Über mich

- Peter
- Ich bin **kein** Python Profi!
- Ich bin Hobbyprogrammierer seit ca. 1980.
- Über die **UIFlow IDE** und **Blockly** bin ich zu **Micropython** gekommen.
- Ich wollte mein Schweizerkäse ähnliches Wissen zu Micropython auf eine **solide Basis** stellen.
- Der Kurs sollte mich bei der Stange halten.
- Zweck des Attraktor ist es **Know How** zu vermitteln.
- Um eine Programmiersprache zu lernen sollte man einen Kurs anbieten oder Artikel dazu schreiben.

Die Hardware

Ich habe mich für den M5Stick C plus als Hardware entschieden:

- Weil ich damit meine Erfahrungen gemacht habe.
- Weil die Firmware Micropython mit diversen Modulen enthält.
- Weil verschiedene Peripherie schon eingebaut ist und so die Probleme mit einem Steckbrett vermieden werden.

https://shop.m5stack.com/collections/m5-controllers/products/m5stickc-plus-esp32-pico-mini-iot-development-kit

https://docs.m5stack.com/en/core/m5stickc_plus

Die IDE

- Um in Python zu programmieren reicht ein einfacher Editor.
- Ein Integrated Development Envoirement (**IDE**) vereinfacht aber vieles.
- Ich habe mich für **Thonny** entschieden.
  - Mit der neusten Version 4.0.0 gibt es aber Probleme.
  - Deshalb werde ich Euch 3.3.13 zur Verfügung stellen.

Über Python

Python wurde Anfang der 1990er Jahre von Guido van Rossum geschrieben. Sein Ziel war es eine Programmiersprache zu schaffen die

- leicht erlernbar
- mächtig
- einfach
- gut lesbar

Ist.

Über Python

- Python ist eine interpretierte Sprache
  - Nachteil: langsam
  - Vorteil: Interaktive Nutzung möglich
- Python ist eine nicht typisierte Sprache
  - In neueren Python Versionen ist eine Typfestlegung möglich
  - Nachteil: Fehlerträchtig
  - Vorteil: einfach

Über Python

- Python selbst ist einfach und überschaubar.
- Python lebt von den vielen Modulen die es zur Erweiterung des Sprachkerns gibt.
- Python ist für fast alle Anwendungen geeignet, die nicht besonders schnell sein müssen.

Über Python

- Python ist objektorientiert programmiert.
- In Python kann man objektorientiert programmieren – muss es aber nicht.
- Im Kurs werden wir **nicht** objektorientiert programmieren.
- Im Python ist alles ein Objekt.
- Deshalb werden in Python immer wieder auf objektorientierte Konzepte stoßen.

Über Python

In Python wird sehr viel Wert auf die Lesbarkeit des Codes gelegt.

Deshalb gibt es da Zen von Python. Dieses enthält die Leitprinzipien für das Programmieren in Python.

In CPython ist es mit der Eingabe von

```
import this
```

Aufrufbar. Allerdings nicht in Micropython.

Das Zen von Python, von Tim Peters

- Schön ist besser als hässlich.
- Explizit ist besser als implizit.
- Einfach ist besser als komplex.
- Komplex ist besser als kompliziert.
- Flach ist besser als verschachtelt.
- Spärlich ist besser als dicht.
- Lesbarkeit zählt.
- Spezialfälle sind nie speziell genug, um die Regeln zu brechen.
- Dennoch geht Praktikabilität vor Regeltreue.
- Fehlermeldungen sollten niemals stillschweigend übergangen werden.
- Fehlermeldungen sollten niemals stillschweigend übergangen werden.
- Außer sie werden ausdrücklich ausgeblendet.

Das Zen von Python, von Tim Peters

- Im Falle von Uneindeutigkeit widerstehe der Versuchung zu raten.
- Es sollte einen -- und wirklich NUR einen -- offensichtlichen Weg es zu tun.
- Obwohl diese zunächst nicht offensichtlich ist, es sei denn Du bist Niederländer.
- Jetzt ist besser als nie.
- Obwohl es nie besser ist, als genau jetzt.
- Wenn die Implementierung schwer zu erklären ist, ist es eine schlechte Idee.
- Wenn die Implementierung einfach zu erklären ist, könnte es eine gute Idee sein.
- Namensräume sind eine tolle Idee -- machen wir mehr davon!

Über Python

- Daraus hat sich der Begriff **pythonisch** für einen pythongemäßen und gut lesbaren Programmierstil entwickelt.
- Der Stylguide für Python ist in **PEP 8** dargelegt.
- PEP (Python Enhancement Proposal) enthält Vorschläge für eine pythonische Anwendung der dort beschriebenen Elemente und Erweiterungsvorschläge.
- Eine deutsche Beschreibung von PEP8 findet Ihr z.B. hier: https://zrezai-dev.de/python/pep-8/

Über Micropython

- Micropython ist eine abgespeckte Version von CPyton.
- Deshalb gilt das was für CPyton gilt im Prinzip auch für Micropython.
- Bedingt durch den geringeren Speicher fehlen einige Dinge aus CPython.
- Micropython enthält aber zusätzliche Elemente um auf Ports, PWM oder ADC zuzugreifen.

Informationen zu Python bekommen

- Python Dokumentation: https://docs.python.org/3/
- Micropython Doku: https://docs.micropython.org/en/latest/
- Auf deutsch für V. 3.3: https://py-tutorial-de.readthedocs.io/de/python-3.3/
- M5Stack: https://docs.m5stack.com/en/quick_start/m5stickc_plus/mpy https://flow.m5stack.com/
- Diverse Seiten im Internet und auf YouTube

Der Kurs

Wie schon Eingangs erwähnt ist der Kurs aus meinem Bestreben entstanden mir eine solide Basis für die Programmierung mit Micropython zu schaffen. Das spiegelt dieser Kurs wieder. Ich habe mich mit den verschiedenen Elementen von Python auseinandergesetzt. Im Kurs werde ich Euch diese Elemente vorstellen. Das Lernen kann ich Euch nicht abnehmen. Ihr selbst müsst Euch anschließend damit beschäftigen, d.h. damit „**herumspielen**“. Ausprobieren was geht und was nicht geht.

Python eignet sich als Interpretersprache durch die interaktive Eingabemöglichkeit hervorragend dafür.

Themen die behandelt werden/können

- Datentypen
- Datenstrukturen
- Operatoren
- Kontrollstrukturen
- Funktionen
- Module
- Scripte
- Exceptions
- Garbage Collection
- Das interne Dateisystem

- M5-GUI
- M5-Taster
- M5-GPIO
- M5-Units-I2C
- M5-ESPNow
- M5-Wlan
- M5-WebServer
- M5-MQTT
- M5-Thread

Datentypen

Übersicht

Python kennt die folgenden Datentypen:

int - ganze Zahlen
float - gebrochene Zahlen
complex - komplexe Zahlen
bool - Wahrheitswerte
none – nichts
str - Zeichenketten

Datenstrukturen - Übersicht

Python kennt folgende Datenstrukturen:

- **Konstanten**
- **Variablen**
- **Listen**
- **Dictionaries**
- **Strings**
- **Sets**
- **Tuples**

Operatoren - Übersicht

- Arithmetrische Operatoren
- Vergleichsoperatoren
- Binäre Operatoren
- Bit-Operatoren
- Boolsche Operatoren
- Logische Operatoren
- Unäre Operatoren

Kontrollstrukturen

Python kennt die folgenden Kontrollstrukturen:

if – elif – else

for

while

Eingebaute Funktionen

Micropython bringt eingebaute Funktionen mit. Hier eine kleine Auswahl:

```
abs()     bin()     bool()       chr()     dict()    enumerate()
filter()  float()   globals()    hex()     id()      input()
int()     len()     max()        min()     oct()     ord()
pow()     print()   range()      reversed()          round()
set()     sorted()  str()        sum()     tuple()      type()
```

Eigene Funktionen schreiben

Eine Funktion wird mit **def** erstellt. Es folgt der Name der Funktion und Klammern **()**. Diese sind ein Kennzeichen von Funktionen. In den Klammern werden die Parameter übergeben.

Mit **return** wird die Funktion beendet und der Rückgabewert angegeben.

```
>>> def quad(x):
        return x * x

>>> quad(5)
25


>>> def test():
        pass
```

Sammelbestellung

M5Stick C Plus:
https://www.berrybase.de/m5stack-stickc-plus-esp32-pico-mini-iot-dev-kit

USB A → C Kabel:
https://www.berrybase.de/usb-c-2.0-sync-ladekabel-a-stecker-c-stecker-schwarz?number=38675