



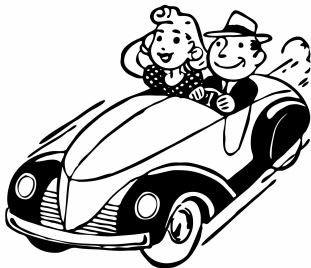
Software Engineering Basics

Martin Thoma

14. August 2018



Software is written by people with different backgrounds and strengths. Not everybody has a Software Engineering background. Those slides should help you to get the basics.



- ▶ Project: Build self-driving car
- ▶ Alice is in the US, Bob in Germany



📁 bin

📁 docs

📁 awesome_project

📁 tests

📄 setup.py

📄 tox.ini

+ \$ grep -rni foobar

Details on [my blog](#).

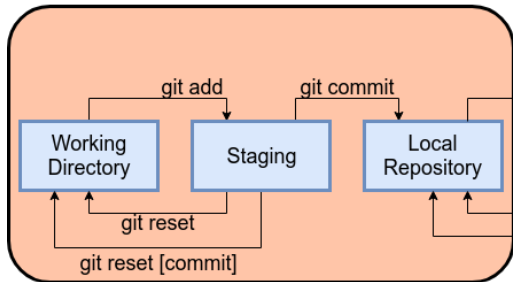


1. `$ git clone repository.git`
2. `$ git add filename`
3. `$ git commit`
4. `$ git push`

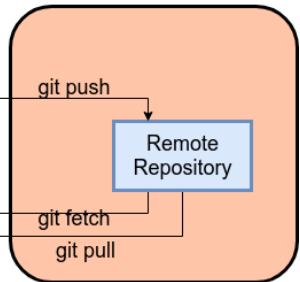


1. `$ git clone repository.git`
2. `$ git add filename`
3. `$ git commit`
4. `$ git push`

Local Machine



Remote Machine





[master] 3d131ee feature 1 - Alice <alice@xyz.com>

[master] f2e99ab feature 2 - Bob <bob@xyz.com>

[master] bb32da6 feature 3 - Alice <alice@xyz.com>

[master] 354b9f1 feature 4 - Alice <alice@xyz.com>



```
commit 204f2c82a5bd7379b66d351130917ec0012ec4fb
```

```
Merge: 175dab397 f7f6c7d4a
```

```
Author: Pauli Virtanen <pav@iki.fi>
```

```
Date: Sun Aug 12 14:23:38 2018 +0000
```

```
Merge pull request #9127 from andyfaff/disablegmpy
```

```
CI: try disabling gmpy assert
```

```
commit 175dab3978031c30c2ec7f1da641069bcca6cc4a
```

```
Merge: 1536d17cc e677a2bb8
```

```
Author: İlhan Polat <ilhanpolat@gmail.com>
```

```
Date: Sun Aug 12 13:27:44 2018 +0200
```

```
Merge pull request #9131 from akahard2dj/DOC_scipy.optimize.tutorial_typo.Quetzalcohuatl
```

```
DOC: Correct the typo in scipy.optimize tutorial page
```

```
commit 1536d17cc7a741e329e1a2a85bbe3d5fd508ebed
```

```
Merge: b87690987 9a94b31da
```

```
Author: Pauli Virtanen <pav@iki.fi>
```

```
Date: Sat Aug 11 22:40:46 2018 +0000
```

```
Merge pull request #9129 from eric-wieser/no-bare-except
```

```
BUG: Do not catch and silence KeyboardInterrupt/SystemExit
```

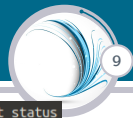
```
Bare excepts are only correct when trying to capture and forward exceptions - in all other
```

```
This only touches the code that is user-facing, not any of the tooling or tests.
```




```
582ae6d284 scipy/ndimage/filters.py (Martin Thoma 2017-04-23 18:37:59 +0200 186) >>> y6 = gaussian_filterid(x, 6)
582ae6d284 scipy/ndimage/filters.py (Martin Thoma 2017-04-23 18:37:59 +0200 187) >>> plt.plot(x, 'k', label='original data')
582ae6d284 scipy/ndimage/filters.py (Martin Thoma 2017-04-23 18:37:59 +0200 188) >>> plt.plot(y3, '...', label='filtered, sigma=3')
582ae6d284 scipy/ndimage/filters.py (Martin Thoma 2017-04-23 18:37:59 +0200 199) >>> plt.plot(y6, ':', label='filtered, sigma=6')
582ae6d284 scipy/ndimage/filters.py (Martin Thoma 2017-04-23 18:37:59 +0200 200) >>> plt.legend()
582ae6d284 scipy/ndimage/filters.py (Martin Thoma 2017-04-23 18:37:59 +0200 201) >>> plt.grid()
582ae6d284 scipy/ndimage/filters.py (Martin Thoma 2017-04-23 18:37:59 +0200 202) >>> plt.show()
ca465a651f lib/ndimage/lib/filters.py (Ed Schofield 2006-03-18 13:52:58 +0000 203) """
ca465a651f lib/ndimage/lib/filters.py (Ed Schofield 2006-03-18 13:52:58 +0000 204) sd = float(sigma)
19eff1c3fb scipy/ndimage/filters.py (Thouis (Ray) Jones 2012-06-05 10:29:22 +0200 205) # make the radius of the filter equal to truncate standard deviations
19eff1c3fb scipy/ndimage/filters.py (Thouis (Ray) Jones 2012-06-05 10:29:22 +0200 206) lw = int(truncate * sd + 0.5)
a83a5dd741 scipy/ndimage/filters.py (Jaime Fernandez del Rio 2016-03-13 21:25:34 +0100 207) # Since we are calling correlate, not convolve, revert the kernel
a83a5dd741 scipy/ndimage/filters.py (Jaime Fernandez del Rio 2016-03-13 21:25:34 +0100 208) weights = _gaussian_kernelid(sigma, order, lw)[::-1]
ca465a651f lib/ndimage/lib/filters.py (Ed Schofield 2006-03-18 13:52:58 +0000 209) return correlateid(input, weights, axis, output, mode, cval, 0)
```

git status



```
moose@pc07 ~/GitHub/LaTeX-examples/presentations/software-engineering-basics master ➤ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   Feathergraphics/1.pdf
    new file:   Feathergraphics/2.pdf
    new file:   Makefile
    new file:   beamercolorthemeFeather.sty
    new file:   beamerinnerthemeFeather.sty
    new file:   beamerouterthemeFeather.sty
    new file:   beamerthemeFeather.sty
    new file:   feather.tex
    new file:   graphics/car.jpg
    new file:   graphics/git-2-fixes.png
    new file:   graphics/git-simple.png
    new file:   graphics/gitgrpah.html
    new file:   software-engineering-basics.tex

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   ../../documents/book-minimal/book-minimal.pdf
    modified:   software-engineering-basics.tex

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

    ../../documents/letter-dsgvo/
```



We read code MUCH more often than we write it.

Problem: Dirty commit history due to bugs / fixes



[master] 5e5f8c8 feature 1 - Alice <alice@xyz.com>

[master] 1bff507 fix feature 1 - Alice <alice@xyz.com>

[master] ae89eb5 fix feature 1 - #2 - Alice <alice@xyz.com>

[master] c1bcc99 now it is really fixed - Alice <alice@xyz.com>

[master] db3bcda feature 2 - Bob <bob@xyz.com>



- ▶ PRs should not be about simple code style



- ▶ PRs should not be about simple code style
- ▶ Choose one **style guide** and stick to it



- ▶ PRs should not be about simple code style
- ▶ Choose one **style guide** and stick to it
- ▶ Trailing spaces are just noise - make your editor remove them automatically.

The Zen of Python, by Tim Peters (1)



Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.



In the face of ambiguity, refuse the temptation to guess.
There should be one— and preferably only one —obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea – let's do more of those!



- Can I understand what the commit was about?

cleanup, fixed bug, new feature,
...



- ▶ Can I understand what the commit was about?
- ▶ Use prefixes
 - ▶ BUG: bug fix
 - ▶ DEV: development tool or utility
 - ▶ DOC: documentation
 - ▶ ENH: Enhancement, a new feature
 - ▶ MAINT: Maintenance task
 - ▶ REL: release
 - ▶ STY: Stylistic change
 - ▶ TST: addition or modification of tests

Source: [Scipy Docs](#)



- ▶ Can I understand what the commit was about?
- ▶ Use prefixes
- ▶ Mention issues
- ▶ See issue #123
- ▶ Closes issue #123



```
$ git commit -amend
```

- ▶ Add to last commit / change message
- ▶ Only on feature branch as it might require force-push!

Commit squashing

Making multiple commits in a row become one

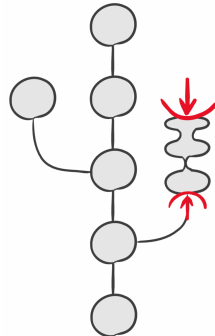


Image source:

stevenschenke.de



[master] 02ddb49 ENH: feature 1 - Alice <alice@xyz.com>

[master] cb0fc5e ENH: feature 2 - Alice <alice@xyz.com>

[master] a0e98b6 ENH: feature 3 - Alice <alice@xyz.com>

[feature] d8ff19b ENH: feature 4 - Alice <alice@xyz.com>

[feature] 166f7b5 ENH: feature 4 - fix 1 - Alice <alice@xyz.com>

[feature] 71fcfc0 ENH: feature 4 - fix 2 - Alice <alice@xyz.com>

[master] b9248de Merge branch 'feature' into 'master' - Sergio Flores <saxo-guy@epic.com>

git merge vs git rebase

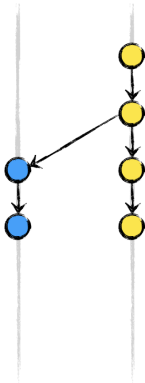


19

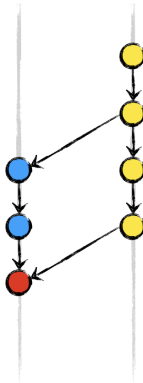
`$ git merge develop`

`$ git rebase develop`

feature/login develop



feature/login develop



feature/login develop

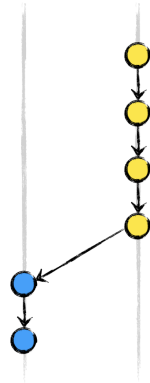


Image source: [Jeff Kreeftmeijer](#)



```
1 from math import ceil
2
3 def f(n=1000000):
4     roundUp = lambda n, prime: int(ceil(float(n) / prime))
5
6     arr = [True] * n
7     arr[0] = False
8     arr[1] = False
9     primeList = []
10
11     for curr in range(2, n):
12         if not arr[curr]:
13             continue
14         primeList.append(curr)
15         for multiplicant in range(2, roundUp(n, curr)):
16             arr[multiplicant * curr] = False
17     return primeList
```



```
4 def round_up(n, prime):
5     return int(ceil(float(n) / prime))
6
7
8 def get_primes_below_n(n=1000000):
9     is_prime_table = [True] * n
10    is_prime_table[0] = False
11    is_prime_table[1] = False
12    prime_list = []
13
14    for current_number in range(2, n):
15        if not is_prime_table[current_number]:
16            continue
17        prime_list.append(current_number)
18        for multiplicand in range(2, round_up(n, current_number)):
19            is_prime_table[multiplicand * current_number] = False
20    return prime_list
```



```
8 def get_primes_below_n(n=1000000):
9     """
10     Get a list of all primes below n.
11
12     Parameters
13     -----
14     n : int
15
16     Returns
17     -----
18     prime_list : list
19
20     Examples
21     -----
22     >>> get_primes_below_n(10)
23     [2, 3, 5, 7]
24     """
```



(1) Floating point numbers always look like this: 1.23456 or 0.000004577 or 12345.467765.



(1) Floating point numbers always look like this: 1.23456 or 0.000004577 or 12345.467765.

- ▶ Scientific notation: 4.577E-5 or 1.2345467765E4



(1) Floating point numbers always look like this: 1.23456 or 0.000004577 or 12345.467765.

- ▶ Scientific notation: 4.577E-5 or 1.2345467765E4
- ▶ German decimal format: 1,23456 or 0,000004577



(2.1) Country names have an unique representation



(2.1) Country names have an unique representation
“Germany” vs “Deutschland”



(2.2) Country names have an unique representation in English



(2.2) Country names have an unique representation in English
“United Kingdom” vs “UK”



(2.3) Country names have an unique unabreviated representation in English



(2.3) Country names have an unique unabreviated representation in English
“United Kingdom” vs “Great Britain” vs “England”



(2.3) Country names have an unique unabreviated representation in English

Solution:

Use/Demand **ISO 3166-1 alpha-3 country codes** everywhere



(3) Data is clean



(3) Data is clean
No.



(3) Data is clean
No.

- ▶ User database: Birth date in the year 3.



(3) Data is clean

No.

- ▶ User database: Birth date in the year 3.
- ▶ User database: Active user who is more than 90 years old.



(3) Data is clean

No.

- ▶ User database: Birth date in the year 3.
- ▶ User database: Active user who is more than 90 years old.
- ▶ User database: User who is younger than 6.



(4) Time has no beginning and no end



(4) Time has no beginning and no end

Unix Time Stamp: Seconds since 1st of January, 1970. Stored in unsigned int.



(4) To avoid the Year-2038 problem, I can store YYYY-mm-dd
HH:MM:ss



(4) To avoid the Year-2038 problem, I can store YYYY-mm-dd
HH:MM:ss

► Python's strftime directives



(4) To avoid the Year-2038 problem, I can store YYYY-mm-dd
HH:MM:ss

- ▶ Python's strftime directives
- ▶ Timezones



(4) To avoid the Year-2038 problem, I can store YYYY-mm-dd
HH:MM:ss

- ▶ Python's strftime directives
- ▶ Timezones
- ▶ Whenever possible, store the timezone and use **ISO 8601**:
2012-04-23T18:25:43.511+02:30 (**reasons**)



(5) The (physical) unit of a column / an API can be guessed.



(5) The (physical) unit of a column / an API can be guessed.

- ▶ Clarify it



(5) The (physical) unit of a column / an API can be guessed.

- ▶ Clarify it
- ▶ See if the distribution / quantiles are reasonable



(5) The (physical) unit of a column / an API can be guessed.

- ▶ Clarify it
- ▶ See if the distribution / quantiles are reasonable
- ▶ Internally, use unit library **Pint**



- ▶ git
 - ▶ meld: Tool for diff and merge (`$ git mergetool`)
 - ▶ A successful Git branching model
- ▶ Debugging Python with ipdb and Sypder - starting at 4:00
- ▶ cprofile: Check where code improvements are effective
- ▶ David Goldberg: What Every Computer Scientist Should Know About Floating-Point Arithmetic
- ▶ Testing with Python
- ▶ Logging with Python
- ▶ UML: Sequence diagrams, Flow charts (e.g. Dia or draw.io)
- ▶ Balsamiq: Draft an UI
- ▶ Web: REST basics