jugend forscht

2022

„Vom Taschenrechner zum Computer und zurück“

Eine Anwendung damit Schülerinnen und Schüler auf dem Computer Algebra System von Texas Instruments komfortabel in Python Programme erstellen können.

Von Elias Freund (15 Jahre)

Erarbeitungsort: Halepaghen – Schule

Projektbetreuer/in: Frau Freund, Herr Schulze

Fachgebiet: Arbeitswelt

Bundesland: Niedersachsen

Die CAS Rechner, TI-Nspire CX-CAS von Texas Instruments bieten in einem zusätzlichen Modul die Programmiersprache Python an, um damit unter anderem Sensoren oder einen Rover in den entsprechenden Programmen einbinden zu können. Das Programmieren auf dem Taschenrechner ist aber aufgrund einer abc-Tastatur statt einer qwertz-Tastatur sehr umständlich und auch das Display zeigt nur wenige Zeilen Programmiercode an. Darum habe ich ein eigenes Python-Modul entwickelt, das es erlaubt, alle Funktionalitäten des TI-Nspire-Moduls auf dem Computer zu programmieren und anschließend wieder auf das Handheld zu übertragen. Zudem habe ich verständliche Fehlermeldungen eingebaut, die ein schnelleres Debugging erlauben. Zur schnellen und optimalen Nutzung und Anpassung habe ich das ganze Python-Modul in GitHub geladen, damit eine umfangreiche Verwendung dieses Moduls für das Handheld sichergestellt ist.

Inhalt

1. Informatik an Schulen

Nicht zuletzt die Corona-Pandemie hat den Ruf nach mehr Digitalisierung in Schulen lauter werden lassen. Bereits 2012 hat die Kultusministerkonferenz in ihrem Beschluss vom 08.03.2012 einen Rahmen für die Medienbildung an deutschen Schulen beschlossen[1]. Darin wird beschrieben, dass der Medienbildung in Schulen ein wichtiger Platz eingeräumt werden soll. Allerdings kritisiert die Gesellschaft für Informatiker, dass sich in diesem Bereich noch zu wenig getan hat, während die Technik fortgeschritten ist und so größere Lücken entstehen, die zu langsam bekämpft werden.[2].

1.1. Informatikunterricht an den Schulen

In der einheitlichen Prüfungsordnung Informatik[3] für die gymnasiale Oberstufe formuliert die KMK die Wichtigkeit des Informatikunterrichts wie folgt: „Der Informatikunterricht in der gymnasialen Oberstufe leistet einen spezifischen Beitrag zur Allgemeinbildung, indem er den Erwerb eines systematischen, zeitbeständigen und über bloße Bedienerfertigkeiten hinausgehenden Basiswissens über die Funktionsweise, die innere Struktur sowie die Möglichkeiten und Grenzen von Informatiksystemen ermöglicht.“[4] Damit ist den deutschen Schulen mit gymnasialer Oberstufe eine Möglichkeit geboten, den Schülerinnen und Schülern in einem definierten Format Informatikunterricht so anzubieten, dass sie in diesem Fach eine entweder schriftliche oder mündliche Abiturprüfung ablegen können.

Niedersachsen ist laut einer Studie der Universität Rostock das einzige Bundesland, das in der Sekundarstufe II Informatikunterricht sowohl auf grundlegendem als auch auf erhöhtem Anforderungsniveau anbietet und den Schülerinnen und Schülern die Möglichkeit bietet, sowohl eine schriftliche oder auch mündliche Abiturprüfung abzulegen (vgl. INFORMATIKUNTERRICHT IN DEUTSCHLAND – EINE ÜBERSICHT, S. 8)[5]. An dieser Stelle muss aber angemerkt werden, dass dieses Angebot immer von der aktuellen Verfügbarkeit einer Fachschaft Informatik an den Schulen abhängt. Laut AVO-GOBAK §5, Abs 1 muss eine Prüfungskommission für ein Unterrichtsfach immer aus drei Mitgliedern bestehen, die die Befähigung für das Lehramt Gymnasium oder das Lehramt an berufsbildenden Schulen besitzen (vgl. S. 6)[6].

Betrachtet man die Situation des Informatikunterrichts in der Mittelstufe, so ist das Bundesland Mecklenburg-Vorpommern hier am besten aufgestellt. In diesem Bundesland besuchen die Schülerinnen und Schüler, sowohl in Schulen mit und ohne angeschlossene Oberstufe, den Informatikunterricht in den Jahrgängen 5 – 10 jeweils einstündig verpflichtend (vgl. S. 5)[7]. Die Situation in Niedersachsen soll sich hier jedoch spätestens ab dem Schuljahr 2023/24 ändern. Ab diesem Schuljahr ist auch in Niedersachsen geplant, den Informatikunterricht als Pflichtfach in der Sekundarstufe I einzuführen[8].

---

1 https://www.kmk.org/fileadmin/Dateien/veroeffentlichungen_beschluesse/2012/2012_03_08_Medienbildung.pdf

2 https://gi.de/themen/beitrag/informatikunterricht-in-deutschland-ein-flickenteppich-mit-loechern

3 https://www.kmk.org/fileadmin/veroeffentlichungen_beschluesse/1989/1989_12_01_EPA_Informatik.pdf

4 ebd

5 https://www.lkv.uni-rostock.de/storages/uni-rostock/Alle_IEF/Inf_PI/files/Vergleich_IU_2020_2020-11-23.pdf

6 https://www.mk.niedersachsen.de/download/136515/AVO-GOBAK_und_EB-AVO-GOBAK_August_2018.pdf

7 https://www.lkv.uni-rostock.de/storages/uni-rostock/Alle_IEF/Inf_PI/files/Vergleich_IU_2020_2020-11-23.pdf

8 https://www.mk.niedersachsen.de/startseite/aktuelles/presseinformationen/informatik-wird-ab-dem-schuljahr-2023-2024-pflichtfach-weitere-qualifizierungskurse-fur-lehrkrafte-starten-184807.html

In Folgenden werde ich mich auf den Teil des Programmierens im Informatikunterricht beschränken, da das Programmieren ein wesentlicher Teil des Informatikunterrichts darstellt.

1.2. Programmiersprachen als Teil des Informatikunterrichts

Bereits ab der Grundschule gibt es Angebote, das Programmieren als Teil des Informatikunterrichts zu lernen. Hier gibt es zum Beispiel von App-Camps bereits kostenlose Materialien ab Klassenstufe 3[9]. In diesem Beispiel, dem Calliope Mini, arbeitet man mit einer Programmiersprache, die große Ähnlichkeiten mit der bekannteren Programmiersprache Scratch aufweist. Da Scratch[10] als bildungsorientierte, visuelle Programmiersprache bekannter ist, werde ich mich im Folgenden auf sie beziehen. Das Ziel von Scratch ist es, den Schülerinnen und Schüler die Grundlagen des Programmierens, wie zum Beispiel das analytische Denken, beizubringen und zu lernen, wie ein Computer ein Programmcode liest.

Die Programmiersprache, die im Beispiel von App-Camps bei dem Calliope Mini genutzt wird, ist Open-Roberta. Es ist ähnlich wie Scratch als visuelles Interface zur richtigen Programmiersprache aufgebaut und funktioniert wie Scratch durch das Zusammenschieben von Code-Blöcken. So findet man den Hinweis, dass Scratch das Ziel verfolgt, „Neueinsteiger – besonders Kinder und Jugendliche – mit den Grundkonzepten der Programmierung vertraut zu machen. Unter dem Motto imagine, program, share („Ausdenken, Entwickeln, Teilen“) wird die kreative und explorative Erstellung eigener Spiele und Multimedia-Anwendungen, verbunden mit dem gegenseitigen Austausch darüber, als Motivation genutzt.“ [11]

Betrachtet man nun die Unterrichtsmaterialien für den Informatikunterricht in der Sekundarstufe I, wie etwa „einfach INFORMATIK“ vom Klett-Verlag[12] so findet man in diesem, wie auch in anderen Unterrichtsmaterialien, Python als Programmiersprache. Eine benutzerfreundliche und frei zugängliche Entwicklungsumgebung wird den Schülerinnen und Schülern mithilfe von tigerJython[13] angeboten. Der wesentliche Unterschied zu Scratch oder Open Roberta ist hier, dass Python nicht nur über ein grafisch aufbereitetes Interface bedient wird. Dass die Wahl auf Python als Einstiegsprogrammiersprache gefallen ist, erscheint logisch, da Python derzeit als die beliebteste Programmiersprache[14] gilt. Dies liegt unter anderem daran, dass Python mehrere Programmierparadigmen, z. B. die objektorientierte, die aspektorientierte und die funktionale Programmierung unterstützt[15]. Die Entwicklung von Python startete vor etwa 30 Jahren und hatte als Ziel, eine Programmiersprache zu entwerfen, einfach und übersichtlich ist. Um dies zu erreichen, wurden zwei wesentliche Maßnahmen verfolgt: Zum einen kommt die Sprache mit relativ wenigen Schlüsselwörtern aus[16]. Zum anderen ist die Syntax reduziert und auf Übersichtlichkeit optimiert. Dadurch lassen sich Python-basierte Skripte deutlich knapper formulieren als in anderen Sprachen.[17]

Ein weiterer, wesentlicher Vorteil von Python als Programmiersprache in Schulen ist, dass die Schülerinnen und Schüler bereits eine Programmiersprache erlernen können, die auch in der Arbeitswelt eine sehr hohe Akzeptanz und Beliebtheit erfährt. So ist Python laut Statista die beliebteste

---

[9] https://appcamps.de/unterrichtsmaterial/calliope-mini/
[10] https://de.scratch-wiki.info/wiki/Sch%c3%bcler-_und_Lehreraccounts
[11] https://de.scratch-wiki.info/wiki/Scratch#Ph.C3.A4nomen_Scratch
[12] Juraj Hromkovic, Tobias Kohn, „einfach INFORMATIK – Sekundarstufe I“, Ernst Klett Verlag, Stuttgart Leipzig, 1. Auflage
[13] https://tigerjython.de/de
[14] https://de.statista.com/statistik/daten/studie/678732/umfrage/beliebteste-programmiersprachen-weltweit-laut-pypl-index/
[15] https://de.wikipedia.org/wiki/Python_(Programmiersprache)
[16] https://docs.python.org/3/reference/lexical_analysis.html#keywords
[17] Marty Alchin: Pro Python. Hrsg.: Apress. 2010,

Programmiersprache weltweit mit mehr als 30% aller Anwendungen[18]. Für die Schülerinnen und Schüler ist es auch leicht, sich zusätzliche Informationen zu Python zu beschaffen, da es für kaum eine andere Programmiersprache so viele Foren, YouTube-Tutorials und Standardlibraries (wie etwa https://docs.python.org/3/library/index.html) gibt, wie für Python.

1.3. Hardwareanforderungen für das Programmieren von Python

Nachdem ich nun die Vorteile von Python als Programmiersprache aufgezeigt habe, möchte ich im Folgenden die Hardwareanforderungen untersuchen. Besonders in Schulen stehen immer noch zu wenig Computer, Tablets oder andere Geräte zur Verfügung, und wenn Geräte zur Verfügung stehen, dann selten nur für alle Schüler im Klassenzimmer. Meistens ist auch der Zugang zu vorhandener Hardware zeitlich begrenzt[19], was ein Arbeiten mit Computern oder anderen Geräten zusätzlich erschwert. Daher wird andere Hardware benötigt, die einige Anforderungen erfüllen muss. Zuerst einmal muss diese allen Schülerinnen und Schüler zeitlich unbegrenzt zur Verfügung stehen. Dies ist wichtig, damit ein Arbeiten mit solchen Geräten für zum Beispiel Hausaufgaben möglich ist. Des Weiteren müssen alle ihr eigenes, bzw. persönliches Gerät zur Verfügung stehen haben, da nur so ein individuelles Lernen möglich ist.

Weil Python eine Interpreter-Sprache ist, das bedeutet, dass die Abfolge von Programmanweisungen in Maschinencode übersetzt werden, und nicht erst in Maschinencode geschrieben werden müssen, benötigt es nur eine geringe Menge an Speicherplatz. Zudem ist auch keine bzw. keine performante Internetanbindung ausreichend. Alle anderen Anforderungen sind lediglich ein Display, auf dem man den Code gut leserlich erkennt und eine Tastatur, mit der der Nutzer umgehen kann. Eine Maus wird im Gegensatz zu Scratch und Open-Roberta nicht benötigt. Es genügen also schon weniger leistungsfähige Endgeräte, damit die Schülerinnen und Schüler Python als Programmiersprache erlernen können. Die Anforderungen an CPU und RAM sind auch minimal. Dies alles muss bei der Wahl der Programmiersprache an Schulen mitberücksichtigt werden, da generell nicht davon ausgegangen werden darf, dass alle Schülerinnen und Schüler sich privat einen leistungsstarken PC anschaffen. Außerdem wird Python von sehr vielen Entwicklungsumgebungen unterstützt, wie zum Beispiel dem bereits erwähnten tigerJython.

1.4. Ein weiteres Endgerät, um mit Python zu programmieren

In den Kapiteln 1.2. und 1.3. habe ich beschrieben, warum Python eine geeignete Programmiersprache ist, um an Schulen für den Unterricht verwendet zu werden. Obwohl, wie in 1.3. beschrieben, die Hardwareanforderungen für Python gering sind, ist trotzdem ein Endgerät wie etwa ein PC oder Laptop zum Programmieren notwendig. An dieser Stelle möchte ich anmerken, dass der Einsatz von iPads, die inzwischen an meiner wie auch vielen weiteren Schulen als digitale Endgeräte verwendet werden, zwar eine Programmierumgebung für Python anbieten, diese aber vor allem für das Ausführen von Kurzbefehlen und Scripts gedacht ist[20]. Zudem wird bei der Entwicklungsumgebung von Python auf dem iPad kritisiert, dass die Eingabe von Sonder- und Steuerzeichen ohne externe Tastatur umständlich und langwierig ist. Es wäre also schön, wenn es noch eine weitere Art von Endgerät gäbe, mit denen Schülerinnen und Schüler Programme mit Python erstellen können. Texas Instruments bietet mit seinem CAS-Taschenrechners TI-Nspire CX-CAS und der Erweiterung einer Python-Programmierumgebung genau so ein Endgerät an. Im nächsten Kapitel werde ich diesen CAS-Taschenrechner genauer vorstellen.

---

[18] https://de.statista.com/statistik/daten/studie/678732/umfrage/beliebteste-programmiersprachen-weltweit-laut-pypl-index/

[19] https://de.statista.com/statistik/daten/studie/180361/umfrage/ausstattung-der-schulen-mit-computern/

[20] https://apps.apple.com/de/app/pyto-python-3/id1436650069

2. Der TI-Nspire CX-CAS

Generell versteht man unter einem CAS ein Computeralgebrasystem, „das der Bearbeitung algebraischer Ausdrücke dient. Es löst nicht nur mathematische Aufgaben mit Zahlen (wie ein einfacher Taschenrechner), sondern auch solche mit symbolischen Ausdrücken (wie Variablen, Funktionen, Polynomen und Matrizen).“[21] Der Einsatz eines CAS Taschenrechners ist bereits an vielen Schulen, wie auch meiner Schule, schon etabliert. In Niedersachsen soll ab dem Schuljahr 2022/23 im Sekundarbereich I des Gymnasiums, sowie an beruflichen Gymnasien aufsteigend ab der Einführungsphase die Nutzung eines CAS verbindlich eingeführt werden. Ab Schuljahr 2026/27 soll dies für alle Schülerinnen und Schüler der Fall sein, die aufsteigend ab der Einführungsphase an einer öffentlichen allgemeinbildenden Schule das Abitur anstreben.[22] Somit ist also in den nächsten Jahren damit zu rechnen, dass alle Schülerinnen und Schüler mit einem CAS ausgestattet sein werden. Diese verbindliche Einführung bietet gleichzeitig allen Schülerinnen und Schülern eine Programmierumgebung in Python an, da auf allen CAS von Texas Instruments das Python-Modul standardmäßig vorhanden ist.

Im Folgenden werde ich den TI NSpire CX CAS genauer beschreiben.

2.1. Der Handheld und die PC-Software

Der TI-Nspire CX-CAS ist das Handheld, also der Taschenrechner an sich. Er besteht aus einem Display, einem Nummernfeld sowie weiteren Tasten für mathematische Sonderzeichen und einer Tastatur.

Das Display hat, um in das Handheld zu passen, nur eine Größe von ungefähr 3 Zoll. Diese Größe ist ausreichend, wenn man Werte in eine Tabelle eintragen möchte, oder wenn man einen mathematischen Term berechnen möchte. Allerdings sind 3 Zoll nicht genug, um ein Programm zu schreiben, da dies dazu führt, dass man nicht mehr als 5 Zeilen auf einmal sehen kann und dementsprechend sehr viel mit scrollen beschäftigt ist und man versuchen muss sich das ganze Programm zu merken (bzw. wo was im Code steht) um eventuelle Änderungen vornehmen zu können.

Das Nummernfeld, als Hauptelement des Taschenrechners, ist groß genug um alle Operationen, bzw. Features mit ihm gut ausführen zu können. Die Tastengröße beträgt hier ungefähr $4\ mm$ x $6\ mm$.

Die Sonderzeichentasten sowie die normalen Tastaturasten sind allerdings zu klein. Diese haben nur ungefähr eine Größe von $2\ mm$ x $2\ mm$, was das Tippen deutlich erschwert. Des Weiteren sind die Tasten auf der Tastatur im „abc“-Format, bzw. in der „abc“-Reihenfolge, was ebenfalls das Programmieren erschwert. Das liegt daran, dass viele Nutzer an ganz normale Computer- oder iPad Tastaturen gewöhnt sind, die sich im „$qwertz$“-Format befinden. Um mit dieser Tastatur also zu arbeiten, muss der Nutzer sich erst an diese „abc“-Tastatur umgewöhnen, nur um das Handheld zu bedienen, da kaum andere Systeme eine solche Tastatur verwenden.

Beim Kauft des Handheld erhält man zusätzlich einen Lizenzschlüssel für die Computersoftware von Texas Instruments. Diese kann man sich dann kostenlos auf einen PC oder Laptop herunterladen und mit dem Lizenzschlüssel aktivieren. Das Handheld kann mit einem passenden, mitgelieferten Kabel mit dem PC verbunden werden, um die Features zu nutzen. Mit dieser Software kann man zum einen auf die auf dem Handheld gespeicherten Dateien zugreifen, diese kopieren, löschen oder erstellen und sie natürlich lesen. Zum anderen kann man diese Dateien auch direkt bearbeiten und hat dann ein virtueller Handheld, auf dem man die entsprechenden Tasten bedienen kann. Allerdings gibt es hier keine Syntax-Hervorhebung, die den Anwender beim Programmieren unterstützen kann. Ein Feature

[21] https://de.wikipedia.org/wiki/Computeralgebrasystem

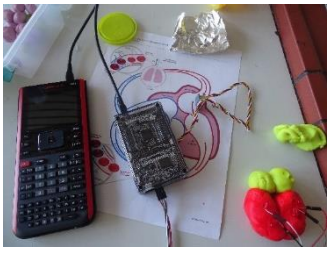
[22] https://www.mk.niedersachsen.de/download/172245/Erlassentwurf_zur_Verwendung_von_Taschenrechnern_im_Abitur.pdf

zum Debugging gibt es auf der PC-Software von Texas Instruments ebenfalls nicht. Man kann nicht einmal das Programm ausführen, weder auf dem PC noch auf dem Handheld, solange das Handheld noch am PC angeschlossen ist.

2.2. Rover und TI Innovator Hub

Texas Instruments hat zusätzlich zu dem eben genannten Handheld einen Rover entwickelt. Auf den Rover kann der Taschenrechner als Steuergerät integriert werden, wie in dem nebenstehenden Bild gezeigt. Mit den geeigneten Programmen auf dem Taschenrechner kann der Rover zum Beispiel Funktionsgraphen nachfahren, die die Schülerinnen und Schüler anschließend untersuchen können. Der Rover kann aber dank eines eingebauten „TI Innovater Hub“ noch viel mehr. So kann er zum Beispiel mithilfe eines Farbsensors eine gesuchte Farbe unter ihm erkennen und darüber halten. Der Farbsensor ist aber nicht der einzige Sensor, der an diesen angeschlossen werden kann. Neben Ultraschallsensoren, die direkt im Rover verbaut sind, können noch viele weiter Sensoren verwendet werden, wie etwa Sensoren für Feuchtigkeitsmessungen, Lautstärke usw. Außerdem können auch Steuersensoren und LEDs angeschlossen werden. Das nebenstehende Bild zeigt die Steuerung eines mit Knetmasse modellierten Herzens mithilfe des Taschenrechners und des Innovator Hubs sowie passender Sensoren für etwa den Biologieunterricht. Zudem können die Messwerte der Sensoren auf dem Taschenrechner empfangen und weiterverarbeitet werden. Dabei bietet Texas Instruments zwei Programmiersprachen an, mit der der Rover und der Innovator Hub mit seinen Sensoren bedient werden können. Die zuerst implementierte Programmiersprache war TI Basic. Ab dem Firmware Update 5.2 (Stand September 2020)[23] bietet Texas Instruments mit Python 3.4.0 auch eine Entwicklungsumgebung für diese Programmiersprache an. Im nächsten Kapitel werde ich die Programmiersprache TI Basic analysieren.

TI Rover mit CAS - eigene Aufnahme



CAS mit Innovator Hub - eigene Aufnahme

2.3. TI Basic

Bevor Texas Instruments ihre Funktionen für den Rover oder das Handheld die Entwicklungsumgebung Python hinzugefügt hat, mussten die Scripts bzw. Programme in der selbstentwickelten Sprache „TI Basic“ geschrieben werden. TI Basic hatte allerdings als Programmiersprachen einige große Nachteile gegenüber anderen Sprachen, wie zum Beispiel Python. TI Basic wurde eigens für den Rover, bzw. die Programme auf dem Handheld erstellt. Das bedeutet, dass diese Programmiersprache nirgendwo anders nutzbar ist als auf dem Handheld, was das Lernen von ihr für diesen einen Zweck eine sehr hohe Anforderung an den Nutzer darstellt. Zweitens gab es kaum Informationen oder Dokumentationen über Fehler oder andere Probleme, was das Debugging deutlich erschwert und verlangsamt hat. Da es kaum Dokumentationen zu Problemen / Fehlern gab blieb nur das „Try-and-Error“ als Möglichkeit, Probleme zu lösen. Außerdem konnte man aufgrund des kleinen Programmierteams auch nicht so einfach den Support kontaktieren, wenn es Probleme oder Fragen gab, die man selbst nicht lösen konnte.

2.4. Python auf dem CAS

Aufgrund dieser Probleme hat Texas Instruments sich dafür entschieden, ihre Funktionen und Scripts Python kompatibel zu machen. Dies löste sehr schnell die oben genannten Probleme, da Python, wie

[23] https://de.wikipedia.org/wiki/TI-Nspire

in Kapitel 1.2. bereits erwähnt, eine sehr bekannte Programmiersprache mit entsprechend viel, bzw. guter Dokumentation und Hilfe bei Problemen ist.

Allerdings löst das allein nicht das Problem, da es noch kaum Debugging Features zu den Funktionen von Texas Instruments gibt. Außerdem ist die Eingabe trotzdem noch schwierig und unkomfortabel. Es gibt zwar die Möglichkeit über das Bedienen einer Ziffernfolge, sich durch Menüs und Untermenüs in denen die Funktionen als Template bzw. Vorlage verfügbar sind zu navigieren, allerdings ist auch dies recht umständlich. Das liegt daran, dass es sehr viele Untermenüs und teilweise auch Untermenüs zu den Untermenüs gibt, was die Orientierung in ihnen deutlich erschwert. Des Weiteren muss man sich dadurch merken, wo welche Funktion ist, wobei einige Funktionen in verschiedenen Untermenüs dieselben Namen haben, was die Orientation noch weiter erschwert. Zusätzlich gibt es keinerlei Beschreibung oder Erklärungen zu den Funktionen. Das gilt sowohl für das Handheld als auch für die Software auf dem PC.

3. Das Python Modul für das Handheld auf dem PC

Aus den im obigen Kapitel genannten Gründen habe ich mich dazu entschieden eine Python Library, bzw. ein Python Modul zu erstellen, dass sowohl das Schreiben als auch das Testen bzw. Debugging von Programmen und Code einfacher und komfortabler gestalten soll. Wie bereits in Kapitel 1.2. beschrieben, ist Python eine geeignete Programmiersprache für den Informatikunterricht. Da Texas Instruments auf dem Handheld sowohl TI Basic als auch Python integriert hat, habe ich mich aufgrund der genannten Punkte in 2.3. und 2.4. dazu entschieden, das Modul mit bzw. für Python zu schreiben. Außerdem erlaubt Python mit einem integrierten Library bzw. Package Installationssystem eine einfachere Anwendung bzw. Installation des Moduls, was es für den Informatikunterricht noch attraktiver macht. Wie mein Vorgehen dabei war, werde ich in den folgenden Kapiteln beschreiben.

3.1. Die Add-ons von TI

Wie in 2.2. erwähnt, hat Texas Instruments eigene Funktionen, wie zum Beispiel für das Messen von Messdaten über Sensoren oder das Fahren des Rovers, erstellt. Diese Funktionen haben gewisse imports für die verschiedenen Module, wie zum Beispiel „ti_hub“ oder „ti_rover“. Die Funktionen in den Modulen haben dann ihre eigenen Argumente und Ergebniswerte. Ein Beispiel für ein Modulimport ist „import ti_rover as rover“. Nach Python Syntax würde das Programm dann in einem IDE (Integrated Development Environment) wie folgt aussehen:

test.py

```
1   rover.forward(1)
2   rover.left(90)
3   text_at(1, "text", "center")
```

IDEs sind Computerprogramme, „mit denen die Aufgaben der Softwareentwicklung möglichst ohne Medienbrüche bearbeitet werden können.“[24]. Dabei sind die gelben Hervorhebungen im Falle des IDEs, das ich nutze, Warnungen, da das IDE die Funktionen nicht kennt und daher denkt, dass es sich hierbei um Fehler handelt. Bei einer Ausführung dieses Programmes bekommt man folgenden Fehler:

```
PS D:\Dateien\Dev\TI Python Module> & C:/Users/Elias/AppData/Local/Microsoft/WindowsApps/python3.10.exe "d:/Dateien/Dev/TI Python Module/test.py"
Traceback (most recent call last):
  File "d:\Dateien\Dev\TI Python Module\test.py", line 1, in <module>
    rover.forward(1)
NameError: name 'rover' is not defined
```

[24] https://de.wikipedia.org/wiki/Integrierte_Entwicklungsumgebung

Dieser Fehler sagt aus, dass das Modul oder die Klasse ‚rover' nicht definiert ist, oder dass das Programm das Modul oder die Klasse nicht gefunden hat, und es daher nicht weiß, was es mit dieser Anweisung tun soll. Solche Warnungen bzw. Fehler sind hinderlich, da IDEs dem Nutzer durch einige Features das Programmeschreiben erleichtern können und sollen. Solche Features sind zum Beispiel Syntaxhervorhebung, Programmausführung im IDE, automatische Vervollständigungen und das Anzeigen von Funktions- oder Klassenbeschreibungen.

3.2. Zusätzliche Features für das Python Modul

Für meine Library habe ich mir als erstes überlegt, welche der Module von Texas Instruments ich implementieren kann und welche nicht. Das Modul „ti_image" konnte ich nicht implementieren, da dieses in der Funktion „createImage" einen Datentyp benötigt, der mittlerweile nicht mehr genutzt wird. Die restlichen Funktionen des Moduls sind implementierbar, jedoch habe ich mich dagegen entschieden dies zu tun, da die Funktion „createImage" die Funktion des Moduls ist, mit deren Ergebnis die anderen Funktionen arbeiten und so ohne diese Funktion nutzlos sind.

Damit die Module mit den Funktionen, die ich erstelle, später einfach und komfortabel installierbar sind, müssen sie in einem „Package" sein. Dieses habe ich „ti_python_module" aufgrund der Duck-Typing[25] Namenskonvention.

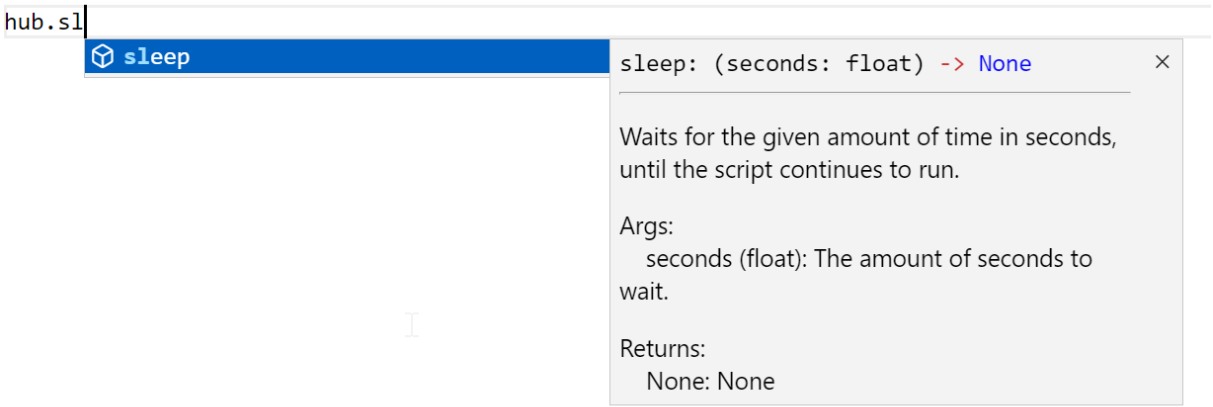
Die Imports für die einzelnen von mir Implementierten Module sehen dann so aus:

test.py

```
1  from ti_python_module import ti_hub as hub
2  from ti_python_module import ti_rover as rover
3  from ti_python_module import ti_system as sys
4
```

Dies hat mehrere Vorteile. Einer davon ist der einfachere Transfer des Codes auf das Handheld, mehr dazu in Kapitel 4. Außerdem erkennt das Programm die Module wie „ti_hub" als vorhandene Module an und bietet entsprechend Vervollständigungen und Syntax Hervorhebungen als Features.

Des Weiteren habe ich alle Funktionen mit den Beschreibungen der offiziellen Dokumentation ausgestattet, sowie die einzelnen Argumente benannt bzw. beschrieben. Das sieht in dem DIE, das ich nutze so aus:



Ich habe außerdem zum einfacheren Debugging eigene Fehlermeldungen erstellt. Diese sieht man dann in der Konsole und das Programm wird unterbrochen. Als Beispiel, die Funktion „text_at"

[25] https://de.wikipedia.org/wiki/Duck-Typing

benötigt drei Argumente. Eine Zeile von 1 – 13, den Text zum Anzeigen und eine Ausrichtung. Wenn man jetzt eine ungültige Zeile angibt, passiert folgendes:

```
  File "d:\Dateien\Dev\TI Python Module\ti_python_module\err.py", line 18, in range_error
    raise ValueError("ERROR: Parameter <" + str(parameter) + "> has to be in range between '" + str(rangeMin) + "' and '" + str(rangeMax) + "'") from None
ValueError: ERROR: Parameter <16> has to be in range between '1' and '13'
```

Man sieht dabei in der Konsole, wo der Fehler ist (in diesem Fall in Zeile 3) und was der Fehler ist („ValueError: ERROR: Parameter <16> has to be in range between „1“ and „13““). Anhand einer solcher Fehlermeldung lassen sich Fehler im Programm viel einfacher erkennen als durch die Standard-meldung: „ValueError: Value out of bounds“.

Führt man ein Programm ohne Fehler aus, indem man zum Beispiel im Beispiel oben die Zeile auf 3 ändert, so sieht man folgendes in der Konsole:

```
Showing text 'Text' at line 1 with alignment 'center'
```

Dies hilft, auch in größeren Programmen nicht den Überblick zu verieren, was sonst passieren würde. Allerdings ist mir dabei aufgefallen, dass ein Output von den Ergebnissen in die Konsole bei langen Programmen unübersichtlich wird. Darum habe ich eine Funktion erstellt, die bei der Ausführung jedes Programmes den Nutzer fragt, ob er eine Log-Datei erstellen möchte. Dafür kann man ganz einfach eine Funktion, die einem in der Konsole mitgeteilt wird, ausführen. Diese wird nur einmal benötigt und erstellt eine Datei, in der der Nutzer angeben kann, ob er diese Log-Datei haben möchte oder nicht. Sobald man dies eingestellt hat, verschwindet die Nachricht in der Konsole, selbst wenn man eingestellt hat, dass man die Log-Datei nicht möchte. Wenn man die Log-Datei aktiviert, wird automatisch beim Ausführen einer beliebigen Funktion eine Log-Datei erstellt. Ist bereits eine Logdatei vorhanden, erkennt die Funktion das, und fügt den Log-Inhalt einfach als neue Zeile an die Log-Datei an. Das sähe im Beispiel von oben so aus:

```
TI Python Module Log File
Format:
[Date - Time] [Logtype] [Module] [Function] <log>

You can delete this file to empty the log file, since a new one will be generated
--------------------------------------------------------------------------------

[04-01-2022 12:22:56] [INFO]     [TI Hub]      [Text At]                    Showing text 'Text' at line 3 with alignment 'center'
```

Dies bietet die gewünschte Übersichtlichkeit auch bei größeren Programmen.

3.3. Der Import des fertigen Programms auf den Handheld

Nachdem der Anwender das Programm erstellt und getestet hat, muss es zur Ausführung wieder auf das Handheld. Dafür gibt es wie in Kapitel 2.1. bereits erwähnt einen mit dem Handheld gelieferten Lizenzschlüssel für eine Software von Texas Instruments. Mit dieser kann man das Handheld an den Computer anschließen und auf den Speicher vom Handheld zugreifen. Die Library ist so gemacht, dass man das ganze Programm ohne großen Aufwand aus dem IDE auf das Handheld kopieren kann, was zusätzlich den Programmierprozess vereinfacht. Lediglich die Import-Statements müssen verändert werden. Hier kann man bei allen den vorderen Teil („from ti_python_module“) bei allen Import-Statements löschen, da dies nur die Identifikation für die Library ist, die nicht im eigentlichen Programm vorhanden ist. Danach kann man einfach das ganze Programm kopieren und so auf den Handheld übertragen. Das sieht dann so aus:

```python
# Auf dem Computer zum Testen / Programm schreiben:
from ti_python_module import ti_hub as hub
from ti_python_module import ti_rover as rover

hub.text_at(3, "Test Programm", "center")
rover.forward(1)
rover.encoders_gyro_measurement()

# Programm für das Handheld

import ti_hub as hub
import ti_rover as rover

hub.text_at(3, "Test Programm", "center")
rover.forward(1)
```

Der untere Teil, also der Teil mit den angepassten Imports, wwird dann einfach in eine leere, zuvor auf dem Handheld erstellte Programmdatei kopiert werden. Diese kann auch über die PC-Software erstellt werden. Das Programm ist dann sofort ausführbar und fehlerfrei, da es ja schon im Vorhinein auf dem PC getestet wurde.

4. GitHub

Da ich immer noch an dem Projekt arbeite und auch weiter daran arbeiten möchte, brauche ich ein Repository, das mir erlaubt, meine Anpassungen in unterschiedlichen Versionsständen zu verwalten. Dieses Repository nutze ich in GitHub. GitHub ist ein „netzbasierter Dienst zur Versionsverwaltung für Software-Entwicklungsprojekte“[26]. GitHubs Hauptfähigkeit ist also, Code oder Programme zu speichern, um diese dann später mit anderen zu teilen oder sie zu veröffentlichen. Dazu muss man ein sogenanntes „Repository“ erstellen. In diesem wird der Code gespeichert und ist je nach Einstellung öffentlich einsehbar oder privat. Die Codedateien kann man dann mit einem Push in das Repository hochladen. Dabei gehen alte Versionen aber nicht verloren, sondern sie sind spätere immer noch verfügbar. GitHub bietet außerdem ein „Actions“ Feature an. Damit kann man seine eigenen Funktionen bzw. Aktionen definieren und bei bestimmten Events, zum Beispiel wenn ein Push passiert, ausführen.

Ich habe mich für mein Python Modul für ein GitHub Repository entschieden, weil bei GitHub, anders als bei ähnlichen Services wie z.B. SourceForge mehr Möglichkeiten zur Zusammenarbeit mit anderen besteht, was bei einer eventuellen Kooperation mit Texas Instruments sehr praktisch wäre. Außerdem habe ich zwei „Actions“ bzw. Aktionen implementiert. Die erste läuft, wann immer ein Push erfolgt und überprüft, ob die Datei, die das Programm als Modul deklariert, auch tatsächlich ein vorhandenes und valides Modul ist. Die zweite Aktion läuft immer, wenn ein sogenannter „Release“ erstellt wird. Releases in GitHub sind Versionsfestlegungen aller Dateien. Das bedeutet, dass das Repository neuere Dateien enthalten kann, der Release greift aber auf die angegebenen Dateiversionen zu. Diese zweite Aktion überprüft im ersten Schritt ebenfalls die Validität und das Vorhandensein der Moduldeklaration. Ist diese Überprüfung erfolgreich, wird die ganze Library auf https://pypi.org hochgeladen, um installierbar zu werden. Wie ich die Installierbarkeit des Moduls erreiche, wird im nächsten Kapitel beschrieben.

[26] https://de.wikipedia.org/wiki/GitHub

5. PyPI als Library bzw. Package Manager

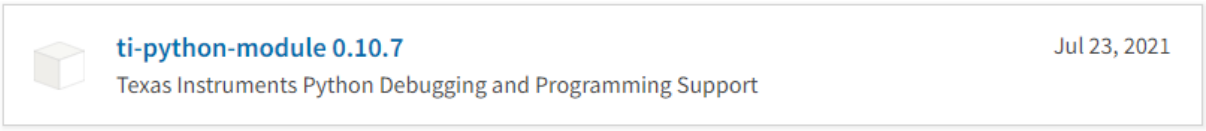
Ein Ziel meiner Arbeit ist es, den Schülerinnen und Schülern eine möglichst komfortable Programmierumgebung zur Verfügung zu stellen. Damit sollen sie im Informatikunterricht nicht nur bequem und schnell Python-Programme erstellen können, sondern diese auch für schulische Zwecke, wie etwa im Mathematikunterricht bei der Untersuchung von Funktionsgraphen oder auch in den Naturwissenschaften durch die Nutzung von Sensoren beim Experimentieren, anwenden können. Wie schon in Kapitel 1.3. geschrieben, gehe ich bei der Erstellung meines Python-Moduls von einer Minimalanforderung an die Hardware der Schülerinnen und Schüler aus. Ebenso möchte ich auch die Installation des Python-Moduls maximal benutzerfreundlich machen. Daher habe ich mich entschieden, dass ich das Modul als installierbares Package im Internet zur Verfügung stelle. Um das realisieren zu können, habe ich mich für PyPI[27] als Packageverwaltungsprogramm entschieden. Dabei steht PyPI für Python Package Index und wurde im April 2018 auf die Website Plattform „Warehouse“ geladen. Laut Python Software Foundation[28] waren es am 30. Oktober 2021 bereits mehr als 336 000 Packages, die in PyPI zum Download und zur Installation zur Verfügung stehen.

Ich habe PyPI als Paket bzw. Library gewählt, da eine Integration für PyPI in fast allen Python Installationen standartmäßig enthalten ist. Hier kann man einfach eine Library bzw. ein Paket hochladen und veröffentlichen. Dieser Upload funktioniert auch über Integrationen, zum Beispiel via GitHub, wie im letzten Kapitel angesprochen. Damit nun die Schülerinnen und Schüler die Library laden und installieren können, müssen sie lediglich einen einzigen, einfachen Befehl in der Konsole eingeben und ausführen. Dieser Befehlt gehorcht dem folgenden, einfachen Format: „pip install <Library ID>“. Die Library ID ist dabei eine eindeutige ID, die von PyPI vergeben wird, wenn das Package erstmalig vom Entwickler hochgeladen wird. Damit die Schülerinnen und Schüler eine aktualisierte Version herunterladen und installieren können, brauchen sie nur den gleichen Befehl mit dem Argument „--upgrade“ verwenden, also „pip install <Library ID> --upgrade“. Dadurch wird sowohl der Installation- wie auch der Aktualisierungsprozess einfach und verständlich wird.

Damit ich das Package für den Download und die Installation zur Verfügung stellen kann, muss ich zuerst meine Python-Programme in ein Paket, ein Package, „schnüren“. Dazu wird eine Datei mit dem Namen „setup.py“ benötigt. Diese Datei ist im Anhang abgebildet. In ihr stehen alle Details und Informationen, die benötigt werden, um dieses Package hochzuladen. Auf Basis dieser Datei erstellt eine Integration von GitHub dann das fertige Package und lädt es auf PyPI hoch, wo die zusätzlichen zur Installation benötigten Dateien erstellt werden.

Bevor ein Upload auf PyPI passieren kann, muss ein Projekt erstellt werden. Dabei muss der Besitzer des Projektes dann eine einzigartige ID eingeben. Diese ID wird benötigt, um den eigentlichen Upload zu machen und muss in der Funktion „setup“ unter dem Parameter „name“ vorhanden sein, damit eindeutig identifizierbar ist, zu welchem Projekt der Upload gehört.

Das Modul ist leicht über die Suchfunktion in https://pypi.org/ mit dem Stichwort „TI Python module“ zu finden:



[27] https://pypi.org
[28] https://en.wikipedia.org/wiki/Python_Package_Index

Die Details und die Beschreibung des Moduls sowie die Befehle für den Download und das Upgrade der Version findet man auch direkt unter https://pypi.org/project/ti-python-module/.

6. Fazit und Ausblick

Mit der in dieser Arbeit vorgestellten Lösung habe ich eine für Schülerinnen und Schüler komfortable Programmierumgebung für Python erstellt. Bei der Entwicklung dieser Umgebung habe ich mich an den Kriterien zu den Qualitätsmerkmalen von Software gemäß der internationalen Norm $ISO\ 25010$[29] orientiert:

Benutzerfreundlichkeit:
Der Anwender soll möglichst einfach alle Informationen und Beschreibungen zu den Funktionen und Argumenten erhalten. Das wird durch Dokumentationen, bzw. Docstrings[30] erreicht, die dem Anwender einfach erklären, was eine Funktion macht, welche Argumente sie braucht und welche Werte diese Funktion als Ergebnis ausgibt. Diese Dokumentationen sind weitestgehend dieselben, wie die in der offiziellen Dokumentation, um Verwirrung zwischen den beiden Quellen zu vermeiden.

Übertragbarkeit:
Das Modul soll für jeden Arbeitsbereich auf dem Computer funktionieren, ohne dass extra Schritte notwendig werden. Dies wird dadurch erreicht, dass das Modul als Ganzes Package installiert wird und so überall auf dem Computer vorhanden ist. Ein Updaten ist ebenfalls dadurch, dass das Modul ein Package ist, möglichst einfach möglich ist.

Effizienz:
Das Modul und seine Funktionen sollen möglichst performant laufen und nicht zu viel Ressourcen verbrauchen. Hier gibt es zum Beispiel das Feature mit der Log-Datei, bei der ich darauf geachtet habe, dass nicht für jeden Programmdurchlauf eine neue Log-Datei erstellt wird, sondern, dass das Programm kurz überprüft, ob eine entsprechende Datei schon vorhanden ist und den Log-Inhalt dann einfach an den Dateiinhalt anfügt.

Datenschutz:
Hier gibt es für mein Modul eher weniger Bedenken, da keine Nutzerrelevanten Daten zu irgendeinem Zeitpunkt aufgenommen oder gespeichert werden.

Somit habe ich ein Modul entwickelt, das in Schulen optimal für den Informatikunterricht eingesetzt werden kann. Zudem können (vgl. Kapitel 2.2.) etwa Experimente mithilfe von Sensoren auch in anderen Fächern durchgeführt werden. Dazu müssen die Python-Programme für die entsprechenden Standard-Experimente einmal zentral oder für individuelle Experimente bei Bedarf erstellt werden und dann auf die verwendeten Handhelds importiert werden. Dies wäre sogar eine kostengünstige Variante für die naturwissenschaftlichen Fächer, da die Kosten für einen TI Innovator Hub Für Schulen bei etwa 62Euro[31] liegt und die entsprechenden Sensoren, etwa von Grove[32], im Vergleich zu anderen Herstellern, etwa GoDirect von Vernier[33], kostengünstiger sind.

Als nächstes möchte ich noch das Modul „ti_image“ (vgl. Kapitel 3.2.) implementieren, um die Library komplett zu machen.

---

[29] https://www.iso.org/standard/35733.html
[30] https://www.programiz.com/python-programming/docstrings
[31] https://epsstore.ti.com/OA_HTML/ibeCCtpItmDspRte.jsp?section=11832&item=820555&sitex=10320:54534:US&XT=WEHH-S7J3-D0GH-6ZIV-UD53-SSFF-7FMU-DVV8
[32] https://www.seeedstudio.com/category/Sensor-for-Grove-c-24.html
[33] https://www.dynatech.de/messwerterfassung/sensoren-nach-erfassungsgerat/vernier-go-direct.html

Anhang: setup.py

setup.py > ...

You, 6 months ago | 1 author (You)

```python
import setuptools as st
import distutils.core as dc
import os
import io

VERSION = "0.10.7"

SUMMARY = "Texas Instruments Python Debugging and Programming Support"

here = os.path.abspath(os.path.dirname(__file__))

try:
    with io.open(os.path.join(here, 'README.md'), encoding='utf-8') as f:
        long_description = '\n' + f.read()
except FileNotFoundError:
    long_description = SUMMARY

st.setup(
    name="ti-python-module",
    version=VERSION,
    author="VampireTechnologyLord",
    author_email="<elias.freund@ewe.net>",
    description=SUMMARY,
    long_description=long_description,
    long_description_content_type='text/markdown',
    packages=st.find_packages(),
    install_requires=[],
    setup_requires=[
    'setuptools>=57.0.1',
    'wheel>=0.33.4'],
    keywords=["python", "ti", "texas instruments"],
    classifiers=[
        "Development Status :: 1 - Planning",
        "Intended Audience :: Developers",
        "Programming Language :: Python :: 3",
        "Operating System :: Unix",
        "Operating System :: MacOS",
        "Operating System :: Microsoft :: Windows",
        "Topic :: Utilities"
    ]
)
```