Matthias Heming 25. November 2009

Hinweise zur Phase »Programmierung«

1 Kompetenzformulierung

Schülerinnen und Schüler ...

- nutzen die PythonScriptShell, d. h. den interaktiven Modus, um auf der Basis der Programmiersprache Python direkt mit ihrem Mobiltelefon zu kommunizieren.
- nutzen den Editor Ped, um Befehlsfolgen in einer Textdatei zu speichern und diese als Paket vom Interpreter bearbeiten zu lassen.
- nutzen die Zwischenablage, um Texte zwischen verschiedenen Programmen auszutauschen (optional).
- kennen die Punktnotation für den Zugriff auf Methoden/Attribute von Objekten/Modulen.
- kennen verschiedene Python-Module und nutzen diese unter Zuhilfenahme zugehöriger Dokumentationen.
- legen eigene Bezeichner für Objekte fest und nutzen diese als Parameter in Methoden.

2 Detaillierte Zielsetzung

Der Beispielquelltext auf dem ersten Arbeitsblatt konfrontiert Schülerinnen und Schüler mit verschiedenen informatischen Konzepten:

- Modularisierung (`import audio`)
- Zuweisungsoperator/Variablenkonzept (`soundobjekt = ...`)

  Achtung, bei Python kann die Analogie eines Containers für Variablen nicht verwendet werden, vielmehr stellt eine Variable einen einfachen Namen für ein im Speicher erzeugtes Objekt dar. Der Name ist prinzipiell unabhängig von dem Objekt, auf das er verweist. Während eines Programmablaufes können Namen durch Zuweisung geändert werden, ein Objekt kann mehrere Namen haben, ein Name kann nacheinander unterschiedlichen Objekten zugewiesen werden.
- Punktoperator zum Zugriff auf Attribute von Objekten – Achtung, in Python sind auch Klassen oder Module Objekte.
- Methoden mit Rückgabewert (`... = audio.Sound.open(...)`) und ohne (`soundobjekt.record()`)

Anstatt jedoch jeden einzelnen Punkt konkret anzusprechen, sei es Schülerinnen und Schülern erst einmal selbst überlassen, den Quelltext zu interpretieren und intuitiv Bedeutungen zu erschließen. Da im bisherigen Verlauf nur geringfügig auf die Syntax der Programmiersprache Python eingegangen wurde und auch Objekte nur in Form von Objektkarten angesprochen wurden, werden sich Schülerinnen und Schüler hoffentlich einige Fragen stellen, so dass verschiedene Dinge geklärt werden können.

- Wofür steht `import audio`? Eine Definition für Python-Module ist unten vorgeschlagen. Hier muss die Wichtigkeit der Dokumentation unterstrichen werden, da man sonst nicht weiß, welche Möglichkeiten ein Modul bietet. Die vorgeschlagene Erläuterung der import-Anweisung soll dabei nicht vom Objektkonzept abweichen und stellt deshalb die Anweisung als Erstellung eines Spezialobjektes (Modulobjekt) vor. Dies sollte den Bezug zu vorher erstellten Objektkarten erleichtern.

- Wieso kann man in `audio.Sound.open()` zwei Punkte verwenden? Ist `Sound` ein Objekt mit der Fähigkeit `open`? Ein Objekt im Modul? Eine Erklärung an dieser Stelle ist besonders schwierig, da zum Einen noch nicht auf das Klassenkonzept eingegangen werden soll, `open` jedoch nur als Klassenmethode sauber zu erklären ist, zum Anderen bisher nur Methoden als Fähigkeiten von Objekten angesprochen wurden.

  Auch wenn dieser Befehl generell unglücklich ist[1], ist es wohl unumgänglich, die tatsächlich dahinterstehende Komplexität zu vermeiden, z. B. indem nur erwähnt wird, dass eine weitere Strukturierung eines Moduls möglich ist. Auf welche Art und Weise dies geschieht, ist aktuell nicht notwendig zu hinterfragen.

- Die Methode `open()` gibt ein konkretes Sound-Objekt zurück. In diesem Sinne kann bereits von einer Art Konstruktor gesprochen werden, da die Methode ein spezielles Soundobjekt erst erstellen musste. Es kann darauf hingewiesen werden, dass später noch gelernt wird, wie man eigene Objekte definiert und erstellt.

  Zusammen mit `soundobjekt.record()` es nun möglich, zwischen Anfragen (mit Rückgabewert) und Aufträgen (ohne Rückgabewert bzw. `None`, welches zum jetzigen Zeitpunkt noch nicht bekannt ist) zu unterscheiden. Dies ist dann notwendig, wenn Zuweisungsoperator und Bezeichner als Begrifflichkeiten eingeführt werden. Denn auf der rechten Seite des Zuweisungsoperators darf nur ein Bezeichner oder eine Methode mit konkretem Rückgabewert stehen.

Auf Arbeitsblatt (2) sollen die Methoden aus den bereits angesprochenen Python-Modulen selbstständig in Zusammenhang gebracht werden. Zudem sorgt die Bearbeitung der Aufgaben zum veränderten Quelltext dafür, dass sich Schülerinnen und Schüler mit dem Variablenkonzept auseinandersetzen. Da bereits vorher mit diesen Objekten gearbeitet wurde, sollten sich keine Erklärungsnotwendigkeiten ergeben.
Die zweite Aufgabe kann zum einen als Ausgangspunkt für eine mögliche Weiterarbeit zum Thema Barcodes dienen, hat jedoch den eigentlichen Zweck, darauf aufmerksam zu machen, dass ein Python-Interpreter eine Skriptdatei in wesentlich höherer Geschwindigkeit ausführt, als eine manuelle Eingabe über die interaktive Konsole. Eine künstliche Wartezeit in Form eines `sleep`-Methodenaufrufes sollte nicht schwierig sein, da der entsprechende Dokumentationsausschnitt bereits vorgegeben wurde. Falls anstelle des Abtippens ein Barcode-Scanner eingesetzt wird, so muss damit das Konzept einer systemweiten Zwischenablage erläutert werden. In Abhängigkeit zu den individuellen Vorerfahrungen der Lerngruppe kann die Ausführlichkeit hierbei besonders variieren.
Anstelle einer Lernzielkontrolle *kann* die dritte Aufgabe des Arbeitsblatt (2) – die Zusammenfassung der Ergebnisse in einem konkreten Programm – als größere Hausaufgabe betrachtet werden. Je nach Lern- bzw. Arbeitsklima kann diese Hausaufgabe ebenfalls von der Lehrkraft eingesammelt werden. Aufgrund der erleichterten Möglichkeiten, Programme untereinander auszutauschen – es ist nicht nur über den heimischen Computer und USB-Sticks/E-Mail möglich, sondern ebenfalls auf dem Schulhof mit einer Bluetooth-Verbindung – , muss die Relevanz einer solchen Hausaufgabe für die Leistungsbewertung in individuellen Bezug zur Lerngruppe gesetzt werden.
Der zweite Bedingung, die an das fertige Programm gestellt wird (das Löschen vorheriger Sound-Dateien), wird eventuell problematisch. Beim ersten Aufruf ist es nicht notwendig, eine Datei zu löschen, ja es würde sogar ein Fehler erzeugt. Bei weiteren Aufrufen muss nun aber die Datei entfernt werden. Damit ist es nicht möglich ein Programm für diese unterschiedlichen Situationen zu schreiben, ohne auf das Konzept der Verzweigung einzugehen.
Der häufigere Fall ist derjenige, der eine bereits existente Datei voraussetzt, und sollte für die Programmrealisierung ausgewählt werden. Da der weitere Programmverlauf unabhängig von dem erfolgreichen Löschvorgang ist (bzw. somit

[1] Es sei zu kritisieren, dass das Soundobjekt nicht über einen Parameter im Konstruktor erzeugt werden kann, sondern die Klassenmethode *open*, dazu verwendet werden muss. Eine als *Bug* zu bezeichnende Designentscheidung, da der Konstruktor eine Instanz erzeugt, der im Nachhinein keine Datei zugeordnet werden kann, womit das resultierende Objekt wertlos ist.

immer dafür gesorgt ist, dass die Sounddatei nicht existiert), soll klar werden, dass Fehler grundsätzlich nicht schlecht sind und für den weiteren Verlauf ersteinmal nicht hinderlich.

3 Mögliche Weiterarbeit

- Im weiteren Verlauf ist der Zugriff auf das Dateisystem erst einmal nicht vorgesehen, weswegen die Kenntnis von Pfaden und deren Beschreibung nicht notwendig ist – der im Arbeitsblatt vorgegebene Pfad kann ohne nähere Hintergrundinformation verwendet werden. Dennoch ist dieses Thema im Sinne der Informatik wichtig und es kann je nach Interessenlage der Schülerinnen und Schüler mit Hilfe des Moduls *Dateipfade* darauf eingegangen werden.
- Bereits erwähnt wurde die Kodierung von Barcodes. Im Rahmen eines noch zu erstellenden Moduls hierzu kann erläutert werden, in welchen Bereichen des täglichen Lebens (mehrdimensionale) Barcodes vorkommen, welche Vor- und Nachteile bzw. welche Auswirkungen sie haben. Gerade bei EAN[2] und ISBN/ISSN[3] können Prüfsummen relativ leicht berechnet werden, um zum allgemeinen Thema der Kodierung (fehlererkennende Kodes) zu schwenken (CSUnplugged bzw. AbenteuerInformatik bieten dazu ebenfalls interessante Übungen an).
- Falls bei Schülerinnen und Schülern das Interesse an den Steuerungsmöglichkeiten des Telefons über die Programmiersprache geweckt wurde, können weitere Beispiele zum Ausprobieren formuliert werden. Hierbei muss verdeutlicht werden, dass die Abläufe aus informatischer Sicht (zumindest am zweiten Beispiel) recht komplex sind. Diese Quelltextbeispiele stellen also nur Daten dar, die zum Hineinschnuppern und Ausprobieren vorgestellt werden.

  Beispiel 1:

  ```
  import camera
  pic = camera.take_photo()
  pic.save("E:\\einfoto.jpg")
  ```

  Beispiel 2:

  ```
  import contacts
  db = contacts.open()
  kontakt = db[1]
  detail = kontakt[1]
  detail.value = "neu"
  ```

- Der Datentyp `str` (String/Zeichenkette) kann hier näher betrachtet werden, es muss an dieser Stelle jedoch noch nicht auf die `unicode`-Variante eingegangen werden. Auch wenn intuitiv klar ist, was Zeichenketten sind, so könnte man deutlicher machen, dass auch diese Dinge nur Objekte sind, welche bereits über verschiedene spezielle Fähigkeiten/Methoden verfügen, die durch die Programmiersprache Python vorgegeben sind.
- Gerade dann, wenn Schülerinnen und Schüler ausprobieren wollen, ist es vielleicht interessant, die Ergebnisse dieses Prozesses speichern zu können. Dies ist mit der PythonScriptShell nicht möglich, es werden höchsten einzelne Befehle mit »previous Command« abrufbar gemacht. So kommt man auf den Editor Ped zu sprechen. Dieser bietet zwar nicht die Interaktivität der Shell (außer, diese wird aus dem Programm gestartet), kann dafür aber Dateien speichern. Damit wird es natürlich ebenfalls notwendig, dass Format der Python-Dateien anzusprechen, welche im Sinne der Quelltextsammlung bei Modulen schon kurz angeklungen ist. Dies ist nicht weiter kompliziert, eventuell muss jedoch erläutert werden, dass der Interpreter die Interaktive Konsole von dem Ausführen eines Skriptes unterscheidet, indem bei einem Skript nur noch explizit die Dinge ausgegeben werden, die innerhalb der Klammern einer **`print`**-Anweisung stehen.

[2] `http://de.wikipedia.org/wiki/European_Article_Number`
[3] `http://de.wikipedia.org/wiki/ISBN`,`http://de.wikipedia.org/wiki/ISSN`

4 Genderaspekt

Bisher keine Besonderheiten gefunden, die zu berücksichtigen wären. Für mich ist jedoch besonders interessant, ob es Unterschiede es hier in der Herangehensweise von Mädchen und Jungen gibt. Ist es tatsächlich der Fall, dass Jungen eher die experimentierfreudigen sind und daher vielleicht erst einmal ohne Dokumentation herumprobieren (eventuell auch sofort die passende Methode `soundobjekt.stop()` bzw. `soundobjekt.play()` finden, die von der Bezeichnung her ja durchaus intuitiv sind)?

5 Für das Merkheft

Skript: Aneinanderreihung von einfachen Pythonbefehlen, gespeichert in einer Skriptdatei (mit der Endung `.py`).

Modul/Paket: In der Programmiersprache Python können vorgefertigte (komplexe) Quelltexte in einzelnen Dateien gesammelt werden, damit sie später in anderen Programmen wiederverwendet werden können. Diese Dateien bekommen ebenfalls die Endung `.py` und sind damit äußerlich nicht von Skriptdateien zu unterscheiden. Man bezeichnet diese Dateien als **Modul**, der Dateiname (ohne `py`-Endung) ist dabei der Modulname.

Fasst man mehrere Moduldateien in einem Verzeichnis zusammen, so spricht man von einem **Paket**. Der Name des Verzeichnis ist dabei der Paket-Name.

Häufig wird anstelle von Modulen/Pakten auch von *Bibliotheken* gesprochen.

Beispiel: Die Module `sound`, `camera` oder `contacts`

import <xxx>: Eine spezielle Python-Anweisung. Sie veranlasst den Interpreter, das Modul oder Paket mit Namen <xxx> zu laden und über ein Spezialobjekt (ein Modulobjekt) zugreifbar zu machen. Für das Spezialobjekt wird automatisch der Bezeichner <xxx> festgelegt. Mit Hilfe der Python-Anweisung **`import`** `<xxx> as <yyy>` kann anstelle des standardmäßigen Bezeichners <xxx> ein alternativer Bezeichner <yyy> angegeben werden.

Punktoperator: Mit Hilfe des Punktoperators kann auf die Attributwerte und Methoden von Objekten (auch Modulobjekten) zugegriffen werden.

Beispiel: Die `play()`, `stop()` oder `record()`-Methoden des Soundobjekts. Zugriff über `soundobjekt.play()`, die `take_photo()`-Methode des speziellen Modul-Objekts `camera`, Zugriff über `camera.take_photo()`.

Zuweisung: Mit Hilfe des Gleichheitszeichung wird unter Python ein Bezeichner für ein Objekt festgelegt.

Beispiele: `neuerbez=dasobj` oder `soundobjekt = audio.Sound.open()`

Dabei steht auf der linken Seite der neue Bezeichner (im Beispiel `neuerbez`) und auf der rechten Seite das Objekt, für welches der neue Bezeichner festgelegt werden soll (im Beispiel `dasobj`).

Man beachte: Ein richtiges Objekt kann man auf die rechte Seite gar nicht schreiben, denn Objekte sind nur durch Bezeichner zugreifbar. Auf der rechten Seite kann daher z. B. ein alter, bereits vorher festgelegter, Bezeichner stehen (im Beispiel 1 `dasobj`). Es ist aber auch möglich, dass dort ein Befehl steht, der erst noch von Python ausgeführt werden muss, der aber als Ergebnis ein konkretes Objekt hat. Im Beispiel 2 wird durch den Befehl `open()` ein neues Soundobjekt erzeugt, welches bisher keinen Bezeichner trug.