Freie Universität Berlin

Computational Sciences Projektseminar

Introduction

Objective

- apply the contents of the lecture
- develop code in a team
- usage of tools and programming techniques

Tasks

- individual and group exercises
- development of a large project
- presentation and application

Schedule

- **Tuesdays, 10:15–11:45, room 017/A6**
  14 meetings for general discussion and lectures

- **group-based extra meetings for specific problems**

- **email support**

Language

`https://www.python.org`

- **Python (anaconda) and C/C++**
- **useful libraries: numpy, scipy, pytorch, …**
- **easy to distribute (packaging, user base)**

`http://conda.pydata.org/miniconda.html`

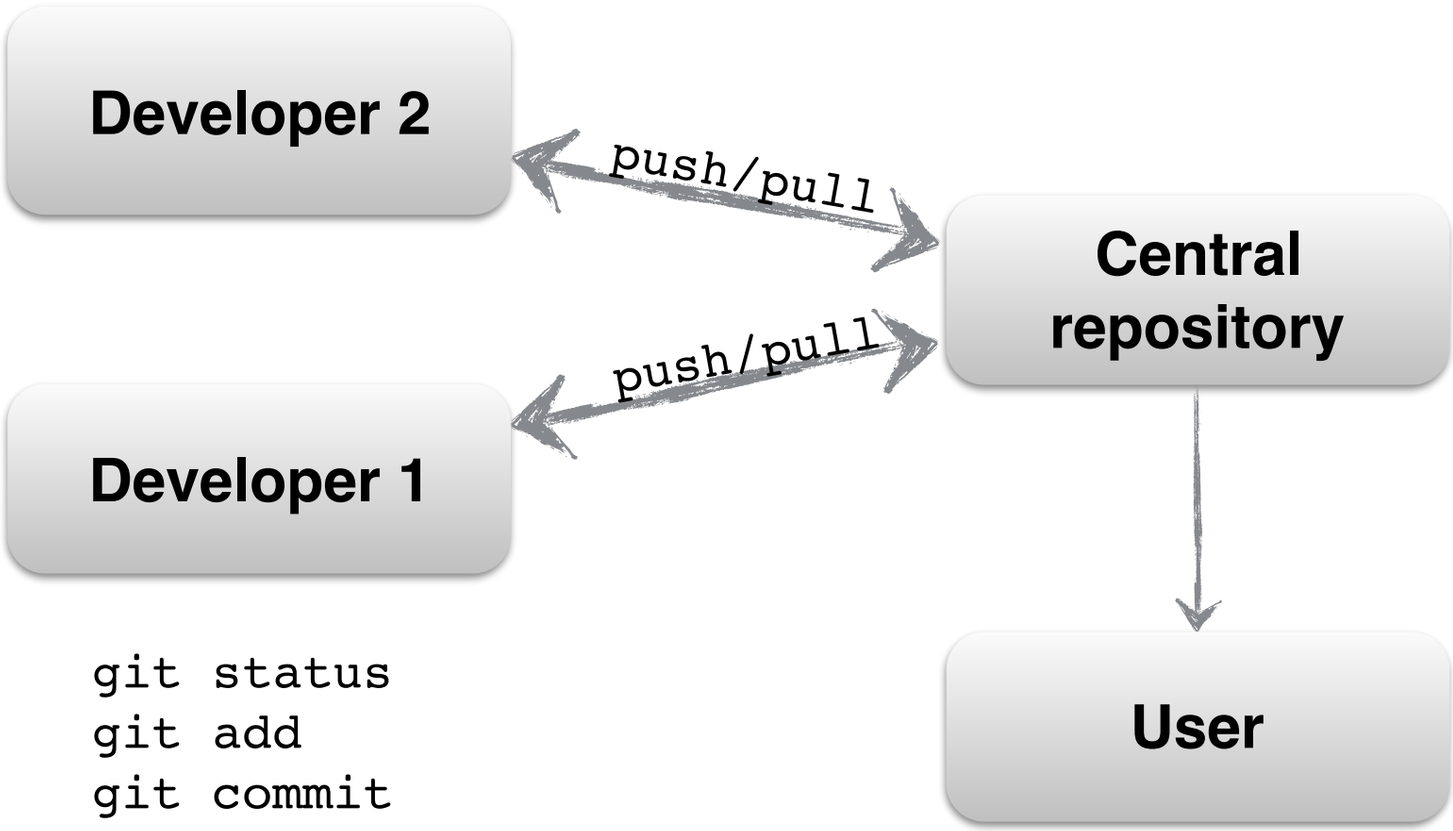
Jupyter notebook (IPython)

`http://jupyter.org`

- **live code + equations + visualisations + explanatory text**
- **fast prototyping**
- **applications, tutorials, examples**

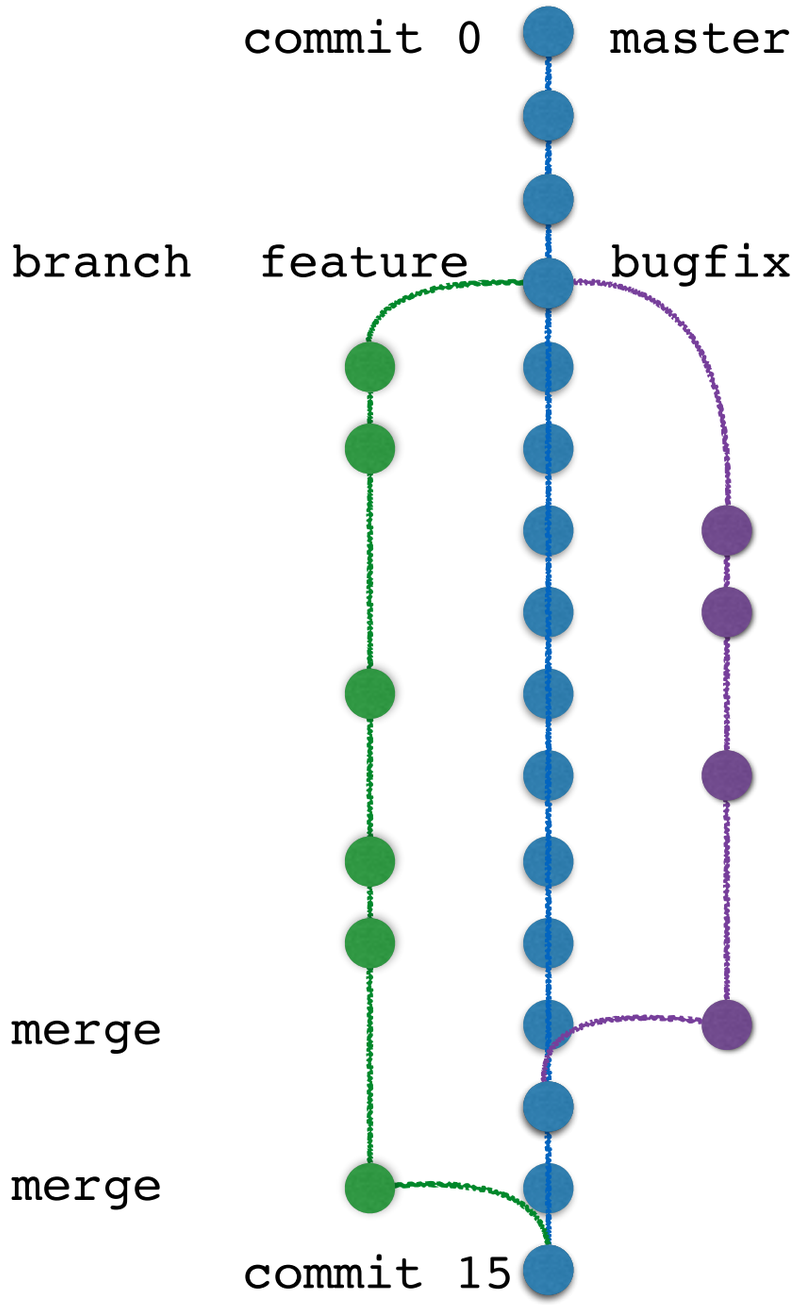
git (version control) & github (social coding)

https://github.com

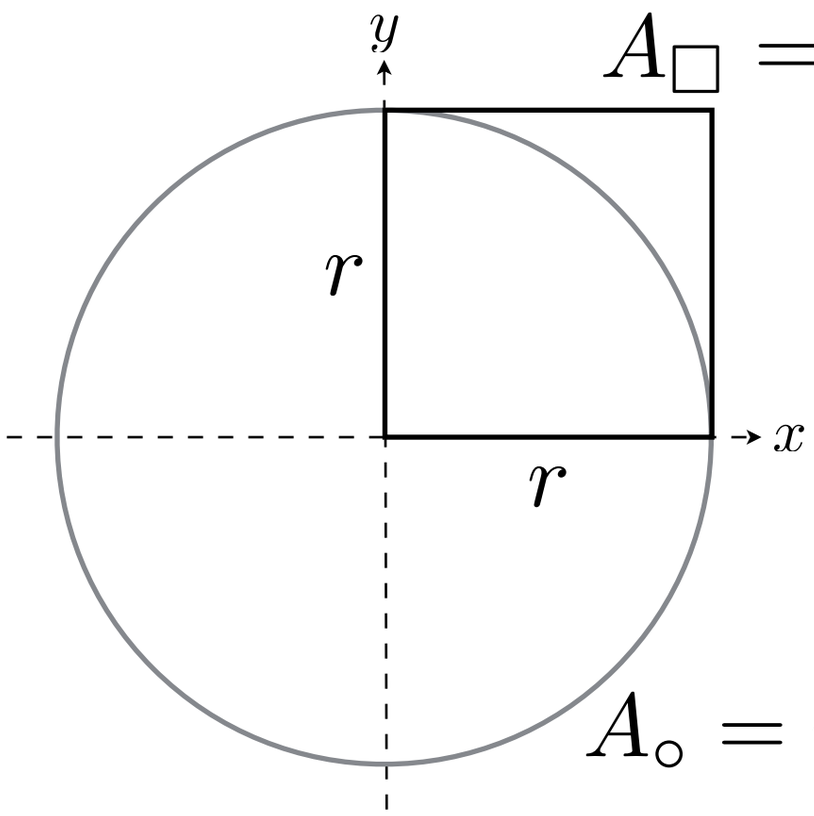



```
git status
git add
git commit
git branch
git merge
git rebase
```

```
git clone
git pull
```



Exercise: approximate π



$$A_\square = r^2$$

$$A_\circ = \pi r^2$$

$$\frac{\frac{1}{4}A_\circ}{A_\square} = \frac{\frac{1}{4}\pi r^2}{r^2} = \frac{\pi}{4}$$

$$\chi(x,y) = \begin{cases} 1, & x^2 + y^2 \leq r^2 \\ 0, & else \end{cases}$$

$$\frac{\frac{1}{4}A_\circ}{A_\square} \approx \frac{1}{N}\sum_{n=0}^{N-1} \chi(x_n, y_n)$$

$$(x_n, y_n) \in [0, r]^2 \; \forall n$$

Exercise: approximate π

- **choose** $r = 1$
- **generate N=10^6 random tuples** $(x_n, y_n) \in [0, 1]^2$
- **compute** $\pi_{\text{sampled}} = \frac{4}{N} \sum_{n=0}^{N-1} \chi(x_n, y_n)$

`https://github.com/markovmodel/compsci-2017`

Exercise: approximate pi — solution

```python
def approximate_pi_naive(n_sample):
    summation = 0
    for i in range(n_sample):
        x, y = np.random.rand(2)
        if x * x + y * y <= 1.0:
            summation += 1
    return 4.0 * summation / float(n_sample)


def approximate_pi_vectorized(n_sample):
    xy = np.random.rand(n_sample, 2)
    rr = np.sum(xy**2, axis=1)
    idx = np.where(rr <= 1.0)[0]
    return 4.0 * len(idx) / float(n_sample)
```