**UV 10.1 Computerprogramme mit System entwickeln – Einstieg in die textorientierte Programmierung**

|  |  |
| --- | --- |
| **Leitfragen:**   * *Wie realisiert man einen Algorithmus in der Programmiersprache Python?* * *Wie realisiert man Fallunterscheidungen und Wiederholungen?* * *Wie strukturiert man Programme sinnvoll?* * *Welche Daten muss der Computer speichern?* * *Wie analysiert und testet man Programmteile?* | **Ausschärfung der Inhaltsschwerpunkte:**   * Gegenüberstellung von Software-Anwendung und Software-Entwicklung * Handlungsvorschriften formulieren und befolgen * Nutzung vorgegebener Methoden * Verwendung von Kontrollstrukturen, Variablen, Methoden und Parametern * Ein-und Ausgaben * Einsatz eines strukturierten Datentyps * Analyse und Erläuterung von Programmen * Überprüfung algorithmischer Eigenschaften * Problemangemessenheit eines Algorithmus |

**Vorhabenbezogene Konkretisierung:**

In diesem Unterrichtsvorhaben erlernen die Schülerinnen und Schüler Grundlagen der textorientierten Programmierung und Algorithmik.

Die Fachkonferenz hat sich auf die textorientierte Programmiersprache Python und die integrierte IDLE als Entwicklungsumgebung geeinigt.

Um den Übergang von der visuell-orientierten Programmierung zu erleichtern, wird zunächst die von Python zur Verfügung gestellte Turtlegrafik verwendet, da hier durch die grafischen Ausgaben die Funktion eines Programms sehr gut schrittweise nachvollzogen werden kann. Die Fachkonferenz entscheidet auf eine weitergehende Verwendung der Bibliothek TkInter zu verzichten, um eine zu große Nähe an ein Unterrichtsvorhaben in der Oberstufe des Joseph-Weizenbaum-Gymnasiums zu vermeiden.

Das Unterrichtsvorhaben knüpft an die vorangegangenen Unterrichtsvorhaben an, in denen die Schülerinnen und Schüler bereits mit Hilfe einer visuellen Programmierumgebung algorithmische Grundkonzepte kennengelernt haben.

Bei der Vorstellung der Programmierumgebung bietet es sich an, den Unterschied zwischen der Entwicklung und der Anwendung eines Computerprogramms anzusprechen.

Anhand von Turtle-Beispielprogrammen erarbeiten die Schülerinnen und Schüler die syntaktisch korrekte Formulierung von Python-Programmen. Mithilfe der Turtle-Befehlsdokumentation erweitern sie bestehende Programme und entwickeln eigene Programme. Die Bedeutung von Einrückungen wird praktisch untersucht.

Die Verwendung von Variablen wird im Kontext konkreter Anwendungen thematisiert.

Der Programmablauf von nicht-linearen Programmen wird mit Hilfe von Flussdiagrammen oder Struktogrammen verdeutlicht. Auch andersherum werden diese grafischen Darstellungen verwendet, um ein Programm oder eine Methode zu entwickeln. Die Schülerinnen und Schüler nutzen Methoden mit und ohne Parameter zur Strukturierung von Programmen. Unterschiede zwischen Bottom-Up- und der Top-Down-Vorgehensweisen können thematisiert werden.

Die Schülerinnen und Schüler entwickeln Programme, die Benutzereingaben einlesen, verarbeiten und Ausgaben erzeugen.

Dabei rücken auch die Fragen nach der Repräsentation von Informationen und geeigneten Datentypen in den Vordergrund, z.B., wenn eine Benutzereingabe als Zahl interpretiert wird.

Neben einfachen Datentypen wird ein strukturierter Datentyp (z.B. Liste, Array, Dictionary) für eine Anwendung zur Lösung einer Problemstellung verwendet.

Die Schülerinnen und Schüler entwickeln eigene Methoden und Programme. Darüber hinaus analysieren und beurteilen sie auch fremde Programme bzw. Programmteile.

**Zeitbedarf**: 20 Std.

**Sequenzierung des Unterrichtsvorhabens:**

|  |  |  |
| --- | --- | --- |
| **Unterrichtssequenzen** | **Zu entwickelnde (inhaltsfeldbezogene konkretisierte) Kompetenzen** | **Beispiele, Medien, Materialien** |
| Lösung von Aufgaben und Problemstellungen mit Hilfe der textbasierten Programmierung in Python:   * Vorstellung der Programmierumgebung: Eine Frage der Sichtweise: Nutzung einer Anwendung oder Erstellung eines Programms * Schleifen und Verzweigungen in Python: Verschiedene Fälle! -   Gleicher Text, andere Wirkung! - Einrücken oder nicht?   * Methodenkonzept: Strukturiere und verwende Teile wieder! * Variablen: Merk dir was! * Parameter: Gib der Methode eine Information mit! * Ein- und Ausgaben: Antworte mir! * Datentypen: Eine Typsache?   Repräsentation von Informationen: Wie sag ich’s dem Rechner bzw. was soll er sich merken?   * Entwurf von Algorithmen unter Verwendung von Methoden, Variablen und Kontrollstrukturen * Methoden mit Rückgabewert: Die Antwort der Methode * Verwendung eines strukturierten Datentyps: Merk dir nicht nur eine Sache! * Untersuchen und Beurteilen von Programmteilen: Was macht die Methode bzw. das Programm? | Die Schülerinnen und Schüler   * repräsentieren Information in natürlicher Sprache, formalsprachlich und graphisch (DI), * codieren Daten für die Verarbeitung mit einem Informatiksystem (DI), * interpretieren Ergebnisse eines Datenverarbeitungsprozesses(DI) * wählen geeignete Datentypen im Kontext eines Anwendungsbeispiels aus (MI), * verarbeiten Daten mithilfe von Informatiksystemen(MI), * verwenden logische und arithmetische Operationen (MI), * verarbeiten gleichartige Daten mit Hilfe eines geeigneten Werkzeugs(DI), * modellieren und implementieren eine Anwendung unter Verwendung eines strukturierten Datentyps in einer Programmiersprache (MI). * entwerfen Algorithmen unter Verwendung des Variablenkonzeptes und von Kontrollstrukturen (MI), * beschreiben den Unterschied zwischen der Bottom-Up- und der Top-Down-Methode (MI), * stellen Algorithmen in verschiedenen Repräsentationen dar (DI), * implementieren und kommentieren Algorithmen in einer textorientierten Programmierumgebung (MI), * strukturieren und zerlegen Algorithmen in Teilalgorithmen (MI), * modifizierenProgramme (MI). * beurteilen die Problemangemessenheit eines Algorithmus (A). * analysieren und testen Algorithmen und Programme (MI). * erstellen syntaktisch korrekte Quelltexte in einer Programmiersprache (MI). * erläutern die Begriffe Syntax und Semantik an Beispielen (A), * analysieren Quelltexte auf syntaktische Korrektheit (A). | Beispiele:   * Turtle-Grafik:   + Kreise, Vierecke, Dreiecke, Bäume, Sterne, Häuser, Blüten, Labyrinthe, Polygone mit verschiedenen Seitenlängen, die eingegeben werden können und in einer Liste oder einem Array gespeichert werden * Kurze Programme zum Testen von Ein- und Ausgabemöglichkeiten * Einfache mathematische Anwendungen:   + Umwandlung einer als Zeichenkette eingegebene Zahl in einen anderen Datentyp   + Produkt/Summe/…/Minimum zweier eingegebener Zahlen   + Test auf Teilbarkeit durch eine Zahl   + Größter gemeinsamer Teiler   + Primzahlen bestimmen   + Lösung einer einfachen linearen Gleichung mit einer Variablen * Umfangreichere Aufgaben / Projektaufgaben   + Cäsar-Verschlüsselung   + Vokabeltrainer   + Nim-Spiel   + Vier gewinnt   Material:  Viele Anwendungsbeispiele und Aufgaben unter <http://www.inf-schule.de/programmierung/imperativeprogrammierung> (abgerufen am 25.06.2017) (mdb)  Anregungen für Aufgabenstellungen (allerdings mit etwas anderem Befehlsumfang als in Python selbst)  <http://www.tigerjython.ch/index.php?inhalt_links=navigation.inc.php&inhalt_mitte=turtle/turtle.inc.php> (abgerufen am 18.04.2017) (mdb)  <https://docs.python.org/2/library/turtle.html>  (abgerufen am 7.5.2017) (mdb)  Die Seite des Computer Science Circles bietet sowohl Texte als auch interaktive Aufgaben, mit denen man den Lernfortschritt testen kann:  <https://cscircles.cemc.uwaterloo.ca/de/>  (abgerufen am 24.10.2017)  (mdb) |
| Lernerfolgsüberprüfung durch Aufgaben in einer Klassenarbeit | | |