```python
import tkinter
import sys
import KlasseRoboter
#import GUI_tkinter
class MyApp(tkinter.Frame):
    programm_laeuft="nein"
    def __init__(self, master=None):
        super().__init__(master)
        self.pack()
        self.createWidgets()
        self.my_roboter=KlasseRoboter.Roboter()



    def createWidgets(self):
        self.howtoLabel_1 = tkinter.Label(self)
        self.howtoLabel_1.pack()
        self.text_1 = tkinter.StringVar()
        self.text_1.set(\
        "Diese Programm steuert den Roboter.\n \
         Er muss dazu wissen, wie viele Scheiben der Turm hat. \n \
         Bitte geben Sie die Anzahl als Zahl (z. Bsp 4) ein:")
        self.howtoLabel_1["textvariable"] = self.text_1



        self.nameEntry_1 = tkinter.Entry(self,width=3)
        self.nameEntry_1.pack()
        self.name_1 = tkinter.StringVar()
        self.name_1.set(1)
        self.nameEntry_1["textvariable"] = self.name_1

        self.howtoLabel_2 = tkinter.Label(self)
        self.howtoLabel_2.pack()
        self.text_2 = tkinter.StringVar()
        self.text_2.set(\
        "Der Turm wird verschoben von Position :")
        self.howtoLabel_2["textvariable"] = self.text_2

        self.nameEntry_2 = tkinter.Entry(self,width=3)
        self.nameEntry_2.pack()
        self.name_2 = tkinter.StringVar()
        self.name_2.set("A")
        self.nameEntry_2["textvariable"] = self.name_2

        self.howtoLabel_3 = tkinter.Label(self)
        self.howtoLabel_3.pack()
        self.text_3 = tkinter.StringVar()
        self.text_3.set(\
        "nach Position :")
        self.howtoLabel_3["textvariable"] = self.text_3

        self.nameEntry_3 = tkinter.Entry(self,width=3)
        self.nameEntry_3.pack()
        self.name_3 = tkinter.StringVar()
        self.name_3.set("B")
        self.nameEntry_3["textvariable"] = self.name_3

        self.nameEntry_4 = tkinter.Entry(self,width=50)
        self.nameEntry_4.pack()
        self.name_4 = tkinter.StringVar()
        self.name_4.set(self.name_1.get()+self.name_2.get()+self.name_3.get())
        self.nameEntry_4["textvariable"] = self.name_4

        #self.ok = tkinter.Button(self)
        #self.ok["text"] = "Ok"
        #self.ok["command"] = self.onReverse()
        #self.ok.pack()

```

```
        self.rev_1 = tkinter.Button(self)
        self.rev_1["text"] = "Start"
        self.rev_1["command"] = self.onReverse
        self.rev_1.pack()




        self.rev_2 = tkinter.Button(self)
        self.rev_2["text"] = "Not aus!"
        self.rev_2["command"] = self.onNotAus
        self.rev_2.pack()


    def onReverse(self):
        erlaubt1="A B C"
        erlaubt2="1 2 3 4"

        if (self.name_1.get() in erlaubt2 and self.name_2.get() \
            in erlaubt1 and self.name_3.get() in erlaubt1 and \
            MyApp.programm_laeuft=="nein"):
            self.name_4.set(self.name_1.get()+" "+ \
            self.name_2.get()+" "+self.name_3.get()+ \
            " ist als Eingabe erlaubt. Das Programm wird gestartet.")
            if self.name_1.get()=="1":
                anzahl=1
            elif self.name_1.get()=="2":
                anzahl=2
            elif self.name_1.get()=="3":
                anzahl=3
            else :
                anzahl=4

            if self.name_2.get()=="A":
                von=1
            elif self.name_2.get()=="B":
                von=2
            else :
                von=3

            if self.name_3.get()=="A":
                nach=1
            elif self.name_3.get()=="B":
                nach=2
            else :
                nach=3
            print ((anzahl),(von),(nach))
            print ("OK!")


            self.my_roboter.turmVersetzen1(anzahl,von,nach)
            MyApp.programm_laeuft="ja"
            #Programmaufruf

        elif (MyApp.programm_laeuft=="ja"):
            self.name_4.set("Das Programm läuft bereits!")
            print ("Fehler!")
        else:
            self.name_4.set(self.name_1.get()+" "+ \
            self.name_2.get()+" "+self.name_3.get()+ \
            " ist als Eingabe nicht erlaubt.Bitte korrigieren!")
            print ("Fehler!")

    def onNotAus(self):
        # Stepper nach oben
        # Schlitten nach links
        sys.exit()

#and self.name_2.get() in erlaubt1 and self.name_3.get() in erlaubt2:
```

```
        #else:
        #   self.name_4.set(self.name_1.get()+self.name_2.get()+self.name_3.get()+\
        #   " sind fehlerhaft. Bitte korrigiere die Eingaben oben")

root=tkinter.Tk()
app=MyApp(root)
app.mainloop()


   # if __name__ == "__main__":

   #    my_gui=GUI_tkinter.MyApp()
       #my_greifer.aufnehmen()
       #time.sleep(0.2)
       #my_greifer.ablegen()

```

```
import RPi.GPIO as gpio
gpio.setmode(gpio.BCM)
import time
import KlasseSchlitten
import KlasseGreifer
import KlasseAblage
import signal
import sys
gpio.setmode(gpio.BCM)

class Roboter:
    def __init__(self,gpio_step=(12,20,16,21),gpio_mot=(27,17),\
        gpio_slt=(13,26,5),gpio_mag=(22,4),winkel_step=430,sleep_step=0.005):
        self.gpio_step=gpio_step
        self.gpio_mot=gpio_mot
        self.gpio_slt=gpio_slt
        self.gpio_mag=gpio_mag
        self.winkel_step=winkel_step
        self.sleep_step=sleep_step

        self.my_schlitten_1=KlasseSchlitten.Schlitten(self.gpio_mot,self.gpio_slt)

        self.my_ablage_1=KlasseAblage.Ablage()
        self.my_greifer_1=KlasseGreifer.Greifer(self.gpio_step,self.winkel_step, \
                          self.sleep_step,self.gpio_mag,self.my_ablage_1)
    def turmVersetzen(self,anzahl,von,ueber,nach):
        print(anzahl,von,ueber,nach)

        if anzahl > 1:

            self.turmVersetzen(anzahl-1,von,nach,ueber)
        #Lege die letzte Scheibe von "von" nach "nach"
        print ("im
turmversetzen",self.my_ablage_1.gib_anzahl(1),self.my_ablage_1.gib_anzahl(2),self.my_ablage_1.gib_anzah
l(3))
        print ("im turmversetzen",von,nach,ueber)
        self.my_schlitten_1.zu_position(von)
        self.my_greifer_1.aufnehmen(von)
        self.my_ablage_1.verringere_um_1(von)
        print ("im
turmversetzen",self.my_ablage_1.gib_anzahl(1),self.my_ablage_1.gib_anzahl(2),self.my_ablage_1.gib_anzah
l(3))
        self.my_schlitten_1.zu_position(nach)
        self.my_greifer_1.ablegen(nach)
        self.my_ablage_1.erhoehe_um_1(nach)
        print ("im
turmversetzen",self.my_ablage_1.gib_anzahl(1),self.my_ablage_1.gib_anzahl(2),self.my_ablage_1.gib_anzah
l(3))
        if anzahl > 1:

            self.turmVersetzen(anzahl-1,ueber,von,nach)

    def turmVersetzen1(self,anzahl,von,nach):
        self.my_ablage_1.setze_anzahl(von,anzahl)
        self.my_ablage_1.setze_anzahl(nach,0)
        ueber=6-von-nach
        self.my_ablage_1.setze_anzahl(ueber,0)
        print ("im
turmversetzen1",self.my_ablage_1.gib_anzahl(1),self.my_ablage_1.gib_anzahl(2),self.my_ablage_1.gib_anza
hl(3))
        self.turmVersetzen(anzahl,von,ueber,nach)




if __name__ == "__main__":

    my_roboter=Roboter()
    my_roboter.turmVersetzen1(4,1,3)
```

```
62          #time.sleep(0.2)
63          #my_greifer.ablegen()
64
65
66
```

```python
import RPi.GPIO as gpio
import time
import KlasseStepper
import KlasseMagnet
import KlasseAblage
import signal
import sys
gpio.setmode(gpio.BCM)
class Greifer():
    def __init__(self,tup4_step,wink_step,slep_step,tup2_mag,ablage):

        self.steppertupel=tup4_step
        self.stepperwinkel=wink_step
        self.steppersleep=slep_step

        self.magnettupel=tup2_mag
        self.my_magnet=KlasseMagnet.Magnet(self.magnettupel)
        self.my_ablage=ablage
        self.my_stepper=KlasseStepper.Stepper(self.steppertupel)

    def aufnehmen(self,position):
        print ("Greiferaufnehmen",self.my_ablage.gib_anzahl(position))
        winkel=self.stepperwinkel-self.my_ablage.gib_anzahl(position)*90
        print ("Greiferaufnehmen runter",winkel,position)
        self.my_stepper.drehe_um_winkel(+winkel,self.steppersleep)
        self.my_magnet.ein()
        print ("Greiferaufnehmen hoch",winkel)
        self.my_stepper.drehe_um_winkel(-winkel,self.steppersleep)

    def ablegen(self,position):
        print ("Greiferablegen",self.my_ablage.gib_anzahl(position))
        winkel=self.stepperwinkel-(self.my_ablage.gib_anzahl(position)+1)*90
        print ("Greiferablegen",winkel)
        self.my_stepper.drehe_um_winkel(winkel,self.steppersleep)
        self.my_magnet.aus()
        self.my_stepper.drehe_um_winkel(-winkel,self.steppersleep)

if __name__ == "__main__":
       #my_ablage.erhoehe_um_1(3)
       my_ablage=KlasseAblage.Ablage()
       print ("Ausgabe1",my_ablage.gib_anzahl(3))
       my_ablage.erhoehe_um_1(3)
       my_ablage.erhoehe_um_1(3)
       print ("Ausgabe2",my_ablage.gib_anzahl(3))

       my_greifer=Greifer((12,20,16,21),360,0.009,(22,4),my_ablage)
       my_greifer.aufnehmen(3)
       #time.sleep(0.2)
       my_greifer.ablegen(3)

       gpio.cleanup()
```

```python
import RPi.GPIO as gpio
import time
import KlasseSchalterreihe
import KlasseMotor

gpio.setmode(gpio.BCM)

class Schlitten():
   def __init__(self,tup2_mot,tup3_Slt):

       self.motortupel=tup2_mot
       #Schalterleiste

       self.Schaltertupel=tup3_Slt

       self.my_motor=KlasseMotor.Motor(self.motortupel)

       self.my_schalterreihe=KlasseSchalterreihe.Schalterreihe(self.Schaltertupel)
   def nach_eins(self):
       if not self.my_schalterreihe.position()==1:
           self.my_motor.links()
           self.warten_auf_ankunft_an(1)
           self.my_motor.stop()

   def nach_drei(self):
       if not self.my_schalterreihe.position()==3:
           self.my_motor.rechts()
           self.warten_auf_ankunft_an(3)
           self.my_motor.stop()

   def nach_zwei(self):
       if not self.my_schalterreihe.position()==2:
           if self.my_schalterreihe.position()==1:
              self.nach_drei()
           if self.my_schalterreihe.position()==3:
              self.my_motor.links()
              self.warten_auf_ankunft_an(2)
              self.my_motor.stop()

   def zu_position(self,s_position):
       if s_position==1:
           self.nach_eins()
       if s_position==2:
           self.nach_zwei()
       if s_position==3:
           self.nach_drei()



   def warten_auf_ankunft_an(self,position):
       while not self.my_schalterreihe.position()==position:
           time.sleep(0.1)

if __name__ == "__main__":

       my_schlitten=Schlitten((27,17),(13,26,5))
       #my_motor=KlasseMotor.Motor((27,17))
       #my_schlitten.nach_drei()

       my_schlitten.zu_position(2)
       #my_schlitten.warten_auf_ankunft_an(3)
       #self.my_motor.links()
       #time.sleep(5)
       #my_motor.stop()
       #self.my_motor.stop()
       gpio.cleanup()
```

```
70
71
72
73
74
```

```python
import RPi.GPIO as gpio
import time
import KlasseStepper
import KlasseMagnet
gpio.setmode(gpio.BCM)
gpio.setwarnings(False)
class Ablage():
    anzahl_an_1=0
    anzahl_an_2=0
    anzahl_an_3=0

    def __init__(self):

        self.anzahl_an_x=0
    @staticmethod
    def verringere_um_1(position):
        if position==1:
            Ablage.anzahl_an_1=Ablage.anzahl_an_1 - 1
        if position==2:
            Ablage.anzahl_an_2=Ablage.anzahl_an_2 - 1
        if position==3:
            Ablage.anzahl_an_3=Ablage.anzahl_an_3 - 1
    @staticmethod
    def erhoehe_um_1(position):
        if position==1:
            Ablage.anzahl_an_1=Ablage.anzahl_an_1 + 1
        if position==2:
            Ablage.anzahl_an_2=Ablage.anzahl_an_2 + 1
        if position==3:
            Ablage.anzahl_an_3=Ablage.anzahl_an_3 + 1
    @staticmethod
    def setze_anzahl(position,zahl):
        if position==1:
            Ablage.anzahl_an_1=zahl
        if position==2:
            Ablage.anzahl_an_2=zahl
        if position==3:
            Ablage.anzahl_an_3=zahl
    @staticmethod
    def gib_anzahl(position):
        if position==1:
            return Ablage.anzahl_an_1
        if position==2:
            return Ablage.anzahl_an_2
        if position==3:
            return Ablage.anzahl_an_3




if __name__ == "__main__":

    Ablage.erhoehe_um_1(3)

    my_ablage=Ablage()

    #my_ablage.verringere_um_1(3)

    my_ablage.setze_anzahl(3,8)
    print (my_ablage.gib_anzahl(3))
    my_ablage.erhoehe_um_1(3)
    print (my_ablage.gib_anzahl(3))
    my_ablage.verringere_um_1(3)
    print (my_ablage.gib_anzahl(3))
```

```python
import RPi.GPIO as gpio
import time
gpio.setmode(gpio.BCM)
class Magnet():
    def __init__(self,tup2_mag):
        self.gpio1=tup2_mag[0]
        self.gpio2=tup2_mag[1]
        gpio.setup(self.gpio1,gpio.OUT)
        gpio.setup(self.gpio2,gpio.OUT)
        gpio.output(self.gpio1,gpio.LOW)
        gpio.output(self.gpio2,gpio.LOW)

    def ein(self):
        gpio.output(self.gpio1,gpio.HIGH)
        print (gpio.input(self.gpio1))
        print (gpio.input(self.gpio2))

    def aus(self):
        gpio.output(self.gpio1,gpio.LOW)
        print (gpio.input(self.gpio1))
        print (gpio.input(self.gpio2))

if __name__ == "__main__":

    my_magnet=Magnet((22,4))

    my_magnet.ein()
    time.sleep(50)
    #my_magnet.aus()

    gpio.cleanup()
```

```python
import RPi.GPIO as gpio
import time
gpio.setmode(gpio.BCM)

class Motor():
   def __init__(self,tup2_mot):
       self.gpio1=tup2_mot[0]
       self.gpio2=tup2_mot[1]

       gpio.setup(self.gpio1,gpio.OUT)
       gpio.setup(self.gpio2,gpio.OUT)

       gpio.output(self.gpio1,gpio.LOW)
       gpio.output(self.gpio2,gpio.LOW)

       #print (self.gpio1,self.gpio2)
       #Schalterleiste
       #self.Schalter1=tup3Slt[0]
       #self.Schalter2=tup3Slt[1]
       #self.Schalter3=tup3Slt[2]
       #self.Schaltertupel=tup3Slt



   def rechts(self):
       #Bewegung nach rechts
       gpio.output(self.gpio1,gpio.HIGH)
       gpio.output(self.gpio2,gpio.LOW)

   def links(self):
       #Bewegung nach links
       gpio.output(self.gpio1,gpio.LOW)
       gpio.output(self.gpio2,gpio.HIGH)

   def stop(self):
       #Bewegung stoppen
       gpio.output(self.gpio1,gpio.LOW)
       gpio.output(self.gpio2,gpio.LOW)

   def ausgabe_gpio(self):
       print(gpio.input(self.gpio1))
       print(gpio.input(self.gpio2))


if __name__ == "__main__":

    my_motor=Motor((27,17))

    my_motor.ausgabe_gpio()
    print("links")
    my_motor.links()
    time.sleep(0.5)
    my_motor.ausgabe_gpio()
    my_motor.stop()
    print("stop")
    time.sleep(0.5)
    print("rechts")
    my_motor.rechts()
    time.sleep(0.5)
    my_motor.stop()
    my_motor.ausgabe_gpio()


    gpio.cleanup()
    gpio.setwarnings(False)
```

```
70
71
72
73
```

```python
import RPi.GPIO as gpio
import time
gpio.setmode(gpio.BCM)
gpio.setwarnings(False)
class Stepper():


    def __init__(self,tup4_step):

        self.steppertupel=tup4_step
        self.in1=tup4_step[0]
        self.in2=tup4_step[1]
        self.in3=tup4_step[2]
        self.in4=tup4_step[3]
        gpio.setup(self.in1,gpio.OUT)
        gpio.setup(self.in2,gpio.OUT)
        gpio.setup(self.in3,gpio.OUT)
        gpio.setup(self.in4,gpio.OUT)



    def einstellung_stepper(self,schalterGpio,asleep):
        gpio.setmode(gpio.BCM)

        schalter=schalterGpio

        print (gpio.input(schalter))
        while (gpio.input(schalter)==0):
               gpio.output(self.in1,gpio.HIGH)
               time.sleep(asleep)
               gpio.output(self.in1,gpio.LOW)
               gpio.output(self.in2,gpio.HIGH)
               time.sleep(asleep)
               gpio.output(self.in2,gpio.LOW)
               gpio.output(self.in3,gpio.HIGH)
               time.sleep(asleep)
               gpio.output(self.in3,gpio.LOW)
               gpio.output(self.in4,gpio.HIGH)
               time.sleep(asleep)
               gpio.output(self.in4,gpio.LOW)



    def drehe_um_winkel(self,winkel,asleep):

        if winkel > 0:
           drehungen=int(round((winkel*512/360)))
           i=1

           while i in range (0,drehungen):
               gpio.output(self.in1,gpio.HIGH)
               time.sleep(asleep)
               gpio.output(self.in1,gpio.LOW)
               gpio.output(self.in2,gpio.HIGH)
               time.sleep(asleep)
               gpio.output(self.in2,gpio.LOW)
               gpio.output(self.in3,gpio.HIGH)
               time.sleep(asleep)
               gpio.output(self.in3,gpio.LOW)
               gpio.output(self.in4,gpio.HIGH)
               time.sleep(asleep)
               gpio.output(self.in4,gpio.LOW)
               i=i+1
        else:
           drehungen=int(round((winkel*512/360*(-1))))
           i=1
           while i in range (0,drehungen):
               gpio.output(self.in4,gpio.HIGH)
```

```
                time.sleep(asleep)
                gpio.output(self.in4,gpio.LOW)
                gpio.output(self.in3,gpio.HIGH)
                time.sleep(asleep)
                gpio.output(self.in3,gpio.LOW)
                gpio.output(self.in2,gpio.HIGH)
                time.sleep(asleep)
                gpio.output(self.in2,gpio.LOW)
                gpio.output(self.in1,gpio.HIGH)
                time.sleep(asleep)
                gpio.output(self.in1,gpio.LOW)
                i=i+1


if __name__ == "__main__":

    my_stepper=Stepper((12,20,16,21))

    my_stepper.drehe_um_winkel(+30,0.009)
    #my_stepper.drehe_um_winkel(+100,0.009)

    gpio.cleanup()

```

```python
import RPi.GPIO as gpio
import time
gpio.setmode(gpio.BCM)

class Schalterreihe(object):

    def __init__(self,tup3Slt):

        self.gpioS1=tup3Slt[0]
        self.gpioS2=tup3Slt[1]
        self.gpioS3=tup3Slt[2]
        gpio.setup(self.gpioS1,gpio.IN,pull_up_down=gpio.PUD_UP)
        print(gpio.input(self.gpioS1))
        gpio.setup(self.gpioS2,gpio.IN,pull_up_down=gpio.PUD_UP)
        print(gpio.input(self.gpioS2))
        gpio.setup(self.gpioS3,gpio.IN,pull_up_down=gpio.PUD_UP)
        print(gpio.input(self.gpioS3))



    def position(self):
        position="X"

        if gpio.input(self.gpioS1)==0:
            position=1
        if gpio.input(self.gpioS2)==0:
            position=2
        if gpio.input(self.gpioS3)==0:
            position=3
        return position







if __name__ == "__main__":

    my_schalterreihe=Schalterreihe((13,26,5))
    gpio.setup(13,gpio.IN,pull_up_down=gpio.PUD_UP)

    gpio.setup(26,gpio.IN,pull_up_down=gpio.PUD_UP)

    gpio.setup(5,gpio.IN,pull_up_down=gpio.PUD_UP)

    #Die Schalter müssen gedrückt werden
    for i in range (10):#my_schlitten.nach_eins()
        print(my_schalterreihe.position())
        time.sleep(2)



    gpio.cleanup()


```