B.IV.7

Algorithmen – Objektorientierte Programmierung

Einheit: Programmieren mit *Python* – Grundlagen und Erstellen eines Textadventures

Christina Hund

In dieser Einheit erlernen Ihre Lernenden nicht nur die Grundlagen der Programmierung, sondern auch wichtige Funktionen der Sprache *Python*. Sie lernen anhand dieser textbasierten Programmiersprache die wichtigsten Befehle (Text-Eingabe, Variablen, Wenn/Dann-Funktionen und Schleifen) kennen und wenden diese *Python*-Grundfunktionen an. Schließlich festigen sie ihre Kenntnisse durch eine vertiefte Anwendung der *Python*-Grundfunktionen in der Entwicklung eines Textadventures als eigenes Spiel. Anhand eines Kriterienkatalogs werden die Entwicklungsergebnisse von den Mitschülerinnen und Mitschülern bewertet.

KOMPETENZPROFIL

Klassenstufe:	8/9
Dauer:	8 Unterrichtsstunden
Lernziele:	Die Lernenden … 1. recherchieren Grundkenntnisse zur Programmierung, 2. wenden *Python*-Grundfunktionen an, 3. definieren die wichtigsten Befehle in *Python*, 4. entwerfen ein eigenes Textadventurespiel und setzen dessen Entwicklung in *Python* um.
Thematische Bereiche:	Programmierung, textbasierte Programmiersprache, *Python*, Computerspiel, Spielentwicklung, Algorithmen, Textadventure
Kompetenzbereiche:	Implementieren, Darstellen und Interpretieren, Produzieren und Präsentieren, Analysieren und Reflektieren

Fachliche Hinweise

Was sollten Sie zum Thema wissen?

Mit dieser Einheit werden die Grundlagen von *Python* vermittelt, sodass kein spezielles Vorwissen nötig ist. *Python* ist mittlerweile eine der führenden Programmiersprachen, was vor allem daran liegt, dass sie einfach zu lernen, aber schwer zu meistern ist. Sie ist vielseitig einsetzbar und deshalb in vielen Bereichen angesiedelt. Es ist hilfreich sich vorab als Lehrkraft mit der Programmiersprache *Python* zu beschäftigen, damit eventuelle Fehlerquellen schneller erkennbar sind. Hier kann auch Grundwissen aus anderen Programmiersprachen unterstützen.

Welches Vorwissen sollten die Lernenden mitbringen?

Prinzipielle Programmierkenntnisse vonseiten der Lernenden sind nicht zwingend vorausgesetzt, aber definitiv eine große Stütze. Vor allem schwächere Lernende können davon profitieren zunächst mit einer *Blockly*-Sprache wie *Scratch* oder *Robot Karol* zu arbeiten. Auf dieser Grundlage aufbauend können die Schülerinnen und Schüler die visuellen Bausteine einfacher in Textprogrammierung übersetzen. Verwenden Sie hierfür beispielsweise unsere hierzu bereits veröffentlichten RAABE-Unterrichtsmaterialien, u. a.:

- Programmierung mit *Scratch* – Erste Schritte in der visuellen Programmierumgebung
- Erste kleine Programme in *Scratch* entwickeln – Einsatz als Einzelprojekte oder Stationenarbeit
- Programmierung mit *Scratch* – Eigene Spiele programmieren
- Programmieren eines *Scratch*-Spiels zum Klimawandel und was wir dagegen tun können
- Selbstlerneinheit: Lerne die visuelle Programmierumgebung *Robot Karol* kennen
- Einführung in die Programmierumgebung *Robot Karol* – Mit Selbstlernstationen

Didaktisch-methodische Hinweise

Vorbereitung

- Stellen Sie ausreichend Laptops/PCs/mobile Endgeräte im Klassenraum zur Verfügung, idealerweise ein Gerät pro Schüler/-in oder mindestens ein Gerät pro Schülerpaar.
- Sorgen Sie für die Bereitstellung von Internet im Klassenraum.
- Stellen Sie sicher, dass *IDLE3* als Entwicklungsumgebung auf den Endgeräten installiert ist.

Hinweis: *IDLE* steht für *Integrated Development and Learning Environment*. Es handelt sich dabei um eine leichtgewichtige integrierte Entwicklungsumgebung (IDE) für Installer der Programmiersprache *Python*. *IDLE* ist als einfache IDE ohne Überladung mit komplizierten Funktionalitäten gedacht, die sich auch für Anfängerinnen und Anfänger, gerade auch im Bildungsumfeld, eignet. Sie funktioniert plattformübergreifend unter *Windows*, *Unix* und *maxOS*.

Benötigte Dateien

- *ggf. Installationsdateien für Python und IDLE3:* python.org
- *print_input.py*
- *labyrinth_1.py & labyrinth_2.py*
- *notenabfrage.py & notenabfrage_lsg.py*
- *rechner.py*

Auf einen Blick

Benötigte Materialien und Dateien

- ☐ PC/Laptop/mobiles Endgerät mit Internetzugang und Installation von *IDLE3 für Lehrkraft und pro Lernenden oder pro Schülerpaar*
- ☐ *ggf. Installationsdateien für Python und IDLE3:* python.org
- ☐ *print_input.py*
- ☐ *labyrinth_1.py & labyrint_2.py*
- ☐ *notenabfrage.py & notenabfrage_lsg.py*
- ☐ *rechner.py*

Einstieg: Programmierung

Thema:	Grundlagen der Programmierung
M 1	**Wozu braucht man Programmiersprachen?**

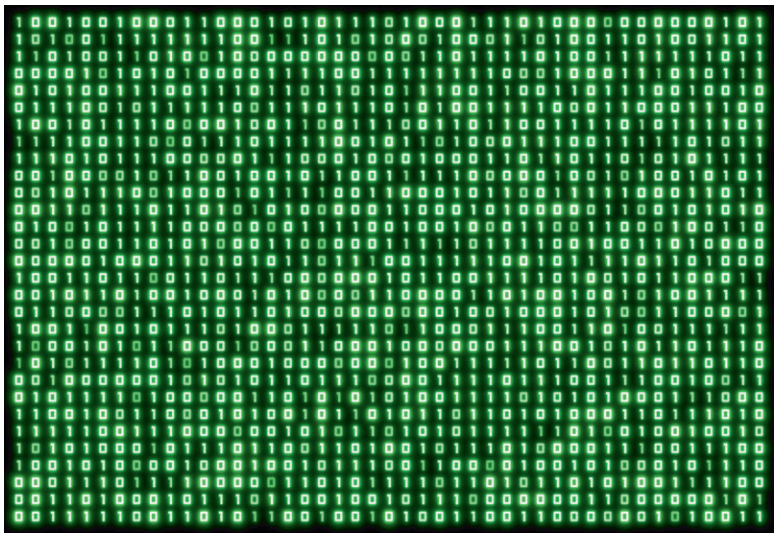
Erarbeitung: Programmieren mit *Python*

Thema:	*Python*-Grundlagen
M 2	**Grundlagen der Programmiersprache *Python***
M 2a	**IDLE3-Anleitung**
M 3	***Python*: Ein- und Ausgabe von Text mit den Befehlen `print` und `input`**
Benötigt:	☐ *HTML_Tags.html*
M 4	***Python:* Variablen**
Benötigt:	☐ **kleine Kartons, Karten** ☐ **rechner.py**
M 5	**Abfragen mit *Python*: Wenn, dann, sonst ...!**
Benötigt:	☐ *notenabfrage.py*

Festigung: Textadventures

Thema:	Ein eigenes Spiel gestalten
M 6	***Python*: Die Spiel-Schleife**
Benötigt:	☐ *labyrinth_1.py* ☐ *labyrinth_2.py*
M 7	**Erstelle dein eigenes Textadventure mit *Python***
M 7a	**Themenideen für Textadventures /Hilfekarte**
M 7b	**Kriterienkatalog zur Bewertung von Textadventures**
M 8	**Übersicht der wichtigsten *Python*-Befehle**

Wozu braucht man Programmiersprachen? M 1



© *Flavio Coelho/Moment*

Computer haben ein sehr kleines Vokabular: Sie kennen nur 1 und 0. Wörter verstehen sie nicht. Doch wie bekomme ich meinen Computer dazu überhaupt etwas zu tun? Muss ich dann in 1en und 0en mit ihm sprechen?
Theoretisch: Ja. Praktisch: Zum Glück gibt es Programmiersprachen! Programmiersprachen dienen dazu eine Sprache zu bieten, die sich in 1en und 0en übersetzen kann. So kann der Computer durchaus auch Anweisungen in Wortform verarbeiten.

Aufgabe


Recherchiert im Internet zu der Frage: Was macht eine Programmiersprache aus?

1. Welche Grundfunktionen haben die meisten Programmiersprachen? Nenne vier Stück.

2. Was ist ein „Compiler“?

3. Welcher Text wird bei den meisten Programmiersprachen als Einstieg ausgegeben?

M 2

Grundlagen der Programmiersprache *Python*

Bei Programmiersprachen ist die Auswahl groß. Es gibt viele verschiedene Ansätze und Einsatzmöglichkeiten, deshalb gibt es Sprachen, die sich für bestimmte Dinge besser eignen als andere.

Die letzten Jahre hat die Programmiersprache *Python* einen großen Sprung in der Beliebtheit gemacht. Anfänger können damit schnell und einfach Ergebnisse erzielen, dennoch ist diese Sprache so vielseitig einsetzbar, dass selbst Profis auf sie schwören. *Python* gilt als einfach zu lernen, aber schwer zu meistern. Der einfache Einstieg in diese Programmiersprache kommt durch den zugänglichen Aufbau der Befehle (Syntax). Auch die Begriffe sind meist sehr einleuchtend, wenn man etwas Englisch spricht. Dennoch ist es eine eigene Sprache mit eigenem Vokabular und eigener Grammatik.

Aufgabe 1: *Python*-Steckbrief

Recherchiere im Internet, um Antworten auf die folgenden Fragen zu finden.

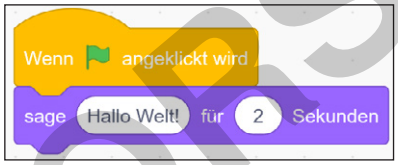
Wie hieß der Entwickler von *Python*? _____

In welchem Jahr erschien die erste *Python*-Version? _____

Zur Erstellung welcher Programme wird *Python* vorrangig verwendet?

_____

Vergleiche *Python* mit *Java* und *Scratch*. Alle drei Programmiersprachen geben das Ergebnis „Hallo Welt!“ aus. Welche Unterschiede fallen auf?



Scratch

```
public class HelloWorld
{
    public static void main (String[] args)
    {
        System.out.println("Hallo Welt!");
    }
}
```

Java

```
print("Hallo Welt!")
```

Python

Aufbau eines *Python*-Befehls

```
print(„Hallo Welt!“)
```

In *Python* gibt es Befehle, wie hier „print“. Dieser Befehl sagt nur aus, dass etwas angezeigt werden soll, aber nicht genau was. Deshalb gibt es genauere Informationen in den Klammern direkt hinter dem Befehl, hier „Hallo Welt!“. Übersetzt heißt das also „zeige den Text ‚Hallo Welt!‘ an“. Ein Satzende ist in *Python* einfach ein Textumbruch.

IDE: Integrated Development Environment

Grundlegend kann man *Python* mit jedem Text-Editor schreiben, was aber die Übersicht schnell erschwert. Deshalb greifen Programmiererinnen und Programmierer eher auf sogenannte IDE (engl.: *Integrated Development Environment*, dt.: Integrierte Entwicklungsumgebung) zurück. Diese haben einige Vorteile, denn sie stellen den Code farblich klarer dar und können die Fehlersuche erleichtern.

M 6 *Python*: Die Spiel-Schleife

Spiele können ganz schön viele Abfragen enthalten. Je komplexer das Spiel, desto mehr Entscheidungen und Eingaben kann ein Spieler oder eine Spielerin machen. Einiges kann zu einem „Game Over“ führen, jedoch wäre es mühselig für jede einzelne Entscheidung ein neues Game Over einzubauen.

Irrgartenspiel „Das Labyrinth des Goldes“
In der Datei *labyrinth_1.py* findet ihr den Anfang eines Irrgartenspiels. Die bisherigen Tester und Testerinnen hatten aber einen Kritikpunkt: Wenn sie eine falsche Antwort gegeben haben, war das Spiel immer vorbei. Dabei hatten sie sich einfach nur vertippt.
Fällt dir eine Lösung ein, wie man das umgehen könnte?

Optimierung des Irrgartenspiels „Das Labyrinth des Goldes“
Die Entwicklerinnen und Entwickler haben sich noch einmal an das Spiel gesetzt, um den Kritikpunkt der Testerinnen und Tester auszumerzen. Das Ergebnis findet ihr in der Datei *labyrinth_2.py*
Im letzten Absatz findet sich die Lösung:

```
while True:
if antwort == “R”:
   print(„… und fällst in ein tiefes Loch. Tja! Game Over.”)
   break
elif antwort =“L“:
   print(„… und siehst einen Pfad, der dich zum Schatz leitet!“)
   break
else:
   print(„Das ist keine richtige Antwort!“)
   antwort = input(„Entscheide dich.“)
```

Aufgabe 1: while-Schleife
Finde heraus, was die folgenden Befehle bewirken:

`while x :`	
`True`	
`break`	

Aufgabe 2: weitere Optimierung des Irrgartenspiels „Das Labyrinth des Goldes“
Das Irrgartenspiel „Das Labyrinth des Goldes“ ist bisher noch sehr kurz. Wie kann man es sinnvoll erweitern? Überlege, wie die folgende weiteren Kritikpunkte der Testerinnen und Tester umsetzen kannst.

- ☐ „Mehr Entscheidungen!“
- ☐ „Es soll nicht jede Entscheidung eine Game-Over-Möglichkeit haben.“
- ☐ „Könnte es neben einem Game Over und einem Goldschatzfund nicht noch andere Enden geben?“