



Introduction to Scientific Computation

Lecture 0

Fall 2018

The rules of the game
Intro

General course info

Lectures can be downloaded and viewed on Github:

<https://github.com/Aelphy/ISC2018>

There you can find information on how to prepare the environment, launch the code, submit assignments, communicate with staff etc.

Team & TAs:

Mikhail Usvyatsov

Communication system

Communication with staff (questions, suggestions, etc) could be done via emails

mikhailu@ethz.ch. Please make sure that the subject in your email is [ISC2018].

Telegram chatroom for ETHZ students is here: <https://t.me/icsfall18>

Plagiarism

Students are encouraged to discuss the material, but all the assignments should be done individually. Assignments will be checked for plagiarism. If noticed, can be punished in various ways from zeroing out some parts of a specific assignment to unsatisfactory grade for the course.

Assignments

This course uses Github for submitting assignments. You should create a dedicated **private!** repository with name **ICS2018-solutions**. E.g. github.com/YOUR_NAME/ICS2018-solutions. And add me (@Aelphy) to collaborators.

Assignment becomes submitted after sending email to mikhailu@ethz.ch with the subject [ICS2018: assignment_N], where N is the number of assignment. In the body of this email there should be a link to a dedicated folder for this particular assignment.

For the first assignment it would be:

github.com/YOUR_NAME/ICS2018-solutions/assignment_0

Deadlines & Attendance

Unless stated otherwise, submission deadline is 00:01 of 10's day after handing out.
You lose 10% of initial amount of points every day until submission.

By the time of grading the assignment will be checked in the state of the last commit before the deadline.

Assignment can't yield negative amount of points, even if submitted 2 years past deadline.

Attendance is not strict, but do not disappoint us.

Grading

The course grade depends entirely on points obtained mostly by completing homework assignments. There will be no course exam at the end.

Some assignment examples are "Implement and optimize factorial computation algorithm", "Implement gradient descent algorithm for logistic regression", "Compare Ridge and Lasso regression for house prices problem", "Implement Perceptron Algorithm"

Materials

The Hitchhiker's Guide to Python

The Primer on Scientific Programming With Python. H. Langtangen

Always use

Google

StackOverflow

Materials

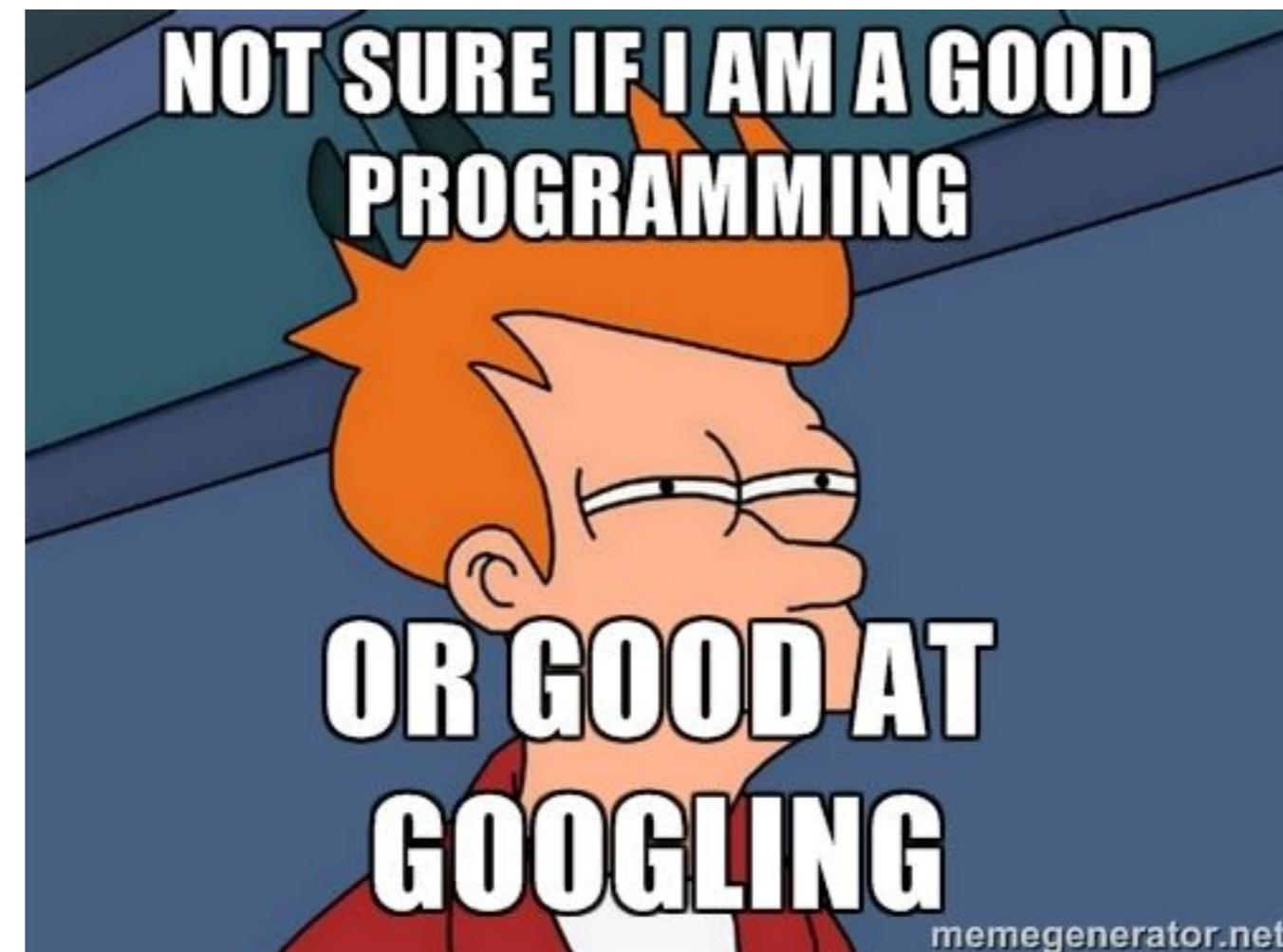
The Hitchhiker's Guide to Python

The Primer on Scientific Programming With Python. H. Langtangen

Always use

Google

StackOverflow



Introduction

How is your code running ?

How is your code running ?



memegenerator.net

Semantic Gaps

Application domain

Train traffic management system:

Trains, velocity, distance, railways, switches, train stations, conflict resolution, time schedule etc.

Simulation of the Universe evolution:

Planet movement rules, planet systems, stars classification, star activity etc.

...

(c) Copyright 2017, E.Zouev

Computer program/ programming language

- Variable, array, ...
- Function, operator, procedure
- Control structures
- Data types
- Module, class, interface
- ...

Computer hardware

- Memory cell
- Memory address
- Register
- Instruction, instruction set
- ...

Semantic Gap

Computer program/ programming language

- Variable, array, ...
- Function, operator, procedure
- Control structures
- Data types
- Module, class, interface
- ...



Computer hardware

- Memory cell
- Memory address
- Register
- Instruction, instruction set
- ...

So, the main compiler task is:

- To map human-written & human-friendly program to computer hardware - to solve a problem

OR:

- To overcome the semantic gap.

Machine Code & Assembly Language

Off-topic

Mnemonic notation:

ADD 5 7



ADD i j

- The ADD instruction denotes the two's complement arithmetic addition. The contents of registers Ri and Rj are arithmetically added, and the result is put into the register Rj.
- The instruction format is as follows:



Binary code of the instruction:

1101010010100111



Hexadecimal code of the instruction:

D4A7

Assembler code:

.format 32



R5 += R7

Suggested assembly statement for the ADD instruction:

Rj += Ri;

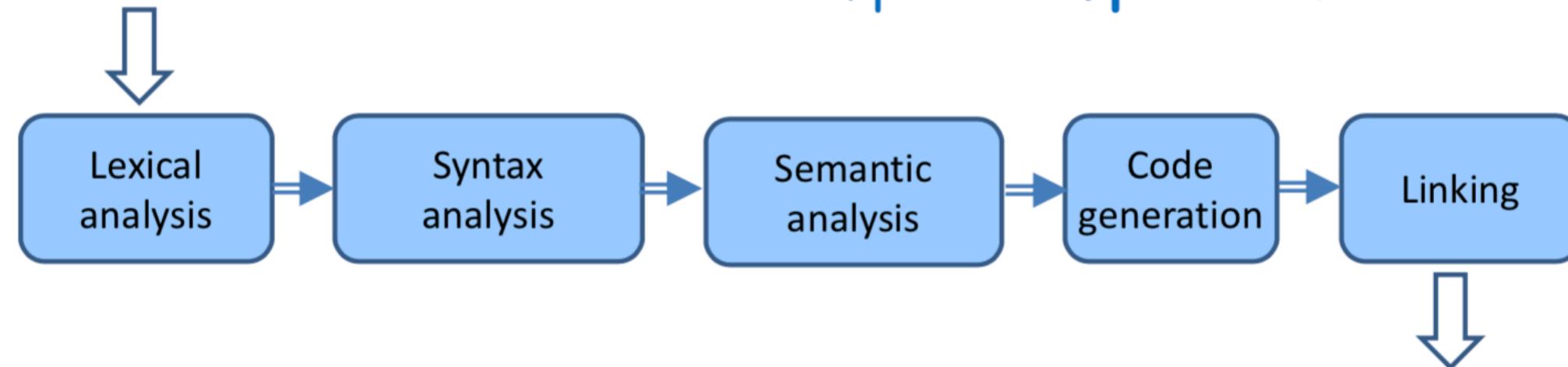
Additional assembly directives specifying the current instruction format:

.format 8; or .format 16; or .format 32;

Compilation: An Ideal Picture

*A program written by a human
(or by another program)*

Source program
text



Blue squares just denote some actions typical to any compiler; they are not necessarily actual compiler components.

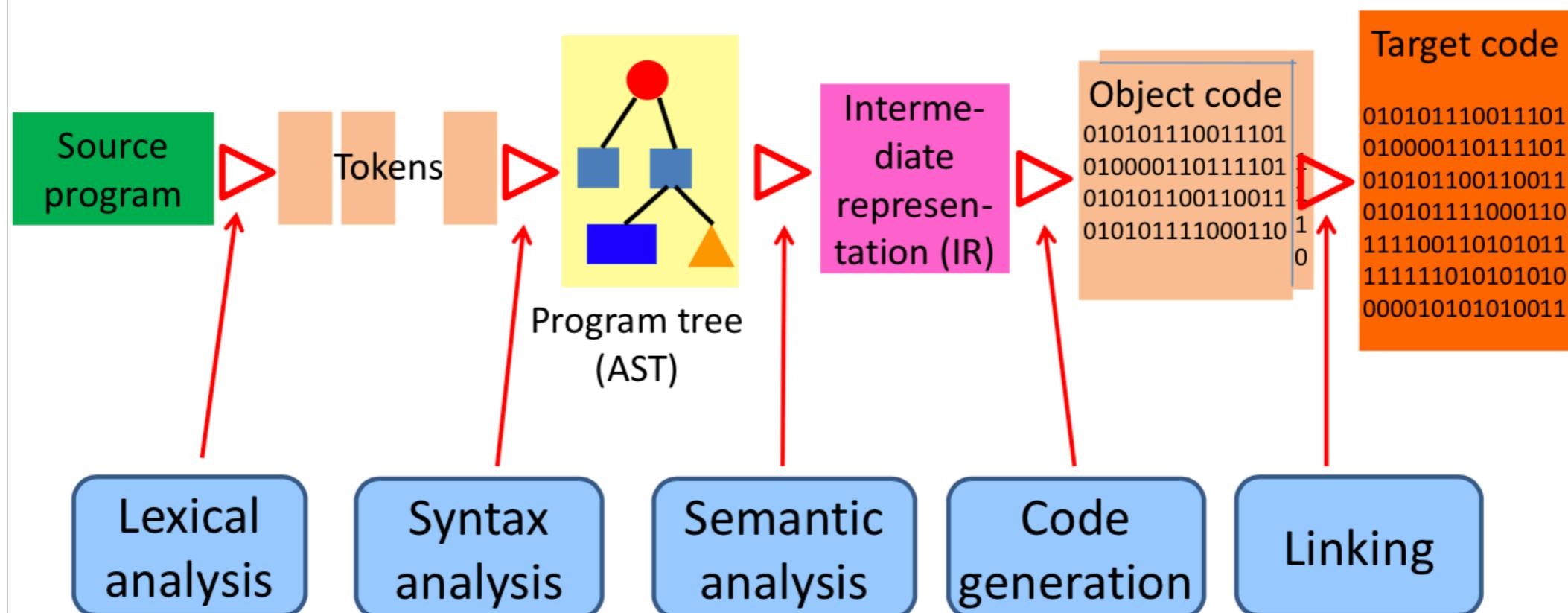
The contents of those actions, their implementation technique, and how they interact with other actions - is just the subject of the course.

A program binary image suitable for immediate execution by a machine

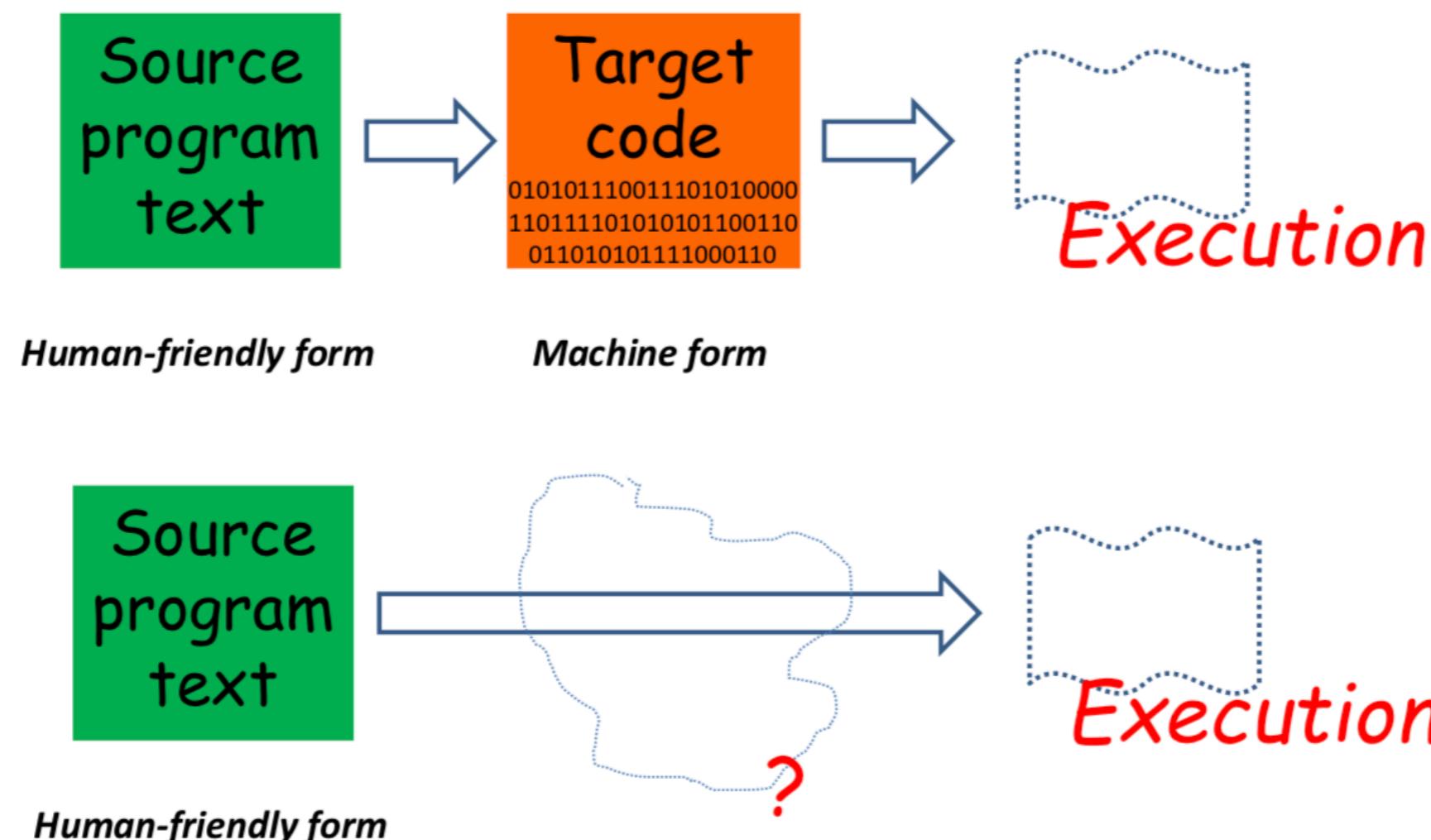
Target code
01010111001110101000
01101111010101011001
10011010101111000110

Compilation Data Structures

This is a different, **data-centric** view at the compilation process. Compilation is a sequence of transformations of a source program.



Compilation vs Interpretation



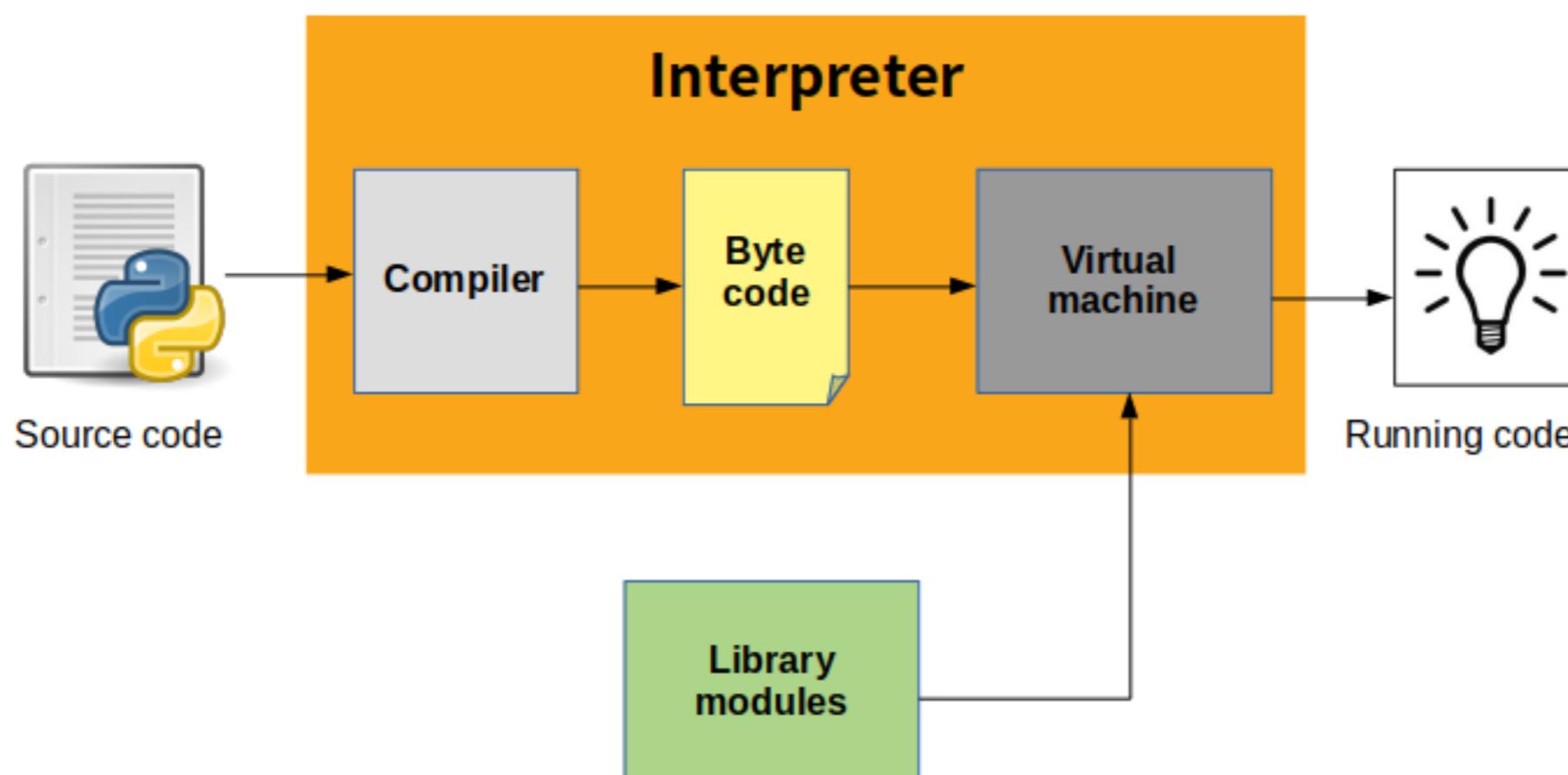
(c) Copyright 2017, E.Zouev

So, what about Python ?

From programmer point of view



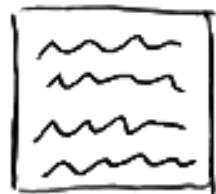
From Python point of view



So, what about Python ?

Source code:

hello.c



→ COMPILER →

Machine code:

11010
1011
10001

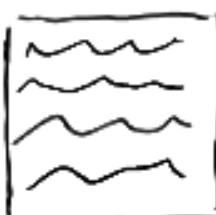
Program (also
called binary,
executable ...)

run the
program
result



Source code:

hello.py



→ INTERPRETER → result



How does Python interpret this:

$$X + Y * Z + U$$

How does Python interpret this:

$X + Y * Z + U$



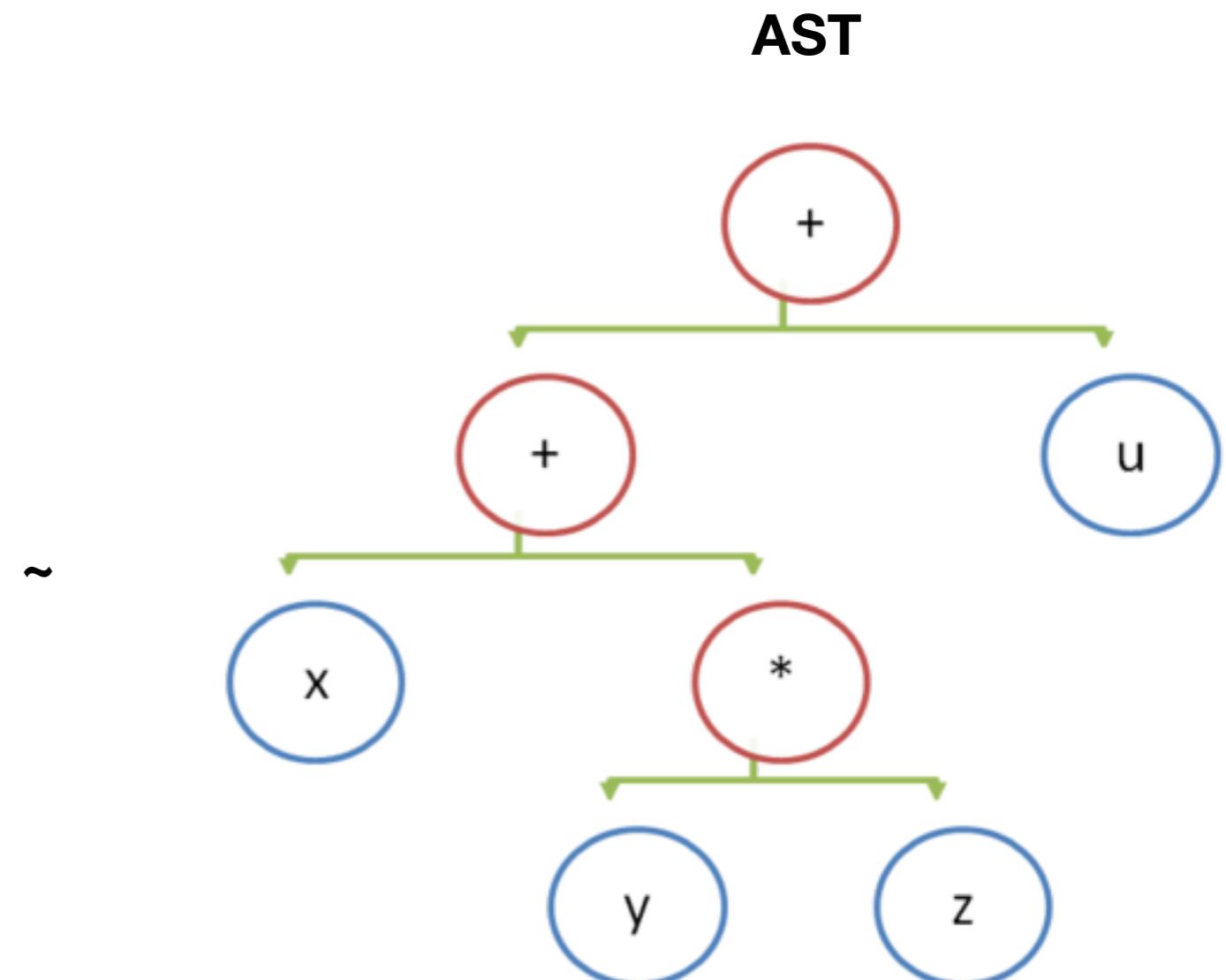
Stack

```
push x  
push y  
push z  
multiply  
add  
push u  
add
```

How does Python interpret this:

Stack

```
push x  
push y  
push z  
multiply  
add  
push u  
add
```



Python

Python



Python

Pros:

- **Extensive Support Libraries**
- **Can use C/C++ libs via SWIG**
- **Cross platform**
- **Relatively low entry level**
- **Free & open-sourced**
- **Huge community**

Cons:

- **Extensive Support Libraries**
- **Slower than C/C++, assembly**
- **Runtime errors**
- **Parallel(GIL)**

Python environments

pip - Pip Installs Packages - package manager for python. Can be used together with conda but very carefully

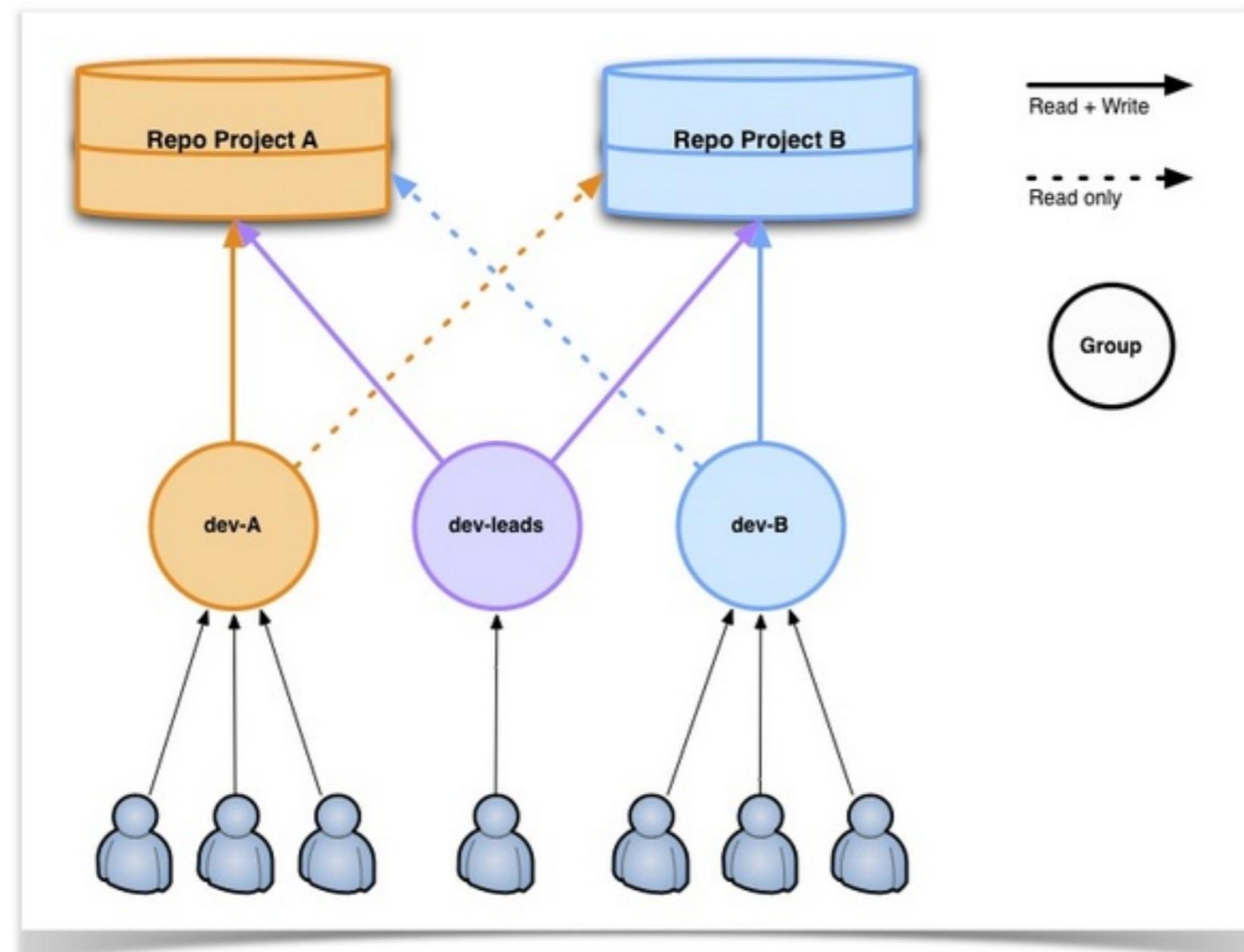
If you use python without anaconda:

- virtualenv, source activate, pip install

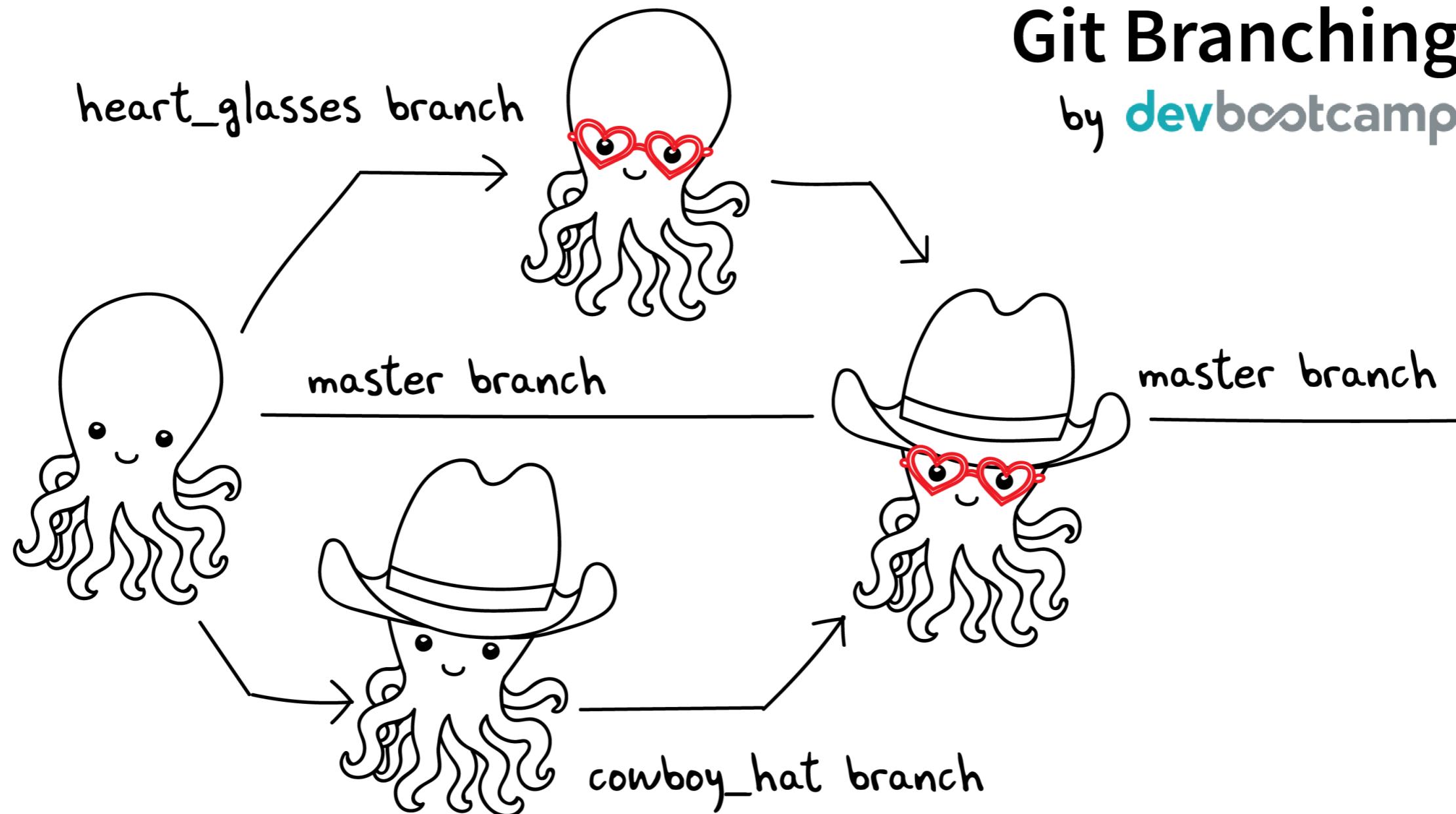
If you use anaconda:

- conda activate, conda install + pip install

Git: repositories



Git: branches



Git: white board slide



Let's code

