

# Einstieg in Git: Tipps für die Verwaltung von Programm- Quellcode in Forschungsprojekten

Christian Peters  
Universitäts- und Landesbibliothek Sachsen-Anhalt  
[christian.peters@bibliothek.uni-halle.de](mailto:christian.peters@bibliothek.uni-halle.de)



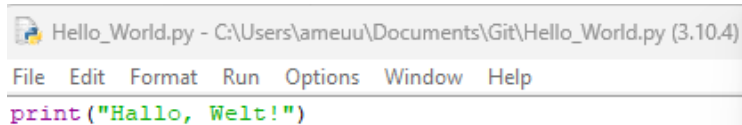
UNIVERSITÄTS- UND  
LANDESBIBLIOTHEK  
SACHSEN - ANHALT

# Inhalt

- Wofür Git?
- Was ist Git?
- Benutzung von Git
- Weiteres über Git

# Wofür Git?

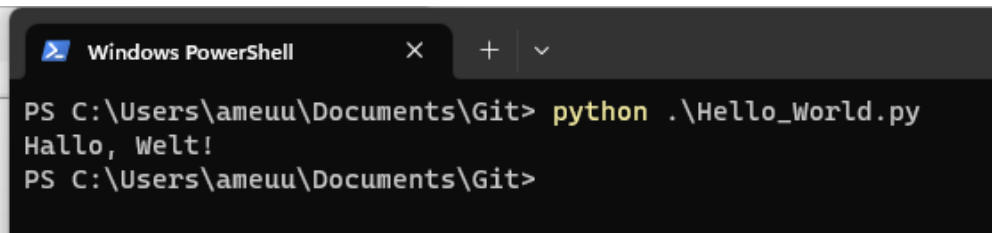
- Softwareentwicklung kann sehr komplex werden
- Fehler werden auftreten



Hello\_World.py - C:\Users\ameuu\Documents\Git\Hello\_World.py (3.10.4)

File Edit Format Run Options Window Help

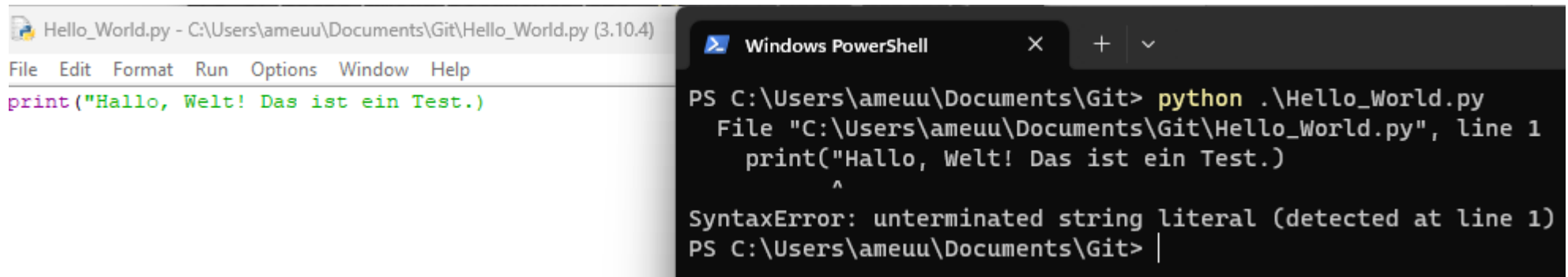
```
print("Hallo, Welt!")
```



Windows PowerShell


```
PS C:\Users\ameuu\Documents\Git> python .\Hello_World.py  
Hallo, Welt!  
PS C:\Users\ameuu\Documents\Git>
```


# Wofür Git? – Sehr simples Beispiel (1)




The image shows a Python IDE window titled 'Hello\_World.py - C:\Users\ameuu\Documents\Git\Hello\_World.py (3.10.4)'. The code in the editor is `print("Hallo, Welt! Das ist ein Test.)`. To the right, a Windows PowerShell terminal window shows the command `python .\Hello_World.py` being executed. The output displays the file path and the code line, followed by a `SyntaxError: unterminated string literal (detected at line 1)` message, with a caret pointing to the closing parenthesis in the print statement.


- Nach Änderungen am Code tritt ein Fehler auf, der unverständlich ist und schwer zu beheben
- Man weiß nicht mehr, wo man überall Code geändert hat und welche Änderung zu dem Fehler führen könnte
- Lösung(?): Kopien von funktionierenden Programmen


 Hello\_World - Kopie.py


 Hello\_World.py


## Wofür Git? – Sehr simples Beispiel (2)


 Hello\_World.py


 Hello\_World - Kopie (4).py


 Hello\_World - Kopie (7).py

 Hello\_World - Kopie (10).py

 Hello\_World - Kopie (13).py


 Hello\_World - Kopie (16).py


 Hello\_World - Kopie (19).py

 Hello\_World - Kopie (22).py


 Hello\_World - Kopie (25).py


 Hello\_World - Kopie (28).py


 Hello\_World - Kopie (31).py


 Hello\_World - Kopie (2).py


 Hello\_World - Kopie (5).py


 Hello\_World - Kopie (8).py


 Hello\_World - Kopie (11).py


 Hello\_World - Kopie (14).py


 Hello\_World - Kopie (17).py

 Hello\_World - Kopie (20).py

 Hello\_World - Kopie (23).py

 Hello\_World - Kopie (26).py


 Hello\_World - Kopie (29).py


 Hello\_World - Kopie.py


 Hello\_World - Kopie (3).py


 Hello\_World - Kopie (6).py

 Hello\_World - Kopie (9).py

 Hello\_World - Kopie (12).py

 Hello\_World - Kopie (15).py

 Hello\_World - Kopie (18).py

 Hello\_World - Kopie (21).py


 Hello\_World - Kopie (24).py


 Hello\_World - Kopie (27).py


 Hello\_World - Kopie (30).py


- Lokale Kopien ohne weitere Informationen sind auf Dauer nicht hilfreich
- Nachvollziehbarkeit und Erreichbarkeit für andere ist nicht vorhanden


# Wofür Git? – Die Lösung: Bessere Dokumentation(?)


 Ausgabe\_Datum.py


 Hello\_World.py

 2024-07-11\_Hello\_World - Hallo, Welt ausgeben.py

 2024-07-12\_Hello\_World - Erweiterung Das ist ein Test.py

 2024-07-20\_Hello\_World - Entfernung Das ist ein Test.py

 2024-07-22\_Hello\_World - Erweiterung Ausgabe aktuelles Datum.py

 2024-07-23\_Hello\_World - Auslagerung Ausgabe Datum in eigene Datei.py

- Eindeutige Bezeichnung der Kopien hilfreich
- System scheitert bei mehreren Dateien, die Einfluss aufeinander haben und gleichzeitig geändert werden
- Erreichbarkeit für andere weiterhin nicht gegeben
- Lösung dieser Probleme: Git

# Was ist Git?

- Software zur Verwaltung von Software
- Einfache Nachvollziehbarkeit von Änderungen
- Einfaches kollaboratives Arbeiten an Code
- Erhöhte Komplexität
- Speicherung auf Servern – Backups sind integriert
- Bekannte Anbieter: Github (kommerziell), Gitlab (viele verschiedene Angebote)
- Grundlegende Befehle sind überall gleich

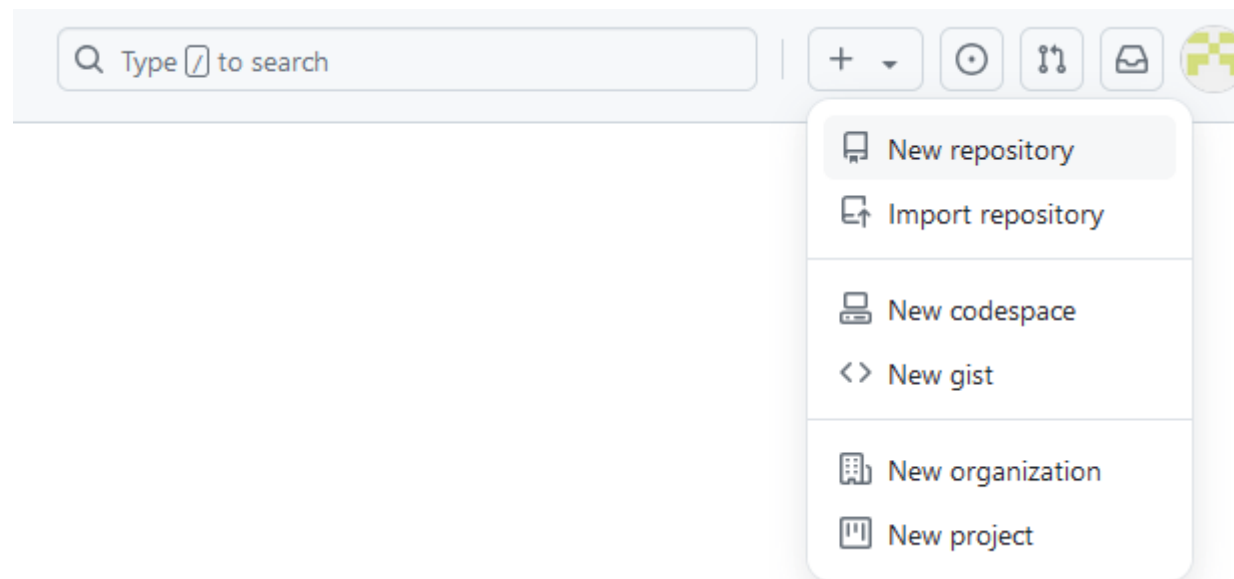
# Was ist Git? – Grundlegendes Prinzip

- Ein Softwareprojekt = Ein „Repository“
- Ein Repository kann beliebig viele Dateien enthalten
  - Da die komplette Änderungsgeschichte gespeichert wird, sollten große Datensätze nicht mit hochgeladen werden, nur der Code
- Das Repository kann alle Änderungen an in ihm enthaltenden Dateien einer Person zuordnen
  - Die Person sucht sich selbst aus, welche Änderungen sie langfristig speichern will
- Alle Änderungen haben eine Änderungsnachricht, die die dort gemachten Änderungen beschreiben soll



# Benutzung von Git – Erzeugen eines neuen Repositories (1)

- Alle Bilder sind von „Github“: <https://www.github.com>
- Registrierung notwendig



# Benutzung von Git – Erzeugen eines neuen Repositories (2)

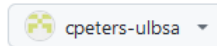
## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?

[Import a repository.](#)

*Required fields are marked with an asterisk (\*).*

Owner \*



Repository name \*

/

✓ hallo\_welt is available.

Great repository names are short and memorable. Need inspiration? How about [bug-free-sniffle](#) ?

Description (optional)



Public

Anyone on the internet can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Initialize this repository with:



Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: None ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license


License: None ▾





A license tells others what they can and can't do with your code. [Learn more about licenses.](#)


You are creating a public repository in your personal account.

Create repository

# Benutzung von Git – Erzeugen eines neuen Repositories (3)

 **hallo\_welt** Public


 Pin  Unwatch **1**  Fork **0**  Star **0**



### Set up GitHub Copilot

Use GitHub's AI pair programmer to autocomplete suggestions as you code.

Get started with GitHub Copilot




### Add collaborators to this repository

Search for people using their GitHub username or email address.

Invite collaborators

## Quick setup — if you've done this kind of thing before

 Set up in Desktop or HTTPS SSH

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

## ...or create a new repository on the command line

```
echo "# hallo_welt" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin git@github.com:cpeters-ulbsa/hallo_welt.git
git push -u origin main
```

## ...or push an existing repository from the command line

```
git remote add origin git@github.com:cpeters-ulbsa/hallo_welt.git
git branch -M main
git push -u origin main
```

# Benutzung von Git – git clone

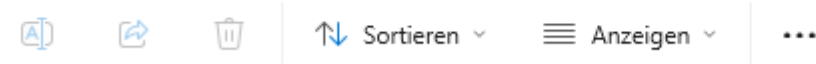
- git muss installiert und vorbereitet sein



```
ameuu@ULB-221219A:/mnt/c/Users/ameuu/Documents$ git clone git@github.com:cpeters-ulbsa/hallo_welt.git
Cloning into 'hallo_welt'...
warning: You appear to have cloned an empty repository.
ameuu@ULB-221219A:/mnt/c/Users/ameuu/Documents$ |
```

> Dokumente > hallo_welt >			
Name	Änderungsdatum	Typ	Größe
.git	11.07.2024 09:40	Dateiordner	

# Benutzung von Git – git status

> Dokumente > hallo\_welt >



Name	Änderungsdatum	Typ	Größe
 .git	11.07.2024 09:40	Dateiordner	
 Hello_World.py	11.07.2024 09:42	Python File	1 KB

```
ameuu@ULB-221219A:/mnt/c/Users/ameuu/Documents$ cd hallo_welt/
ameuu@ULB-221219A:/mnt/c/Users/ameuu/Documents/hallo_welt$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
      Hello_World.py

nothing added to commit but untracked files present (use "git add" to track)
ameuu@ULB-221219A:/mnt/c/Users/ameuu/Documents/hallo_welt$
```

# Benutzung von Git – git add


- „add“ merkt Dateien für Kopieerzeugung vor
- Nur nötig für neue/geänderte Dateien


```
ameuu@ULB-221219A:/mnt/c/Users/ameuu/Documents/hallo_welt$ git add Hello_World.py
ameuu@ULB-221219A:/mnt/c/Users/ameuu/Documents/hallo_welt$ git status
On branch master


No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   Hello_World.py

ameuu@ULB-221219A:/mnt/c/Users/ameuu/Documents/hallo_welt$ |
```

 .git

 Ausgabe\_Datum.py

 Hello\_World.py

```
ameuu@ULB-221219A:/mnt/c/Users/ameuu/Documents/hallo_welt$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   Hello_World.py

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    Ausgabe_Datum.py

ameuu@ULB-221219A:/mnt/c/Users/ameuu/Documents/hallo_welt$ |
```

# Benutzung von Git – git commit

- „commit“ erzeugt lokale Kopie von allen vorgemerkten und unveränderten Dateien

```
ameuu@ULB-221219A:/mnt/c/Users/ameuu/Documents/hallo_welt$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   Hello_World.py

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    Ausgabe_Datum.py

ameuu@ULB-221219A:/mnt/c/Users/ameuu/Documents/hallo_welt$ git commit -m "Hallo, Welt-Programm entwickelt"
[master (root-commit) 81dd2b0] Hallo, Welt-Programm entwickelt
 1 file changed, 1 insertion(+)
 create mode 100644 Hello_World.py
ameuu@ULB-221219A:/mnt/c/Users/ameuu/Documents/hallo_welt$ git status
On branch master
Your branch is based on 'origin/master', but the upstream is gone.
  (use "git branch --unset-upstream" to fixup)



Untracked files:
  (use "git add <file>..." to include in what will be committed)
    Ausgabe_Datum.py


nothing added to commit but untracked files present (use "git add" to track)
```

# Benutzung von Git – git push (1)

- „push“: Alle lokalen „Commits“ an den Server schicken

```
ameuu@ULB-221219A:/mnt/c/Users/ameuu/Documents/hallo_welt$ git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 275 bytes | 5.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:cpeters-ulbsa/hallo_welt.git
 * [new branch]      master -> master
```

 **cpeters-ulbsa** Hallo, Welt-Programm entwickelt 81dd2b0 · 3 minutes ago  1 Commit

 Hello\_World.py Hallo, Welt-Programm entwickelt 3 minutes ago

[hallo\\_welt](#) / [Hello\\_World.py](#) 



**cpeters-ulbsa** Hallo, Welt-Programm entwickelt

81dd2b0 · 4 minutes ago  History

Code

Blame

1 lines (1 loc) · 23 Bytes

Raw



1 `print("Hallo, Welt!")`



## Benutzung von Git – git push (2)




```
ameuu@ULB-221219A:/mnt/c/Users/ameuu/Documents/hallo_welt$ git status
On branch master
Your branch is up to date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
      Ausgabe_Datum.py

nothing added to commit but untracked files present (use "git add" to track)
ameuu@ULB-221219A:/mnt/c/Users/ameuu/Documents/hallo_welt$ git add Ausgabe_Datum.py
```

```
ameuu@ULB-221219A:/mnt/c/Users/ameuu/Documents/hallo_welt$ git commit -m "Datumsausgabe in eigener Datei"
[master dec14e7] Datumsausgabe in eigener Datei
 1 file changed, 1 insertion(+)
 create mode 100644 Ausgabe_Datum.py
ameuu@ULB-221219A:/mnt/c/Users/ameuu/Documents/hallo_welt$ git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 287 bytes | 5.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:cpeters-ulbsa/hallo_welt.git
 81dd2b0..dec14e7  master -> master
```

# Benutzung von Git – Commit-Historie (1)

 <b>cpeters-ulbsa</b>	Datumsausgabe in eigener Datei	dec14e7 · now	🕒 2 Commits
 Ausgabe_Datum.py	Datumsausgabe in eigener Datei	now	
 Hello_World.py	Hallo, Welt-Programm entwickelt	7 minutes ago	


## Commits


 master ▾

 All users ▾


 All time ▾

 Commits on Jul 11, 2024



**Datumsausgabe in eigener Datei**  
 cpeters-ulbsa committed now

dec14e7  

**Hallo, Welt-Programm entwickelt**  
 cpeters-ulbsa committed 7 minutes ago

81dd2b0  

# Benutzung von Git – git rm

Name	Änderungsdatum	Typ	Größe
 .git	11.07.2024 10:02	Dateiordner	
 Hello_World.py	11.07.2024 10:01	Python File	1 KB

```
ameuu@ULB-221219A:/mnt/c/Users/ameuu/Documents/hallo_welt$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    Ausgabe_Datum.py
        modified:   Hello_World.py

no changes added to commit (use "git add" and/or "git commit -a")
ameuu@ULB-221219A:/mnt/c/Users/ameuu/Documents/hallo_welt$ git add *
ameuu@ULB-221219A:/mnt/c/Users/ameuu/Documents/hallo_welt$ git rm Ausgabe_Datum.py
rm 'Ausgabe_Datum.py'
ameuu@ULB-221219A:/mnt/c/Users/ameuu/Documents/hallo_welt$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    Ausgabe_Datum.py
        modified:   Hello_World.py

ameuu@ULB-221219A:/mnt/c/Users/ameuu/Documents/hallo_welt$ git commit -m "Datumsausgabe in einer Datei"
[master 90d13ef] Datumsausgabe in einer Datei
 2 files changed, 2 insertions(+), 1 deletion(-)
 delete mode 100644 Ausgabe_Datum.py
ameuu@ULB-221219A:/mnt/c/Users/ameuu/Documents/hallo_welt$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 337 bytes | 5.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:cpeters-ulbsa/hallo_welt.git
   dec14e7..90d13ef  master -> master
ameuu@ULB-221219A:/mnt/c/Users/ameuu/Documents/hallo_welt$
```

# Benutzung von Git – Commit-Historie (2)

master

Commits on Jul 11, 2024

Datumsausgabe in einer Datei cpeters-ulbsa committed 7 minutes ago	90d13ef		
Datumsausgabe in eigener Datei cpeters-ulbsa committed 11 minutes ago	dec14e7		
Hallo, Welt-Programm entwickelt cpeters-ulbsa committed 17 minutes ago	81dd2b0		

## Commit

Datumsausgabe in einer Datei [Browse files](#)

master

cpeters-ulbsa committed 8 minutes ago 1 parent dec14e7 commit 90d13ef

Showing 2 changed files with 2 additions and 1 deletion. [Whitespace](#) [Ignore whitespace](#) [Split](#) [Unified](#)

Filter changed files

- Ausgabe\_Datum.py
- Hello\_World.py

1 Ausgabe\_Datum.py

[Load diff](#)

This file was deleted.

2 Hello\_World.py

```
@@ -1,3 @@  
1 print("Hallo, Welt!")  
2 + import datetime  
3 + print(str(datetime.date.today()))
```

# Benutzung von Git – git checkout (1)

master ▾

🔍 All users ▾ 📅 All time ▾

🔗 Commits on Jul 11, 2024

Datumsausgabe in einer Datei cpeters-ulbsa committed 7 minutes ago	90d13ef	📄 <>
Datumsausgabe in eigener Datei cpeters-ulbsa committed 11 minutes ago	dec14e7	📄 <>
Hallo, Welt-Programm entwickelt cpeters-ulbsa committed 17 minutes ago	81dd2b0	📄 <>

dec14e7

```
ameuu@ULB-221219A:/mnt/c/Users/ameuu/Documents/hallo_welt$ git checkout dec14e7 .
Updated 2 paths from 8e40beb
ameuu@ULB-221219A:/mnt/c/Users/ameuu/Documents/hallo_welt$
```

+

> Dokumente > hallo\_welt >

Name	Änderungsdatum	Typ	Größe
.git	11.07.2024 10:25	Dateiordner	
Ausgabe_Datum.py	11.07.2024 10:25	Python File	1 KB
Hello_World.py	11.07.2024 10:25	Python File	1 KB

## Benutzung von Git – git checkout (2)













```
ameuu@ULB-221219A:/mnt/c/Users/ameuu/Documents/hallo_welt$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   Ausgabe_Datum.py
    modified:   Hello_World.py

ameuu@ULB-221219A:/mnt/c/Users/ameuu/Documents/hallo_welt$ git add *
ameuu@ULB-221219A:/mnt/c/Users/ameuu/Documents/hallo_welt$ git commit -m "Datumsausgabe in extra Datei (commit dec14e7)"
[master f67967c] Datumsausgabe in extra Datei (commit dec14e7)
 2 files changed, 1 insertion(+), 2 deletions(-)
 create mode 100644 Ausgabe_Datum.py
ameuu@ULB-221219A:/mnt/c/Users/ameuu/Documents/hallo_welt$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 331 bytes | 4.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:cpeters-ulbsa/hallo_welt.git
 90d13ef..f67967c  master -> master
ameuu@ULB-221219A:/mnt/c/Users/ameuu/Documents/hallo_welt$
```


# Benutzung von Git – git checkout (3)


Commits on Jul 11, 2024

Datumsausgabe in extra Datei (commit <code>dec14e7</code> )	<code>f67967c</code>	 
 cpeters-ulbsa committed now		
Datumsausgabe in einer Datei	<code>90d13ef</code>	 
 cpeters-ulbsa committed 25 minutes ago		
Datumsausgabe in eigener Datei	<code>dec14e7</code>	 
 cpeters-ulbsa committed 28 minutes ago		
Hallo, Welt-Programm entwickelt	<code>81dd2b0</code>	 
 cpeters-ulbsa committed 35 minutes ago		


Datumsausgabe in extra Datei (commit `dec14e7`)

[Browse files](#)

 master

 cpeters-ulbsa committed 3 minutes ago

1 parent `90d13ef` commit `f67967c`

 Showing 2 changed files with 1 addition and 2 deletions.

[Whitespace](#)

[Ignore whitespace](#)

[Split](#)

[Unified](#)



 Filter changed files

 Ausgabe\_Datum.py





 Hello\_World.py



> 1  Ausgabe\_Datum.py 

...

2  Hello\_World.py 

...

... @@ -1,3 +1 @@

1 - import datetime

2 print("Hallo, Welt!")

3 - print(str(datetime.date.today()))

1 print("Hallo, Welt!")

# Benutzung von Git – git pull

- Lädt aktuellen Stand des Online-Repository herunter
- Änderungen, die nicht lokal vorhanden sind, müssen eingepflegt werden
  - Das geschieht bei vielen Konflikten automatisch
- In der Regel nur notwendig, wenn mehrere Menschen an einem Repository arbeiten

Code	Blame	2 lines (2 loc) · 51 Bytes
1		<code>print("Hallo, Welt!")</code>
2		<code>print("Das ist ein Test.")</code>

```
ameuu@ULB-221219A:/mnt/c/Users/ameuu/Documents/hallo_welt$ git pull
hint: Pulling without specifying how to reconcile divergent branches is
hint: discouraged. You can squelch this message by running one of the following
hint: commands sometime before your next pull:
hint:
hint:   git config pull.rebase false  # merge (the default strategy)
hint:   git config pull.rebase true   # rebase
hint:   git config pull.ff only       # fast-forward only
hint:
hint: You can replace "git config" with "git config --global" to set a default
hint: preference for all repositories. You can also pass --rebase, --no-rebase,
hint: or --ff-only on the command line to override the configured default per
hint: invocation.
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 1.00 KiB | 8.00 KiB/s, done.
From github.com:cpeters-ulbsa/hallo_welt
  5747c48..dfd7c86  master    -> origin/master
Updating 5747c48..dfd7c86
Fast-forward
 Hello_World.py | 1 +
 1 file changed, 1 insertion(+)
ameuu@ULB-221219A:/mnt/c/Users/ameuu/Documents/hallo_welt$
```



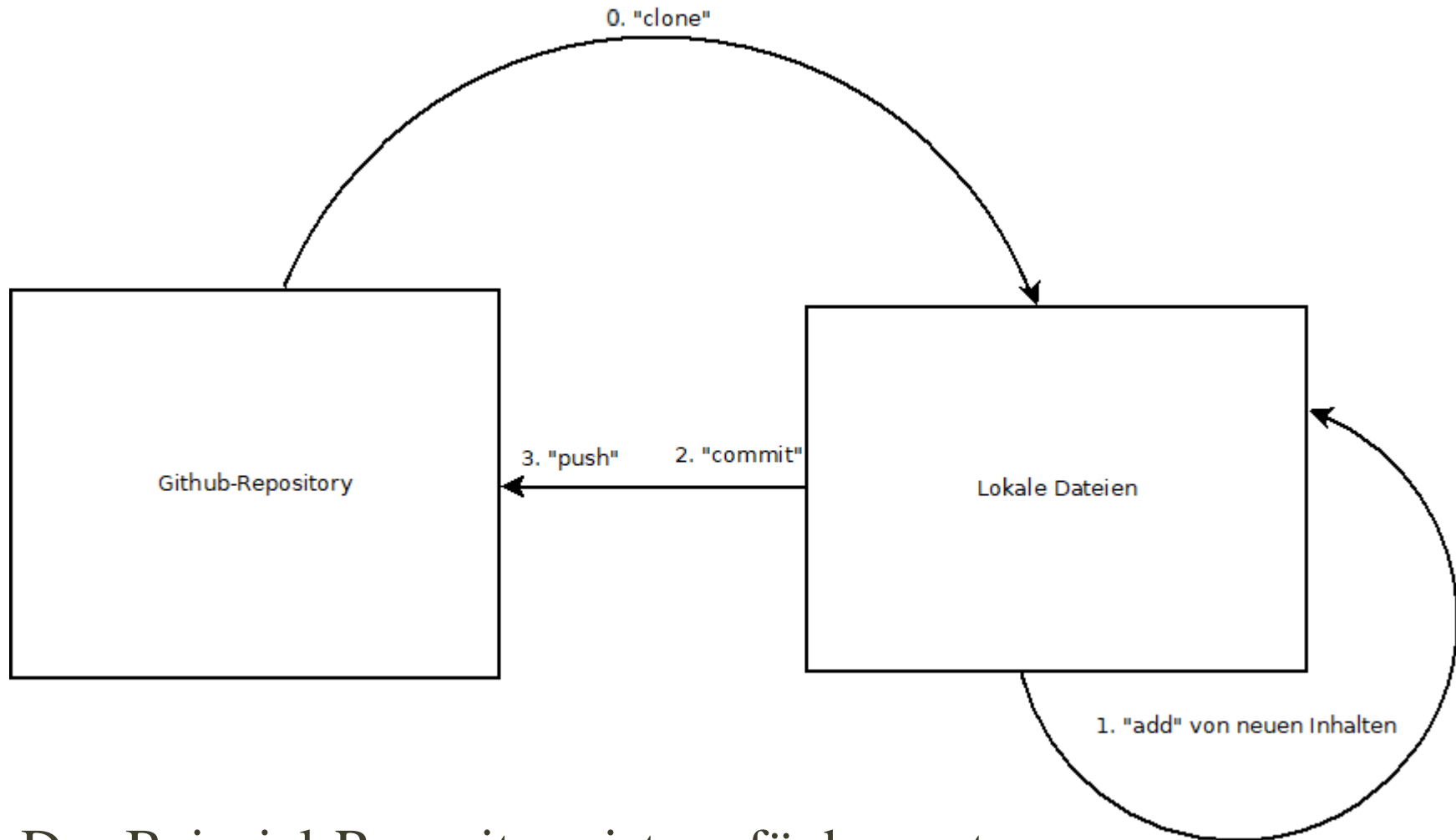
## Weiteres über Git (1)

- „Branches“ (Zweige) sind alternative Versionen des Repositories, die den „Hauptzweig“ nicht beeinflussen
  - Genutzt bei Featureentwicklung
- „Forks“ (Gabeln) sind Kopien eines Repositories, mit dem andere Menschen ihren Code ändern können, ohne, dass ihr Repository davon beeinflusst wird
  - In der Regel genutzt, wenn ein Projekt nicht mehr weiterentwickelt wird oder ein Feature im ursprünglichen Projekt nicht gewünscht ist, man es aber selbst implementieren möchte
- „Pull Requests“ sind Anfragen eines „Forks“ an das Hauptrepository, die Änderungen des „Forks“ auch im Hauptrepository einzubauen

## Weiteres über Git (2)

- Es gibt eine Vielzahl von grafischen Oberflächen für git:  
<https://git-scm.com/downloads/guis>
- Es gibt zahlreiche Anleitungen für git, die auch beim grundlegenden Aufsetzen von git helfen können
  - <http://stefanfrings.de/git/index.html>
  - <https://boolie.org/git-github-anfaenger-tutorial/>
  - <https://git-scm.com/doc>
  - <https://librarycarpentry.org/lc-git/>
- git kann auch vollständig auf dem lokalen Rechner genutzt werden (es entfällt der „push“-Schritt)
  - Ohne weitere Programme wenig übersichtlich

# Überblick



Das Beispiel-Repository ist verfügbar unter:

[https://github.com/cpeters-ulbsa/hallo\\_welt](https://github.com/cpeters-ulbsa/hallo_welt)