



# Drohnen programmieren

mit Python 3 und der DJI Tello



---

# Heute

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

---

# DJI Tello kennenlernen und fliegen

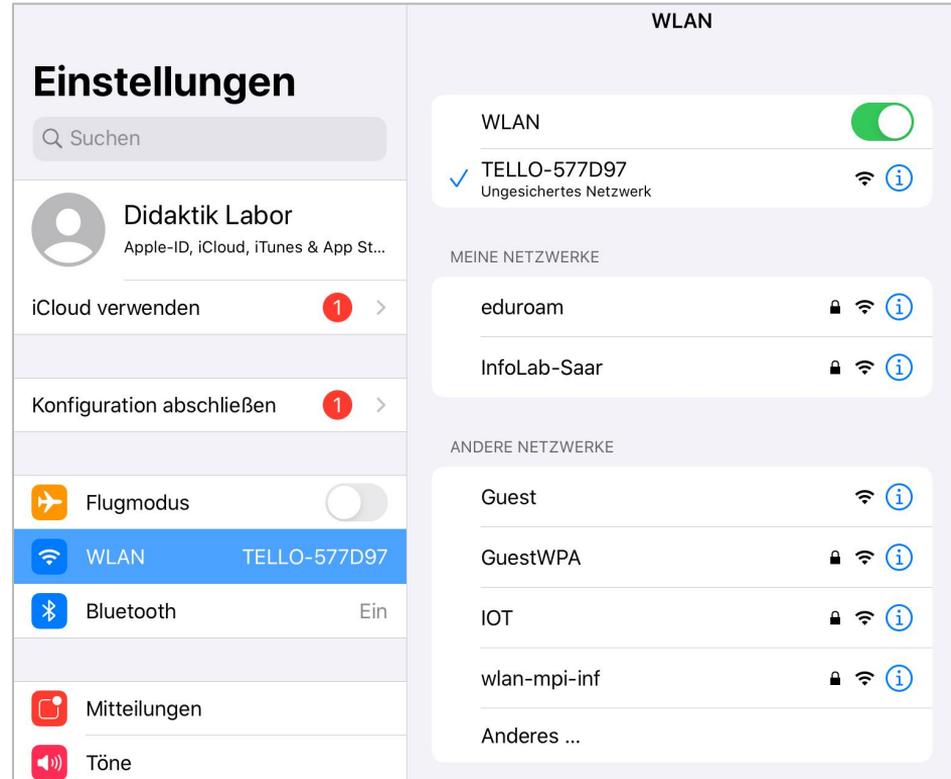
# Bestandteile der DJI Tello



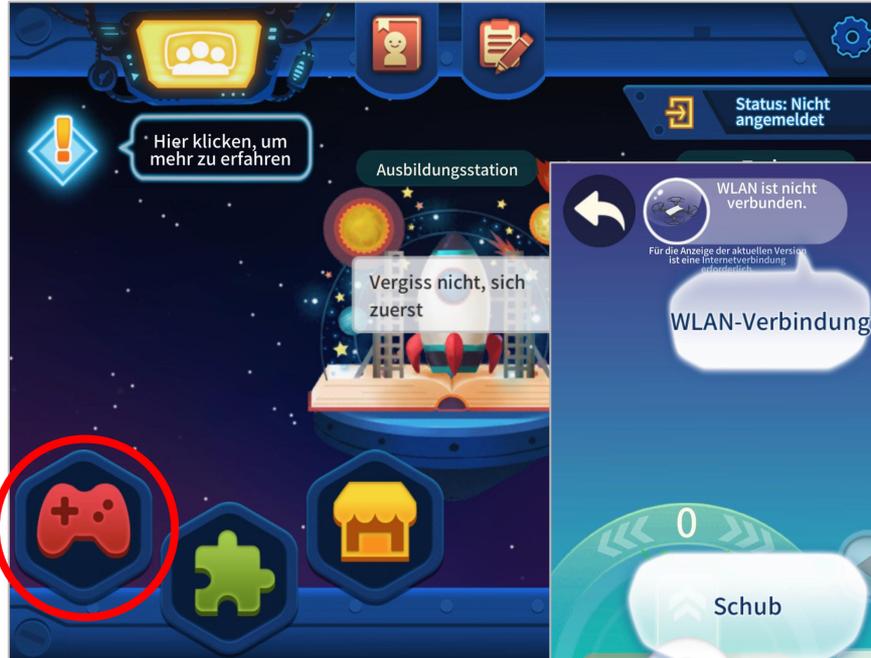
# Wlan verbinden



Mit Drohne verbinden  
(Nummer beachten !)



# App "Tello Edu" starten



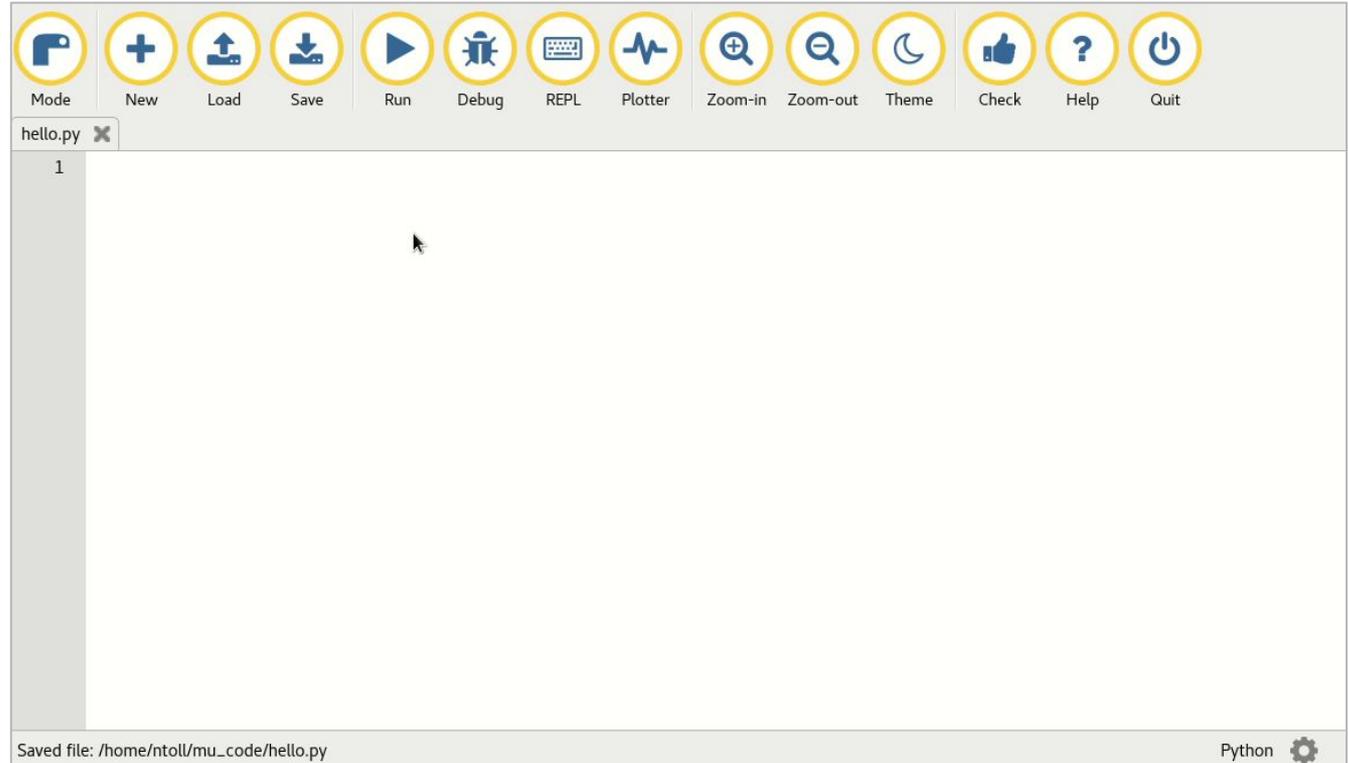
---

# Python 3 Basics

# Mu Editor für Python 3



[codewith.mu](http://codewith.mu)



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

# 01 - "Hello World!":

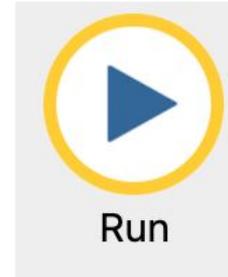
## Funktion mit String-Parameter aufrufen

```
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

### Ausgabe in der Konsole:

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



Beim ersten Mal muss gespeichert werden

## 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 Definition 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

```
1 class Calculator:
2     def __init__(self, name):
3         self.name = name
4
5     def add(self, x, y):
6         print(self.name + " hat berechnet " + str(x) + " + " + str(y) + " = " + str(x+y))
7
8     def sub(self, x, y):
9         print(self.name + " hat berechnet " + str(x) + " - " + str(y) + " = " + str(x-y))
10
11 calculator = Calculator("Toller Taschenrechner")
12 calculator.add(5,3)
13 calculator.sub(5,3)
```

- **Konstruktor** mit Parameter
- Konstruktor definieren mit `__init__`
- Wert der Variablen `name` im Objekt merken
- Wenn bei `print` String und Zahlen gemischt werden sollen, müssen die Zahlen **konvertiert** werden: `str()`
- **String-Konkatenation** (Aneinanderhängen) mit `+`

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

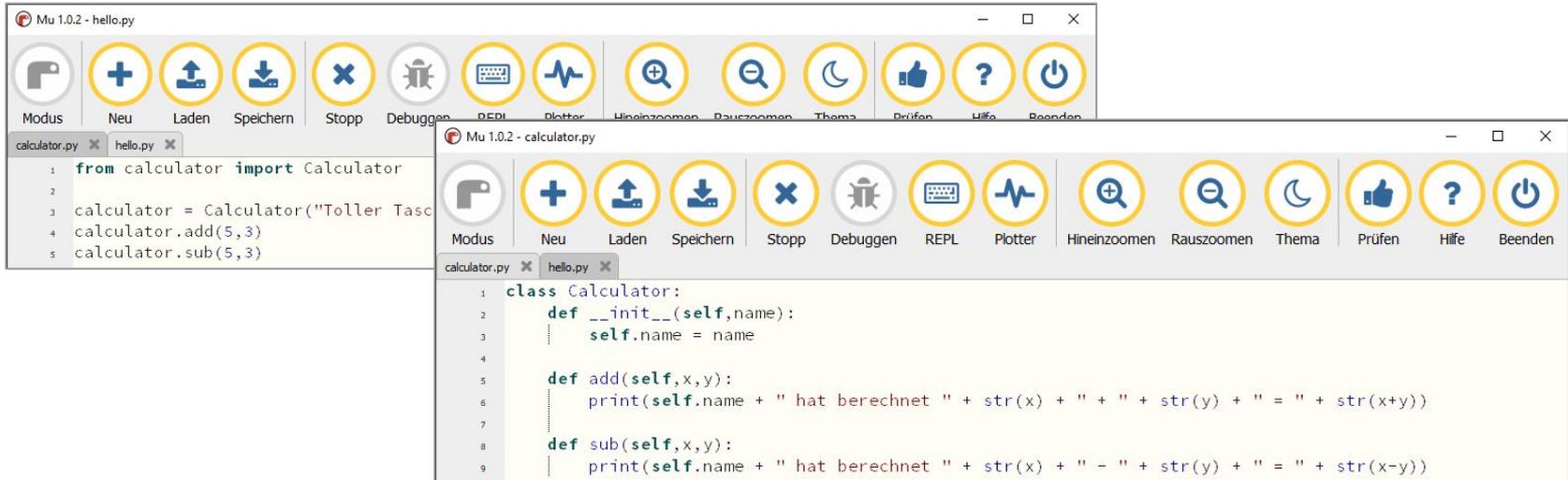
# 08 - Calculator: Import der Klasse aus Datei

```

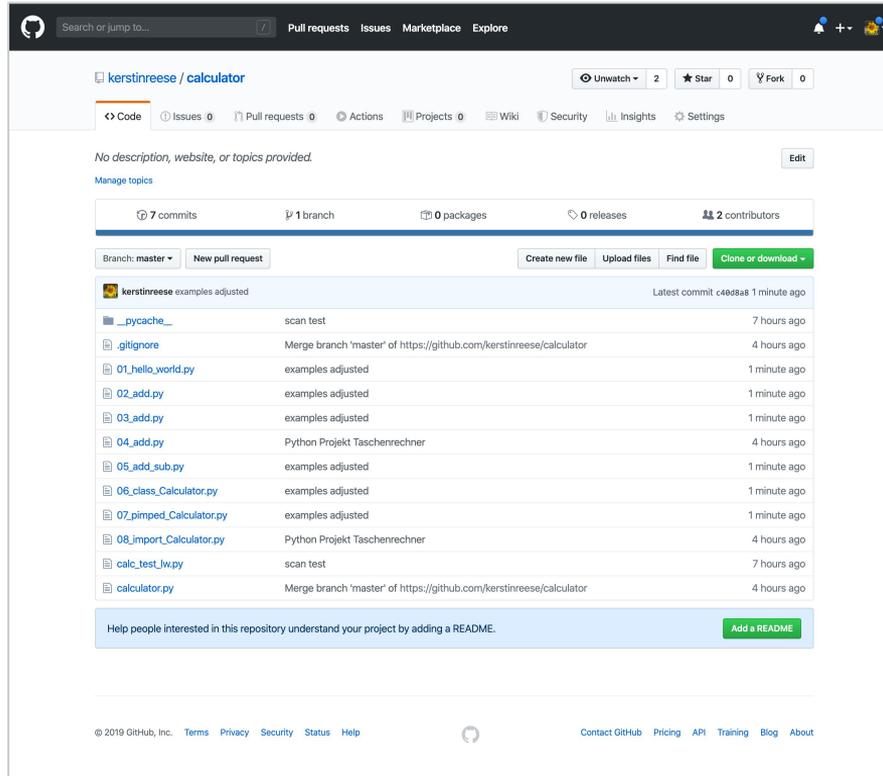
1 from calculator import Calculator
2
3 calculator = Calculator("Toller Taschenrechner")
4 calculator.add(5,3)
5 calculator.sub(5,3)

```

Auslagern der Klassendefinition in eine eigene Datei



# Alle Programme auf github



The screenshot shows the GitHub repository page for `kerstinreese/calculator`. The repository has 7 commits, 1 branch, 0 packages, 0 releases, and 2 contributors. The file list includes:

File Name	Description	Last Commit	Time Ago
<code>__pycache__</code>	scan test	c48da8	7 hours ago
<code>.gitignore</code>	Merge branch 'master' of https://github.com/kerstinreese/calculator		4 hours ago
<code>01_hello_world.py</code>	examples adjusted		1 minute ago
<code>02_add.py</code>	examples adjusted		1 minute ago
<code>03_add.py</code>	examples adjusted		1 minute ago
<code>04_add.py</code>	Python Projekt Taschenrechner		4 hours ago
<code>05_add_sub.py</code>	examples adjusted		1 minute ago
<code>06_class_Calculator.py</code>	examples adjusted		1 minute ago
<code>07_pimped_Calculator.py</code>	examples adjusted		1 minute ago
<code>08_import_Calculator.py</code>	Python Projekt Taschenrechner		4 hours ago
<code>calc_test_hw.py</code>	scan test		7 hours ago
<code>calculator.py</code>	Merge branch 'master' of https://github.com/kerstinreese/calculator		4 hours ago

 [01\\_hello\\_world.py](#)

 [02\\_add.py](#)

 [03\\_add.py](#)

 [04\\_add.py](#)

 [05\\_add\\_sub.py](#)

 [06\\_class\\_Calculator.py](#)

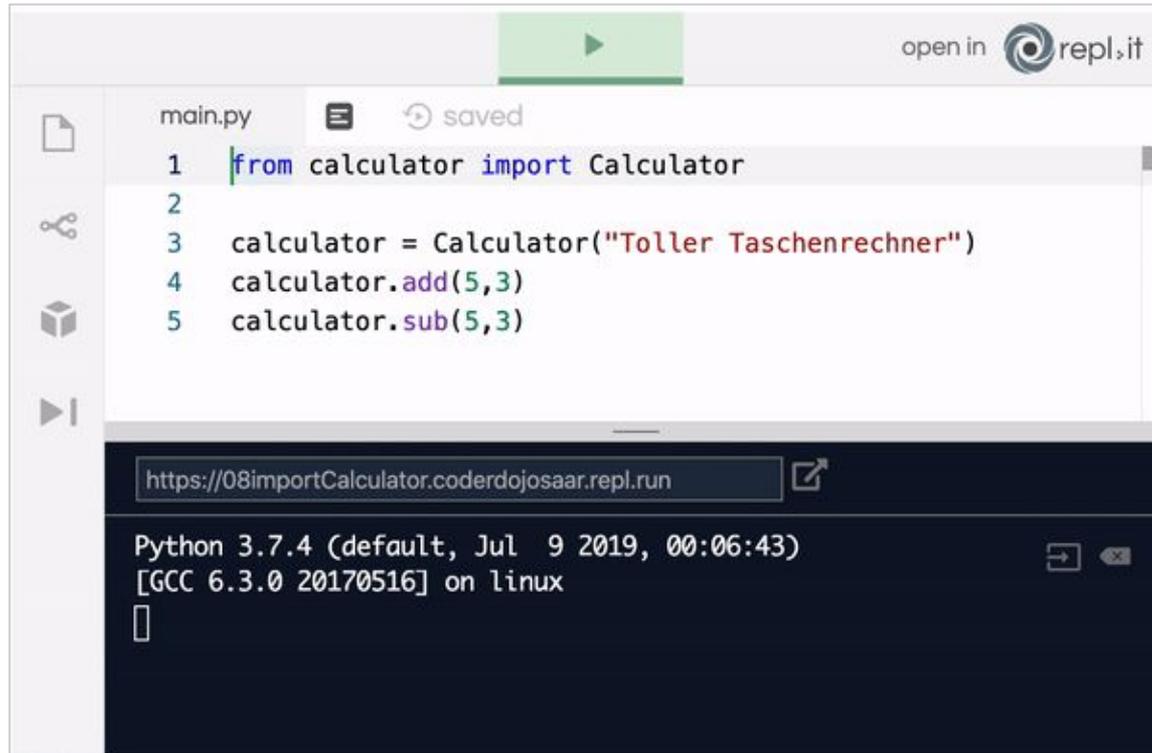
 [07\\_pimped\\_Calculator.py](#)

 [08\\_import\\_Calculator.py](#)

 [calculator.py](#)

[github.com/kerstinreese/calculator](https://github.com/kerstinreese/calculator)

# Alle Programme auf repl.it



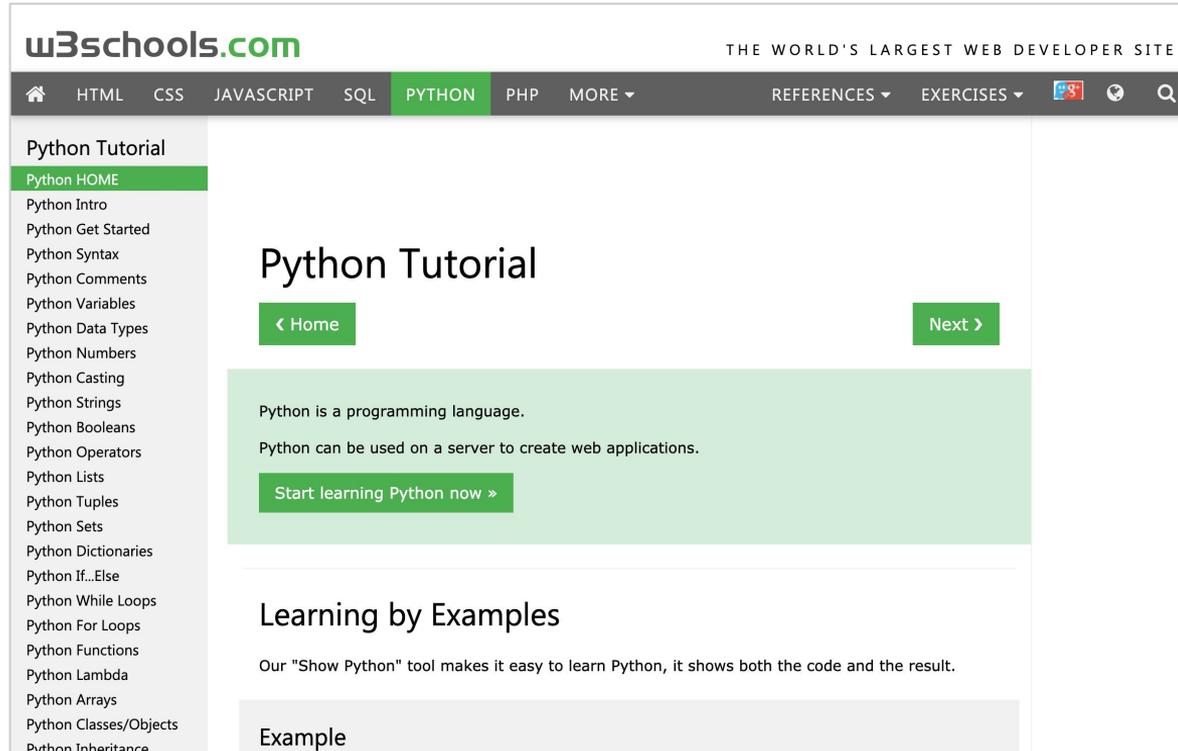
The screenshot shows a Repl.it Python REPL interface. At the top right, there is a green play button and a link that says "open in repl.it". Below this, the file name "main.py" is shown with a "saved" status. The code editor contains the following Python code:

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

Below the code editor, there is a terminal window with the following output:

```
https://08importCalculator.coderdojosaar.repl.run
Python 3.7.4 (default, Jul 9 2019, 00:06:43)
[GCC 6.3.0 20170516] on linux
█
```

# Python Tutorial



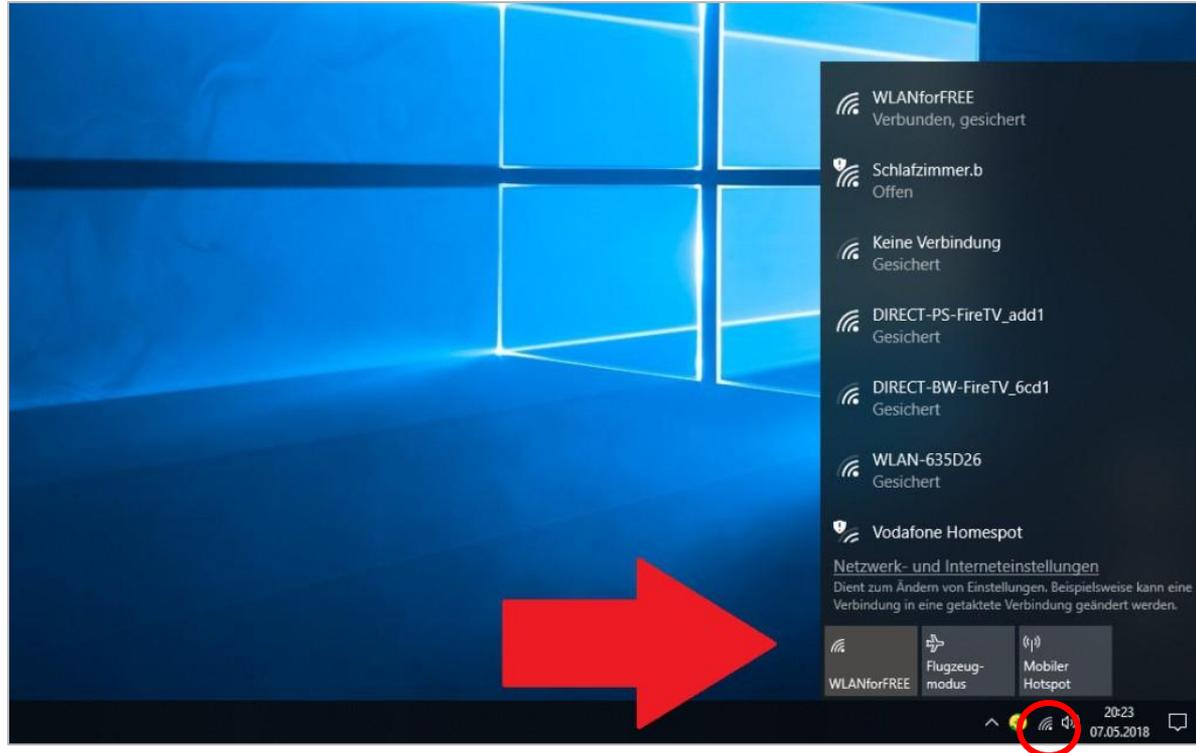
The screenshot shows the Python Tutorial page on w3schools.com. The page has a dark navigation bar with links for HTML, CSS, JAVASCRIPT, SQL, PYTHON (highlighted), PHP, and MORE. There are also dropdown menus for REFERENCES and EXERCISES, and icons for social media and search. A left sidebar lists various Python topics, with 'Python HOME' highlighted. The main content area features the title 'Python Tutorial', navigation buttons for '< Home' and 'Next >', and a green call-to-action box that says 'Start learning Python now >'. Below this, there is a section titled 'Learning by Examples' with a brief description of the 'Show Python' tool. At the bottom, there is a section titled 'Example'.

[www.w3schools.com/python/](http://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	<code>tello.send_command("command")</code>
Geschwindigkeit	speed x x = 10 - 100 (in cm/s)	<code>tello.send_command("speed 10")</code>
Richtung	up/down/ left/right/ forward/back x x = 20 - 500 (in cm)	<code>tello.send_command("up 20")</code>
Drehen	cw/ccw x (counter)clockwise x = 1 - 360 (in Grad)	<code>tello.send_command("cw 90")</code>
Flip	flip x x = l oder r oder f oder b	<code>tello.send_command("flip b")</code>
Stop / Schweben	stop	<code>tello.send_command("stop")</code>
Alle Motoren aus	emergency	<code>tello.send_command("emergency")</code>

## 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")



# Aufgabe



Die Drohne soll ein Quadrat fliegen!

# DJI Tello: Quadrat fliegen

```

1 from tello import Tello
2
3 tello = Tello()
4 tello.send_command("command")
5 tello.send_command("takeoff")
6 tello.send_command("speed 10")
7
8 for i in range(4):
9     tello.send_command("forward 20")
10    tello.send_command("cw 90")
11
12 tello.send_command("land")

```

- Python Sets
- Python Dictionaries
- Python If...Else
- Python While Loops
- Python For Loops
- Python Functions
- Python Lambda

ASCRIP
SQL
PYTHON
PHP
BOOTSTRAP
HOW TO
MORE ▾
REFERENCES ▾
EXERCISE

## The range() Function

To loop through a set of code a specified number of times, we can use the `range()` function,

The `range()` function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

### Example

Using the range() function:

```

for x in range(6):
    print(x)

```

Run example >

Note that `range(6)` is not the values of 0 to 6, but the values 0 to 5.

[www.w3schools.com/python/python\\_for\\_loops.asp](http://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

<pre> 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 </pre>	<p>← Import der Definition der Klasse <code>Tello</code> aus der Datei <code>tello.py</code></p> <p>← Instanziierung eines Objekts der Klasse <code>Tello</code></p> <p>← Definition der Funktion <code>fly8</code> mit Parameter <code>n</code></p> <p>← <code>for</code>-Schleife, die den Parameter <code>n</code> benutzt</p> <p>← Befehle innerhalb der <code>for</code>-Schleife</p> <p>← Aufruf der Funktion <code>send_command</code> des Objekts <code>tello</code></p> <p>← Aufruf der Funktion <code>fly8</code> mit Parameter <code>2</code></p> <p>← Aufruf der Funktion <code>send_command</code> des Objekts <code>tello</code></p>
--	--

# Idee 2: Drohne kann zurückkommen

```
1 from tello import Tello
2
3 def queue_com():
4     com = input("Next command: ")
5     if com != "":
6         tello.send_command(com)
7         queue_com()
8         tello.send_command(negate(com))
9     else:
10        pass
11
12
13 def negate(com):
14     switch = {
15         "takeoff": "land",
16         "forward 50": "back 50",
17         "up 50": "down 50",
18         "cw 90": "ccw 90"
19     }
20     return switch.get(com, "emergency")
21
22 tello = Tello()
23
24 tello.send_command("command")
25 queue_com()
```

Python Scope

Python Modules

Python Dates

Python JSON

Python RegEx

Python PIP

Python Try...Except

Python User Input

## Python 3.6

```
username = input("Enter username:")
print("Username is: " + username)
```

Run example »

Python Operators

Python Lists

Python Tuples

Python Sets

Python Dictionaries

Python If...Else

Python While Loops

Python For Loops

Python Functions

Python Lambda

## Example

Get the value of the "model" key:

```
x = thisdict.get("model")
```

Run example »

## Idee 3: Mathe-Drohne

```
1 from tello import Tello
2 import random
3
4 tello = Tello()
5
6
7 def createExercise():
8     x = random.randint(0, 10)
9     y = random.randint(0, 10)
10    result = x + y
11    print("Was ist " + str(x) + " + " + str(y) + "=?\n")
12    return result
13
14
15 tello.send_command("command")
16 tello.send_command("takeoff")
17
18 result = createExercise()
19 if input("Ergebnis: ") == str(result):
20     tello.send_command("flip b")
21 else:
22     pass
23
24 tello.send_command("land")
25 |
```

# Idee 4: schönere Befehle für die Drohne

```
1 from tello import Tello
2
3
4 def forward(x):
5     tello.send_command("forward " + str(x))
6
7
8 def turnLeft(x):
9     tello.send_command("ccw " + str(x))
10
11
12 def begin():
13     tello.send_command("command")
14     tello.send_command("takeoff")
15
16
17 def end():
18     tello.send_command("land")
19
20
21 tello = Tello()
22 begin()
23
24 # put your code here
25
26 end()
```

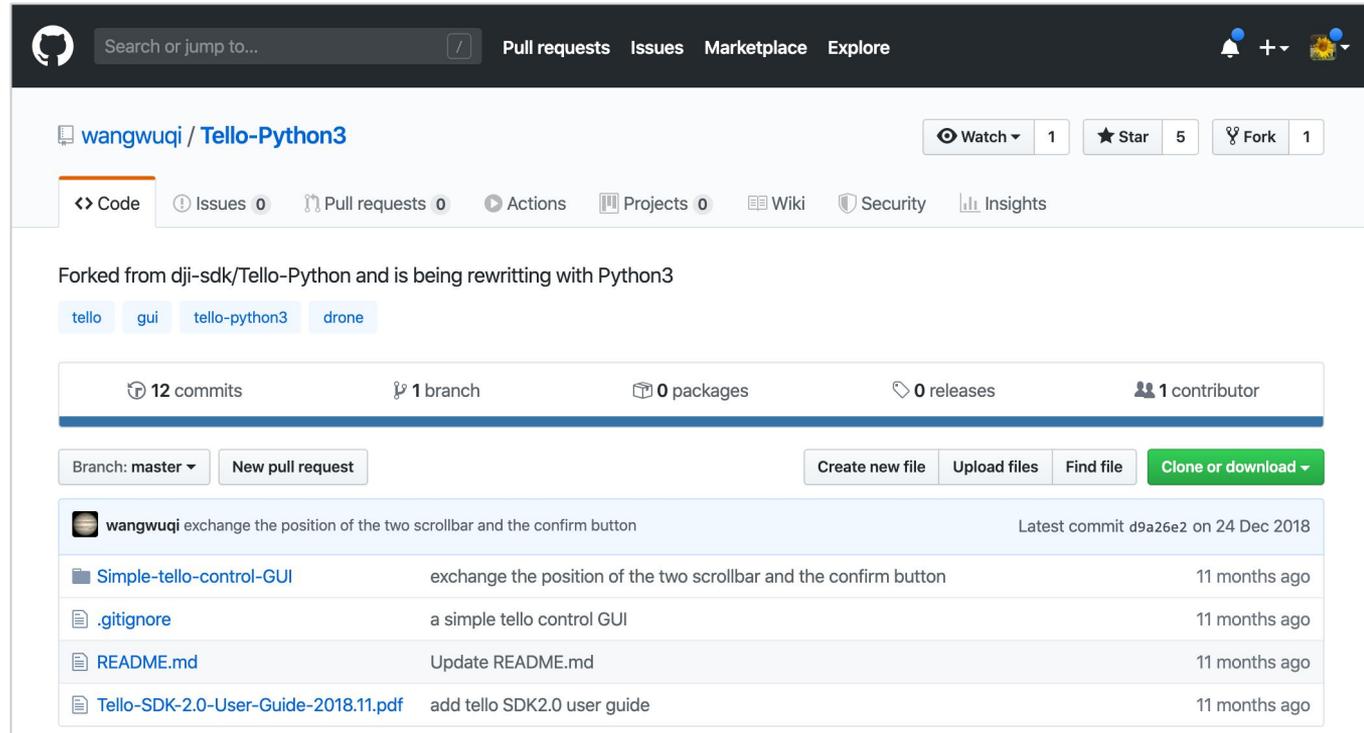
## Idee 5: Befehle aus Datei lesen

```
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



The screenshot shows the GitHub interface for the repository `wangwuqi / Tello-Python3`. At the top, there is a search bar and navigation links for Pull requests, Issues, Marketplace, and Explore. The repository name is followed by statistics: 1 Watch, 5 Stars, and 1 Fork. Below this, there are tabs for Code, Issues (0), Pull requests (0), Actions, Projects (0), Wiki, Security, and Insights. A note indicates the repository is forked from `dji-sdk/Tello-Python` and is being rewritten with Python3. There are tags for `tello`, `gui`, `tello-python3`, and `drone`. A summary bar shows 12 commits, 1 branch, 0 packages, 0 releases, and 1 contributor. Below this, there are buttons for Branch: master, New pull request, Create new file, Upload files, Find file, and Clone or download. The commit history shows a recent commit by wangwuqi: "exchange the position of the two scrollbar and the confirm button" (Latest commit d9a26e2 on 24 Dec 2018). Below the commit history, there is a list of files:

File Name	Description	Time
<code>Simple-tello-control-GUI</code>	exchange the position of the two scrollbar and the confirm button	11 months ago
<code>.gitignore</code>	a simple tello control GUI	11 months ago
<code>README.md</code>	Update README.md	11 months ago
<code>Tello-SDK-2.0-User-Guide-2018.11.pdf</code>	add tello SDK2.0 user guide	11 months ago

[github.com/wangwuqi/Tello-Python3](https://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

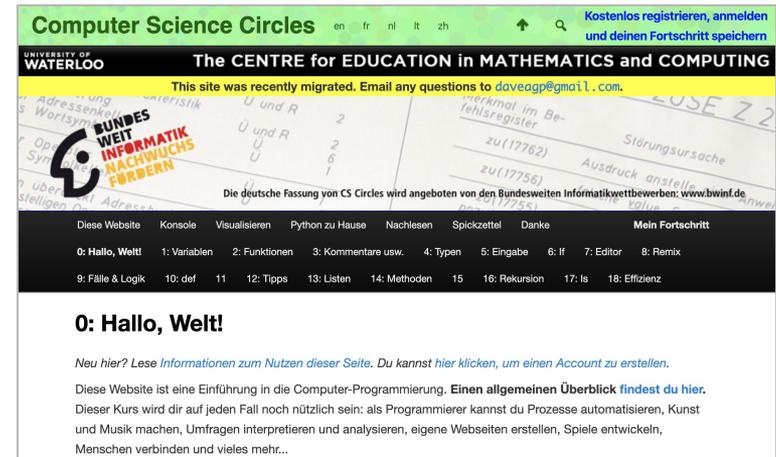
# Einladungen

## Krypto im Advent - ab 01.12.



[www.krypto-im-advent.de](http://www.krypto-im-advent.de)

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



[hourofcode.com/de](http://hourofcode.com/de)  
[bwinf.de/jugendwettbewerb/programmierenlernen/python/](http://bwinf.de/jugendwettbewerb/programmierenlernen/python/)