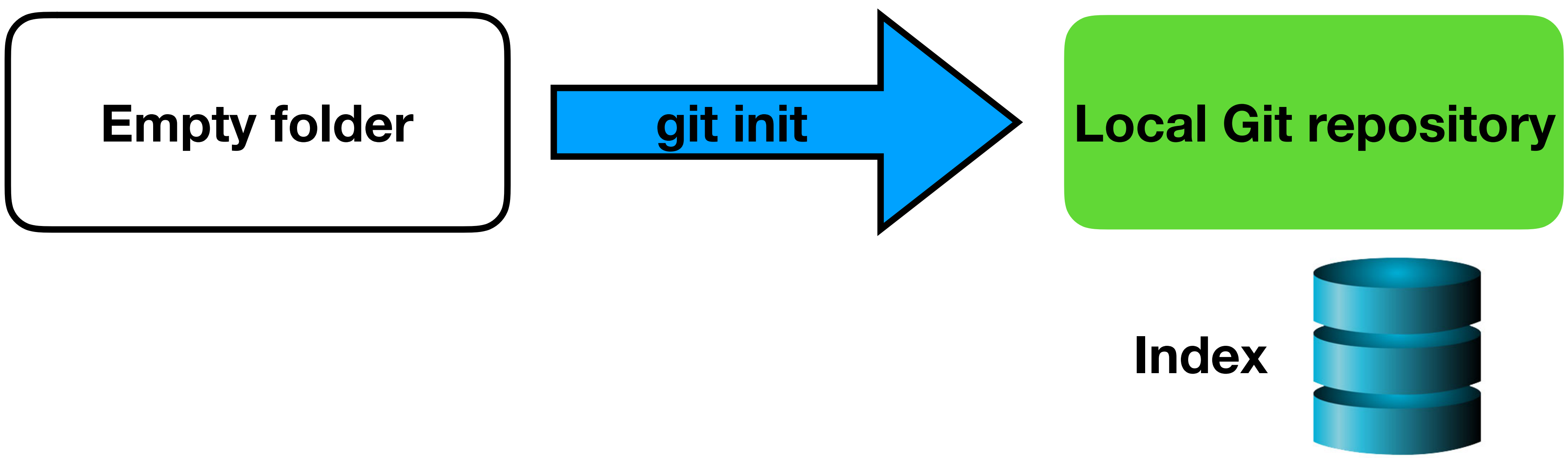
Introduction to Scientific Computation
Lecture 1
Fall 2018

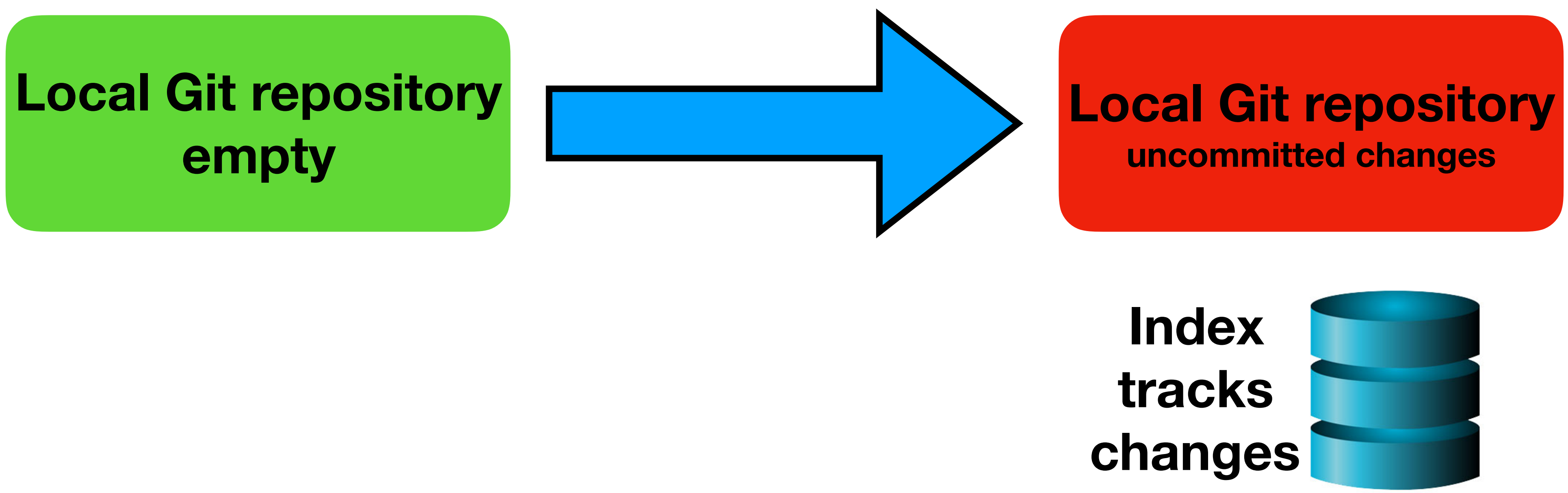
Git v2, Complexity, Floating-point arithmetic
Numerical stability

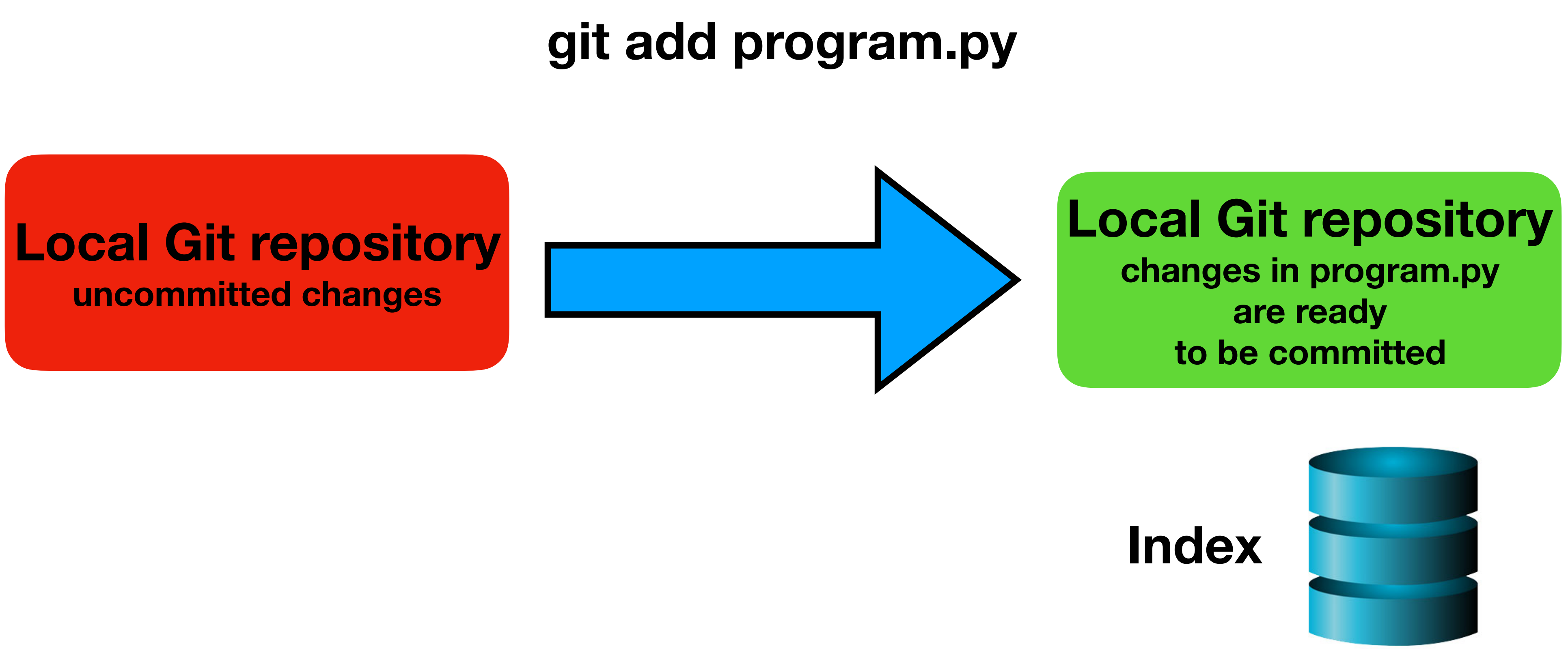
Mikhail Usvyatsov September 26, 2018

Git v2

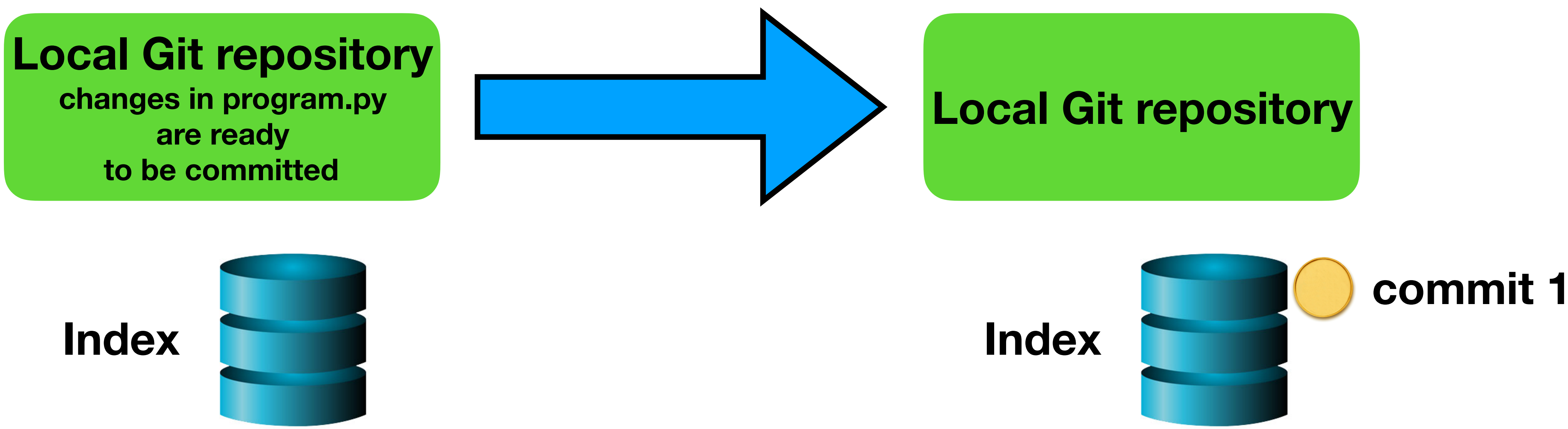


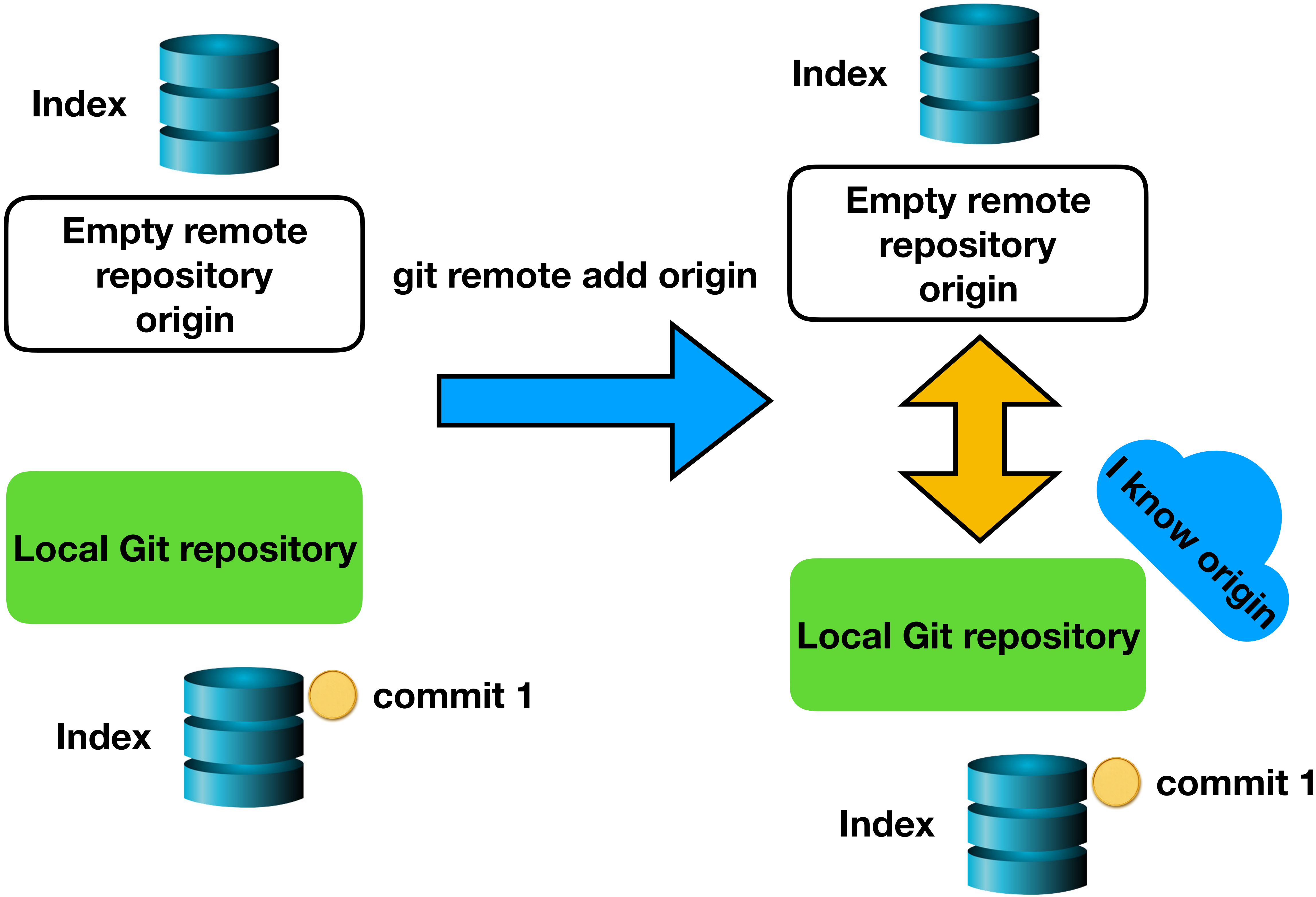
Create program.py consisting of K lines

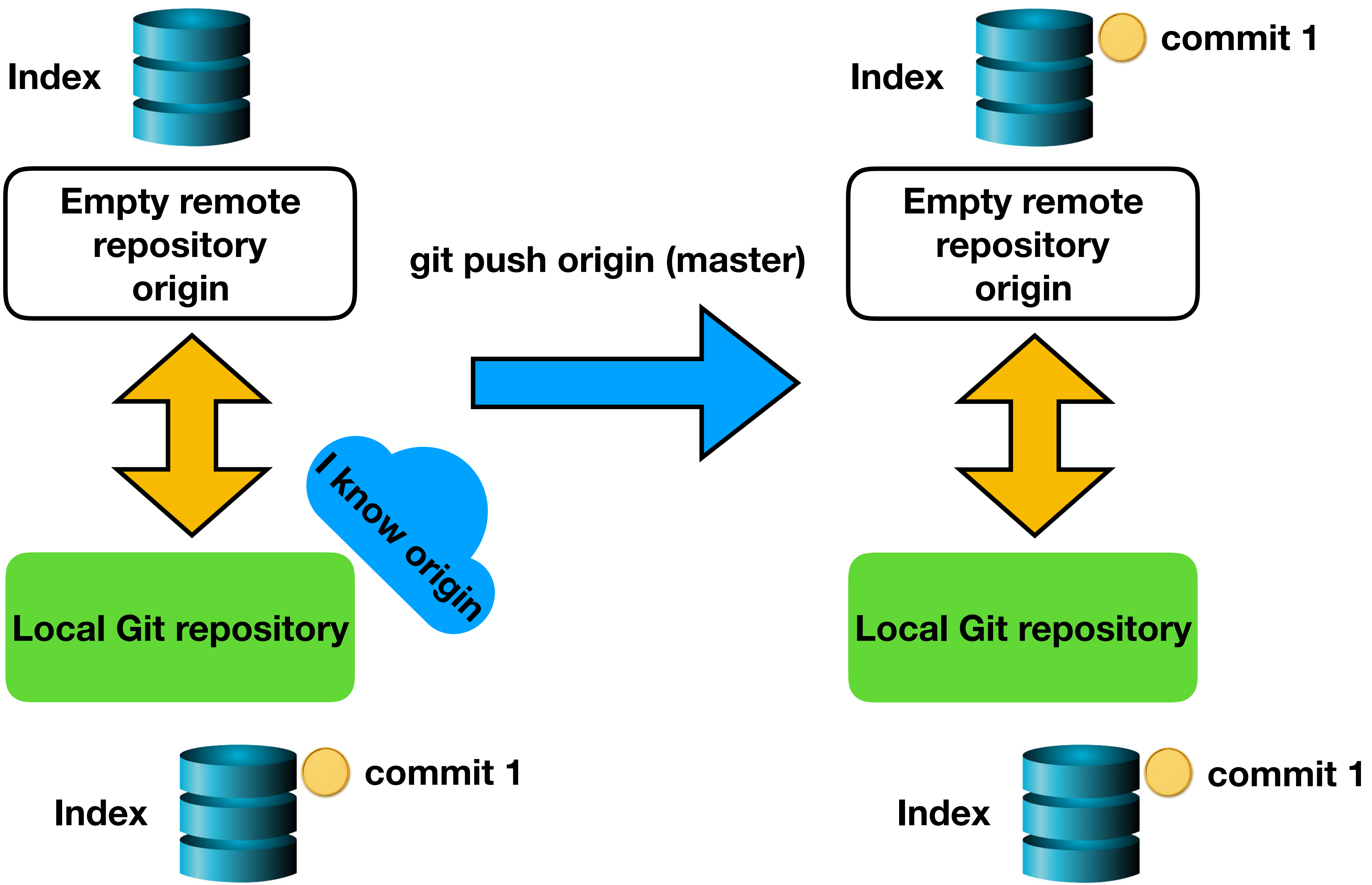


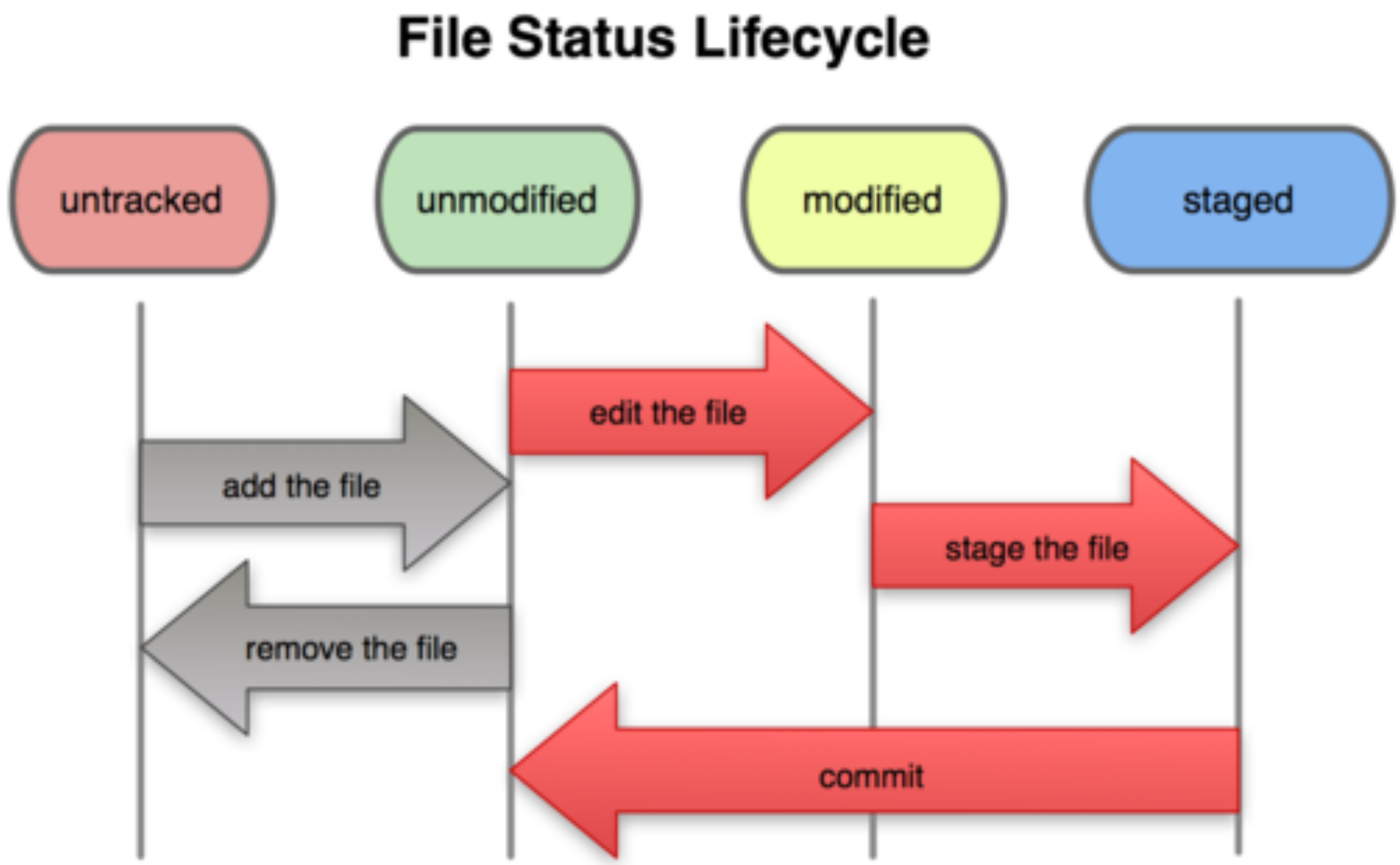

git add program.py
Local Git repository
uncommitted changes
Local Git repository
changes in program.py
are ready
to be committed
Index

git commit -m “implemented program”

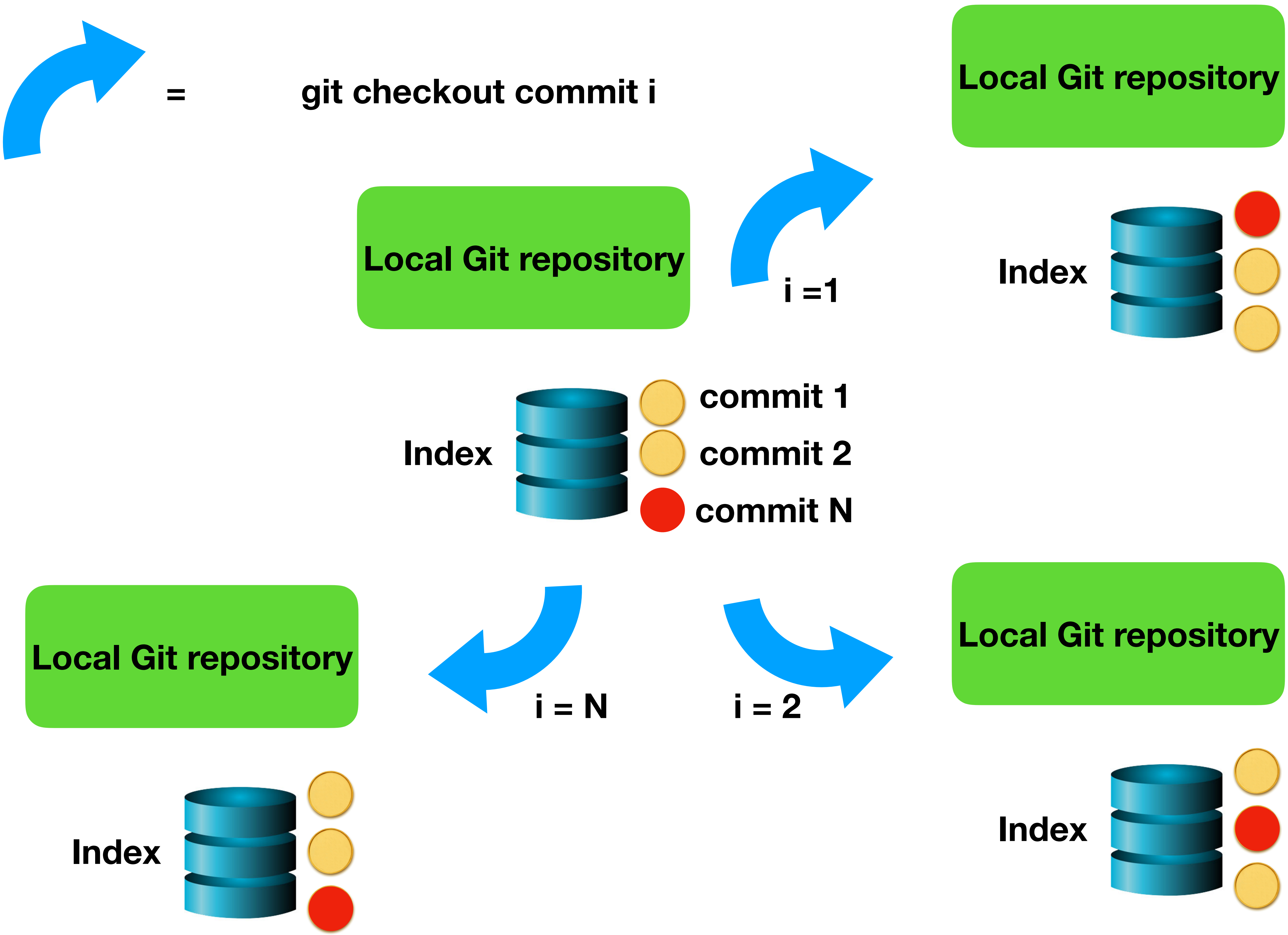



Index
Empty remote repository origin
git remote add origin
Index
Empty remote repository origin
I know origin
Local Git repository
Local Git repository
commit 1
Index
commit 1
Index


Index
Empty remote repository origin
Local Git repository
I know origin
Index
commit 1
git push origin (master)
Index
commit 1
Empty remote repository origin
Local Git repository
Index
commit 1



(c) http://git-lectures.github.io


=
git checkout commit i
Local Git repository
Index
commit 1
commit 2
commit N
i =1
Local Git repository
Index
i = N
Local Git repository
Index
i = 2
Local Git repository
Index

Check here for more examples:

http://git-lectures.github.io

Complexity

One of definitions: it is a measure of computational cost
In order to give a more formal definition quite a big formalism should be introduced (see materials for this week).
We will operate with a concept.

Complexity

The notion of complexity is about how good or bad is the algorithm.
We want to think in an abstract way, independent of:

- implementation details (Python, C++, matlab)
- hardware (stm32, intel Pentium D, Intel Core i7-8700K)
- etc and whatnot

Complexity

Let us operate constant time operation (elementary operations).

E.g. + and *

Remember: real runtime doesn't matter (doesn't it ?)

$$3 + 5 = \begin{array}{r} 0011 \\ +\ 0101 \\ \hline 1000 \end{array}$$

Complexity

Let us operate constant time operation (elementary operations).

E.g. + and *

Remember: real runtime doesn't matter (doesn't it ?)

$$3 + 5 = \begin{array}{r} 0011 \\ +\ 0101 \\ \hline 1000 \end{array}$$

$T(n) = c \cdot n$ c **- is a cost for summation of one bits**

Complexity

Let us operate constant time operation (elementary operations).

E.g. + and *

Remember: real runtime doesn't matter (does it ?)

$$3 + 5 = \begin{array}{r} 0011 \\ +\,0101 \\ \hline 1000 \end{array}$$

$T(n) = c \cdot n$

c - is a cost for summation of one bits

c_1 - for cpu1

c_2 - for cpu2

ETH zürich

Complexity

In order to get rid of c we will use **Asymptotic complexity** with big-O notation.

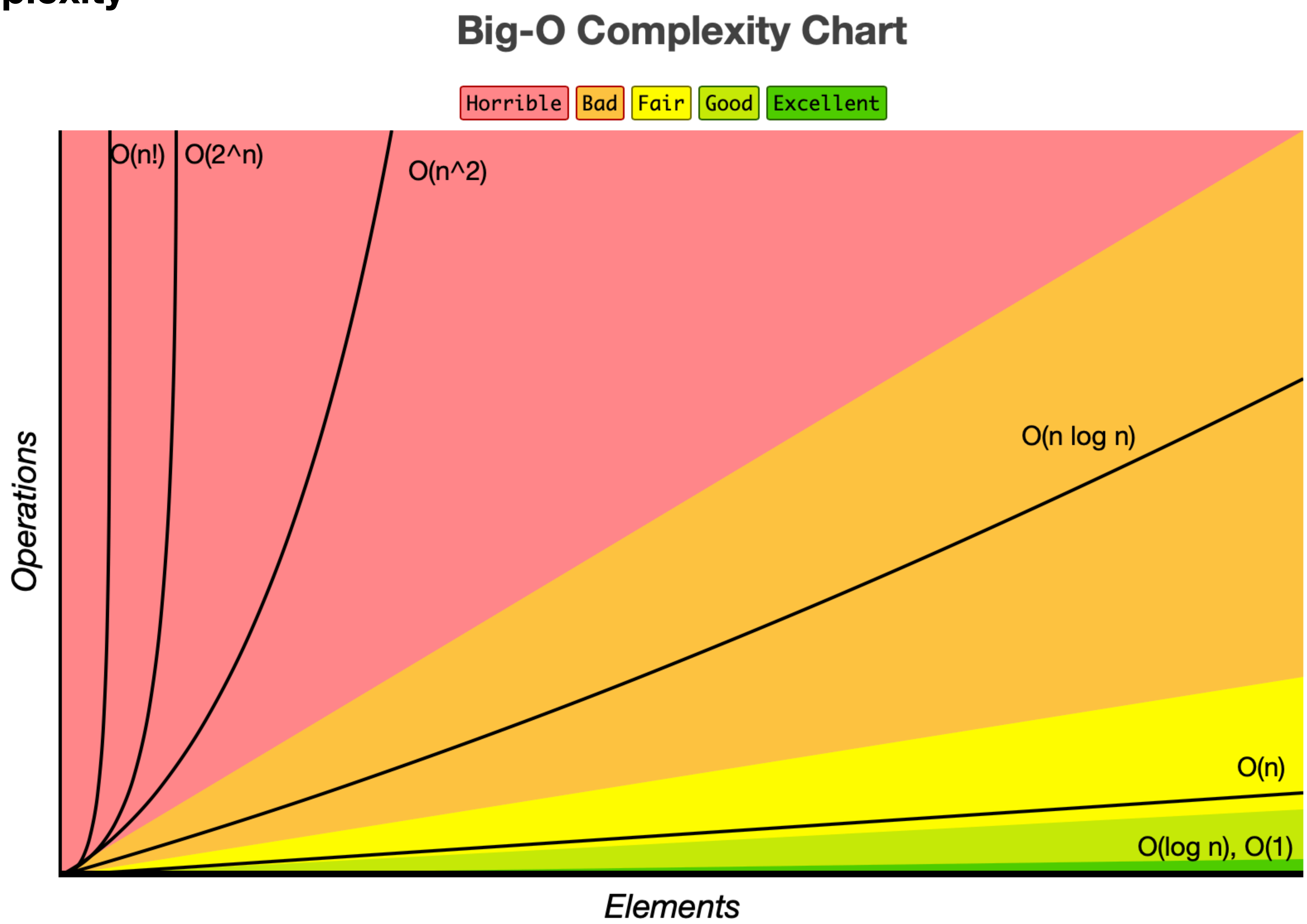
Complexity

In order to get rid of c we will use **Asymptotic complexity** with big-O notation.

Definition: for any monotonic functions $f(n)$ and $g(n)$ from positive integers to the positive integers $f(n) = O(g(n))$ when there exists constants $c > 0$ and $n_0 > 0$ such that

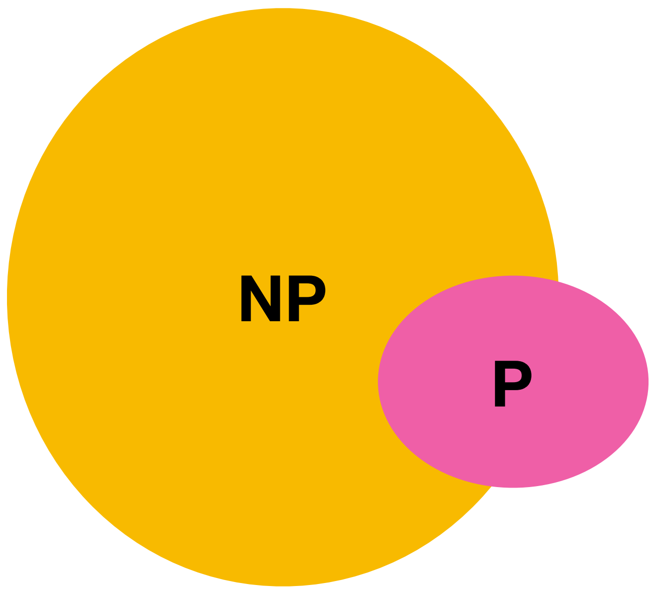
$$f(n) \leq c \cdot g(n), \textbf{ for all } n \geq n_0$$

Complexity



(c) http://www.bigocheatsheet.com

Complexity



???

Representation of numbers

Fixed point

The most straightforward format for the representation of real numbers is **fixed point** representation, a.k.a **Qm.n** format.

Qm.n number is in the range $[-2^m, 2^m - 2^{-n}]$ with the resolution 2^{-n} requires m + n + 1 bits for storage.

Representation of numbers

Floating point

We are mostly interested in **floating point numbers** represented as e.g.

$$1.2345 = \underbrace{12345}_{\textbf{significand}} \times \underbrace{10}_{\textbf{base}}{}^{\overbrace{-4}^{\textbf{exponent}}}$$

IEEE 754

Single and double precision

$$X = \pm 1.\overline{b_1 b_2 \ldots b_K} \cdot 2^e$$

$$X \in [X - \Delta X, X + \Delta X],$$

Single

```
S EEEEEEEE FFFFFFFFFFFFFFFFFFFFFFF
0 1      8 9                    31
```

Double

```
S EEEEEEEEEEE FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
0 1        11 12                                                63
```

Absolute accuracy

$$\Delta X \leq \frac{1}{2} 2^{-K} * 2^e \leq |X| \cdot 2^{-K-1}$$

Relative accuracy

$$\frac{\Delta X}{X} \leq 2^{-K-1}$$

The **relative accuracy** of single precision is $10^{-7} - 10^{-8}$, while for double precision is $10^{-14} - 10^{-16}$.
A **float32** takes **4 bytes**, **float64**, or double precision, takes **8 bytes.**
These are the only two floating point-types supported in hardware.
You should use **double precision** in CSE and **single** on GPU/Data Science.

Format	Total bits	Significand bits	Exponent bits	Smallest number	Largest number
Single precision	32	23 + 1 sign	8	ca. 1.2 · 10^{-38}	ca. 3.4 · 10^{38}
Double precision	64	52 + 1 sign	11	ca. 5.0 · 10^{-324}	ca. 1.8 · 10^{308}

Machine zero exists

```
import numpy as np

n = 1.0
d = np.inf
i = 0

while d > 0:
    d = n - n / 2
    n = n / 2
    i += 1

(1 / 2)**(i - 2)
```

```
5e-324
```

```
import numpy as np

n = 1.0
d = np.inf
i = 0

while d > 0:
    d = n - n / 2
    n = n / 2
    i += 1

(1 / 2)**(i - 1)
```

```
0.0
```

See code