

# Python Kurs 2019/2020

## 7: Entwicklungstooling

### Versionskontrolle 1, Testing 1

Bernhard Mallinger

`b.mallinger [at] gmx.at`

<https://totycro.github.io/python-kurs>

# Versionskontrolle: git

# Versionskontrolle

- Häufig:
  - Kleine Änderung  
⇒ Programm funktioniert nicht mehr
  - Änderung rückgängig gemacht  
⇒ Programm funktioniert auch nicht?!?
- Änderungen sollen getrackt werden
- Universelle Versionskontrollsysteme
  - Leider nur in Softwareindustrie weit verbreitet (Masterarbeit?)
- Heutzutage Standardsystem: git

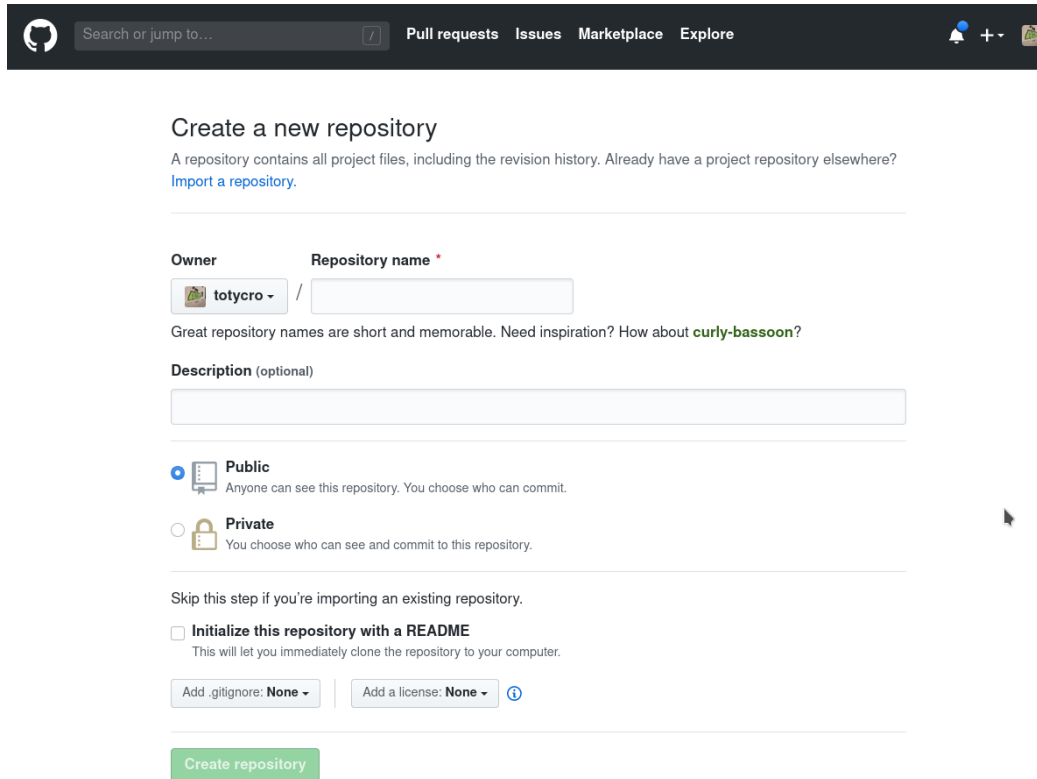
# git Workflow

- Ein Hauptverzeichnis verwaltet alle Dateien des Projekts (*Repository*)
- Dateien wie gewohnt bearbeiten
- Bei Zwischenstufen aktuelle Änderungen einchecken (*commit*: Menge von Änderungen)
  - Gerne oft committen, z.b. wenn Tests durchgehen, wenn ein (Teil-)Feature funktioniert
  - Alle Änderungen immer nachvollziehbar und umkehrbar!
- Hervorragende Unterstützung für gleichzeitiges Arbeiten verschiedener Leute an einem Projekt (branches)
  - Mergen von Änderungen, auch an gleichen Dateien
  - inheränte Komplexität trotzdem hoch
  - out of scope hier

# git Repositories

- git benötigt keinen Server, lokales Arbeiten möglich
- Zum Austausch Server notwendig
  - Gibt gratis Hoster
  - de-facto Standard: github.com (Microsoft)
- Workflow mit Server:
  - Repository z.B. auf github.com anlegen
  - Repository *clonen*
  - Commits zu Server *pushen*
  - Andere können dort *pullen*
  - Konflikte ggf. mit git lösen (out of scope hier)


# Repository anlegen auf GitHub: public/private



The screenshot shows the GitHub interface for creating a new repository. At the top is a dark navigation bar with the GitHub logo, a search bar, and links for Pull requests, Issues, Marketplace, and Explore. Below this, the main heading is 'Create a new repository'. A subtext explains that a repository contains all project files and revision history, with a link to 'Import a repository'. The form fields include: 'Owner' (set to 'totycro'), 'Repository name' (empty), 'Description (optional)' (empty), and visibility options for 'Public' (selected) and 'Private'. Below these are checkboxes for 'Initialize this repository with a README' and 'Add .gitignore' and 'Add a license', both currently set to 'None'. A green 'Create repository' button is at the bottom.


Create a new repository


A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner:  totycro /

Great repository names are short and memorable. Need inspiration? How about [curly-bassoon](#)?

Description (optional):

☒  **Public**  
Anyone can see this repository. You choose who can commit.

☐  **Private**  
You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

☐ **Initialize this repository with a README**  
This will let you immediately clone the repository to your computer.

Add .gitignore: **None** | Add a license: **None** ⓘ

[Create repository](#)

# Demo

- Graphische Oberfläche für git:
  - gitk: Log ansehen
  - git-gui: Commits erstellen(Gibt Alternativen, alle mit Vor- und Nachteilen)
- Kommandezeptool `git` sehr gebräuchlich
- Unter Linux normalerweise mitgeliefert, für Win & Mac gibt es Downloads: <https://git-scm.com/downloads>

## Schritte

- Repo lokal anlegen
- Dateien erzeugen/bearbeiten
- Staging: Dateien bei git vormerken
- Commit
- Log + Diff
- Diff über Versionen
- Revert

# git Hinweise

- Beispiel für Verwendung:  
<https://github.com/adob360/production-optimization/commits/master>
- Geschichte kann auch verändert werden
  - Das verkompliziert das Pushen auf den Server  
⇒ vermeiden (vorerst)

## AUFGABE

Verwende git bei *allen* künftigen Programmierprojekten.

Erweiterung: Verwende git bei deinen Uniprojekten, vor allem länger andauernde.



# git: Weiterführende Anleitungen

<https://towardsdatascience.com/getting-started-with-git-and-github-6fcd0f2d4ac6>

<https://www.freecodecamp.org/news/learn-the-basics-of-git-in-under-10-minutes-da548267cc91/>

<https://product.hubspot.com/blog/git-and-github-tutorial-for-beginners>

# Automatisches Testen

# Automatisches Testen

- Während dem Entwickeln test man Code immer wieder  
⇒ Warum nicht automatisch ausführen lassen?
- Zusätzlich zum Code schreibt man auch Testcases, die den eigentlichen Code ausführen
- Bei Weiterentwicklungen wird alles nochmal getestet, damit nichts kaputt wird
- Nachteil: Bei Verhaltensänderungen muss der Test auch geändert werden

# Automatisches Testen

- Ohne Tests:
  - Code "stirbt", man traut sich nicht, ihn weiter anzupassen
- Mit Tests:
  - Änderungen auch nach Wochen/Monaten ohne Angst möglich

# Test-driven development (TDD)

- Noch besser: Tests schreiben, bevor man den Code schreibt
- Hilft im Designprozess: Nachdenken über Verhalten
- Ablenkung und fehlende Konzentration sind weniger problematisch
  - Man weiß genau, was schon funktioniert und woran man jetzt arbeiten muss
- Beim Codeschreiben hat man instant feedback
- Theoretisches Modell: Red → Green → Refactor

# pytest

- Ein verbreitetes und einfach Testtool
- Tests sind Funktionen, deren Name mit `test_` beginnt
- Überprüfungen mit `assert()`

```
def invert_dictionary(l):  
    # dummy implementation to be able to write tests  
    return d  
  
def test_invert_dictionary_handles_empty_dict():  
    assert invert_dictionary({}) == {}  
  
def test_invert_dictionary_inverts_single_elements():  
    assert invert_dictionary({1: "a"}) == {"a": [1]}  
  
def test_invert_dictionary_merges_duplicate_values():  
    assert invert_dictionary({1: "a", 2: "a"}) == {"a": [1, 2]}
```

# pytest

- Ausführen in Thonny über Tools → Open system shell
- Dort eingeben: `pytest dateiname.py` und Enter drücken

```
[totycro@unten5 playground]$ pytest invert_dictionary.py
===== test session starts =====
collected 3 items

invert_dictionary.py .FF                                     [100%]

===== FAILURES =====
_____ test_invert_dictionary_inverts_single_elements _____

    def test_invert_dictionary_inverts_single_elements():
>     assert invert_dictionary({1: "a"}) == {"a": [1]}
E     AssertionError: assert {1: 'a'} == {'a': [1]}
E         Left contains 1 more item:
E         {1: 'a'}
E         Right contains 1 more item:
E         {'a': [1]}
E         Use -v to get the full diff
[...]
```

# pytest

Test möchte, dass bei Input `{1: 'a'}` das Ergebnis `{'a': [1]}` herauskommt.

```
def invert_dictionary(d):  
    inverted_d = {}  
    for k, v in d.items():  
        inverted_d[v] = [k]  
    return inverted_d
```



# pytest

```
===== test session starts =====
collected 3 items

invert_dictionary.py ..F                                     [100%]

===== FAILURES =====
_____ test_invert_dictionary_merges_duplicate_values _____

    def test_invert_dictionary_merges_duplicate_values():
>         assert invert_dictionary({1: "a", 2: "a"}) == {"a": [1, 2]}
E         AssertionError: assert {'a': [2]} == {'a': [1, 2]}
E             Differing items:
E             {'a': [2]} != {'a': [1, 2]}
E             Use -v to get the full diff

invert_dictionary.py:11: AssertionError
===== 1 failed, 2 passed in 0.03s =====
```

# pytest

Jetzt brauchen wir nur noch das Mergen bei `{1: "a", 2: "a"}`

```
inverted_d = {}  
for key, value in d.items():  
    if value not in d:  
        d[value] = []  
    d[value].append(key)  
return inverted_d
```

```
inverted_d = {}  
for key, value in d.items():  
    inverted_d[value] = d.get(value, []).append(key)  
return inverted_d
```

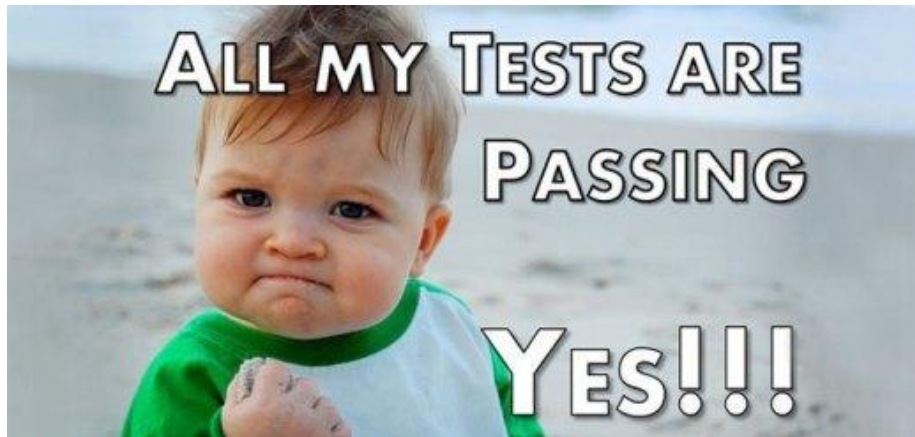
```
inverted_d = {}  
for key, value in d.items():  
    inverted_d.setdefault(value, []).append(key)  
return inverted_d
```

# pytest

```
[totycro@unten5 playground]$ pytest invert_dictionary.py
===== test session starts =====
collected 3 items

invert_dictionary.py ... [100%]

===== 3 passed in 0.02s =====
```



# Testing Hinweise

- Jeder Test sollte möglichst kurz sein und möglichst nur einen Aspekt abdecken
- Testdaten können je nach Anwendung schwer zu generieren sein  
⇒ Datengenerierung von Tests trennen
- Wartbarkeit auch von Tests ist essentiell