PROFESSOR ERKLÄRBÄR

Serie Informatik · Folge 7 · Advanced

INFORMATIK

LEHRERINFO ZUM VIDEO

Vom Quelltext zur Maschine

FACH	STUFE	DAUER	VIDEO
Informatik	Gymnasium · Sek II	13 min	youtube.com/@Erklaerbaer

01 Lernziele

Nach dem Video können die Schülerinnen und Schüler …

- Maschinencode, Assembler und Hochsprache unterscheiden und einen Befehl über alle drei Ebenen verfolgen,
- Compiler, Interpreter und Bytecode mit virtueller Maschine erklären und nach Tempo, Plattform und Fehlerzeitpunkt vergleichen,
- die Phasen einer Übersetzung beschreiben: Lexer (Tokens), Parser (Syntaxbaum nach einer Grammatik), Codeerzeugung,
- eine einfache Grammatik in EBNF lesen und Syntaxbäume für Rechenausdrücke zeichnen und auswerten,
- Syntax und Semantik unterscheiden sowie statische/dynamische und starke/schwache Typisierung vergleichen,
- die Paradigmen imperativ, deklarativ, objektorientiert und funktional einordnen.

02 Voraussetzungen

Variablen, Datentypen und Fehlermeldungen in Python (Folge 7 · Grundlagen); Maschinenbefehle und Befehlszyklus der Mini-CPU (Folge 5 · Advanced); Hexadezimalzahlen (Folge 2 · Advanced).

03 Ablauf des Videos

ZEIT	KAPITEL	INHALT UND BEISPIEL
0:00	Einstieg	Der Prozessor versteht kein Python · Ident · Fahrplan Text → Tokens → Baum → Bytecode
0:31	Was du brauchst	Variablen, Datentypen (07G) · Maschinenbefehle (05A) · Hex (02A)
0:55	Drei Sprachebenen	total = a + b → LADE/ADDIERE/SPEICHERE → 1E 2F 3D · Grace Hopper

1:39	Compiler oder Interpreter?	C, C++, Rust gegen Interpreter · Vor- und Nachteile · Python-Bytecode · Java JVM + JIT PAUSE-FRAGE Fehlerzeitpunkt · Merksatz
3:17	Schritt eins: Tokens	Lexer zerlegt «total = price * 3» · 17 Zeichen → 5 Tokens
3:52	Schritt zwei: Grammatik und Syntaxbaum	EBNF für Ausdruck/Term/Faktor · Bäume für 2 + 3 · 4 und (2 + 3) · 4 · Syntaxfehler PAUSE-FRAGE 8 − 2 − 3 · Merksatz
5:26	Schritt drei: Code erzeugen	dis.dis in Python 3.12 · Vergleich Mini-CPU · Konstantenfaltung 2 + 3 * 4 → 14
6:04	Syntax, Semantik, Typen	Logik- und Laufzeitfehler · Java statisch, Python dynamisch · "3" + 4 in Python und JavaScript · Merksatz
7:32	Programmierstile	imperativ, deklarativ (SQL), objektorientiert, funktional (map)
8:26	Projekt-Baustein	Lexer + Grammatik für den Punkt-vor-Strich-Rechner (15A) · Assembler von Hand: LADE 14 → 1E (17A)
8:48	Prüfungsfallen	fünf Fallen (siehe 04)
9:41	Zusammenfassung	Kernaussagen · zurück zum Einstieg
10:24	Quiz	4 Fragen · Endkarte Folge 8 Advanced

04 Prüfungsfallen

- Python ist eine reine Interpretersprache – Python übersetzt zuerst in Bytecode, den eine virtuelle Maschine ausführt.
- Ein Programm ohne Syntaxfehler ist korrekt – Logik- und Laufzeitfehler bleiben; testen!
- Kompilierte Programme laufen überall – Maschinencode ist an Prozessor und Betriebssystem gebunden.
- Dynamische Typisierung heisst: keine Typen – jeder Wert hat einen Typ; geprüft wird erst zur Laufzeit.
- Der Parser rechnet das Ergebnis aus – er baut nur den Baum; gerechnet wird bei der Ausführung (Ausnahme: Konstantenfaltung durch den Übersetzer).

05 Einsatz im Unterricht

Einstieg: `import dis; dis.dis("x = a + b")` in Thonny ausführen und die Ausgabe gemeinsam «übersetzen».

Lexer von Hand: Ausdrücke in Tokens zerlegen (Kärtchen: NAME, ZAHL, OPERATOR, KLAMMER) und zu Bäumen legen.

Vergleich: Dasselbe kleine Programm in Python und Java tippen; Typfehler einbauen und vergleichen, wann sie auffallen.

Prüfungsvorbereitung: Aufgaben 2, 3 und 5 sind im Stil schriftlicher Prüfungen gestellt.

Hinweis: Die Bytecode-Ausgabe stammt aus Python 3.12 (vereinfacht ohne Offsets und RESUME/RETURN); andere Versionen benennen die Befehle teils anders. Die Hex-Bytes 1E 2F 3D beziehen sich auf die Mini-CPU

der Serie (4 Bit Befehl + 4 Bit Adresse, LADE = 1, ADDIERE = 2, SPEICHERE = 3). Grace Hopper entwickelte 1952 den A-0-Compiler.

06 Quellen

- Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman: „Compilers: Principles, Techniques, and Tools“, 2. Aufl., Pearson (2006)
- Python-Dokumentation: dis – Disassembler for Python bytecode (Version 3.12) · The Python Language Reference, Kapitel 2 (Lexical analysis)
- ISO/IEC 14977: Extended BNF (1996)
- Tim Lindholm u. a.: „The Java Virtual Machine Specification“, Oracle
- Kurt W. Beyer: „Grace Hopper and the Invention of the Information Age“, MIT Press (2009)
- Gesellschaft für Informatik: Bildungsstandards Informatik für die Sekundarstufe II (2016)

NAME _____ KLASSE _____ DATUM _____

Vom Quelltext zur Maschine

1 Tokens ●○○

Zerlege `preis = (netto + 5) * 1.08` in Tokens und **benenne** ihre Art (Name, Zuweisung, Operator, Zahl, Klammer). Wie viele Tokens sind es?

2 Syntaxbäume ●●○

Zeichne nach der Grammatik aus dem Video die Syntaxbäume und **werte** sie **aus**: a) 6 + 4 / 2 b) (6 + 4) / 2 c) 20 / 2 / 5

3 Grammatik erweitern ●●●

Die Grammatik soll auch Potenzen `**` kennen, die stärker binden als Mal und Geteilt. **Ergänze** eine Regel und **begründe**, an welcher Stelle sie stehen muss.

4 Compiler oder Interpreter? ●○○

Ordne zu und **begründe**: a) Ein Spiel soll auf einer Konsole möglichst schnell laufen. b) Ein Skript wird auf Windows, Mac und Linux ohne Änderung genutzt. c) Der Fehler in Zeile 80 fällt erst auf, als Zeile 80 läuft.

5 Fehlerarten und Typen ●●○

Bestimme die Fehlerart (Syntax-, Laufzeit-, Logikfehler): a) `if x > 3 print(x)` b) `mittel = a + b / 2` c) `zahl = int(input())` bei der Eingabe «zwölf». d) Was liefert `"5" * 3` in Python, was `"5" - 3` in JavaScript?

6 Paradigmen ●○○

Ordne jedem Paradigma ein Beispiel **zu**: «SELECT name FROM schueler», «for i in range(10): …», «list(map(len, woerter))», «konto.einzahlen(50)».

Lösungen

1 Tokens

preis (Name) · = (Zuweisung) · ((Klammer) · netto (Name) · + (Operator) · 5 (Zahl) ·) (Klammer) · * (Operator) · 1.08 (Zahl) → 9 Tokens.

2 Syntaxbäume

a) Wurzel +, links 6, rechts / mit 4 und 2 → 6 + 2 = 8. b) Wurzel /, links + mit 6 und 4, rechts 2 → 10 / 2 = 5. c) Linksassoziativ: Wurzel /, links / mit 20 und 2, rechts 5 → 10 / 5 = 2 (nicht 20 / 0,4 = 50).

3 Grammatik erweitern

Term = Potenz { ("*" | "/") Potenz } · Potenz = Faktor ["**" Potenz] · Faktor = Zahl | "(" Ausdruck ")". Je tiefer eine Regel in der Kette steht, desto stärker bindet ihr Operator; Potenz kommt also zwischen Term und Faktor. Die rechte Seite «Potenz» macht ** rechtsassoziativ wie in Python (2 ** 3 ** 2 = 2 ** 9).

4 Compiler oder Interpreter?

a) Compiler – Maschinencode ist am schnellsten. b) Interpreter (oder Bytecode + VM) – läuft überall, wo der Interpreter installiert ist. c) Interpreter – ein Compiler hätte den Fehler beim Übersetzen gemeldet (sofern er ihn erkennen kann).

5 Fehlerarten und Typen

a) Syntaxfehler (Doppelpunkt fehlt) · b) Logikfehler (Punkt vor Strich, richtig: (a + b) / 2) · c) Laufzeitfehler (ValueError) · d) Python: "555" (Text wird wiederholt); JavaScript: 2 (der Text wird stillschweigend in eine Zahl umgewandelt – schwache Typisierung).

6 Paradigmen

SELECT … → deklarativ · for-Schleife → imperativ · map → funktional · konto.einzahlen → objektorientiert

Q Quiz im Video

1 B (Tokens) · 2 C (Bytecode in einer virtuellen Maschine) · 3 A (die Struktur nach der Grammatik) · 4 B (deklarativ)