WHZ Westsächsische
Hochschule Zwickau
University of Applied Sciences

(Tool-)Freiheit im E-Assessment. Weniger ist mehr!

Jens Flemming

jens.flemming@fh-zwickau.de

Westsächsische Hochschule Zwickau

Netzwerktreffen Mathematik/Physik und E-Learning, HS Mittweida, 6. März 2025

Worum geht es?

Stecken viel Zeit in Tool-Entwicklung und -Integration:

MAXIMA, JSXGraph, sStatics, JSME, ALADIN, PyRope, Nbgrader, STACK, PhySTACK, MATLAB Grader, LTI, ...

Werden wir eines Tages am Ziel sein?

Anforderungen und Wünsche an die E-Assessment-Umgebung nehmen ständig zu!

inhaltliche Möglichkeiten, Versionierung, kollaboratives Arbeiten, OER, Mehrsprachigkeit, Portierbarkeit, einfaches Erstellen, Abspielmodi (freies Üben vs. Prüfung), ...

Sind wir auf dem richtigen Weg?

Heute: ein alternativer Weg

Zielgruppe: Entwickler komplexer E-Assessment-Aufgaben für MINT

Wichtig: keine finalen Antworten, sondern Ideen, Anregungen, erste Prototypen

Beispiel 1: Briefwaage

Finde die Abbildung von Masse zu Winkel!

Wir hätten gern:

- Visualisierung des Problems
- Randomisierung
- CAS (Lösung ist eine Funktion)
- Lösung visualisieren
- interaktiv Gewichte auflegen
- bei Bedarf Hinweise zum Vorgehen
- Feedback (was ist falsch an der Lösung)
- „weiche“ Bewertung (Abweichung zum Soll)

Diverse andere komplexe Aufgabenideen für Ingenieurmathe liegen auf Halde **mangels Umsetzbarkeit** (und Zeit)!

Projekt ist beantragt...

Beispiel 2: Quadratische Gleichung

Jochen Merker, Konrad Schöbel (23. NWT):
„Quadratische Gleichung - Nichts leichter als das!?“

Hauptproblem: Wollen Lösungsanzahl nicht vorgeben. Wie Eingabe organisieren?

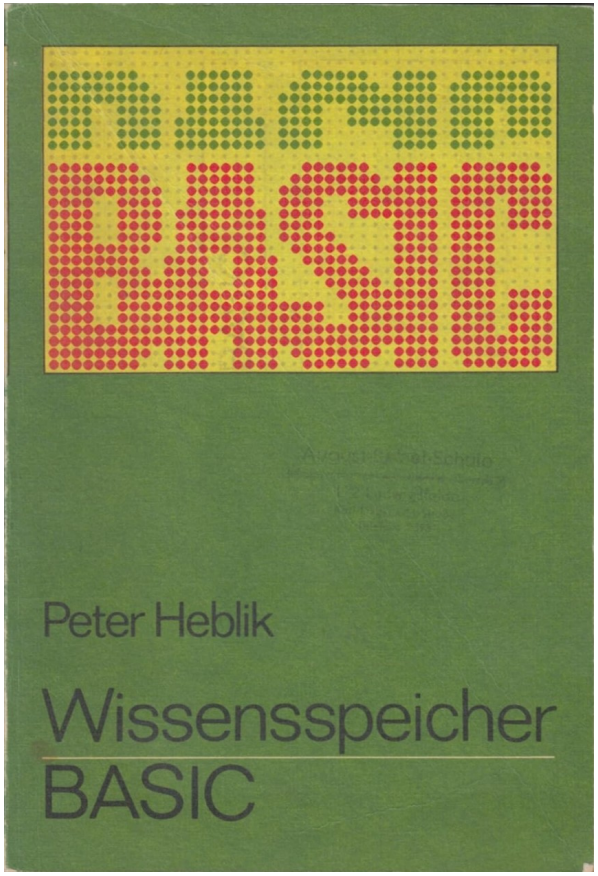
Auch PyRope hat Grenzen! (aber sehr wenige)

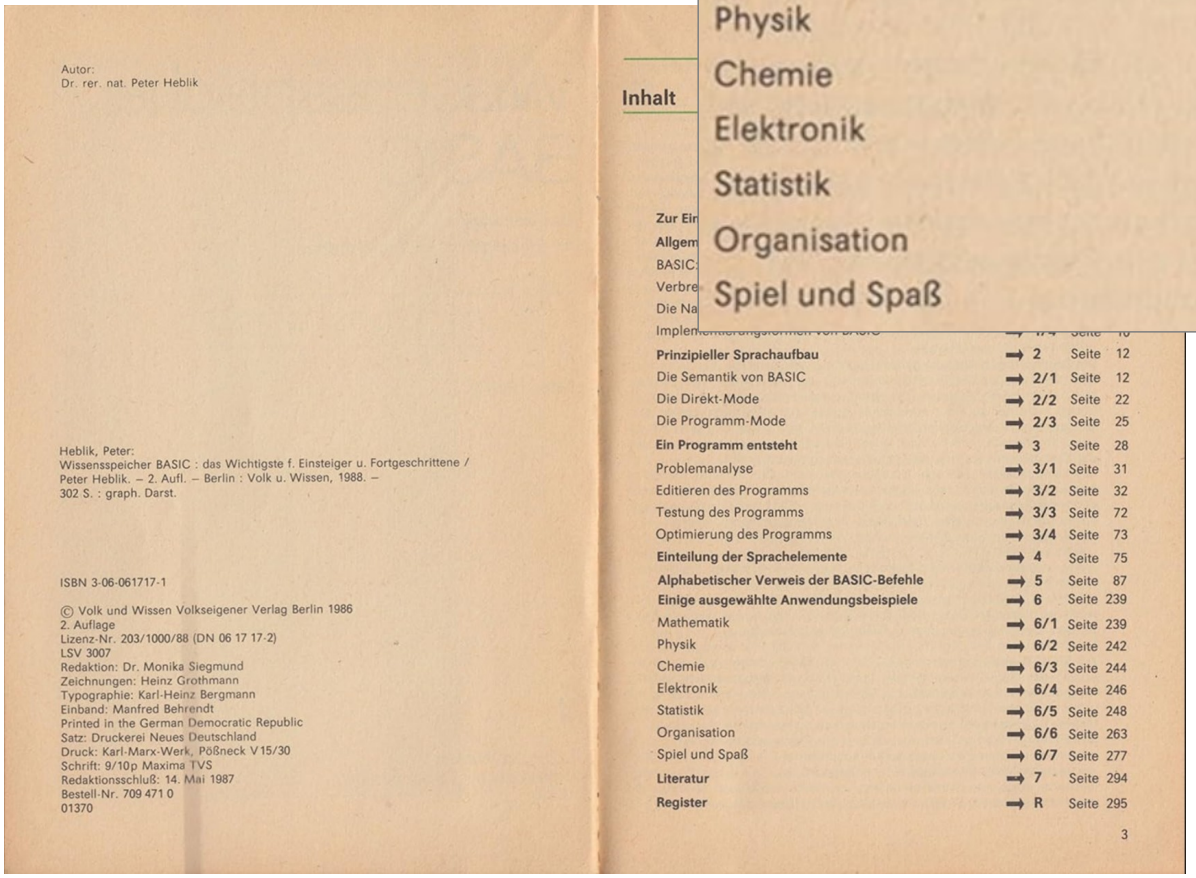
Sind wir auf dem richtigen Weg, wenn wir selbst solche Standardaufgaben nicht zufriedenstellend in E-Assessments abbilden können?

Was wäre denn zufriedenstellend?
- Lerner wählt Anzahl der Lösungen (einschließlich „keine“ und „unendlich viele“)
- Lerner tippt jede Lösung in ein Eingabefeld

Nicht:
- Syntax für Listen/Mengen
- Lösungsvarianten ausschließen
- je Lösungsanzahl eine separate Aufgabe bauen

Back to the Roots




Autor:
Dr. rer. nat. Peter Heblik

Heblik, Peter:
Wissensspeicher BASIC : das Wichtigste f. Einsteiger u. Fortgeschrittene /
Peter Heblik. – 2. Aufl. – Berlin : Volk u. Wissen, 1988. –
302 S. : graph. Darst.

ISBN 3-06-061717-1

© Volk und Wissen Volkseigener Verlag Berlin 1986
2. Auflage
Lizenz-Nr. 203/1000/88 (DN 06 17 17-2)
LSV 3007
Redaktion: Dr. Monika Siegmund
Zeichnungen: Heinz Grothmann
Typographie: Karl-Heinz Bergmann
Einband: Manfred Behrendt
Printed in the German Democratic Republic
Satz: Druckerei Neues Deutschland
Druck: Karl-Marx-Werk, Pößneck V 15/30
Schrift: 9/10p Maxima TVS
Redaktionsschluß: 14. Mai 1987
Bestell-Nr. 709 471 0
01370

Inhalt

Prinzipieller Sprachaufbau → 2 Seite 12
Die Semantik von BASIC → 2/1 Seite 12
Die Direkt-Mode → 2/2 Seite 22
Die Programm-Mode → 2/3 Seite 25
Ein Programm entsteht → 3 Seite 28
Problemanalyse → 3/1 Seite 31
Editieren des Programms → 3/2 Seite 32
Testung des Programms → 3/3 Seite 72
Optimierung des Programms → 3/4 Seite 73
Einteilung der Sprachelemente → 4 Seite 75
Alphabetischer Verweis der BASIC-Befehle → 5 Seite 87
Einige ausgewählte Anwendungsbeispiele → 6 Seite 239
Mathematik → 6/1 Seite 239
Physik → 6/2 Seite 242
Chemie → 6/3 Seite 244
Elektronik → 6/4 Seite 246
Statistik → 6/5 Seite 248
Organisation → 6/6 Seite 263
Spiel und Spaß → 6/7 Seite 277
Literatur → 7 Seite 294
Register → R Seite 295

3

Freie Programmierung! Einfache Sprache! Kein Schnickschnack!

Minimalanforderungen

Was brauchen wir für Aufgabenbau und Abspielen von Aufgaben?

ganz grundsätzlich:

- ein Stück **Bildschirm**
- Zugriff auf **Tastatur und Maus** (jederzeit, nicht nur Eingabefelder und „OK“-Buttons)
- eine allgemeine **Programmiersprache** (kein MAXIMA, MATLAB oder ähnliches)

Zusatzforderungen, damit das praktisch einsetzbar wird:

- Aus- und Eingabe muss im Webbrowser laufen
- Programm wird nicht im Browser ausgeführt (kein ungewollter Eingriff durch Nutzer!)
- leicht integrierbar in existierende Lernumgebungen (Websites, LMS, ...)
- kein Admin-Overhead für Aufgabenersteller und -löser (Software-Installation, ...)

Das sind alles **administrative Forderungen**, mit denen der Aufgabenersteller nichts zu tun haben sollte!

Proof of Concept

Thebe/Voila bringt Python in den Browser ohne GUI und sonstigen Schnickschnack.

github.com/jupyter-book/thebe
github.com/voila-dashboards/voila

OPAL-Kurs „Jupyter goes OPAL“

> „Einschreiben“
> „Ausprobieren“
> „Thebe-Demo-Aufgaben“
> „Quadratische Gleichungen“

Demo…

So viel Code?

200 Zeilen Python-Code (plus 60 Leerzeilen/Kommentare), davon:

- **60 Zeilen randomisierte Aufgabe** erstellen
    - zufälliger Lösungstyp mit unterschiedlichen Wahrscheinlichkeiten
    - zufällige ganzzahlige Lösungen in vorgegebenem Bereich
    - Koeffizienten berechnen
    - LaTeX-Code für Darstellung (einschließlich „-x“ statt „-1 x“ usw.)
    - Aufgabentext
    - Ausgabe
- **90 Zeilen Bewertung und Feedback** generieren
    - 25 Fälle zu je 3 Zeilen (Fall, Punktzahl, Feedback-Text)
    - „Preprocessing“ (z.B. Lösungen „2, 2“ zu „2“ umwandeln)
    - Ausgabe (Punktzahl, Feedback-Text, Farbe je nach Punktzahl)
- **40 Zeilen GUI** (Slider, Eingabefelder, Radiobuttons)
- 10 Zeilen Klebecode (Funktionsdefinitionen, …)

Vorteile

Wenig Code für viel Funktion dank „Python-Universum“:
- komplexe GUIs (ipywidgets)
- Computeralgebra (SymPy)
- MATLAB-like Matrixoperationen (NumPy)
- Plots in 2D und 3D (Matplotlib, Plotly, ...)

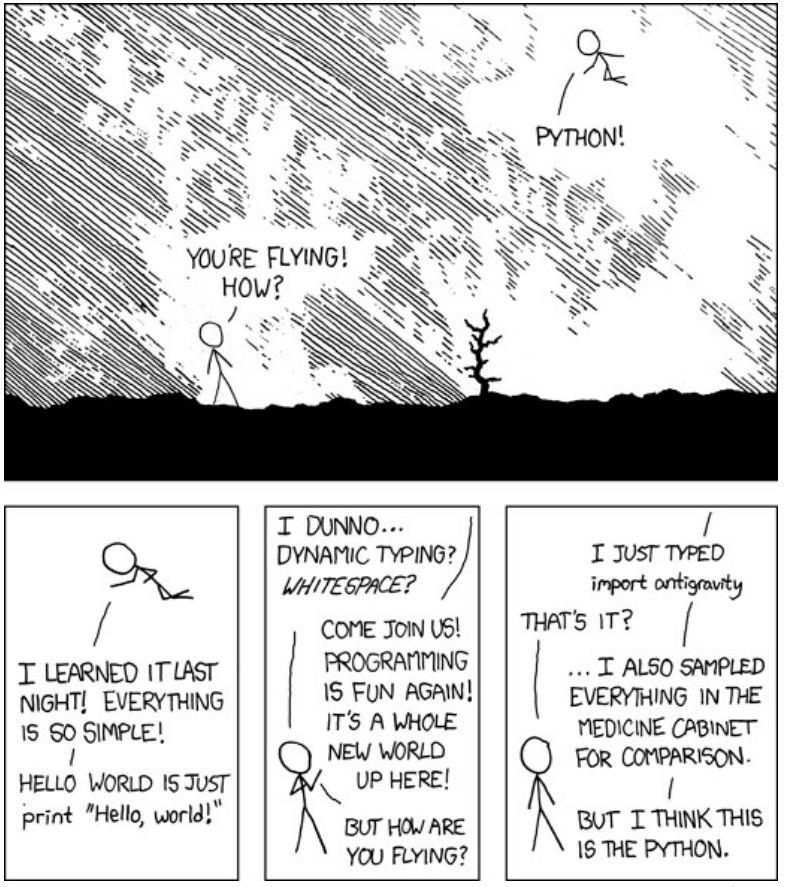
einfache, menschenfreundliche Sprache:
- lesbare Syntax
- einfache, klare Semantik (von Mathematiker...)
- frei und open-source

Portierbarkeit:
- Aufgabe kann lokal entwickelt und getestet werden
- Aufgabe auch im Browser ohne Server (JupyterLite)

Versionierbarkeit (Textdatei)

Mehrsprachigkeit (Lösungen aus der Software-Welt direkt nutzbar)

nicht nur Python, sondern fast jede Sprache nutzbar



Randall Munroe, xkcd.com/353

Nachteile

Python lernen (kein „Zusammenklicken“ von Aufgaben)

Jemand muss den Jupyter-Server aufsetzen und verwalten.

Zu viel Freiheit?

- Testablaufsteuerung selbst bauen oder einzelne Aufgaben in LMS-Tests integrieren?
- Testergebnisse selbst verwalten oder an LMS schicken?
- Content-Auswahl über eigene Website oder LMS?
- …

Neue Möglichkeiten

einfache **Kombination** verschiedener Tools zu komplexen Aufgaben/Tests:
- JSXGraph, JSME und andere können in Jupyter-Umgebungen integriert werden
- PyRope-Aufgaben
- Einbinden in ONYX-Tests (div-Box + Lücke + JavaScript)?

Learning-Analytics (Nutzerverhalten beim Lösen genau beobachtbar)
Timing bei Eingaben, Korrekturen vor dem Abgeben, ...

freier ONYX-Player auf Python-Basis?
ONYX = HTML/JavaScript + LaTeX + MAXIMA + Sandbox + Aufgaben-XML
Thebe = HTML/JavaScript + LaTeX + Python + Sandbox + tausende Bibliotheken
Pythons SymPy interpretiert auch MAXIMA-Code!

NW-Aufgabenpool auch außerhalb OPAL/ONYX nutzbar machen?

selbst rumspielen:
lokal: whz.de/~jef19jdw/blog/thebe-core.html
Server: mich fragen