Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES
Näher dran.

MMT Fakultät für Maschinenbau
und Mechatronik

Befehlszusammenfassung Python3 - Crashkurs

Stand: 03.12.2019

Hochschule Karlsruhe – Technik und Wirtschaft
Fakultät für Maschinenbau und Mechatronik

1 Python-Befehle

Diese Zusammenfassung stellt kurz die wichtigsten Sprachelemente für das Experimentallabor vor.

1.1 Bibliotheken und Module

Python bietet dem Benutzer eine umfangreiche Liste an Funktionen an. Diese sind jedoch nicht im Sprachkern von Python realisiert, sondern in Form von Modulen. Hierbei handelt es sich um eine einfache Python-Datei in der bereits Funktionen vorgeschrieben sind. Es ist somit auch möglich eigene Funktionen in einer Datei abzuspeichern und diese als Modul zu laden. Um Module in ein Python Programm einzubinden, stehen folgende Möglichkeiten zur Verfügung:

- `import modulname`: Diese Anweisung lädt das Modul in das Script. Auf die darin enthaltenen Funktionen kann mittels folgender Schreibweise zugegriffen werden: `modulname.funktionsname()`
- `import modulname as x`: Bei dieser Variante können alle Funktionen mittels folgender Schreibweise aufgerufen werden: `x.funktionsname()`. Hierbei wird der Tippaufwand bei langen Modulnamen minimiert und der Code übersichtlicher gehalten.
- `from modulname import f1, f2`: Bei dieser Anweisung werden nur die Funktionen f1 und f2 des Moduls gelesen. Diese können ohne das voranstellen des Modulnamens verwendet werden.
- `from modulname import *`: Importiert alle Symbole aus dem Modul, deren Name nicht mit __ beginnt (mit zwei Unterstrichen). Vorsicht: Hierbei können Variablen unbeabsichtigt überschrieben werden. Siehe dazu Namensräume in Python.

Nützliche Module zur Programmierung sind zum Beispiel:

- `math`: Einbindungen von Mathematischen Funktionen wie z.B. sin(x), exp(x)…
- `csv`: Ermöglicht den Datenzugriff auf eine CSV-Datei (Tabellendatei)
- `tkinter`: Erlaubt die Programmierung einer einfachen graphischen Benutzeroberfläche

1.2 Variablen und Datentypen

Im Gegensatz zu anderen Programmiersprachen, wird eine Python-Variable nicht auf einen bestimmten Typ eingeschränkt. Somit muss der Variablen bei der Deklaration auch kein bestimmter Typ zugewiesen werden. Python merkt sich den Datentyp und weiß auf welche Art von Daten eine Variable verweist. Dies führt jedoch dazu, dass es ohne weiteres Möglich ist eine Variable mit einem anderen Datentyp zu überschreiben. Jeder Variablen muss zudem ein Startwert zugewiesen werden, bevor sie in einer Anweisung ausgeführt werden kann. In der nachfolgenden Tabelle sind die gängigsten Datentypen aufgeführt:

Tabelle 1 Wichtige Datentypen in Python

Datentyp	Funktion	Beispiel	veränderlich
`int`	ganze Zahlen	x=3	nein
`float`	Fließkommazahlen	x=3.0	nein
`complex`	komplexe Zahlen	x=3+4j	nein
`bool`	boolesche Zahlen	x=bool(1)	nein
`str`	Zeichenketten	x='abc'	nein
`tupel`	Tupel	x=(1, 2, 3)	nein
`list`	Listen	x=[1, 2, 3]	ja

1.3 Operatoren

Die Operatoren entsprechen im Wesentlichen den meisten Programmiersprachen. Folgende zwei Auflistungen beinhalten die gebräuchlichsten Operatoren:

Tabelle 2 Rechen-, String- und Vergleichsoperatoren

Operator	Funktion
+ -	Vorzeichen
+ - * /	Grundrechenarten; / liefert in Python 3 `float`-Ergebnisse
//	ganzzahlige Division (25 // 8 ergibt 3)
%	Rest der ganzzahligen Division (25 % 8 ergibt 1)
**	Exponentialfunktion bzw. Hochstellen (2**8 ergibt 256)
+ *	Strings verbinden bzw. vervielfachen (‘ha‘*2 ergibt ‘haha‘)
\%	Zeichenkette formatieren
=	Zuweisung
+=	Zuweisung und Addition (var+=3 entspricht var=var+3)
-=	Zuweisung und Subtraktion
*=	Zuweisung und Multiplikation
/=	Zuweisung und Division
==	Gleichheit testen
!=	Ungleichheit testen
< > <= >=	kleiner, größer, kleiner-gleich, größer-gleich
is	testen, ob zwei Variablen auf das gleiche Objekt zeigen
is not	testen, ob zwei Variablen auf unterschiedliche Objekte zeigen

Tabelle 3 Binäre und logische Operatoren

Operator	Funktion
& |	binäres Und und binäres Oder
^	binäres Exklusiv-Oder
~	binäres Nicht
<<	binär nach links schieben (2<<4 ergibt 32)
>>	binär nach rechts schieben (768>>2 ergibt 192)
or	logisches Oder
and	logisches Und
not	logisches Nicht

1.4 if-Verzweigungen

Um bei einer bestimmten Eingabe eine entsprechende Ausgabe auszuführen, werden if-Verzweigungen verwendet. Der Interpreter läuft die Verzweigung von oben nach unten durch, bis zur ersten zutreffenden Bedingung und führt anschließend den zugehörigen Anweisungsblock aus. Diese Bedingungen werden dabei mit Vergleichsoperatoren, siehe Tabelle 2 und 3, gebildet. Falls keine der Bedingungen zutrifft, wird der else-Block ausgeführt. Dieser ist bei der if-Verzweigung optional und kann auch weggelassen werden. Dabei ist noch zu erwähnen, dass beliebig viele elif-Bedingungen erlaubt sind. Eine if-Verzweigung ist wie folgt aufgebaut:

```
if bedingung1:
    Anweisungsblock1
elif bedingung2:
    Anweisungsblock2
else:
    Anweisungsblock3
```

1.5 Schleifen

for-Schleifen

Bei der for-Schleife nimmt die Schleifenvariable (im nachfolgenden Beispiel i) der Reihe nach jedes der angegebenen Elemente an. Die Schleifenvariable wird dabei global angelegt und kann nach dem Ende der Schleife verwendet werden. Beispiel:

```
for i in [8, 9, 25]:
    print (i, end=' ')
print (', Variablenwert nach der Schleife: ', i)

# Ausgabe: 8 9 25, Variablenwert nach der Schleife: 25
```

while-Schleife

While-Schleifen werden wiederholt ausgeführt, solange die Bedingung erfüllt ist. Sie eigenen sich daher als Dauerschleife. Beispiel:

```
x=0
while x<=10:
    x=x+1
    print(x, end=' ')

# Ausgabe: 1 2 3 4 5 6 7 8 9 10 11
```

For und while-Schleifen können beide vorzeitig beendet (`break`) werden oder es kann auch der aktuelle Schleifendurchgang übersprungen werden (`continue`), wenn eine bestimmte Bedingung erfüllt ist. Beispiel:

```
while bedingung1:
    if bedingung2: continue      # überspringt anweisung1 und break
    anweisung1
    if bedinung3: break          # beendet die Schleife
```

Schleifen über Zahlenbereiche

Um Schleifen in einem vorgegebenen Zahlenraum auszuführen werden Elemente durch `range(Start, Ende)` erzeugt. Dabei ist der Endwert exklusiv, das heißt er wird nicht mehr berücksichtigt. Beispiel:

```
for i in range(1, 10):
    print(i,", ")

# Ausgabe: 1, 2, 3, 4, 5, 6, 7, 8, 9,
```

Alternativ kann diese Variante noch durch die Schrittweite mittels `range(Start, Ende, Schrittweite)` ergänzt werden.

```
for i in range(0, 20, 3):
    print(i,", ")

# Ausgabe: 0, 3, 6, 9, 12, 15, 18,
```

2 Raspberry Pi

2.1 Pinbelegung des Raspberry Pi 2B+

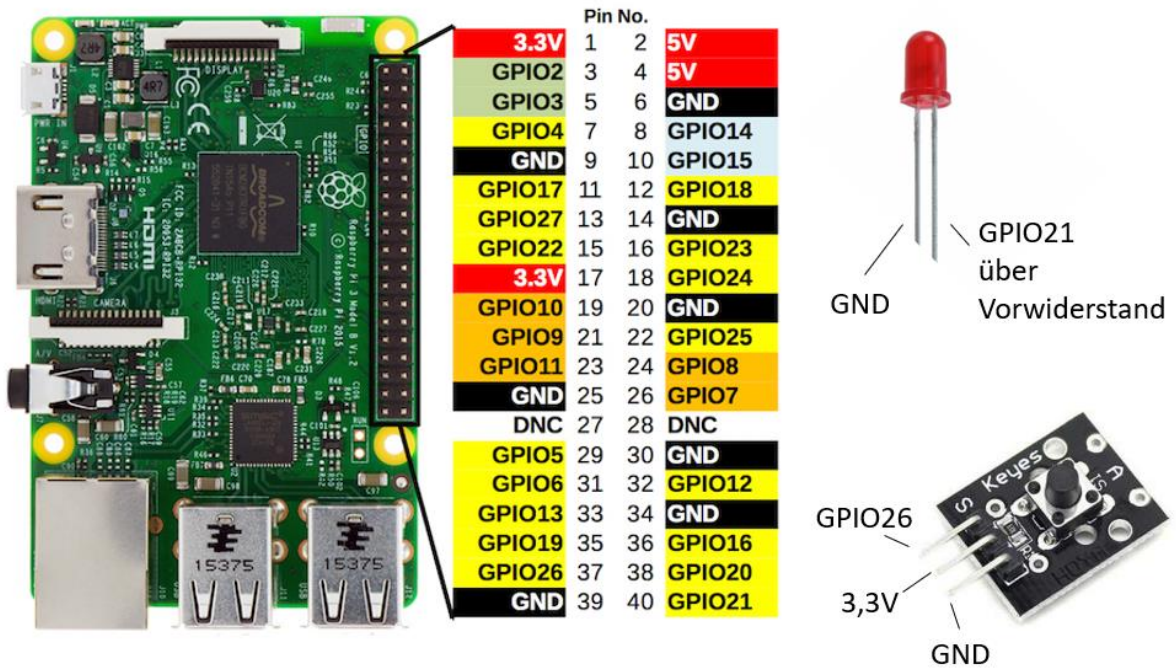

Abbildung 1 Raspberry Pinout, Quelle: https://www.bigmessowires.com/2018/05/26/raspberry-pi-gpio-programming-in-c/

2.2 Einbindung von GPIOs

Zur Festlegung und Ansteuerung der einzelnen Pins eines Raspberry Pis folgende grundlegende Befehle verwendet.

- Einbindung des GPIO-Moduls in Python: `import RPi.GPIO as GPIO`
- Festlegen der Pin- Zählweise: `GPIO.setmode(GPIO.BOARD)`
- Festlegen der Pin- Zählweise `GPIO.setmode(GPIO.BCM)`
- Einen Pin (hier Pin22/ GPIO 25) als Ausgang festlegen: `GPIO.setup(22, GPIO.OUT)`
- Einen Pin (hier Pin22/ GPIO 25) als Eingang festlegen: `GPIO.setup(22, GPIO.IN)`
- Status eines Pins (hier Pin 22/ GPIO 25) abfragen: `GPIO.input(22) == GPIO.HIGH`
  In diesem Beispiel wird abgefragt, ob eine Spannung an Pin 22 anliegt. Mit `GPIO.LOW` kann auch abgefragt werden ob keine Spannung anliegt. Alternativ können auch boolesche Variablen zur Abfrage verwendet werden.

Flankenerkennung

Um einen Eingang nicht dauerhaft abzufragen und den code somit ständig zu durchlaufen, wird die Flankenerkennung verwendet. Hier wird bei einer steigenden Flanke (0 → 1) oder bei einer fallenden Flanke (1 → 0) ein Ereignis ausgelöst. Nachfolgendes Beispiel gibt den grundlegenden Aufbau einer Flankenerkennung wieder:

```
import time
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BOARD)
GPIO.setup(18, GPIO.IN, pull_up_down = GPIO.PUD_DOWN)

# Ereignis-Funktion, die bei einer steigenden Flanke ausgeführt wird
def flanke(channel):
    print("steigende Flanke an Eingang", channel )

# Ereignis deklarieren
channel = 18
GPIO.add_event_detect(channel, GPIO.RISING, callback = flanke,
bouncetime = 200)

while True:           # Das eigentliche Hauptprogramm, dient in
    time.sleep(0.1)     diesem Fall nur als Dauerschleife
```

In diesem Beispiel wird Pin 18 auf eine steigende Flanke (`GPIO.RISING`) überwacht. Dabei wird mit `bouncetime` ein Zeitintervall für das Abfragen des Eingangs festgelegt. Damit wird bei prellenden Schaltern verhindert, dass ein Ereignis mehrfach ausgelöst wird. Mit `callback` wird definiert, welche Funktion anschließend ausgeführt wird.

Fallende Flanken können entsprechend mit `GPIO.FALLING` erkannt werden. Es ist zu dem möglich sowohl steigende als auch fallende Flanken an einem Pin zu erkennen. Dazu wird `GPIO.BOTH` verwendet.

3 Kommentierte Beispielaufgaben

Beispielaufgabe 1 – Grundlagen Programmaufbau

```
# Es wird ein Radius eingelesen, der Umfang und die Fläche des Kreises werden ausgegeben.

import math # Einbindung des math Moduls

r=float(input("Radius eingeben: ")) #Einlesen des Radius und Deklaration als Gleitkommazahl

a=r**2*math.pi #Berechnung der Fläche mit der pi-Funktion aus dem math-Modul
u=2*r*math.pi #Berechnung des Umfangs mit der pi-Funktion aus dem math-Modul

print("Fläche: ",a) # Ausgabe der Fläche
print("Umfang: ",u) # Ausgabe des Umfangs
```

Beispielaufgabe 2 – if-Verzweigung und Schleifen

```
# Beispiel: Passwortabfrage
# Der Benutzer soll ein Passwort festlegen und wird anschließend aufgefordert das Passwort zu bestätigen.
# Bei falscher eingabe soll er das Passwort solange bestätigen, bis die Eingabe stimmt.

passwort=input("Geben Sie ein Passwort ein: ") # Variable passwort mit dem Eingabewert erzeugen

while True: # Starten einer Dauerschleife mit der Bedingung True
    passwort2=input("Bestätigen Sie das Passwort: ")

    if passwort==passwort2:  # Wenn passwort=passwort2, dann verlasse die Schleife
        break
    else:
        print("Falsche Eingabe! Bestätigen Sie erneut das Passwort.")

print("Passwort wurde bestätigt!") # Ausgabe, dass das Passwort zwei mal eingegeben wurde
```

Beispielaufgabe 3 - if-Verzweigung und Schleifen

```
# Beispiel 2: Passwortsicherung mit drei Versuchen
# Der Benutzer legt ein Passwort fest. Anschließend hat er drei Versuche das richtige Passworte inzugeben.
# Nach jedem Fehlversuch muss der Benutzer 10 Sekunden warten bis er erneut eine Eingabe durchfürhen kann.
# Bei drei Fehlversuchen wird das Gerät gesperrt.

import time # Wird für die Wartezeit benötigt

versuche=0 # Zählvariable für die Versuche
pw=input("Geben Sie ein Passwort ein: ") # Variable in der das Passwort gespeichert wird

while True: # Dauerschleife für die Passwortabfragung
    pw2=input("Passworteingabe: ")
    if pw2==pw: # Eingabe wird mit dem gespeicherten Passwort verglichen
        print("Das Gerät wird entsperrt!") # Bei korrekter Eingabe, wird die Schleife abgebrochen
        break
    else:
        versuche=versuche+1 # Bei falscher Eingabe, zähle die Anzahl der Versuche hoch
        print(versuche,end=". Fehleingabe\n")
        if versuche!=3: # Solange die Anzahl der Versuche ungleich 3, Sperre die eingabe für 10 Sekunden
            for x in range(10,0,-1): # Schleife zum runterzählen der 10 Sekunden
                print(x, end=" Sekunden bis zur nächsten Eingabe\n")
                time.sleep(1)

    if versuche==3: # Wurde das Passwort drei mal falsch eingegeben, wird das Gerät gesperrt
        print("Das Gerät wird gesperrt. Wenden Sie sich an den Systemadministrator.")
        break
```

Beispielaufgabe 4 – Einbindung der GPIOs

```
# Beispiel: Jedes Mal wenn der Taster gedrückt wird:
# - leuchtet die LED für eine Sekunde auf
# - wird eine Zählvariable um 1 erhöht
# - wird die Zählvariable ausgegeben

#-------------------------------------------------------------------------------------------

#Einbindung von Bibliotheken
import RPi.GPIO as GPIO
import time

#-------------------------------------------------------------------------------------------

#Zählweise der Pins festlegen (Pinzählweise oder GPIO Zählweise) hier: GPIO Zählweise
GPIO.setmode(GPIO.BCM)

#-------------------------------------------------------------------------------------------

# GPIOs definieren und als Eingang oder Ausgang festlegen
GPIO.setup(21,GPIO.OUT)
GPIO.setup(26,GPIO.IN)

#-------------------------------------------------------------------------------------------

# Definition von Variablen
i = 0 #Zählervariable, Startwert ist 0

#-------------------------------------------------------------------------------------------

#Hardware Check
#GPIO.output(21,True)#Einschalten GPIO 21
#time.sleep(1)#Wartezeit 1 Sekunde

#while True:
    #print(GPIO.input(26))

#-------------------------------------------------------------------------------------------

# Beginn Endlosschleife
while True:

    if GPIO.input(26) == GPIO.LOW: # Wenn Taster gedrückt ist:
        GPIO.output(21,True)# Schalte LED an
        time.sleep(2)# Warte zwei Sekunden
        GPIO.output(21,False)# Schalte LED wieder aus
        i = i + 1# Zählervariable um 1 erhöhen
        print("Zählerstand " + str(i))# Ausgabe des Zählerstands
#Ende Endlosschleife

#-------------------------------------------------------------------------------------------

```

Beispielaufgabe 5 - Flankenerkennung

```
# Beispiel: Jedes Mal wenn der Taster gedrückt wird:
# - wird eine Zählvariable um 1 erhöht
# - wird die Zählvariable ausgegeben
# - Programmierung mittels Interrupt

#-------------------------------------------------------------------------------

# Einbindung der erforderlichen Bibliotheken
import time
import RPi.GPIO as GPIO

#-------------------------------------------------------------------------------

# Zählweise der Pins festlegen
GPIO.setmode(GPIO.BCM)

#-------------------------------------------------------------------------------

# GPIOs definieren und als Eingang oder Ausgang festlegen
GPIO.setup(26, GPIO.IN)
GPIO.setup(21, GPIO.OUT)

#-------------------------------------------------------------------------------

# Definition von Variablen
i = 0# Zählervariable, Startwert ist 0

#-------------------------------------------------------------------------------

# Ereignis-Funktion, wird aufgerufen wenn positive Flanke an Taster anliegt
def zählhoch(pin):# Definition der Funktion
    global i# Zugriff auf Variable i ermöglichen
    i = i + 1# Zählervariable um 1 erhöhen
    print ("Zählerstand:" + str(i))# Ausgabe des Zählerstands

#-------------------------------------------------------------------------------

# Ereignis deklarieren
GPIO.add_event_detect(26, GPIO.FALLING, callback = zählhoch, bouncetime = 200)

#-------------------------------------------------------------------------------

# Beginn Endlosschleife
while True:
    time.sleep(0.1)
# Ende Endlosschleife

#-------------------------------------------------------------------------------

```

4 Übungsaufgaben

Aufgabe 1)

Zwei Zahlen werden eingelesen, die Strecke und die Zeit, in der die Strecke zurückgelegt wird. Ausgegeben werden soll die Durchschnittsgeschwindigkeit in m/s und km/s.

Aufgabe 2)

Es sollen zwei Zahlen eingelesen werden. Die Zahl 2 muss größer als Zahl 1 sein. Das Programm soll anschließend alle ganzen Zahlen aufsummieren, die zwischen den beiden liegen. Abschließend soll die Summe aller Zahlen ausgegeben werden.

Zusätzliche Aufgabenstellung für Fortgeschrittene:

a) Es sollen nur alle geraden Zahlen aufsummiert werden.
b) Die Reihenfolge der eingegebenen Zahlen soll egal sein.

Aufgabe 3)

Eine LED soll dauerhaft mit einer Frequenz von 5 Hz blinken, sobald der Taster gedrückt wird, verändert sich die Frequenz auf 1 Hz.

Aufgabe 4)

LED wird mit dem ersten Tastendruck eingeschaltet und mit dem zweiten ausgeschaltet. Programmierung soll mittels Interrupt durchgeführt werden.

Zusätzliche Aufgabenstellung für Fortgeschrittene:

a) Bei Tastendruck soll die LED mit einer Frequenz von 10Hz acht Mal blinken und anschließend wieder ausgehen.

5 Lösungen zu den Übungsaufgaben

Aufgabe 1)

```
1    # Zwei Zahlen werden eingelesen, die Strecke und die Zeit, in der die
       Strecke zurückgelegt wird.
2    # Ausgegeben werden soll die Durchschnittsgeschwindigkeit in m/s und
       km/s.
3
4    s=float(input("Geben Sie eine Strecke in Kilometer ein: "))
5    t=float(input("Geben Sie eine Zeit in Stunden an: "))
6
7    print("Durchschnittsgeschwindigkeit in km/h: ", s/t)
8    print("Durchschnittsgeschwindigkeit in m/s: ", s/(3.6*t))
```

Aufgabe 2)

```
1  # Es sollen zwei Zahlen eingelesen werden. Das Programm summiert anschließend alle ganzen Zahlen, die
2  # zwischen den beiden liegen auf und gibt die Summe aus
3
4  a=int(input("Geben Sie eine Zahl 1 ein: "))
5  b=int(input("Geben Sie eine Zahl 2 ein, die größer als Zahl 1 ist: "))
6
7  summe=0 #Vairable für die Summe
8  i=a #Laufvariable für die Schleife. Laufvariable beginnt immer bei der Zahl 1
9
10 while i<=b: # Schleife wird solange durchlaufen, bis die Laufvariable den Wert der Zahl b angenommen hat
11     summe=summe+i # Aufsummieren mit der Laufvariablen
12     i=i+1 # Nach einem Durchgang wird die Laufvariable um eins erhöht
13 print("Die Summe lautet: ",summe) # Ausgabe der Summe
14
15
```

Aufgabe 3)

```
1  # Aufgabe: LED soll dauerhaft mit einer Frequenz von 5 Hz blinken
2  # ,sobald der Taster gedrückt wird, verändert sich die Frequenz auf 1 Hz
3
4  #-----------------------------------------------------------------------------------------
5
6  # Einbindung der erforderlichen Bibliotheken
7  import RPi.GPIO as GPIO
8  import time
9
10 #-----------------------------------------------------------------------------------------
11
12 # Zählweise der Pins festlegen (Pinzählweise oder GPIO Zählweise) hier: GPIO Zählweise
13 GPIO.setmode(GPIO.BCM)
14
15 #-----------------------------------------------------------------------------------------
16
17 # GPIOs zuweisen und als Eingang oder Ausgang definieren
18 GPIO.setup(21, GPIO.OUT)
19 GPIO.setup(26, GPIO.IN)
20
21 #-----------------------------------------------------------------------------------------
22
23 # Beginn Endlosschleife
24 while True:
25
26     if GPIO.input(26) == GPIO.LOW: # wenn Taster gedrückt ist:
27         GPIO.output(21,True)# Schalte LED an
28         time.sleep(0.5)# Warte eine halbe Periodenddauer (bei 1Hz: 0,5s)
29         GPIO.output(21,False)
30         time.sleep(0.5)# Warte eine halbe Periodenddauer (bei 1Hz: 0,5s)
31
32     else:# wenn Taster nicht gedrückt ist:
33         GPIO.output(21,True)# Schalte LED an
34         time.sleep(0.1)# Warte eine halbe Periodenddauer (bei 5Hz: 0,1s)
35         GPIO.output(21,False)# Schalte LED wieder aus
36         time.sleep(0.1)# Warte eine halbe Periodenddauer (bei 5Hz: 0,1s)
37 # Ende Endlosschleife
38
39 #-----------------------------------------------------------------------------------------
```

Aufgabe 4)

```
# Beispiel: LED wird mit dem ersten Tastendruck eingeschaltet und mit dem zweiten ausgeschaltet:
# - Programmierung mittels Interrupt

#-----------------------------------------------------------------------------------------------

# Einbindung der erforderlichen Bibliotheken
import time
import RPi.GPIO as GPIO

#-----------------------------------------------------------------------------------------------

# Zählweise der Pins festlegen
GPIO.setmode(GPIO.BCM)

#-----------------------------------------------------------------------------------------------

# GPIOs definieren und als Eingang oder Ausgang festlegen
GPIO.setup(26, GPIO.IN)
GPIO.setup(21, GPIO.OUT)

# Ereignis-Funktion, wird aufgerufen wenn positive Flanke an Taster anliegt
def led_steuerung(pin):# Definition der Funktion
    if GPIO.input(21) == GPIO.LOW:
        GPIO.output(21,True)
    else:
        GPIO.output(21,False)

#-----------------------------------------------------------------------------------------------

# Ereignis deklarieren
GPIO.add_event_detect(26, GPIO.FALLING, callback = led_steuerung, bouncetime = 200)

#-----------------------------------------------------------------------------------------------

# Beginn Endlosschleife
while True:
    time.sleep(0.1)
    print(GPIO.input(26))
# Ende Endlosschleife

#-----------------------------------------------------------------------------------------------
```

6 Beispielcode Motoransteuerung

```
# Einbindung der Module
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)  # Festlegen der Zählweise
GPIO.setwarnings(False) #

#Einbindung der Magnete und Mosfet, Reedschalter, LED und Taster
GPIO.setup(21,GPIO.OUT) # Zylinder rechts
GPIO.setup(26,GPIO.OUT) # Zylinder links
GPIO.setup(20,GPIO.IN)  # Reedschalter links
GPIO.setup(16,GPIO.IN)  # Reedschalter rechts
GPIO.setup(12,GPIO.IN)  # Taster
GPIO.setup(17,GPIO.OUT) # Led Frabe rot
GPIO.setup(27,GPIO.OUT) # Led Frabe grün

anzugszeit=0.038 # Variable für die Anzugszeit
zaehler=0
umdrehungen=0
reed1=time.time()
reed2=time.time()

while True:

    if GPIO.input(12) == False:      ### Bedingung: wenn Taster gedrückt, dann...

        GPIO.output(27, False)            # Ausschalten der Lampe rot
        GPIO.output(17, True)             # Einschalten der Lampe grün

        if GPIO.input(16) == False:       # Wenn Reedschalter 1 ausgelöst wird, dann...
           GPIO.output(21,True)           # Schalte Magnet 1
           reed1=time.time()              # Speichere den Zeitpunkt, an dem Reedschalter 1 ausgelöst wird
           zaehler=zaehler+0.5            # Erhöhe den Zähler um 0.5
           umdrehungen=umdrehungen+0.5    # Erhöhe die Anzahl der Umdrehungen um 0.5
           time.sleep(anzugszeit)         # Lasse den Magneten 1 noch um die Variable anzugszeit weiter angezogen
        else:
            GPIO.output(21,False)         # Schalte den Magneten 1 wieder aus

        if GPIO.input(20) ==False:        # Gleiche Vorgehensweise wie bei Magnet 1, nur mit Magnet 2 und Reedschalter 2
            GPIO.output(26,True)
            reed2=time.time()
            zaehler=zaehler+0.5
            umdrehungen=umdrehungen+0.5
            time.sleep(anzugszeit)
        else:
            GPIO.output(26,False)

    else:                             ### Wenn der Taster nicht gedrückt wird, dann...
        GPIO.output(27,True)           # Schalte die LED rot ein
        GPIO.output(17,False)          # Schalte die LED grün aus
        GPIO.output(21,False)          # Schalte beide Magnete wieder aus
        GPIO.output(26,False)

        if umdrehungen!=0: # Gebe die anzahl der Umdrehungen aus, wenn mehr als eine Umdrehung geschehen
                           #-> Dauerausgabe umbedingt vermeiden, da es sonst zu Verzögerungen beim Ablauf kommt
                           #->Auswirkungen auf die Drehzahl
            print('Die Anzahl der Umdrehungen ab Start beträgt: ', umdrehungen)
            umdrehungen=0  # Rücksetzen der Umdrehunge nach einer Ausgabe

    Drehzahl=abs(round(60/(2*(reed2-reed1)))) # Betrag der Drehzahl bestimmen (Befehl abs())
                                              # und runden auf ganze Stellen (Befehl round())

    if zaehler%50==0 and zaehler!=0:          # Ausgabe der Drehzahl nach einer bestimmten Anzahl an Umdrehungen
        zaehler=0                             # Rücksetzen des Zählers
        print('Die aktuelle Drehzahl beträgt: ',Drehzahl, ' 1/min')

```