universität wien

MmF

PH NÖ PÄDAGOGISCHE HOCHSCHULE NIEDERÖSTERREICH

Lehrkonzept Programmieren

Stand: 31.07.2022

1. Lernziele

1.1. Verwendung von Programmiersprachen

(APP1) Die Lernenden können den Programmablauf als Hintereinanderausführung von Befehlen erklären.

(APP2) Die Lernenden können Programmiersprachen für elementare Berechnungen verwenden.

(APP3) Die Lernenden können das Prinzip *Eingabe – Verarbeitung – Ausgabe* bewusst verwenden.

(APP4) Die Lernenden können Programme für elementare praktische Anwendungen erstellen.

1.2. Programmelemente und Programmstrukturen

(STR1) Die Lernenden können Variablen anlegen (deklarieren) und mit Werten belegen (initialisieren).

(STR2) Die Lernenden können elementare Datentypen (Integer, Fließkommazahlen, Zeichen, Zeichenketten, Boole'sche Werte) und deren Funktion nennen.

(STR3) Die Lernenden können Vergleichsoperatoren einsetzen.

(STR4) Die Lernenden können Ablaufstrukturen (bedingte Anweisungen, Schleifen) verwenden.

(STR5) Die Lernenden können elementare Datenstrukturen (eindimensionale Listen) verwenden.

(STR6) Die Lernenden können Funktionen erstellen.

1.3. Funktionalität und Fehlerbehandlung

(FCT1) Die Lernenden können feststellen, ob Variablen geeignete Datentypen besitzen.

(FCT2) Die Lernenden können den Nutzen von Schleifen erläutern.

(FCT3) Die Lernenden können Grenzen erläutern, die sich aus den Wertebereichen von Datentypen ergeben.

(FCT4) Die Lernenden können erläutern, warum aufgrund von falschen Datentypen der Programmablauf fehlerhaft sein kann.

(FCT5) Die Lernenden können Fehler behandeln, die sich aufgrund der Geltungsbereiche von Variablen ergeben.

(FCT6) Die Lernenden können Fehler behandeln, die sich aufgrund inkompatibler Datentypen von Vergleichsoperanden ergeben.

(FCT7) Die Lernenden können Fehler behandeln, die sich aus den Indexbereichen von Datenstrukturen ergeben.

2. Grobplanung

1 Einheit = 45 Minuten; Erweiterungen bzw. Inhalte für Fortgeschrittene in rot

Einheit	Inhalte	Lernziele
1-2	• Erkunden einer graphischen Programmierumgebung (z. B. Scratch, mblock 5) • Herstellen von Referenzpunkten in einer graphischen Programmiersprache für späteres textbasiertes Programmieren (*Eingabe – Verarbeitung – Ausgabe*, Variablen, arithmetische Operatoren, Vergleichsoperatoren, Ablaufstrukturen, Funktionen)	
3-4	Konzepte: Operationen (Operand, Operator), *Eingabe – Verarbeitung – Ausgabe* • Kennenlernen einer textbasierten Programmierumgebung (z. B. Python-Kommandozeileninterface) • Verwenden der arithmetischen Operatoren `+`, `-`, `*`, `/`, `**`, `//`, `%` • Verwenden der Vergleichsoperatoren `<`, `<=`, `==`, `>=`, `>`, `!=` • Veranschaulichen des Prinzips *Eingabe – Verarbeitung – Ausgabe* anhand von arithmetischen Operationen, Vergleichsoperationen und der Methode `print()`	
5	Konzepte: Datentypen, Variablen, Kommentare • Speichern von Ergebnissen arithmetischer Operationen in Variablen • Zuweisungsoperator `=` • Datentypen `int`, `float` • Werte von Variablen mit der Methode `print()` ausgeben • Code zur besseren Verständlichkeit kommentieren	
6-7	Konzepte: Datentypen, Variablen, Indizes • Speichern von Zeichenketten in Variablen • Datentyp `str` • Operatoren für Zeichenketten `+`, `*` • Bereichsoperatoren `<str>[<int>]`, `<str>[<int>:<int>]` • Funktionen für Zeichenketten `print()`, `len()`	
8	Konzepte: Eingabe, Fehlerbehandlung • Funktion `input(<str>)` • Wichtigkeit von Datentypen erkennen: Zahlen werden bei Eingabe als Zeichenkette übergeben • Typumwandlung mittels `int(<str>)` bzw. `str(<int>)` • Interpretieren von Fehlermeldungen • Datentypen mit der Methode `type()` abfragen	
9	Konzepte: Bedingte Anweisung, Datentypen, Programmstruktur, Vergleichsoperatoren • `if` – `elif` – `else` • Struktur durch Einzüge • Auswerten der Bedingung für das Ausführen einer bedingten Anweisung (Ergebnis der Auswertung `True`/`False`) • Datentyp `bool`	
10	Konzepte: Schleifen • Schleife mit `while`	
11	Konzepte: Schleifen	

	• Schleife mit `for` • Bereichsangaben mit `range(<int>)`	
12	Konzepte: Listen, Listenindex; Laufindex, Schleifen • Listen anlegen • Listenelemente adressieren • Elemente zu Listen hinzufügen • Elemente aus Listen entfernen • Funktion `len(<list>)` • Schleife mit `for <float/int/str> in <list>`	
13-14	Konzepte: Funktionen • Übergabeparameter • Rückgabewert • Reflexion: Nutzen von Funktionen	