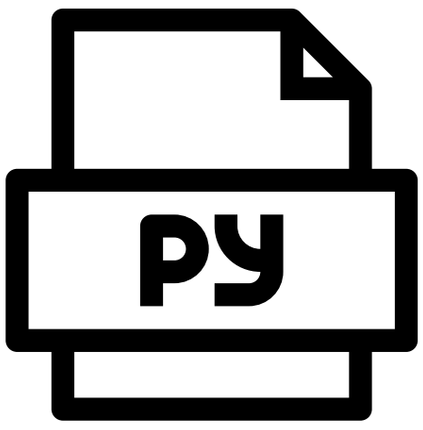
# Meta: Enabling Programming Languages to Learn from the Crowd

Ethan Fast, Michael Bernstein

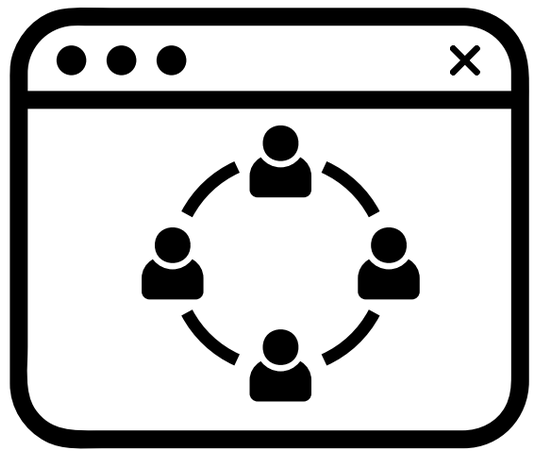
Stanford HCI

# What can programming languages learn from how users write programs?

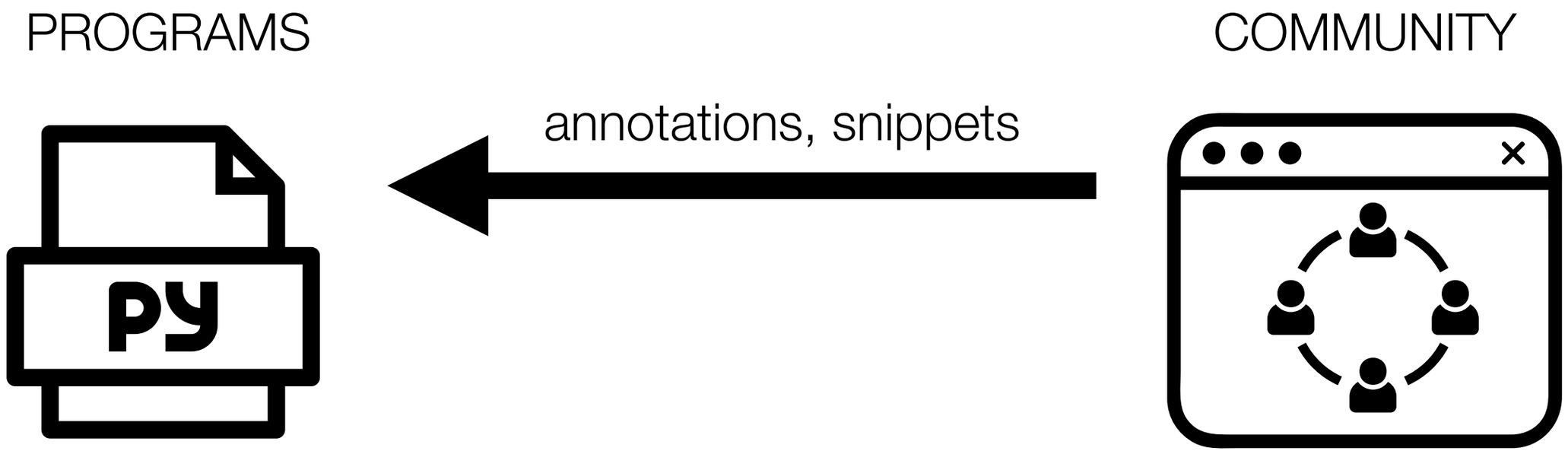
PROGRAMS




COMMUNITY

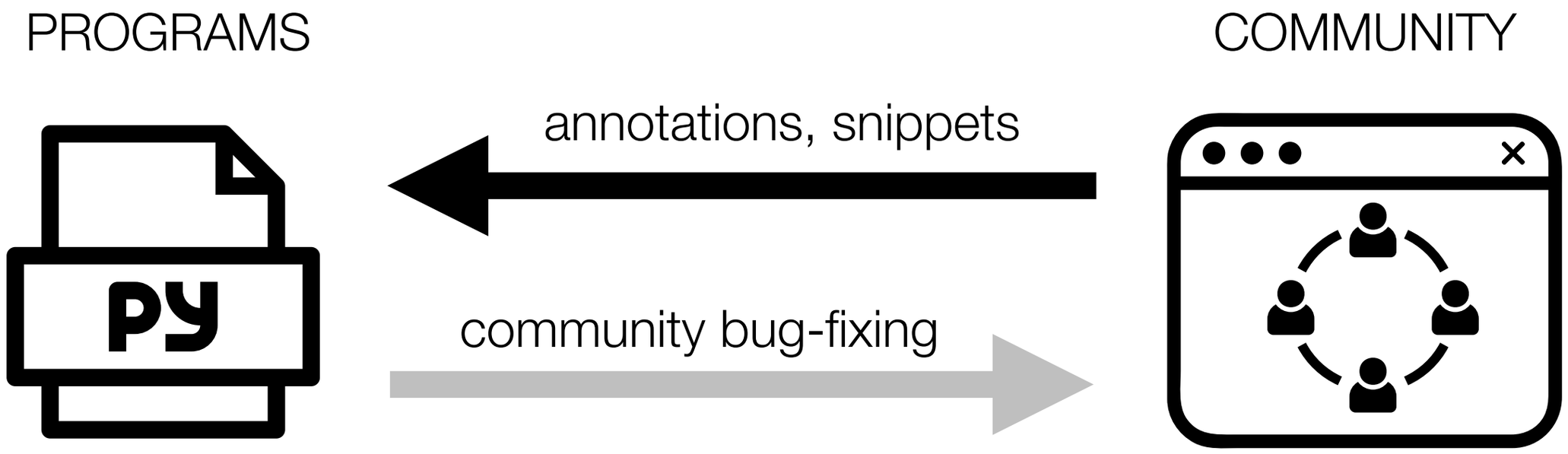


# What can programming languages learn from how users write programs?



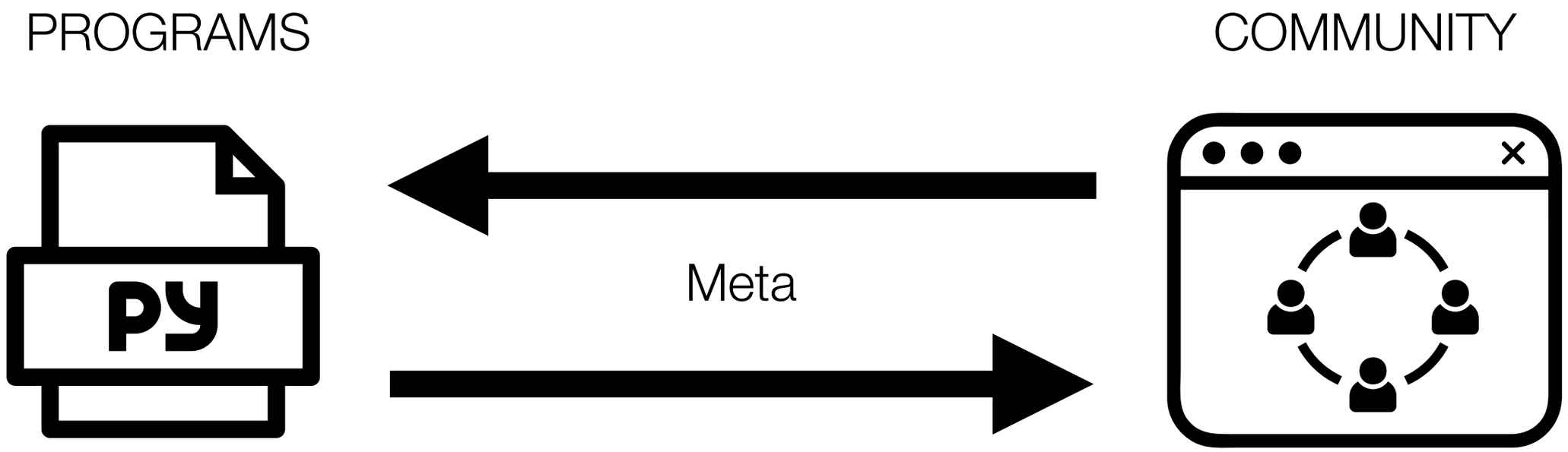

Codex (Fast et al., CHI 2014)
Blueprint (Brandt et al., CHI 2011)

# What can programming languages learn from how users write programs?




Codex (Fast et al., CHI 2014)
Blueprint (Brandt et al., CHI 2010)
HelpMeOut (Hartmann et al., CHI 2010)

# What can programming languages learn from how users write programs?




Meta is a language extension for Python that allows users to share functions and records their behavior

```
Meta.search(…)
```

```
Meta.search(…)

MetaFunction.optimize()
```

```
Meta.search(…)

MetaFunction.optimize()

MetaFunction.auto_patch()
```

```
Meta.search(…)

MetaFunction.optimize()

MetaFunction.auto_patch()

MetaFunction.get_type()

MetaFunction.examples()
```

```
Meta.search(…)

MetaFunction.optimize()

MetaFunction.auto_patch()

MetaFunction.get_type()

MetaFunction.examples()

MetaFunction.bugs()
```

# DEMO

Documentation (types and examples)

Search (leverage types and examples)

Optimization (make code faster)

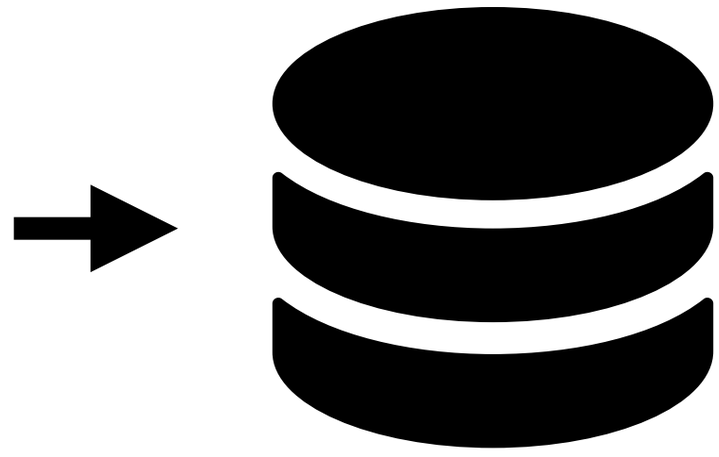
Patching (recover from run-time bugs)

# System Architecture

## CREATING AND LOADING META FUNCTIONS

`@meta(...)`

function name
source code
imported libraries



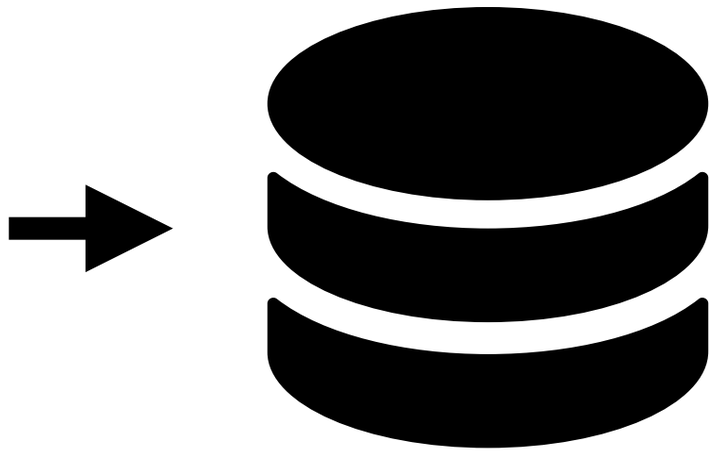
COMMUNITY DB

# System Architecture

## CREATING AND LOADING META FUNCTIONS

`@meta(...)`

function name
source code
imported libraries



COMMUNITY DB

`meta.load(...)`

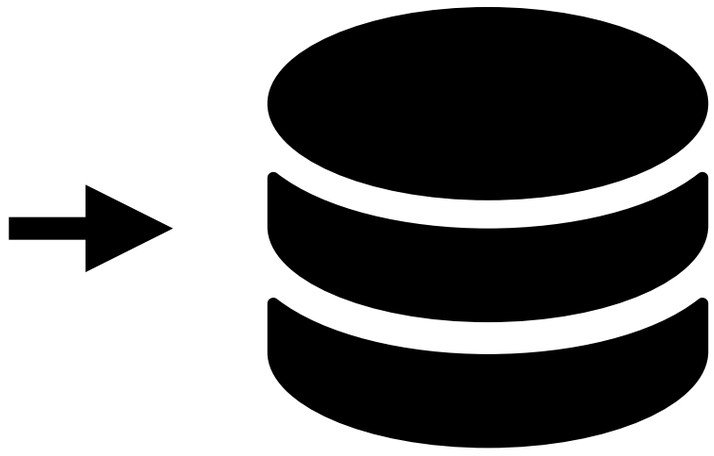
load libraries
execute source

# System Architecture

## CREATING AND LOADING META FUNCTIONS

`@meta(...)`

function name
source code
imported libraries



COMMUNITY DB

`meta.load(...)`

load libraries
execute source

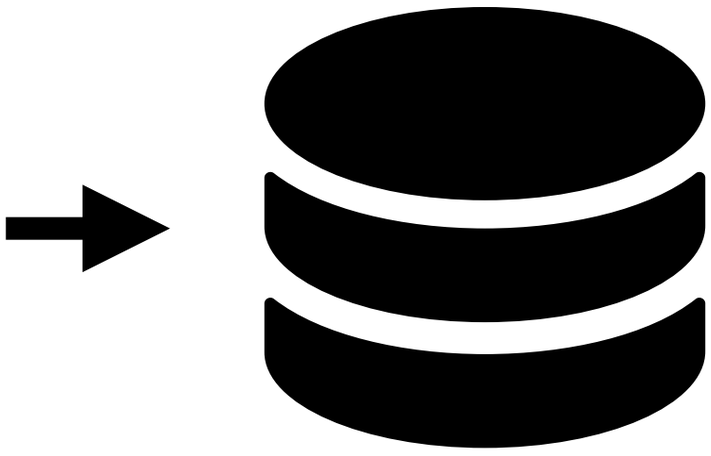
Detect IO by instrumenting core system libraries, no global state

# System Architecture

CREATING AND LOADING META FUNCTIONS

`@meta(...)`

function name
source code
imported libraries



COMMUNITY DB

`meta.load(...)`

load libraries
execute source

Detect IO by instrumenting core system libraries, no global state

Cache loaded functions locally to speed things up and work outside network

# System Architecture

## LEVERAGING COMMUNITY RUN-TIME DATA

`metafunc(...)`

arguments
return value
types of data
execution time
run-time exceptions



COMMUNITY DB

# System Architecture

LEVERAGING COMMUNITY RUN-TIME DATA

`metafunc(...)`

arguments
return value
types of data
execution time
run-time exceptions



COMMUNITY DB

APIs

```
get_type()
examples()
optimize()
bugs()
auto_patch()
...
```

Sample arguments with P(x) = 1/n for better performance

# Evaluation

## STUDY OF PROFESSIONAL PROGRAMMERS

# Evaluation

STUDY OF PROFESSIONAL PROGRAMMERS

**7** programmers, machine learning task

**746** lines of code, **26**% Meta functions

Created **17** Meta functions, loaded **6** unique, executed **15,000** times

# Evaluation

STUDY OF PROFESSIONAL PROGRAMMERS

**7** programmers, machine learning task

**746** lines of code, **26**% Meta functions

Created **17** Meta functions, loaded **6** unique, executed **15,000** times

**Design concerns**: security, privacy, community

# Evaluation

## PERFORMANCE COST

# Evaluation

## PERFORMANCE COST

**47**ms to load Meta function

**133**ms to create Meta function

**0.31**ms of overhead to run Meta function

For comparison, sorting a list of 5000 words takes **27**ms

Note: creating and loading Meta functions are fixed costs

# Evaluation

## EFFECTIVENESS OF COMMUNITY OPTIMIZATION

# Evaluation

## EFFECTIVENESS OF COMMUNITY OPTIMIZATION

**44** correct optimizations and **7** incorrect

Median optimized function was **1.45** times faster

**1.15**-**2.91** interquartile range

# Evaluation

EFFECTIVENESS OF COMMUNITY OPTIMIZATION

**44** correct optimizations and **7** incorrect

Median optimized function was **1.45** times faster

**1.15**-**2.91** interquartile range

```
is_capitalized('abc') #=> False
str_is_number('abc') #=> False

# with only this datapoint, can conclude functions are equal
is_capitalized('abc') == str_is_number('abc')
```

# Future Work

# Future Work

How does a system like Meta scale as its community grows to hundreds of thousands of functions?

# Future Work

How does a system like Meta scale as its community grows to hundreds of thousands of functions?

What other features can we design when programming languages learn from how people code?

# Future Work

How does a system like Meta scale as its community grows to hundreds of thousands of functions?

What other features can we design when programming languages learn from how people code?

Is it possible to extend a system like Meta to work beyond pure functions, handling global variables and other kinds of complex state?

# Conclusion

Meta envisions a world where an enormous amount of data about how people program is available for analysis, enabling a new class of intelligent programming environments

Check out www.meta-lang.org/tutorial