

Software Engineering for Economists

7. Version Control, Part II: Git with a Shared Repository

Philipp Zahn

Department of Economics, University of St. Gallen

Collaboration

- ▶ Science in general (Hand, 2010):

[...] half of EU research articles had international co-authors in 2007, more than twice the level of two decades ago.

- ▶ Economics I (Ellison, 2002):

In the 1970s only 30 percent of the articles in the top five journals were coauthored. In the 1990s about 60 percent were coauthored. In the longer run the trend is even more striking: in 1959 only 3 percent of the articles in the JPE were coauthored.

- ▶ Economics II (Card and DellaVigna, 2013):

[...] the number of authors per paper [in one of the top-5 journals] has increased from 1.3 in 1970 to 2.3 in 2012

Collaboration

- ▶ Need effective means for working with others
- ▶ Same issue is to keep multiple machines in sync

Just work on Dropbox?

- ▶ Sensitive data

- ▶ Simultaneous work

Some Paper (Philipp's conflicted copy 2016-10-15).tex

⇒ **Fully automated synchronisation tools do not scale to complex (=real-world) workflows**

What Git has to add

- ▶ A (clear) protocol that supports complex workflows
- ▶ (Easy) merging of plain text files

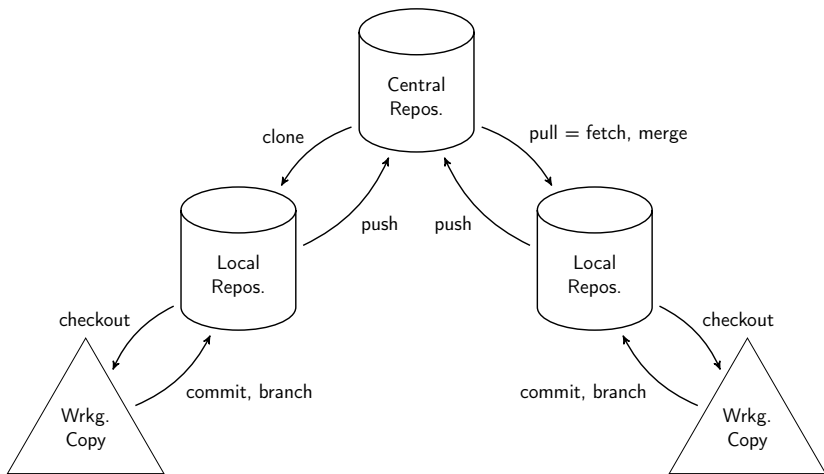
Plan for this lecture

- ▶ “Vocabulary” and basic concepts
- ▶ Creating a new project on github
- ▶ Basic workflow with a central repository
- ▶ Dealing with conflicting merges
- ▶ Local files unknown to Git that would be overwritten
- ▶ Recommended workflow

Github

- ▶ Please create a github account
- ▶ There are educational accounts
- ▶ Use it for your group project
- ▶ Add me as a user: *pzahn*
- ▶ Github usefull tool for software collaborations

Schematic Git workflow



Create a new project on github

- ▶ We initiate a project on github
- ▶ Will see other features later

New project on github

[Pull requests](#) [Issues](#) [Gist](#)

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner



pzahn

Repository name

tristan-and-isolde



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

Description (optional)



Public

Anyone can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

☐ Initialize this repository with a README

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add gitignore: [None](#)

Add a license: [None](#)



Create repository



New project on github

The screenshot shows the GitHub interface for the repository `pzahn / tristan-and-isolde`. The repository has 0 Watchers, 0 Stars, and 0 Forks. The main navigation bar includes links for Pull requests, Issues, and Gist. Below the repository name, there are tabs for Code, Issues, Pull requests, Projects, Wiki, Pulse, Graphs, and Settings. The 'Code' tab is selected, displaying the 'Quick setup' section.

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

or

We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

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

```
echo "# tristan-and-isolde" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/pzahn/tristan-and-isolde.git
git push -u origin master
```

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

```
git remote add origin https://github.com/pzahn/tristan-and-isolde.git
git push -u origin master
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

[Import code](#)

ProTip! Use the URL for this page when adding GitHub as a remote.

© 2016 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Help](#) [Contact GitHub](#) [API](#) [Training](#) [Shop](#) [Blog](#) [About](#)

Making a local clone of the project

Go to the parent folder of where you want your project to live

```
$ git clone https://github.com/pzahn/tristan-and-isolde.git
```

```
Cloning into 'tristan-and-isolde'...
```

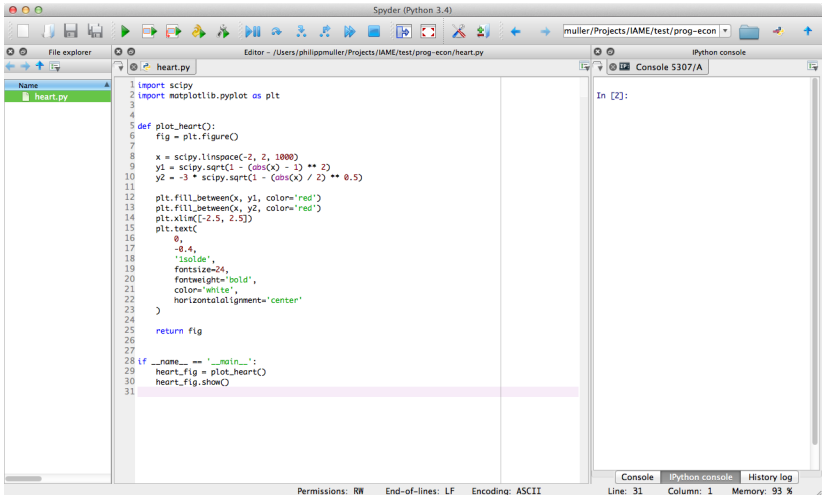
```
Username for 'https://github.com/pzahn/tristan-and-isolde.git': xxx
```

```
Password for 'https://xxx@git.yyy.de':
```

```
warning: You appear to have cloned an empty repository.
```

Tristan's heart script

Add a Python script and the .gitignore file to the project



The screenshot shows the Spyder Python IDE interface. The top toolbar contains various icons for file operations and execution. The main window is divided into three panes: a File explorer on the left showing the project structure, an Editor in the center displaying the code for 'heart.py', and an IPython console on the right.

File explorer: Shows a folder named 'heart.py'.

Editor: Displays the code for 'heart.py' with line numbers 1 through 31. The code imports 'scipy' and 'matplotlib.pyplot as plt', defines a 'plot_heart()' function, and includes a main guard.

```
1 import scipy
2 import matplotlib.pyplot as plt
3
4
5 def plot_heart():
6     fig = plt.figure()
7
8     x = scipy.linspace(-2, 2, 1000)
9     y1 = scipy.sqrt(1 - (abs(x) - 1) ** 2)
10    y2 = -3 * scipy.sqrt(1 - (abs(x) / 2) ** 0.5)
11
12    plt.fill_between(x, y1, color='red')
13    plt.fill_between(x, y2, color='red')
14    plt.xlim([-2.5, 2.5])
15    plt.text(
16        0,
17        -0.4,
18        'Isolde',
19        fontsize=24,
20        fontweight='bold',
21        color='white',
22        horizontalalignment='center'
23    )
24
25    return fig
26
27
28 if __name__ == '__main__':
29     heart_fig = plot_heart()
30     heart_fig.show()
31
```

IPython console: Shows the prompt 'In [2]:'.

Status bar: At the bottom, it displays 'Permissions: RW', 'End-of-lines: LF', 'Encoding: ASCII', 'Line: 31', 'Column: 1', and 'Memory: 93 %'.

Tristan's local Git preparations

Add the Python script to the index to prepare first commit

```
$ git add heart.py
```

```
$ git status
```

```
On branch master
```

```
Initial commit
```

```
Changes to be committed:
```

```
(use "git reset HEAD <file>..." to unstage)
```

```
new file:   heart.py
```

Tristan's local Git preparations

Commit first version to the **local** repository

```
$ git commit -m "First version of heart."
```

```
[master (root-commit) 819aec2] First version of heart.  
1 file changed, 30 insertions(+)  
create mode 100644 heart.py
```

```
$ git status
```

```
On branch master  
nothing to commit, working directory clean
```

First push to the remote repository

Now push the changes made to the local repository to the remote (central) repository

```
$ git push origin master
```

```
Username for 'https://github.com/pzahn/tristan-and-isolde.git': tri  
Password for 'https://tristan@git.yyy.de':
```

```
Counting objects: 4, done.
```

```
Delta compression using up to 2 threads.
```

```
Compressing objects: 100% (4/4), done.
```

```
Writing objects: 100% (4/4), 631 bytes | 0 bytes/s, done.
```

```
Total 4 (delta 0), reused 0 (delta 0)
```

```
To https://github.com/pzahn/tristan-and-isolde.git/tristan-and-isol
```

```
* [new branch]      master -> master
```


First push to the remote repository

Next, have a look at:

<http://git-scm.com/docs/git-credential-store>

Automate, automate, automate!

Isolde finds the code online

The screenshot shows the GitHub interface for the repository 'pzahn / tristan-and-isolde'. At the top, there's a navigation bar with links for 'Pull requests', 'Issues', and 'Gist'. Below this, the repository name is displayed along with 'Watch', 'Star', and 'Fork' buttons. A secondary navigation bar includes links for 'Code', 'Issues', 'Pull requests', 'Projects', 'Wiki', 'Pulse', 'Graphs', and 'Settings'. The main content area starts with a message 'No description or website provided. — Edit'. Below this, statistics show '1 commit', '1 branch', '0 releases', and '1 contributor'. A row of buttons includes 'Branch: master', 'New pull request', 'Create new file', 'Upload files', 'Find file', and 'Clone or download'. The commit history shows a single commit by 'phmato' titled 'First version of heart.py' from 3 minutes ago. Below the commit list, there's a prompt to 'Add a README' to help people understand the project. The footer contains copyright information for GitHub, Inc. and links for 'Contact GitHub', 'API', 'Training', 'Shop', 'Blog', and 'About'.

This repository Search

Pull requests Issues Gist

++

pzahn / tristan-and-isolde

Watch 0 Star 0 Fork 0

Code Issues Pull requests Projects Wiki Pulse Graphs Settings

No description or website provided. — Edit

1 commit 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

phmato First version of heart.py Latest commit 3 minutes ago

heart.py First version of heart.py 3 minutes ago

Help people interested in this repository understand your project by adding a README. Add a README

© 2016 GitHub, Inc. Terms Privacy Security Status Help

Contact GitHub API Training Shop Blog About

Isolde clones the repository

Assume Isolde changed to the folder of her workspace on the command line. There, she clones the project from the **remote** repository

```
$ git clone https://github.com/pzahn/tristan-and-isolde.git/tristan-and
```

```
Cloning into 'tristan-and-isolde'...
```

```
Username for 'https://github.com/pzahn/tristan-and-isolde.git': isolde
```

```
Password for 'https://isolde@git.yyy.de':
```

```
fatal: unable to access 'https://github.com/pzahn/tristan-and-isolde.git':
```

```
The requested URL returned error: 500
```

Wrong credentials are the No. 1 cause of 500 errors.

Isolde clones the repository

Make sure to enter the username and password correctly

```
$ git clone https://github.com/pzahn/tristan-and-isolde.git/tristan
```

```
Cloning into 'tristan-and-isolde'...
```

```
Username for 'https://github.com/pzahn/tristan-and-isolde.git': iso
```

```
Password for 'https://isolde@git.yyy.de':
```

```
remote: Counting objects: 4, done.
```

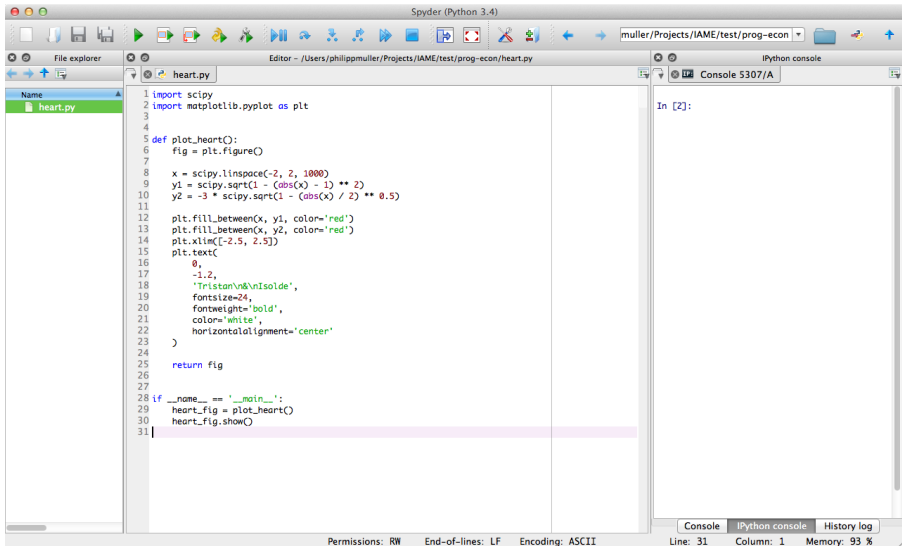
```
remote: Compressing objects: 100% (4/4), done.
```

```
remote: Total 4 (delta 0), reused 0 (delta 0)
```

```
Unpacking objects: 100% (4/4), done.
```

```
Checking connectivity... done
```

Isolde changes the code



The image shows the Spyder Python IDE interface. The top toolbar contains various icons for file operations and code execution. The main window is divided into three panes:

- File explorer:** Shows the file structure with 'heart.py' selected.
- Editor:** Displays the code for 'heart.py'. The code defines a function 'plot_heart()' that generates a heart shape using NumPy and Matplotlib, and includes a main block to execute the plot.
- IPython console:** Shows the prompt 'In [2]:'.

The code in the editor is as follows:

```
1 import scipy
2 import matplotlib.pyplot as plt
3
4
5 def plot_heart():
6     fig = plt.figure()
7
8     x = scipy.linspace(-2, 2, 1000)
9     y1 = scipy.sqrt(1 - (abs(x) - 1) ** 2)
10    y2 = -3 * scipy.sqrt(1 - (abs(x) / 2) ** 0.5)
11
12    plt.fill_between(x, y1, color='red')
13    plt.fill_between(x, y2, color='red')
14    plt.xlim([-2.5, 2.5])
15    plt.text(
16        0,
17        -1.2,
18        'Tristan\n&\nIsolde',
19        fontsize=24,
20        fontweight='bold',
21        color='white',
22        horizontalalignment='center'
23    )
24
25    return fig
26
27
28 if __name__ == '__main__':
29     heart_fig = plot_heart()
30     heart_fig.show()
31
```

The status bar at the bottom indicates: Permissions: RW, End-of-lines: LF, Encoding: ASCII, Line: 31, Column: 1, Memory: 93 %.

She is happy, commits, and pushes

Add Python file to the index and commit

```
$ git add heart.py
```

```
$ git status
```

```
On branch master
```

```
Changes to be committed:
```

```
(use "git reset HEAD <file>..." to unstage)
```

```
modified:   heart.py
```

```
$ git commit -m "Included Tristan in the heart's message."
```

```
[master 10737a6] Included Tristan in the heart's message.
```

```
1 file changed, 2 insertions(+), 2 deletions(-)
```

She is happy, commits, and pushes

```
$ git push origin master
```

```
Username for 'https://github.com/pzahn/tristan-and-isolde.git': iso
```

```
Password for 'https://isolde@git.yyy.de':
```

```
Counting objects: 5, done.
```

```
Delta compression using up to 2 threads.
```

```
Compressing objects: 100% (2/2), done.
```

```
Writing objects: 100% (3/3), 325 bytes | 0 bytes/s, done.
```

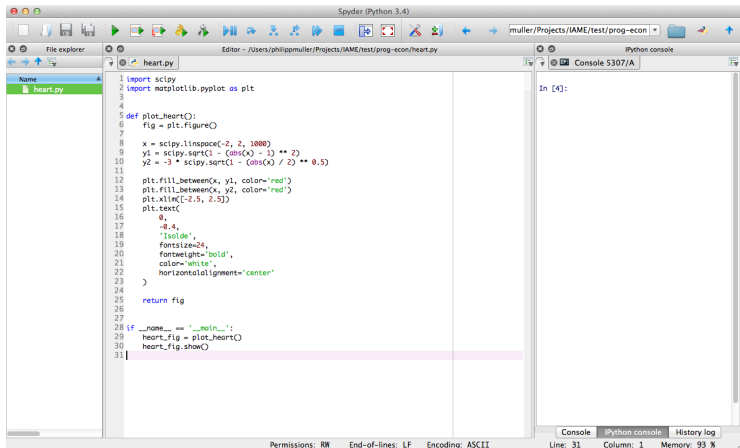
```
Total 3 (delta 1), reused 0 (delta 0)
```

```
To https://github.com/pzahn/tristan-and-isolde.git/tristan-and-isol
```

```
819aec2..ba7e4bb master -> master
```

At the same time ...

At the same time ...



The image shows the Spyder Python IDE interface. The main editor window displays a Python script named `heart.py` located at `/Users/philippmuller/Projects/IAME/test/program-econ/heart.py`. The script defines a function `plot_heart()` that uses `matplotlib.pyplot` to create a plot of a heart shape. The heart is filled with red, and the axes are labeled with 'Isolde' in bold white font. The script also includes a main block that calls `plot_heart()` and `show()` the figure.

```
1 import scipy
2 import matplotlib.pyplot as plt
3
4 def plot_heart():
5     fig = plt.figure()
6
7     x = scipy.linspace(-2, 2, 1000)
8     y1 = scipy.sqrt(1 - (abs(x) - 1) ** 2)
9     y2 = -3 * scipy.sqrt(1 - (abs(x) / 2) ** 0.5)
10
11     plt.fill_between(x, y1, color='red')
12     plt.fill_between(x, y2, color='red')
13     plt.xlim([-2.5, 2.5])
14     plt.text(
15         0,
16         -0.4,
17         'Isolde',
18         fontsize=24,
19         fontweight='bold',
20         color='white',
21         horizontalalignment='center'
22     )
23
24     return fig
25
26
27
28 if __name__ == '__main__':
29     heart_fig = plot_heart()
30     heart_fig.show()
31
```

The IPython console on the right shows the prompt `In [4]:`. The status bar at the bottom indicates the file permissions are `RW`, the end-of-line character is `LF`, the encoding is `ASCII`, and the current position is `Line: 31, Column: 1, Memory: 93 %`.

At the same time ...

He commits again and intends to push ...

```
$ git add heart.py
```

```
$ git commit -m "Fixed typo."
```

```
[master 466dfba] Fixed typo.
```

```
1 file changed, 1 insertion(+), 1 deletion(-)
```

What has happened?

...but there is an error message on the command line

```
$ git push origin master
```

```
Username for 'https://github.com/pzahn/tristan-and-isolde.git': tristan
Password for 'https://tristan@git.yyy.de':
To https://github.com/pzahn/tristan-and-isolde.git/tristan-and-isolde.g
! [rejected]          master -> master (fetch first)
error: failed to push some refs to
'https://github.com/pzahn/tristan-and-isolde.git/tristan-and-isolde.git'
```

Updates were rejected because the remote contains work that you do not have locally. This is usually caused by another repository pushing to the same ref. You may want to first merge the remote changes (e.g., 'git pull') before pushing again.

See the 'Note about fast-forwards' in 'git push --help' for details.

What has happened?

Translation

This error can be a bit overwhelming at first, do not fear. **Simply put, git cannot make the change on the remote without losing commits, so it refuses the push.** Usually this is caused by another user pushing to the same branch. You can remedy this by fetching and merging the remote branch, or using git pull to perform both at once.

origin/master
Included Tristan in ...
c36927205

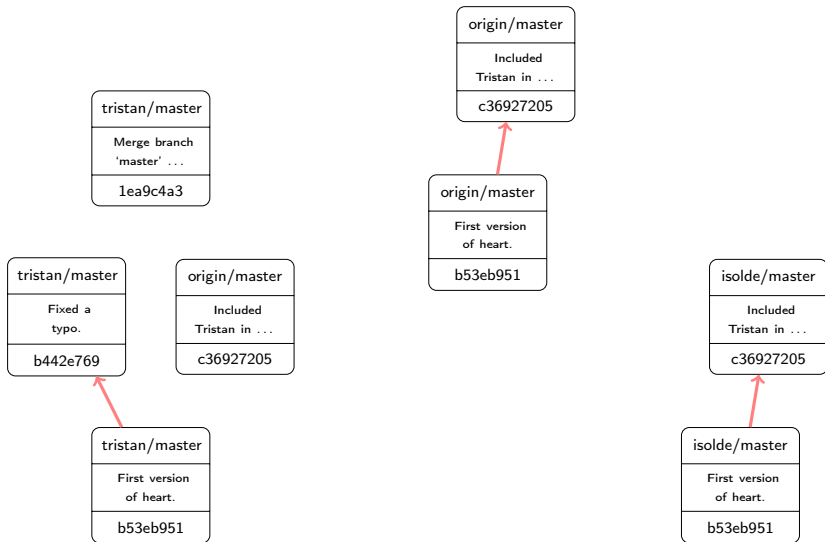
origin/master
First version of heart.
b53eb951

isolde/master
Included Tristan in ...
c36927205

tristan/master
Fixed a typo.
b442e769

tristan/master
First version of heart.
b53eb951

isolde/master
First version of heart.
b53eb951



Tristan configures and pulls the remote

The pull is essentially a fetch followed by a merge.

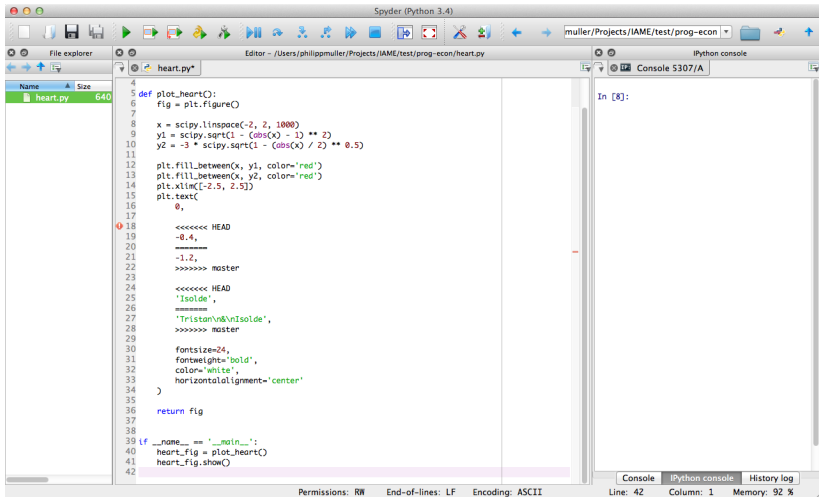
However, there are conflicts

```
$ git pull origin master
```

```
Username for 'https://github.com/pzahn/tristan-and-isolde.git': tris  
Password for 'https://tristan@git.yyy.de':
```

```
From https://github.com/pzahn/tristan-and-isolde.git/tristan-and-isolde  
* branch                master          -> FETCH_HEAD  
Auto-merging heart.py  
CONFLICT (content): Merge conflict in heart.py  
Automatic merge failed; fix conflicts and then commit the result.
```

Tristan solves merge conflicts



The screenshot shows the Spyder Python IDE interface. The File explorer on the left shows the file `heart.py` with a size of 640 bytes. The Editor window displays the code for `heart.py`, which defines a function `plot_heart()` to plot a heart shape. The code includes comments for a merge conflict resolution, with lines 18-22 and 24-29 showing the conflict and the resolution. The IPython console on the right shows the command `In [8]:`. The status bar at the bottom indicates the file permissions are `RW`, the end-of-lines are `LF`, the encoding is `ASCII`, and the current line is 42, column 1, with 92% memory usage.

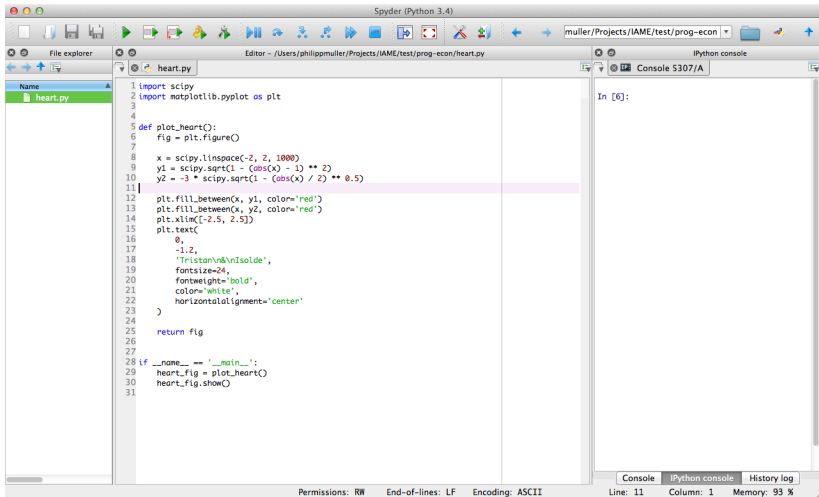
```
4
5 def plot_heart():
6     fig = plt.figure()
7
8     x = scipy.linspace(-2, 2, 1000)
9     y1 = scipy.sqrt(1 - (abs(x) - 1) ** 2)
10    y2 = -3 * scipy.sqrt(1 - (abs(x) / 2) ** 0.5)
11
12    plt.fill_between(x, y1, color='red')
13    plt.fill_between(x, y2, color='red')
14    plt.xlim([-2.5, 2.5])
15    plt.text(
16        0,
17
18        <<<<<< HEAD
19        -0.4,
20        =====
21        -1.2,
22        >>>>>> master
23
24        <<<<<< HEAD
25        'Isolde',
26        =====
27        'Tristan\n&\nIsolde',
28        >>>>>> master
29
30        fontsize=24,
31        fontweight='bold',
32        color='white',
33        horizontalalignment='center'
34    )
35
36    return fig
37
38
39 if __name__ == '__main__':
40     heart_fig = plot_heart()
41     heart_fig.show()
42
```

Console: Console 5307/A

In [8]:

Permissions: RW End-of-lines: LF Encoding: ASCII Line: 42 Column: 1 Memory: 92 %

Tristan solves merge conflicts



The image shows the Spyder Python IDE interface. The main editor window displays the file `heart.py` with the following Python code:

```
1 import scipy
2 import matplotlib.pyplot as plt
3
4
5 def plot_heart():
6     fig = plt.figure()
7
8     x = scipy.linspace(-2, 2, 1000)
9     y1 = scipy.sqrt(1 - (abs(x) - 1) ** 2)
10    y2 = -3 * scipy.sqrt(1 - (abs(x) / 2) ** 0.5)
11
12    plt.fill_between(x, y1, color='red')
13    plt.fill_between(x, y2, color='red')
14    plt.xlim([-2.5, 2.5])
15    plt.text(
16        0,
17        -1.2,
18        'Tristan\n&\nIsolde',
19        fontsize=24,
20        fontweight='bold',
21        color='white',
22        horizontalalignment='center'
23    )
24
25    return fig
26
27
28 if __name__ == '__main__':
29     heart_fig = plot_heart()
30     heart_fig.show()
31
```

The IPython console on the right shows the prompt `In [6]:`. The status bar at the bottom indicates: Permissions: RW, End-of-lines: LF, Encoding: ASCII, Line: 11, Column: 1, Memory: 93 %.

Tools to resolve conflicts

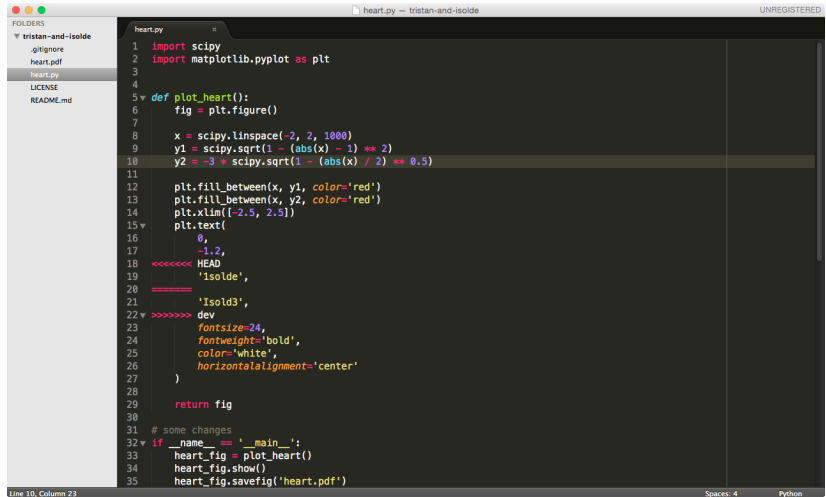
- ▶ Different editors come with different support tools to make resolving conflicts easier
- ▶ Vim: vimdiff
- ▶ Sublime: Git Conflict Editor

Git Conflict Resolver (Sublime Text Editor)

Git Conflict Resolver ships with five commands: "Find Next Conflict", "Keep Ours", "Keep Theirs", "Keep Common Ancestor" and "Show Conflict Files".

Git Conflict Resolver (Sublime Text Editor)

Say we encounter a merge conflict



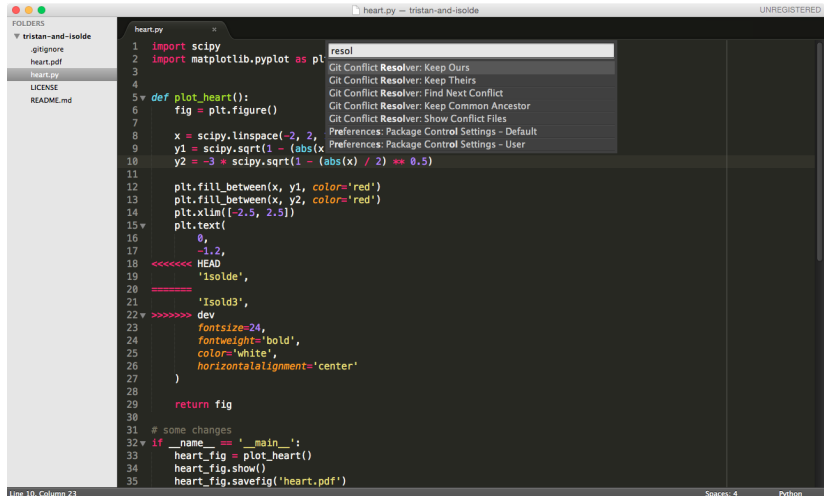
```
heart.py
1 import scipy
2 import matplotlib.pyplot as plt
3
4
5 def plot_heart():
6     fig = plt.figure()
7
8     x = scipy.linspace(-2, 2, 1000)
9     y1 = scipy.sqrt(1 - (abs(x) - 1) ** 2)
10    y2 = -3 * scipy.sqrt(1 - (abs(x) / 2) ** 0.5)
11
12    plt.fill_between(x, y1, color='red')
13    plt.fill_between(x, y2, color='red')
14    plt.xlim([-2.5, 2.5])
15    plt.text(
16        0,
17        -1.2,
18        <<<<<< HEAD
19        'Isolde',
20        =====
21        'Isold3',
22    >>>>>> dev
23        fontsize=24,
24        fontweight='bold',
25        color='white',
26        horizontalalignment='center'
27    )
28
29    return fig
30
31 # some changes
32 if __name__ == '__main__':
33     heart_fig = plot_heart()
34     heart_fig.show()
35     heart_fig.savefig('heart.pdf')
```

Line 10, Column 23

Spaces: 4 Python

Git Conflict Resolver (Sublime Text Editor)

We can invoke the plugin to choose from standardized options

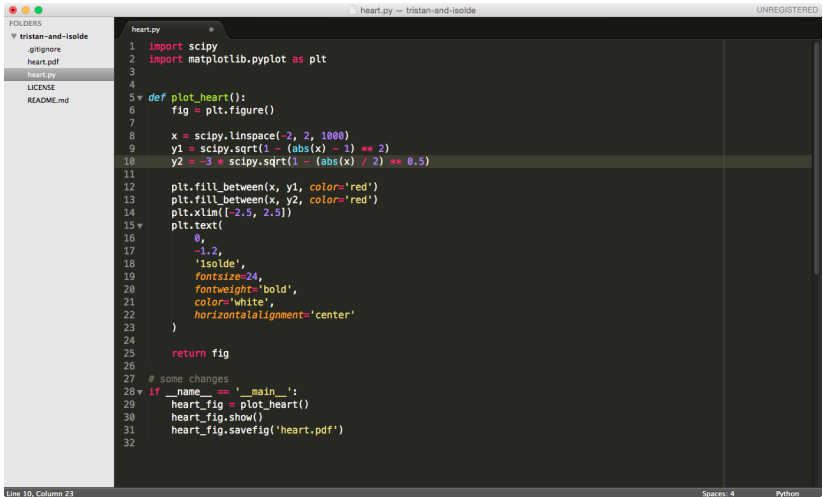


The screenshot shows the Sublime Text Editor interface with a file named `heart.py` open. The editor is displaying a Python script that uses `scipy` and `matplotlib.pyplot` to create a plot. A Git conflict is visible in the code, with lines 18-21 marked with `<<<<<<` and lines 20-21 marked with `=====`. A dropdown menu is open, showing the `resolve` command and a list of options: `Git Conflict Resolver: Keep Ours`, `Git Conflict Resolver: Keep Theirs`, `Git Conflict Resolver: Find Next Conflict`, `Git Conflict Resolver: Keep Common Ancestor`, `Git Conflict Resolver: Show Conflict Files`, `Preferences: Package Control Settings - Default`, and `Preferences: Package Control Settings - User`. The status bar at the bottom indicates 'Line 10, Column 23', 'Spaces: 4', and 'Python'.

```
1 import scipy
2 import matplotlib.pyplot as plt
3
4
5 def plot_heart():
6     fig = plt.figure()
7
8     x = scipy.linspace(-2, 2,
9     y1 = scipy.sqrt(1 - (abs(x
10    y2 = -3 * scipy.sqrt(1 - (abs(x) / 2) ** 0.5)
11
12    plt.fill_between(x, y1, color='red')
13    plt.fill_between(x, y2, color='red')
14    plt.xlim([-2.5, 2.5])
15    plt.text(
16        0,
17        -1.2,
18    <<<<<< HEAD
19    'Isolde',
20    =====
21    'Isold3',
22    >>>>>> dev
23    fontsize=24,
24    fontweight='bold',
25    color='white',
26    horizontalalignment='center'
27    )
28
29    return fig
30
31 # some changes
32 if __name__ == '__main__':
33     heart_fig = plot_heart()
34     heart_fig.show()
35     heart_fig.savefig('heart.pdf')
```

Git Conflict Resolver (Sublime Text Editor)

Sublime will then adjust our code accordingly



```
1 import scipy
2 import matplotlib.pyplot as plt
3
4
5 def plot_heart():
6     fig = plt.figure()
7
8     x = scipy.linspace(-2, 2, 1000)
9     y1 = scipy.sqrt(1 - (abs(x) - 1) ** 2)
10    y2 = -3 * scipy.sqrt(1 - (abs(x) / 2) ** 0.5)
11
12    plt.fill_between(x, y1, color='red')
13    plt.fill_between(x, y2, color='red')
14    plt.xlim([-2.5, 2.5])
15    plt.text(
16        0,
17        -1.2,
18        'Isolde',
19        fontsize=24,
20        fontweight='bold',
21        color='white',
22        horizontalalignment='center'
23    )
24
25    return fig
26
27 # some changes
28 if __name__ == '__main__':
29     heart_fig = plot_heart()
30     heart_fig.show()
31     heart_fig.savefig('heart.pdf')
32
```

Line 10, Column 23 Spaces: 4 Python

Commit the merged file

Check the current status and add the modified file

```
$ git status
```

```
On branch master
```

```
You have unmerged paths.
```

```
(fix conflicts and run "git commit")
```

```
Unmerged paths:
```

```
(use "git add <file>..." to mark resolution)
```

```
both modified:      heart.py
```

```
no changes added to commit
```

```
(use "git add" and/or "git commit -a")
```

```
$ git add heart.py
```

Commit the merged file

Commit by using the default commit message after a merge

```
$ git commit
```

```
[master 3842604] Merge branch 'master' of  
https://github.com/pzahn/tristan-and-isolde.git/tristan-and-isolde
```

```
$ git log
```

```
commit 38426040e4843a690b54287dc4f2f9f12424391c  
Merge: 466dfba ba7e4bb  
Author: Hans-Martin v. Gaudecker <hmgaudecker@uni-bonn.de>  
Date: Thu Oct 24 14:07:29 2013 +0200
```

```
Merge branch 'master' of https://github.com/pzahn/tristan-and-isolde
```

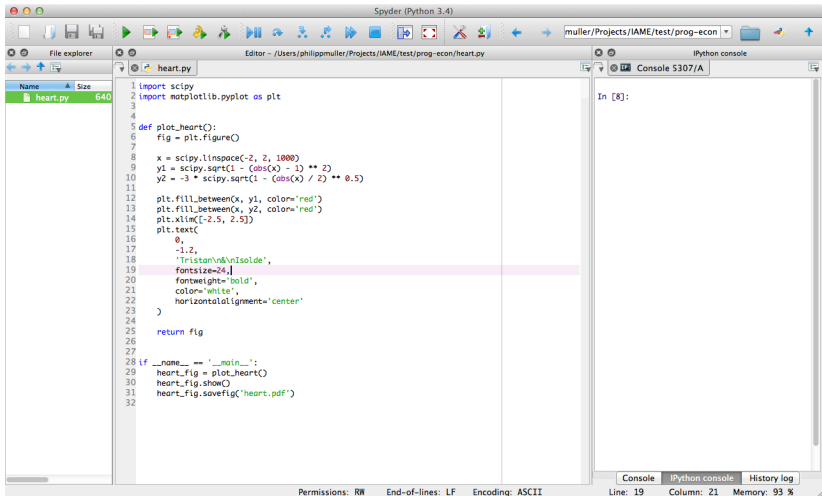
```
Conflicts:  
    heart.py
```


The standard case

- ▶ Explicitly distinguishing between the **fetch** and **merge** steps in a **pull** is seldom necessary
- ▶ Git will quietly do a **fast-forward** or **recursive** merge
 - ▶ No changes at all in your repository and working copy
 - ▶ Non-conflicting changes in your repository and working copy (different files or separate chunks of the same file affected in the two branches)
- ▶ Need to watch out for some weird cases
 - ▶ Add the same function at the top and at the bottom of a Python script and wonder why the first one never gets called...

Existing files would be overwritten by a pull

Tristan adjusts his code so heart.pdf is created



```
1 import scipy
2 import matplotlib.pyplot as plt
3
4
5 def plot_heart():
6     fig = plt.figure()
7
8     x = scipy.linspace(-2, 2, 1000)
9     y1 = scipy.sqrt(1 - (abs(x) - 1) ** 2)
10    y2 = -3 * scipy.sqrt(1 - (abs(x) / 2) ** 0.5)
11
12    plt.fill_between(x, y1, color='red')
13    plt.fill_between(x, y2, color='red')
14    plt.xlim([-2.5, 2.5])
15    plt.text(
16        0,
17        -1.2,
18        'Tristan\nIsolde',
19        fontsize=24,
20        fontweight='bold',
21        color='white',
22        horizontalalignment='center'
23    )
24
25    return fig
26
27
28 if __name__ == '__main__':
29     heart_fig = plot_heart()
30     heart_fig.show()
31     heart_fig.savefig('heart.pdf')
32
```

In [8]:

Console 5307/A

Permissions: RW End-of-lines: LF Encoding: ASCII

Line: 19 Column: 21 Memory: 93 %

Existing files would be overwritten by a pull

He adds heart.pdf (Note: we need the force switch here), commits and ...

```
$ ls
```

```
heart.pdf  heart.py
```

```
$ git add heart.pdf -f
```

```
$ git commit -m "Added the heart itself to Git. Bad idea..."
```

```
[master aeeac81] Added the heart itself to Git. Bad idea...  
1 file changed, 0 insertions(+), 0 deletions(-)  
create mode 100644 heart.pdf
```

Existing files would be overwritten by a pull

...pushes

```
$ git push origin master
```

```
Username for 'https://github.com/pzahn/tristan-and-isolde.git': trista
```

```
Password for 'https://tristan@git.yyy.de':
```

```
Counting objects: 13, done.
```

```
Delta compression using up to 2 threads.
```

```
Compressing objects: 100% (7/7), done.
```

```
Writing objects: 100% (9/9), 37.62 KiB | 0 bytes/s, done.
```

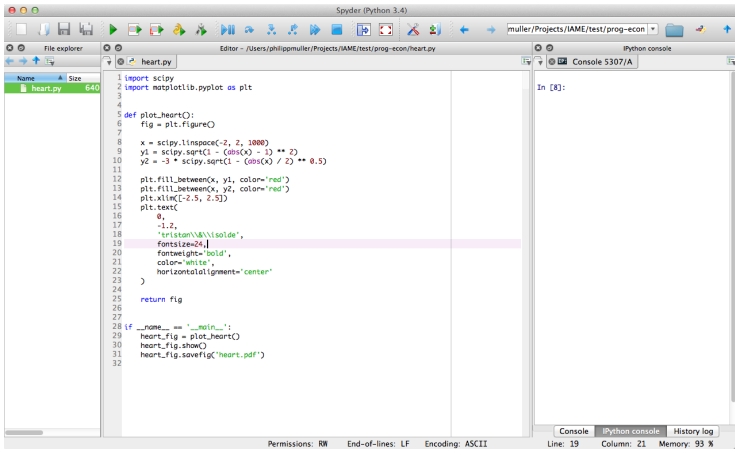
```
Total 9 (delta 2), reused 0 (delta 0)
```

```
To https://github.com/pzahn/tristan-and-isolde.git/tristan-and-isolde.
```

```
ba7e4bb..aeeac81 master -> master
```

Existing files would be overwritten by a pull

Isolde also created `heart.pdf`, not added to index or committed



```
1 import scipy
2 import matplotlib.pyplot as plt
3
4
5 def plot_heart():
6     fig = plt.figure()
7
8     x = scipy.linspace(-2, 2, 1000)
9     y1 = scipy.sqrt(1 - (abs(x) - 1) ** 2)
10    y2 = -3 * scipy.sqrt(1 - (abs(x) / 2) ** 0.5)
11
12    plt.fill_between(x, y1, color='red')
13    plt.fill_between(x, y2, color='red')
14    plt.xlim([-2.5, 2.5])
15    plt.text(
16        0,
17        -1.2,
18        'riscan\\Isolde',
19        fontsize=24,
20        fontweight='bold',
21        color='white',
22        horizontalalignment='center'
23    )
24
25    return fig
26
27
28 if __name__ == '__main__':
29     heart_fig = plot_heart()
30     heart_fig.show()
31     heart_fig.savefig('heart.pdf')
32
```

File explorer: heart.py 640

Editor: /Users/philippmuller/Projects/IAME/test/prog-econ/heart.py

IPython console: Console 5307/A

In [8]:

Permissions: RW End-of-lines: LF Encoding: ASCII

Line: 19 Column: 21 Memory: 93 %

Existing files would be overwritten by a pull

- ▶ Tristan's push happend before
- ▶ Thus, while Isolde was making her changes, the **remote** repository on the Server changed

Existing files would be overwritten by a pull

Isolde pulls from the **remote** repository and runs into trouble

```
$ git pull origin master
```

```
Username for 'https://github.com/pzahn/tristan-and-isolde.git': isolde
Password for 'https://isolde@git.yyy.de':
```

```
remote: Counting objects: 13, done.
```

```
remote: Compressing objects: 100% (7/7), done.
```

```
remote: Total 9 (delta 2), reused 0 (delta 0)
```

```
Unpacking objects: 100% (9/9), done.
```

```
From https://github.com/pzahn/tristan-and-isolde.git/tristan-and-isolde
```

```
* branch          master      -> FETCH_HEAD
```

```
Updating ba7e4bb..aeeac81
```

```
error: The following untracked working tree files would be
overwritten by merge:
```

```
    heart.pdf
```

```
Please move or remove them before you can merge.
```

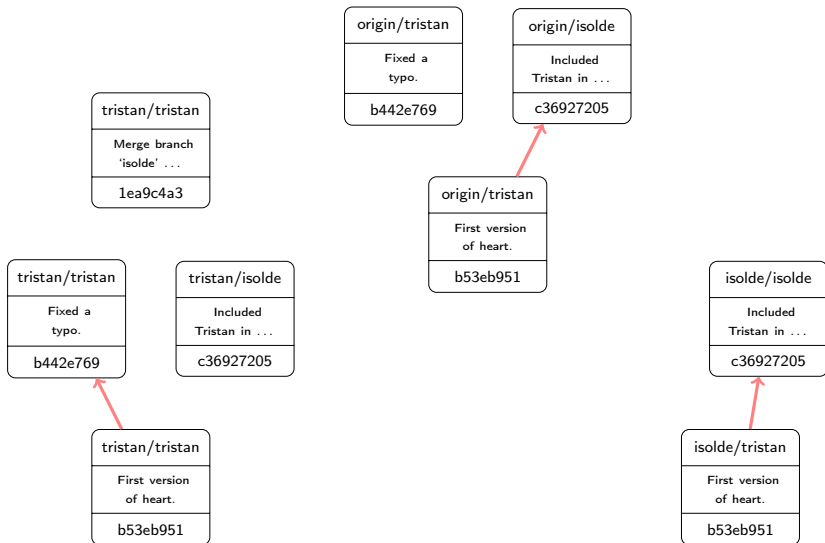
```
Aborting
```

Existing files would be overwritten by a pull

This issue can be overcome by moving the file or deleting it beforehand

Recommended workflow in teams

- ▶ Everybody has his or her own branch
- ▶ Frequent merges
- ▶ Potentially a master branch where only universally accepted changes enter
- ▶ Benefits:
 - ▶ Freedom to merge only when it is convenient
 - ▶ You can always push your changes upstream
 - ▶ No “murky” FETCH_HEAD’s etc.



Recommended workflow in teams

- ▶ If you encounter merge conflicts frequently, it is a sign that something is wrong with the workflow in your project
 - ▶ Not talking to co-authors as often as you should?
 - ▶ Responsibilities not clearly assigned?
- ▶ Git helps you detect this

Wrapping up

- ▶ Git provides powerful tools for teams' workflows
- ▶ Scales up all the way from single developers to Linux kernel development
- ▶ It sure is not for the faint of heart
- ▶ But what we have seen should get you started
 - ▶ Don't be afraid to play around
 - ▶ Not much can go wrong if you push to the central repository frequently
- ▶ Ask questions in the forum on the server

The easy way out . . .

THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.

COOL. HOW DO WE USE IT?

NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.



Acknowledgements

- ▶ This course is based on the course Effective Programming Practices for Economists designed by Hans-Martin von Gaudecker for economists.
- ▶ The material has been adapted and shortened. Remaining errors are mine.
- ▶ The Effective Programming Practices for Economists course material is made available under a Creative Commons Attribution License.