git-handson.md - Grip

Git - finally do something

You should have a linux computer - debian or ubuntu based, otherwise at least installation sdteps will differ. For the following: `Typewriter font` is a command to be typed in a terminal, the rest is most likely a comment.

Preparation

`sudo apt install git markdown rsync python-pil meld gitk; sudo apt install python-grip; sudo apt install grip` - some of these might be installed but it doesnt hurt, one of the grip commands will fail.

```
git config --global user.email "b.simpson@springfield.com"
git config --global user.name "Bart Simpson"
```

Should be done once, sets default email and username for commit messages, there is a `git blame` to find you! *Without the `-global` but still within a project sets a possibly different username/mail for this project*

Script

The following is kind of a script for the workshop, no needs to prepare something.

Visit https://github.com/softimpulse/gitintro.git

```
mkdir gitexamples  # sets up a directory for this lecture
cd gitexamples     # P.S. instead of typing gitexamples just hit Alt-.
git clone https://github.com/softimpulse/gitintro.git
```

`ls` - nothing really unexpected, i assume? Maybe you expected a `.git`, well historical bla, you can even clone without this `.git`

`git status` - well we are not in a git repo

`cd gitintro` and again: `git status` we are on master branch, the default and we are in sync with origin (the github repo, where we got this stuff) and branch master (It is not necessary that branchnames on remote and master are equal) take a look at the README with `less README.md` (quit with `q`), do an `ls -la`, all the GIT magic is in `.git` nowhere else!

Go back to gitexamples/

```
mkdir mine
cp -a gitintro/examples/first mine/
cd mine
git status
```

Remember: The files inside a git repo have no git magic, also not subdirs - only .git in projects root!

Create your own git- repo:

```
git init
git status
git add .
git status
git commit 'I will never forget my first commit'
git status
```

Start executable: `./PIL-ex1.py`, 1. curse 2. fix 3. commit

Fix the bug in central repo: `git push` -- Why?

goto ../gitintro, fix the source there, do a `git push` - also just a message -- Why?

Create Patch file with:

`git format-patch -M HEAD~` and do a `cat 0001- ...` , this thing can be send to origins maintainer.

visit https://gitlab.uni-oldenburg.de, login, create a repo and add this new gitlab repo to your newly created.

Got the dir with PIL-ex1.py

```
echo origin.show() >> PIL_ex1.py
```

try it out and commit the as 'first beta', `git tag beta1` sets a flag to this commit - kind of a version number.

Lets make a branch: `git checkout -b processing` creates a branch and jumps onto it, check with `git status` , `ls` .

We can change by hand or apply a patch we got from heaven: `cat 0001-blur-infile.patch` - a collegue

Apply with `git apply 0001-blur-infile.patch` , do a `git status` , `git diff` , ... (Could fail, sorry - in that case do cp ... ill tell you ,)

commit this change

go back to master branch `git checkout master`

add the lines

```
import argparse

parser= argparse.ArgumentParser()
parser.add_argument('img', nargs='?', default="img/vc-xkcd.jpg", help='Image
args= parser.parse_args()

infile= args.img
```

between `from PIL import ...` and `try` by replacing `infile='img/...'

test it, try also `PIL-ex1.py -h`

Get into little trouble: `git checkout processing`

and learn about `git stash` and `git stash apply`

To be continued