Drohnen programmieren

mit Python 3 und der DJI Tello

InfoLabSaar

InfoLabSaar

Heute

1. DJI Tello kennenlernen und fliegen
2. Python 3 Basics
3. DJI Tello mit Python 3 programmieren

DJI Tello kennenlernen und fliegen

Bestandteile der DJI Tello

1. Akku einsetzen

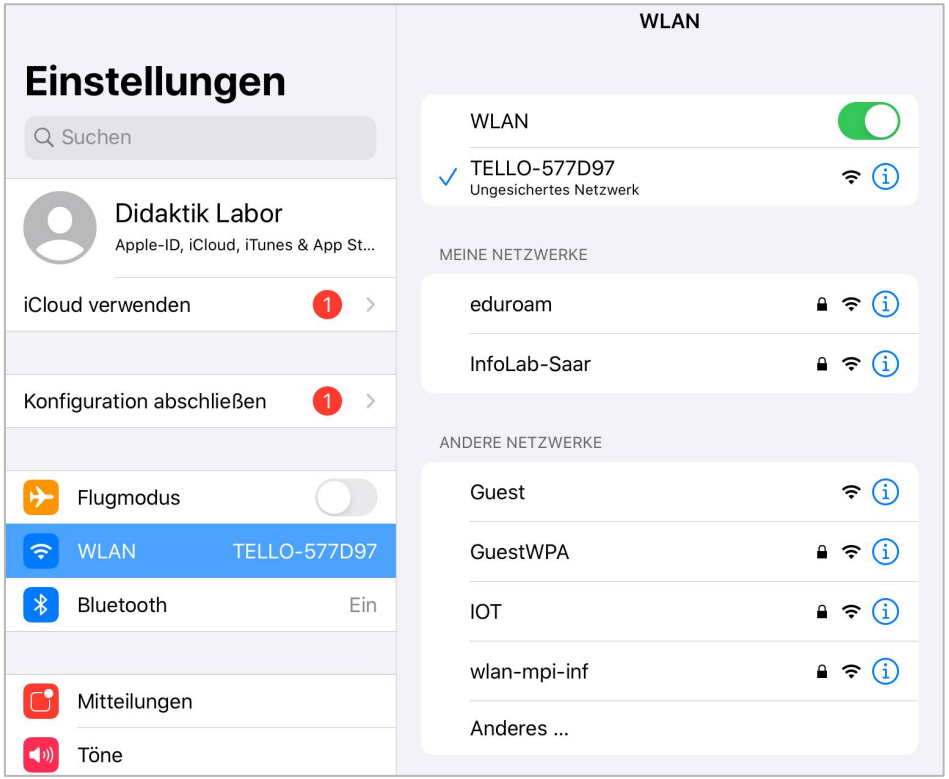
2. Drohne einschalten

3. Mit Drohne verbinden
(Nummer beachten !)

4. App starten →
Joystick Modus wählen

Wlan verbinden

Mit Drohne verbinden
(Nummer beachten !)

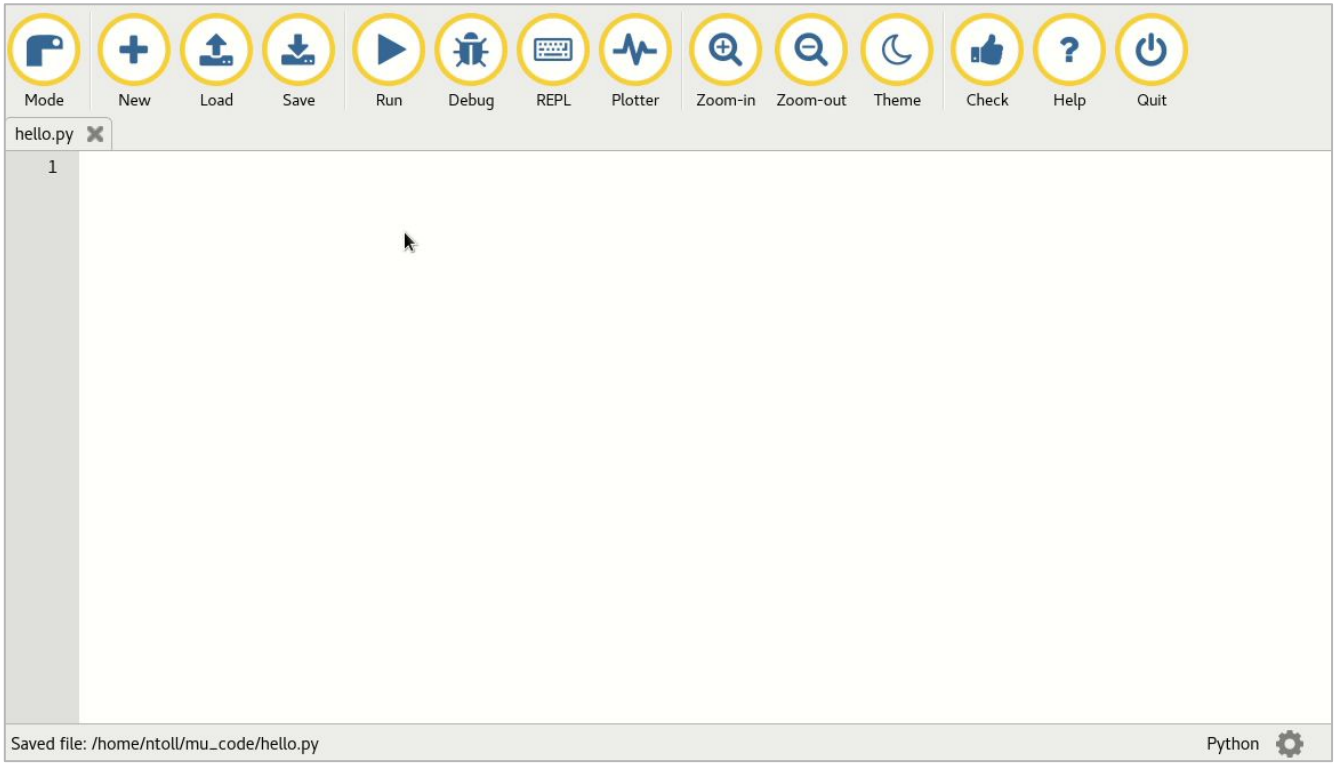


App “Tello Edu” starten

Python 3 Basics

Mu Editor für Python 3

codewith.mu



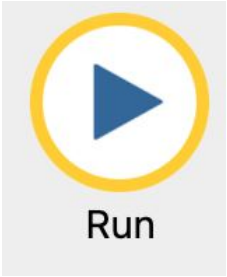
Python Programme im Ordner “Python” auf dem Desktop speichern!
Falls eine Datei geöffnet ist, bitte schließen.

01 - “Hello World!”:

Funktion mit String-Parameter aufrufen

```
1  print("Hello World!")
```

- ***Funktion*** `print` aufrufen
- ***Parameter*** in Klammern
- String-Parameter in Anführungszeichen
- Bei Python ist kein Semikolon am Ende des Befehls!
  So ist die ***Syntax***.
- Ausführen durch Klick auf Run/Ausführen
- Ausgabe des `print`-Befehls auf die Konsole



Beim ersten Mal muss gespeichert werden

Ausgabe in der *Konsole*:

```
Führe aus: hello.py
Hello World!
>>>
```

02 - Addition:

Funktion mit Integer-Parametern aufrufen

```
1  print(2+3)
```

- Addition der Zahlen als Parameter
- Es wird erst gerechnet, dann das Ergebnis ausgegeben

Ausgabe:

```
Führe aus: hello.py
5
>>>
```

- Parameter von `print` ist entweder ein String oder eine Zahl.
- Mischungen aus String und Zahl können nicht Parameter sein!

03 - Addition: Funktion definieren

```
1  def add():
2      print(2+3)
3
4  add()
5  add()
6  add()
```

- ***Funktionsdefinition*** nach ***Schlüsselwort*** `def`
- Abschluss der Zeile mit “ : ”
- Einrückung ist in Python sehr wichtig!
- Aufruf der selbstdefinierten Funktion wie bei `print`
- Unsere Funktion hat keine Parameter. Die ***Parameterliste*** ist leer.
- Funktion kann mehrfach aufgerufen werden

Ausgabe:

```
Führe aus: hello.py
5
5
5
>>>
```

04 - Addition:

Funktion mit Parametern definieren

```
1  def add(x,y):
2      print(x+y)
3
4  add(2,3)
5  add(5,8)
6  add(1212332, 87982678)
```

- Funktion mit Parametern
- Parameter können innerhalb der Funktion verwendet werden
- Aufruf mit verschiedenen Werten ist möglich

Ausgabe:

```
Führe aus: hello.py
5
13
89195010
>>>
```

05 - Subtraktion: Zweite Funktion definieren

```
1  def add(x,y):
2      print(x+y)
3
4  def sub(x,y):
5      print(x-y)
6
7  add(5,3)
8  sub(5,3)
```

Die Definiton der Funktion muss vor dem ersten Aufruf stehen!

Ausgabe:

```
Führe aus: hello.py
8
2
>>>
```

06 - Calculator: Klasse definieren

```
1  class Calculator:
2      def add(self,x,y):
3          print(x+y)
4
5      def sub(self,x,y):
6          print(x-y)
7
8  calculator = Calculator()
9  calculator.add(5,3)
10 calculator.sub(5,3)
```

- Schlüsselwort `class` am Anfang der Definition
- Doppelpunkt am Zeilenende
- Einrückung ist wichtig!
- Schlüsselwort `def` zur Methodendefinition
- Der erste Parameter jeder ***Methode*** ist immer das Objekt selber. Die Benennung ist egal. Meist heißt der Parameter `self`
- Bei `calculator = Calculator()` wird ein ***Objekt*** der Klasse Calculator ***instanziiert*** und der Variablen `calculator` zugewiesen.
- Variablen erhalten bei der ersten Zuweisung ihren Typ. Der Typ kann sich danach nicht mehr ändern. Python ist ***stark typisiert.***
- Durch einen Punkt getrennt kann man eine Methode eines Objekts aufrufen, z.B. `calculator.add(5,3)` oder `calculator.sub(5,3)`

Bonus: 07 - Calculator: Klasse pimpen

```python
class Calculator:
    def __init__(self,name):
        self.name = name

    def add(self,x,y):
        print(self.name + " hat berechnet " + str(x) + " + " + str(y) + " = " + str(x+y))

    def sub(self,x,y):
        print(self.name + " hat berechnet " + str(x) + " - " + str(y) + " = " + str(x-y))

calculator = Calculator("Toller Taschenrechner")
calculator.add(5,3)
calculator.sub(5,3)
```

- ***Konstruktor*** mit Parameter
- Konstruktor definieren mit `__init__`
- Wert der Variablen `name` im Objekt merken

Ausgabe:

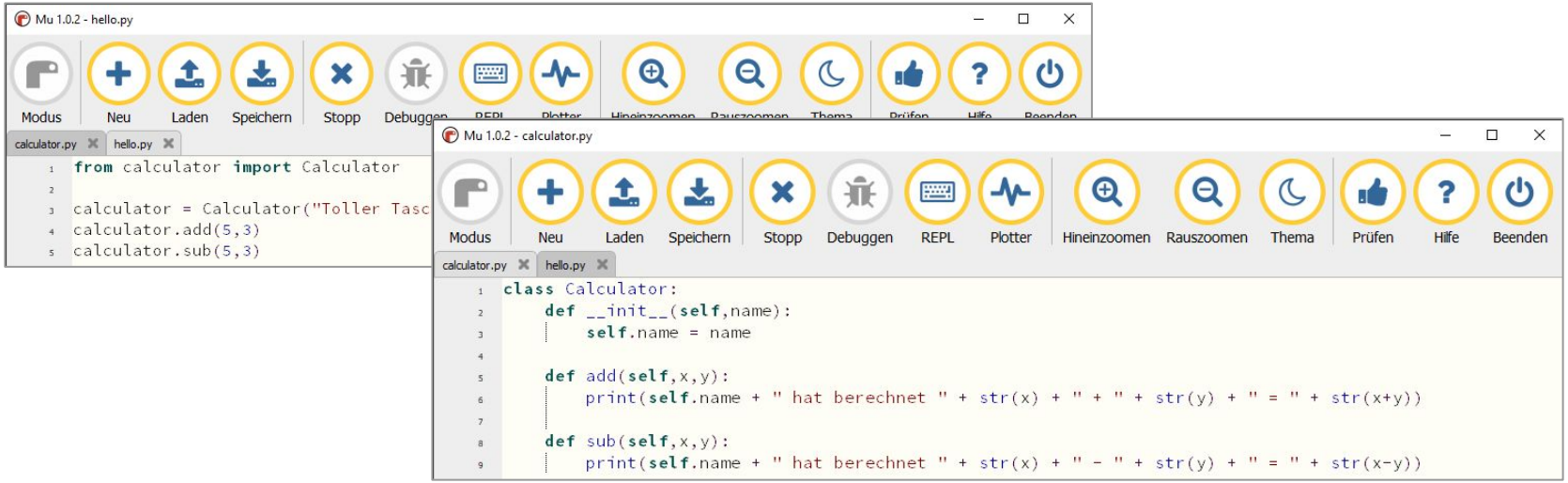
```
Toller Taschenrechner hat berechnet 5 + 3 = 8
Toller Taschenrechner hat berechnet 5 - 3 = 2
```

- Wenn bei `print` String und Zahlen gemischt werden sollen, müssen die Zahlen ***konvertiert*** werden: `str()`
- String-***Konkatenation*** (Aneinanderhängen) mit `+`

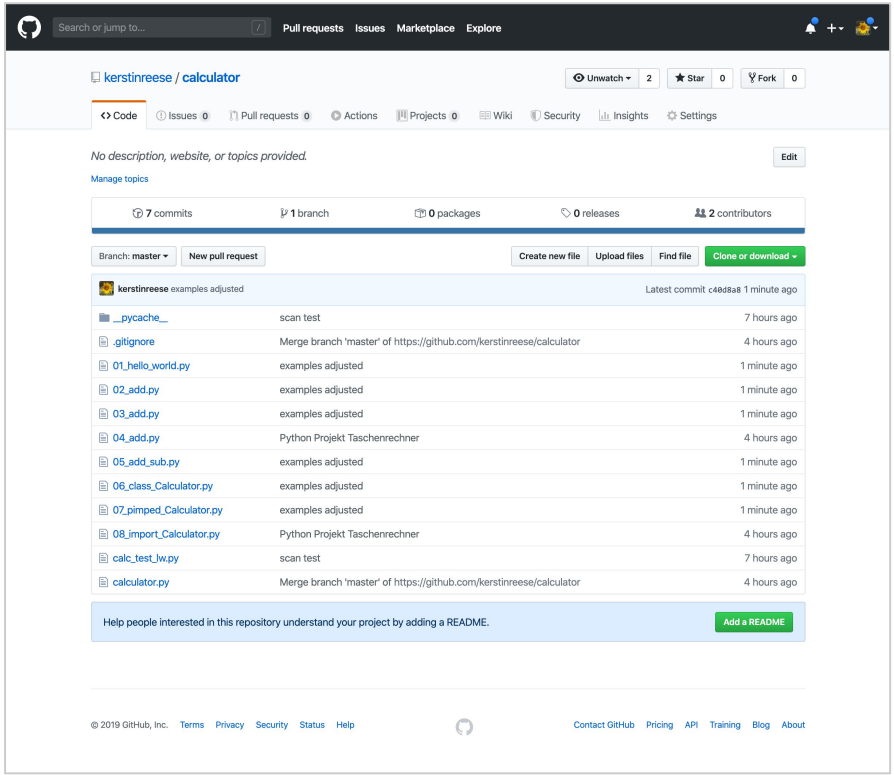
08 - Calculator: Import der Klasse aus Datei

```
1  from calculator import Calculator
2
3  calculator = Calculator("Toller Taschenrechner")
4  calculator.add(5,3)
5  calculator.sub(5,3)
```

Auslagern der Klassendefinition in eine eigene Datei



Alle Programme auf github



01_hello_world.py

02_add.py

03_add.py

04_add.py

05_add_sub.py

06_class_Calculator.py

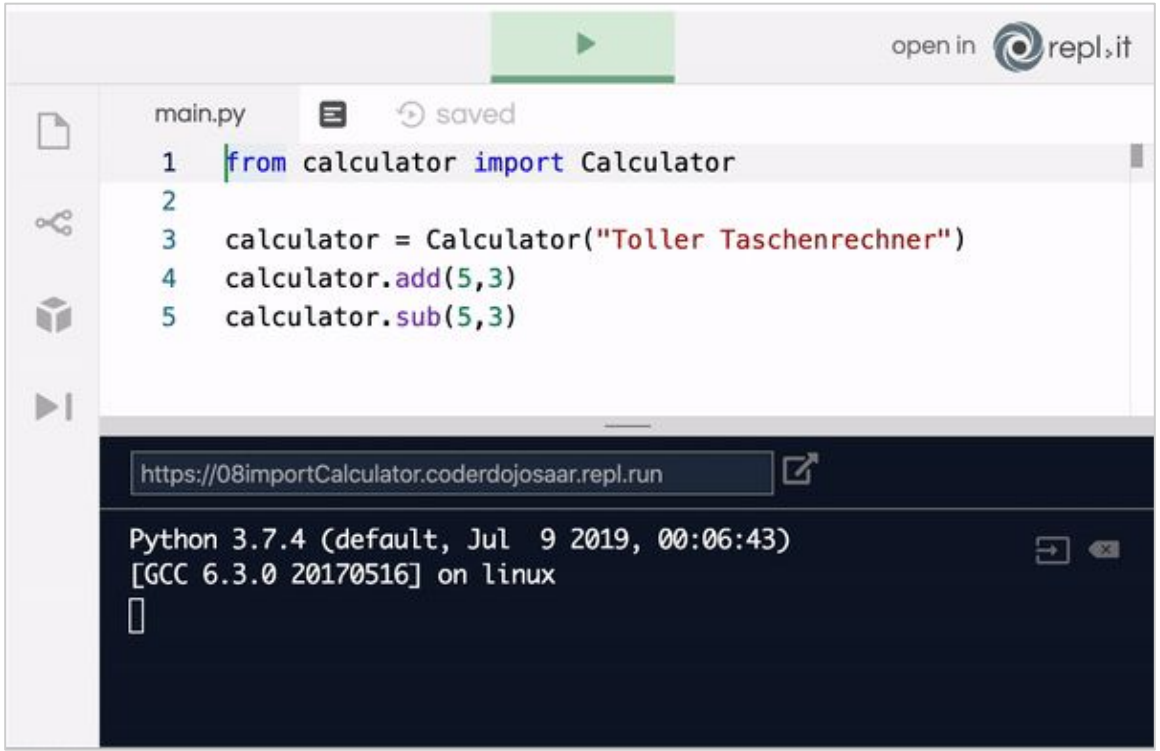
07_pimped_Calculator.py

08_import_Calculator.py

calculator.py

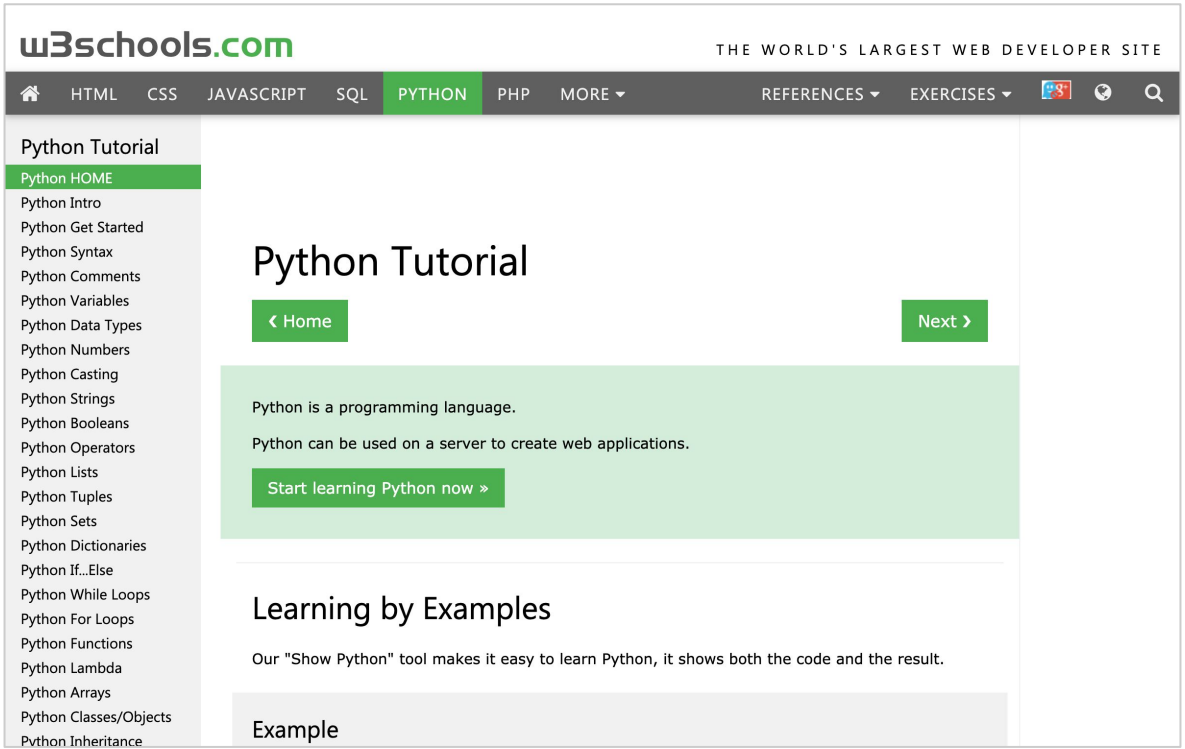
github.com/kerstinreese/calculator

Alle Programme auf repl.it



infolab.cs.uni-saarland.de/drohnen-mit-python-3-programmieren/

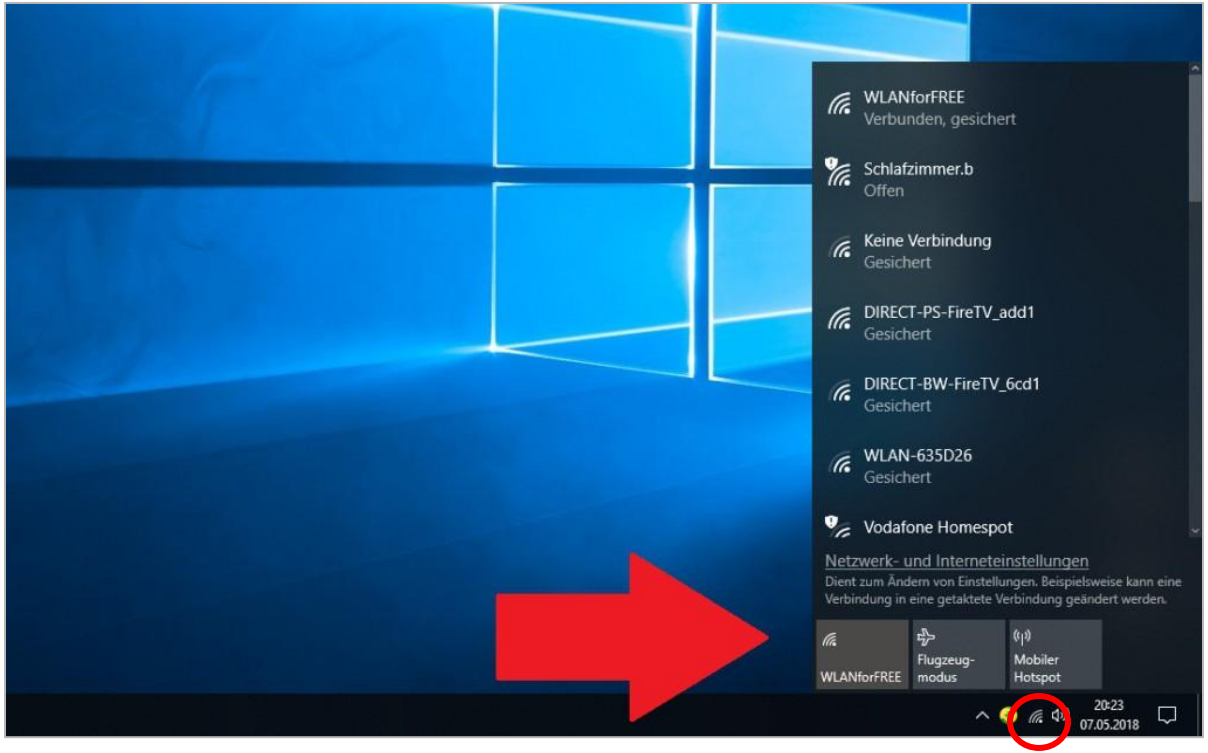
Python Tutorial



www.w3schools.com/python/

DJI Tello mit Python 3 programmieren

Laptop mit dem Wlan der Drohne verbinden



DJI Tello mit Python 3 programmieren

```
1  from tello import Tello
2
3  tello = Tello()
4  tello.send_command("command")
5  tello.send_command("takeoff")
6  tello.send_command("land")
```

Neue Datei erzeugen und im Ordner “tello” auf dem Desktop speichern!

- Datei `tello.py` mit Klassen-Definition `Tello` muss gefunden werden können!
- **Als erstes muss immer `command` an die Drohne geschickt werden.**

Kommandos für die DJI Tello

Immer als erstes!	`command`	`tello.send_command("command")`
Geschwindigkeit	`speed x` x = 10 -100 (in cm/s)	`tello.send_command("speed 10")`

Richtung	`up/down/`
	`left/right/`
	`forward/back x`
	x = 20 - 500 (in cm)
		`tello.send_command("up 20")`

Drehen	`cw/ccw x` (counter)clockwise x = 1 - 360 (in Grad)	`tello.send_command("cw 90")`
Flip	`flip x` x = `l` oder `r` oder `f` oder `b`	`tello.send_command("flip b")`
Stop / Schweben	`stop`	`tello.send_command("stop")`
Alle Motoren aus	`emergency`	`tello.send_command("emergency")`

Kommandos für die DJI Tello

Immer als erstes!	`command`	`tello.send_command("command")`
Geschwindigkeit	`speed x` x = 10 -100 (in cm/s)	`tello.send_command("speed 10")`

Richtung	`up/down/`
	`left/right/`
	`forward/back x`
	x = 20 - 500 (in cm)
		`tello.send_command("up 20")`

Drehen	`cw/ccw x` (counter)clockwise x = 1 - 360 (in Grad)	`tello.send_command("cw 90")`
Flip	`flip x` x = `l` oder `r` oder `f` oder `b`	`tello.send_command("flip b")`
Stop / Schweben	`stop`	`tello.send_command("stop")`
Alle Motoren aus	`emergency`	`tello.send_command("emergency")`

Präsentation und Links: infolab.cs.uni-saarland.de/drohnen-mit-python-3-programmieren/

Aufgabe

Die Drohne soll ein Quadrat fliegen!

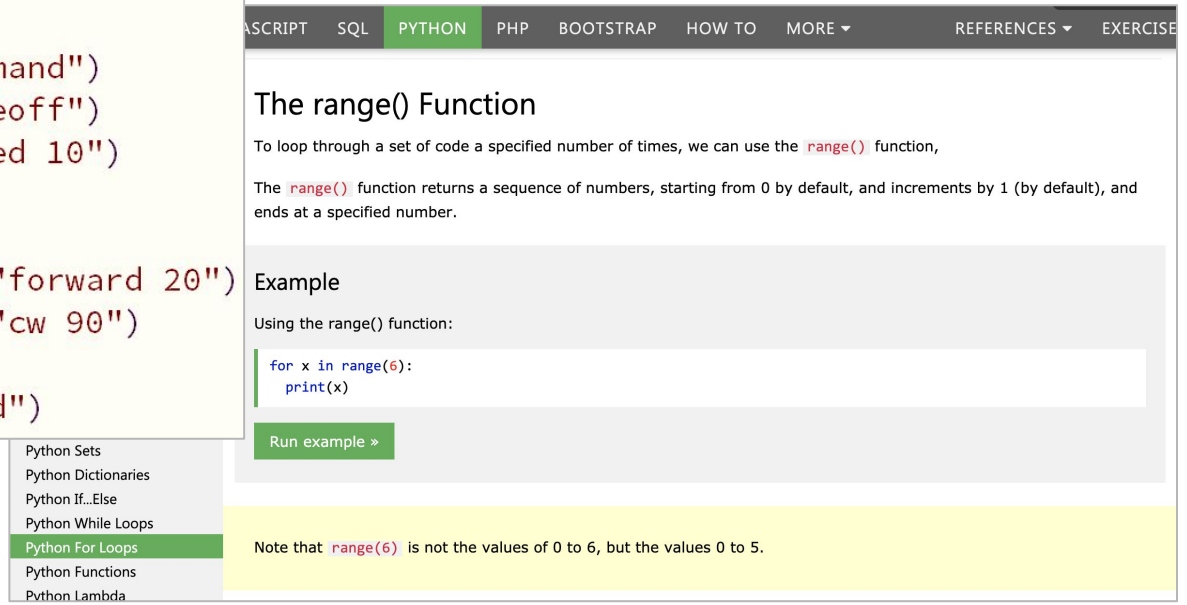
DJI Tello: Quadrat fliegen

```python
from tello import Tello

tello = Tello()
tello.send_command("command")
tello.send_command("takeoff")
tello.send_command("speed 10")

for i in range(4):
    tello.send_command("forward 20")
    tello.send_command("cw 90")

tello.send_command("land")
```



www.w3schools.com/python/python_for_loops.asp

Ideen für Drohnen-Programmierung

1. Andere Figuren fliegen, z.B. eine Acht

2. Drohne fragt einen Befehl ab, führt die Bewegung aus und direkt danach die entgegengesetzte Bewegung

3. Mathe-Drohne: Das Programm erzeugt Matheaufgaben und fragt nach der Lösung. Wenn die Lösung richtig war, fliegt die Drohne ein Kunststück zur Belohnung.

4. Schreibe Funktionen, mit denen die Drohne einfacher und lesbarer programmierbar ist, z.B. `forward(30), backward(20)`,...

5. Lies die Kommandos für die Drohne aus einer Datei.

Idee 1: Drohne fliegt eine Acht

```
1   from tello import Tello
2
3   tello = Tello()
4
5
6   def fly8(n):
7       for i in range(n + 1):
8           tello.send_command("cw 45")
9           tello.send_command("forward 100")
10          tello.send_command("cw 135")
11          tello.send_command("forward 60")
12          tello.send_command("cw 135")
13          tello.send_command("forward 100")
14          tello.send_command("ccw 45")
15          tello.send_command("forward 60")
16          tello.send_command("ccw 180")
17
18
19  tello.send_command("takeoff")
20
21  fly8(2)
22
23  tello.send_command("land")
24
```

- Import der Definition der Klasse `Tello` aus der Datei `tello.py`
- Instanziierung eine Objekts der Klasse `Tello`
- Definition der Funktion `fly8` mit Parameter `n`
- `for`– Schleife, die den Parameter `n` benutzt
- Befehle innerhalb der `for`– Schleife
- Aufruf der Funktion `send_command` des Objekts `tello`
- Aufruf der Funktion `fly8` mit Parameter `2`
- Aufruf der Funktion `send_command` des Objekts `tello`

Idee 2: Drohne kann zurückkommen

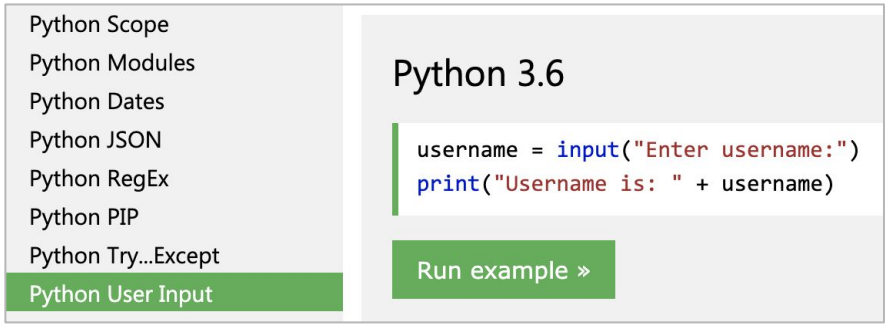
```python
from tello import Tello

def queue_com():
    com = input("Next command: ")
    if com != "":
        tello.send_command(com)
        queue_com()
        tello.send_command(negate(com))
    else:
        pass


def negate(com):
    switch = {
        "takeoff": "land",
        "forward 50": "back 50",
        "up 50": "down 50",
        "cw 90": "ccw 90"
    }
    return switch.get(com, "emergency")

tello = Tello()

tello.send_command("command")
queue_com()
```



Idee 3: Mathe-Drohne

```
from tello import Tello
import random

tello = Tello()


def createExercise():
    x = random.randint(0, 10)
    y = random.randint(0, 10)
    result = x + y
    print("Was ist " + str(x) + " + " + str(y) + "?\n")
    return result


tello.send_command("command")
tello.send_command("takeoff")

result = createExercise()
if input("Ergebnis: ") == str(result):
    tello.send_command("flip b")
else:
    pass

tello.send_command("land")
```

Idee 4: schönere Befehle für die Drohne

```
from tello import Tello


def forward(x):
    tello.send_command("forward " + str(x))


def turnLeft(x):
    tello.send_command("ccw " + str(x))


def begin():
    tello.send_command("command")
    tello.send_command("takeoff")


def end():
    tello.send_command("land")


tello = Tello()
begin()

# put your code here

end()
```

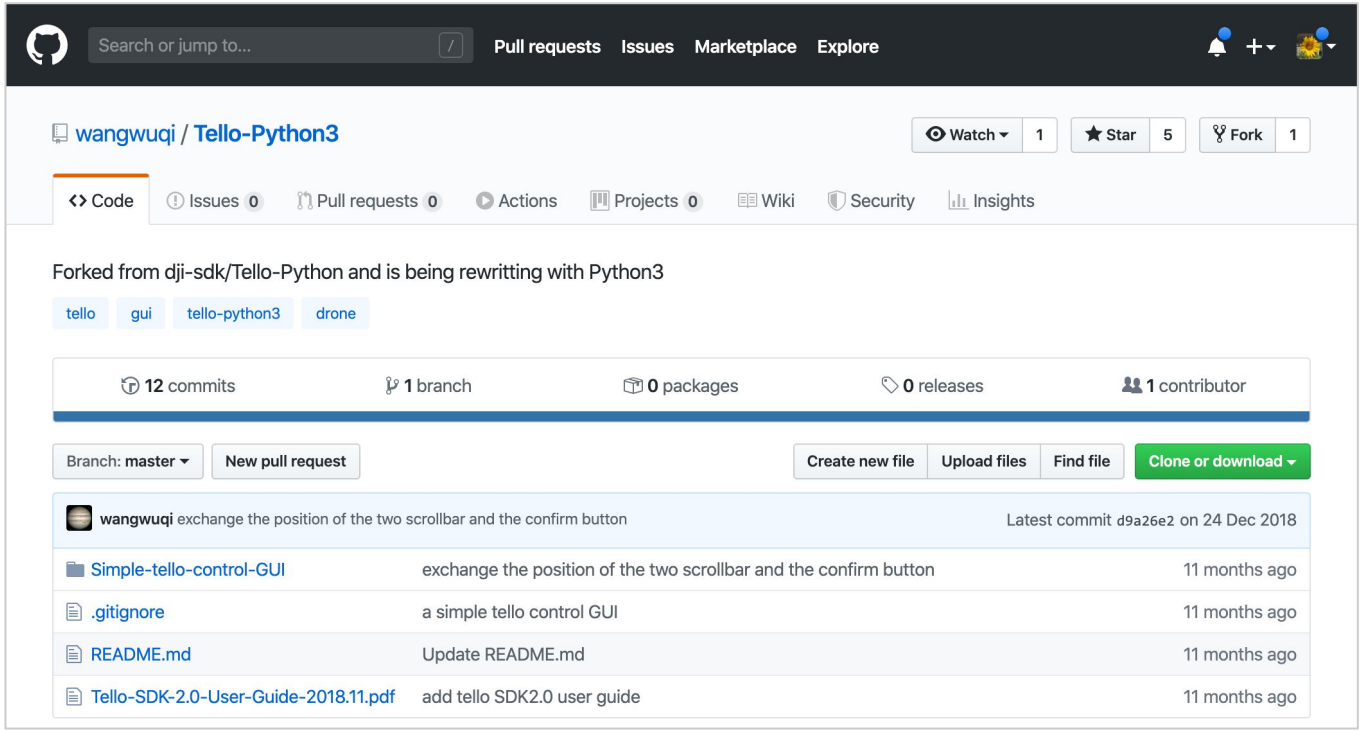
Idee 5: Befehle aus Datei lesen

```
1   from tello import Tello
2   import time
3
4   f = open('commands.txt', "r")
5   commands = f.readlines()
6
7   tello = Tello()
8   for command in commands:
9       if command != '' and command != '\n':
10          command = command.rstrip()
11
12          if command.find('delay') != -1:
13              sec = float(command.partition('delay')[2])
14              print('delay %s'%(sec))
15              time.sleep(sec)
16              pass
17          else:
18              tello.send_command(command)
```

```
command
takeoff
delay 1
up 20
delay 1
foward 20
delay 1
right 20
delay 1
back 20
delay 1
left 20
delay 1
land
```

commands.txt

FYI: Repo mit Tello-Bibliothek



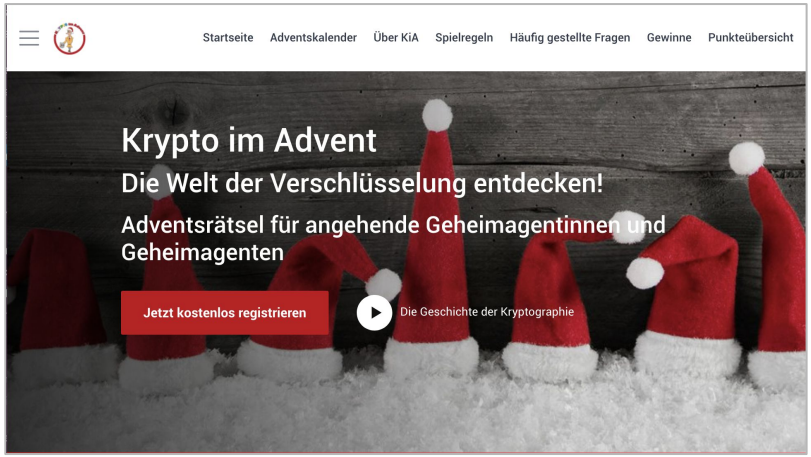
github.com/wangwuqi/Tello-Python3

Informatik-Fachbegriffe

Editor	Funktionsdefinition	Instanziierung
Konsole	Syntax	stark typisiert
Funktion (aufrufen)	Parameterliste	Zuweisung
Parameter	Klasse	Konstruktor
Zeichenkette/String	Klassenmethode	Konvertierung
Number/Integer	Objekt	Konkatenation

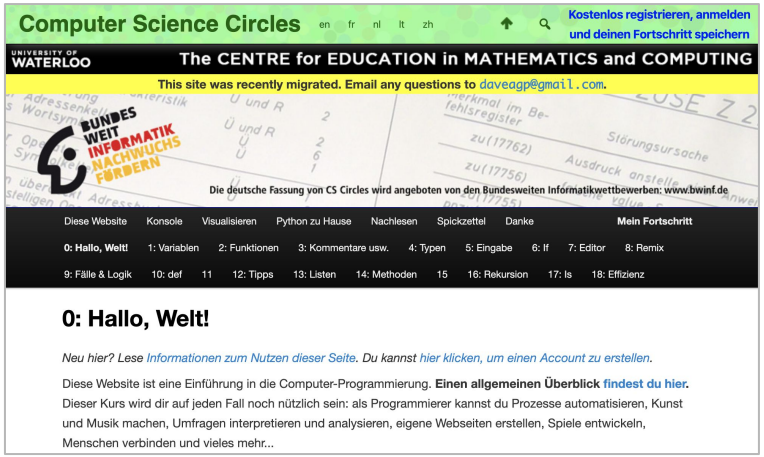
Einladungen

Krypto im Advent - ab 01.12.



www.krypto-im-advent.de

“Hour of Code” mit Python - 09.-15.12.



hourofcode.com/de
bwinf.de/jugendwettbewerb/programmierenlernen/python/