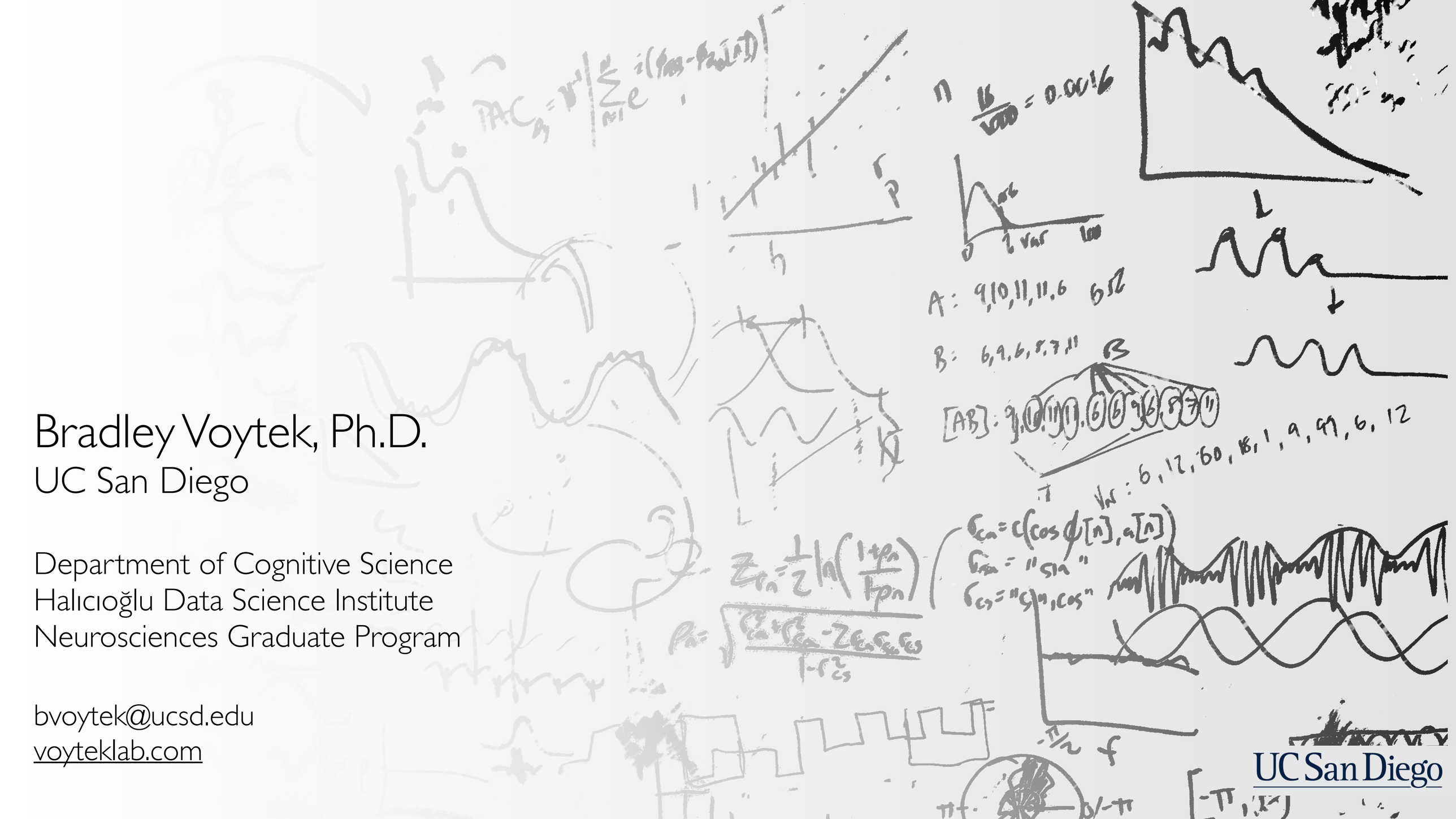
Bradley Voytek, Ph.D.

UC San Diego

Department of Cognitive Science
Halıcıoğlu Data Science Institute
Neurosciences Graduate Program

bvoytek@ucsd.edu
voyteklab.com



UC San Diego

COGS 9
Introduction to Data Science

Fermi
(stray thoughts)

UC San Diego

Wisdom of the crowds!

COGS 9
Introduction to Data Science

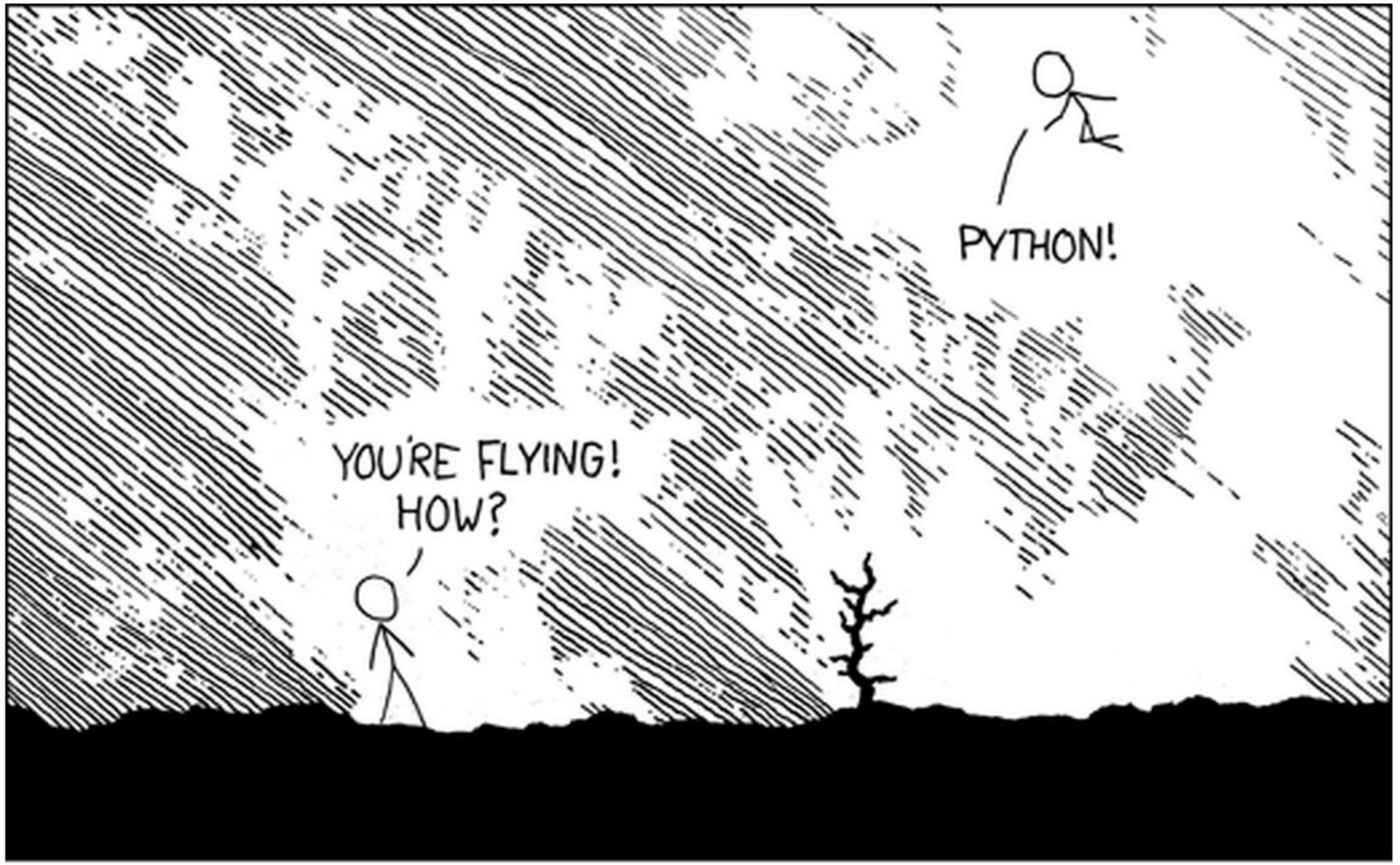
Programming and Version Control

UC San Diego

Today's Learning Objective

Understand the utility of Python and higher-level programming languages

Why Python?



Source: *XKCD*

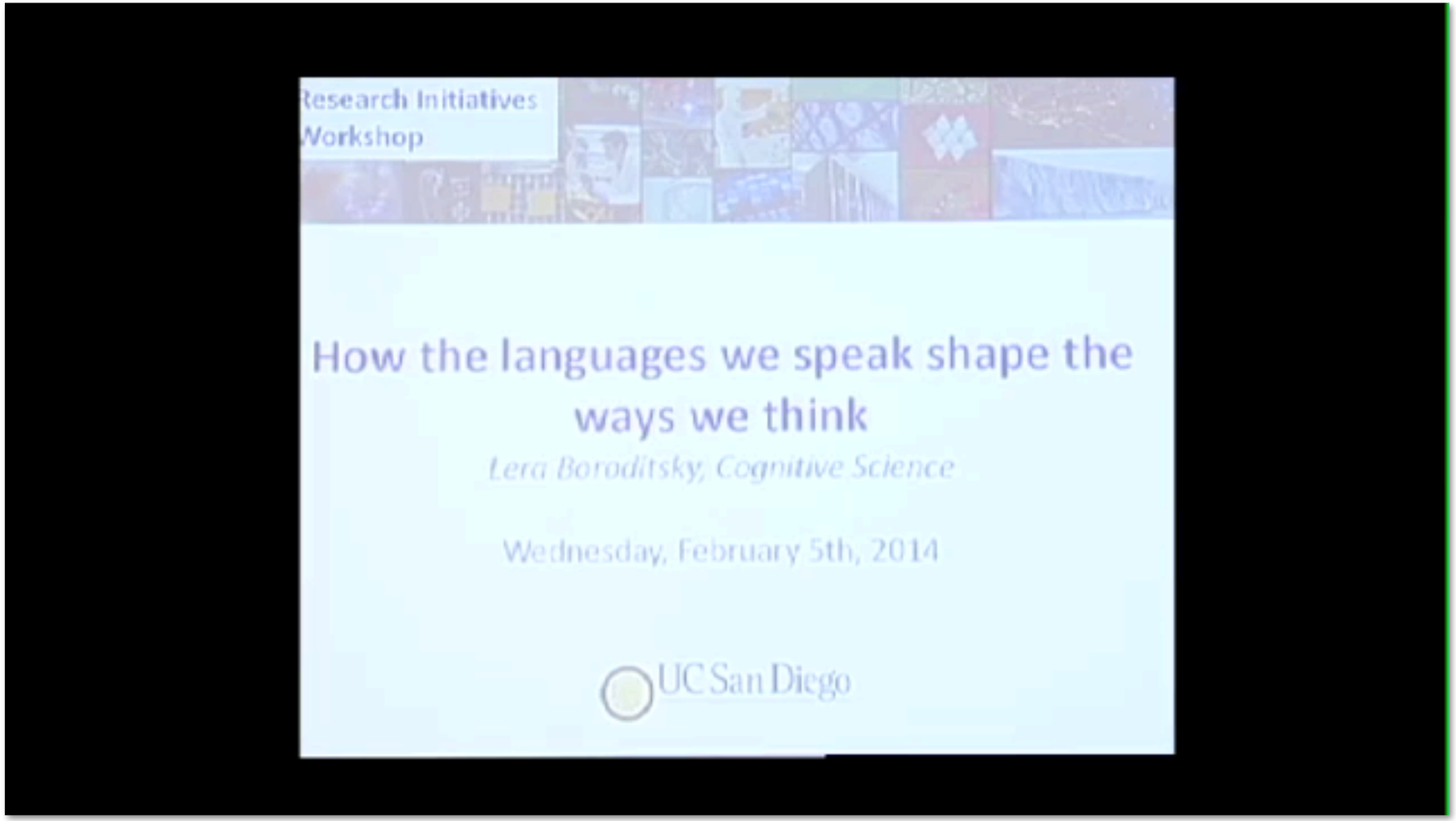
The Zen of Python

```
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!
```

Guido
van Rossum

Source: Python PEP (Python Enhancement Proposals) 20

Languages shape thought



Source: Lera Boroditsky

Programming

Why Python?

EASY TO LEARN

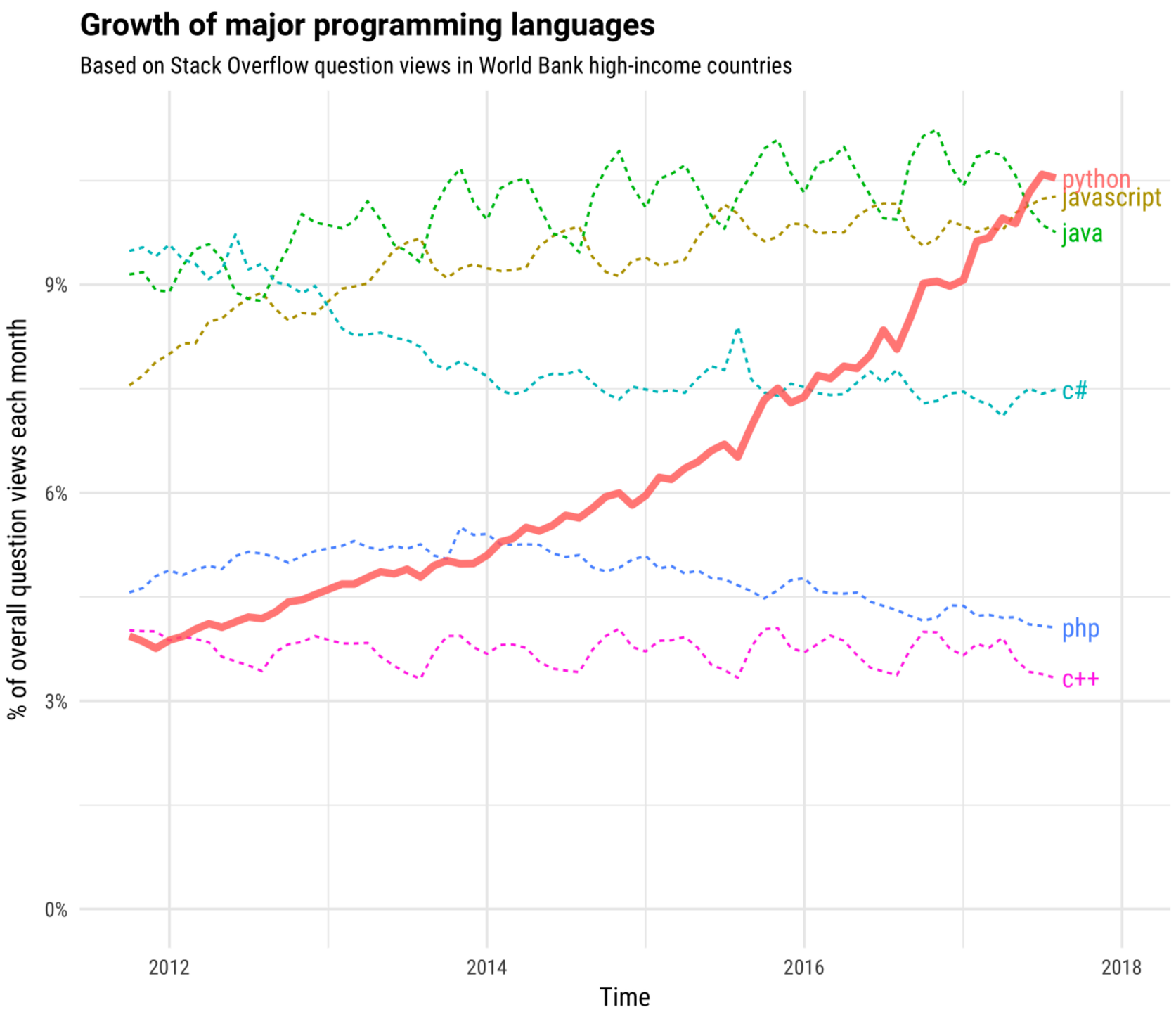
clear, simple syntax and structure

RAPID DEVELOPMENT AND PROTOTYPING

get more done with less, and in less time

HUGE SUPPORT

cross-platform compatible (OS X, Windows, Linux/Unix)



Source: *Stack Overflow*

Why Python?

Key Results

Here are a few of the top takeaways from this year's results.

- Python, the fastest-growing major programming language, has risen in the ranks of programming languages in our survey yet again, edging out Java this year and standing as the second most loved language (behind Rust).

Source: https://insights.stackoverflow.com/survey/2019

Why Python?

Most Loved, Dreaded, and Wanted

Most Loved, Dreaded, and Wanted Languages

Loved | Dreaded | Wanted

Source: https://insights.stackoverflow.com/survey/2019

Why Python?

Most Loved, Dreaded, and Wanted

Most Loved, Dreaded, and Wanted Languages

Loved | Dreaded | Wanted

Source: https://insights.stackoverflow.com/survey/2019

Why Python?

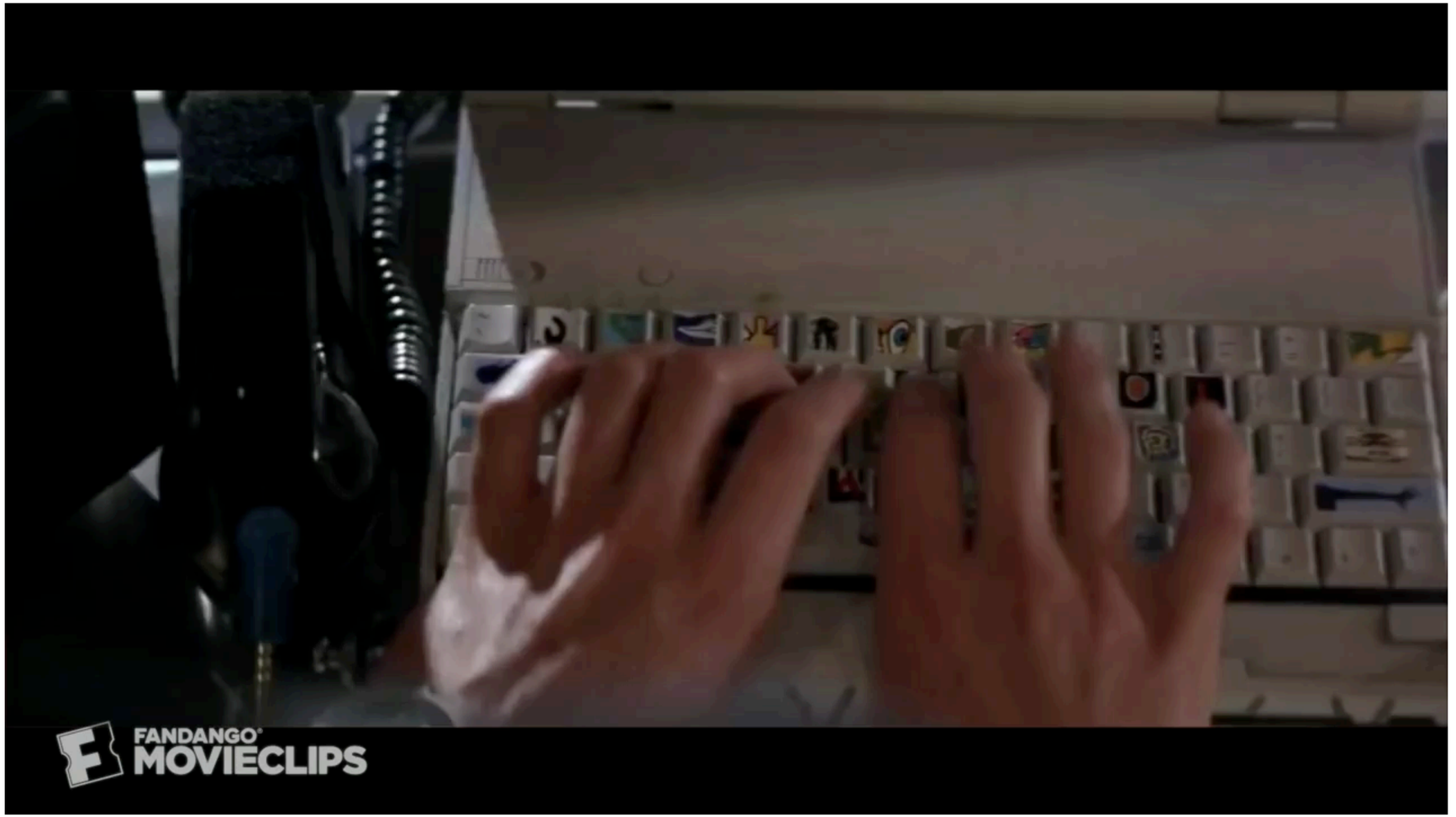
Most Loved, Dreaded, and Wanted

Most Loved, Dreaded, and Wanted Languages

Loved | Dreaded | Wanted

Source: https://insights.stackoverflow.com/survey/2019

Getting computers to work for you

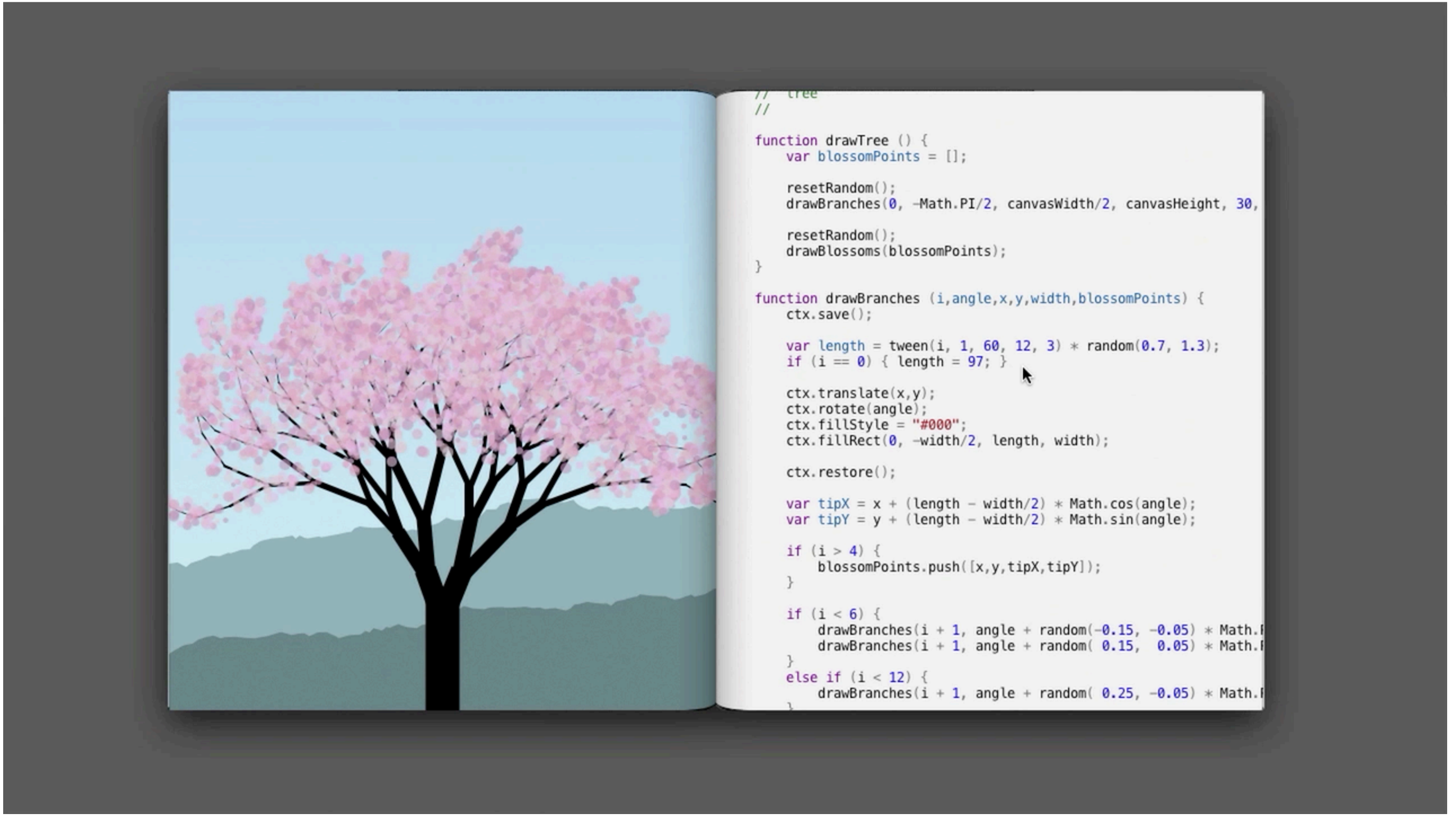


Source: *Hackers*

The future of programming

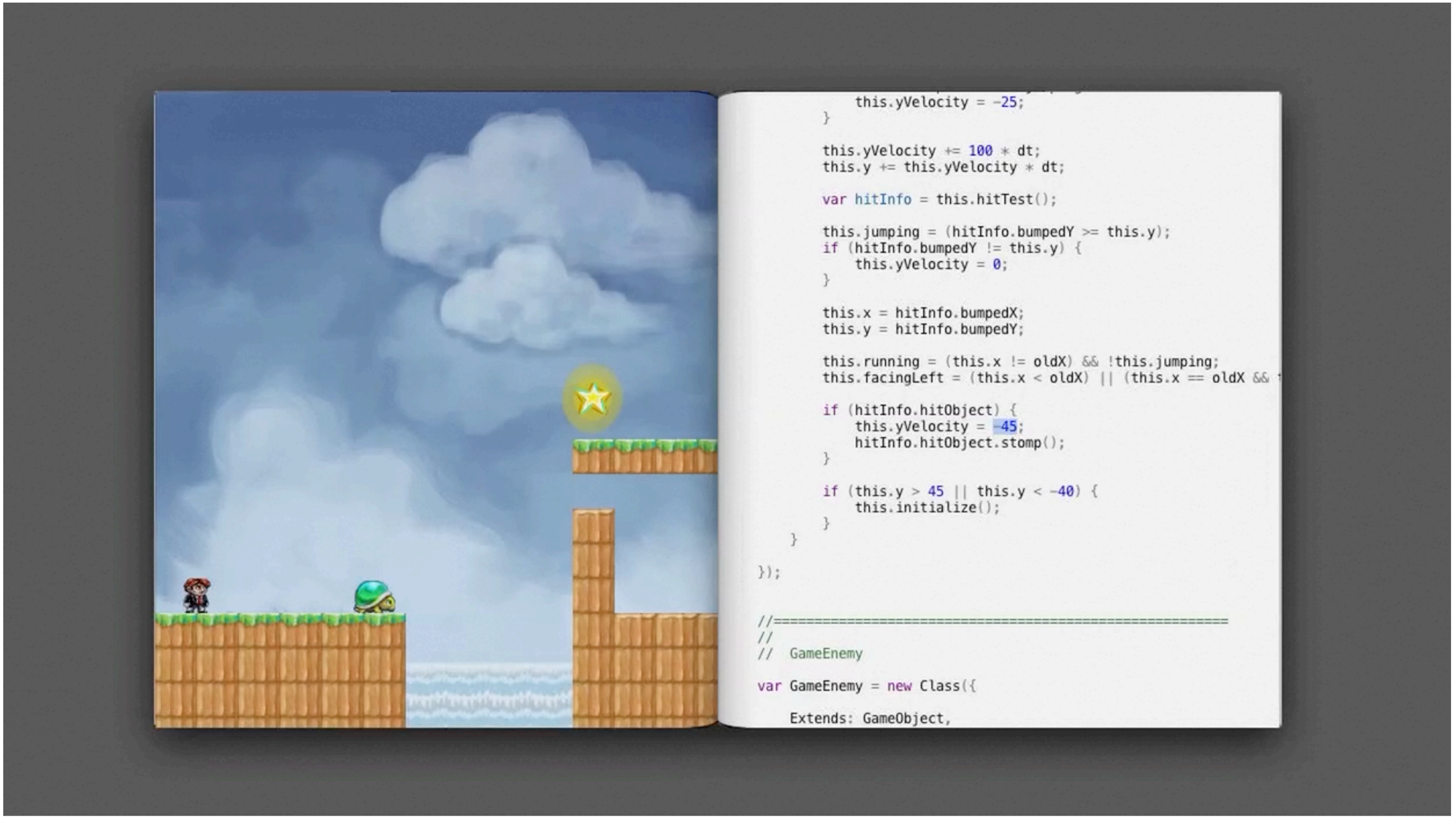
The future of programming

Source: Bret Victor, *Turing Complete* 2012



```
//  tree
//

function drawTree () {
    var blossomPoints = [];

    resetRandom();
    drawBranches(0, -Math.PI/2, canvasWidth/2, canvasHeight, 30,

    resetRandom();
    drawBlossoms(blossomPoints);
}

function drawBranches (i,angle,x,y,width,blossomPoints) {
    ctx.save();

    var length = tween(i, 1, 60, 12, 3) * random(0.7, 1.3);
    if (i == 0) { length = 97; }

    ctx.translate(x,y);
    ctx.rotate(angle);
    ctx.fillStyle = "#000";
    ctx.fillRect(0, -width/2, length, width);

    ctx.restore();

    var tipX = x + (length - width/2) * Math.cos(angle);
    var tipY = y + (length - width/2) * Math.sin(angle);

    if (i > 4) {
        blossomPoints.push([x,y,tipX,tipY]);
    }

    if (i < 6) {
        drawBranches(i + 1, angle + random(-0.15, -0.05) * Math.
        drawBranches(i + 1, angle + random( 0.15,  0.05) * Math.
    }
    else if (i < 12) {
        drawBranches(i + 1, angle + random( 0.25, -0.05) * Math.
    }
```

Source: Bret Victor, *Turing Complete* 2012



```
            this.yVelocity = -25;
        }

        this.yVelocity += 100 * dt;
        this.y += this.yVelocity * dt;

        var hitInfo = this.hitTest();

        this.jumping = (hitInfo.bumpedY >= this.y);
        if (hitInfo.bumpedY != this.y) {
            this.yVelocity = 0;
        }

        this.x = hitInfo.bumpedX;
        this.y = hitInfo.bumpedY;

        this.running = (this.x != oldX) && !this.jumping;
        this.facingLeft = (this.x < oldX) || (this.x == oldX &&

        if (hitInfo.hitObject) {
            this.yVelocity = -45;
            hitInfo.hitObject.stomp();
        }

        if (this.y > 45 || this.y < -40) {
            this.initialize();
        }
    }

});


//=========================================================
//
//  GameEnemy

var GameEnemy = new Class({

    Extends: GameObject,
```

Source: Bret Victor, *Turing Complete* 2012

Today's Learning Objective

Understand basic Python syntax

Python Basics: Data Types

Numbers

- integer
- floating-point
- complex

Booleans are True/False (False and 0 are the same thing)

Strings

- characters are strings of length 1
- use your choice of ‘ or “
- multi line option: ”””
- unicode: u’café’

Python Basics: Numbers

Integers: the numbers you can easily count, like 1, 2, 3, as well as 0 and the negative numbers, like –1, –2, –3

Decimal numbers: the numbers with a decimal point and some digits after it, like 1.25, 0.3752, and –101.2

In computer programming, decimal numbers are also called **floating-point numbers**

Variable type conversions

Determine type: type()

Drop decimal: int()
Round up / down: round()
Change to decimal: float()
Change to string: str()

Python Basics: Operators

- + - * / %
- += -= (no ++ or --)
- Assignment using =
  - a = 1
  - a = "foo"
- Testing using ==, results in a boolean value
  - a == “foo”
- Can also use + to concatenate strings

Python Basics: Math

Syntax	Math	Operation Name
`a+b`	$a + b$	addition
`a-b`	$a - b$	subtraction
`a*b`	$a \times b$	multiplication
`a/b`	$a \div b$	division (see note below)
`a//b`	$\lfloor a \div b \rfloor$	floor division (e.g. 5//2=2) - Available in Python 2.2 and later
`a%b`	$a \bmod b$	modulo
`-a`	$-a$	negation
`abs(a)`	$\lvert a \rvert$	absolute value
`a**b`	a^b	exponent
`math.sqrt(a)`	$\sqrt{a}$	square root

Python Basics: Math

ORDER OF OPERATIONS!

PEMDAS: Parentheses, Exponents, Multiplication, Division, Addition, Subtraction

```
result = 18 + 9 / 3
print(result)
result = (18+9) / 3
print(result)
```

```
21
9
```

UC San Diego

Python Basics: Strings

In Python, strings are specified by enclosing a sequence of characters within a matching pair of *either single or double quotes*

"fire truck" or 'fire truck'

Comments

In [8]:
```
# Check the r^2 and error of the model fit
print('R-squared: \n', fm.r2_)
print('Fit error: \n', fm.error_)
```

Variables

YES movie_ratings, movieRatings

NO mvrs, x, someData

Modules

numpy: math, matrices
scipy: math, algorithms, time-series, linear algebra…
scikit-learn: machine learning
pandas: data structures
matplotlib: visualizations

Modules & namespaces

```
>>> import numpy
>>> import scipy
>>> a = numpy.arange(10)
>>> scipy.stats.describe(a)
DescribeResult(nobs=10, minmax=(0, 9), mean=4.5, variance=9.1666666666666661,
               skewness=0.0, kurtosis=-1.2242424242424244)
```

```
>>> import numpy as np
>>> from scipy import stats
>>> a = np.arange(10)
>>> stats.describe(a)
DescribeResult(nobs=10, minmax=(0, 9), mean=4.5, variance=9.1666666666666661,
               skewness=0.0, kurtosis=-1.2242424242424244)
```

Debugging

```
In [1]: print hello world
```

```
  File "<ipython-input-1-0542465381fc>", line 1
    print hello world
                    ^
SyntaxError: invalid syntax
```

```
In [3]: import numpy as np
        # generate random number
        u = np.random.rand()

        if u < 0.5
            coinFlip = 1
        else:
            coinFlip = 0

        # print result
        print(coinFlip)
```

```
  File "<ipython-input-3-a1e1c818d2ce>", line 5
    if u < 0.5
             ^
SyntaxError: invalid syntax
```

COGS 9
Introduction to Data Science

Version Control

UC San Diego

Today's Learning Objective

What is version control software and how is it used?

This sucks

Voytek-NatNeuro-Craniectomy_2008Dec14.doc	Dec 16, 2008, 4:22 PM
Voytek-NatNeuro-Craniectomy_2008Dec26.doc	Jan 14, 2009, 5:26 PM
Voytek-NatNeuro-Craniectomy_2008Feb23.doc	Feb 23, 2009, 1:05 PM
Voytek-NatNeuro-Craniectomy_2008Jan14.doc	Jan 19, 2009, 9:54 PM
Voytek-NatNeuro-Craniectomy_2008Jan19.doc	Jan 20, 2009, 7:06 PM
Voytek-NatNeuro-Craniectomy_2009Mar4.doc	Mar 4, 2009, 10:03 PM
Voytek-NatNeuro-Craniectomy_2009Mar12.doc	Mar 12, 2009, 1:58 PM
Voytek-NatNeuro-Craniectomy_2009Mar16-FINALtoBOB.doc	Mar 16, 2009, 4:41 PM
Voytek-NatNeuro-Craniectomy_2009Mar20.doc	Apr 3, 2009, 6:08 PM

So does this

[redacted] Feb 26

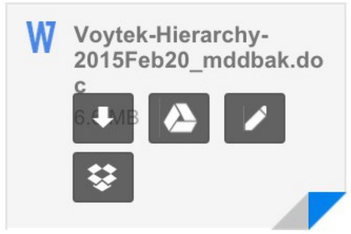
Hi Brad (and everyone),

Attached is the manuscript incorporating all of our comments. I think most of the issues are fairly minor, which is great. To my mind, there are only two conceptual issues that we need to nail down before getting it back in:

1) The justification for a 100ms theta phase encoding window
2) The explanation for an earlier M1/PMC but later PFC response in some conditions

Brad, did you want to take a look through all the comments & make sure you agree with them? Maybe after you've had a chance to look, we can hammer out the answers to 1 & 2 above, if the way forward isn't already clear.

3 Attachments



And so does this…

[redacted] May 11

Hi Brad,

Thanks for chatting with me earlier today. I added the link to the visualization project into my resume and attached the resume. Thanks for any connections you can make for me. I'd love to know where you send it, so I can keep track of that. Thanks again!

Best,

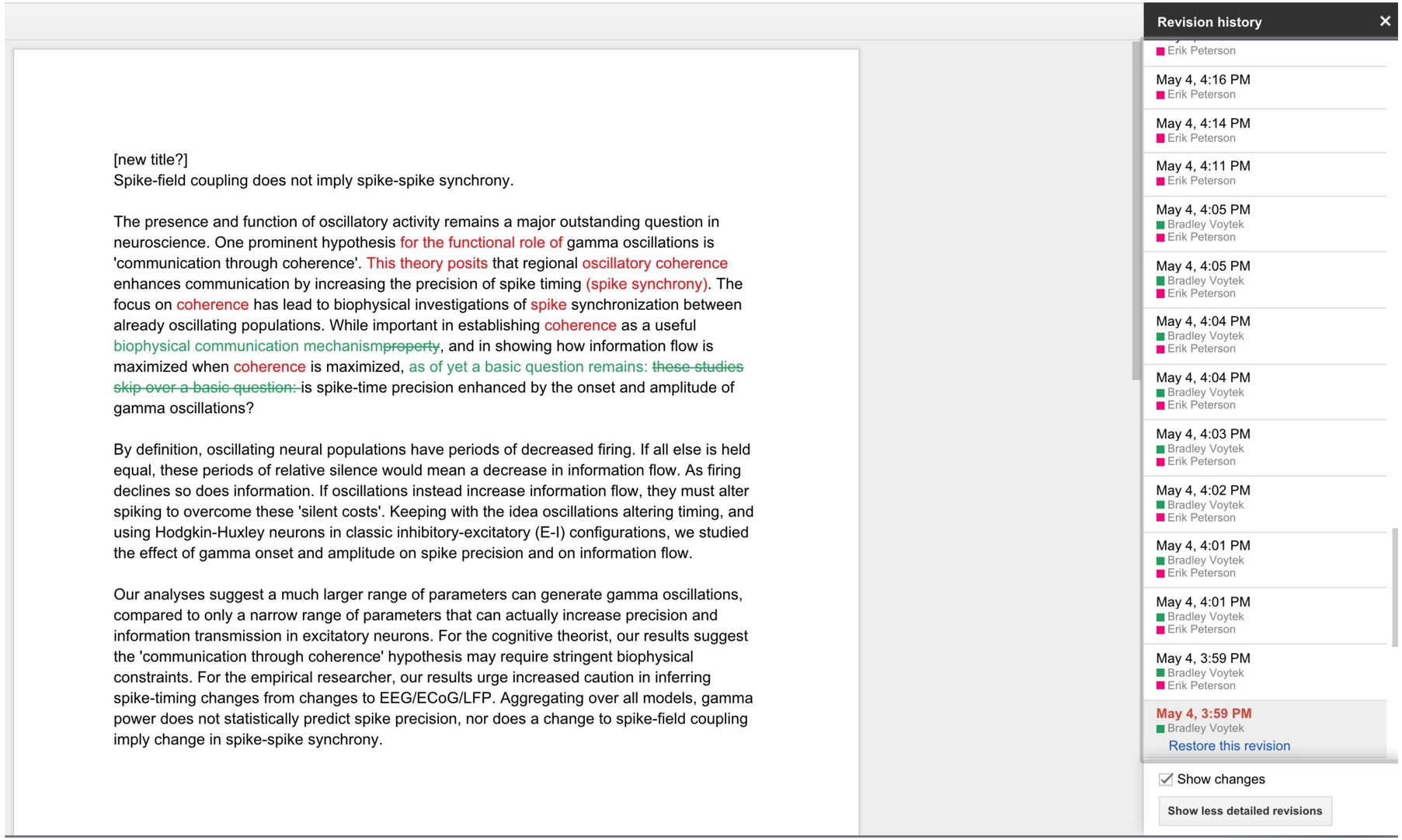
[redacted] May 11

Actually, please use this one. I fixed a typo that was previously missed. Thanks!

[redacted] May 11

Final copy, I swear. Thanks for helping out.

This is better



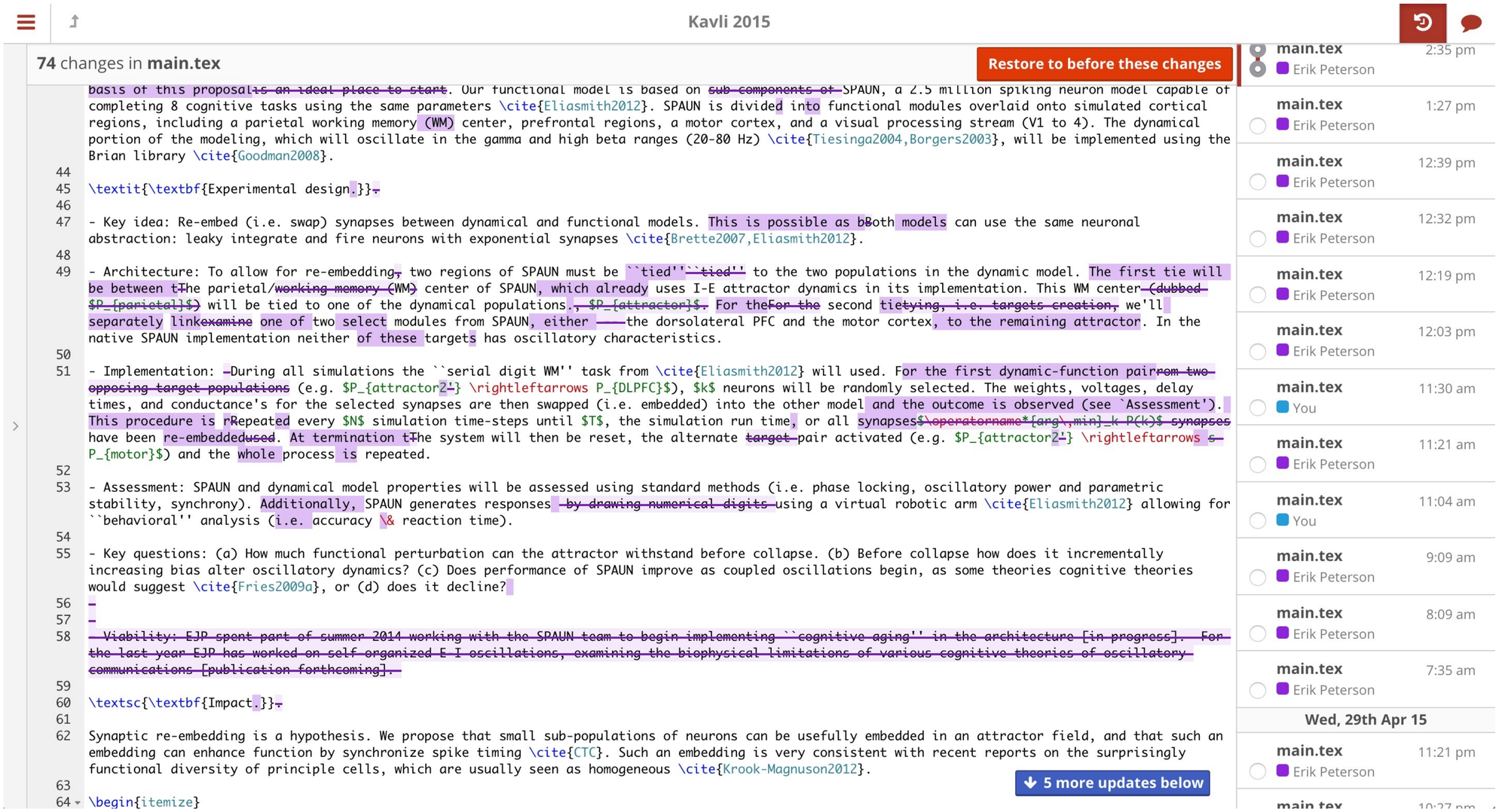
[new title?]
Spike-field coupling does not imply spike-spike synchrony.

The presence and function of oscillatory activity remains a major outstanding question in neuroscience. One prominent hypothesis for the functional role of gamma oscillations is 'communication through coherence'. This theory posits that regional oscillatory coherence enhances communication by increasing the precision of spike timing (spike synchrony). The focus on coherence has lead to biophysical investigations of spike synchronization between already oscillating populations. While important in establishing coherence as a useful biophysical communication mechanism~~property~~, and in showing how information flow is maximized when coherence is maximized, as of yet a basic question remains: ~~these studies skip over a basic question:~~ is spike-time precision enhanced by the onset and amplitude of gamma oscillations?

By definition, oscillating neural populations have periods of decreased firing. If all else is held equal, these periods of relative silence would mean a decrease in information flow. As firing declines so does information. If oscillations instead increase information flow, they must alter spiking to overcome these 'silent costs'. Keeping with the idea oscillations altering timing, and using Hodgkin-Huxley neurons in classic inhibitory-excitatory (E-I) configurations, we studied the effect of gamma onset and amplitude on spike precision and on information flow.

Our analyses suggest a much larger range of parameters can generate gamma oscillations, compared to only a narrow range of parameters that can actually increase precision and information transmission in excitatory neurons. For the cognitive theorist, our results suggest the 'communication through coherence' hypothesis may require stringent biophysical constraints. For the empirical researcher, our results urge increased caution in inferring spike-timing changes from changes to EEG/ECoG/LFP. Aggregating over all models, gamma power does not statistically predict spike precision, nor does a change to spike-field coupling imply change in spike-spike synchrony.

This is even better



Version control

- Enables multiple people to simultaneously work on a single project.

Source: Michael Ernst, *University of Washington*

Version control

- Enables multiple people to simultaneously work on a single project.
- Each person edits their own copy of the files and chooses when to share those changes with the rest of the team.

Source: Michael Ernst, *University of Washington*

Version control

- Enables multiple people to simultaneously work on a single project.
- Each person edits their own copy of the files and chooses when to share those changes with the rest of the team.
- Thus, temporary or partial edits by one person do not interfere with another person's work.

Source: Michael Ernst, *University of Washington*

Version control

- Version control also enables one person to use multiple computers to work on a project, so it is valuable even if you are working by yourself.

Version control

- Version control also enables one person to use multiple computers to work on a project, so it is valuable even if you are working by yourself.

- Version control integrates work done simultaneously by different team members. In most cases, edits to different files or even the same file can be combined without losing any work.

Source: Michael Ernst, *University of Washington*

Version control

- Version control also enables one person to use multiple computers to work on a project, so it is valuable even if you are working by yourself.

- Version control integrates work done simultaneously by different team members. In most cases, edits to different files or even the same file can be combined without losing any work.

- In rare cases, when two people make conflicting edits to the same line of a file, then the version control system requests human assistance in deciding what to do.

Source: Michael Ernst, *University of Washington*

Version control

- Version control gives access to historical versions of your project.

Version control

- Version control gives access to historical versions of your project.
- This is insurance against computer crashes or data lossage.

Source: Michael Ernst, *University of Washington*

Version control

- Version control gives access to historical versions of your project.
- This is insurance against computer crashes or data lossage.
- If you make a mistake, you can roll back to a previous version.

Version control

- Version control gives access to historical versions of your project.
- This is insurance against computer crashes or data lossage.
- If you make a mistake, you can roll back to a previous version.
- You can also undo specific edits without losing all the work that was done in the meanwhile.

Source: Michael Ernst, *University of Washington*

Version control

- Version control gives access to historical versions of your project.
- This is insurance against computer crashes or data lossage.
- If you make a mistake, you can roll back to a previous version.
- You can also undo specific edits without losing all the work that was done in the meanwhile.
- For any part of a file, you can determine when, why, and by whom it was ever edited.

Source: Michael Ernst, *University of Washington*

Repository

- A database containing the files and change history of your project.

Source: Lars Kellogg-Stedman, *Harvard University*

UC San Diego

Working copy

- A local copy of files from a repository.

Revision

- The state of the repository at a certain point in time.

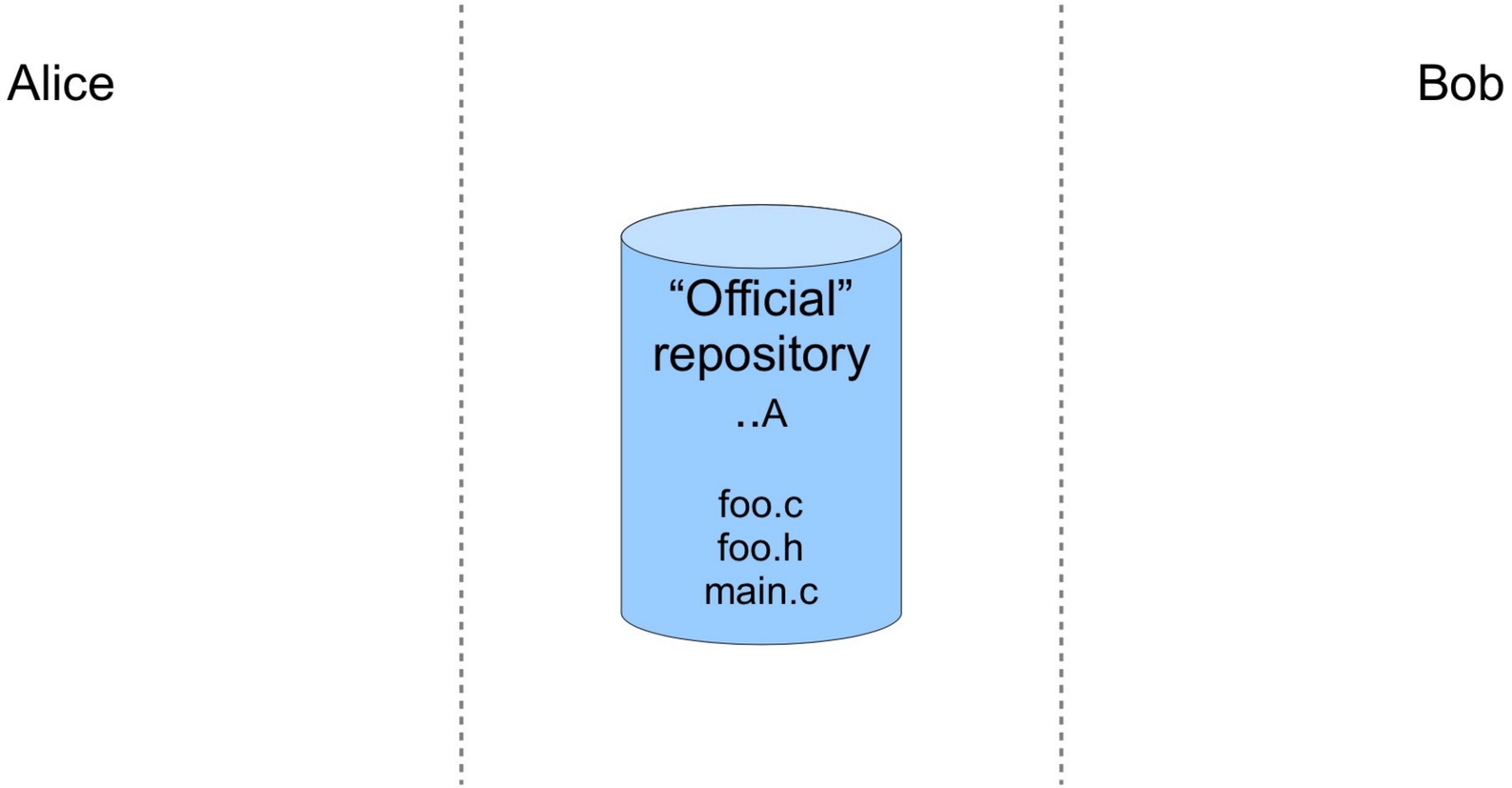
Commit

- To save your changes back to the repository.

Merge

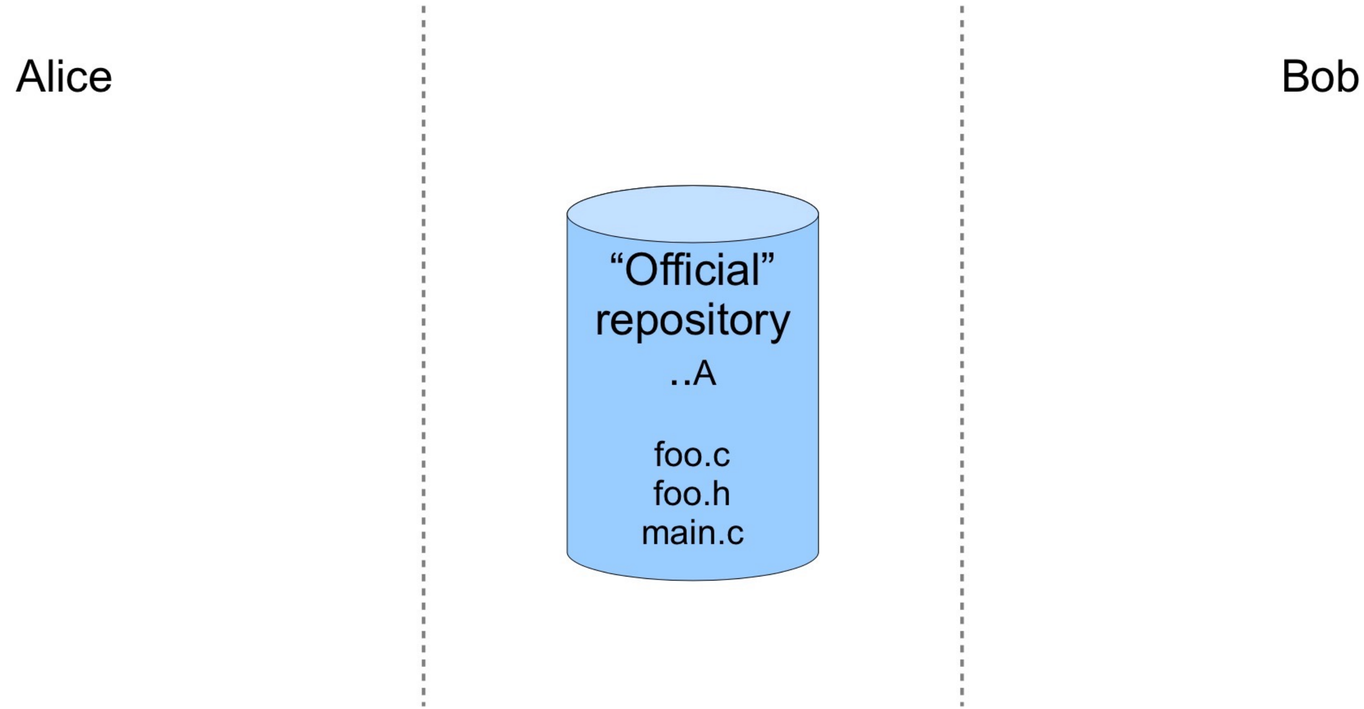
- To combine two sets of changes to the files in your project.

Distributed version control



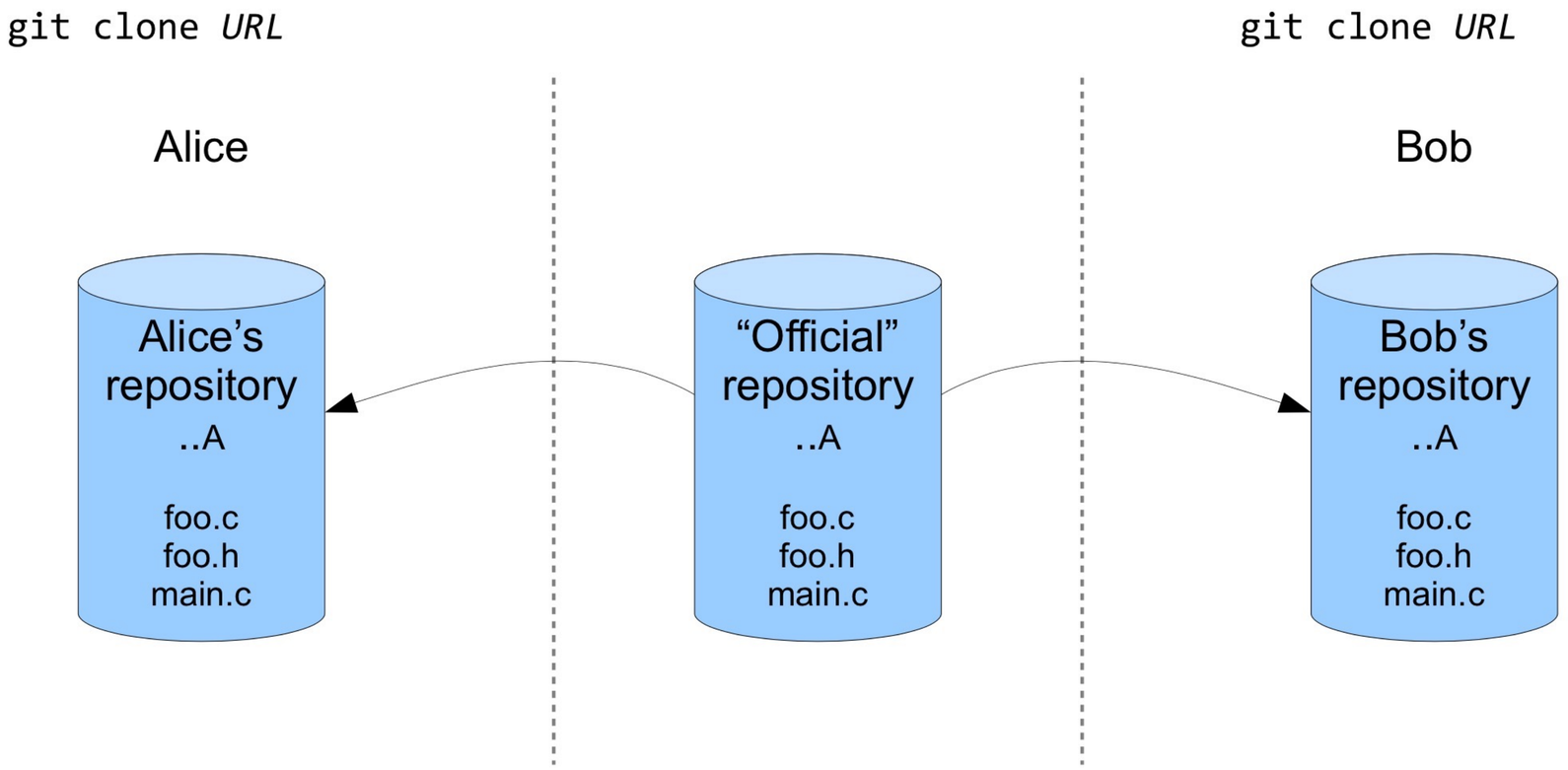
- Spot the differences

Distributed version control



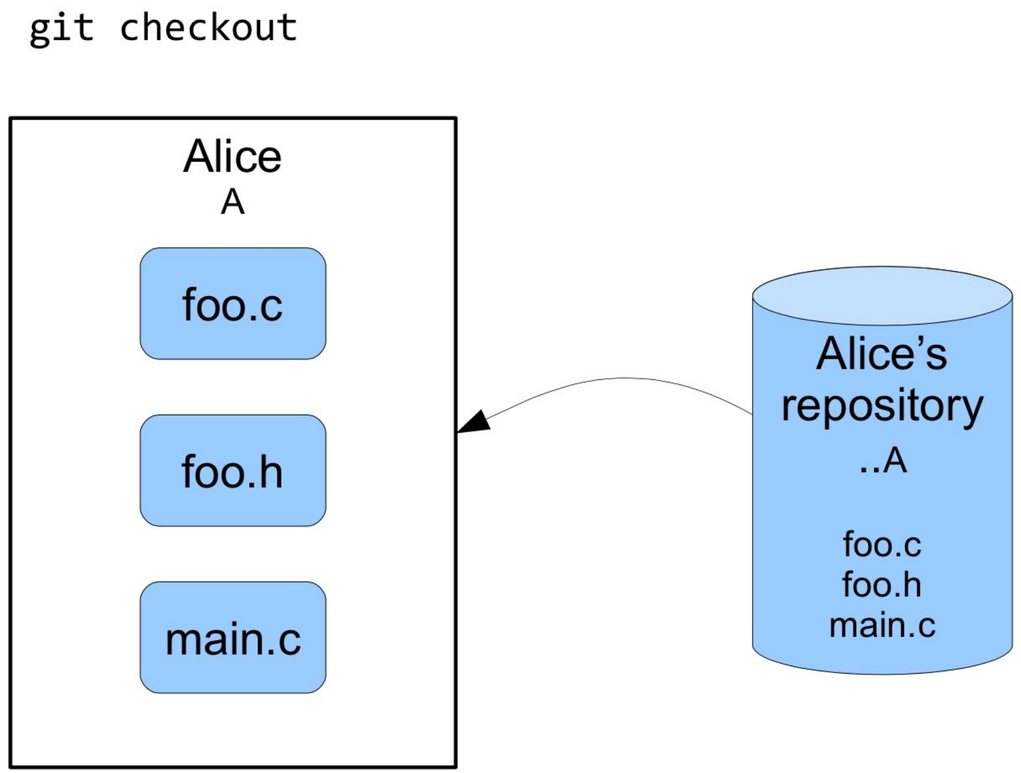
- Alice and Bob will both have their own repositories, no different from the "official" one.

Distributed version control



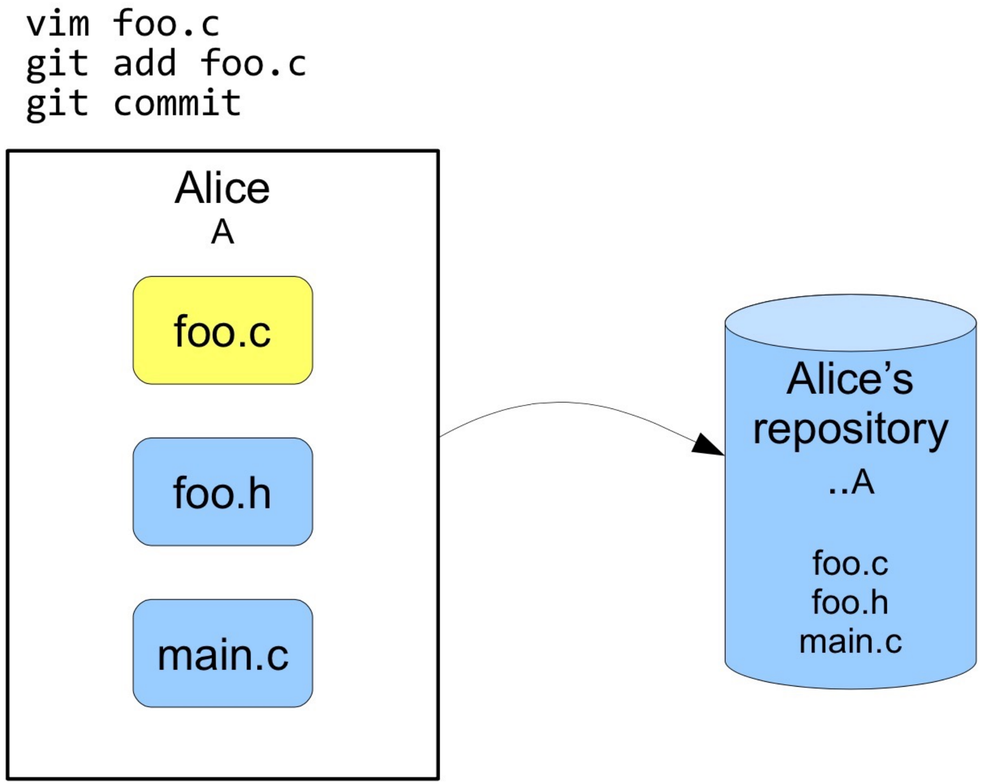
- First step is to *clone* the repository, not check it out.
- This gives you a local clone of the *entire* repo!

Distributed version control



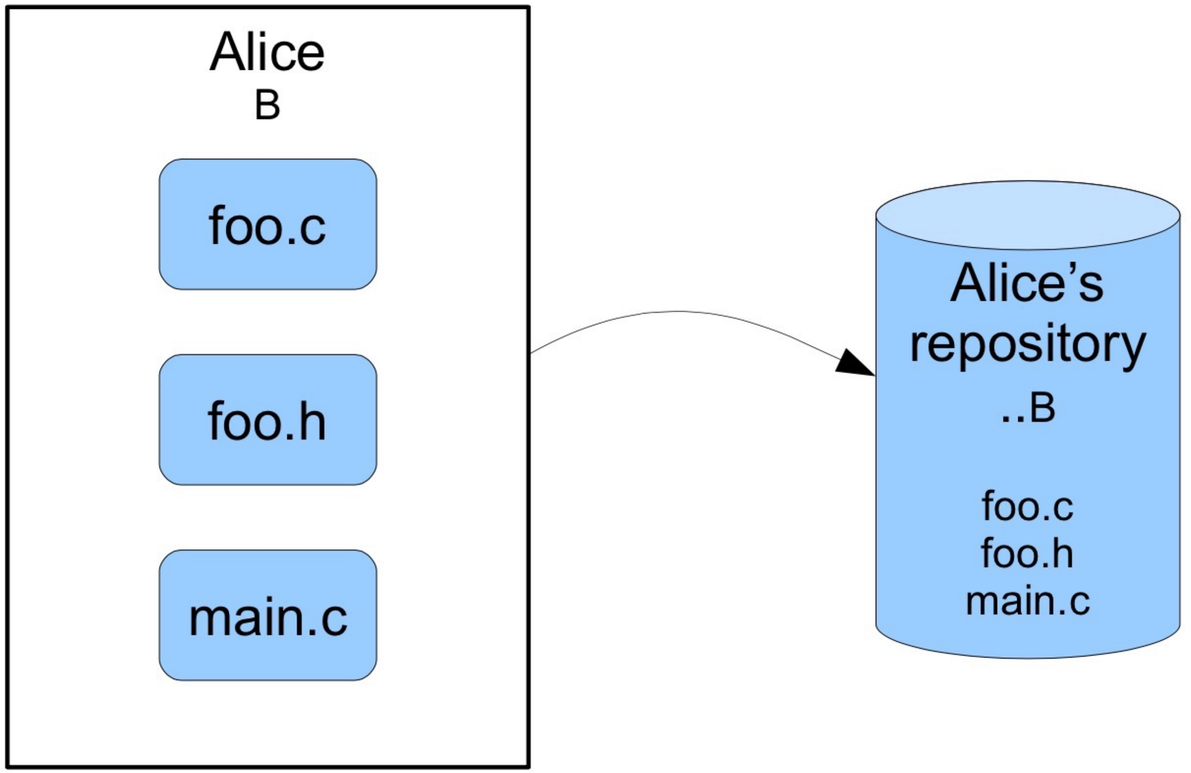
- Alice can now work locally (and offline), without worrying about other repositories.

Distributed version control



- Alice edits foo.c as usual, adds the changes to her commit, and commits.

Distributed version control

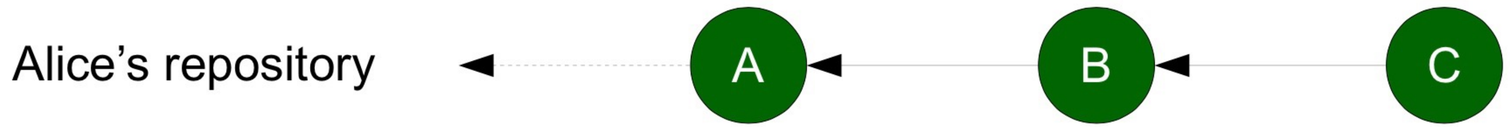


- Revision B, based on A, is now in Alice's repository.

Distributed version control

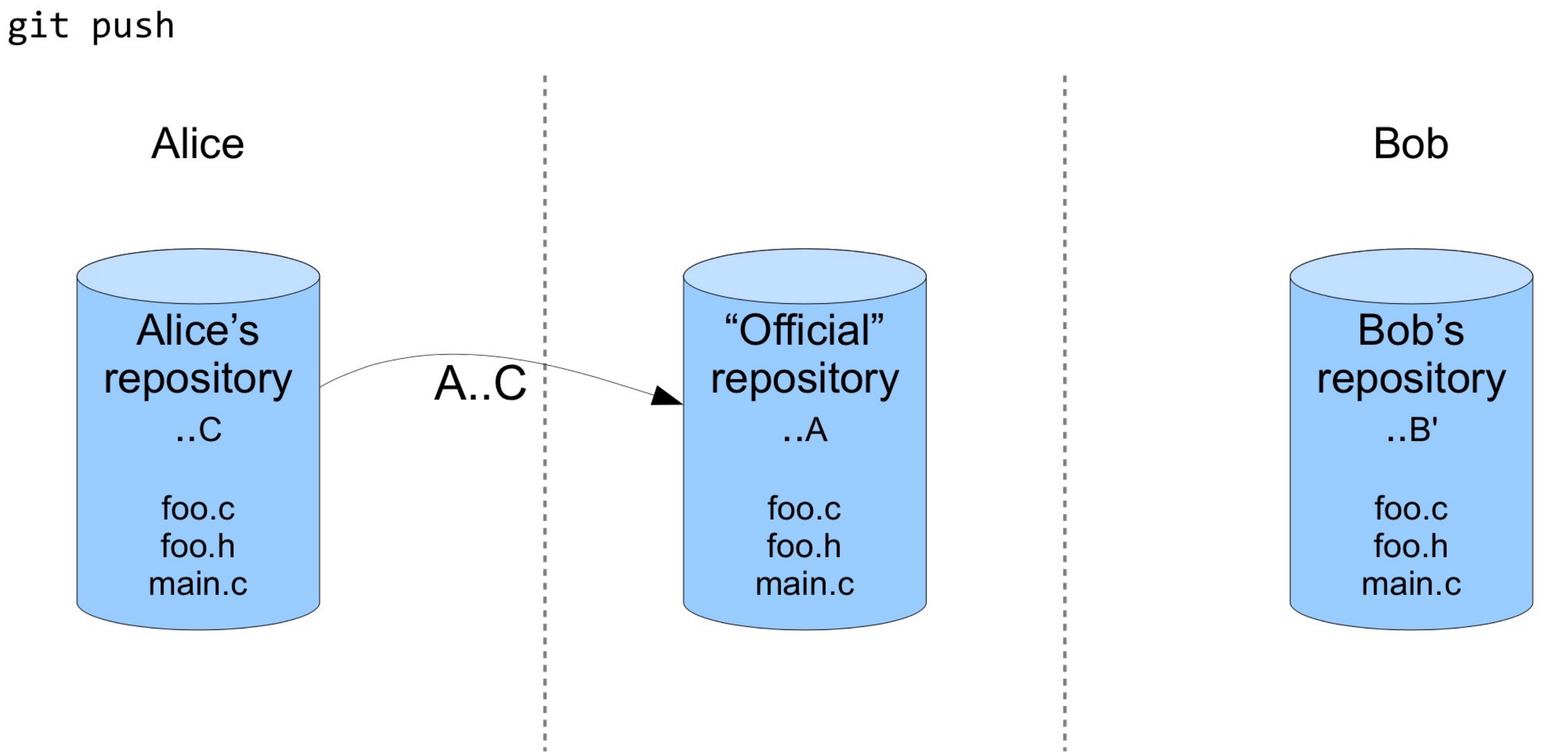
- In our Git example, Alice has committed revision B "onto" revision A.

Distributed version control



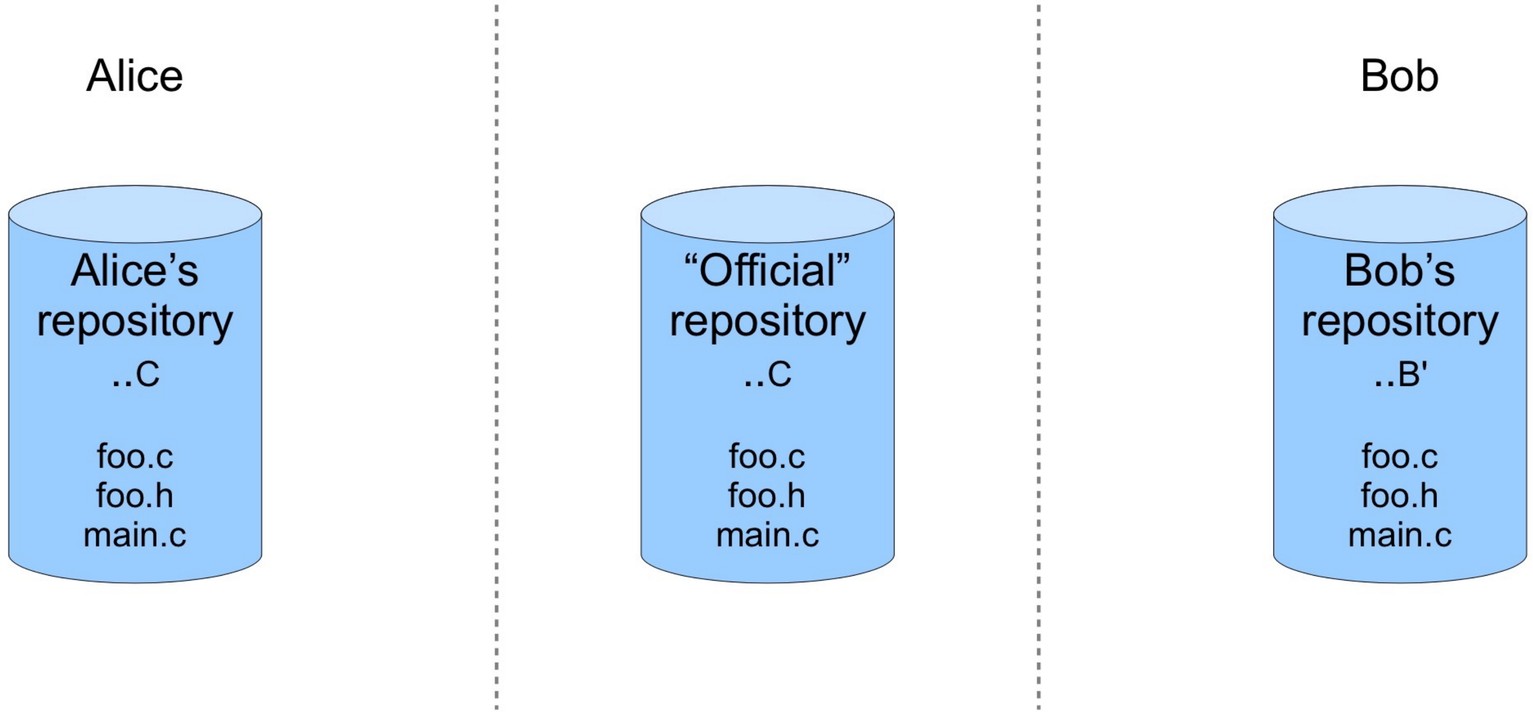
- In our Git example, Alice has committed revision B "onto" revision A.
- She can then commit another revision C onto that.

Distributed version control



- Alice can *push* her new commits to another repository, such as the "official" one.
- The commit being pushed must be a *descendant* of the remote one.

Distributed version control

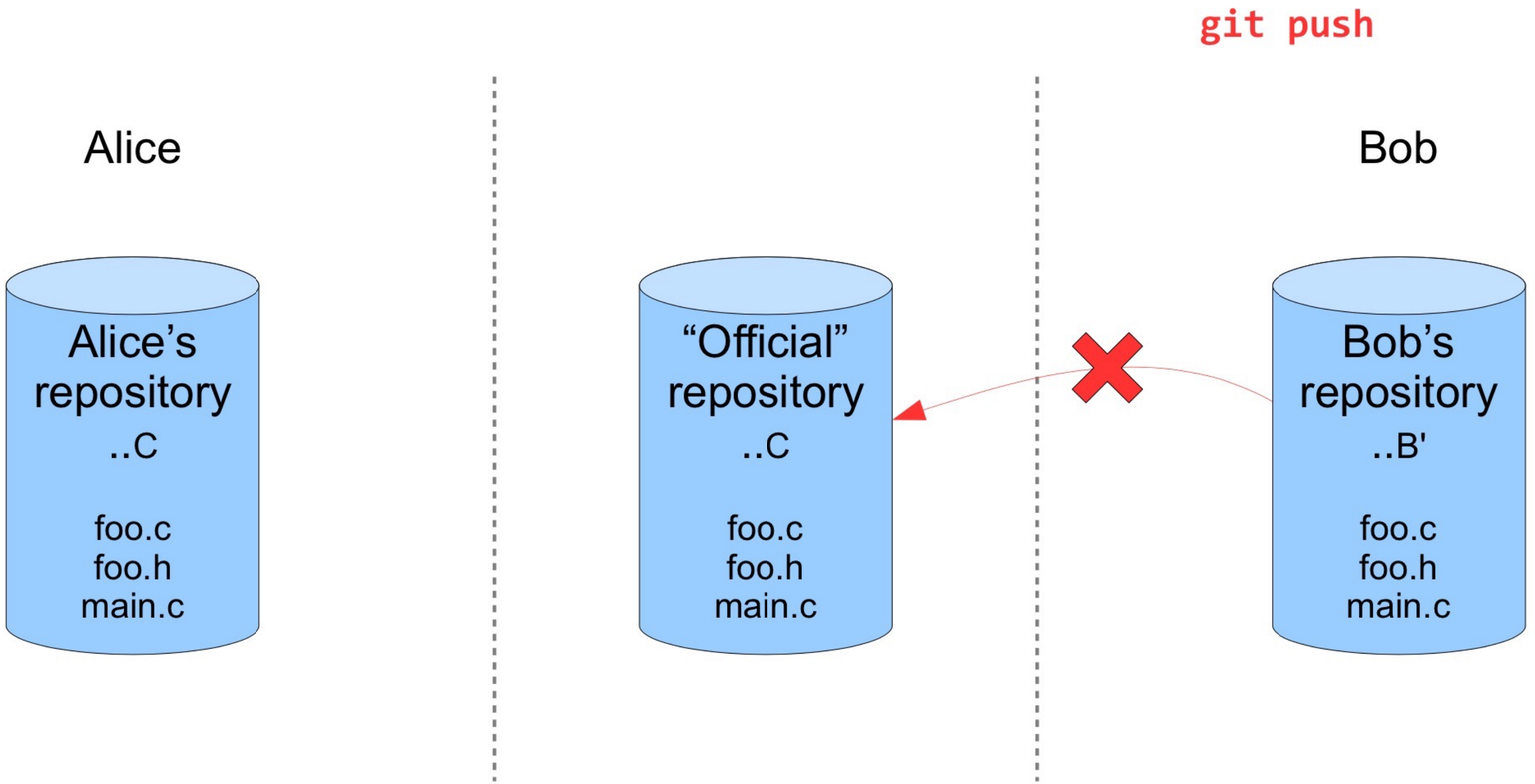


- The official repository now contains Alice's commits.
- Notice Bob has also committed B' but not yet pushed!

Distributed version control

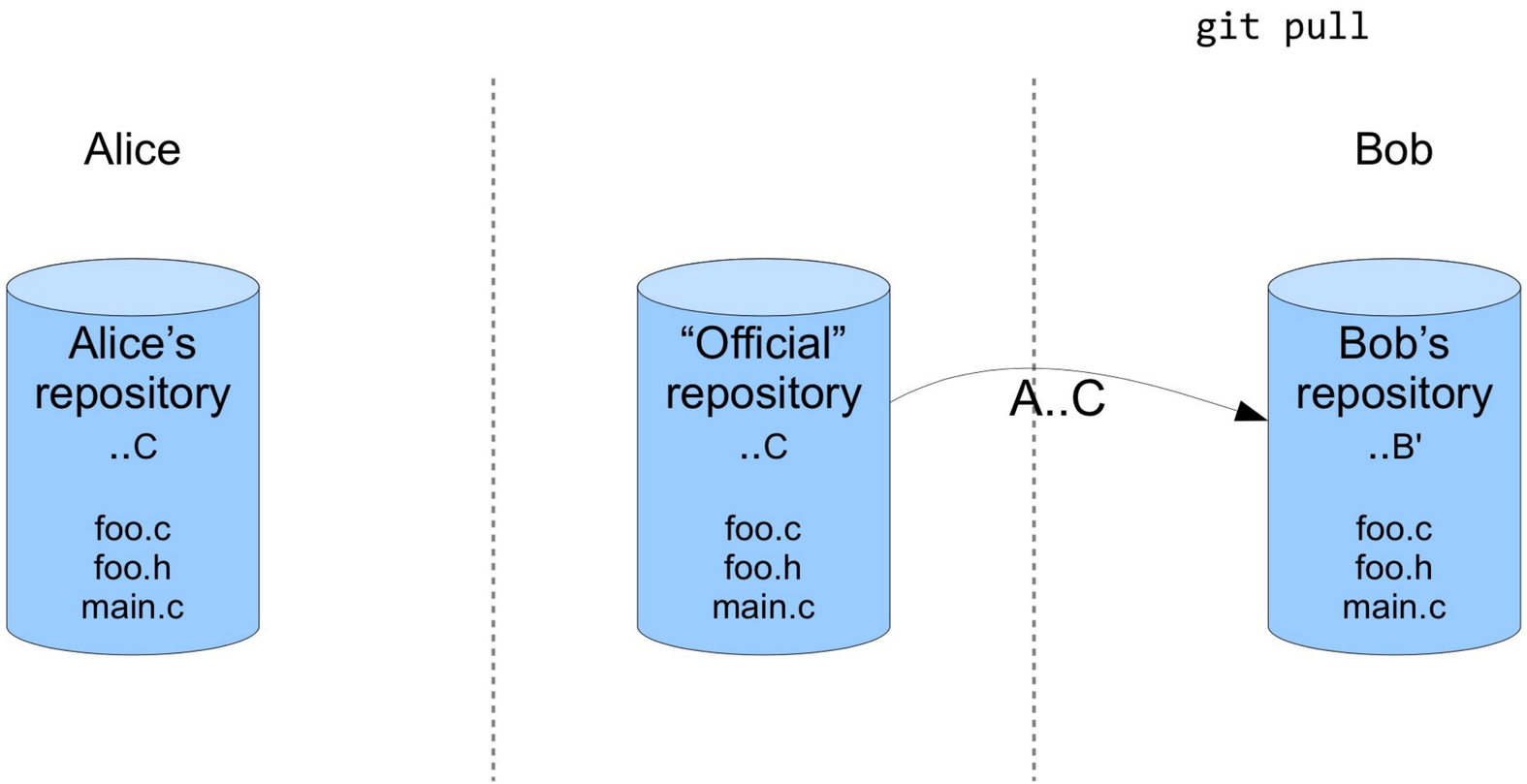
- Bob's commit graph has his new revision, but none of Alice's.

Distributed version control



- Bob cannot push his changes yet, because B' is not a descendant of C.

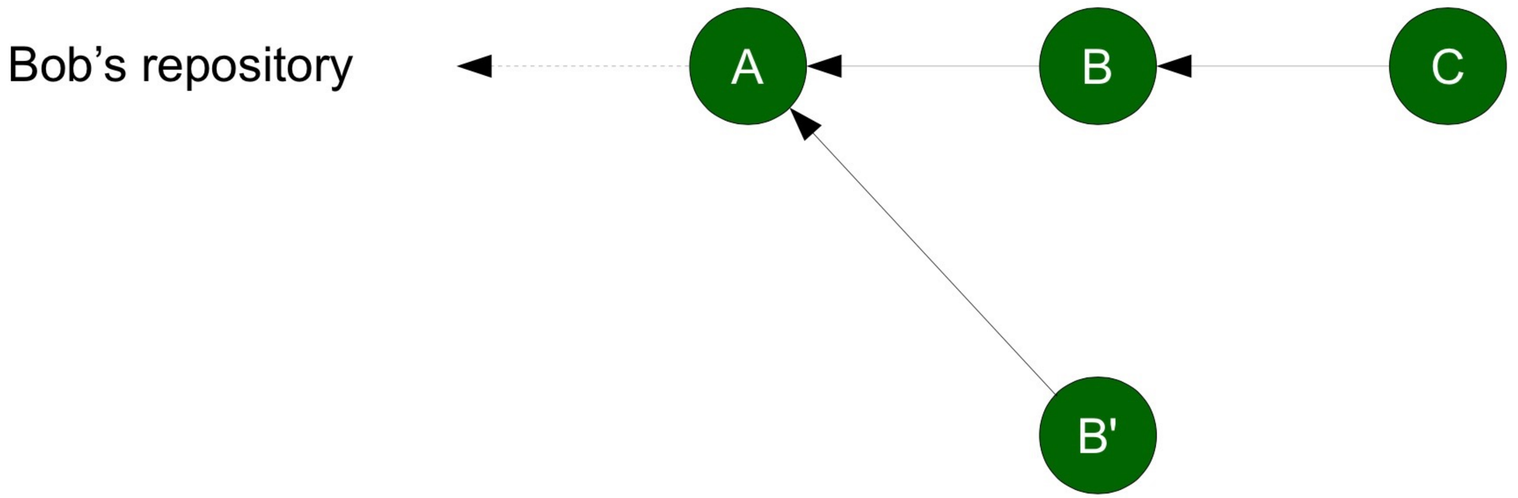
Distributed version control



- Bob needs to get Alice's new changes and merge them.
- First he *pulls* the changes from the official repo...

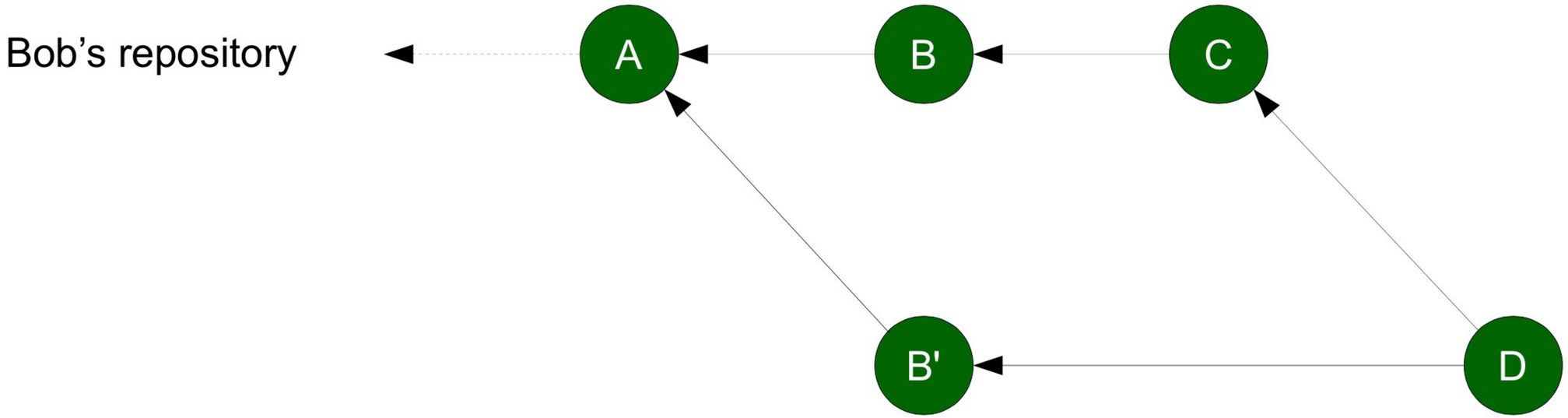
Source: Devin Pohly, *Pennsylvania State University*

Distributed version control



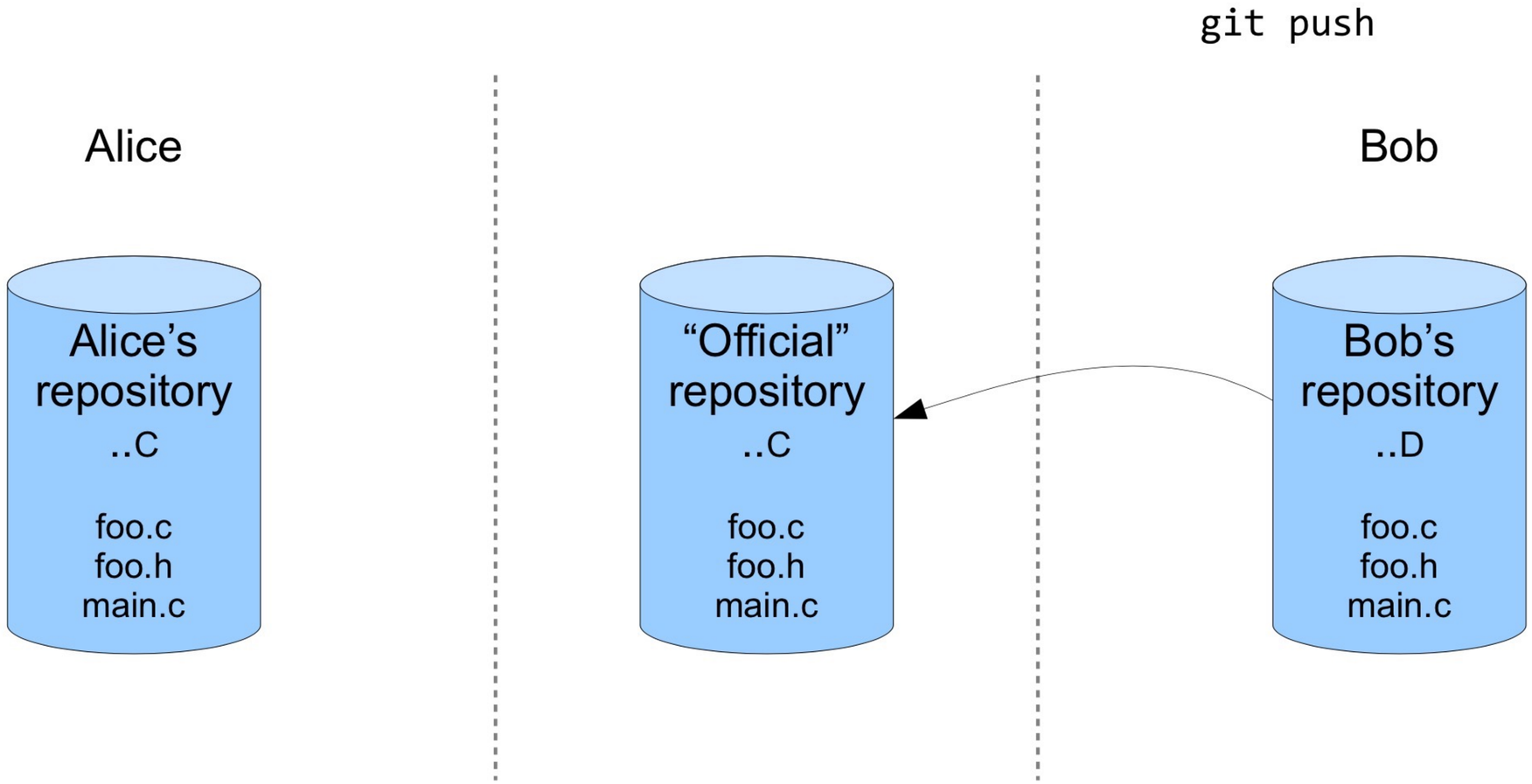
- ... and they are added to his repository.
- He can now *merge* Alice's changes with his.

Distributed version control



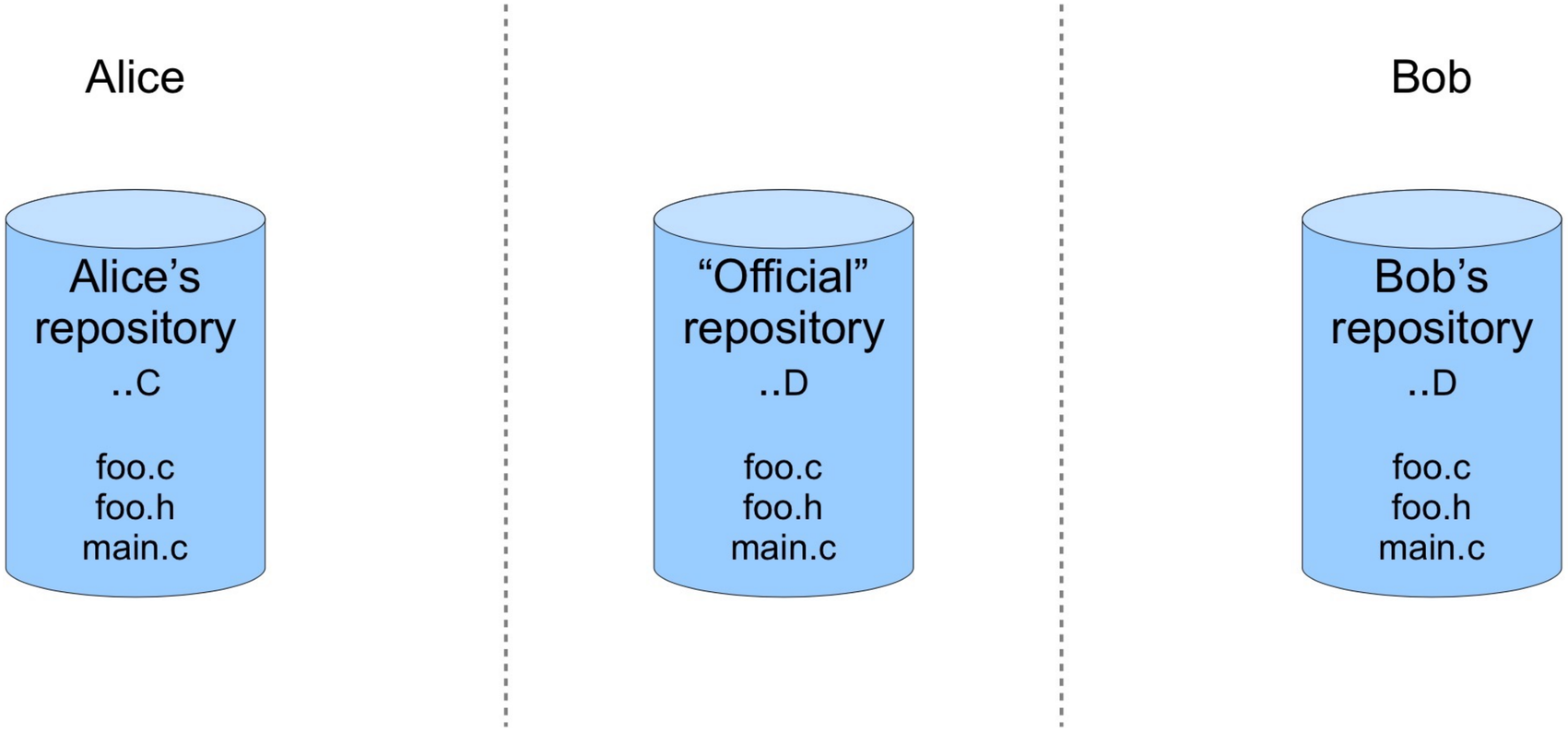
- This creates a new *merge revision* D which is a child of both B' and C.

Distributed version control



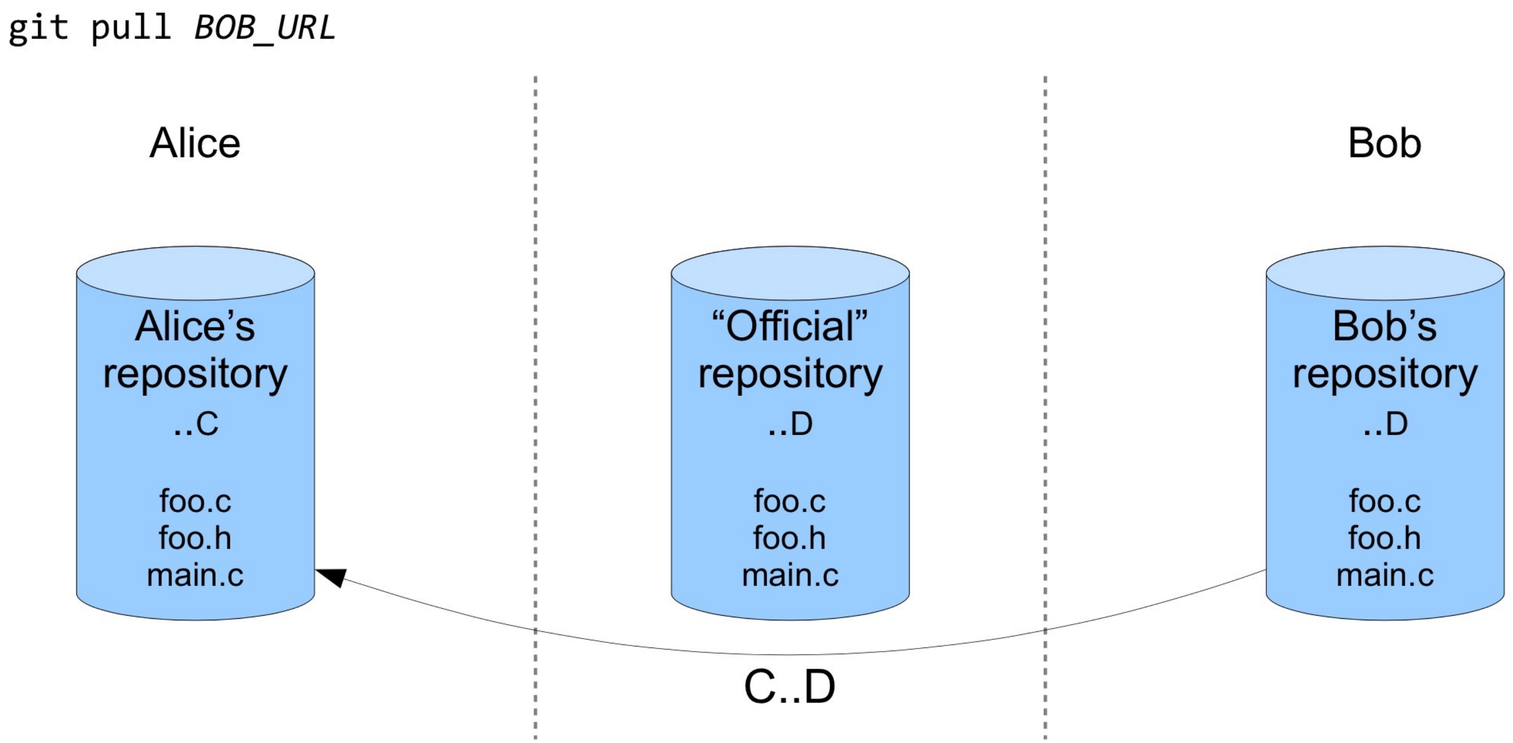
- Since D *is* a descendant of C, Bob can now push!

Distributed version control



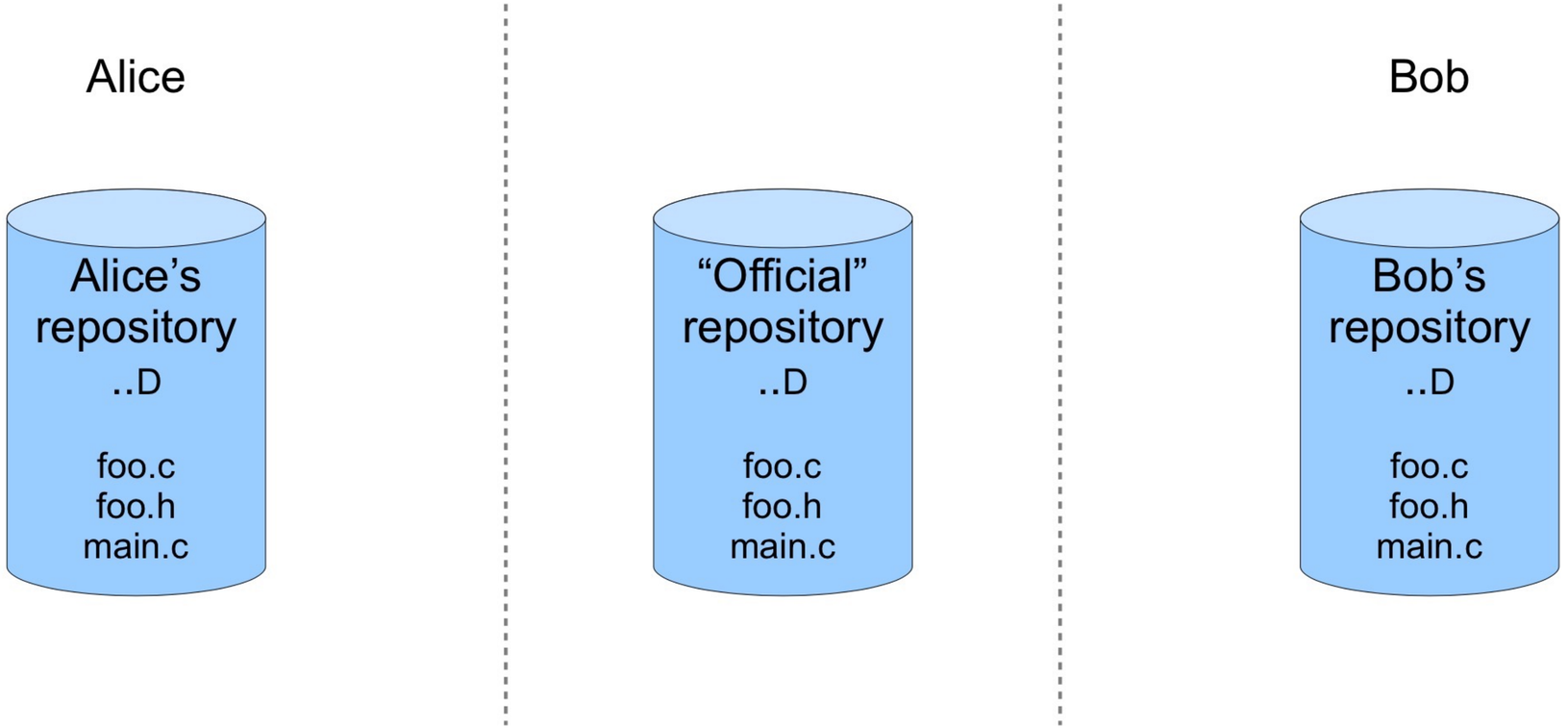
- The official repository now has revision D, which contains both Alice's and Bob's changes.

Distributed version control



- Developers can also collaborate directly.
- Here Alice gets Bob's latest commits from Bob himself instead of from the "official" repository.

Distributed version control



- This could be used, for instance, to collaborate on experimental features that aren't ready for prime-time.

git practice!

- https://try.github.io/

Crap. Tonight is raid night and I am already late.

 Kartik2004sharma 1 day ago

saving before I fuck shit up

 RamzCas 1 day ago

more unworking shit

 karseny99 1 day ago

Fml I hate this shit I am never doing this shit agian

 DeathSurfing 1 day ago

API break, fuck it, we're < 1.0

 DickerDackel 1 day ago

delet fuck

 kotla4444 1 day ago

Source: latenightcommits.com