Lehrpersonenkommentar

Das Aufgabenset zur Erlangung der Fähigkeiten, um den Astro Pi programmieren zu können, ist in drei Teile gegliedert:

- Teil 1: Informatische Grundkonzepte / Repetition
- Teil 2: Die Sprache Python
- Teil 3: Projektspezifische Kenntnisse

Dieser Aufbau ermöglicht es den SuS, sich Schrittweise der Programmierung des Astro Pi in der Programmiersprache Python zu nähern.

Teil 1 – Informatische Grundkonzepte / Repetition

Die SuS erlernen die Grundkonzepte des Programmierens und des algorithmischen Denkens.
Um die Grundlagen zu erarbeiten, wird auf das Lehrmittel connected 03 verwiesen, insbesondere auf das *Kapitel 3 Abheben mit Algorithmen*.
Empfohlen wird, dass die SuS mindestens folgende Aufgaben aus dem connected Lösen, um die Grundkonzepte des Programmierens zu erlernen und mit der algorithmischen Problemlösung vertraut zu werden:

- 305: Ablaufdiagramm ergänzen (Schleifen)
- 308: Ablaufdiagramme zu Programmen zuordnen (Algorithmische Problemlösung)
- 311: Ablaufdiagramm in Programmbefehle übersetzen (Schleifen)
- 313: Bestehendes Programm erweitern (Schleife)
- 314: Bestehendes Programm verstehen (Unterprogramme)
- 315: Bestehendes Programm situativ ändern (Unterprogramme)
- 316: Bestehendes Programm überarbeiten (Unterprogramme)
- 318: Unterprogramm in bestehendes Programm einfügen (Unterprogramme)
- 319: Bestehendes Programm korrigieren (sequenzielle Ausführung)
- 321: Bestehendes Programm überarbeiten (Parameter übergeben)
- 322: Ablaufdiagramm ergänzen (Verzweigung)
- 325: Bestehendes Programm erweitern (Variabeln)
- 326: Bestehendes Programm erweitern (Schleife)
- 327: Bestehendes Programm erweitern (Verzweigung)
- 328: Regeln ergänzen (Bedingungen)

SuS welche die Grundkonzepte des Programmierens bereits kennen, können direkt mit den Aufgabenkarten im Teil 2 starten.

Teil 2 – Die Sprache Python

Die SuS lernen die Grundkonzepte des Programmierens in der Sprache Python anzuwenden und mit dieser Sprache einfache algorithmische Probleme zu lösen. Voraussetzung, für diese Aufgaben ist das Kennen der Grundkonzepte des Programmierens (Variablen, Schleifen, Verzweigungen), wie sie in Teil 1 – Grundlagen erarbeitet und gefestigt wurden.

Programmierumgebung

Die Aufgaben können prinzipiell in jeder Programmierumgebung gelöst werden, welche Python interpretieren kann. Empfohlen wird die Onlineumgebung Python Tutor

(https://pythontutor.com/visualize.html). Mit dieser Umgebung kann der Code einfach Zeile für Zeile nachvollzogen werden und es entfällt die Installation einer entsprechenden Umgebung.

Eine kurze Einführung zu der Programmierumgebung gibt es auf der Aufgabenkarte Programmierumgebung. Wird mit einer anderen Programmierumgebung gearbeitet, kann diese Karte weggelassen werden.

Aufgabenreihenfolge

Die Aufgabenkarten sollen in der angegebenen Reihenfolge (1-7) durchgearbeitet werden. Dies da Prinzipien, die in früheren Karten eingeführt worden sind, in späteren Karten wieder verwendet werden. Folgende Inhalte werden in den Aufgabenkarten erarbeitet:

- Aufgabenkarte 0: Programmierumgebung kennen lernen
- Aufgabenkarte 1: Variabeln
- Aufgabenkarte 2: Bedingungen und Verzweigungen
- Aufgabenkarte 3: Schleifen
- Aufgabenkarte 4: Repetition von Variabeln, Bedingungen und Verzweigungen
- Aufgabenkarte 5: Funktionen, Funktionen mit Parametern
- Aufgabenkarte 6: Listen
- Aufgabenkarte 7: Modul Zufallszahl (optional)

Für das Durcharbeiten aller Aufgabenkarten wird ein Zeitrahmen von 4 bis 6 Lektionen benötigt.

Die Aufgabenkarten

Die Aufgabenkarten eignen sich optimal, um doppelseitig gedruckt im A5-Format abgegeben zu werden. Die Karten sind jeweils nach dem gleichen Muster aufgebaut. Dies ist wie folgt:

- Einführung in die Aufgabe und Nennung des Zieles der jeweiligen Karten
- Überblick über die einzelnen Teilaufgaben und Platz für Notizen
- Einzelne Teilaufgaben
    - Einführung in die Aufgabe
    - Aufgabenstellung
    - Tipps zum Vorgehen
    - Lösungsvorschlag inkl. Erläuterungen

Da die Lösungen zu den Aufgaben unmittelbar und inklusive Erläuterungen zur Verfügung stehen, muss darauf geachtet werden, dass die SuS die Aufgaben selbstständig lösen. Das hier grundlegende Verständnis für die textbasierte Programmiersprache Phyton ist essenziell für die erfolgreiche Programmierung des Astro Pi.

Hilfsmittel

Als zusätzliches Hilfsmittel steht eine Befehlsübersicht bereit, welche einen Befehl jeweils erklärt, in Phyton (textbasiert) zeigt sowie in Scratch (blockbasiert) zeigt. Damit kann den SuS der Transfer von der blockbasierten in die textbasierte Programmiersprache erleichtert werden.

Teil 3 – Projektspezifische Kenntnisse

Die SuS lernen projektspezifische Kenntnisse, die zum erfolgreichen Programmieren des Astro Pi benötigt werden. In einem ersten Schritt werden die SuS in die Programmierumgebung des Astro Pi eingeführt. Anschliessend lernen die SuS am Beispiel des Temperatursensors, wie die Sensordaten ausgelesen und in eine Datei geschrieben werden können.

Programmierumgebung

Die Aufgaben 1-4 sollen im Emulator, die Aufgabe 5 auf dem Astro Pi gelöst werden. Entsprechend ist im Aufgabenset die Einführung in den Emluator den Aufgaben 1-4 vorangestellt. Die Einführung zum Astro Pi ist der Aufgabe 5 vorangestellt.

Der Emulator simuliert alle Funktionen des Astro Pi Sense HAT in einem Webbrowser. Auf dem Sense HAT-Webemulator kann der Code ausgeführt und selbst erlebt werden, was mit dem Sense HAT des Astro Pi möglich ist. Im Web-Emulator geschriebene Codes sind ohne Änderungen direkt auf einen physischen Astro Pi übertragbar.

Für die Arbeit mit dem Astro-Pi ist empfohlen, diesen an einen externen Monitor anzuschliessen und nicht an den Laptop der SuS. Dafür wird pro Astro Pi zusätzlich eine Tastatur und Maus sowie je nach Version des Astro Pi ein HDMI- oder Micro-HDMI-Kabel benötigt.

Aufgabenreihenfolge

Die Aufgabenkarten sollen in der angegebenen Reihenfolge (X-Y) durchgearbeitet werden. Dies da die Karten aufeinander aufbauend sind. Folgende Inhalte werden in den Aufgabenkarten erarbeitet:

- *Programmierumgebung Emulator*
- Aufgabenkarte 1: Temperatursensor auslesen
- Aufgabenkarte 2: Die Temperatur speichern
- Aufgabenkarte 3: Zeit und Temperatur speichern
- Aufgabenkarte 4: Mehrere Temperaturen speichern
- *Programmierumgebung Astro Pi*
- Aufgabenkarte 5: Gemessene Temperatur verifizieren

Für das Durcharbeiten aller Aufgabenkarten wird ein Zeitrahmen von 10 bis 14 Lektionen benötigt.

Die Aufgabenkarten

Die Aufgabenkarten eignen sich optimal, um doppelseitig gedruckt im A5-Format abgegeben zu werden. Die Karten sind jeweils nach dem gleichen Muster aufgebaut. Dies ist wie folgt:

- Einführung in die Aufgabe und Nennung des Zieles der jeweiligen Karten
- Überblick über die einzelnen Teilaufgaben und Platz für Notizen
- Einzelne Teilaufgaben
    - Einführung in die Aufgabe
    - Aufgabenstellung
    - Tipps zum Vorgehen
    - Lösungsvorschlag inkl. Erläuterungen

Da die Lösungen zu den Aufgaben unmittelbar und inklusive Erläuterungen zur Verfügung stehen, muss darauf geachtet werden, dass die SuS die Aufgaben selbstständig lösen. Die Aufgaben sind aufeinander aufbauend, so dass immer wieder auf das Vorwissen aus vorhergehenden Aufgaben zurückgegriffen wird.

Hilfsmittel

Als Hilfsmittel steht eine Übersicht bereit, welche die wichtigsten Funktionen zu den einzelnen Sensoren und der Kamera des Astro Pi kurz erläutert.

Debugging

Nachfolgende Checkliste kann beim Debugging, der Suche von Fehlern im Programmiercode hinzugezogen werden. In dieser wird auf die häufigsten Fehler beim Programmieren hingewiesen, am Beispiel des Temperatursensors.

- ☐ Die im Code benötigten Pakete und Module werden ganz zu Beginn des Codes importiert. Hier ein Beispiel eines Importes vom Modul *SenseHat* aus dem Paket *sense_hat* und vom Paket *time*.

```
1  from sense_hat import SenseHat
2  import time
```

- ☐ Wenn Funktionen über sense. aufgerufen werden: Der Variablen sense sind die Module aus SenseHat zu Beginn des Codes zugewiesen worden.

```
3  sense = SenseHat()
```

- ☐ Den aufgerufenen Funktionen werden die benötigten Parameter mitgegeben, oder es hat eine öffnende und schliessende Klammer.
  - Beispiel mit Parameter der mitgegeben wird

```
print(temp)
```

  - Beispiel ohne Parameter

```
sense.get_temperature()
```

- ☐ Die Parameter werden im richtigen Datenformat übergeben. Wo ein String übergeben wird, ist dieser in Anführungszeichen gesetzt. Hier bei der Übergabe des Dokumentnamens und des Modus der Funktion open sowie der Übergabe des Textes Hallo Welt! an die Funktion write.

```
file = open("Datafile.csv", "a")
file.write("Hallo Welt!")
```

- ☐ Die Befehle, welche innerhalb einer Schleife mehrmals ausgeführt werden sollen, sind eingerückt. So wie hier das Auslesen der aktuellen Temperatur und das schreiben dieser in eine Datei.

```
file = open("Datafile.csv", "a")

anzahl = 1
while anzahl <= 20:
  temp = sense.get_temperature()
  temp = round(temp,2)

  file.write(temp)
  file.write("\n")
  anzahl = anzahl + 1
```

- ☐ Die Reihenfolge der Befehle ist sinnvoll. So wird zum Beispiel zuerst die Temperatur ausgelesen und diese erst danach in eine Datei geschrieben.

- Beim write-Befehl kann die Fehlermeldung *TypeError: write() argument must be str, not float* auftreten. Die zu schreibende Variable kann in diesem Fall durch voranstellen von str(variable) in einen String umgewandelt werden. Beispiel anhand der Variable temp:
  ```
  file.write(str(temp))
  ```
- Wird nichts in das Dokument geschrieben, kann dies daran liegen, dass das Dokument nicht geschlossen wird. Das Dokument am Ende des Codes mittels file.close() schliessen:
  ```
  file.close()
  ```

Code-Beispiel

Untenstehend ein vollständiges Beispiel für einen Code der:

20-mal im Abstand von einer Sekunde die aktuelle Zeit, gemessene und gerundete Temperatur und um – 2.3 Grad korrigierte und gerundete Temperatur in eine csv-Datei mit dem Namen «Datafile» speichert.

```
1  from sense_hat import SenseHat
2  import time
3  sense = SenseHat()
4
5  file = open("Datafile.csv", "a")
6  file.write("Time, Temperatur, korr Temperatur \n")
7
8  anzahl = 1
9  while anzahl <= 20:
10   temp = sense.get_temperature()
11   temp = round(temp,2)
12   tempkorr = round(temp - 2.3,2)
13
14   file.write(time.strftime('%X'))
15   file.write(" ")
16   file.write(temp)
17   file.write(" ")
18   file.write(tempkorr)
19   file.write("\n")
20   anzahl = anzahl + 1
21   time.sleep(1)
22
23 file.close()
```

Es empfiehlt sich, in der Datei jeweils auf der ersten Zeile zu schreiben, welche Daten, dass in der entsprechenden Datei gespeichert werden (im Code erfolgt dies in Zeile 6). Dies erleichtert später die Auswertung, da sofort klar ist, um welche Daten es sich handelt.

Anstelle einer csv-Datei können auch andere Dateitypen erzeugt werden, so z.B. ein Word mittels der Endung .docx oder eine Excel-Datei mittels der Endung .xlsx.