

Effective Programming Practices for Economists

# Version Control and collaboration with Git and Github

Pre-commit hooks

Janoš Gabler and Hans-Martin von Gaudecker

# What are pre-commit hooks

- We saw the importance of following style guides
- Pre-commit hooks are tools to help you automate style guides
- Examples of pre-commit hooks are:
  - The black formatter that automatically formats your code
  - The ruff linter that tells you about problems and fixes some of them
  - Line-ending fixers for better compatibility across platforms
- Save a lot of time but have a learning curve

# Activating pre-commit hooks

- Install the `pre_commit` python package
  - `conda install pre_commit`
- Open a terminal in the root of your repository
- Execute `pre-commit install`

Just needs to be done once after cloning the repository

# Configuring pre-commit hooks

```
repos:  
- repo: https://github.com/psf/black  
  rev: 23.9.1  
  hooks:  
    - id: black  
      language_version: python3.11
```

- Pre-commit hooks are configured in `.pre-commit-config.yaml`
- Typically inherited from the project templates or copy it from another project
- Example shows just the black formatter

## A useful git command

- `git commit -am "Your message."` stages and commits all modified files
- Does not work for untracked files
- Important because pre-commit hooks run over staged files!

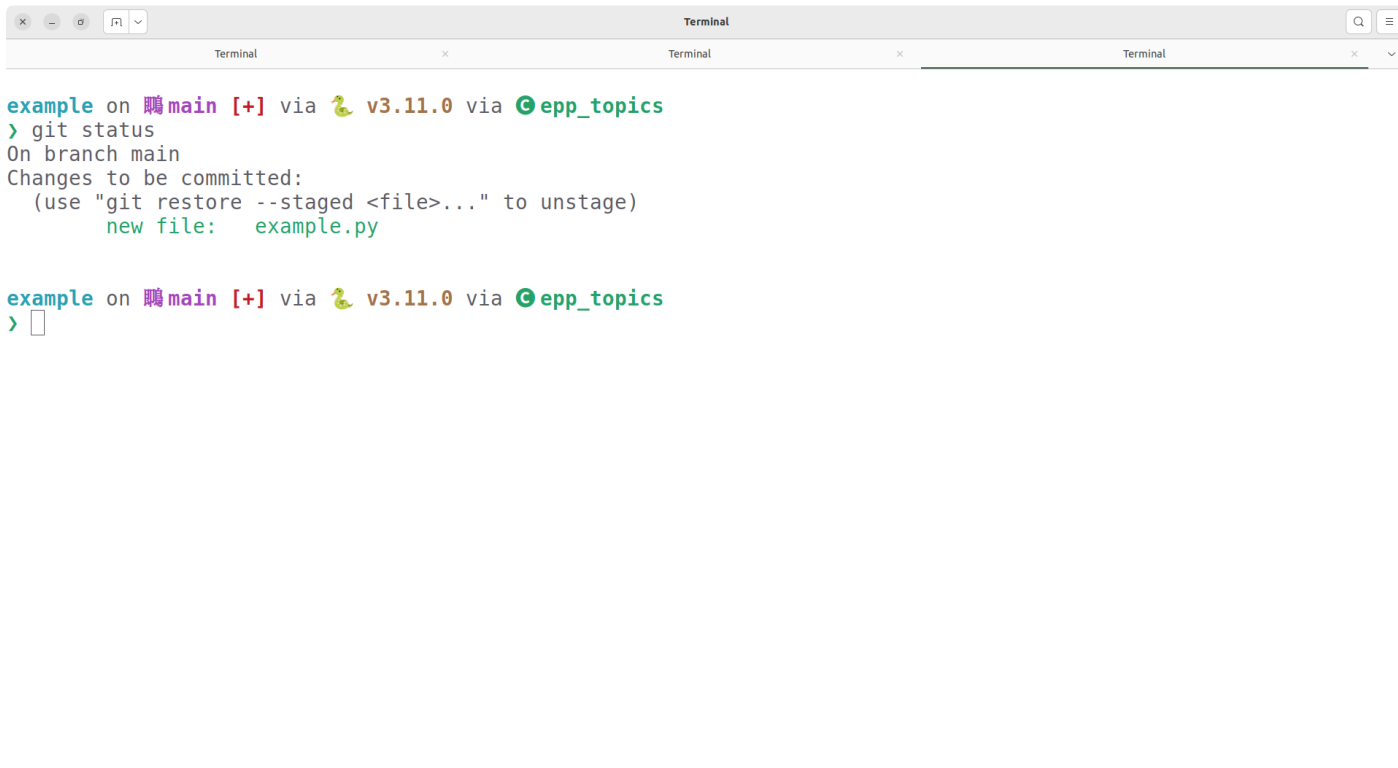
# A badly formatted python file

```
def clean_data(raw):
    df = pd.DataFrame(index=raw.index)
    df["coding_genius"]=clean_agreement_scale(raw['Q001'])
    df['learned_a_lot']=clean_agreement_scale(

        raw['Q002'])
    df['favorite_language'] = clean_favorite_language(raw['Q003'])
    return df

def clean_agreement_scale(sr):
    sr = sr.replace({'-77': pd.NA, '-99':pd.NA})
    categories = ['strongly disagree',
        'disagree',
        'neutral', 'agree',
        'strongly agree'
    ]
    dtype = pd.CategoricalDtype(categories=categories,
        ordered=True)
    return sr.astype(dtype)
```

# Git status

A screenshot of a macOS-style terminal window with three tabs labeled 'Terminal'. The active tab shows the output of a 'git status' command. The prompt is 'example on 鸚 main [+] via 2 v3.11.0 via 鸚 epp\_topics'. The output indicates the user is on the 'main' branch and lists 'example.py' as a new file to be committed. A second prompt is shown below with a cursor.

```
example on 鸚 main [+] via 2 v3.11.0 via 鸚 epp_topics
> git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   example.py

example on 鸚 main [+] via 2 v3.11.0 via 鸚 epp_topics
> 
```

# First commit fails



```
example on 鸚 main [+] via 🐍 v3.11.0 via 🟢 epp_topics
> git commit -am "Add example.py."
black..... failed
- hook id: black
- files were modified by this hook

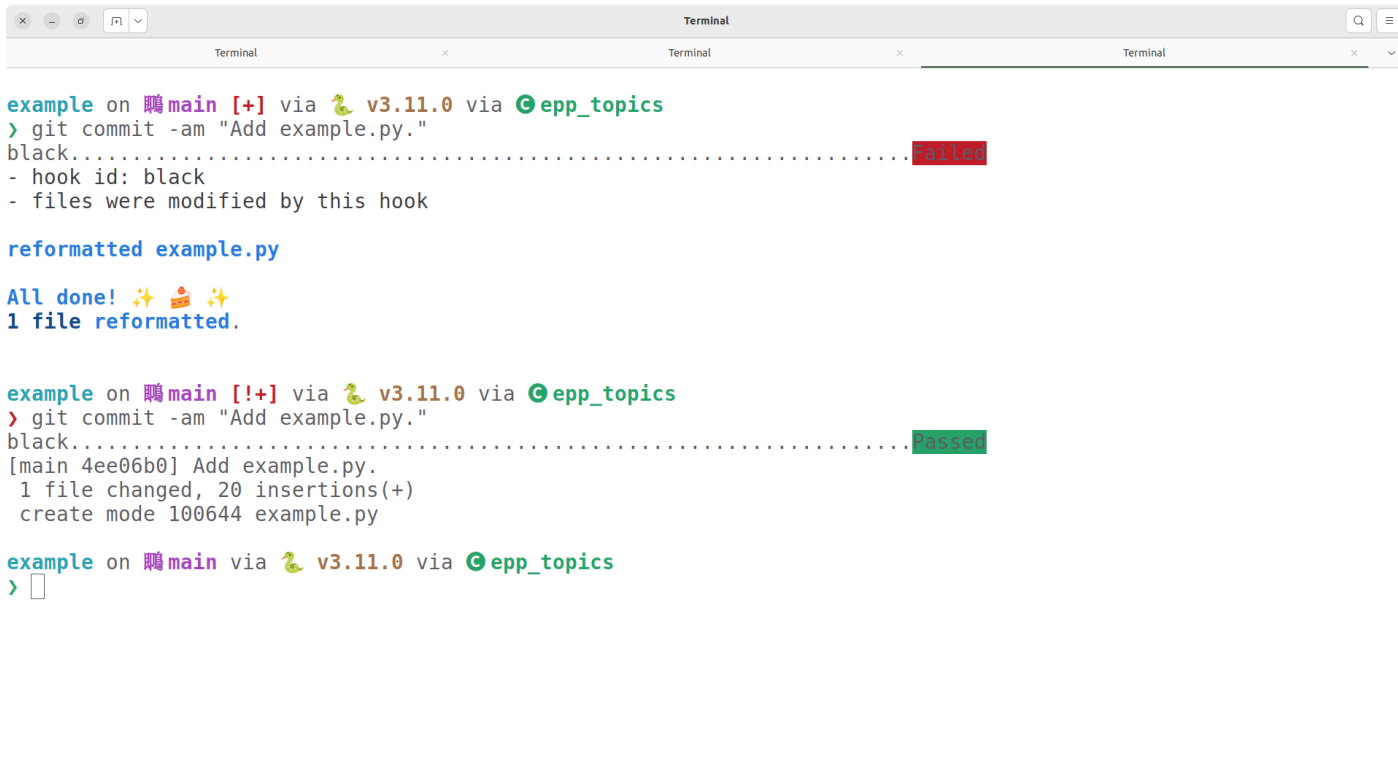
reformatted example.py

All done! ✨ 📦 ✨
1 file reformatted.

example on 鸚 main [!+] via 🐍 v3.11.0 via 🟢 epp_topics
> 
```



# Second commit works



```
example on 鸚 main [+] via 🐍 v3.11.0 via 🍷 epp_topics
> git commit -am "Add example.py."
black..... Fail
- hook id: black
- files were modified by this hook

reformatted example.py

All done! ✨ 📦 ✨
1 file reformatted.

example on 鸚 main [!+] via 🐍 v3.11.0 via 🍷 epp_topics
> git commit -am "Add example.py."
black..... Pass
[main 4ee06b0] Add example.py.
1 file changed, 20 insertions(+)
create mode 100644 example.py

example on 鸚 main via 🐍 v3.11.0 via 🍷 epp_topics
> 
```