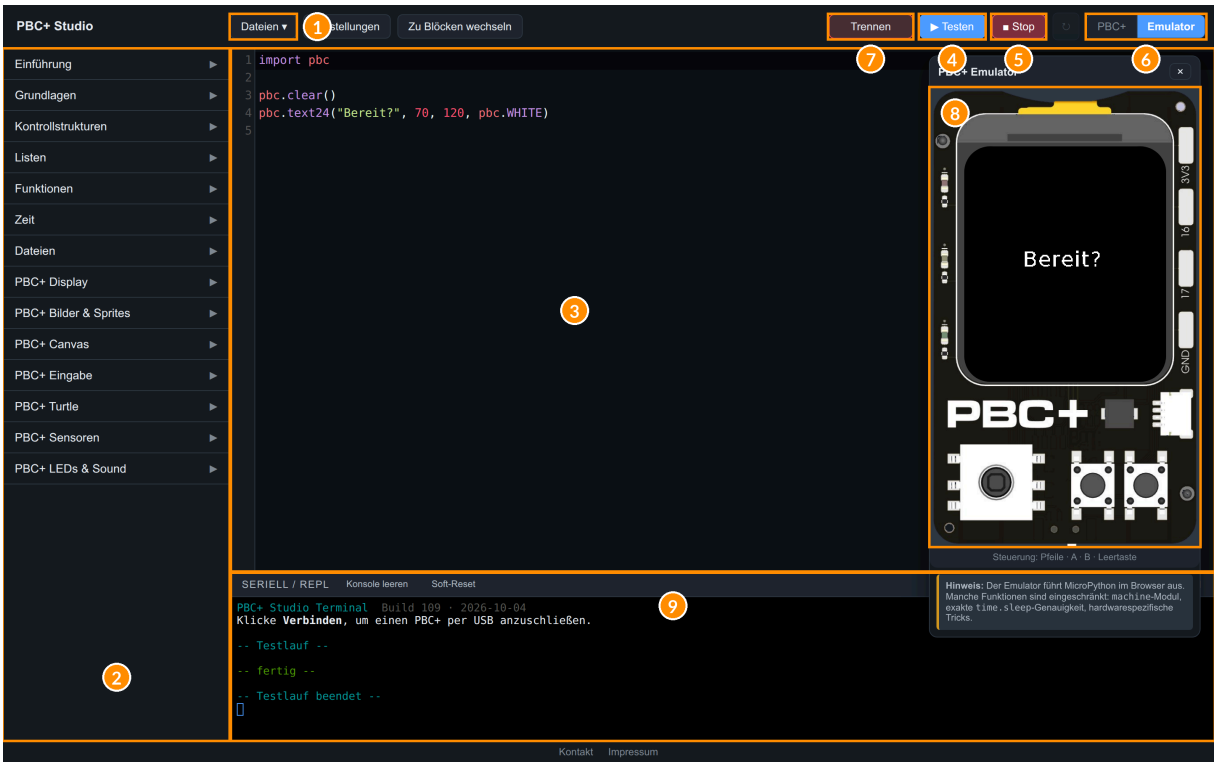
Dieses Handbuch wurde mit künstlicher Intelligenz (KI) generiert. Fehler sind möglich.

Umlaute: Die Display-Schriften kennen keine Umlaute; „Grün“ erscheint als „Gr??n“ (ein Fragezeichen pro Byte). In der **Konsole** (`print`, `input`) funktionieren Umlaute. Lassen Sie in Bildschirmtexten „ae, oe, ue, ss“ schreiben und Namen im Programm ohne Umlaute wählen.



PBC+ Studio im Textmodus

PBC+ Studio ist eine einzelne Webseite; eine Installation ist nicht nötig. Welche Version läuft, steht beim Start im Terminal („Build 110 · 2026-10-05“). Beim ersten Öffnen zeigt PBC+ Studio eine Blockansicht. Für dieses Material wechseln die Schülerinnen und Schüler **einmal** in den Textmodus: **Zu Text wechseln** → **Bearbeitung aktivieren** → **OK**. PBC+ Studio merkt sich den Modus und das letzte Programm im Browser.



Nr.	Bereich	Hinweis für die Lehrkraft
1	**Dateien**	Speichern/Öffnen (`.py`), Bilder, Dateien auf dem Gerät, Firmware, Startmenü
2	**Hilfe-Leiste**	Erklärungen mit Beispielen zu Python und allen `pbc`-Befehlen; **In den Editor einfügen** übernimmt ein Beispiel
3	**Editor**	mit Zeilennummern, Syntaxfarben und automatischem Einrücken nach `:`

Nr.	Bereich	Hinweis für die Lehrkraft
4 / 5	**▶ Testen / ■ Stop**	startet bzw. beendet das Programm im gewählten Ziel
6	**Ziel**	**Emulator** oder **PBC+**
7	**Verbinden / Trennen**	USB-Verbindung
8	**Emulator**	nachgebildeter PicoBoy, Tasten: Pfeile, Leertaste, A, B
9	**Terminal**	Ausgaben von `print`, Eingaben für `input`, Fehlermeldungen

> **Zu Blöcken wechseln** überschreibt den Python-Text mit dem Inhalt der Blockansicht. PBC+ Studio fragt vorher nach. Weisen Sie in der ersten Stunde darauf hin: bei der Rückfrage **Abbrechen**.

Fehlermeldungen: Bei einem Fehler zeigt das Terminal den Traceback von MicroPython mit der Zeilennummer im Editor, zusätzlich öffnet PBC+ Studio ein Hinweisfenster mit einer Erklärung in einfacher Sprache (z. B. „Unbekannter Name … Schreibe ganz oben in dein Programm die Zeile import pbc“). Das Lesen von Fehlermeldungen ist ab AB 01 ausdrücklich Unterrichtsgegenstand.

Imports: Im Textmodus ergänzt PBC+ Studio keine Imports. Fehlt `import pbc`, `import time` oder `import random`, gibt es – wie in jedem Python – einen `NameError`.

Emulator oder Gerät?

	Emulator	Gerät
Vorbereitung	keine	Geräte laden, USB-Kabel, Chrome/Edge mit Web Serial
Geschwindigkeit	etwa wie das Gerät (die Display-Übertragung wird nachgebildet)	–
Bewegungssensor	liegt immer flach (x = 0, y = 0, z = 9,81)	echt
Ton	über die Lautsprecher des Rechners	Lautsprecher am Gerät
Dateien (`open`)	bis zum Neuladen der Seite	dauerhaft im Flash
Bilder für Sprites	im Browser gespeichert	im Speicher des Geräts

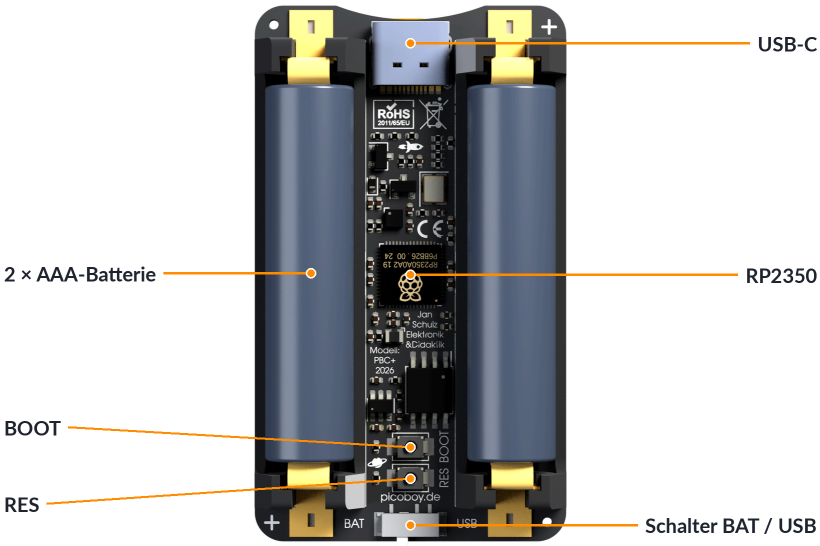
Alle Arbeitsblätter lassen sich im Emulator bearbeiten. Kapitel 20 des Buchs (Bewegungssensor) braucht Geräte. Mit Geräten ist die Motivation erfahrungsgemäß höher; eine bewährte Mischung: im Emulator entwickeln, am Ende der Stunde auf dem Gerät testen.

Einrichtung im Computerraum

- Chrome oder Edge; Zugriff auf **pbcstudio.de**, **cdnjs.cloudflare.com** und **cdn.jsdelivr.net** (dort lädt PBC+ Studio Editor, Terminal und Emulator).
- Für Geräte: USB-Ports zugänglich, Web Serial im Browser nicht durch Richtlinien gesperrt.
- Werden Browserdaten beim Abmelden gelöscht, müssen die Schülerinnen und Schüler den Textmodus jedes Mal neu einschalten und hochgeladene Bilder neu laden. Programme immer als Datei sichern (**Dateien → Auf den Computer…**).
- Bilder für Sprites (Ordner `material/`) vorab im Netzlaufwerk bereitstellen; sie werden mit **Dateien → Bilder verwalten…** hochgeladen (im Emulator in den Browser, am Gerät auf das Gerät), mehrere auf einmal mit **ZIP hochladen…**.

Firmware und Startmenü

Auf neuen PicoBoys ist MicroPython bereits installiert. Falls nicht: **Dateien → MicroPython installieren…** zeigt die Anleitung (BOOT halten, RES drücken, `.uf2`-Datei auf das Laufwerk RPI-RP2 ziehen). Dabei wird alles auf dem Gerät gelöscht. **Dateien → Startmenü installieren…** richtet ein Menü ein, aus dem gespeicherte Programme ohne Rechner gestartet werden können. Einzelheiten: Buch, Anhang D.



MicroPython im Unterricht – was anders ist

Das Material bleibt bei Python-Sprachmitteln, die in MicroPython und CPython gleich funktionieren. Unterschiede, die auffallen können:

- Fehlermeldungen sind kürzer und teils anders formuliert (`NameError: name 'x' isn't defined`).

- Einige Bibliotheksfunktionen fehlen oder heißen anders; `time.sleep_ms`, `time.ticks_ms` und `time.ticks_diff` sind MicroPython-spezifisch.
- Es gibt kein `turtle`-Modul im Sinne von CPython: Das PicoBoy-Modul hat eigene Befehle (`setDirection`, `setPenColor` …), siehe L 07 Vorlage E.

Wer möchte, kann reine Konsolenprogramme (Kapitel 3, 4, 7, 10, 12, 13) auch in einem PC-Python ausführen lassen – ein guter Anlass, über Plattformen und Bibliotheken zu sprechen.

Unterrichtsreihe

Stoffverteilungsplan für alle Arbeitsblätter, Varianten für kürzere und längere Reihen und eine Zuordnung zu den Kompetenzbereichen der Bildungsstandards Informatik der Gesellschaft für Informatik (GI) für die Sekundarstufe I.

Stoffverteilungsplan (Langfassung, etwa 32 Stunden)

Stunde	AB	Thema	Neue Konzepte	Buch
1	AB 01	PBC+ Studio kennenlernen	Editor, Testen, Fehlermeldung, `.py` speichern	1, 2
2	AB 02	Hallo PBC+!	Funktionsaufruf, Argumente, Koordinaten	2
3	AB 03	Frage und Antwort	`print`, `input`, `int`	3
4	AB 04	Rechnen mit Variablen	Variablen, Datentypen, `//`, `%`, Zufall, f-Strings	4
5	AB 05	Ein Bild aus Formen	Zeichenbefehle, Farben	5
6	AB 06	Muster mit Schleifen	`for`, `range`, verschachtelte Schleifen	6
7	AB 07	Der springende Ball	`while True`, Animation, Zeit	6
8	AB 08	Zahlen raten	`if/elif/else`, Vergleiche, `while` mit Bedingung	7
9	AB 09	Steuern mit dem Joystick	Tasten abfragen, Spielschleife, `and/or`	8
10	AB 10	Der Reaktionstest	`was_pressed`, Zeit messen	8
11	AB 11	Musik und Licht	LEDs, Töne	9
12	AB 12	Die Geheimschrift	Strings, Index, `ord` / `chr`	10
13	AB 13	Ordnung mit Funktionen	`def`, Parameter	11
14	AB 14	Werkzeuge mit Rückgabewert	`return`, Testen von Funktionen	12
15	AB 15	Listen 1: Bestenliste	Listen, `append`, `sum`, `max`, `sorted`	13
16–17	AB 16	Listen 2: Merk dir die Folge	Index, Listen und Funktionen	13
18	AB 17	Listen 3: Sternenregen	viele Objekte in Listen	14
19	AB 18	Der Highscore bleibt	Dateien lesen und schreiben	15

Stunde	AB	Thema	Neue Konzepte	Buch
20	AB 19	Turtle 1: Vielecke	Turtle, Funktionen mit Parametern	16
21	AB 20	Turtle 2: Die Turtle sieht Farben	`on_color`, Bedingungen	16
22	AB 21	Canvas und Sprites	Puffer, Sprites	17, 18
23	AB 22	Kollisionen	`touches`, `on_color`	19
24–26	AB 23a–c	Projekt Weltraumspiel	Zusammenführung	21
27–32	AB 24	Eigenes Projekt	Planen, Umsetzen, Testen, Präsentieren	22

Kürzere Varianten

Grundkurs, etwa 14 Stunden: AB 01–04, 06, 08, 09, 13, 14, 15, 17, dann AB 23a–c als gemeinsames Projekt. Zeichnen (AB 05) und Animation (AB 07) lassen sich als Hausaufgabe oder Differenzierung einsetzen.

Schwerpunkt Algorithmen ohne Grafik, etwa 10 Stunden: AB 01, 03, 04, 08, 12, 13, 14, 15, 16, 18. Diese Blätter arbeiten überwiegend mit der Konsole; die Programme laufen auch in einem PC-Python, wenn man die Zeichen- und Tastenbefehle weglässt.

Mit Vorkenntnissen (z. B. aus einer blockbasierten Sprache): AB 01 und 02 in einer Stunde, AB 03–07 als Stationenlernen, danach normal weiter.

Mit und ohne Geräte

Alle Arbeitsblätter funktionieren im Emulator. Mit einem Klassensatz Geräte lohnt es sich besonders bei AB 09–11 (Tasten, Licht, Ton) und bei den Projekten. Mit wenigen Geräten: im Emulator entwickeln, am Stundenende auf einem Gerät pro Gruppe vorführen. Das Buch hat mit Kapitel 20 (Bewegungssensor) ein Kapitel, das nur auf Geräten sinnvoll ist – geeignet als Zusatzangebot.

Zuordnung zu den Bildungsstandards (GI, Sek I)

Inhaltsbereich	Wo im Material
Algorithmen (Anweisungsfolgen, Verzweigungen, Schleifen; Algorithmen entwerfen, implementieren, testen)	AB 01–02, 06–10, 13–14, 16, 19–20

Inhaltsbereich	Wo im Material
Sprachen und Automaten (Syntax und Semantik einer Programmiersprache, Fehlermeldungen)	durchgehend; AB 01, 02, 08 (Fehler lesen)
Information und Daten (Datentypen, Variablen, Listen, Dateien; Kodierung von Zeichen und Farben)	AB 03–05, 12 (Zeichenkodierung), 15–18
Informatiksysteme (Aufbau eines Rechners, Ein- und Ausgabe, Sensoren)	AB 01, 09–11; Buch Kap. 20
Informatik, Mensch und Gesellschaft	Projekte AB 23–24 (Spielregeln, Fairness, Urheberrecht bei Bildern), Gesprächsanlass bei AB 12 (Verschlüsselung)

Prozessbereiche: **Modellieren und Implementieren** (alle AB), **Strukturieren und Vernetzen** (Funktionen AB 13–14, Listen AB 15–17), **Begründen und Bewerten** (Code-Lese- und Fehleraufgaben), **Kommunizieren und Kooperieren** (Projekte), **Darstellen und Interpretieren** (Koordinaten, Tabellen, Testprotokolle).

Hausaufgaben und Wiederholung

Jedes Buchkapitel endet mit Übungsaufgaben in drei Stufen und einer Zusammenfassung („Das Wichtigste in Kürze“). Die Lösungen stehen im Buch (Anhang C), die Programme liegen als `.py`-Dateien bei. Ohne Gerät und ohne Internet lassen sich die Aufgaben „Was gibt das Programm aus?“ und „Finde den Fehler“ auch auf Papier bearbeiten.

LEHRERMATERIAL · AB 01

PBC+ Studio kennenlernen

Überblick

Lernziel	Die Schülerinnen und Schüler öffnen PBC+ Studio im Textmodus, tippen ein erstes Python-Programm ab, testen es im Emulator, verändern Text, Farbe und Position, lesen eine Fehlermeldung und speichern das Programm als `.py`-Datei.
Dauer	45 Minuten (Aufgabe 1–5 Pflicht, 6 Differenzierung)
Voraussetzungen	keine
Material	Rechner mit Chrome oder Edge, Internetzugang (pbcstudio.de, PBC+ Studio Build 110 oder neuer). Optional PicoBoys mit USB-Kabel. Lösungsdateien auf Anfrage
Buch	Kapitel 1 und 2

Vorbereitung: Einmal selbst ausprobieren, ob pbcstudio.de im Schulnetz erreichbar ist. Beim ersten Öffnen zeigt PBC+ Studio die Blockansicht; der Wechsel in den Textmodus (**Zu Text wechseln** → **Bearbeitung aktivieren** → **OK**) ist nur einmal pro Browser nötig, danach startet PBC+ Studio im Textmodus. Auf Rechnern, deren Browserdaten beim Abmelden gelöscht werden, müssen die Schülerinnen und Schüler ihn jedes Mal wiederholen.

Stundenverlauf

Zeit	Phase	Inhalt	Sozialform
5 min	Einstieg	PicoBoy zeigen (oder Abbildung aus Buch 1.1): ein kleiner Computer, den man in Python programmiert. Ziel der Stunde: ein eigenes Programm darauf bringen.	Plenum
5 min	Aufgabe 1	Oberfläche erkunden, Nummern zuordnen	Einzel
15 min	Aufgabe 2–3	Abtippen, testen, ändern	Einzel
10 min	Aufgabe 4–5	Fehlermeldung lesen, speichern, öffnen	Einzel/Partner
5 min	Aufgabe 6	Hilfe-Leiste (für Schnelle)	Einzel
5 min	Sicherung	Fehlermeldung gemeinsam an der Tafel lesen: von unten nach oben, Fehlerart und Zeilennummer. „Fehler sind normal, sie zu lesen ist eine Fähigkeit.“	Plenum

Typische Fehler

Beobachtung	Ursache	Hilfe
Man kann im Editor nichts tippen.	Der Editor ist nach **Zu Text wechseln** noch schreibgeschützt.	**Bearbeitung aktivieren** rechts oben im Editor, Rückfrage mit **OK** bestätigen.
`SyntaxError: invalid syntax`, nichts erscheint.	Klammer oder Anführungszeichen fehlt, oft am Zeilenende.	Zeichen für Zeichen mit der Vorlage vergleichen; der Fehler steht in der genannten Zeile oder kurz davor.
`NameError: name 'pbc' isn't defined`	Zeile `import pbc` fehlt oder ist falsch geschrieben.	`import pbc` als erste Zeile. Das Hinweisfenster sagt das auch.
`AttributeError: module 'pbc' has no attribute 'Text24'`	Groß-/Kleinschreibung: Befehle klein, Farben groß.	Auf `pbc.text24` und `pbc.WHITE` achten.
Anführungszeichen „falsch“, `SyntaxError`	Typografische Anführungszeichen („…“) aus einem Textprogramm kopiert.	Mit **Umschalt + 2** gerade Anführungszeichen tippen.
Text ist rechts abgeschnitten.	x zu groß für die Textlänge.	x verkleinern.
Klick auf **Zu Blöcken wechseln**, Programm weg	Rückwechsel überschreibt den Python-Text.	Bei der Rückfrage **Abbrechen**; sonst Datei neu öffnen.

Differenzierung

- **Schneller fertig:** Aufgabe 6; danach mit PicoBoy: Ziel **PBC+**, **Verbinden**, Programm auf dem Gerät testen (Buch 1.4) und mit **Dateien → Auf den PBC+…** speichern (Buch 1.6).
- **Mehr Unterstützung:** `ab01_start.py` mit **Dateien → Vom Computer…** öffnen lassen statt abzutippen; Aufgabe 1 in Partnerarbeit.

LEHRERMATERIAL · AB 02

Hallo PBC+!

Überblick

Lernziel	Die Schülerinnen und Schüler verstehen das Koordinatensystem des Bildschirms (Ursprung oben links, y nach unten), positionieren Texte gezielt und nutzen die vier Schriftgrößen und die Farbkonstanten.
Dauer	45 Minuten (Aufgabe 1–3 Pflicht, 4–5 Differenzierung)
Voraussetzungen	AB 01 (Editor, Testen, Fehlermeldungen)
Material	PBC+ Studio. Lösungsdateien auf Anfrage. Kopiervorlage Bildschirmraster (L 07) für Aufgabe 2
Buch	Kapitel 2

Stundenverlauf

Zeit	Phase	Inhalt	Sozialform
5 min	Einstieg	Koordinatensystem aus dem Matheunterricht an die Tafel; Frage: „Wo ist beim Bildschirm (0, 0)?“ Auflösung mit der Abbildung auf dem AB: oben links, y nach unten.	Plenum
10 min	Aufgabe 1–2	Verschieben ausprobieren, Vorhersage zeichnen und prüfen	Einzel
15 min	Aufgabe 3	Vier Ecken: Ausprobieren und Nachjustieren	Einzel
10 min	Aufgabe 4–5	Wetterbericht, Plakat	Einzel
5 min	Sicherung	Warum y nach unten? (Textzeilen, Bildaufbau von oben). Gelungene Plakate zeigen.	Plenum

Typische Fehler

Beobachtung	Ursache	Hilfe
Text erscheint nicht, keine Fehlermeldung	y größer als 279 oder x größer als 239: Der Text liegt außerhalb.	Koordinaten mit der Abbildung vergleichen.
Text steht zu weit oben/unten „falsch herum“	y wie im Matheunterricht nach oben gedacht.	Abbildung: y zählt von oben.

Beobachtung	Ursache	Hilfe
Buchstaben in der Ecke angeschnitten	Abgerundete Ecken des Displays.	10–15 Pixel Abstand zum Rand.
`??` statt Umlaut	Die Display-Schriften kennen keine Umlaute (ein „?“ pro Byte).	„ae“, „oe“, „ue“, „ss“ schreiben.
Hintergrundfarbe übermalt den Text	`pbc.clear(farbe)` steht unter den Text-Befehlen.	`clear` immer zuerst: Anweisungen laufen von oben nach unten.
`SyntaxError` bei `pbc.text24("Hallo", 20 120, pbc.WHITE)`	Komma zwischen den Zahlen fehlt.	Argumente durch Kommas trennen.

Differenzierung

- **Schneller fertig:** Buch Kapitel 2, Aufgabe 5 (Stundenplan mit `pbc.text` als Tabelle).
- **Mehr Unterstützung:** Für Aufgabe 3 `ab02_a1.py` als Startpunkt nehmen und die Zeile viermal kopieren lassen (**Strg + C**, **Strg + V**); nur Text und Zahlen ändern.

LEHRERMATERIAL · AB 03

Frage und Antwort

Überblick

Lernziel	Die Schülerinnen und Schüler geben mit `print` Texte und Werte in der Konsole aus, lesen mit `input` Eingaben ein, speichern sie unter einem Namen und wandeln Text mit `int` in eine Zahl bzw. mit `str` in Text um.
Dauer	45 Minuten (Aufgabe 1–3 Pflicht, 4–5 Differenzierung)
Voraussetzungen	AB 01, AB 02 (Editor, Testen, `pbc.text…`, Fehlermeldungen lesen)
Material	PBC+ Studio. Lösungsdateien auf Anfrage
Buch	Kapitel 3

Vorbereitung: `input` funktioniert im Emulator und auf dem Gerät gleich: Die Frage erscheint im Terminal, die Antwort wird dort eingetippt. Wichtig ist, vor dem Tippen **ins Terminal zu klicken**, damit die Eingabe dort ankommt. Die REPL (Buch 3.4) gibt es nur mit angeschlossenem Gerät; dort lässt sich Aufgabe 2 direkt hinter `>>>` ausprobieren. Im Emulator nimmt das Terminal nur Eingaben an, während ein Programm auf `input` wartet.

Stundenverlauf

Zeit	Phase	Inhalt	Sozialform
5 min	Einstieg	Lehrkraft startet `ab03_a1.py` am Beamer, lässt eine Schülerin oder einen Schüler antworten. Frage: „Woher weiß das Programm mein Lieblingsfach?“ → Variable als „Kästchen mit Etikett“.	Plenum
10 min	Aufgabe 1–2	Abtippen, Sonderfälle testen, Vorhersage	Einzel
15 min	Aufgabe 3	Interview: drei Fragen, Ausgabe auf dem Display	Einzel/Partner
10 min	Aufgabe 4–5	`TypeError` lesen, `int` und `str` einsetzen	Einzel
5 min	Sicherung	Tafelbild: `input` → Text → `int(…)` → Zahl → rechnen → `str(…)` → Display. Merksatz: „`input` liefert immer Text.“	Plenum

Typische Fehler

Beobachtung	Ursache	Hilfe
Programm „hängt“, nichts passiert	Es wartet auf eine Eingabe im Terminal.	Frage im Terminal lesen, ins Terminal klicken, antworten, **Enter**.
Getippte Antwort erscheint nicht im Terminal (oder im Editor)	Vor dem Tippen nicht ins Terminal geklickt.	Erst ins Terminal klicken. Versehentliche Änderungen im Editor mit **Strg + Z** rückgängig machen.
`TypeError: unsupported types for __sub__: 'int', 'str'` (bzw. `__add__`)	Mit der Antwort von `input` gerechnet, ohne `int`.	„`input` liefert immer Text“ → `int(input(…))`.
`ValueError: invalid syntax for integer with base 10: '…'`	Bei `int(input(…))` ein Wort oder eine Kommazahl eingegeben.	Eine ganze Zahl eintippen; für Kommazahlen `float` (mit Punkt: `1.5`).
`NameError: name 'pbc' isn't defined`	`import pbc` fehlt (Aufgabe 3, 5); die Frage wird noch gestellt, dann bricht das Programm ab.	`import pbc` als erste Zeile.
Ausgabe zeigt den Namen der Variable statt der Antwort	Variable in Anführungszeichen gesetzt: `"fach"`.	Variablen ohne Anführungszeichen.
`TypeError: can't convert 'int' object to str implicitly` (Aufgabe 5)	Zahl direkt an `pbc.text24` gegeben (`ab03_fehler_display_zahl.py`).	Display-Befehle brauchen Text: `str(zahl)`.

Differenzierung

- **Schneller fertig:** Buch Kapitel 3, Aufgabe 5 (Kinokarte) oder Aufgabe 3 erweitern: Alter abfragen und „In 10 Jahren bist du …“ ausrechnen.
- **Mehr Unterstützung:** Für Aufgabe 3 `ab03_a1.py` als Startpunkt nehmen: Zeile 2 zweimal kopieren und nur Namen und Frage ändern. Die Display-Ausgabe zuerst nur für eine Antwort schreiben.

LEHRERMATERIAL · AB 04

Rechnen mit Variablen

Überblick

Lernziel	Die Schülerinnen und Schüler rechnen mit Variablen und den Operatoren `+ - * / // % **`, verändern Variablen mit `+=` usw., erzeugen Zufallszahlen mit `random.randint` und geben Ergebnisse mit f-Strings (auch `:.2f`) aus.
Dauer	45 Minuten (Aufgabe 1–3 Pflicht, 4–6 Differenzierung)
Voraussetzungen	AB 03 (`print`, `input`, `int`, `float`)
Material	PBC+ Studio. Lösungsdateien auf Anfrage
Buch	Kapitel 4

Stundenverlauf

Zeit	Phase	Inhalt	Sozialform
5 min	Einstieg	„17 Gummibärchen, 5 Kinder – wie viele bekommt jedes, wie viele bleiben übrig?“ Kopfrechnen, dann Notation `17 // 5` und `17 % 5` einführen.	Plenum
10 min	Aufgabe 1	Vorhersagen eintragen, prüfen, Abweichungen besprechen (`/` liefert immer Kommazahl)	Einzel, dann Partner
15 min	Aufgabe 2–3	Klassenfahrt (f-String mit `:.2f`), Gummibärchen selbst schreiben	Einzel
10 min	Aufgabe 4–6	Würfel, Punkte, Kopfrechentrainer	Einzel
5 min	Sicherung	Tabelle der Operatoren an der Tafel ergänzen; `x = x + 1` als „rechts ausrechnen, dann links ablegen“	Plenum

Typische Fehler

Beobachtung	Ursache	Hilfe
`NameError: name 'random' isn't defined`	`import random` fehlt (nur `import pbc` geschrieben). Das Hinweisfenster nennt die fehlende Zeile.	Beide Imports oben: `import pbc` und `import random`.

Beobachtung	Ursache	Hilfe
Im Terminal steht `{wurf}` statt der Zahl	Das `f` vor dem Anführungszeichen fehlt.	`print(f"…")`
`ZeroDivisionError: divide by zero`	Bei Aufgabe 2 oder 3 `0` als Teiler eingegeben.	Andere Eingabe; Abfangen mit `if` kommt in Kapitel 7.
`TypeError: unsupported types for __truediv__: 'str', 'int'` o. Ä.	`int` / `float` um `input` vergessen.	„`input` liefert immer Text“ (AB 03).
Ergebnis wie `161.11111111111111`	Kein Format angegeben.	`{wert:.2f}`
`ValueError: invalid syntax for number`	Kommazahl mit Komma eingegeben (`12,50`).	Punkt statt Komma: `12.50`.
`x + 5` ändert `x` nicht	Ergebnis nicht zugewiesen.	`x = x + 5` oder `x += 5`.

Differenzierung

- **Schneller fertig:** Buch Kapitel 4, Aufgabe 3 (Sekunden umrechnen) und Aufgabe 5 (Pizza-Party, Aufrunden mit `//`).
- **Mehr Unterstützung:** Aufgabe 3 aus `ab04_a2.py` entwickeln lassen: Die Struktur (zwei Eingaben, eine Rechnung, eine Ausgabe) ist dieselbe, nur die Rechnung ändert sich.
- **Erweiterung Aufgabe 6:** Drei Rechenaufgaben nacheinander stellen und die richtigen Ergebnisse am Ende zusammen ausgeben.

LEHRERMATERIAL · AB 05

Ein Bild aus Formen

Überblick

Lernziel	Die Schülerinnen und Schüler zeichnen Bilder aus Linien, Rechtecken und Ellipsen, unterscheiden Umriss und Füllung, wissen, dass bei Rechtecken die linke obere Ecke und bei Ellipsen der Mittelpunkt angegeben wird, berücksichtigen die Zeichenreihenfolge und mischen eigene Farben mit `pbc.rgb`.
Dauer	45 Minuten (Aufgabe 1–4 Pflicht, 5–6 Differenzierung)
Voraussetzungen	AB 02 (Koordinatensystem), AB 04 (Variablen)
Material	PBC+ Studio. Startprogramm `ab05_start.py` (muss für die Schülerinnen und Schüler bereitliegen). Lösungsdateien auf Anfrage
Buch	Kapitel 5

Stundenverlauf

Zeit	Phase	Inhalt	Sozialform
5 min	Einstieg	Bild eines Gesichts oder Roboters zeigen: „Aus welchen Grundformen besteht das?“ Befehlstabelle auf dem AB gemeinsam lesen; Ecke und Mittelpunkt an der Tafel skizzieren.	Plenum
10 min	Aufgabe 1–2	Startprogramm öffnen, Füllung und Radien ausprobieren, Gesicht ergänzen	Einzel
10 min	Aufgabe 3–4	Reihenfolge und Ecke/Mitte: erst vorhersagen, dann testen	Partner
15 min	Aufgabe 5–6	Roboter oder eigenes Bild	Einzel
5 min	Sicherung	Zwei Regeln an die Tafel: „Rechteck: Ecke – Ellipse: Mitte“ und „Was später kommt, liegt oben“. Gelungene Bilder zeigen.	Plenum

Typische Fehler

Beobachtung	Ursache	Hilfe
`TypeError: function missing required`	Farbe vergessen, z. B. `pbc.ellipse(120, 140, 80, 90, fill=True)`. Python zählt	Befehlstabelle: Jeder Zeichenbefehl braucht

Beobachtung	Ursache	Hilfe
`positional argument #4`	die Argumente ab 0; #4 ist das fünfte, die Farbe.	alle Zahlen **und** eine Farbe.
`SyntaxError: invalid syntax`	Komma zwischen zwei Zahlen fehlt: `pbc.fill_rect(70, 50 100, 80, pbc.GRAY)`	Argumente zählen und durch Kommas trennen.
Kreis sitzt verschoben, halb außerhalb einer Fläche	Mittelpunkt wie eine Ecke behandelt (oder umgekehrt).	Aufgabe 4 als Merkhilfe; Mittelpunkt = Ecke + halbe Breite.
Teile des Bildes fehlen, keine Fehlermeldung	Eine spätere große Fläche überdeckt sie.	Reihenfolge von hinten nach vorn: Hintergrund zuerst, Details zuletzt.
Kreis wird oval	`rx` und `ry` verschieden.	Für Kreise beide Radien gleich.
`pbc.rgb(300, 0, 0)` ist nicht „röter“ als `pbc.rgb(255, 0, 0)`	Werte über 255 werden auf 255 begrenzt (beide ergeben 63488).	Jede Zahl von 0 bis 255.
`TypeError: can't convert float to int`	Koordinate mit `/` berechnet, z. B. `pbc.WIDTH / 3`: Das ergibt eine Kommazahl.	Mit `//` rechnen oder `int()` benutzen.

Die Meldungen im Terminal (Emulator):

Farbe vergessen

```
Traceback (most recent call last):
  File "<stdin>", line 4, in <module>
TypeError: function missing required positional argument #4
```

Komma vergessen

```
Traceback (most recent call last):
  File "<stdin>", line 4
SyntaxError: invalid syntax
```

pbc.fill_rect(pbc.WIDTH / 3, 100, 80, 40, pbc.RED)

```
Traceback (most recent call last):
  File "<stdin>", line 4, in <module>
TypeError: can't convert float to int
```

Differenzierung

- **Schneller fertig:** Buch Kapitel 5, Aufgabe 4 (Zielscheibe) und 5 (Balkendiagramm); oder das eigene Bild mit Variablen für die Position bauen, sodass es sich durch Ändern von zwei Zahlen verschieben lässt (wie das Haus im Buch, Ausprobieren).
- **Mehr Unterstützung:** Für Aufgabe 5 die Zeilen für Kopf und Körper vorgeben; Ellipsen erst nur mit `rx = ry` (Kreise) zeichnen lassen. Kopiervorlage Bildschirmraster (L 07) zum Vorzeichnen.

LEHRERMATERIAL · AB 06

Muster mit Schleifen

Überblick

Lernziel	Die Schülerinnen und Schüler wiederholen Anweisungen mit `for` und `range`, berechnen aus der Schleifenvariablen Positionen und Farben, unterscheiden Schleifenkörper und folgende Anweisungen an der Einrückung und zeichnen mit verschachtelten Schleifen Zeilen und Spalten.
Dauer	45 Minuten (Aufgabe 1–3 Pflicht, 4–6 Differenzierung)
Voraussetzungen	AB 04 (Variablen, `%`), AB 05 (Zeichenbefehle)
Material	PBC+ Studio. Startprogramm `ab06_start.py`. Lösungsdateien auf Anfrage
Buch	Kapitel 6 (6.2–6.4)

Stundenverlauf

Zeit	Phase	Inhalt	Sozialform
5 min	Einstieg	Ein Zebramuster ohne Schleife: „Wie viele Zeilen bräuchten wir für 7 Streifen? Für 70?“ Dann das Programm aus dem Kasten: Kopf, Doppelpunkt, Einrückung, `range(7)` = 0 … 6.	Plenum
10 min	Aufgabe 1–2	Ausgabe vorhersagen; Streifen verändern	Einzel
10 min	Aufgabe 3	Treppe: Höhe als Term finden (Tabelle stufe → Höhe an der Tafel)	Partner
15 min	Aufgabe 4–6	Verschachtelte Schleifen, Schachbrett, Tunnel	Einzel
5 min	Sicherung	„Was gehört zur Schleife?“ an Aufgabe 1 klären; Anzahl der Durchläufe bei verschachtelten Schleifen (5 · 6).	Plenum

Typische Fehler

Beobachtung	Ursache	Hilfe
`SyntaxError: invalid syntax` in der `for`-Zeile	Doppelpunkt am Ende des Schleifenkopfs fehlt.	Hinweisfenster lesen; „Kopf endet mit `:`“.
`IndentationError: unindent doesn't match`	Eine Zeile im Körper ist um ein Leerzeichen zu wenig eingerückt (7	Zeilen untereinander vergleichen; Einrückung

Beobachtung	Ursache	Hilfe
`any outer indent level`	statt 8). **Achtung:** MicroPython nennt bei diesem Fehler die Zeile **davor** (siehe unten).	in Vierer-Stufen.
`IndentationError: unexpected indent`	Eine Zeile ist weiter eingerückt als die vorige.	Gleiche Einrückung für alle Zeilen des Körpers.
`SyntaxError: invalid syntax` in der Zeile **unter** dem Schleifenkopf	Körper gar nicht eingerückt (siehe unten).	Nach dem Doppelpunkt **Enter** drücken, der Editor rückt selbst ein; sonst **Tab**.
Ein Streifen/eine Stufe zu wenig	`range(1, 7)` statt `range(7)`, oder Ende vergessen: `ende` ist nie dabei.	Werte von `range` mit `print` ausgeben lassen.
Treppe „schwebt“	Höhe fest 30 statt mit `stufe` berechnet.	Tabelle Stufe → y → Höhe anlegen.

Beispiel zur Einrückung (Zeile 7 hat nur 7 Leerzeichen); gemeldet wird Zeile 6:

ab06_fehler_einrueckung.py

```
1 import pbc
2
3 pbc.clear()
4 for zeile in range(5):
5     for spalte in range(6):
6         x = 30 + spalte * 36
7        y = 50 + zeile * 45
8         pbc.ellipse(x, y, 12, 12, pbc.RED, fill=True)
```

Terminal

```
Traceback (most recent call last):
  File "<stdin>", line 6
IndentationError: unindent doesn't match any outer indent level
```

Körper nicht eingerückt (Zeile 5 steht ganz links):

ab06_fehler_ohne.py

```
1 import pbc
2
3 pbc.clear(pbc.BLUE)
4 for i in range(7):
5 pbc.fill_rect(0, i * 40, 240, 20, pbc.WHITE)
```

Terminal

```
Traceback (most recent call last):
  File "<stdin>", line 5
SyntaxError: invalid syntax
```

Differenzierung

- **Schneller fertig:** Buch Kapitel 6, Aufgabe 3 (Lineal) und 4 (Abendhimmel mit 280 Linien); eigenes Fliesenmuster mit zwei verschachtelten Schleifen und Farben aus `zeile` und `spalte`.
- **Mehr Unterstützung:** Aufgabe 1 mit `print(i)` in Aufgabe 2 und 3 kombinieren, damit die Werte der Schleifenvariablen sichtbar werden; Aufgabe 3 mit Wertetabelle (stufe 0 … 3) auf Papier vorbereiten.

LEHRERMATERIAL · AB 07

Der springende Ball

Überblick

Lernziel	Die Schülerinnen und Schüler animieren eine Figur nach dem Muster „zeichnen – warten – übermalen“, steuern das Tempo mit `time.sleep_ms` und der Schrittweite, zählen mit `range` vorwärts und rückwärts und wiederholen den Ablauf mit `while True`.
Dauer	45 Minuten (Aufgabe 1–3 Pflicht, 4–6 Differenzierung)
Voraussetzungen	AB 06 (`for`, `range`), AB 05 (Ellipsen)
Material	PBC+ Studio. Startprogramm `ab07_start.py`. Lösungsdateien auf Anfrage
Buch	Kapitel 6 (6.1, 6.3, 6.5, 6.6)

Hinweis zur Planung: Ein Ball, der an beliebigen Rändern abprallt, braucht eine Entscheidung (`if`), die erst in Kapitel 7 / AB 08 kommt. Dieses Blatt löst das Springen deshalb **ohne `if`**: Fallen und Steigen sind zwei `for`-Schleifen (vorwärts und rückwärts), die in `while True` endlos wiederholt werden. Nach Kapitel 7 kann man darauf zurückkommen („Jetzt prallt der Ball dort ab, wo er gerade ist“).

Stundenverlauf

Zeit	Phase	Inhalt	Sozialform
5 min	Einstieg	Daumenkino oder Trickfilm: Bewegung = viele Einzelbilder. Am Startprogramm: zeichnen – warten – übermalen.	Plenum
10 min	Aufgabe 1–2	Tempo verändern; Spur ohne Übermalen	Einzel
15 min	Aufgabe 3	Endlos springen: zweite Schleife rückwärts, `while True`, Einrückung	Einzel/Partner
10 min	Aufgabe 4–6	Aufprall, Zähler, wandernder Ball	Einzel
5 min	Sicherung	Warum flackert es? (Jeder Befehl geht sofort zum Display.) Ausblick: Canvas (Kapitel 17), Abprallen mit `if` (Kapitel 7).	Plenum

Typische Fehler

Beobachtung	Ursache	Hilfe
`NameError: name 'time' isn't defined`	`import time` fehlt.	Imports oben: `import pbc`, `import time`.
Roter Balken statt Ball	Übermalen vergessen oder in falscher Farbe.	Übermalen in der Hintergrundfarbe, gleiche Zahlen wie beim Zeichnen.
Ball bleibt unsichtbar, nur kurzes Blitzen	Reihenfolge vertauscht: übermalen – warten – zeichnen.	Muster an die Tafel: zeichnen – **warten** – übermalen.
Ball fällt nur einmal	Zweite Schleife oder `while True` fehlt, oder die `for`-Schleifen sind nicht in `while True` eingerückt.	Einrückung prüfen: `for` 4, Körper 8 Leerzeichen.
Ball steigt nicht, keine Fehlermeldung	`range(230, 30, 5)`: positive Schrittweite beim Rückwärtszählen liefert keine Zahlen.	Rückwärts braucht `-5`.
Zähler zählt in Riesenschritten	`spruenge += 1` steht in einer `for`-Schleife.	Eine Stufe weniger einrücken.
Das Programm hört nicht auf	`while True` ist endlos.	■ **Stop**.

Terminal: import time fehlt

```
Traceback (most recent call last):
  File "<stdin>", line 6, in <module>
NameError: name 'time' isn't defined
```

Differenzierung

- **Schneller fertig:** Zwei Bälle nebeneinander mit verschiedenen Sprunghöhen; Buch Kapitel 6, Aufgabe 5 (Laufschrift); Schatten unter dem Ball, der beim Fallen größer wird (`rx` aus `y` berechnen).
- **Mehr Unterstützung:** `ab07_a3.py` als Lösungsgerüst ausgeben, in dem nur das zweite `range` fehlt; Aufgabe 4–6 weglassen.

LEHRERMATERIAL · AB 08

Zahlen raten

Überblick

Lernziel	Die Schülerinnen und Schüler verzweigen mit `if`, `elif` und `else`, werten Vergleiche und `and` / `or` aus, rücken Blöcke korrekt ein und wiederholen mit `while`, bis eine Bedingung erfüllt ist (`break`, `while versuche < 6`).
Dauer	45 Minuten (Aufgabe 1–4 Pflicht, 5–6 Differenzierung)
Voraussetzungen	AB 03 (`input`, `int`), AB 04 (Variablen, `random.randint`, f-Strings), AB 07 (`while True`)
Material	PBC+ Studio (nur Terminal, kein Display nötig). Startprogramm `ab08_start.py`, Lösungen `ab08_a3.py`, `ab08_a4.py`, `ab08_a5.py`, `ab08_a6.py` auf Anfrage
Buch	Kapitel 7 – Entscheidungen

Stundenverlauf

Zeit	Phase	Inhalt	Sozialform
5 min	Einstieg	Ratespiel an der Tafel: Lehrkraft denkt sich eine Zahl von 0 bis 99, die Klasse rät, die Lehrkraft antwortet nur „größer“/„kleiner“. Frage: Nach welcher Regel antworte ich? → „Wenn …, dann …, sonst …“	Plenum
10 min	Aufgabe 1–2	`if` / `else` testen, Wahrheitstabelle; `=` und `==` besprechen	Einzel
15 min	Aufgabe 3–4	`elif` ergänzen, Schleife mit Zähler und `break`	Einzel/Partner
10 min	Aufgabe 5–6	Zufall und Versuchsgrenze; der PicoBoy rät	Einzel
5 min	Sicherung	Einrückung an einem Tafelbeispiel: Welche Zeile gehört wozu? Strategie „immer die Mitte“ – warum reichen 7 Versuche?	Plenum

Typische Fehler

Beobachtung	Ursache	Hilfe
`SyntaxError: invalid syntax` in der `if`-Zeile	`if tipp = code:` (Zuweisung statt Vergleich) oder	`==` zum Vergleichen; jede `if` / `elif` / `else`-Zeile endet mit

Beobachtung	Ursache	Hilfe
	Doppelpunkt vergessen	`:`. Das Hinweisfenster nennt den Doppelpunkt als erste Ursache.
`SyntaxError: invalid syntax` in der Zeile **nach** dem `if`	Zeile unter `if` nicht eingerückt (MicroPython meldet hier keinen `IndentationError`)	Mit Tab oder 4 Leerzeichen einrücken.
`IndentationError: unexpected indent`	Eine Zeile ist weiter eingerückt als die darüber, ohne neuen Block	Alle Zeilen eines Blocks gleich weit einrücken.
Auch bei 42 erscheint „Falscher Code.“, keine Fehlermeldung	`int()` vergessen: Der Text `"42"` ist nicht gleich der Zahl `42`.	`tipp = int(input(…))`
`TypeError: unsupported types for __lt__: 'str', 'int'`	`int()` vergessen und mit `<` verglichen	wie oben; das Hinweisfenster rät ebenfalls zu `str(...)` bzw. `int(...)`.
Terminal läuft voll, Programm hört nicht auf	In der Schleife wird nicht neu gefragt (`input` steht vor der Schleife)	`input` in die Schleife; mit ■ **Stop** abbrechen.
„ALARM!“ erscheint auch nach dem Treffer	`if tipp != code:` fehlt oder `print` steht eingerückt in der Schleife	Einrückung prüfen: Die Alarmzeile steht nach der Schleife.

Differenzierung

- **Schneller fertig:** Aufgabe 6, dann Buch Kapitel 7, Aufgabe 6 (Kopfrechentrainer). Oder: Tresor mit Display – bei „offen“ wird der Bildschirm grün (`pbc.clear(pbc.GREEN)`), bei Alarm rot.
- **Mehr Unterstützung:** Lösung von Aufgabe 3 (`ab08_a3.py`) als Grundlage für Aufgabe 4 öffnen lassen; nur `while True:` darüber schreiben, alles darunter markieren und mit **Tab** einrücken.

Steuern mit dem Joystick

Überblick

Lernziel	Die Schülerinnen und Schüler fragen mit `pbc.pressed_…()` den Zustand von Joystick und Tasten ab, bewegen eine Figur durch Löschen – Verschieben – Neuzeichnen, begrenzen die Bewegung mit `and` am Bildschirmrand und verknüpfen Bedingungen mit `or` und `not`.
Dauer	45 Minuten (Aufgabe 1–3 Pflicht, 4–6 Differenzierung)
Voraussetzungen	AB 05 (`fill_rect`, Koordinaten), AB 07 (`while True`, `time.sleep_ms`), AB 08 (`if`, Vergleiche)
Material	PBC+ Studio, Emulator oder PicoBoys. Startprogramm `ab09_start.py`, Lösungen `ab09_a1.py`, `ab09_a2.py`, `ab09_a4.py`, `ab09_a5.py`, `ab09_a6.py` auf Anfrage
Buch	Kapitel 8 – Tasten und Joystick (8.1, 8.4), Kapitel 7.3 (`and`, `or`, `not`)

Vorbereitung: Im Emulator steuern die Pfeiltasten den Joystick, die Leertaste ist MITTE, die Tasten A und B sind A und B. Reagiert der Emulator nicht auf die Tastatur, einmal auf das Bild des PicoBoy klicken. Mit echten Geräten macht die Stunde deutlich mehr Spaß.

Stundenverlauf

Zeit	Phase	Inhalt	Sozialform
5 min	Einstieg	Frage: Wie bewegt sich eine Figur in einem Spiel? Daumenkino-Idee: löschen, ein Stück weiter, neu zeichnen – viele Male pro Sekunde.	Plenum
15 min	Aufgabe 1–2	Startprogramm testen, zwei Richtungen ergänzen, Ränder mit `and`	Einzel
5 min	Aufgabe 3	Wahrheitstabelle `and` / `or` / `not`, im Emulator überprüfen (dazu `ab09_a5.py` nutzen)	Partner
15 min	Aufgabe 4–6	Turbo, Alarmfarbe, Ziel	Einzel
5 min	Sicherung	Warum 230 und nicht 240? Skizze an der Tafel: rechte Kante der Figur = x + 10.	Plenum

Typische Fehler

Beobachtung	Ursache	Hilfe
`AttributeError: module 'pbc' has no attribute 'press_right'`	Name falsch geschrieben (`press_right`, `pressed_Right`)	Namen aus dem Blatt oder der Hilfe-Leiste übernehmen.
Figur zieht eine Spur	Löschen (schwarzes Rechteck an der alten Stelle) fehlt oder steht nach dem Verschieben	Reihenfolge: löschen – Position ändern – zeichnen.
Hoch und runter vertauscht	y wie in Mathe nach oben gedacht	y wächst nach unten: hoch heißt `y -= 2`.
Figur läuft ohne Tastendruck los	`if pbc.pressed_right:` ohne Klammern: Die Funktion selbst zählt als wahr.	`()` ergänzen.
Figur bleibt am Rand nicht stehen oder bleibt zu früh stehen	Grenze ohne Breite der Figur gerechnet (240 statt 230) oder `<` / `>` vertauscht	Skizze: linke Kante x, rechte Kante x + 9.
Turbo: Figur verschwindet trotz Randprüfung teilweise	`x < 230` bei Schrittweite 6	`x + schritt <= 230`.
Emulator reagiert nicht auf Pfeiltasten	Fokus liegt woanders	Einmal auf das Bild des PicoBoy klicken.

Differenzierung

- **Schneller fertig:** Das Ziel springt nach dem Erreichen an eine zufällige neue Stelle (`random.randint`), und ein Zähler zählt die erreichten Ziele; oder Buch Kapitel 8, Aufgabe 3 (Zaubertafel mit Farbwechsel).
- **Mehr Unterstützung:** `ab09_a1.py` als Startpunkt geben, sodass in Aufgabe 2 nur die Grenzen ergänzt werden müssen. Für die Grenzen ein Bildschirmraster (L 07) mit eingezeichneter Figur bereitlegen.

LEHRERMATERIAL · AB 10

Der Reaktionstest

Überblick

Lernziel	Die Schülerinnen und Schüler messen Zeitspannen mit `time.ticks_ms()` und `time.ticks_diff()`, warten mit einer `while not …`-Schleife auf eine bestimmte Taste, nutzen, dass `was_pressed_…()` einen Druck speichert, und bewerten Ergebnisse mit `if` / `elif` / `else`.
Dauer	45 Minuten (Aufgabe 1–3 Pflicht, 4–5 Differenzierung)
Voraussetzungen	AB 04 (`random.randint`, f-Strings), AB 08 (`if` / `elif` / `else`, `while`, `break`), AB 09 (`pressed_…`)
Material	PBC+ Studio, Emulator oder PicoBoys. Startprogramm `ab10_start.py`, Lösungen `ab10_a2.py` (Test „zu früh“), `ab10_a3.py`, `ab10_a4.py`, `ab10_a5.py` auf Anfrage
Buch	Kapitel 8 – Tasten und Joystick (8.2, 8.3, 8.5, Ausprobieren)

Vorbereitung: Im Emulator ist MITTE die Leertaste. Reagiert er nicht, einmal auf das Bild des PicoBoy klicken. Reaktionszeiten im Emulator hängen auch von Tastatur und Browser ab und sind meist etwas länger als am Gerät. Für das Duell (Aufgabe 5) eignet sich ein echtes Gerät besser; im Emulator teilen sich zwei Personen eine Tastatur (A und B).

Stundenverlauf

Zeit	Phase	Inhalt	Sozialform
5 min	Einstieg	Lineal-Fangtest: Eine Person lässt ein Lineal fallen, die andere fängt es. Wie könnte der PicoBoy die Reaktionszeit messen? → Startzeit merken, Endzeit merken, Differenz.	Partner/Plenum
10 min	Aufgabe 1–2	Messen, Tabelle füllen, Programm lesen, „zu früh“ ausprobieren	Einzel
15 min	Aufgabe 3	Bewertung mit `if` / `elif` / `else`	Einzel
10 min	Aufgabe 4–5	Schummeln erkennen, Duell	Einzel/Partner
5 min	Sicherung	Unterschied `pressed_…` / `was_pressed_…` an Aufgabe 2.3 erklären; Bestzeiten der Klasse sammeln	Plenum

Typische Fehler

Beobachtung	Ursache	Hilfe
Zeit ist negativ	`time.ticks_diff(start, ende)` – Reihenfolge vertauscht	Erst der spätere Zeitpunkt: `ticks_diff(ende, start)`.
Zeit ist immer etwa 0 ms	Zu früh gedrückt, oder `ende = time.ticks_ms()` steht vor der Warteschleife	Reihenfolge prüfen: Start merken – warten – Ende merken.
Programm wartet nicht auf die Taste	`while not pbc.was_pressed_center:` ohne Klammern: Die Funktion selbst zählt als wahr, `not` macht daraus `False`.	`()` ergänzen.
Programm reagiert nicht auf die Leertaste	Es wird `was_pressed_a()` abgefragt, gedrückt wird aber MITTE (oder umgekehrt)	Abfrage und gedrückte Taste vergleichen.
„Zu frueh!“ und Kreis erscheinen beide	Kreis und Messung stehen nicht eingerückt im `else`-Block	Einrückung prüfen.
Duell: Wer früh drückt, gewinnt sofort	Gemerkte Drücke nicht abgeholt	Vor dem Kreis einmal `pbc.was_pressed_a()` und `pbc.was_pressed_b()` aufrufen.

Differenzierung

- **Schneller fertig:** Fünf Runden mit `for`, Bestzeit merken (Buch Kapitel 8, Ausprobieren), oder Buch Kapitel 8, Aufgabe 5 (Taktgefühl).
- **Mehr Unterstützung:** Für Aufgabe 3 die Lösung von Aufgabe 1 eines Nachbarn nutzen und nur die Bewertung schreiben; die Tabelle auf dem Blatt gibt die Grenzen vor. Aufgabe 4 als Lückentext vorgeben (`if …:` / `else:` an der richtigen Stelle).

LEHRERMATERIAL · AB 11

Musik und Licht

Überblick

Lernziel	Die Schülerinnen und Schüler schalten die drei Status-LEDs, mischen Farben mit der RGB-LED (additive Farbmischung), spielen Töne mit `pbc.tone` und programmieren eine kurze Melodie. Sie verstehen, dass `pbc.tone` nicht wartet.
Dauer	45 Minuten (Aufgabe 1–3 Pflicht, 4–6 Differenzierung)
Voraussetzungen	AB 06/07 (`for`, `time.sleep`), AB 09/10 (Tasten, für Aufgabe 6)
Material	PBC+ Studio. Kopfhörer sehr empfohlen (der Emulator spielt die Töne über den Rechner). Startprogramm `ab11_start.py`, Lösungsdateien auf Anfrage
Buch	Kapitel 9

Vorbereitung: Im Emulator erscheinen die drei LEDs am linken Rand des Geräts, die RGB-LED im Pluszeichen des Schriftzugs „PBC+“. Der Ton kommt aus den Lautsprechern des Rechners; ohne Kopfhörer wird es in einem Raum mit 30 Rechnern laut. Mit echten Geräten hört man den kleinen Lautsprecher des PicoBoy.

Stundenverlauf

Zeit	Phase	Inhalt	Sozialform
5 min	Einstieg	Ampel an der Tafel: Welche Phasen gibt es in welcher Reihenfolge? (Rot, Rot-Gelb, Grün, Gelb)	Plenum
10 min	Aufgabe 1	Ampel vervollständigen, Schleife	Einzel
5 min	Aufgabe 2	Farben raten und prüfen; additive Farbmischung kurz sichern (Rot + Grün = Gelb)	Partner
10 min	Aufgabe 3	„Was hörst du?“, Erklärung gemeinsam: `pbc.tone` startet den Ton nur, das Programm läuft sofort weiter.	Einzel, dann Plenum
12 min	Aufgabe 4–6	Melodie, Licht zur Melodie, Klingel	Einzel
3 min	Sicherung	Muster „Ton – Warten“ an die Tafel; ein, zwei Melodien vorspielen lassen	Plenum

Typische Fehler

Beobachtung	Ursache	Hilfe
Nur ein Ton zu hören, obwohl mehrere programmiert sind	Kein `time.sleep_ms` nach `pbc.tone(f, ms)`; der nächste Ton ersetzt den vorigen sofort.	Muster „Ton – Warten“; Aufgabe 3 besprechen.
Der Ton hört nicht auf, auch nach dem Programmende	`pbc.tone(f)` ohne Dauer ist ein Dauerton.	`pbc.tone_off()` ergänzen oder `pbc.tone(f, ms)` benutzen; ■ **Stop** schaltet ihn ab.
Zwei gleiche Noten klingen wie eine lange	Ton genauso lang wie die Wartezeit, keine Lücke.	Ton 50 ms kürzer als die Wartezeit.
LEDs leuchten nach dem Programm weiter	LEDs bleiben an, bis man sie ausschaltet; auf dem Gerät sogar nach ■ **Stop**.	`pbc.leds_off()` und `pbc.rgb_led_off()` ans Ende.
In der Ampelschleife leuchten zwei Farben gleichzeitig	Eine LED wurde in der Schleife nicht wieder ausgeschaltet.	Jede Phase: erst aus, dann an. Ablauf als Tabelle aufschreiben lassen.
`NameError: name 'time' isn't defined`	`import time` fehlt.	Beide Imports oben hinschreiben.
`TypeError: rgb_led() takes 3 positional arguments but 1 were given`	`pbc.rgb_led("rot")` oder `pbc.rgb_led(pbc.RED)`; die RGB-LED braucht drei Zahlen.	`pbc.rgb_led(255, 0, 0)`

Differenzierung

- **Schneller fertig:** Buch Kapitel 9, „Ausprobieren: Das Mini-Klavier“ und Aufgabe 5 (Metronom).
- **Mehr Unterstützung:** Für Aufgabe 4 die ersten vier Noten (C D E C) als Vorlage an die Tafel schreiben; dann wird nur noch kopiert und angepasst.
- **Mit Geräten:** Den Lautsprecher des PicoBoy hören; die Ampel auf dem Gerät speichern (**Dateien → Auf den PBC+...**) und über das Menü starten.

LEHRERMATERIAL · AB 12

Die Geheimschrift

Überblick

Lernziel	Die Schülerinnen und Schüler greifen mit Index und Slicing auf Teile von Strings zu, durchlaufen Texte mit `for`, rechnen mit `ord` und `chr` zwischen Zeichen und Zahlen um und setzen die Caesar-Verschlüsselung (mit `% 26`) um.
Dauer	45 Minuten (Aufgabe 1–4 Pflicht, 5–6 Differenzierung)
Voraussetzungen	AB 03 (`input`), AB 04 (`%`), AB 06 (`for`), AB 08 (`if`)
Material	PBC+ Studio. Startprogramm `ab12_start.py`, Lösungsdateien auf Anfrage. Optional: Caesar-Scheibe aus Pappe (zwei Kreise mit Alphabet) für den Einstieg
Buch	Kapitel 10

Stundenverlauf

Zeit	Phase	Inhalt	Sozialform
5 min	Einstieg	Geheimnachricht `WUHIIHQ XP GUHL` an der Tafel. Wer knackt sie? Hinweis: Jeder Buchstabe ist um gleich viele Stellen verschoben.	Plenum
10 min	Aufgabe 1–2	Index ab 0, negativer Index, Slicing; Zeichen zählen	Einzel
5 min	Aufgabe 3	`ord` / `chr`; sichern: Buchstaben haben Nummern, mit denen man rechnen kann	Partner
15 min	Aufgabe 4	Caesar-Programm vervollständigen; `% 26` an Z → C erklären	Einzel
7 min	Aufgabe 5–6	Entschlüsseln, Schlüssel einstellen	Einzel/Partner
3 min	Sicherung	Warum ist Caesar unsicher? (nur 25 Schlüssel – ein Programm probiert alle in einem Augenblick)	Plenum

Typische Fehler

Beobachtung	Ursache	Hilfe
`IndexError: string index out of range`	Index zu groß, meist um eins verrutscht (`wort[9]` bei 9 Zeichen).	Indizes 0 bis Länge − 1 unter das Wort schreiben.
`TypeError: 'str' object doesn't support item assignment`	`wort[0] = "B"`: Strings kann man nicht verändern.	Neuen Text bauen: `wort = "B" + wort[1:]`.
Aus X, Y, Z werden `[`, `\`, `]`	`% 26` vergessen.	Alphabet als Kreis an die Tafel; Rest bei Division.
`TypeError: unsupported types for __add__: 'str', 'int'`	Mit dem Buchstaben statt mit der Nummer gerechnet: `zeichen + 3`.	Erst `ord`, dann rechnen, dann `chr`.
`TypeError: can't convert str to int`	`chr("A")` statt `ord("A")`.	`ord`: Zeichen → Zahl, `chr`: Zahl → Zeichen.
Kleinbuchstaben werden nicht verschlüsselt	Die Bedingung prüft nur `"A"` bis `"Z"`.	Text vorher mit `text = text.upper()` großschreiben.
Ergebnis ist leer oder nur der letzte Buchstabe	`geheim = ""` steht in der Schleife oder `geheim = …` statt `geheim += …`.	Mit `print` in der Schleife zeigen, was in `geheim` steht.

Differenzierung

- **Schneller fertig:** Buch Kapitel 10, Aufgabe 6 (alle 25 Schlüssel durchprobieren und den Code knacken) oder Aufgabe 4 (Melodie als Text).
- **Mehr Unterstützung:** Aufgabe 4 erst auf Papier mit einer Caesar-Scheibe durchspielen; die Zeile 8 gemeinsam für den Buchstaben A, dann für Z ausrechnen.
- **Fächerverbindend:** Häufigkeitsanalyse (Aufgabe 2 für alle Buchstaben): Im Deutschen ist E der häufigste Buchstabe. So lässt sich der Schlüssel auch ohne Durchprobieren finden.

Ordnung mit Funktionen

Überblick

Lernziel	Die Schülerinnen und Schüler erkennen Wiederholungen in einem Programm, fassen sie in einer Funktion mit Parametern zusammen, rufen diese mit verschiedenen Argumenten auf und nutzen Standardwerte.
Dauer	45 Minuten (Aufgabe 1–3 Pflicht, 4–6 Differenzierung)
Voraussetzungen	AB 05 (Formen und Farben), AB 06/07 (Schleifen, `time.sleep`), AB 08 (`if`, `while`); für Aufgabe 6 AB 11 (LEDs)
Material	PBC+ Studio. Startprogramm `ab13_start.py` (vorher auf die Rechner kopieren oder zum Download bereitstellen). Lösungsdateien auf Anfrage
Buch	Kapitel 11

Stundenverlauf

Zeit	Phase	Inhalt	Sozialform
5 min	Einstieg	`ab13_start.py` am Beamer zeigen. Frage: „Was müsstet ihr ändern, damit alle Ampeln 20 Pixel weiter oben stehen?“ – zwölf Zahlen an zwölf Stellen. Das Ziel der Stunde: eine Stelle.	Plenum
5 min	Erarbeitung	„So funktioniert’s“: `def`, Einrückung, Aufruf, Parameter. Betonen: Die Definition allein tut nichts.	Plenum
20 min	Aufgabe 1–3	Unterschiede finden, Ausgabe vorhersagen, Funktion `ampel` schreiben	Einzel/Partner
10 min	Aufgabe 4–6	Standardwerte, Ampelphasen, LEDs	Einzel
5 min	Sicherung	Vorher/Nachher vergleichen (21 zu 15 Zeilen, vor allem: Änderungen an *einer* Stelle). Begriffe Parameter/Argument an der Tafel.	Plenum

Typische Fehler

Beobachtung	Ursache	Hilfe
Programm läuft, aber nichts wird gezeichnet	Funktion definiert, aber nie aufgerufen (oder Aufruf ohne	Die Definition allein tut nichts; Aufrufe unter die Funktion

Beobachtung	Ursache	Hilfe
	`()`	schreiben.
`NameError: name 'ampel' isn't defined`	Aufruf steht über dem `def`	Erst alle `def`, dann das Hauptprogramm.
`NameError: name 'x' isn't defined` in einer Zeile, die zur Funktion gehören sollte	Zeile nicht eingerückt: Sie gehört nicht mehr zur Funktion und läuft sofort, wo es kein `x` gibt.	Alle Zeilen der Funktion mit 4 Leerzeichen einrücken.
`TypeError: ampel() takes 4 positional arguments but 2 were given`	Beim Aufruf fehlen Argumente	Anzahl der Parameter und Argumente vergleichen; ab Aufgabe 4 helfen Standardwerte.
`SyntaxError: invalid syntax` in der `def`-Zeile	Doppelpunkt am Ende fehlt	`def ampel(x, oben, mitte, unten):`
`SyntaxError: non-default argument follows default argument`	In Aufgabe 4 nur einem Teil der Farben einen Standardwert gegeben, `x` dahinter oder dazwischen	Parameter mit Standardwert ans Ende.
Alle Ampeln an derselben Stelle	In der Funktion feste Zahlen (45, 120 …) statt `x + 25`	Fragen: „Was ändert sich von Ampel zu Ampel?“ – das muss aus `x` berechnet werden.

Differenzierung

- **Schneller fertig:** Buch Kapitel 11, Aufgabe 5 (Flaggen) oder „Ausprobieren: Das Gesicht in drei Größen“ mit eigener Erweiterung.
- **Mehr Unterstützung:** Aufgabe 3 gemeinsam beginnen: Ampel 1 aus dem Startprogramm in die Funktion kopieren und Zahl für Zahl fragen „Hängt das von der Ampel ab?“. Bei Aufgabe 5 die Funktion `phase` vorgeben und nur die Schleife schreiben lassen.

LEHRERMATERIAL · AB 14

Werkzeuge mit Rückgabewert

Überblick

Lernziel	Die Schülerinnen und Schüler schreiben Funktionen, die mit `return` einen Wert (auch `True` / `False` oder zwei Werte) zurückgeben, unterscheiden `return` und `print` und testen ihre Funktionen mit `print` und `assert` an typischen Fällen und Grenzfällen.
Dauer	45 Minuten (Aufgabe 1–3 Pflicht, 4–6 Differenzierung)
Voraussetzungen	AB 13 (Funktionen mit Parametern), AB 08 (`if`, Vergleiche), AB 04 (`%`)
Material	PBC+ Studio. Startprogramm `ab14_start.py`. Lösungsdateien auf Anfrage
Buch	Kapitel 12

Stundenverlauf

Zeit	Phase	Inhalt	Sozialform
5 min	Einstieg	Taschenrechner-Vergleich: Man tippt etwas ein, es kommt ein Ergebnis zurück. `len("Hallo")` liefert 5. Ziel: eigene Funktionen, die etwas zurückliefern.	Plenum
5 min	Erarbeitung	„So funktioniert's“: `return`, Rückgabewert, Test mit Erwartung; `assert` an einem Beispiel vorführen, das scheitert (`ab14_fehler_test.py`).	Plenum
20 min	Aufgabe 1–3	Teilbarkeit, `return` vs. `print`, Maximum mit Tests	Einzel/Partner
10 min	Aufgabe 4–6	Abstand, Note, zwei Rückgabewerte	Einzel
5 min	Sicherung	Aufgabe 2 besprechen: Wann `print`, wann `return`? Merksatz: „Rechnen mit `return`, anzeigen im Hauptprogramm.“ Welche Grenzfälle wurden getestet?	Plenum

Typische Fehler

Beobachtung	Ursache	Hilfe
`TypeError: unsupported types for __add__:`	Funktion gibt mit `print` aus statt mit `return`	Aufgabe 2 als Erklärung heranziehen; `print` in der Funktion durch `return`

Beobachtung	Ursache	Hilfe
`'NoneType', 'int'`	zurück; das Ergebnis ist `None`.	ersetzen.
Im Terminal steht `None`	Ergebnis einer Funktion ohne `return` ausgegeben	Hat jeder Weg durch die Funktion ein `return`?
`AssertionError:` ohne weiteren Text	Ein Test ist gescheitert. Die Zeilennummer zeigt, welcher. PBC+ Studio öffnet dafür kein Hinweisfenster.	Den Test einzeln mit `print(abstand(3, 10))` ausgeben und mit der Erwartung vergleichen. Optional `assert …, "Text"` für eine eigene Meldung.
Tests bestehen, aber Ergebnis falsch	Nur einfache Fälle getestet (z. B. Abstand nur mit a > b)	Nach Grenzfällen fragen: gleich, vertauscht, negativ, 0.
Note an der Grenze falsch (84 % ergibt 1)	`>` und `>=` verwechselt oder Reihenfolge der `if` falsch	Grenztests 85/84 ergänzen; von der besten Note abwärts prüfen.
Code nach `return` läuft nicht	`return` beendet die Funktion sofort.	Ausgaben vor das `return` setzen oder ins Hauptprogramm verlagern.
`ValueError: need more than 2 values to unpack`	Bei Aufgabe 6 drei Variablen links vom `=`, aber zwei Rückgabewerte	Gleich viele Variablen wie Werte.

Differenzierung

- **Schneller fertig:** Buch Kapitel 12, Aufgabe 5 (Quersumme: erst die Tests, dann die Funktion) und Aufgabe 6 (Pasch).
- **Mehr Unterstützung:** Für Aufgabe 3 das Gerüst vorgeben (`if a > b:` / `return a` / `return b`). Aufgabe 4 zuerst mit `print`-Tests statt `assert` bearbeiten lassen.

Listen 1: Messwerte und Bestenliste

Überblick

Lernziel	Die Schülerinnen und Schüler legen Listen an, greifen über den Index (auch negativ) auf Elemente zu, füllen eine leere Liste mit `append`, werten Listen mit `len`, `sum`, `min`, `max` aus und sortieren mit `sorted` (auf- und absteigend).
Dauer	45 Minuten (Aufgabe 1–3 Pflicht, 4–6 Differenzierung)
Voraussetzungen	AB 03 (`input`, `int`), AB 06 (`for`, `range`), AB 08 (`while`, `break`), f-Strings
Material	PBC+ Studio. Lösungsdateien auf Anfrage. Für Aufgabe 3 eine Uhr mit Sekundenzeiger (Puls 15 s zählen, mal 4)
Buch	Kapitel 13

Stundenverlauf

Zeit	Phase	Inhalt	Sozialform
5 min	Einstieg	Tafel: sieben Variablen `mo = 14`, `di = 17` … „Und wie rechne ich den Durchschnitt eines ganzen Jahres?“ → eine Liste mit Index. Index ab 0 als „Hausnummern“ unter die Werte schreiben.	Plenum
10 min	Aufgabe 1–2	Abtippen, Vorhersagen notieren, im Emulator prüfen	Einzel/Partner
15 min	Aufgabe 3	Puls messen (Partner), Programm schreiben	Partner, dann Einzel
10 min	Aufgabe 4–6	Diagramm, Bestenliste	Einzel
5 min	Sicherung	`sorted` liefert eine neue Liste; `append` verändert die Liste. Warum ist bei Laufzeiten aufsteigend „gut“?	Plenum

Typische Fehler

Beobachtung	Ursache	Hilfe
`IndexError: list index out of range`	Index zu groß, z. B. `temperaturen[7]` bei 7 Elementen.	Index läuft von 0 bis `len(liste) - 1`. Indizes unter die Werte schreiben.

Beobachtung	Ursache	Hilfe
`max` liefert in Aufgabe 3 „88“ statt 120, danach `TypeError: unsupported types for __add__: 'int', 'str'`	`input` liefert Text; `int(...)` vergessen. Texte werden Zeichen für Zeichen verglichen („8“ > „1“), `sum` kann Texte nicht addieren.	`puls.append(int(input("Puls: ")))`
`AttributeError: 'NoneType' object has no attribute 'append'`	`puls = puls.append(72)`: `append` liefert nichts (`None`), die Liste ist danach weg.	Nur `puls.append(72)` schreiben, ohne `puls =`.
Liste ist nach `sorted(zeiten)` unverändert	`sorted` liefert eine neue Liste, die nicht gespeichert wurde.	`zeiten = sorted(zeiten)` oder `zeiten.sort()`
Bestenliste absteigend statt aufsteigend	`reverse=True` übernommen	Bei Zeiten ist kleiner besser.

Differenzierung

- **Schneller fertig:** Buch Kapitel 13, Aufgabe 6 (Notenspiegel als Balkendiagramm).
- **Mehr Unterstützung:** Für Aufgabe 3 das Gerüst vorgeben: `puls = []`, `for i in range(5):` und die erste `print`-Zeile; nur `append` und die Auswertung ergänzen lassen. Aufgabe 2 gemeinsam Zeile für Zeile an der Tafel mitschreiben (Liste nach jeder Zeile notieren).

Listen 2: Merk dir die Folge

Überblick

Lernziel	Die Schülerinnen und Schüler speichern eine wachsende Folge in einer Liste (`append`), durchlaufen sie mit `for`, greifen über den Index zu, übergeben die Liste an Funktionen und schreiben eine Funktion, die `True` oder `False` liefert. Ergebnis ist ein vollständiges Gedächtnisspiel.
Dauer	2 × 45 Minuten (Stunde 1: Aufgabe 1–4, Stunde 2: Aufgabe 5–7; Pflicht sind 1–3 und 5)
Voraussetzungen	AB 09/10 (`was_pressed_…`), AB 13/14 (Funktionen, `return`), AB 15 (Listen)
Material	PBC+ Studio, Startprogramm `ab16_start.py` (vorher an die Klasse verteilen). Lösungsdateien auf Anfrage. Kopfhörer sind angenehm, aber nicht nötig.
Buch	Kapitel 13 (13.1–13.3, 13.5), Kapitel 12 (Funktionen mit `True` / `False`)

Vorbereitung: `ab16_start.py` auf die Rechner oder in den Kursordner legen. Das Startprogramm enthält Zeichen-, Ton- und Tastenfunktionen, damit sich die Stunde auf die Liste konzentriert. Ein kurzer Blick auf `lies_taste` lohnt sich: Die Funktion wartet in einer Schleife und **liefert** mit `return` die Nummer des gedrückten Feldes.

Stundenverlauf

Zeit	Phase	Inhalt	Sozialform
5 min	Einstieg	Spiel „Merk dir die Folge“ an der Tafel: Lehrkraft zeigt eine Folge von Farben, die Klasse wiederholt; jede Runde kommt eine Farbe dazu. Frage: Was muss sich das Programm merken? → eine Liste, die wächst.	Plenum
10 min	Aufgabe 1	Startprogramm öffnen, testen, Folge verlängern, `IndexError` provozieren	Einzel
15 min	Aufgabe 2	Schleife und Funktion `spiele_folge`	Einzel
10 min	Aufgabe 3–4	Zufällige Folge, Code lesen	Einzel/Partner
5 min	Sicherung 1	Warum ist die Schleife besser als fünf Zeilen? (beliebige Länge)	Plenum

Zeit	Phase	Inhalt	Sozialform
5 min	Einstieg 2	Wiederholung: Liste wächst mit `append`, Funktion `lies_taste` liefert eine Zahl	Plenum
15 min	Aufgabe 5	`spieler_ist_richtig`, Frage zu `return True`	Partner
20 min	Aufgabe 6–7	Spielschleife, Erweiterungen	Einzel
5 min	Sicherung 2	Rekorde vergleichen; wie viele Runden schafft die Klasse?	Plenum

Typische Fehler

Beobachtung	Ursache	Hilfe
`IndexError: list index out of range`	Index zu groß, z. B. `folge[5]` bei 5 Elementen, oder `range(1, len(folge) + 1)`	Indizes 0 bis `len(folge) - 1`; besser `for nr in folge:`
`NameError: name 'spiele_folge' isn't defined`	Funktion steht unter dem Aufruf.	Funktionen über Zeile 45 einfügen.
Spiel endet immer sofort mit „Falsch“	`if taste == nr: return False` (Bedingung umgedreht)	Bedingung laut vorlesen: „Wenn die Taste **nicht** stimmt …“
Spiel ist schon nach dem ersten richtigen Feld „Richtig!“	`return True` in der Schleife eingerückt	Einrückung prüfen: `return True` auf Höhe von `for`.
Folge wird jede Runde ganz neu ausgelost	`folge = []` steht in der Spielschleife	Die leere Liste gehört **vor** die Schleife.
Ein Druck während der Vorführung zählt als erste Eingabe	`was_pressed_…` merkt sich jeden Druck, bis er abgefragt wird.	Erklären: erst drücken, wenn „Du bist dran!“ erscheint.

Differenzierung

- **Schneller fertig:** Aufgabe 7 vollständig; außerdem: Statt Zufall vier verschiedene Töne als „Melodie“ nachspielen lassen.
- **Mehr Unterstützung:** In Stunde 2 `ab16_a5.py` als Ausgangspunkt ausgeben; dann bleibt nur die Spielschleife (Aufgabe 6). Die Schleife vorher als Ablauf an die Tafel schreiben: anhängen → zeigen → nachspielen → prüfen.

LEHRERMATERIAL · AB 17

Listen 3: Sternenregen

Überblick

Lernziel	Die Schülerinnen und Schüler bewegen viele Objekte mit parallelen Listen und mit Listen in Listen, erzeugen Listen mit einer List Comprehension und entfernen Elemente sicher, indem sie über eine Kopie (`liste[:]`) laufen.
Dauer	45 Minuten (Aufgabe 1–2 Pflicht, 3–5 Differenzierung)
Voraussetzungen	AB 07 (Animation mit Endlosschleife), AB 09 (Joystick), AB 15/16 (Listen)
Material	PBC+ Studio, Startprogramm `ab17_start.py`. Lösungsdateien auf Anfrage
Buch	Kapitel 14

Stundenverlauf

Zeit	Phase	Inhalt	Sozialform
5 min	Einstieg	Ein fallender Stern mit `x`, `y` – „Und jetzt 30?“ Tafel: Tabelle mit den Spalten x, y; jede Zeile ein Stern → Spalte = Liste, Zeile = Index.	Plenum
10 min	Aufgabe 1	Startprogramm verstehen, vierter Stern	Einzel
10 min	Aufgabe 2	List Comprehension, Tempo, Farben	Einzel
15 min	Aufgabe 3–5	Listen in Listen, Entfernen, Ausweichspiel	Einzel/Partner
5 min	Sicherung	`ab17_falsch.py` an der Tafel Schritt für Schritt durchspielen: Warum bleibt 290 übrig?	Plenum

Typische Fehler

Beobachtung	Ursache	Hilfe
`IndexError: list index out of range` in der Zeile mit `xs[i], ys[i]`	Parallele Listen sind unterschiedlich lang (z. B. vierter Wert nur in `xs`).	Jede Liste gleich lang halten, oder gleich Listen in Listen benutzen.

Beobachtung	Ursache	Hilfe
Nicht alle gelandeten Sterne verschwinden, ohne Fehlermeldung	`for stern in sterne:` und `sterne.remove(stern)` in derselben Schleife	Über die Kopie laufen: `for stern in sterne[:]:`
Sterne hinterlassen Spuren	Alte Stelle wird nach dem Verschieben übermalt statt davor.	Reihenfolge: löschen – bewegen – zeichnen.
Alle Sterne fallen gleich schnell	`ys[i] += 3` statt `ys[i] += tempo[i]`	Für jede Eigenschaft eine Liste, gleicher Index.
`SyntaxError` bei der List Comprehension	Klammer vergessen, etwa `[random.randint(0, 236) for i in range(30)`	Klammern paarweise zählen.
Text „Sterne: 9“ zeigt Reste („90“)	Kürzere Zahl überschreibt die längere nur teilweise.	`bg=pbc.BLACK` und Leerzeichen anhängen oder vorher Rechteck löschen.

Differenzierung

- **Schneller fertig:** Buch Kapitel 14, Aufgabe 6 (Meteore im Sternenregen) oder Ausprobieren „Der Sternenregen“ (Sterne mit einer Schale fangen).
- **Mehr Unterstützung:** Aufgabe 2 mit festen Listen statt List Comprehension lösen lassen (zehn Werte von Hand), dann gemeinsam auf die Kurzschreibweise umstellen. Für Aufgabe 4 `ab17_a3.py` als Ausgangspunkt geben.

LEHRERMATERIAL · AB 18

Der Highscore bleibt

Überblick

Lernziel	Die Schülerinnen und Schüler speichern Werte mit `open`, `write` und `with` in einer Datei, lesen sie mit `read` / `readline` wieder ein, wandeln dabei zwischen Zahl und Text um (`str`, `int`) und prüfen mit `os.listdir()`, ob eine Datei existiert. Sie unterscheiden die Modi `"r"`, `"w"` und `"a"`.
Dauer	45 Minuten (Aufgabe 1–3 Pflicht, 4–5 Differenzierung)
Voraussetzungen	AB 10 (Reaktionstest, `was_pressed_a`, `ticks_ms`), AB 14 (Funktionen mit `return`), AB 15 (Listen, `sum`, `len`)
Material	PBC+ Studio; Startprogramm `ab18_start.py` zum Öffnen. Lösungsdateien auf Anfrage. Optional PicoBoys, um zu zeigen, dass der Wert auch das Ausschalten übersteht
Buch	Kapitel 15

Vorbereitung: `ab18_start.py` bereitstellen. Wichtig für den Emulator: Dateien bleiben nur erhalten, bis die Seite neu geladen wird. Das ist in Aufgabe 1 gewollt (der Fehler zeigt, dass die Datei wirklich „weg“ ist), kann aber verwirren, wenn jemand zwischendurch neu lädt. Mit Geräten lässt sich am Stundenende zeigen: ausschalten, einschalten, Programm starten – die Bestzeit ist noch da.

Stundenverlauf

Zeit	Phase	Inhalt	Sozialform
5 min	Einstieg	Frage: „Warum zeigt euer Reaktionstest aus AB 10 nach jedem Start wieder keine Bestzeit?“ Variablen leben nur, solange das Programm läuft. Idee: auf einen Zettel schreiben – die Datei.	Plenum
10 min	Aufgabe 1–2	Schreiben und Lesen in zwei Programmen; Modi `"w"` und `"a"` vorhersagen	Einzel
20 min	Aufgabe 3	Bestzeit laden und speichern, mehrmals testen	Einzel/Partner
5 min	Aufgabe 4–5	Name speichern, alle Versuche sammeln	Einzel

Zeit	Phase	Inhalt	Sozialform
5 min	Sicherung	Muster an der Tafel: nachsehen (`os.listdir`) → lesen oder Startwert → ändern → speichern. Warum nur bei neuer Bestzeit schreiben? (Speicher schonen)	Plenum

Typische Fehler

Beobachtung	Ursache	Hilfe
`OSError: [Errno 2] ENOENT`, Hinweis „Datei nicht gefunden“	Datei noch nie geschrieben, Name vertippt oder (Emulator) Seite neu geladen.	Erst schreiben, dann lesen; vorher mit `"name" in os.listdir()` prüfen.
`TypeError: object with buffer protocol required`	`datei.write(zeit)` mit einer Zahl.	`datei.write(str(zeit))`.
`TypeError: unsupported types for __lt__: 'int', 'str'`	Gelesener Text wird mit einer Zahl verglichen (`zeit < bestzeit`).	Beim Lesen `int(datei.read())`.
`ValueError: invalid syntax for integer with base 10: ''`	Datei ist leer, z. B. weil sie zum Lesen mit `"w"` geöffnet und dabei gelöscht wurde.	Zum Lesen `"r"`; leere Datei löschen oder neu schreiben.
Es gibt nie eine neue Bestzeit.	Startwert 0 statt 9999 bei „kleiner ist besser“.	Überlegen: Welcher Wert ist sicher schlechter als jeder echte Versuch?
`NameError: name 'os' isn't defined`	`import os` fehlt.	Import oben ergänzen.
Bestzeit ist nach dem Neuladen weg.	Emulator: Dateien nur bis zum Neuladen.	Kein Fehler im Programm; auf dem Gerät testen.

Differenzierung

- **Schneller fertig:** Frühstart erkennen: Wer vor „grün“ drückt, bekommt eine Strafzeit (das Startprogramm verwirft frühe Drücke nur). Oder Buch Kapitel 15, Aufgabe 5 (Würfelstatistik über viele Starts).
- **Mehr Unterstützung:** Für Aufgabe 3 das Muster aus „So funktioniert’s“ wörtlich in die Funktionen übernehmen lassen; nur Dateiname und Variablennamen anpassen. Buch Abschnitt 15.4 (`k15_starts.py`) zeigt dasselbe Muster an einem Startzähler.

LEHRERMATERIAL · AB 19

Turtle 1: Vielecke mit Funktionen

Überblick

Lernziel	Die Schülerinnen und Schüler steuern eine Turtle mit `forward`, `right` und `left`, leiten den Drehwinkel eines regelmäßigen n-Ecks (360 / n) her und kapseln das Zeichnen in einer Funktion mit Parametern, aus der sie Rosetten und Sterne bauen.
Dauer	45 Minuten (Aufgabe 1–3 Pflicht, 4–6 Differenzierung)
Voraussetzungen	Schleifen (AB 06), Funktionen mit Parametern (AB 13), Listen (AB 15)
Material	PBC+ Studio. Lösungsdateien auf Anfrage. Optional Kopiervorlage Turtle-Winkelscheibe (L 07)
Buch	Kapitel 16, Abschnitte 16.1–16.4

Stundenverlauf

Zeit	Phase	Inhalt	Sozialform
5 min	Einstieg	„Menschliche Turtle“: Eine Person läuft im Raum nach Ansage („3 Schritte vor, 90 Grad rechts“) ein Quadrat. Frage: Um wie viel Grad hat sie sich insgesamt gedreht?	Plenum
10 min	Aufgabe 1–2	Dreieck abtippen, Winkel verändern, Tabelle ausfüllen und Regel 360 / n finden	Einzel/Partner
15 min	Aufgabe 3	Vieleck-Funktion ergänzen und mit verschiedenen Argumenten aufrufen	Einzel
10 min	Aufgabe 4–6	Rosette, Sterne, Sternenhimmel	Einzel
5 min	Sicherung	Regel an der Tafel: n-Eck = n-mal vorwärts und um 360 / n drehen. Warum ist die Turtle ein guter Parameter? (Dieselbe Funktion für mehrere Turtles.)	Plenum

Typische Fehler

Beobachtung	Ursache	Hilfe
`NameError: name 'forward' isn't defined`	`forward(60)` ohne `t.` geschrieben.	Methoden gehören zur Turtle: `t.forward(60)`.

Beobachtung	Ursache	Hilfe
`TypeError: forward() takes 2 positional arguments but 1 were given` in der Zeile mit `t.forward(60)`	`t = turtle.Turtle` ohne Klammern: `t` ist dann keine Turtle.	`t = turtle.Turtle()`.
`TypeError: left() takes 2 positional arguments but 1 were given`	Winkel vergessen: `t.left()`. Python zählt die Turtle selbst mit, daher „2“.	Winkel eintragen: `t.left(90)`.
Die Turtle zeichnet weiß statt farbig, keine Fehlermeldung	Deutscher oder falscher Farbname (`"gruen"`); unbekannte Namen ergeben Weiß.	Englische Namen: `"green"`, `"red"` …
Nur eine lange Linie statt eines Vielecks	`t.right(winkel)` nicht eingerückt, gehört nicht zur Schleife.	Beide Befehle gleich weit einrücken.
Vieleck schließt sich nicht	Innenwinkel statt Außenwinkel gedreht oder `360 // ecken` bei 7 Ecken.	Regel 360 / n, `/` statt `//`.
`AttributeError: 'Turtle' object has no attribute 'setdirection'`	Groß-/Kleinschreibung: Die Methode heißt `setDirection`.	Schreibweise aus der Hilfe-Leiste (**PBC+ Turtle**) übernehmen.

Differenzierung

- **Schneller fertig:** Buch Kapitel 16, Aufgabe 2 (Häuserreihe) oder 4 (Olympische Ringe).
- **Mehr Unterstützung:** Aufgabe 3 mit festen Zahlen beginnen lassen (`for i in range(5)`, `t.right(72)`) und erst dann die Zahlen durch die Parameter ersetzen. Für Aufgabe 4 `ab19_a3.py` als Startpunkt geben.

LEHRERMATERIAL · AB 20

Turtle 2: Die Turtle sieht Farben

Überblick

Lernziel	Die Schülerinnen und Schüler lassen eine Turtle mit `on_color` auf Farben reagieren, stellen Richtung und Startpunkt mit `setDirection` und `setPosition` ein und steuern einen Roboter mit einer Liste von Abzweigungen durch ein Labyrinth.
Dauer	45 Minuten (Aufgabe 1–3 Pflicht, 4–6 Differenzierung)
Voraussetzungen	AB 19 (Turtle-Grundbefehle), `while` mit Bedingung (AB 08), Funktionen mit Rückgabewert (AB 14), Listen (AB 15)
Material	PBC+ Studio, Startprogramm `ab20_start.py` (an die Klasse verteilen). Lösungsdateien auf Anfrage
Buch	Kapitel 16, Abschnitte 16.2, 16.3 und 16.5

Vorbereitung: `ab20_start.py` auf die Rechner kopieren oder zum Herunterladen bereitstellen. Das Labyrinth steht in der Liste `WAENDE` (Zeile 4 bis 13): Jede Wand ist `[x, y, richtung, laenge]` und wird von der versteckten Turtle `maurer` gezeichnet. Wände mit `pbc.fill_rect` gehen **nicht**: Die Turtle hat eine eigene Zeichenfläche und sieht nur, was Turtles gezeichnet haben.

Stundenverlauf

Zeit	Phase	Inhalt	Sozialform
5 min	Einstieg	Saugroboter/Rasenmäher-Roboter: Woher weiß er, wo die Wand ist? Er hat einen Sensor, der nach vorn schaut. Die Turtle hat so einen Sensor: `on_color`.	Plenum
10 min	Aufgabe 1–2	Startprogramm testen, Bedingung in Worte fassen, abbiegen	Einzel
15 min	Aufgabe 3	Weg als Liste, Schleife mit `if` / `else`	Partner
10 min	Aufgabe 4–6	Ziel erkennen, neuer Start, Rückweg	Einzel
5 min	Sicherung	Warum `forward(1)` und nicht `forward(10)`? Wie würde ein Roboter ohne Liste den Weg finden (Ausblick)?	Plenum

Typische Fehler

Beobachtung	Ursache	Hilfe
Roboter fährt durch die Wand und verschwindet, Programm endet nicht	Zu große Schritte (`forward(5)` o. Ä.) oder falscher Farbname (`"rot"`): `on_color` wird nie `True`.	`forward(1)`, englische Farbnamen; mit ■ **Stop** anhalten.
Roboter bewegt sich gar nicht	`not` vergessen: `while robo.on_color("red")` ist sofort `False`.	Bedingung laut vorlesen: „Solange **nicht** Rot …“
Roboter dreht in die falsche Richtung	`setDirection` (feste Richtung) und `right` (Drehung um) verwechselt.	Abbildung auf dem AB: 0 oben, 90 rechts, 180 unten, 270 links.
Roboter fährt durch die Ziellinie	Bedingung mit `or` statt `and` verknüpft.	„Weder Rot noch Grün“ = `not … and not …`.
Eigene Wände mit `pbc.fill_rect` werden nicht erkannt bzw. verschwinden	Die Turtle hat eine eigene Zeichenfläche.	Wände mit der Turtle `maurer` zeichnen (Liste `WAENDE` erweitern).

Differenzierung

- **Schneller fertig:** Eigenes Labyrinth: Wände in `WAENDE` ergänzen oder verschieben und den passenden Weg als Liste finden lassen. Partner tauschen Labyrinthe aus.
- **Mehr Unterstützung:** Für Aufgabe 3 zuerst ohne Liste arbeiten: viermal „fahren, drehen“ untereinander kopieren (wie Aufgabe 2), dann erst die Wiederholung in eine Schleife mit Liste umbauen.
- **Ausblick:** Buch Kapitel 16, Aufgabe 5: Eine Funktion misst den Abstand zur Wand in alle vier Richtungen.

LEHRERMATERIAL · AB 21

Canvas und Sprites

Überblick

Lernziel	Die Schülerinnen und Schüler zeichnen auf einem Canvas im Speicher und zeigen ihn mit `pbc.show`, machen eine flackernde Animation flimmerfrei, legen einen Hintergrund für Sprites fest und steuern ein Sprite mit dem Joystick, ohne dass es den Bildschirm verlässt.
Dauer	45 Minuten (Aufgabe 1–3 Pflicht, 4–6 Differenzierung)
Voraussetzungen	Zeichenbefehle (AB 05), Schleifen und Animation (AB 07), Joystick (AB 09)
Material	PBC+ Studio, Bild `heissluftballon.png` (Ordner `material`). Startdateien `ab21_flackern.py`, `ab21_start.py`. Lösungsdateien auf Anfrage
Buch	Kapitel 17 und 18

Vorbereitung: Die Bilder müssen in **jedem Browser** einmal hochgeladen werden (**Dateien → Bilder verwalten… → PNG hochladen… → Bild von der Festplatte wählen… → Hochladen**). Für den Emulator bleiben sie im Browser gespeichert, auch nach dem Neuladen der Seite; auf Rechnern, deren Browserdaten beim Abmelden gelöscht werden, muss das jede Stunde neu geschehen. Wer auf dem Gerät testet, lädt die Bilder mit dem Ziel **PBC+** zusätzlich auf das Gerät. `heissluftballon.png` am besten vorab auf ein Netzlaufwerk oder in die Lernplattform legen.

Stundenverlauf

Zeit	Phase	Inhalt	Sozialform
5 min	Einstieg	`ab21_flackern.py` am Beamer zeigen: Warum flackert es? Idee „erst fertig malen, dann zeigen“ (Daumenkino, Tafelbild hinter einem Vorhang).	Plenum
10 min	Aufgabe 1–2	Canvas abtippen, Flackern beseitigen	Einzel
5 min	Bilder	Bild hochladen gemeinsam vormachen	Plenum
15 min	Aufgabe 3–4	Ballon mit Hintergrund und Joystick, Randbegrenzung	Einzel/Partner
5 min	Aufgabe 5–6	Text im Hintergrund, Wind	Einzel

Zeit	Phase	Inhalt	Sozialform
5 min	Sicherung	Tafel: Sprites zeichnen bei jeder Bewegung den ganzen Bildschirm neu (Hintergrund + Sprites). Folge: Was nicht im Hintergrund-Canvas steht, verschwindet.	Plenum

Typische Fehler

Beobachtung	Ursache	Hilfe
`OSError: [Errno 2] ENOENT` in der Zeile mit `pbc.sprite`	Bild nicht hochgeladen, Name falsch geschrieben (Groß-/Kleinschreibung, z. B. `"Heissluftballon.png"`) oder für das andere Ziel hochgeladen.	**Dateien → Bilder verwalten…** öffnen und den Namen vergleichen.
`AttributeError: 'Canvas' object has no attribute 'clear'`	Auf dem Canvas heißt es `fill`.	`bild.fill(farbe)`
`TypeError: function doesn't take keyword arguments`	`bild.ellipse(…, fill=True)`: Canvas-Formen kennen das Wort `fill=` nicht.	`True` ohne `fill=` schreiben.
Bildschirm bleibt unverändert	`pbc.show(bild)` vergessen oder außerhalb der Schleife.	Drei Schritte: zeichnen – zeigen – warten.
Sprite unsichtbar, keine Fehlermeldung	`show()` vergessen.	Sprite nach dem Anlegen zeigen.
Hintergrund schwarz, nur der Ballon ist zu sehen	`pbc.Sprite._background = himmel` fehlt; Sprites malen ohne Hintergrund auf Schwarz.	Hintergrund festlegen, dann `pbc.show(himmel)`.
Ballon „klebt“ am Rand oder springt nicht mehr zurück	Bedingung prüft den falschen Rand (`ballon.x` statt `ballon.x + ballon.width`).	Skizze: Position = linke obere Ecke.
Im Emulator ist der Canvas nach `pbc.show` plötzlich einfarbig	Emulator-Besonderheit: Nach `name = pbc.Canvas(…)` wirkt `pbc.clear(farbe)` dort auf diesen Canvas (auf dem Gerät nicht).	Canvas und direktes Zeichnen nicht mischen.

Differenzierung

- **Schneller fertig:** Buch Kapitel 18, Aufgabe 5 (Rakete mit Tankanzeige im Hintergrund-Canvas) oder Aufgabe 4 (Rundflug mit `glide_to`).

- **Mehr Unterstützung:** Für Aufgabe 2 die Lösung `ab21_a2.py` neben `ab21_flackern.py` legen und die Unterschiede markieren lassen. Für Aufgabe 4 eine Skizze des Bildschirms mit dem Ballon-Rechteck (x, y, Breite, Höhe) an die Tafel zeichnen.

LEHRERMATERIAL · AB 22

Kollisionen

Überblick

Lernziel	Die Schülerinnen und Schüler erkennen Kollisionen mit `touches_edge` (Rand), `on_color` (Farbe im Hintergrund-Canvas) und `touches` (zwei Sprites) und reagieren darauf: zurücksetzen, Punkte zählen, Spiel beenden. Sie wissen, dass `touches` mit den Rechtecken der Bilder rechnet.
Dauer	45 Minuten (Aufgabe 1–3 Pflicht, 4–6 Differenzierung)
Voraussetzungen	AB 21 (Canvas, Sprites, Hintergrund), `random.randint` (AB 04)
Material	PBC+ Studio, Bilder `rennauto.png`, `stern.png`, `huetchen.png` (Ordner `material`). Startdatei `ab22_start.py`. Lösungsdateien auf Anfrage; die Lösungen bauen aufeinander auf
Buch	Kapitel 19

Vorbereitung: Die drei Bilder müssen in jedem Browser hochgeladen werden (**Dateien → Bilder verwalten… → PNG hochladen…**, siehe AB 21). Die Aufgabennummern der Lösungsdateien: `ab22_a1.py` gehört zu Aufgabe 2, `ab22_a2.py` zu Aufgabe 3, `ab22_a3.py` zu Aufgabe 4, `ab22_a4.py` zu Aufgabe 5, `ab22_a6.py` zu Aufgabe 6. Aufgabe 1 braucht keine eigene Datei.

Stundenverlauf

Zeit	Phase	Inhalt	Sozialform
5 min	Einstieg	Ein Spiel (z. B. ein Jump-‚n'-Run) zeigen oder beschreiben lassen: Welche Zusammenstöße prüft das Spiel ständig? Sammeln: Rand, Boden/Farbe, Gegner, Münzen.	Plenum
10 min	Aufgabe 1–2	Startprogramm lesen, Rand	Einzel
10 min	Aufgabe 3	Gras mit `on_color`	Einzel
15 min	Aufgabe 4–6	Sammeln, Ausweichen, Punkteanzeige	Partner
5 min	Sicherung	Tafel: drei Befehle, drei Fragen („Bin ich draußen? Stehe ich auf …? Berühre ich …?“). Abbildung „Rechtecke“ aus Buch 19.1 zeigen.	Plenum

Typische Fehler

Beobachtung	Ursache	Hilfe
`on_color` meldet nie etwas	Kein Hintergrund-Canvas (`pbc.Sprite._background`) oder das Gras wurde mit `pbc.fill_rect` direkt auf den Bildschirm gemalt.	Spielfeld in einen Canvas zeichnen, als Hintergrund setzen, `pbc.show(strecke)`.
`on_color` meldet nie etwas, obwohl der Hintergrund stimmt	Gras mit einer eigenen Farbe (`pbc.rgb(0, 200, 0)`) gemalt, aber auf `pbc.GREEN` geprüft.	Mit derselben Farbe prüfen, mit der gemalt wurde.
`AttributeError: module 'pbc' has no attribute 'Green'`	Farbkonstanten werden groß geschrieben.	`pbc.GREEN`
`OSError: [Errno 2] ENOENT` in der Zeile mit `pbc.sprite`	Bild nicht (im richtigen Ziel) hochgeladen oder Name falsch.	**Dateien → Bilder verwalten…**
Punkte zählen sehr schnell hoch	Der Stern wird nach dem Treffer nicht versetzt; `touches` ist in jedem Durchlauf wieder `True`.	Nach dem Punkt den Stern sofort an eine neue Stelle setzen.
Auto „zittert“ am Rand oder bleibt hängen	`set_by(dx, dy)` statt `set_by(-dx, -dy)` zum Zurücknehmen.	Schritt rückwärts = gleiche Zahlen mit Minus.
Hütchen kommt nach dem ersten Mal nicht wieder	Prüfung `huetchen.y > pbc.HEIGHT` fehlt oder setzt es unten statt oben neu an.	Skizze: Wo ist „unten hinaus“?

Differenzierung

- **Schneller fertig:** Mehrere Hütchen in einer Liste (wie im Buch Kapitel 19, Aufgabe 5), oder das Hütchen wird mit jedem Punkt schneller.
- **Mehr Unterstützung:** Aufgabe 3 als Lückentext an die Tafel: `if auto._____(pbc.GREEN):` / `print("Gras!")` / `auto._____(113, 240)`. Für Aufgabe 4 die Zeilen aus `ab22_a3.py` vorgeben und nur einordnen lassen.

Projekt Weltraumspiel 1: Raumschiff und Sterne

Überblick

Lernziel	Die Schülerinnen und Schüler richten einen Sprite-Hintergrund ein, steuern ein Sprite mit dem Joystick, lagern die Steuerung in eine Funktion mit Randprüfung aus und verstehen, dass Sprites erst nach `show()` sichtbar sind.
Dauer	45 Minuten (Aufgabe 1–3 Pflicht, 4–5 Differenzierung); erste von drei Projektstunden (AB 23a–c)
Voraussetzungen	AB 13 (Funktionen), AB 21 (Canvas und Sprites), AB 22 (Kollisionen)
Material	PBC+ Studio; Bilder `weltraum.png`, `raumschiff.png`, `meteor.png`, `stern.png` aus dem Material-Paket; Startprogramm `ab23a_start.py`; Lösungsdateien auf Anfrage
Buch	Kapitel 18, 19 und 21 (das Buch baut dort ein anderes Spiel, „Apfelfang“, nach demselben Muster)

Vorbereitung: Die vier PNG-Dateien und `ab23a_start.py` für die Klasse bereitstellen (Tauschordner, Lernplattform). Die Bilder müssen in jedem Browser einmal über **Dateien → Bilder verwalten…** hochgeladen werden, alle vier auf einmal mit **ZIP hochladen…**; sie bleiben dort gespeichert, solange die Browserdaten nicht gelöscht werden. Für Tests auf dem Gerät müssen die Bilder auch auf dem PBC+ liegen (Buch, Anhang D).

Stundenverlauf

Zeit	Phase	Inhalt	Sozialform
5 min	Einstieg	Fertiges Spiel (`ab23c_a4.py`) am Beamer vorführen: „Das baut ihr in drei Stunden.“ Grob-Plan an die Tafel: Teil 1 Schiff, Teil 2 Meteore und Leben, Teil 3 Punkte und Highscore.	Plenum
5 min	Vorbereitung	Bilder hochladen, Startprogramm öffnen	Einzel
15 min	Aufgabe 1–2	Ausprobieren, Vorhersagen prüfen	Einzel/Partner
15 min	Aufgabe 3–5	Funktion `steuere`, Erweiterungen	Einzel
5 min	Sicherung	Warum eine Funktion? (Spielschleife bleibt kurz, Steuerung an einer Stelle änderbar.) Jede/r speichert `weltraum_1.py`.	Plenum

Typische Fehler

Beobachtung	Ursache	Hilfe
`OSError: [Errno 2] ENOENT` bei `pbc.load_image` oder `pbc.sprite`	Bild nicht hochgeladen oder Name falsch geschrieben.	**Dateien → Bilder verwalten…** zeigt die hochgeladenen Namen.
Raumschiff ist nicht zu sehen, keine Fehlermeldung	`schiff.show()` fehlt.	Sprites sind anfangs unsichtbar.
Hintergrund wird schwarz, sobald das Schiff erscheint	`pbc.Sprite._background = himmel` fehlt.	Hintergrund festlegen, bevor Sprites gezeigt werden.
`TypeError: set_by() takes 3 positional arguments but 2 were given`	`schiff.set_by(-5)` statt `schiff.set_by(-5, 0)`. Python zählt das Sprite selbst als erstes Argument mit, daher „3“ statt „2“.	`set_by` braucht immer zwei Zahlen: dx und dy.
`NameError: name 'random' isn't defined` (Aufgabe 5)	`import random` fehlt.	Imports oben ergänzen.
Schiff bewegt sich nicht mehr	Nach dem Auslagern in `steuere` wird die Funktion in der Schleife nicht aufgerufen, oder `steuere` steht unter der Schleife.	Funktionen oben definieren, in der Schleife `steuere(schiff)` aufrufen.
Steuerung reagiert „zäh“	Jede Bewegung eines Sprites zeichnet den ganzen Bildschirm neu (etwa 40 ms).	Normal; größeres `TEMPO` statt kürzerer Wartezeit.

Die Meldung bei einem fehlenden Bild (Emulator und Gerät gleich); PBC+ Studio öffnet dazu das Hinweisfenster „Datei nicht gefunden“ mit dem Tipp, das Bild über **Dateien → Bilder verwalten…** hochzuladen:

```
Terminal
Traceback (most recent call last):
  File "<stdin>", line 3, in <module>
OSError: [Errno 2] ENOENT
```

Differenzierung

- **Schneller fertig:** Aufgabe 5; danach die genaue Randprüfung aus dem Buch (Kapitel 21.2) übernehmen und mit `TEMPO = 15` testen.

- **Mehr Unterstützung:** Aufgabe 3 mit `ab23a_a2.py` vergleichen lassen; Aufgabe 2 als Partnerarbeit (eine Person sagt vorher, die andere testet). AB 23b startet mit `ab23b_start.py`, sodass niemand abgehängt wird.

LEHRERMATERIAL · AB 23B

Projekt Weltraumspiel 2: Asteroiden aus der Liste

Überblick

Lernziel	Die Schülerinnen und Schüler verwalten mehrere gleichartige Sprites in einer Liste, bewegen sie mit einer `for`-Schleife, erkennen Treffer mit `touches`, zählen Leben, zeigen Text auf dem Hintergrund-Canvas an und beenden das Spiel mit einer Bedingung.
Dauer	45 Minuten (Aufgabe 1–3 Pflicht, 4–5 Differenzierung); zweite von drei Projektstunden
Voraussetzungen	AB 23a; AB 17 (Listen im Spiel)
Material	Startprogramm `ab23b_start.py` (Lösung von AB 23a, Aufgabe 3); Lösungsdateien auf Anfrage; Bilder wie in AB 23a
Buch	Kapitel 14, 19 und 21.3–21.4

Stundenverlauf

Zeit	Phase	Inhalt	Sozialform
5 min	Einstieg	Rückblick: Was kann das Spiel schon? Frage: „Wie bekommen wir zehn Meteore, ohne zehnmal dasselbe zu schreiben?“ → Liste.	Plenum
10 min	Aufgabe 1	Ein Meteor fällt und kommt wieder	Einzel
15 min	Aufgabe 2–3	Liste, Treffer, Leben	Einzel/Partner
10 min	Aufgabe 4–5	Spielende, eigenes Tempo	Einzel
5 min	Sicherung	Warum gehört der Text auf `himmel`? (Jede Sprite-Bewegung zeichnet den Bildschirm aus dem Hintergrund neu.) Speichern als `weltraum_2.py`.	Plenum

Typische Fehler

Beobachtung	Ursache	Hilfe
Keine Meteore zu sehen, keine Fehlermeldung	`meteore.append(meteor)` fehlt, die Liste bleibt leer; oder `meteor.show()`	Mit `print(len(meteore))` nachsehen.

Beobachtung	Ursache	Hilfe
	fehlt.	
Nur ein Meteor bewegt sich	`meteor.set_by(…)` ist nicht in die `for`-Schleife eingerückt; bewegt wird nur der zuletzt angelegte.	Einrückung prüfen.
„Leben“ blinkt kurz auf und verschwindet	Text direkt mit `pbc.text12` statt auf `himmel` geschrieben.	Auf `himmel` schreiben, dann `pbc.show(himmel)`.
`NameError: name 'himmel' isn't defined` in `zeige_leben`	`zeige_leben(leben)` wird aufgerufen, bevor `himmel = pbc.load_image(…)` gelaufen ist.	Aufruf erst nach dem Laden des Hintergrunds.
Ein Treffer kostet mehrere Leben	`nach_oben(meteor)` fehlt nach dem Treffer; der Meteor berührt das Schiff mehrere Durchläufe lang.	Nach jedem Treffer den Meteor neu starten.
Spiel wird sehr langsam	Viele Meteore: Jede Sprite-Bewegung zeichnet den ganzen Bildschirm neu (etwa 40 ms).	3–4 Meteore, dafür größeres `TEMPO_METEOR`.

So sieht der `NameError` aus, wenn `zeige_leben` zu früh aufgerufen wird (Testprogramm `ab23_fehler_himmel.py`):

Terminal

```
Traceback (most recent call last):
  File "<stdin>", line 10, in <module>
  File "<stdin>", line 5, in zeige_leben
NameError: name 'himmel' isn't defined
```

Differenzierung

- **Schneller fertig:** Aufgabe 5; zusätzlich die Meteore beim Neustart auch ein neues Tempo bekommen lassen (`tempi[i] = random.randint(4, 12)`).
- **Mehr Unterstützung:** Aufgabe 2 mit `ab23b_a2.py` vergleichen; für Aufgabe 3 die Bedingung `if meteor.touches(schiff):` vorgeben.
- **Nachzügler:** Wer AB 23b nicht beendet, startet AB 23c mit `ab23c_start.py`.

LEHRERMATERIAL · AB 23C

Projekt Weltraumspiel 3: Punkte und Highscore

Überblick

Lernziel	Die Schülerinnen und Schüler ergänzen Punkte und einen Endbildschirm, speichern einen Highscore mit `open` / `with` in einer Datei und lesen ihn wieder, unterscheiden das Verhalten von Dateien im Emulator und auf dem Gerät und planen eine eigene Erweiterung in kleinen Schritten.
Dauer	45 Minuten (Aufgabe 1–3 Pflicht, 4–5 Differenzierung); dritte von drei Projektstunden, bei Bedarf eine weitere Stunde für Aufgabe 4–5
Voraussetzungen	AB 23b; AB 18 (Highscore in einer Datei)
Material	Startprogramm `ab23c_start.py` (Lösung von AB 23b, Aufgabe 4); Lösungsdateien auf Anfrage; Bilder wie in AB 23a
Buch	Kapitel 15 und 21.5–21.6

Stundenverlauf

Zeit	Phase	Inhalt	Sozialform
5 min	Einstieg	„Wann ist ein Spiel ein Spiel?“ – Ziel, Punkte, Ende, Ansporn (Rekord).	Plenum
10 min	Aufgabe 1	Punkte und Endbildschirm	Einzel
15 min	Aufgabe 2–3	Highscore-Datei, Testen, Neuladen	Einzel/Partner
10 min	Aufgabe 4	Erweiterung planen (Schritte notieren) und beginnen	Einzel
5 min	Sicherung	Kurzes Vorführen von zwei, drei Spielen; Frage: Wo bleibt die Datei im Emulator, wo auf dem Gerät?	Plenum

Typische Fehler

Beobachtung	Ursache	Hilfe
`NameError: name 'os' isn't defined`	`import os` fehlt.	Import oben ergänzen; das Hinweisfenster sagt es auch.

Beobachtung	Ursache	Hilfe
`TypeError: object with buffer protocol required`	`f.write(punkte)`: `write` braucht einen Text.	`f.write(str(punkte))`
`OSError: [Errno 2] ENOENT` beim ersten Spiel	Datei wird gelesen, ohne vorher mit `os.listdir()` zu prüfen, ob sie existiert.	Prüfung wie in `lade_rekord`.
`ValueError: invalid syntax for integer with base 10: ''`	Die Datei ist leer (z. B. weil beim Testen `"w"` ohne `write` benutzt wurde).	Datei mit `os.remove("weltraum_rekord.txt")` löschen oder neu schreiben.
Rekord ist nach dem Neuladen weg	Emulator: Dateien nur bis zum Neuladen.	Kein Fehler; auf dem Gerät bleibt die Datei.
Punkte steigen auch bei Treffern	`punkte += 1` steht vor dem `if` / `elif` statt im `elif`-Zweig.	Einrückung und Zweige prüfen.

Die Meldungen im Terminal (Testprogramme `ab23_fehler_write.py` und `ab23_fehler_datei.py`):

Terminal

```
Traceback (most recent call last):
  File "<stdin>", line 3, in <module>
TypeError: object with buffer protocol required
```

Terminal

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
OSError: [Errno 2] ENOENT
```

Beim `OSError` öffnet PBC+ Studio zusätzlich das Hinweisfenster „Datei nicht gefunden“.

Differenzierung

- **Schneller fertig:** Aufgabe 5; danach Neustart mit Taste A (die Spielschleife und das Zurücksetzen der Werte in eine Funktion `spiele()` packen, die `punkte` zurückgibt).
- **Mehr Unterstützung:** Aufgabe 2 mit `ab23c_a2.py` vergleichen; für Aufgabe 4 die Idee „Immer schneller“ empfehlen (eine Zeile plus eine Änderung).
- **Weiterführung:** AB 24 (eigenes Projekt). Das Weltraumspiel ist ein gutes Muster für die Gliederung eigener Programme.

Projekt: Mein eigenes Programm

Überblick

Lernziel	Die Schülerinnen und Schüler entwickeln ein eigenes Programm von der Idee bis zur Präsentation: Sie beschreiben die Idee, planen Stufen mit Testkriterien, bauen schrittweise, dokumentieren Fehler und ihre Lösung, räumen den Code auf und stellen das Ergebnis vor.
Dauer	4–6 Unterrichtsstunden (Vorschlag unten für 6 Stunden)
Voraussetzungen	möglichst AB 23a–c; je nach Projekt die passenden AB (Listen, Dateien, Turtle, Sprites …)
Material	AB 24 als Planungs- und Arbeitsbogen (einmal pro Person, ggf. A3); Gerüst `ab24_vorlage.py`; Beispielprojekt `ab24_beispiel.py` (Reaktionsduell) auf Anfrage; alle Bilder aus dem Material-Paket; Buch Kapitel 22 als Nachschlagewerk; optional Geräte für Projekte mit Bewegungssensor
Buch	Kapitel 22 (Vorgehen, Fehlersuche, Vorstellen), Kapitel 21 (Beispiel für einen Stufenplan)

Vorbereitung: Rechtzeitig klären, ob Projekte mit dem Bewegungssensor erlaubt sind; im Emulator liegt das Gerät immer flach, diese Projekte brauchen echte Geräte. Für die Präsentation einen Rechner mit Beamer und hochgeladenen Bildern bereitstellen; Schülerinnen und Schüler geben ihre `.py`-Datei und eigene Bilder vorher ab. Im Emulator gehen gespeicherte Dateien (z. B. Highscores) beim Neuladen verloren – für die Vorführung unproblematisch, aber vorher ansagen.

Stundenverlauf

Stunde	Phase	Inhalt	Sozialform
1	Idee und Plan	Beispielprojekt `ab24_beispiel.py` vorführen und gemeinsam rückwärts planen: Was wäre Stufe 1? AB Aufgabe 1–3 ausfüllen. **Freigabe** des Plans durch die Lehrkraft (Umfang prüfen!).	Plenum, Einzel
2–4	Bauen	Stufen umsetzen, nach jeder Stufe testen und abhaken, Fehlertabelle führen (Aufgabe 4). Kurze Stand-up-Runde zu Beginn jeder Stunde: „Welche Stufe heute?“	Einzel/Partner
5	Aufräumen	Checkliste (Aufgabe 5), Kopfkommentar, Extras (Aufgabe 6); Probedurchlauf der Vorstellung mit einem Partner	Einzel/Partner

Stunde	Phase	Inhalt	Sozialform
6	Vorstellen	Präsentationen à 3–5 Minuten, kurzes Feedback (ein Lob, eine Frage)	Plenum

Bei nur 4 Stunden: Stunde 1 wie oben, 2–3 Bauen, 4 Aufräumen und Vorstellen in Kleingruppen (Galerie-Rundgang statt Beamer).

Bewertungshinweise

Bewertet werden Produkt **und** Prozess. Der Planungsbogen ist Teil der Leistung. Vorschlag für ein Raster mit 20 Punkten:

Kriterium	4 Punkte	2 Punkte	0–1 Punkte
Funktion	Programm läuft ohne Fehler und tut, was der Steckbrief verspricht.	Kernfunktion läuft, einzelne Teile fehlen oder sind fehlerhaft.	Programm startet nicht oder tut kaum etwas.
Planung	Steckbrief vollständig, Stufen sinnvoll und mit Testkriterium, Plan realistisch.	Plan vorhanden, Stufen ungenau oder zu groß.	kein oder nur nachträglich ausgefüllter Plan
Arbeitsweise und Fehlersuche	Stufen nachweisbar einzeln gebaut (Zwischenstände), Fehlertabelle mit Ursache und Lösung.	einige Zwischenstände oder Fehlernotizen	keine Spuren des Vorgehens
Code-Qualität	Kopfkommentar, sprechende Namen, sinnvolle Funktionen, Konstanten, keine Test-`print`-Reste.	teilweise gegliedert, Namen gemischt	alles in einer Schleife, Namen wie `a`, `x2`
Vorstellen	klar, Code-Stück verständlich erklärt, Schwierigkeit benannt.	vorgeführt, Erklärung knapp	nur gezeigt, keine Erklärung

- **Schwierigkeit berücksichtigen:** Ein kleines, sauber gebautes Projekt kann besser abschneiden als ein großes, halbfertiges. Bei sehr anspruchsvollen Projekten (Snake, Labyrinth) die Kriterien „Funktion“ und „Arbeitsweise“ entsprechend wohlwollend auslegen.
- **Eigenleistung:** Bei Unsicherheit eine Stelle im Code erklären lassen („Was passiert, wenn ich hier 5 statt 3 schreibe?“). Übernommene Teile aus Buch oder AB sind erlaubt, wenn sie im Kopfkommentar genannt werden.
- **Partnerarbeit:** Möglich; dann Steckbrief gemeinsam, Fehlertabelle und Vorstellung mit klarer Aufteilung, wer welche Stufe gebaut hat.

Typische Fehler

Beobachtung	Ursache	Hilfe
Nach zwei Stunden läuft noch nichts	Idee zu groß, alles auf einmal geschrieben.	Zurück zur kleinsten Fassung; Stufe 1 so klein machen, dass sie in 20 Minuten fertig ist.
`NameError: local variable referenced before assignment`	Eine Funktion verändert eine Variable des Hauptprogramms (z. B. `punkte = punkte + 1`).	`global punkte` in der Funktion oder besser: Wert als Parameter übergeben und mit `return` zurückgeben (Buch Kapitel 12).
`??` statt Umlaut auf dem Display	Die Display-Schriften kennen keine Umlaute.	„Tschuess“ statt „Tschüss“; in der Konsole gehen Umlaute.
Programm war gestern fertig, heute kaputt	Ohne Zwischenstände weitergebaut.	Jede Stufe unter eigenem Namen speichern.
Bei der Vorstellung fehlen Bilder (`OSError: [Errno 2] ENOENT`)	Bilder nur im eigenen Browser hochgeladen.	Bilder mit dem Programm abgeben, am Präsentationsrechner hochladen.
Highscore „weg“	Emulator: Dateien nur bis zum Neuladen der Seite.	Auf dem Gerät vorführen oder vorher ansagen.

Die Meldung beim Ändern einer Variablen in einer Funktion (`ab24_fehler_global.py`):

Terminal

```
Traceback (most recent call last):
  File "<stdin>", line 8, in <module>
  File "<stdin>", line 5, in treffer
NameError: local variable referenced before assignment
```

Differenzierung

- **Schwächere Schülerinnen und Schüler:** Projekt aus einem bekannten AB weiterentwickeln (z. B. Weltraumspiel mit eigener Grafik, Würfel aus Kapitel 22, Ampel aus AB 11). Stufenplan gemeinsam mit der Lehrkraft.
- **Stärkere Schülerinnen und Schüler:** anspruchsvolle Idee mit Listen in Listen (Labyrinth, Snake) oder Dateien (Bestenliste mit Namen); als Extra ein zweiter Modus oder ein Startmenü.
- **Partnerarbeit:** Pair-Programming mit festen Wechseln (alle 10 Minuten Tastatur tauschen).

Bewertung

Kriterien und Raster für die Projekte (AB 23 und AB 24) und Hinweise zur Bewertung kleinerer Programmieraufgaben. Die Raster sind Vorschläge; passen Sie Gewichtung und Punkte an Ihre Lerngruppe an.

Grundsätze

- **Prozess und Produkt:** Bewertet wird nicht nur das fertige Programm, sondern auch Planung (Planungsbogen, L 07 Vorlage G), Vorgehen in kleinen Schritten und der Umgang mit Fehlern (Testprotokoll).
- **Verstehen vor Umfang:** Ein kleines, sauberes Programm, das die Gruppe erklären kann, ist mehr wert als ein großes, das sie nicht versteht. Ein kurzes Gespräch am Rechner („Was passiert in Zeile 23? Was würde passieren, wenn …?“) klärt das schnell.
- **Fremder Code:** Übernommene Teile (aus dem Buch, von Mitschülern, aus dem Internet, von KI-Werkzeugen) müssen gekennzeichnet sein (Kommentar) und erklärt werden können.

Raster für das Projekt (AB 24)

Kriterium	0	1	2	3
Planung	kein Plan	Idee beschrieben	Plan mit Skizze, Bedienung, Regeln	vollständiger Plan inkl. Variablen, Funktionen, Arbeitsplan; angepasst, wenn nötig
Funktion	läuft nicht	läuft mit Fehlern	Kernidee funktioniert	alle geplanten Funktionen, robust (auch bei unerwarteten Eingaben)
Struktur	ein langer Block	erste Funktionen	sinnvoll in Funktionen gegliedert, Parameter genutzt	Funktionen mit Rückgabewerten, keine Wiederholungen, Listen statt vieler Einzelvariablen
Lesbarkeit	unverständliche Namen	teils sprechende Namen	sprechende Namen, Einrückung sauber	zusätzlich Kommentare, die das *Warum* erklären
Python-Konzepte	nur Anweisungsfolge	Schleifen und Verzweigungen	zusätzlich Funktionen und Listen	zusätzlich Dateien, Canvas/Sprites oder

Kriterium	0	1	2	3
				eigene Erweiterungen sinnvoll eingesetzt
Testen und Fehler	kein Testprotokoll	einzelne Einträge	Fehler beschrieben und behoben	systematisch getestet (Grenzfälle), Fehlerursachen erklärt
Präsentation	–	zeigt das Programm	erklärt Bedienung und Aufbau	erklärt einen Programmteil im Code und eine gelöste Schwierigkeit

Notenvorschlag (21 Punkte): ab 19 sehr gut, ab 16 gut, ab 12 befriedigend, ab 8 ausreichend, ab 4 mangelhaft.

Raster für das geführte Projekt (AB 23a–c)

Kriterium	erreicht?
Stufe 1: Spielfeld und Spielfigur, Steuerung in Funktionen	☐
Stufe 2: Gegner aus einer Liste, Kollision, Leben	☐
Stufe 3: Punkte, Spielende, Highscore-Datei	☐
Eigene Erweiterung umgesetzt und erklärt	☐
Programm gespeichert, Namen und Kommentare sinnvoll	☐

Kleinere Aufgaben

Für einzelne Arbeitsblätter eignet sich eine einfache Rückmeldung statt einer Note:

- **läuft / läuft nicht** (mit der Fehlermeldung als Gesprächsanlass),
- **Code-Lese-Aufgaben** („Was gibt das Programm aus?“) prüfen das Verständnis unabhängig vom Tippen,
- **„Finde den Fehler“-Aufgaben** prüfen das Lesen von Fehlermeldungen.

Für schriftliche Lernkontrollen bieten sich Aufgaben aus den Übungsaufgaben des Buchs an, besonders die mit ⭐ und ⭐⭐, die Code lesen, Ausgaben vorhersagen oder Fehler finden lassen.

LEHRERMATERIAL · L 07

Kopiervorlagen

Druckfertige Vorlagen für den Unterricht. Jede Vorlage steht auf einer eigenen Seite und kann einzeln kopiert werden.

Vorlage	Inhalt	Einsatz
A	Bildschirm-Koordinaten 240 × 280, (0, 0) oben links	AB 02, 05–07, 09, 21–23: Positionen planen, Zeichnungen entwerfen. Auch als Folie.
B	Turtle-Winkelscheibe: `setDirection(0)` = oben, 90 = rechts	AB 19–20: Drehungen und Richtungen
C	Turtle-Koordinaten, (0, 0) in der Mitte, y nach oben	AB 19–20: `setPosition`, Figuren planen
D	Farbkonstanten mit Rot/Grün/Blau-Werten	AB 05, 21–22: Farben mischen mit `pbc.rgb`, Kollisionsfarben in Bildern
E	Befehlskarte `pbc` und `turtle` (2 Seiten)	ab AB 02 als Nachschlagekarte, laminiert am Platz
F	Fehlermeldungen-Spickzettel	ab AB 01
G	Projekt-Planungsbogen (2 Seiten)	AB 23–24
H	Tasten-Übersicht Gerät und Emulator	ab AB 09

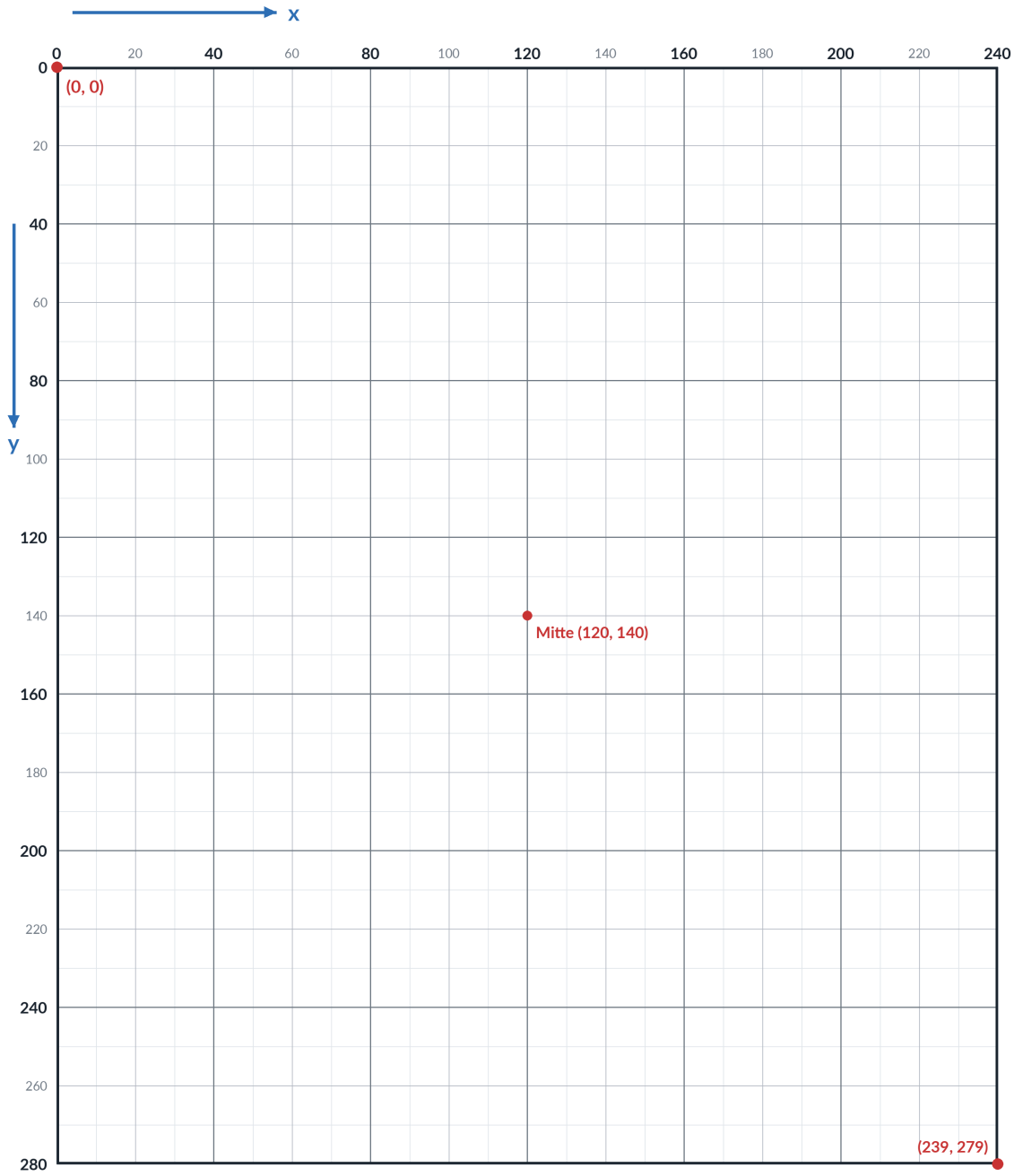
Hinweise zu den Vorlagen

- **A und C** sind maßstabsgetreu: ein Kästchen = 10 Pixel. Achten Sie beim Wechsel von A zu C auf die unterschiedliche Richtung von y.
- **B** zeigt die Richtungen so, wie das Modul `turtle` sie verwendet (wie ein Kompass, im Uhrzeigersinn): `setDirection(0)` = oben, 90 = rechts, 180 = unten, 270 = links. `right(w)` dreht im Uhrzeigersinn, `left(w)` dagegen. Eine neue Turtle startet in der Mitte und blickt nach oben.
- **D:** Das Display zeigt 65 536 Farben (RGB565). `pbc.rgb(r, g, b)` rundet die Werte deshalb leicht; die Tabelle nennt Werte, die genau die Konstante ergeben.

KOPIERVORLAGE A

Der Bildschirm: 240 × 280 Pixel

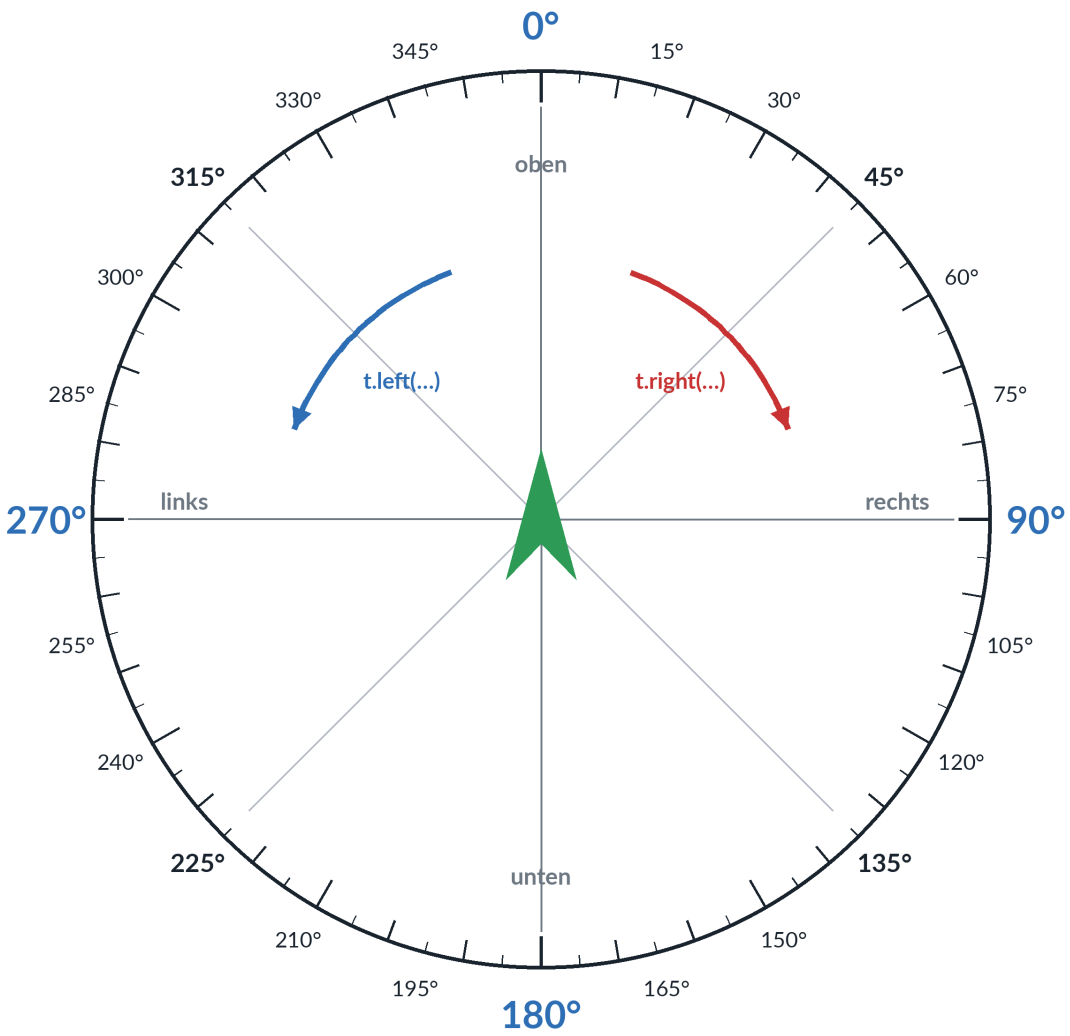
(0, 0) ist oben links. x wächst nach rechts, y wächst nach **unten**. Ein Kästchen = 10 Pixel.



KOPIERVORLAGE B

Turtle: Richtungen und Drehungen

`t.setDirection(0)` zeigt nach oben, 90 nach rechts. `t.right(w)` dreht im Uhrzeigersinn, `t.left(w)` dagegen.



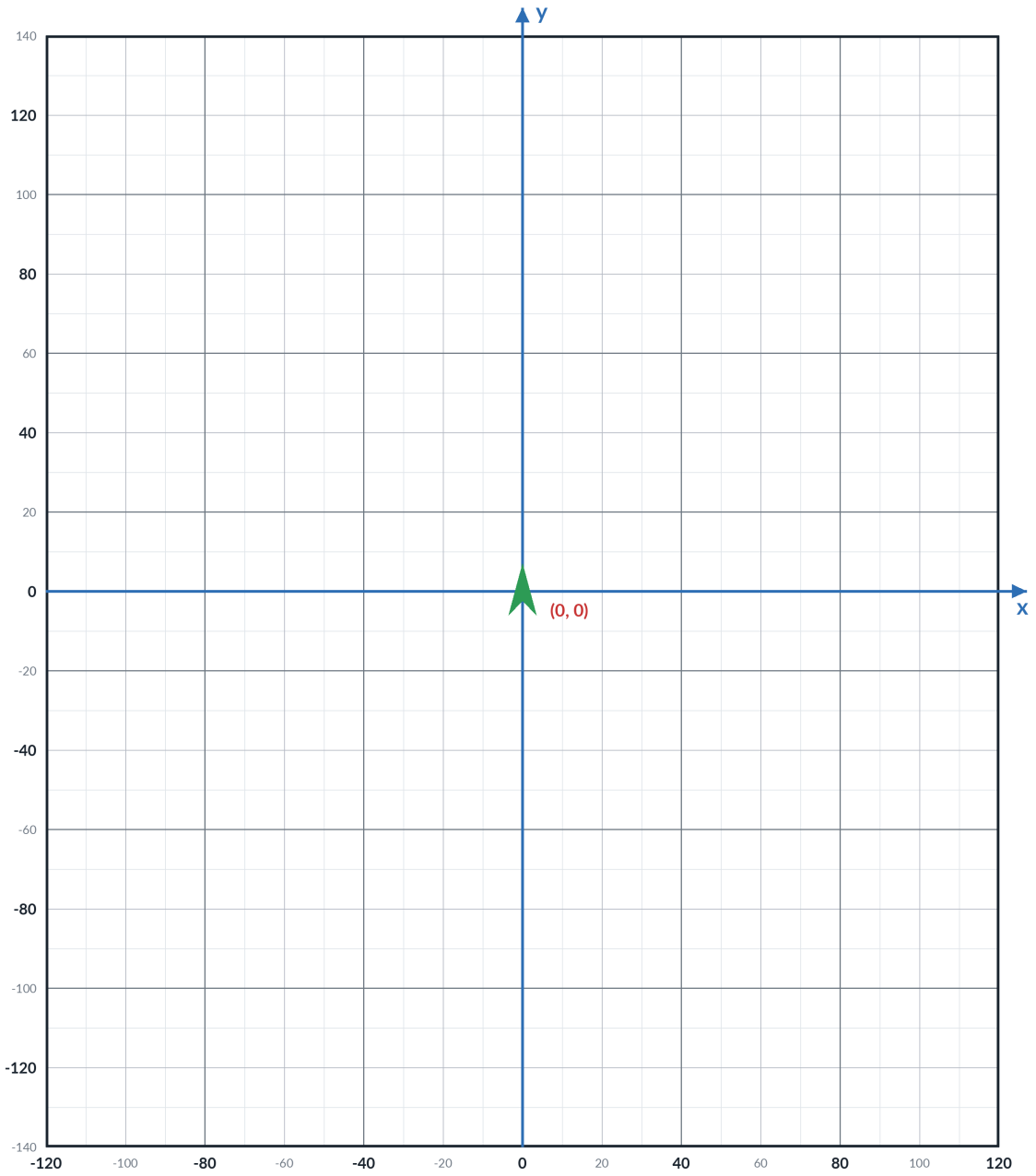
Drehwinkel für regelmäßige Vielecke: An jeder Ecke dreht die Turtle um 360° geteilt durch die Zahl der Ecken: `t.right(360 / n)`.

Figur	Dreieck	Quadrat	Fünfeck	Sechseck	Achteck	Stern (5 Zacken)
Ecken × Drehung	3 × 120°	4 × 90°	5 × 72°	6 × 60°	8 × 45°	5 × 144°

KOPIERVORLAGE C

Turtle-Koordinaten

(0, 0) ist in der Mitte. x wächst nach rechts, y wächst nach **oben**. Die Turtle startet hier und blickt nach oben. Ein Kästchen = 10 Pixel.



KOPIERVORLAGE D

Farben

Das Modul `pbc` kennt elf fertige Farben. Mit `pbc.rgb(rot, gruen, blau)` mischst du jede Farbe selbst: Jeder Wert geht von **0** (gar nichts) bis **255** (ganz viel).

Farbe	Konstante	Rot	Grün	Blau	Hinweis
	`pbc.WHITE`	255	255	255	alles voll
	`pbc.BLACK`	0	0	0	alles aus
	`pbc.RED`	255	0	0	
	`pbc.GREEN`	0	255	0	
	`pbc.BLUE`	0	0	255	
	`pbc.YELLOW`	255	255	0	rot + grün
	`pbc.CYAN`	0	255	255	grün + blau
	`pbc.MAGENTA`	255	0	255	rot + blau
	`pbc.GRAY`	128	128	128	alles halb
	`pbc.ORANGE`	255	164	0	
	`pbc.PURPLE`	120	0	120	lila

Zum Ausprobieren – gemischte Farben mit `pbc.rgb`:

Farbe	Idee	Rot	Grün	Blau	Farbe	Idee	Rot	Grün	Blau
	braun	139	69	19		rosa	255	192	203
	dunkelgrün	0	100	0		himmelblau	135	206	235
	dunkelblau	0	0	128		gold	255	215	0

> **Für Lehrkräfte – Kollisionsfarben in Bildern:** `sprite.on_color(farbe)` und `turtle.on_color(farbe)` vergleichen exakt. Wer Wände oder Lava in einem Bildprogramm malt, verwendet genau die Werte aus der oberen Tabelle und zeichnet ohne Kantenglättung. Durchsichtig sind in PNG-Bildern die Pixel mit Transparenz (Alpha-Kanal), nicht eine bestimmte Farbe.

KOPIERVORLAGE E · SEITE 1

Befehlskarte: Modul `pbc`

Befehl	Bedeutung
`import pbc`	ganz oben in jedem Programm
`pbc.clear()` / `pbc.clear(farbe)`	Bildschirm schwarz / in Farbe füllen
`pbc.text12("Hi", x, y, farbe)`	Text; auch `text18`, `text24`, `text` (8×8)
`pbc.text12_width("Hi")`	Breite des Texts in Pixeln (auch 18, 24)
`pbc.pixel(x, y, farbe)`	ein Pixel
`pbc.line(x1, y1, x2, y2, farbe)`	Linie
`pbc.rect(x, y, b, h, farbe)` / `pbc.fill_rect(…)`	Rechteck (Rand / gefüllt), (x, y) = links oben
`pbc.ellipse(mx, my, rx, ry, farbe)` / `…, fill=True)`	Ellipse um den Mittelpunkt
`pbc.rgb(r, g, b)`	eigene Farbe (je 0–255)
`pbc.WIDTH`, `pbc.HEIGHT`	240, 280
`pbc.pressed_a()` … `_b`, `_up`, `_down`, `_left`, `_right`, `_center`	`True`, **solange** gedrückt
`pbc.was_pressed_a()` …	`True` **einmal** pro Druck
`pbc.wait_for_button()`	warten, bis eine Taste gedrückt wurde
`pbc.led_red(True)` … `led_yellow`, `led_green`, `pbc.leds_off()`	Status-LEDs
`pbc.rgb_led(r, g, b)` / `pbc.rgb_led_off()`	RGB-LED
`pbc.tone(440)` / `pbc.tone(440, 200)` / `pbc.tone_off()`	Ton (Hz), ggf. mit Dauer in ms; wartet nicht
`pbc.accel_x()`, `accel_y()`, `accel_z()`	Bewegungssensor in m/s² (Emulator: 0, 0, 9.81)
`pbc.delay(ms)`	warten
`c = pbc.Canvas(240, 280)`	Bild im Speicher; `c.fill`, `c.text12`, `c.fill_rect` … wie oben; gefüllt: `c.ellipse(…, farbe, True)` (ohne `fill=`)
`pbc.show(c)`	Canvas anzeigen

Befehl	Bedeutung
`pbc.load_image("bild.png")`	Bild laden (Canvas)
`s = pbc.sprite("bild.png")`	Sprite (unsichtbar bei 0, 0)
`s.show()`, `s.hide()`, `s.set_to(x, y)`, `s.set_by(dx, dy)`	zeigen, verstecken, setzen, verschieben
`s.glide_to(x, y, ms)`, `s.x`, `s.y`, `s.width`, `s.height`	gleiten, Position und Größe lesen
`s.touches(t)`, `s.touches_edge()`, `s.on_color(farbe)`	Kollisionen
`pbc.Sprite._background = c`	Canvas als Hintergrund hinter den Sprites

Befehlskarte: `turtle`, `time`, `random`

Befehl	Bedeutung
`import turtle` / `t = turtle.Turtle()`	neue Turtle, Bildschirm schwarz, Blick nach oben
`t.forward(d)` / `t.back(d)`	vorwärts / rückwärts (Pixel)
`t.left(w)` / `t.right(w)`	drehen (Grad)
`t.setDirection(w)`	Richtung setzen: 0 oben, 90 rechts
`t.setPosition(x, y)` / `t.home()`	springen (ohne zu zeichnen) / zur Mitte
`t.moveTo(x, y)`	hinlaufen (zeichnet mit)
`t.penUp()` / `t.penDown()`	Stift heben / senken
`t.setPenColor(pbc.RED)` oder `"red"`	Stiftfarbe
`t.speed(0)` … `t.speed(10)`	0 = sofort, 1 langsam, 10 schnell
`t.hide()` / `t.show()` / `t.clear()`	Turtle verstecken/zeigen, Bild löschen
`t.on_color(farbe)`	Ist kurz vor der Turtle diese Farbe?
`import time`	
`time.sleep(1)` / `time.sleep_ms(200)`	warten (Sekunden / Millisekunden)
`time.ticks_ms()`	Millisekunden seit dem Start
`time.ticks_diff(neu, alt)`	Zeitunterschied in ms
`import random`	
`random.randint(1, 6)`	ganze Zufallszahl von 1 bis 6 (beide möglich)
`random.choice(liste)`	zufälliges Element

KOPIERVORLAGE F

Fehlermeldungen lesen

So gehst du vor: Lies die Meldung im Terminal **von unten nach oben**. Die letzte Zeile nennt die Fehlerart, darüber steht `line …`: die Zeile im Editor. Korrigiere immer nur den **ersten** Fehler und teste neu.

Fehlerart	Was Python sagen will	Typische Ursachen
`SyntaxError`	„Diese Zeile kann ich nicht lesen." Nichts läuft.	fehlende Klammer, fehlendes `"` oder Komma, `:` fehlt nach `if` / `for` / `while` / `def`; Fehler oft in der Zeile **davor**
`IndentationError`	„Die Einrückung stimmt nicht."	nach `:` nicht eingerückt; Zeilen eines Blocks unterschiedlich weit eingerückt
`NameError`	„Diesen Namen kenne ich nicht."	Tippfehler, `import` vergessen, Variable vor dem ersten Zuweisen benutzt
`AttributeError`	„Das Modul/Objekt hat das nicht."	`pbc.Text12` statt `pbc.text12`, `pbc.white` statt `pbc.WHITE`
`TypeError`	„Diese Werte passen nicht zusammen."	`"Punkte: " + 5` (Text + Zahl), falsche Anzahl Argumente
`ValueError`	„Mit diesem Wert geht das nicht."	`int("zwölf")`
`IndexError`	„Diesen Platz gibt es in der Liste nicht."	Index zu groß; Zählung beginnt bei 0
`ZeroDivisionError`	„Durch null kann ich nicht teilen."	Teiler ist 0
`OSError` (ENOENT)	„Diese Datei gibt es nicht."	Dateiname falsch, Datei noch nicht geschrieben

Keine Fehlermeldung, aber falsches Ergebnis? Füge `print(...)` ein und lass dir Zwischenwerte ausgeben. Prüfe: Läuft die Zeile überhaupt? Haben die Variablen die Werte, die ich erwarte?

Projekt-Planungsbogen: Mein Programm

Namen: _____ Klasse: _____ Datum: _____

1. Name und Idee

Mein Programm heißt: _____

Darum geht es (2–3 Sätze):

2. So sieht es aus

Bildschirm 240 × 280

3. Figuren, Bilder, Daten

Name im Programm	Was ist das?	Bild / Datei

4. Bedienung

Taste	Was passiert?
Joystick	
MITTE	
A	
B	

5. Regeln und Ablauf

Start:	
Punkte / Ergebnis:	
Ende:	
Speichern (Datei)?	

6. Variablen und Listen

Name	Typ (int, str, list …)	Was wird gespeichert?	Startwert

7. Funktionen

`def` …(Parameter)	Was macht sie? Was gibt sie zurück?	Wer?	fertig
			☐
			☐
			☐
			☐

8. Arbeitsplan

Stunde	Das wollen wir schaffen (lauffähige Stufe)	geschafft
1		☐
2		☐
3		☐
4		☐
5		☐

9. Testprotokoll

Beobachtung / Fehlermeldung	Ursache	Lösung

Tipp: Speichere am Ende jeder Stunde mit **Dateien → Auf den Computer…** unter einem Namen mit der Stunde, z. B. `spiel_std2.py`.

KOPIERVORLAGE H

Tasten am PicoBoy und im Emulator



Am Gerät	In Python	Im Emulator (Tastatur)
Joystick nach oben	`pbc.pressed_up()`	Pfeil nach oben ↑
Joystick nach unten	`pbc.pressed_down()`	Pfeil nach unten ↓
Joystick nach links	`pbc.pressed_left()`	Pfeil nach links ←
Joystick nach rechts	`pbc.pressed_right()`	Pfeil nach rechts →
Joystick drücken (MITTE)	`pbc.pressed_center()`	Leertaste
Taste A	`pbc.pressed_a()`	Taste **A**
Taste B	`pbc.pressed_b()`	Taste **B**

Im Emulator kannst du die Knöpfe auch mit der Maus anklicken.

`pressed_…` oder `was_pressed_…`? `pressed_…()` ist `True`, **solange** die Taste unten ist – gut zum Bewegen. `was_pressed_…()` meldet jeden Druck **genau einmal** – gut zum Zählen oder Umschalten.

In PBC+ Studio	
Starten / anhalten	**▶ Testen** / **■ Stop**
Rückgängig im Editor	**Strg + Z**
Alles markieren	**Strg + A**
Sichern	**Dateien → Auf den Computer…**