Rebase is base

git basics workshop

Jiri Vlasak

Sep 2024

Introduction

Introduction

What?

- We know: that we don't know much
- We will: workshop

Overview of content

- Introduction
- Baby steps
- Creating Python3 package
- Rewriting git history
- Alternative history
- Conflicting history
- Rather rebase (those) conflicts

Not considered

- Collaboration, patches, merge/pull requests
- Remotes, synchronization of remote branches

I will trick you

See Pro Git or `git COMMAND --help` to avoid being tricked.

This is workshop

- I will be typing a lot, soon.
- You should be typing a lot, too.

Why (bother to) learn git?

- Craftsmanship
- Altruism

It's like . . .

- Typing all ten fingers
- Using tiling window manager
- Watching youtube videos speed 2x

. . . but you effect efficiency of others

- Writing nice and understandable code
- Clean history of changes (e.g., for code review)

Git is . . .

. . . program for tracking changes to the lines of files in a repository.

That is

- Repository is a directory.
- There are files in that directory.
- The lines of that files change.
- Git tracks those changes.
- A set of changes is called a commit.
- Commits are stacked one on the other.
- Commits form the Directed Acyclic Graph.

Git tracks *how* the current state of the repository is built.

The workshop goal (not really)

Create Python3 package called `args` ...

```
$ cd py-args-pkg ; tree
.
|-- args
|   |-- evaluator.py
|   `-- __init__.py
|-- LICENSE
|-- README
`-- tests
    |-- test_sum.py
    |-- test_prod.py
    |-- test_nonincr.py
    `-- test_nondecr.py

    2 directories, 8 files
```

The workshop goal (not really)

... that contains module `evaluator` with the `ev` procedure:

```
def ev(op, *args):
    """Return value computed from ``args`` based on ``op``.

    Procedure ``ev`` returns:
    - sum of arguments for ``op == "+"``,
    - product of arguments for ``op == "*"``,
    - true if arguments are non-descending for ``op == "<"``,
    - true if arguments are non-increasing for ``op == ">"``.

    :param op: A (string) operator to be applied to ``args``.
    :param args: The list (of numbers) to apply ``op`` to.
    """
    pass  # TODO
```

TODO

1. Baby steps
   1. Initialize repository
   2. Add license, readme
2. Creating Python3 package
   1. Add operator “+”
3. Rewriting git history
   1. Reorder commits
   2. Add operator "*"
   3. Add docstrings
4. Alternative history
   1. Designate operators “+” and "*" in history
5. Conflicting history
   1. Add operators “<” and “>”
   2. While updating `README`
6. Rather rebase (those) conflicts
   1. Add operators “<” and “>” the rebase way

Baby steps

Baby steps

What?

- We know: what git is and what we'll do
- We will: initialize repository and add first files

Git history before

Git history after

```
* d0dbc16 (HEAD -> master) Add license, readme
```

Table of content

- Repository initialization
- Add license, readme

New Git commands

```
git init
git config user.email 'foo@bar.buzz'
git config --global user.name 'Foo Bar'

git config core.autocrlf false
git config core.eol lf
git config --global core.editor 'notepad'

git status, git commit, git log, git show
git add LICENSE README
git diff --cached
git log --oneline --graph --decorate --all
```

Summary and questions

Set `git logg` command

```
git config --global --add alias.logg \
    'log --oneline --graph --decorate --all'
git logg
```

Creating Python3 package

Creating Python3 package

What?

- We know: how to add files and see git history
- We will: calm down and practice a bit

Git history before

```
* d0dbc16 (HEAD -> master) Add license, readme
```

Git history after

```
* 1fe5611 (HEAD -> master) Add gitignore
* 096d3c8 Add operator "+" unit test
* 0da38ec Add evaluator module, operator "+"
* fe22dec Add args Python3 package
* d0dbc16 Add license, readme
```

Table of content

- Add package (`args/__init__.py`)
- Add module (`args/evaluator.py`)
- Define `ev` and "+" in the `evaluator` module
- Add tests for "+"
- Add `.gitignore`

New Git and Python commands

```
python3 -m unittest discover tests

git logg
git show HEAD^
git commit -m'When applied, this commit will ... (max. 52)'
```

Summary and questions

Unit test for operator “+”

```
$ cat tests/test_sum.py
from unittest import TestCase

from args.evaluator import ev


class TestEvaluator(TestCase):
    def test_sum(self):
        assert 6 == ev("+", 1, 2, 3)
```

Rewriting git history

Rewriting git history

What?

- We know: how to add files, see git history, and run tests
- We will: manipulate git history

Git history

Before

```
* 1fe5611 (HEAD -> master) Add gitignore
* 096d3c8 Add operator "+" unit test
* 0da38ec Add evaluator module, operator "+"
* fe22dec Add args Python3 package
* d0dbc16 Add license, readme
```

After

```
* 0681891 (HEAD -> master) Add operator "*"
* 03b9ab3 Add operator "*" unit test
* 5ed5048 Add evaluator module, operator "+"
* e77e2a0 Add operator "+" unit test
* 24784c8 Add gitignore
* fe22dec Add args Python3 package
* d0dbc16 Add license, readme
```

Table of content

- Reorder commit adding `.gitignore`
- Rename `test_sum` to `test_operators`
- Add unit tests for operator "*"
  - Check that tests fails
- Implement operator "*"
- Reorder commits for TDD
- Add docstrings for “+” and “*” unit tests

New Git commands

```
git rebase -i IDENTIFIKATOR-ZMENY
git mv STARE-JMENO-SOUBORU NOVE-JMENO-SOUBORU
git commit --amend

git add -p JMENO-SOUBORU
git commit -m'F IDENTIFIKATOR-ZMENY'
```

Summary and questions

Alternative history

Alternative history

What?

- We know: how to create and manage git history
- We will: time travel, emphase where feature begins and ends

Git history before

```
* 0681891 (HEAD -> master) Add operator "*"
* 03b9ab3 Add operator "*" unit test
* 5ed5048 Add evaluator module, operator "+"
* e77e2a0 Add operator "+" unit test
* 24784c8 Add gitignore
* fe22dec Add args Python3 package
* d0dbc16 Add license, readme
```

Git history after

```
*   8febb4c (HEAD -> master) Merge branch 'add-sum-prod-op'
|\
| * f8467dc Add operator "*"
| * 4e81de9 Add operator "*" unit test
| * 2eaeb68 Add evaluator module, operator "+"
| * 0715dc9 Add operator "+" unit test
|/
* 24784c8 Add gitignore
* fe22dec Add args Python3 package
* d0dbc16 Add license, readme
```

Table of content

- Time travel to commit adding `.gitignore`
- Create `add-sum-and-prod-operators` branch
- Merge `add-sum-and-prod-operators` branch

New Git commands

```
git checkout IDENTIFIKATOR-ZMENY
git branch JMENO-NOVE-VETVE
git reset --hard IDENTIFIKATOR-ZMENY
git checkout JMENO-VETVE
git merge --no-ff JMENO-VETVE
git branch -d JMENO-VETVE
```

Summary and questions

Conflicting history

Conflicting history

What?

- We know: (almost) all we need
- We will: calm down and solve conflicts

Git history before

```
*   8febb4c (HEAD -> master) Merge branch 'add-sum-prod-op'
|\
| * f8467dc Add operator "*"
| * 4e81de9 Add operator "*" unit test
| * 2eaeb68 Add evaluator module, operator "+"
| * 0715dc9 Add operator "+" unit test
|/
* 24784c8 Add gitignore
* fe22dec Add args Python3 package
* d0dbc16 Add license, readme
```

Git history after

```
*-.   3ef21b4 (HEAD -> master) Merge branches 'add-...' and 'a
|\ \
| | * 7d1e491 Implement operator ">"
| | * 676ab0f Add operator ">" unit test
| * | 715cfee Implement operator "<"
| * | 4d80713 Add operator "<" unit test
| |/
* / 250bc4e Update README with how to run tests
|/
*   8febb4c Merge branch 'add-sum-prod-op'
|\
| * f8467dc Add operator "*"
| * 4e81de9 Add operator "*" unit test
| * 2eaeb68 Add evaluator module, operator "+"
| * 0715dc9 Add operator "+" unit test
|/
```

Table of content

- New branch for operator "<" test and implementation
- Rewrite README in main branch
- New branch for operator ">" test and implementation
- Merge and solve conflicts – merge is the goal!

New Git commands

```
git branch JMENO-NOVE-VETVE IDENTIFIKATOR-ZMENY
git merge JMENO-PRVNI-VETVE JMENO-DRUHE-VETVE
git merge --continue
git branch -d JMENO-PRVNI-VETVE JMENO-DRUHE-VETVE
```

Summary and questions

Rather rebase (those) conflicts

Rather rebase (those) conflicts

What?

- We know: how to resolve conflicts
- We will: learn the rebase way to do that

Git history before

```
*-.   3ef21b4 (HEAD -> master) Merge branches 'add-...' and 'a
|\ \
| | * 7d1e491 Implement operator ">"
| | * 676ab0f Add operator ">" unit test
| * | 715cfee Implement operator "<"
| * | 4d80713 Add operator "<" unit test
| |/
* / 250bc4e Update README with how to run tests
|/
*   8febb4c Merge branch 'add-sum-prod-op'
|\
| * f8467dc Add operator "*"
| * 4e81de9 Add operator "*" unit test
| * 2eaeb68 Add evaluator module, operator "+"
| * 0715dc9 Add operator "+" unit test
|/
```

Git history after

```
*   f44762e (HEAD -> master) Merge branch 'add-non-incr-op'
|\
| * f7e1afd Implement operator ">"
| * 1d15dbc Add operator ">" unit tests
|/
*   0239969 Merge branch 'add-non-decr-op'
|\
| * 819c3f8 Implement operator "<"
| * 31d0867 Add operator "<" test
|/
* 8b51a37 Add how to run unit tests to readme
*   275f11c Merge branch 'add-sum-and-prod-operators'
|\
| * 84b563e Implement operator "*"
| * 4567aef Add operator "*" tests
| * 4a18cf5 Add evaluator and operator "+"
```

Table of content

- New branch for the last commit of “<” addition
- New branch for the last commit of “>” addition
- Reset the main branch to the commit updating `README`
- Rebase “<” branch
- Then merge “<” branch
- Then rebase “>” branch
  - And fix the conflicts
- Finally, merge “>” branch

Summary and questions

Commands used

Commands used

What?

- We know: everything now
- We will: try to not forget

Table of content

- Repository initialization
- Adding files, repository exploration
- Editor and alias setup
- Reordering and fixing the history
- Alternative history (branch)
- Conflicting history

Repository initialization

```
git init
git config user.email 'foo@bar.buzz'
git config --global user.name 'Foo Bar'
```

Adding files, repository exploration

```
git status
git add FILENAME
git diff --cached
git commit
git config --global core.editor 'vim'
git log
git show
git log --oneline --graph --decorate --all
```

Editor and alias setup

```
git config --global --add alias.logg \
    'log --oneline --graph --decorate --all'
git logg

python3 -m unittest discover tests
git show HEAD^
git commit -m'When applied, this commit will ... (max. 52)'
```

Reordering and fixing the history

```
git rebase -i COMMIT-IDENTIFIER
git mv OLD-FILENAME NEW-FILENAME
git commit --amend

git add -p FILENAME
git commit -m'F COMMIT-IDENTIFIER'
```

Alternative history (branch)

```
git checkout COMMIT-IDENTIFIER
git branch NEW-BRANCH-NAME
git reset --hard COMMIT-IDENTIFIER
git checkout BRANCH-NAME
git merge --no-ff BRANCH-NAME
git branch -d BRANCH-NAME
```

Conflicting history

```
git branch NEW-BRANCH-NAME COMMIT-IDENTIFIER
git add -p
git merge --no-ff FIRST-BRANCH-NAME SECOND-BRANCH-NAME
git merge --continue
git branch -d FIRST-BRANCH-NAME SECOND-BRANCH-NAME
```

Summary and questions

- Repository initialization
- Adding files, repository exploration
- Editor and alias setup
- Reordering and fixing the history
- Alternative history (branch)
- Conflicting history

- Why to be pedant about a patchset?