

---

# Drohnen programmieren

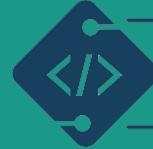
mit Python 3 und der DJI Tello



---

# Heute

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



InfoLabSaar

---

# DJI Tello kennenlernen und fliegen

# Bestandteile der DJI Tello



2. Drohne einschalten



1. Akku einsetzen



3. Mit Drohne verbinden  
(Nummer beachten !)

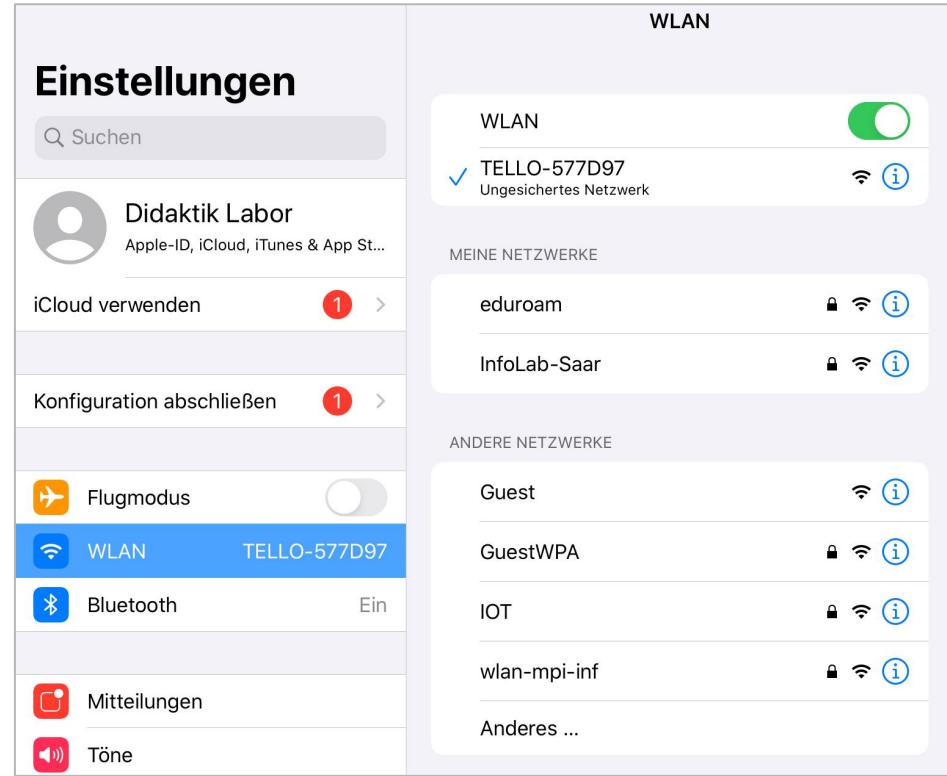


4. App starten →  
Joystick Modus wählen

# Wlan verbinden



Mit Drohne verbinden  
(Nummer beachten !)



The screenshot shows the 'Einstellungen' (Settings) screen on an iOS device. The 'WLAN' section is highlighted. The 'WLAN' switch is turned on, and the network 'TELLO-577D97' is listed as an 'Ungesichertes Netzwerk' (unsecured network) with a signal strength icon and an information button.

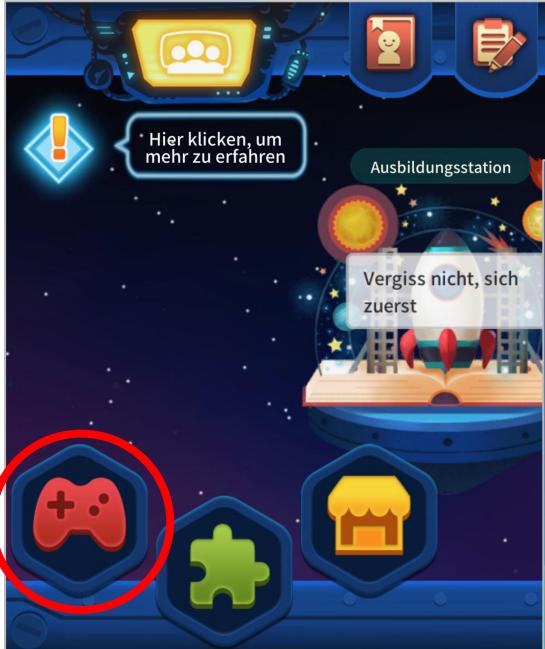
WLAN	
WLAN	<input checked="" type="checkbox"/>
TELLO-577D97	Ungesichertes Netzwerk

Below the WLAN section, there are sections for 'MEINE NETZWERKE' (My Networks) and 'ANDERE NETZWERKE' (Other Networks), each listing available networks with their respective icons and connection status.

MEINE NETZWERKE	
eduroam	
InfoLab-Saar	

ANDERE NETZWERKE	
Guest	
GuestWPA	
IOT	
wlan-mpi-inf	
Anderes ...	

# App “Tello Edu” starten





Info**Lab**Saar

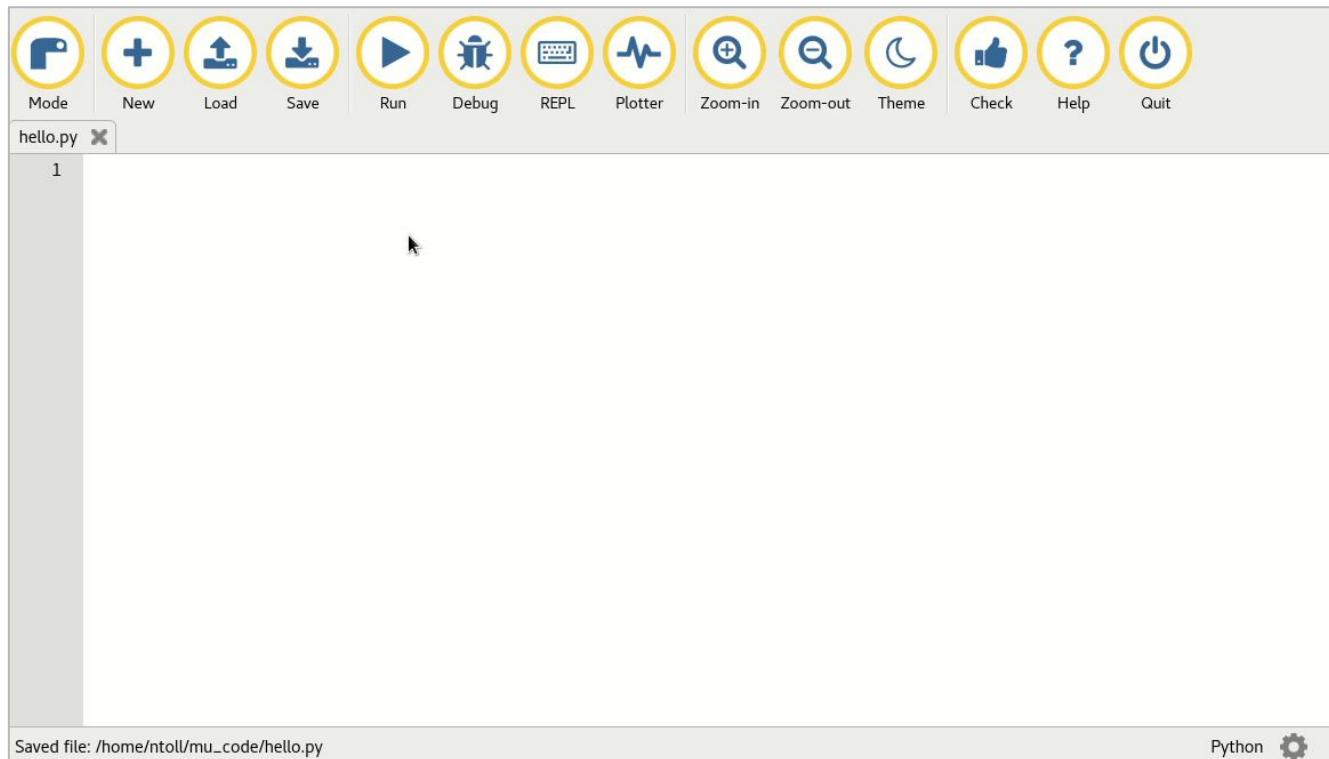
---

# Python 3 Basics

# Mu Editor für Python 3



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



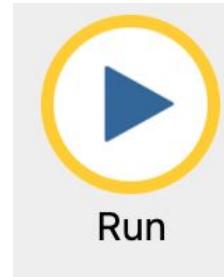
Python Programme im Ordner “Python” auf dem Desktop speichern!  
Falls eine Datei geöffnet ist, bitte schließen.

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

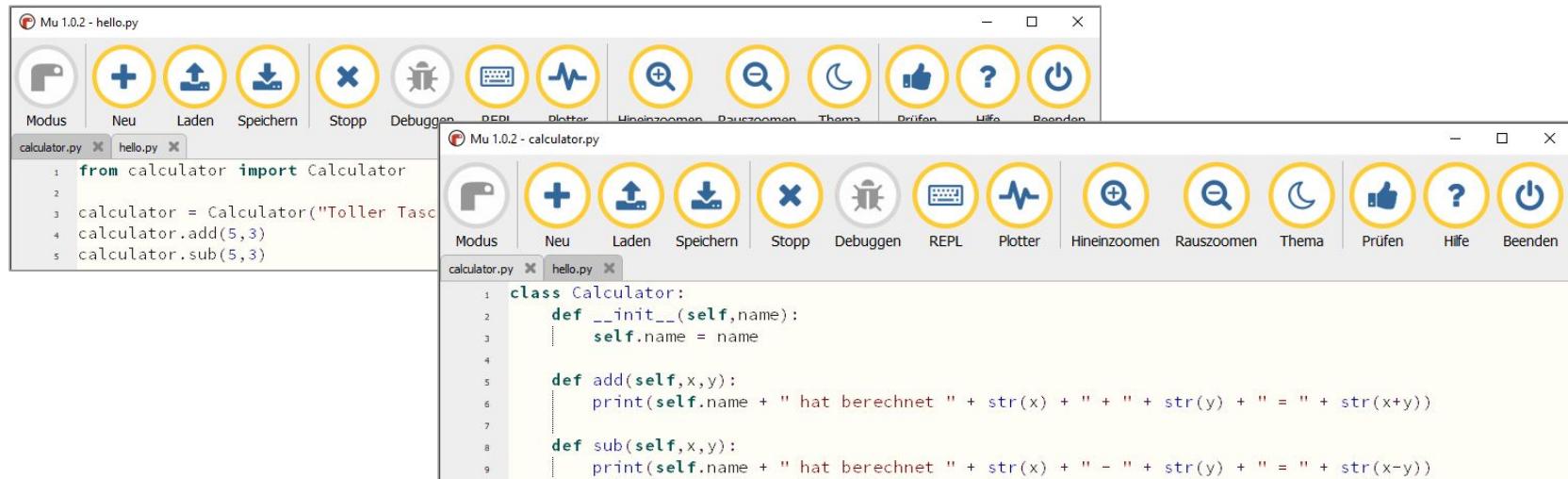
Toller Taschenrechner hat berechnet  $5 + 3 = 8$   
 Toller Taschenrechner hat berechnet  $5 - 3 = 2$

# 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



The image shows two instances of the Mu Python IDE running side-by-side. Both windows have identical toolbars at the top with various icons for file operations, debugging, and help.

**Left Window (calculator.py):**

```

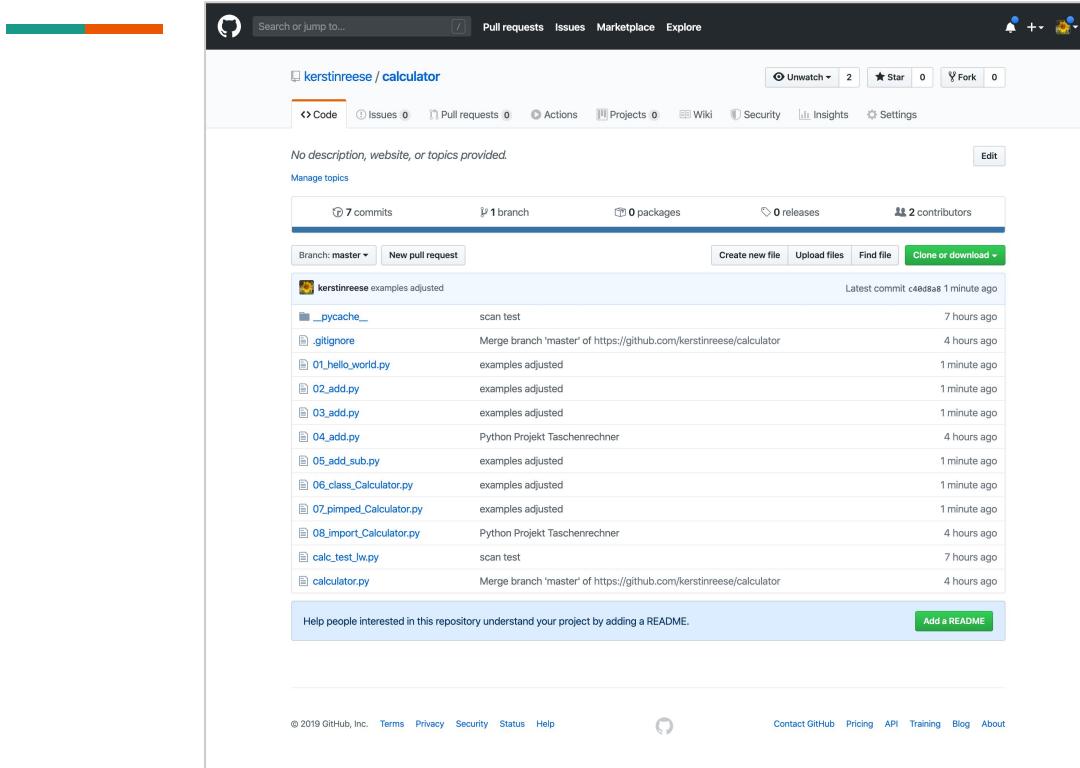
1 from calculator import Calculator
2
3 calculator = Calculator("Toller Taschenrechner")
4 calculator.add(5,3)
5 calculator.sub(5,3)
  
```

**Right Window (hello.py):**

```

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))
  
```

# 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 latest commit was made by `kerstinreese` on master, c46d8a8, 1 minute ago. The commits are:

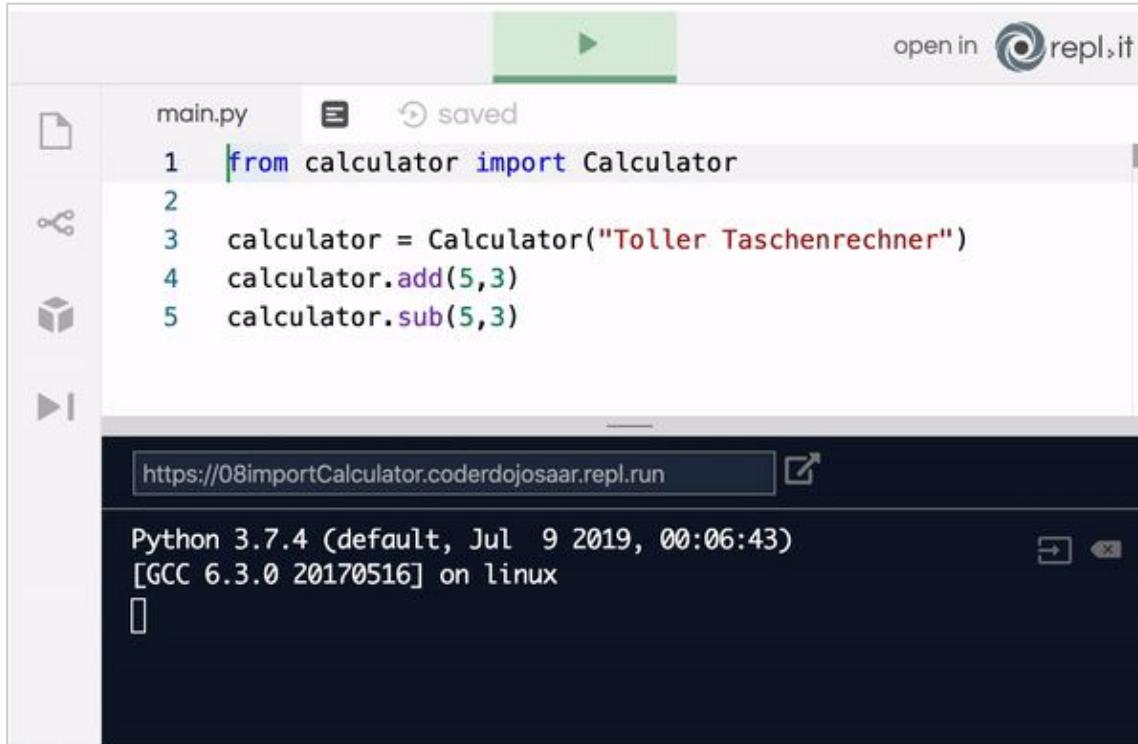
- `examples adjusted` by `kerstinreese`, latest commit c46d8a8, 1 minute ago
- `scan test` by `_pycache_`, 7 hours ago
- `Merge branch 'master' of https://github.com/kerstinreese/calculator` by `gitignore`, 4 hours ago
- `examples adjusted` by `01_hello_world.py`, 1 minute ago
- `examples adjusted` by `02_add.py`, 1 minute ago
- `examples adjusted` by `03_add.py`, 1 minute ago
- `Python Projekt Taschenrechner` by `04_add.py`, 4 hours ago
- `examples adjusted` by `05_add_sub.py`, 1 minute ago
- `examples adjusted` by `06_class_Calculator.py`, 1 minute ago
- `examples adjusted` by `07_pimped_Calculator.py`, 1 minute ago
- `Python Projekt Taschenrechner` by `08_import_Calculator.py`, 4 hours ago
- `scan test` by `calc_test_lw.py`, 7 hours ago
- `Merge branch 'master' of https://github.com/kerstinreese/calculator` by `calculator.py`, 4 hours ago

At the bottom, there is a button to "Add a README".

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

-  [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](#)

# Alle Programme auf repl.it



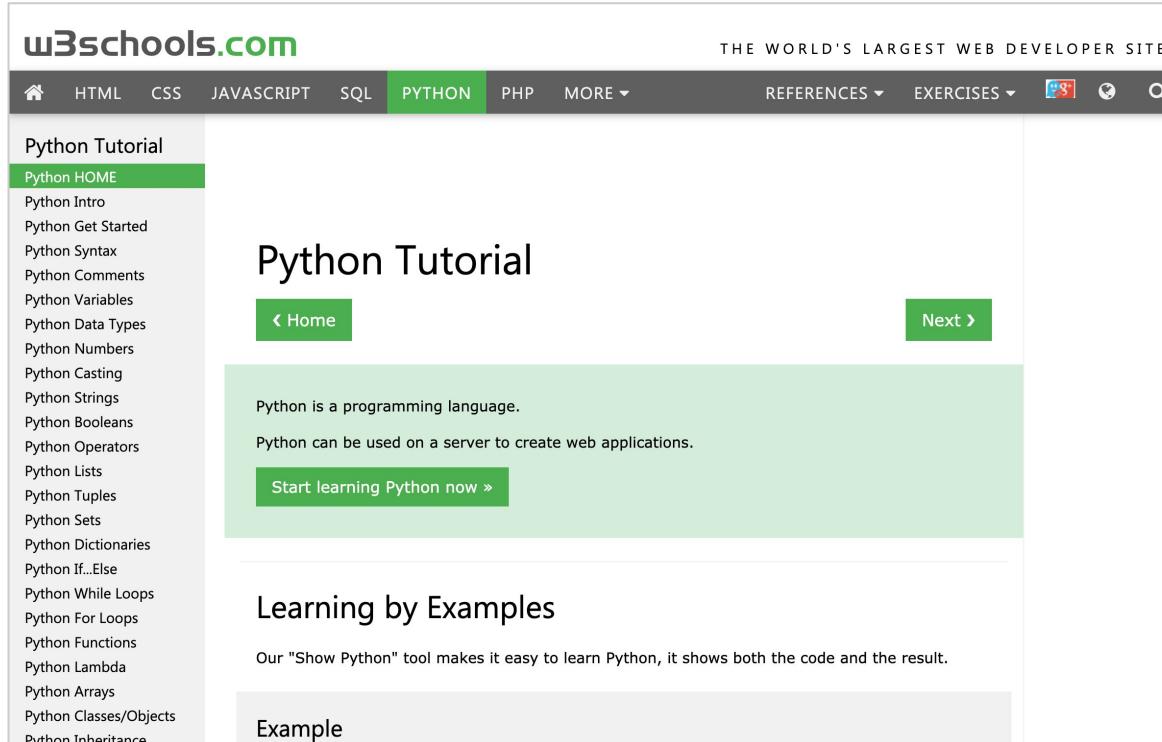
The screenshot shows a repl.it interface. At the top, there's a green play button and a link to "open in repl.it". On the left, there's a sidebar with icons for file operations and a terminal window. The main area displays a Python script named "main.py" with the following code:

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

Below the code, a terminal window shows the output of the script. The URL <https://08importCalculator.coderdojojosaar.repl.run> is shown above the terminal. The terminal output is:

```
Python 3.7.4 (default, Jul 9 2019, 00:06:43)
[GCC 6.3.0 20170516] on linux
[]
```

# Python Tutorial



The screenshot shows the w3schools.com Python tutorial page. The header includes the w3schools.com logo, the text "THE WORLD'S LARGEST WEB DEVELOPER SITE", and navigation links for Home, HTML, CSS, JavaScript, SQL, Python (which is highlighted in green), PHP, More, References, Exercises, and a search icon.

The main content area has a title "Python Tutorial" and two green buttons: "< Home" and "Next >". Below the buttons, there is a green box containing the text: "Python is a programming language. Python can be used on a server to create web applications." A green button labeled "Start learning Python now »" is also present in this box.

Below this section, there is a heading "Learning by Examples" followed by the text: "Our 'Show Python' tool makes it easy to learn Python, it shows both the code and the result." A grey box labeled "Example" is partially visible at the bottom.

The URL [www.w3schools.com/python/](http://www.w3schools.com/python/) is displayed at the bottom of the screenshot.

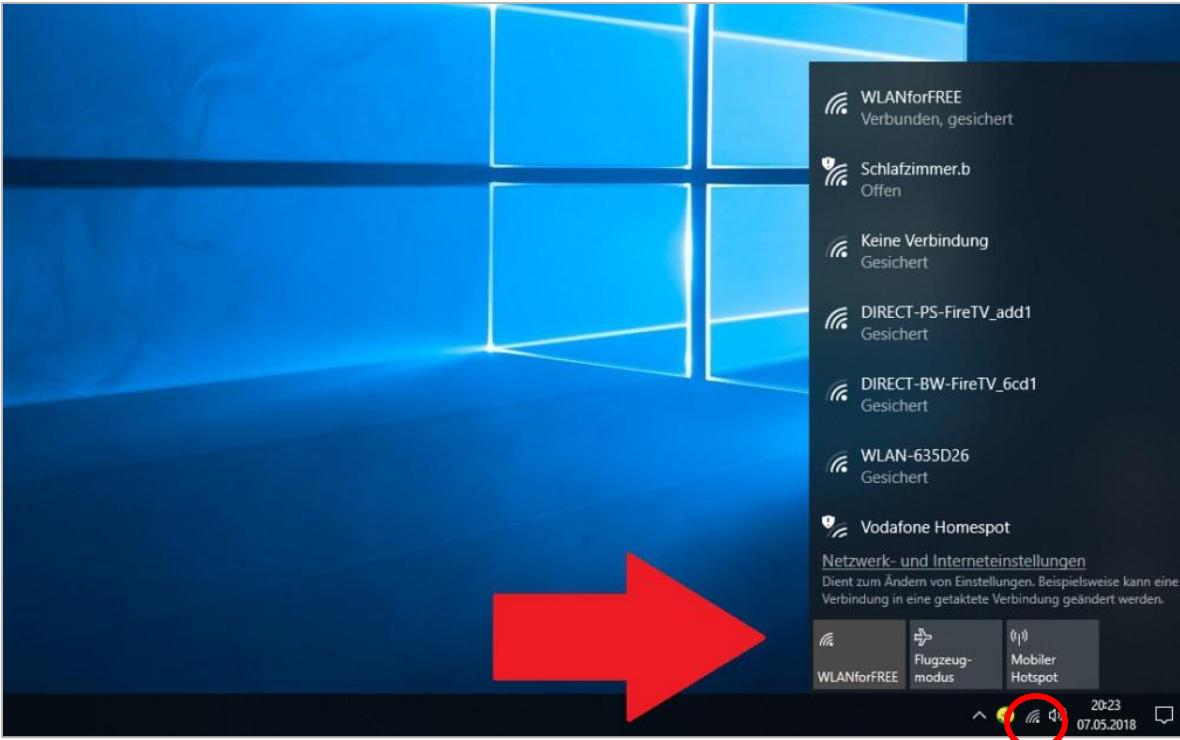


InfoLabSaar

---

# 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")



# 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](#)

[JAVASCRIPT](#) [SQL](#) **PYTHON** [PHP](#) [BOOTSTRAP](#) [HOW TO](#) [MORE ▾](#) [REFERENCES ▾](#) [EXERCISES](#)

## 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

```

1  from tello import Tello ← Import der Definition der Klasse Tello aus der Datei tello.py
2
3  tello = Tello() ← Instanziierung eines Objekts der Klasse Tello
4
5
6  def fly8(n): ← Definition der Funktion fly8 mit Parameter n
7      for i in range(n + 1): ← for-Schleife, die den Parameter n benutzt
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") ← Befehle innerhalb der for-Schleife
17
18
19     tello.send_command("takeoff") ← Aufruf der Funktion send_command des Objekts tello
20
21     fly8(2) ← Aufruf der Funktion fly8 mit Parameter 2
22
23     tello.send_command("land") ← Aufruf der Funktion send_command des Objekts tello
24

```

# 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()
```

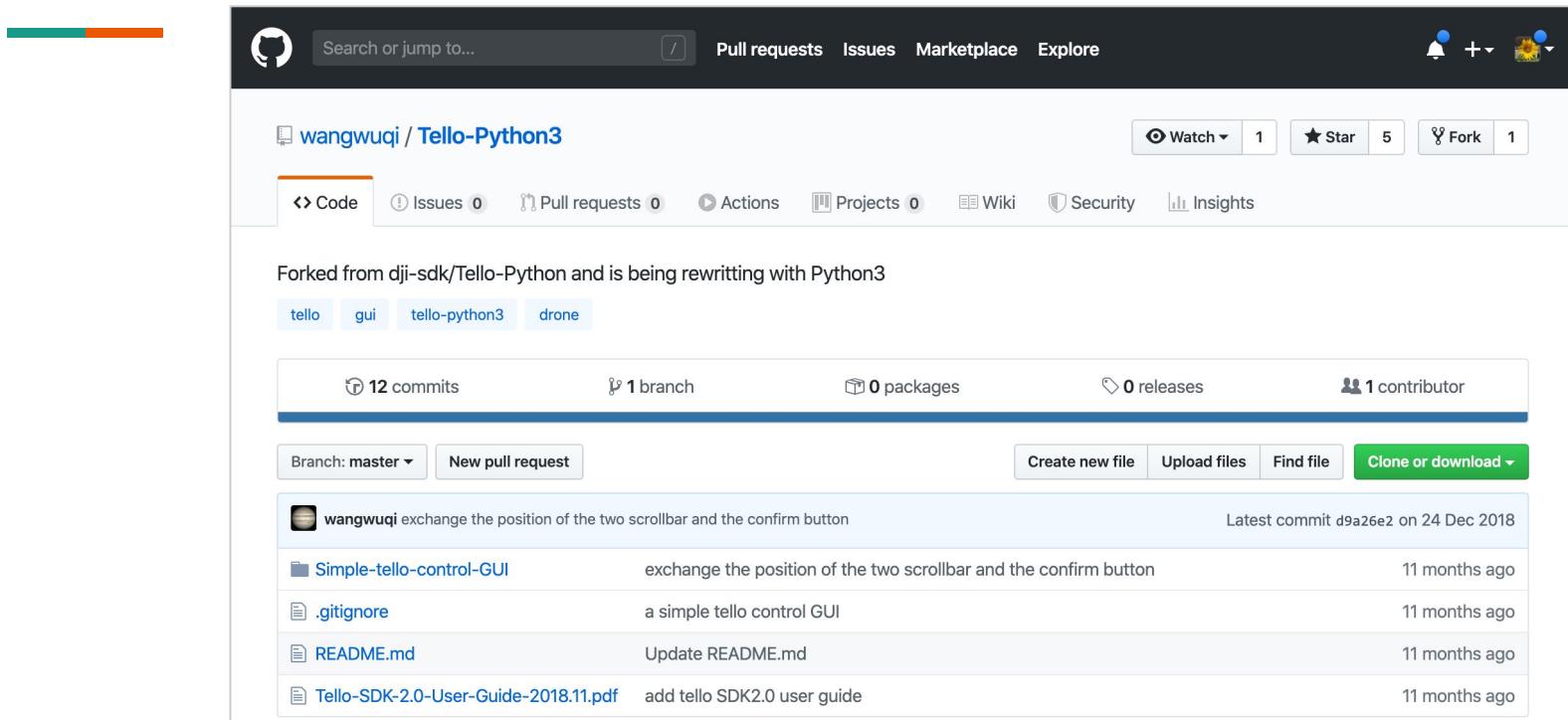
# Idee 5: Befehle aus Datei lesen

```
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 a GitHub repository page for `wangwuqi / Tello-Python3`. The repository is described as forked from `dji-sdk/Tello-Python` and being rewritten with Python3. It features tabs for Code, Issues (0), Pull requests (0), Actions, Projects (0), Wiki, Security, and Insights. Key repository metrics shown include 12 commits, 1 branch, 0 packages, 0 releases, and 1 contributor. A dropdown menu shows the current branch is `master`, and there is a button to `New pull request`. A green button at the bottom right allows cloning or downloading the repository. The commit history lists several changes, including a commit by `wangwuqi` to exchange scrollbar and confirm button positions, and other commits for a control GUI and user guides.

Commit	Description	Date
wangwuqi	exchange the position of the two scrollbar and the confirm button	Latest commit d9a26e2 on 24 Dec 2018
Simple-tello-control-GUI	exchange the position of the two scrollbar and the confirm button	11 months ago
.gitignore	a simple tello control GUI	11 months ago
README.md	Update README.md	11 months ago
Tello-SDK-2.0-User-Guide-2018.11.pdf	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.



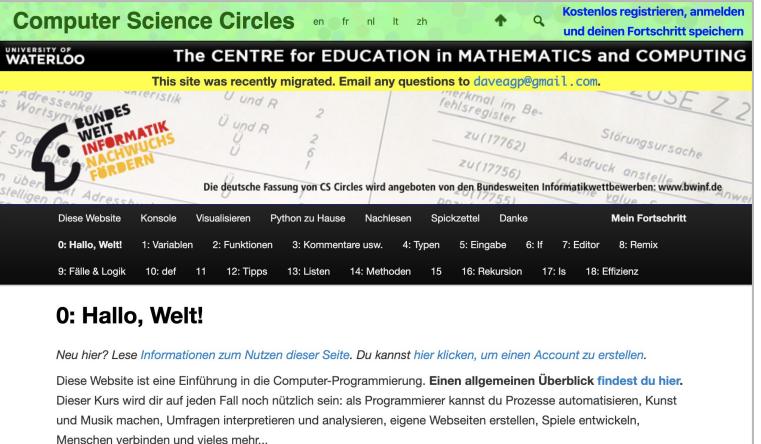
Krypto im Advent  
Die Welt der Verschlüsselung entdecken!  
Adventsrätsel für angehende Geheimagentinnen und Geheimagenten

[Jetzt kostenlos registrieren](#)

Die Geschichte der Kryptographie

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

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



Computer Science Circles

The CENTRE for EDUCATION in MATHEMATICS and COMPUTING

This site was recently migrated. Email any questions to [daveagp@gmail.com](mailto:daveagp@gmail.com).

Diese Website Konsole Visualisieren Python zu Hause Nachlesen Spickzettel Danke Mein Fortschritt

0: Hallo, Welt! 1: Variablen 2: Funktionen 3: Kommentare usw. 4: Typen 5: Eingabe 6: If 7: Editor 8: Remix

9: Fälle & Logik 10: def 11 12: Tipps 13: Listen 14: Methoden 15 16: Rekursion 17: Is 18: Effizienz

**0: Hallo, Welt!**

Neu hier? Lese [Informationen zum Nutzen dieser Seite](#). Du kannst [hier klicken](#), um einen Account zu erstellen.

Diese Website ist eine Einführung in die Computer-Programmierung. Einen allgemeinen Überblick [findest du hier](#). Dieser Kurs wird dir auf jeden Fall noch nützlich sein: als Programmierer kannst du Prozesse automatisieren, Kunst und Musik machen, Umfragen interpretieren und analysieren, eigene Webseiten erstellen, Spiele entwickeln, Menschen verbinden und vieles mehr...

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