Find, Fix, Explain

Using machine learning to help with novice programmer mistakes

Ben Cosman

Yao-Yuan Yang

Leon Medvinsky

Current work

Prof. Kamalika Chaudhuri

Prof. Westley Weimer

Prof. Ranjit Jhala

The Problem

Finding and fixing bugs is hard,
especially for newer programmers

```
def double(x):
    result = x*2

y = double(3)
z = y + 4
```

```
def double(x):
    result = x*2

y = double(3)
z = y + 4
```

```
TypeError: unsupported operand type(s) for +: 'NoneType' and 'int'
```

```
def double(x):
    result = x*2

y = double(3)
z = y + 4
```

```
TypeError: unsupported operand type(s) for +: 'NoneType' and 'int'
```

Error messages can:

- blame the wrong line

```
def double(x):
    result = x*2

y = double(3)
z = y + 4
```

```
TypeError: unsupported operand type(s) for +: 'NoneType' and 'int'
```

Error messages can:

- blame the wrong line
- be difficult to interpret

The Goal

An analysis that can point to
the true source of a bug

The Goal

An analysis that can point to
the true source of a bug

(...and **fix** it too)

The Goal

An analysis that can point to
the true source of a bug

(...and **fix** it too)

(...and **explain** what was wrong)

Related work

Related work

Refazer

- Learns rewrite rules from repetitive edits
- Can find/fix bugs, but bad at generalizing

Related work

Refazer

- Learns rewrite rules from repetitive edits
- Can find/fix bugs, but bad at generalizing

"Naturalness Bug Finder"

- Creates language model from correct code
- Flags unnatural code as suspicious

Approach: Machine Learning

Approach: Machine Learning

We have lots of data

Approach: Machine Learning

We have lots of data

Recent work shows this approach is promising

Related work: NATE

Finds type errors in OCaml

Dataset: homework from CSE 130

Overview

Overview

Buggy program:

```
def double(x):
    result = x*2

y = double(3)
z = y + 4
```

Fixed program:

```
def double(x):
    return x*2

y = double(3)
z = y + 4
```

Overview

Buggy program:

```
def double(x):
    result = x*2

y = double(3)
z = y + 4
```

Features

1	0	1	1	0
0	0	1	0	1
1	0	1	0	0

Fixed program:

```
def double(x):
    return x*2

y = double(3)
z = y + 4
```

Overview

Buggy program:

```
def double(x):
    result = x*2

y = double(3)
z = y + 4
```

Features

1	0	1	1	0	?
0	0	1	0	1	?
1	0	1	0	0	?

Label: is it the bug?

Fixed program:

```
def double(x):
    return x*2

y = double(3)
z = y + 4
```

Overview

Buggy program:

```
def double(x):
    result = x*2

y = double(3)
z = y + 4
```

Features

1	0	1	1	0
0	0	1	0	1
1	0	1	0	0

~~Label: is it the bug?~~

Yes

No

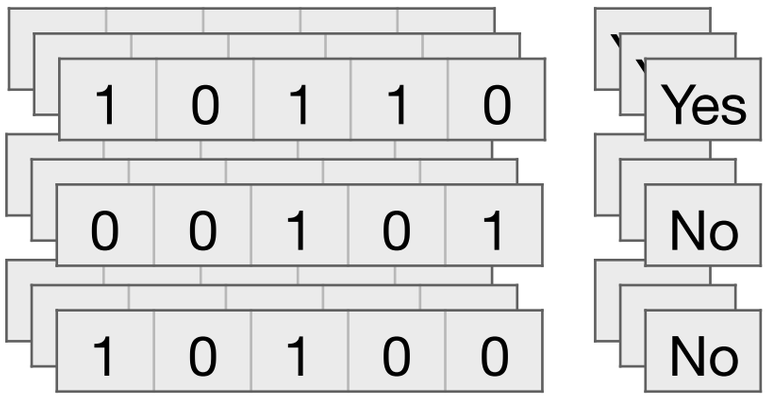
No

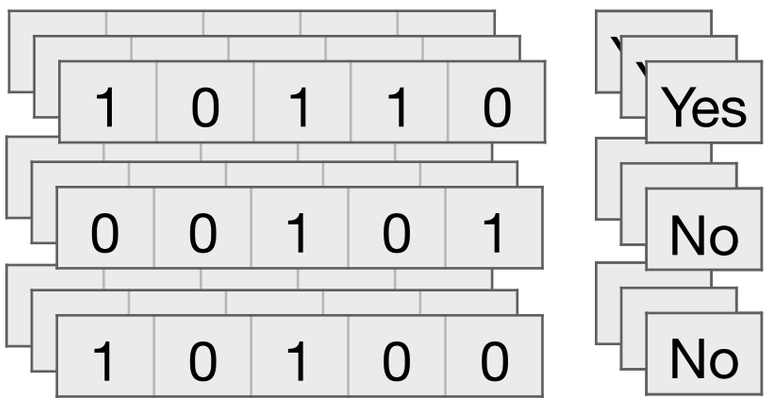
Label: did it change?

Fixed program:

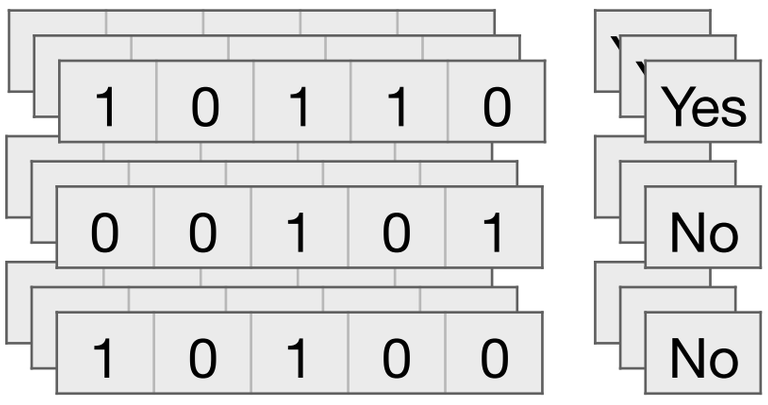
```
def double(x):
    return x*2

y = double(3)
z = y + 4
```

Overview



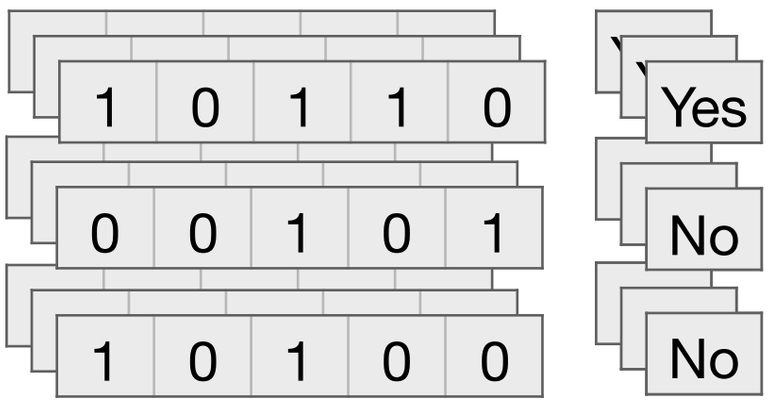


Overview

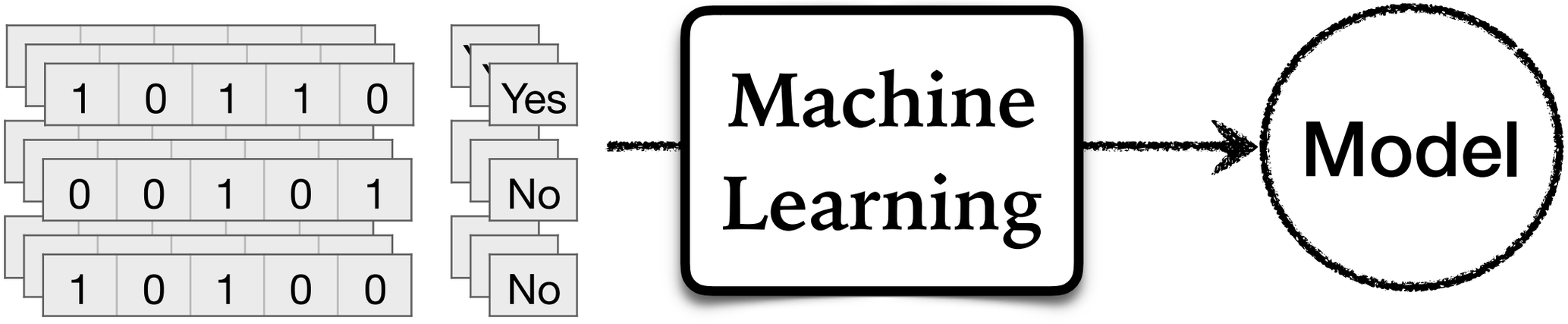


Training

Testing

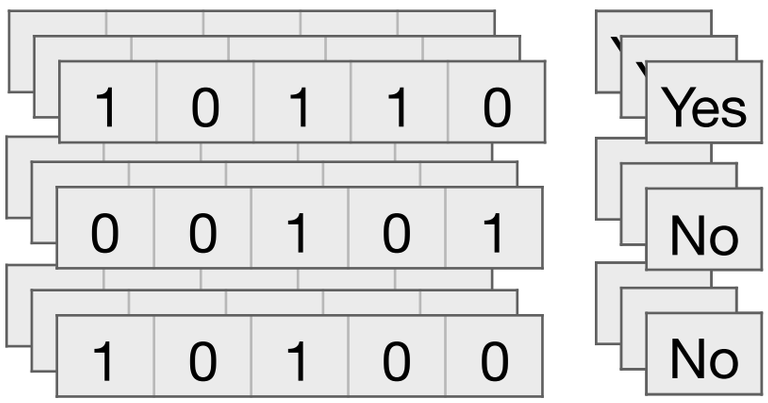


Overview

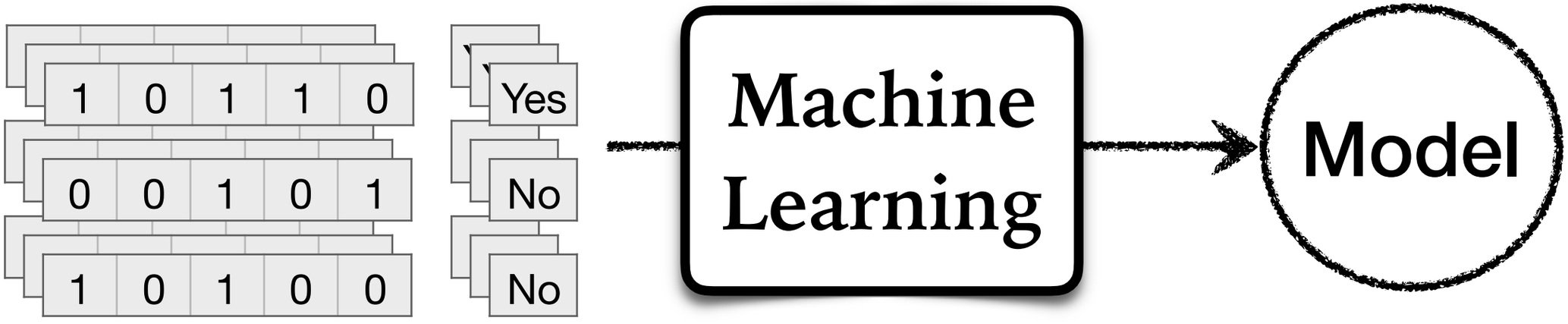


Training

Testing

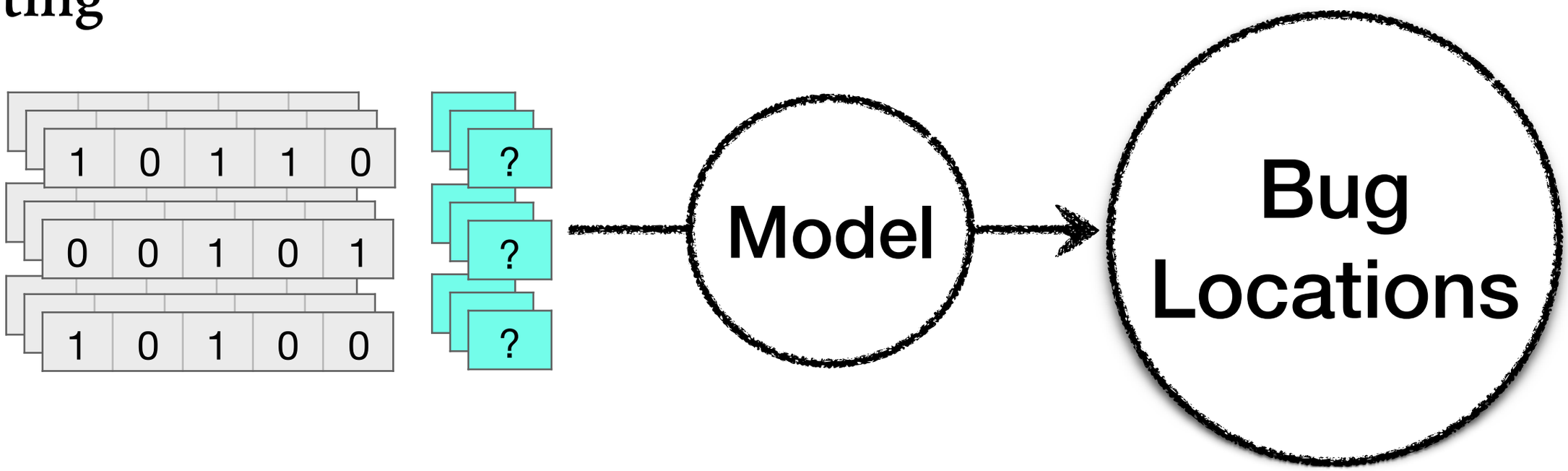


Overview

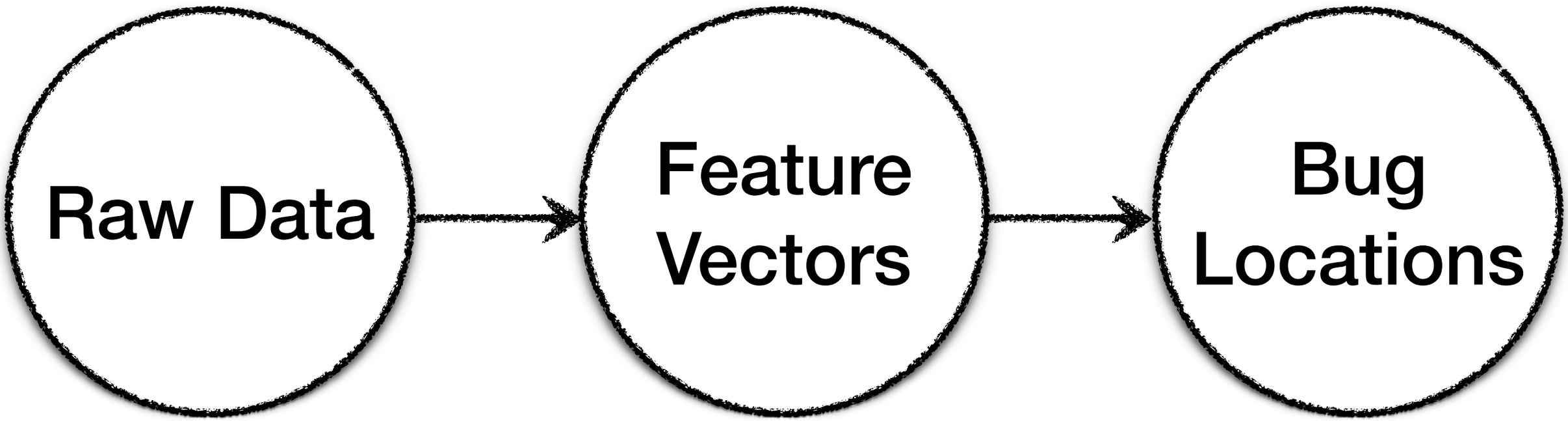


Training

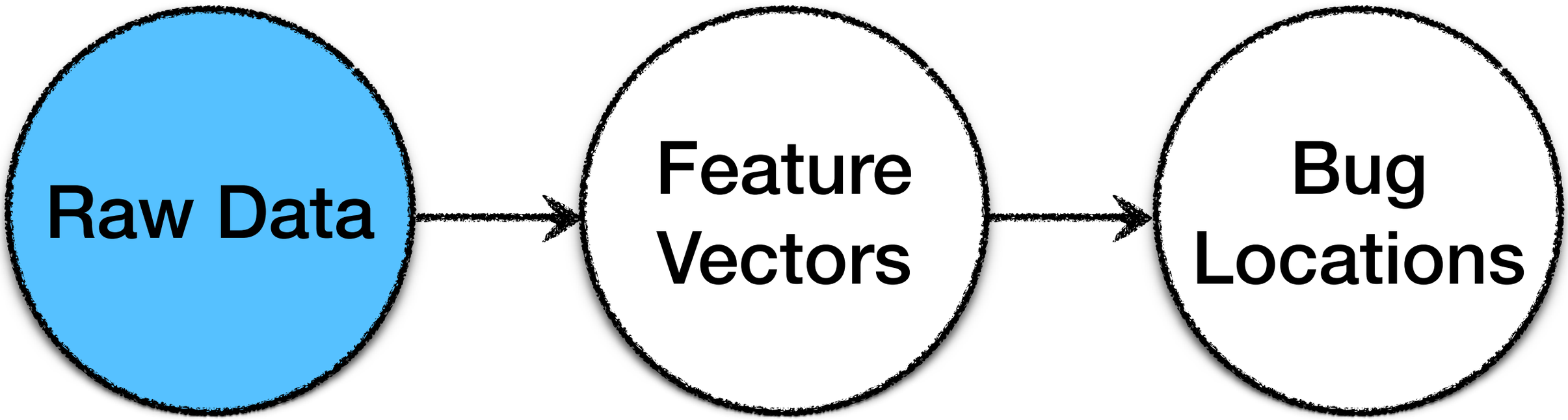
Testing



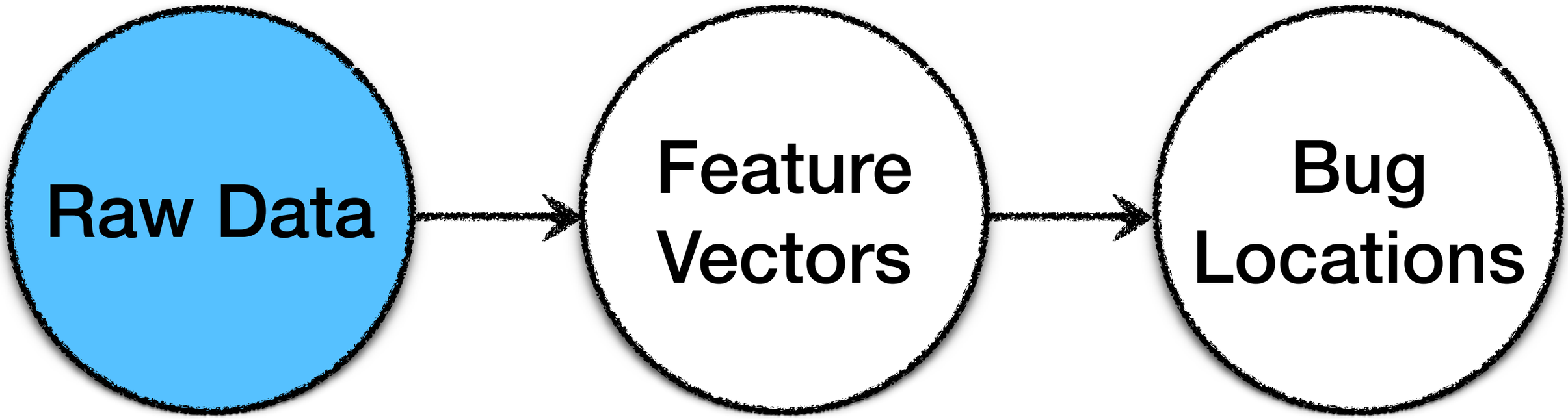
Outline



Outline



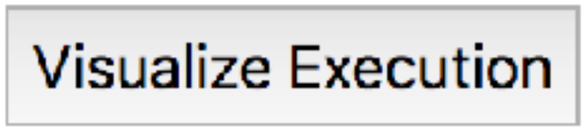
Outline



- Source
- Pairing
- Challenges

Dataset: Submissions to PythonTutor

Dataset: Submissions to PythonTutor

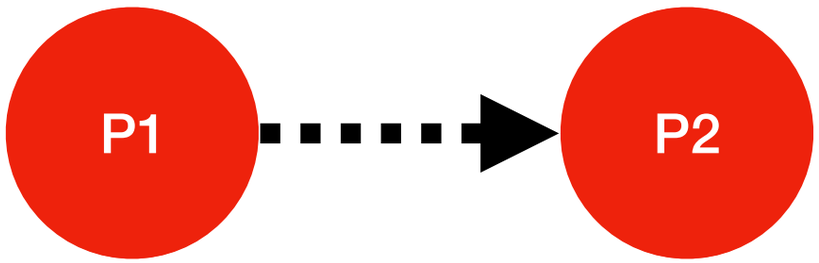
One entry for each click of 

in 2017

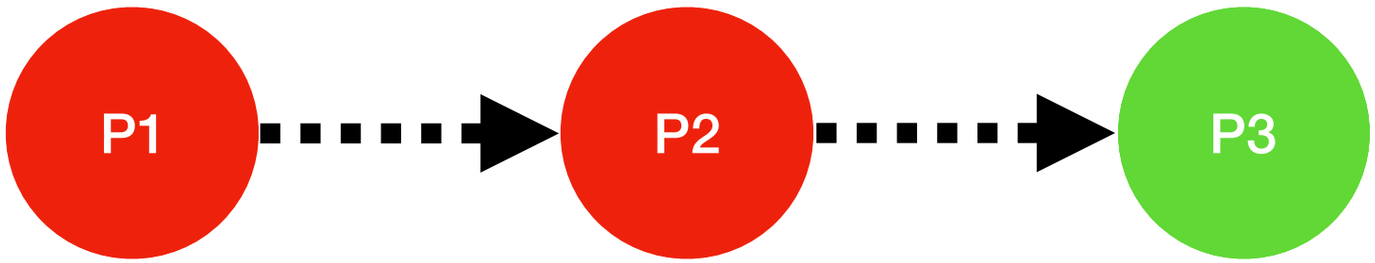
~5 million programs total

User submissions:

User submissions:



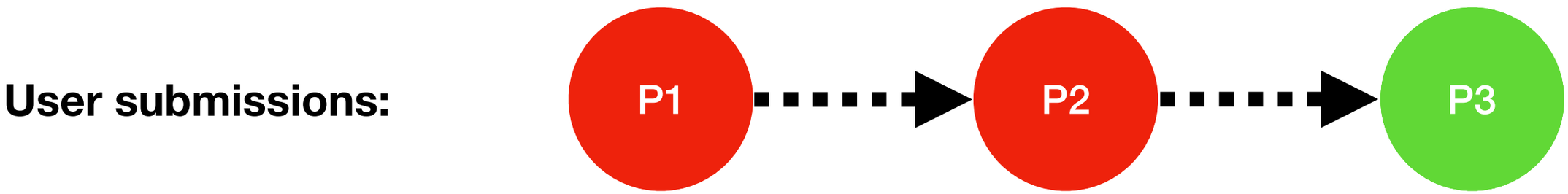
User submissions:



Each program that crashes

is paired with

the next program that works

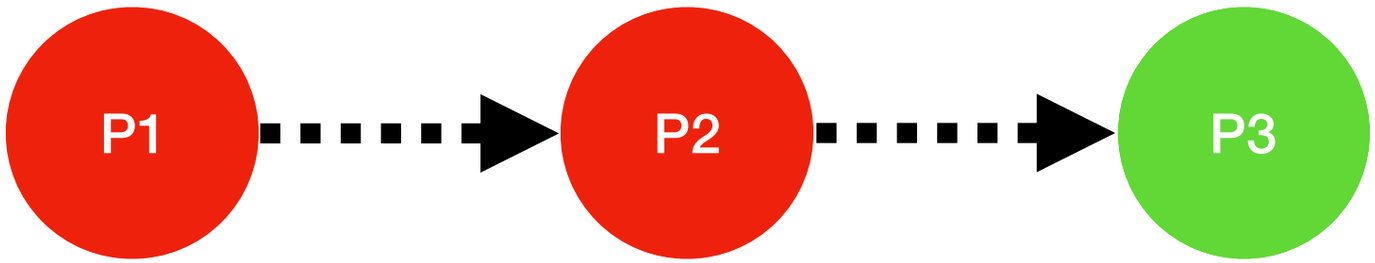


Each program that crashes

is paired with

the next program that works

User submissions:



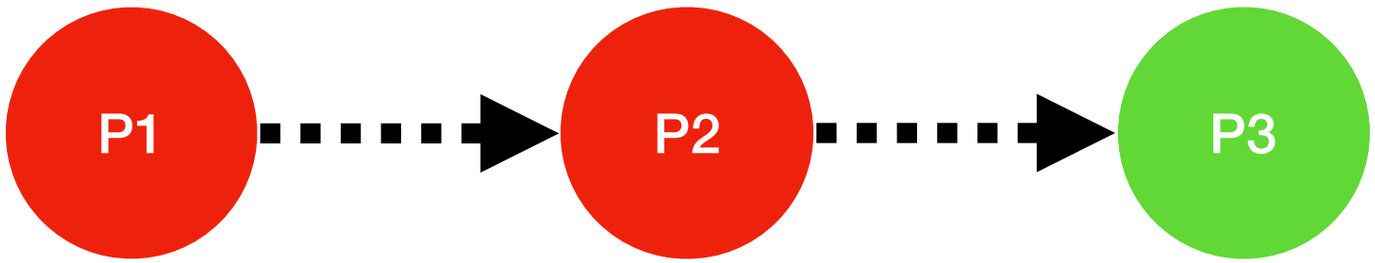
Data:

Each program that crashes

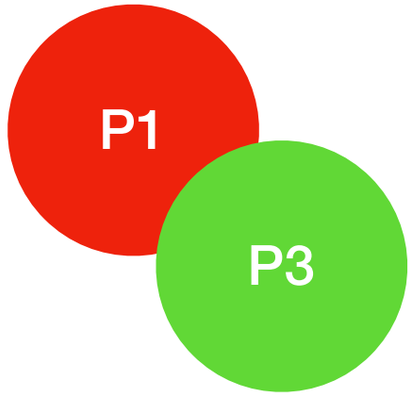
is paired with

the next program that works

User submissions:



Data:

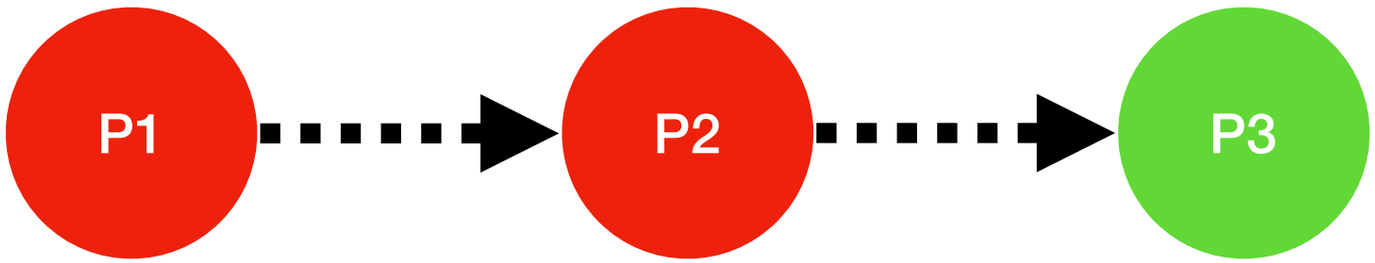


Each program that crashes

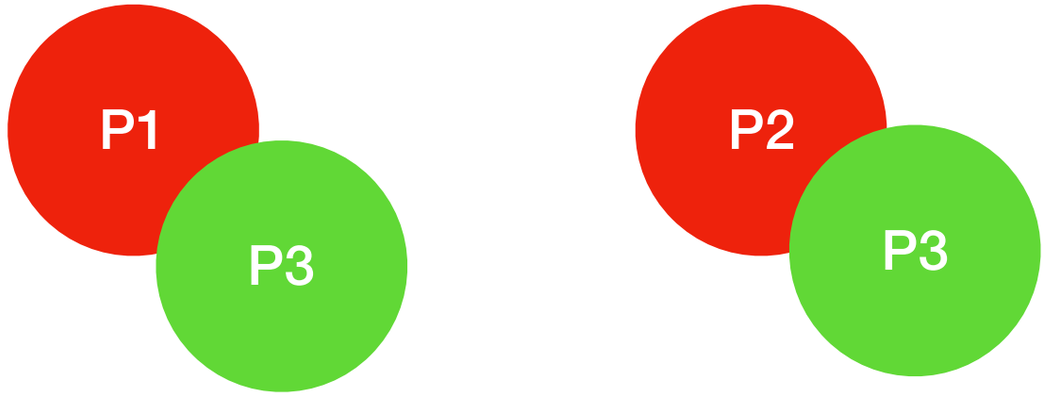
is paired with

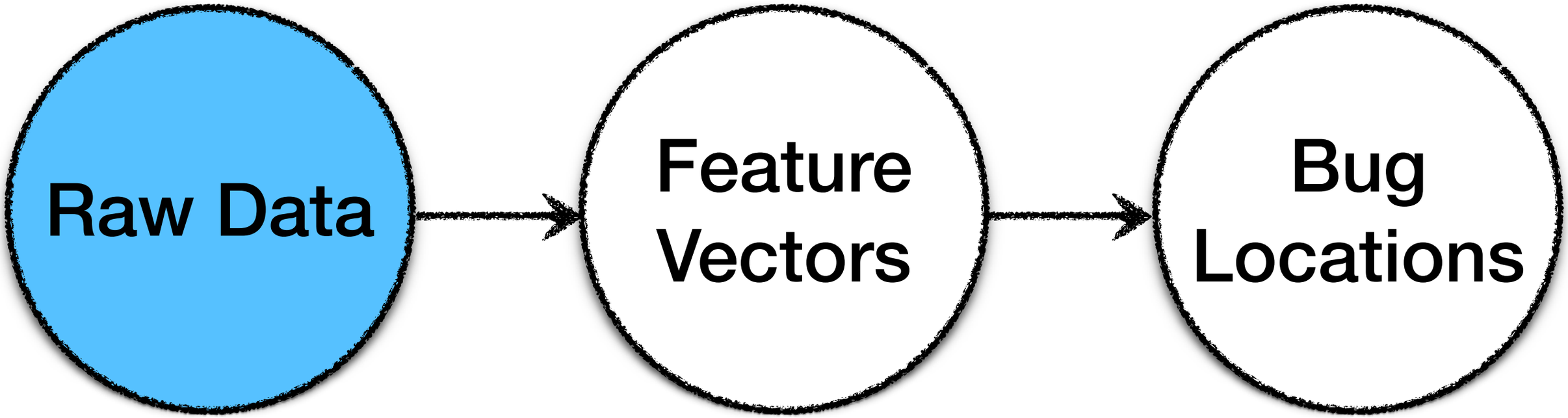
the next program that works

User submissions:

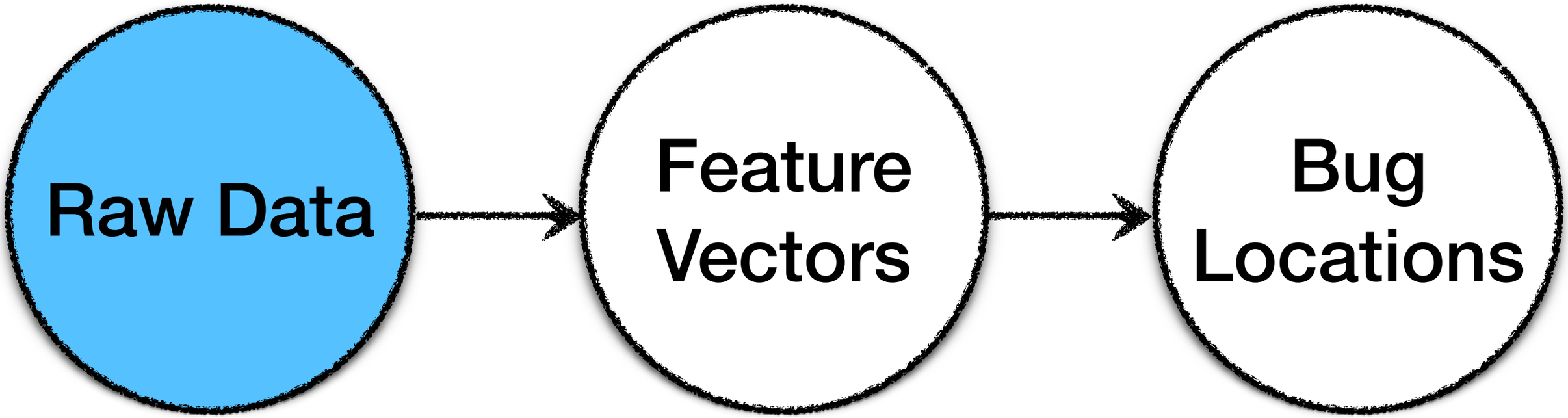


Data:

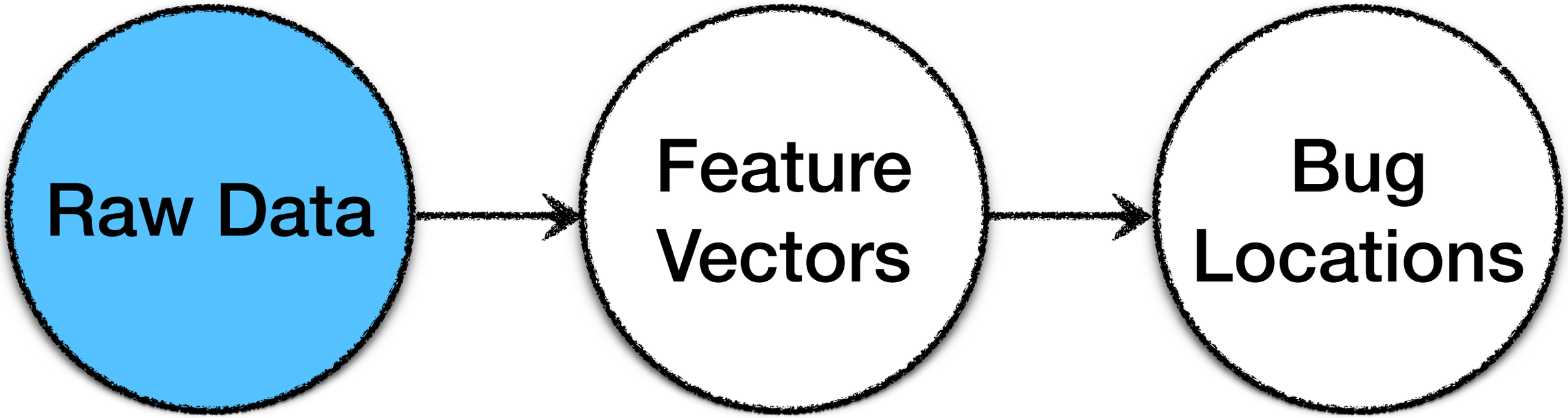




- Source
- Pairing
- Challenges



- Source
- Pairing
- Challenges



- Source
- Pairing
- Challenges

- Complete rewrites
- False fixes
- Variety of errors

No idea what the user was *trying* to write

- Crash = Buggy
- No crash = Fixed

Complete rewrites

- Crash = Buggy
- No crash = Fixed

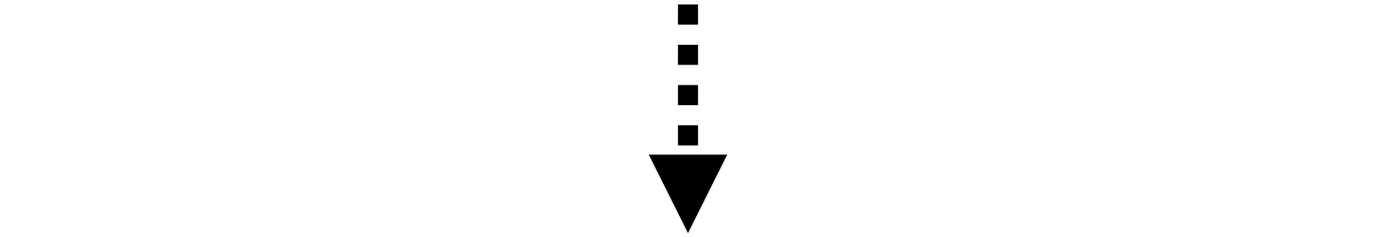
Complete rewrites

```
i = 0
while i < 10:
    doBuggyThing()
```

- Crash = Buggy
- No crash = Fixed

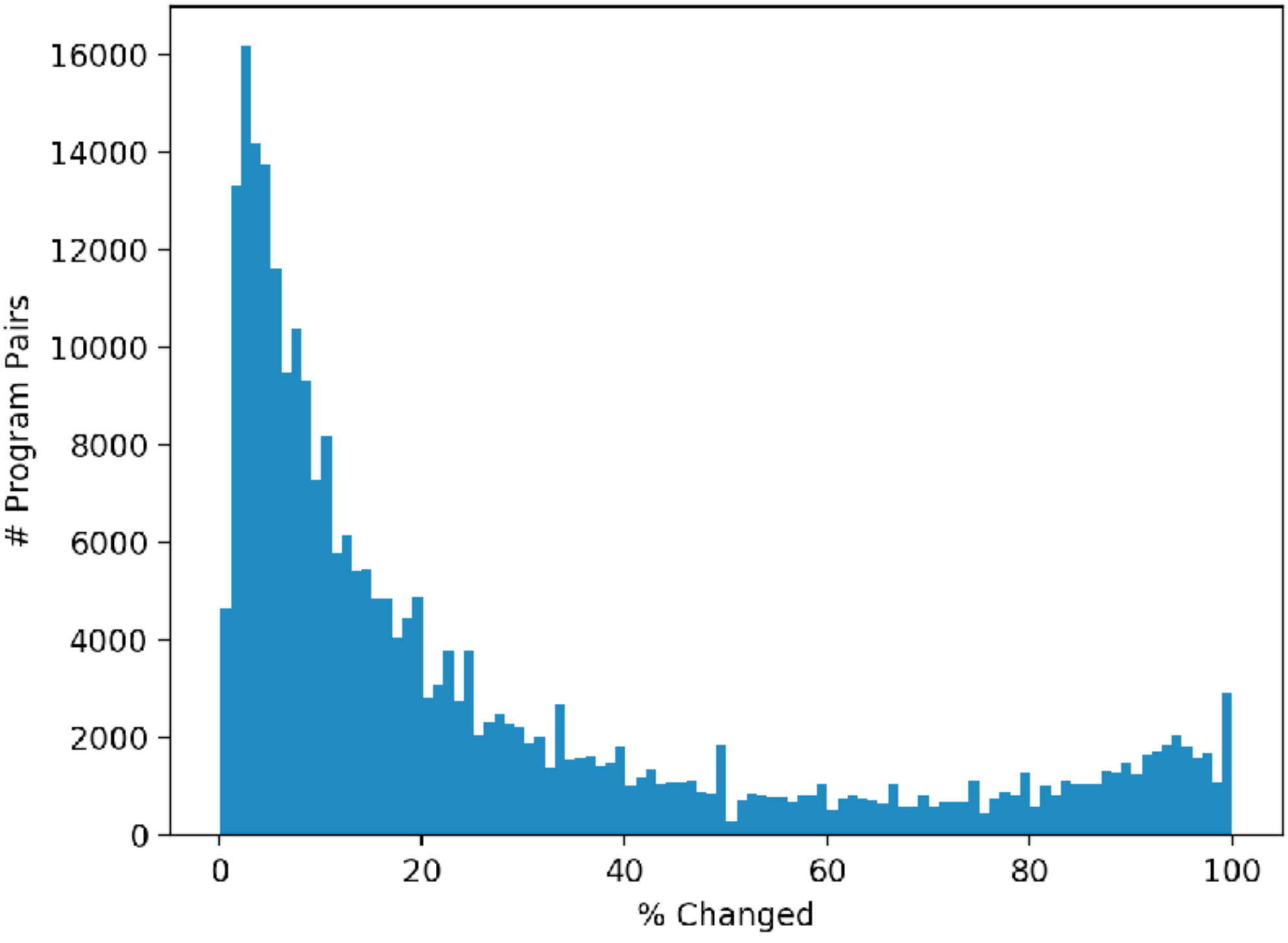
Complete rewrites

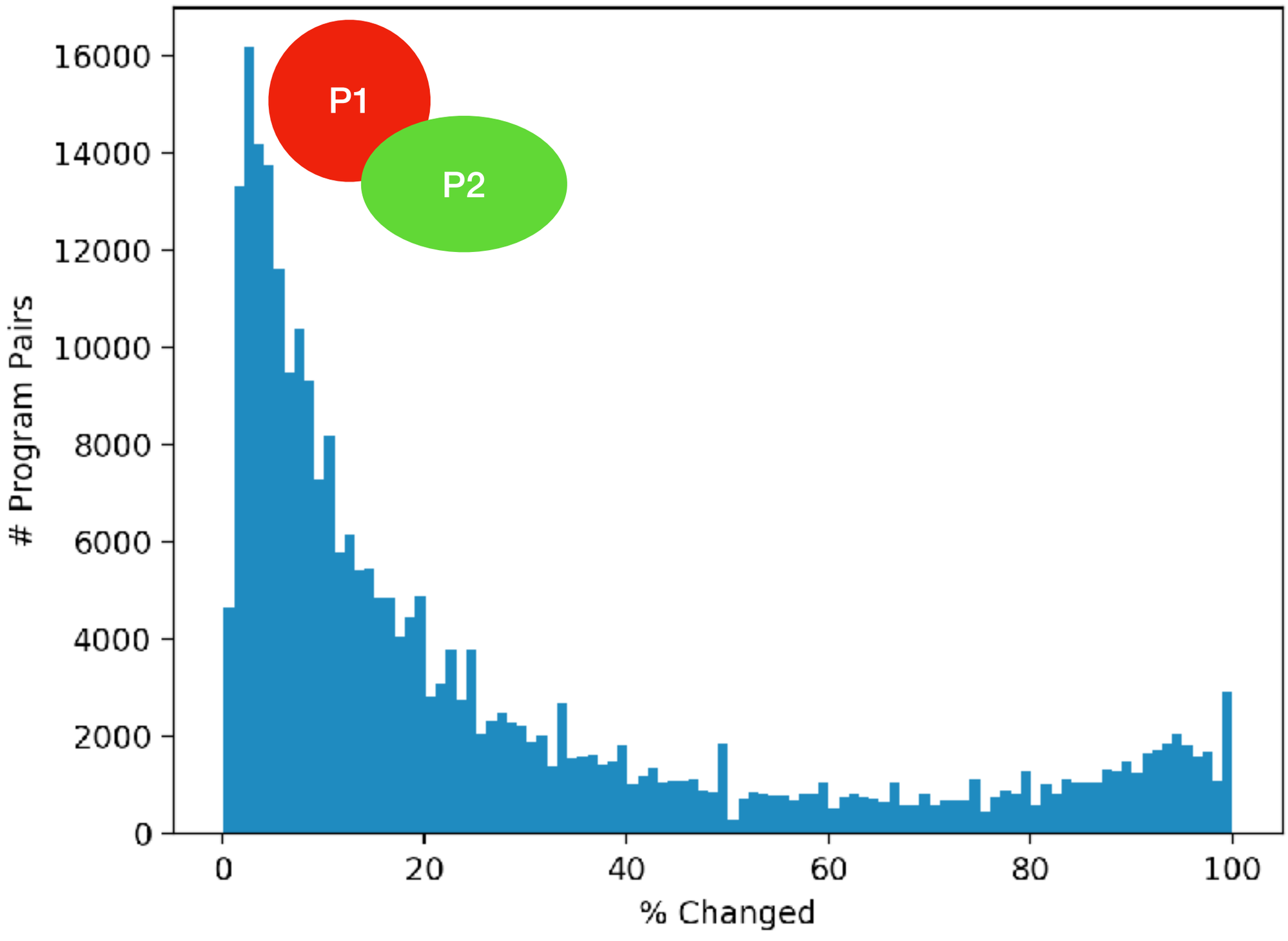
```
i = 0
while i < 10:
    doBuggyThing()
```

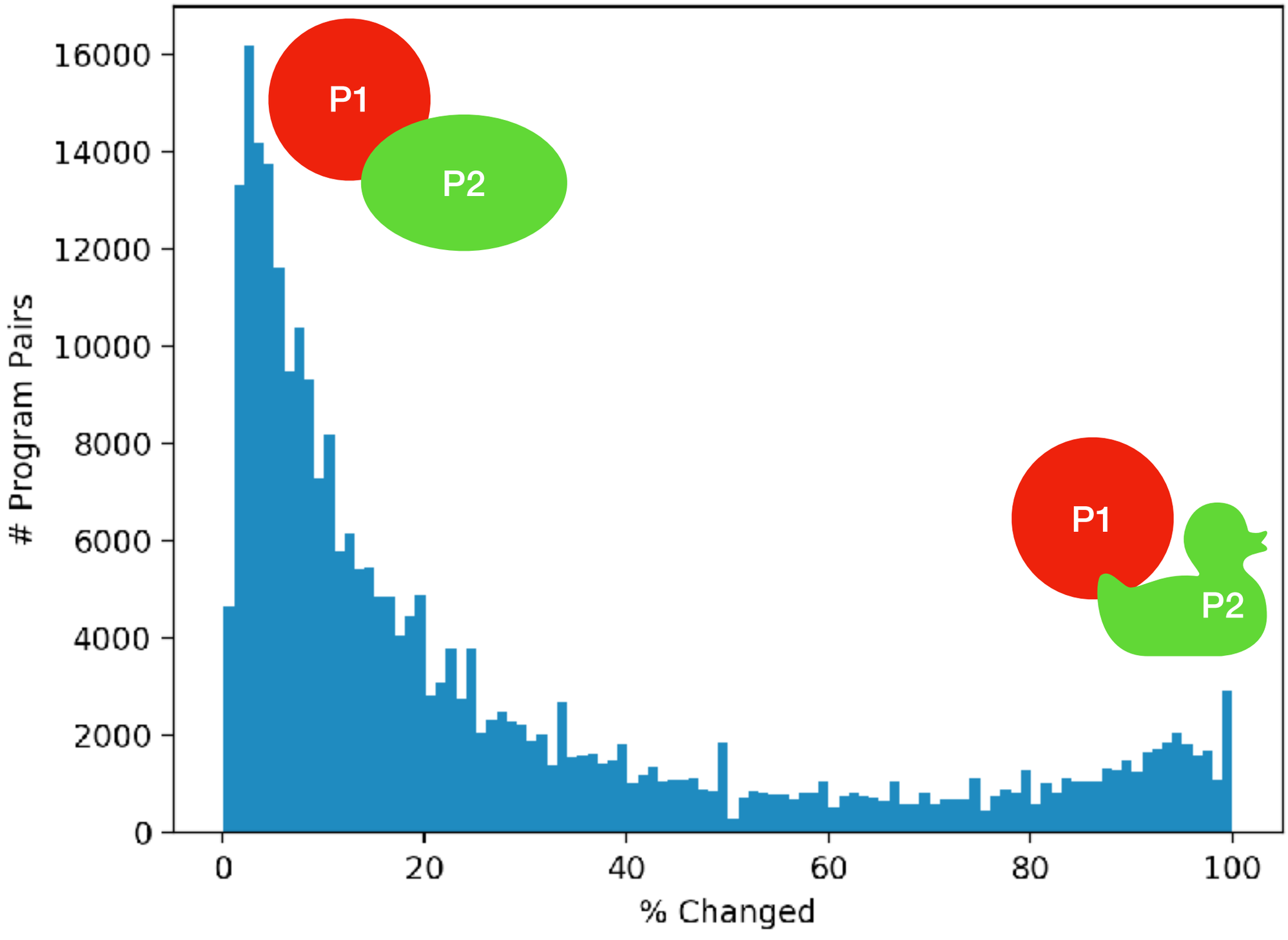


```
print("Hello, world!")
```

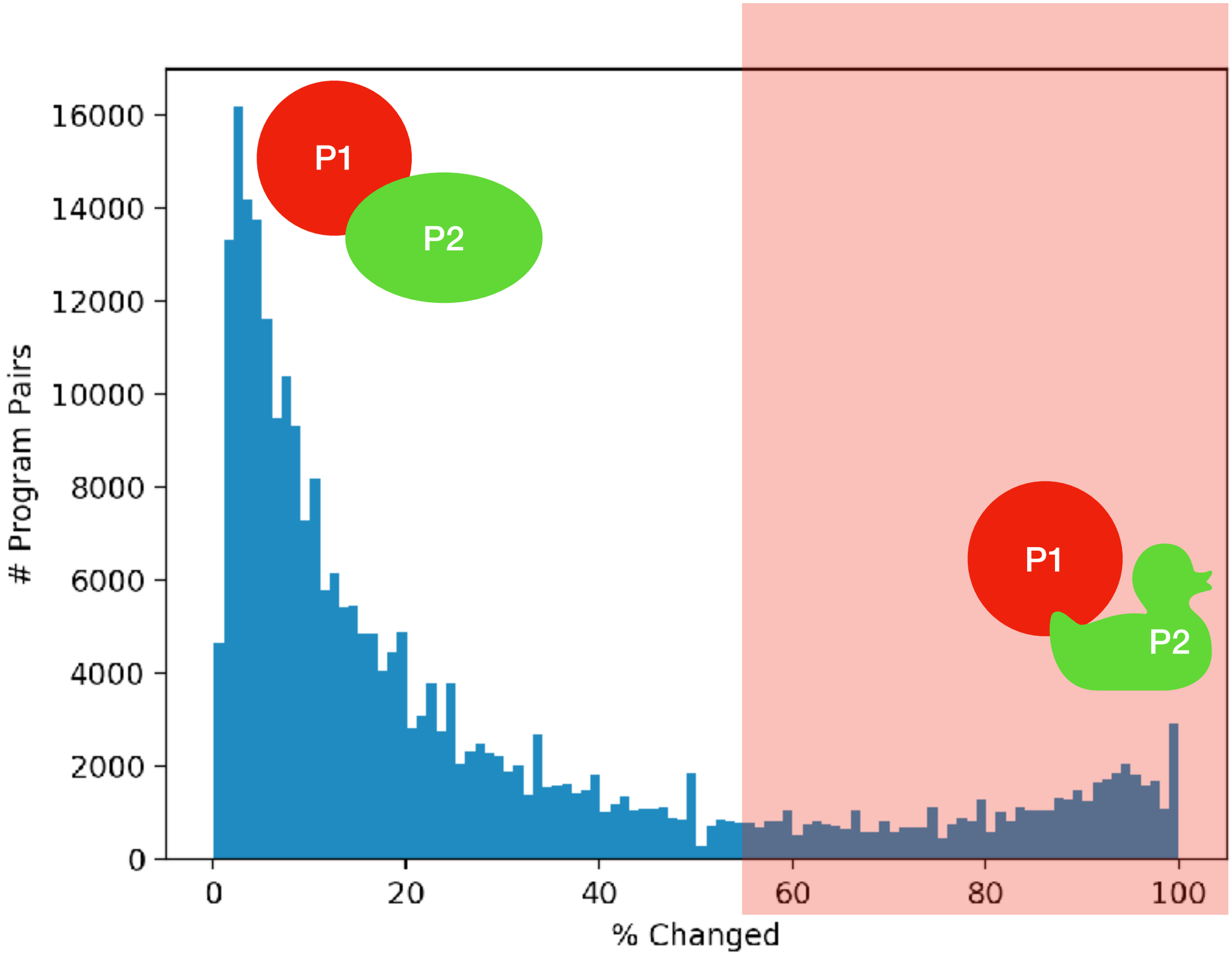
- Crash = Buggy
- No crash = Fixed ??


16000
14000
12000
10000
8000
6000
4000
2000
0
Program Pairs
0
20
40
60
80
100
% Changed


P1
P2
16000
14000
12000
10000
8000
6000
4000
2000
0
Program Pairs
0
20
40
60
80
100
% Changed


16000
14000
12000
10000
8000
6000
4000
2000
0
Program Pairs
0
20
40
60
80
100
% Changed
P1
P2
P1
P2

Throw out program pairs with too much changed



False fixes

- Crash = Buggy
- No crash = Fixed

False fixes

```
i = 0
while i < 10:
    doBuggyThing()
```

- Crash = Buggy
- No crash = Fixed

False fixes

```
i = 0
while i < 10:
    doBuggyThing()
```

```
i = 0
while i > 10:
    doBuggyThing()
```

- Crash = Buggy
- No crash = Fixed ??

Variety of errors

- TypeError
- ZeroDivisionError
- SyntaxError
- NameError
- etc

Variety of errors

- TypeError ✓
- ZeroDivisionError
- SyntaxError
- NameError
- etc

Variety of errors

- TypeError ✓
- ZeroDivisionError ✓
- SyntaxError
- NameError
- etc

Variety of errors

- TypeError ✓
- ZeroDivisionError ✓
- SyntaxError
- NameError
- etc ✓

Variety of errors

- TypeError ✓
- ZeroDivisionError ✓
- SyntaxError ✗
- NameError
- etc ✓

Variety of errors

- TypeError 
- ZeroDivisionError 
- SyntaxError 
- NameError
- etc 

Feature extraction requires:

- building an AST
- running the code

Variety of errors

- TypeError ✓
- ZeroDivisionError ✓
- SyntaxError ✗
- NameError
- etc ✓

Variety of errors

- TypeError 
- ZeroDivisionError 
- SyntaxError 
- NameError 
- etc 

Variety of errors

- TypeError 
- ZeroDivisionError 
- SyntaxError 
- NameError 
- etc 

Python's error message is good enough

Variety of errors

- TypeError 
- ZeroDivisionError 
- SyntaxError 
- NameError 
- etc 

Python's error message is good enough

Unusually difficult to guess where to fix

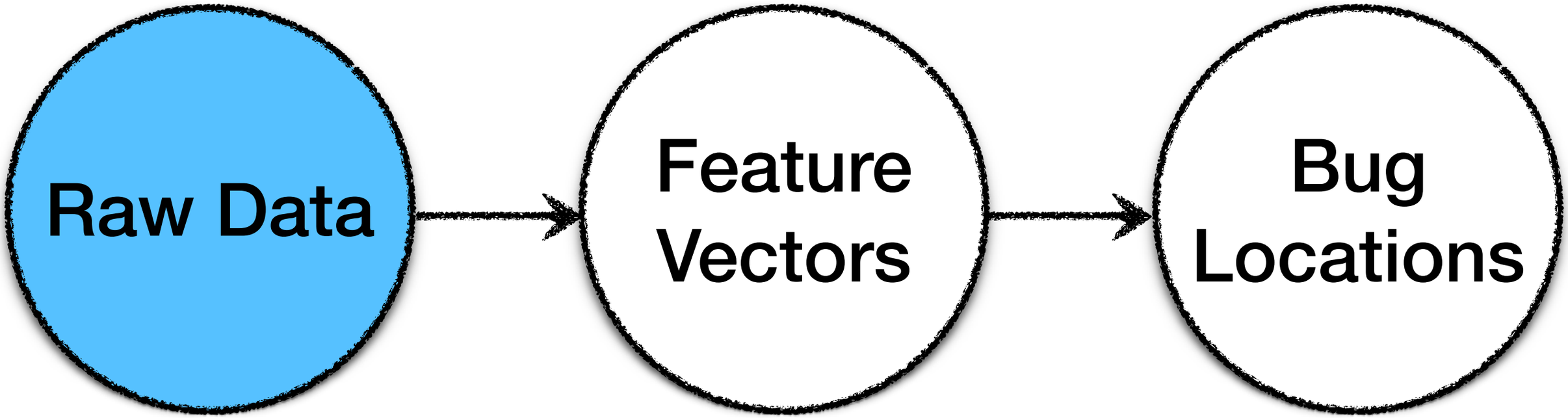
Variety of errors

17% of data {

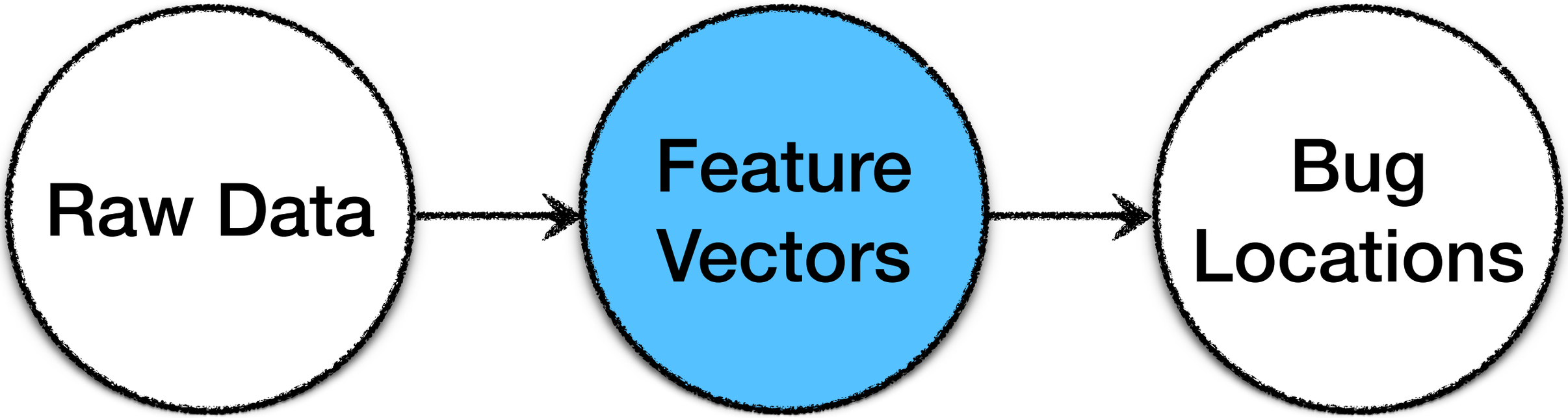
- TypeError 
- ZeroDivisionError 
- SyntaxError 
- NameError 
- etc 

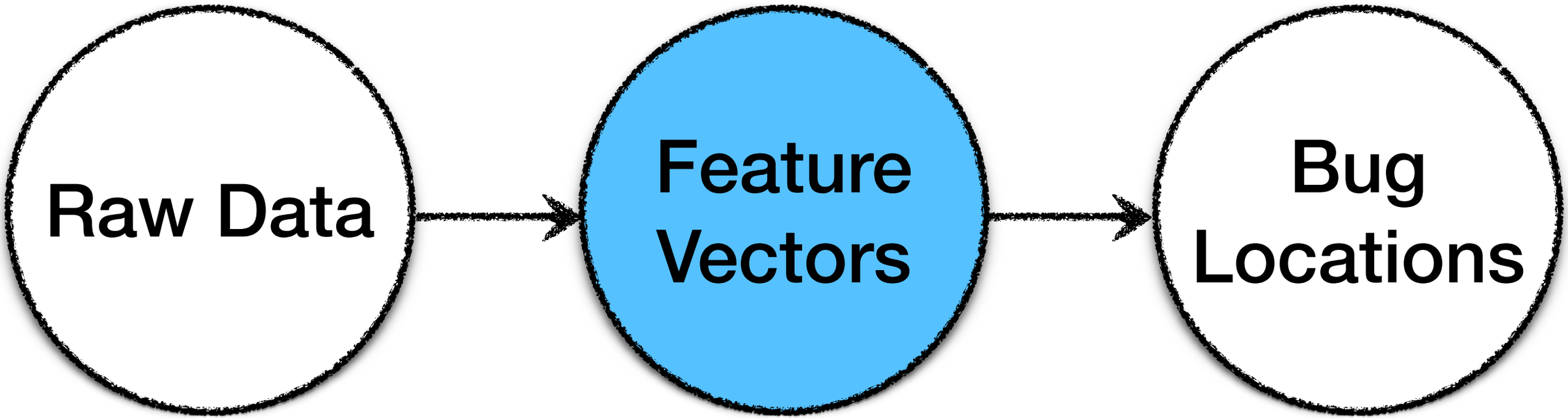
Python's error message is good enough

Unusually difficult to guess where to fix

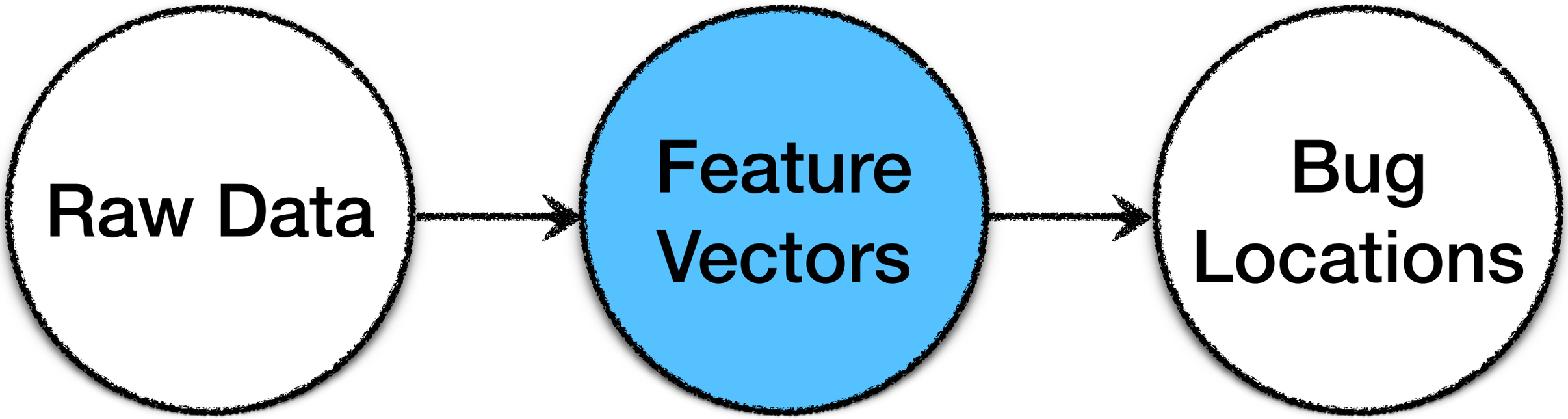


- Source
- Pairing
- Challenges

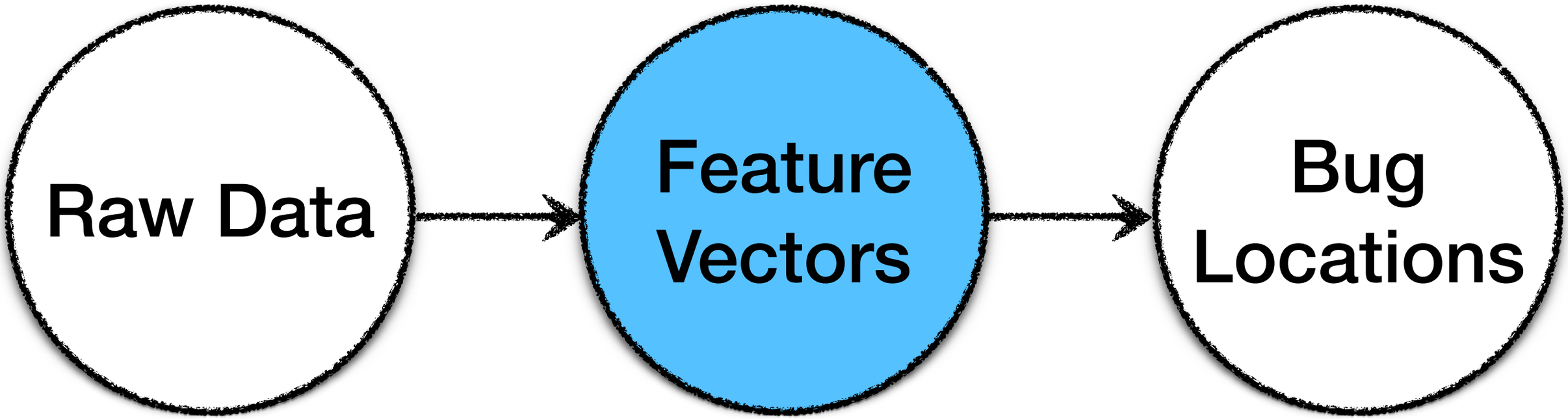

Raw Data
Feature Vectors
Bug Locations



- Static/Syntactic



- Static/Syntactic
- Dynamic



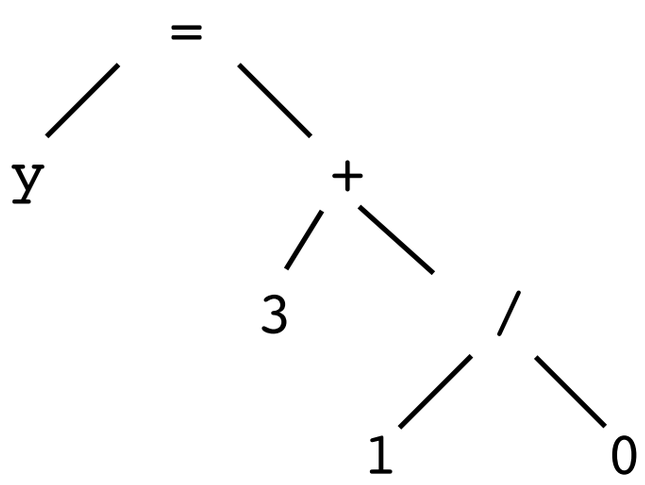
- Static/Syntactic
- Dynamic
- Contextual

Features

```
y = 3 + 1 / 0
```

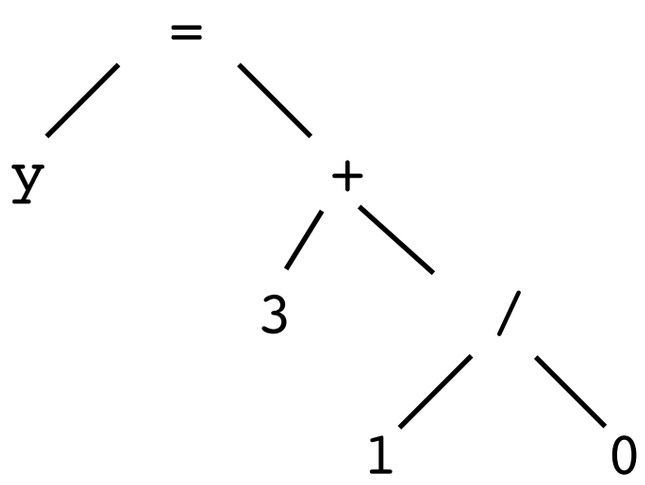
Features

```
y = 3 + 1 / 0
```



Features

```
y = 3 + 1 / 0
```



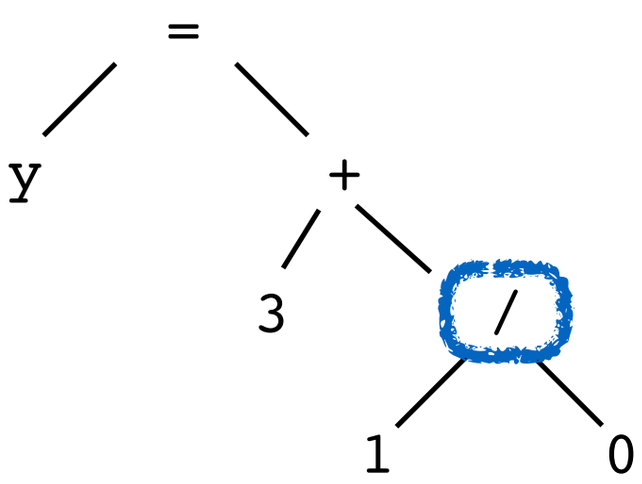
Terms

Features

…	…	…	…	…	…
…	…	…	…	…	…
…	…	…	…	…	…

Features

```
y = 3 + 1 / 0
```

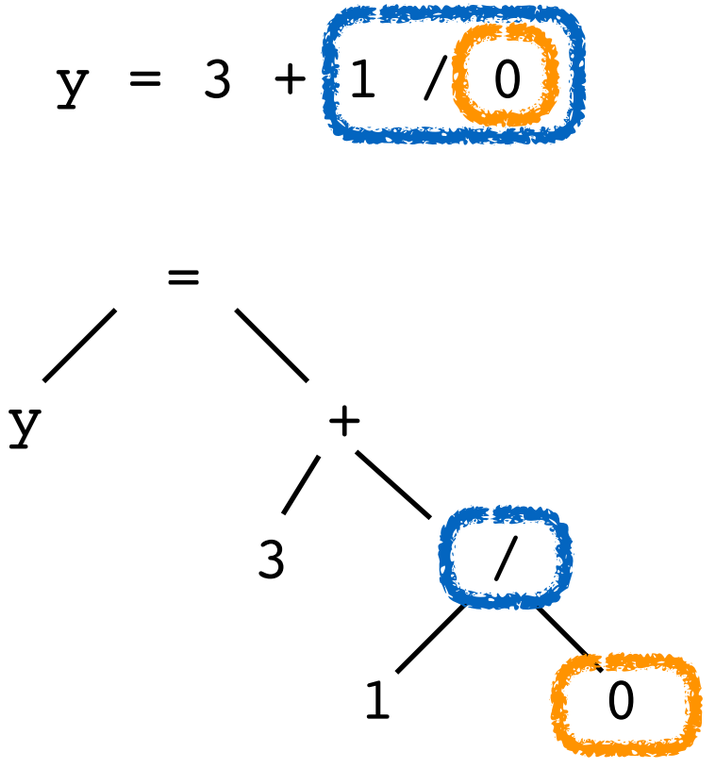


Terms

Features

…	…	…	…	…	…
…	…	…	…	…	…
…	…	…	…	…	…

Features



Terms

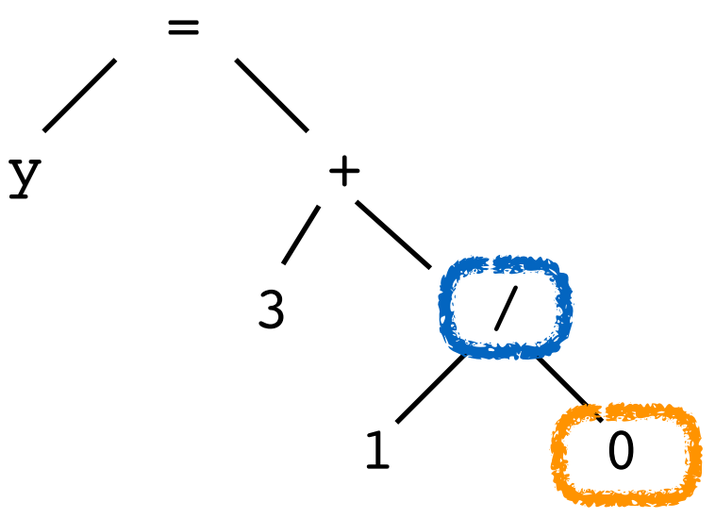
Features

…	…	…	…	…	…
…	…	…	…	…	…
…	…	…	…	…	…

Static/Syntactic features

What kind of node is it?

```
y = 3 + 1 / 0
```



Terms

Features

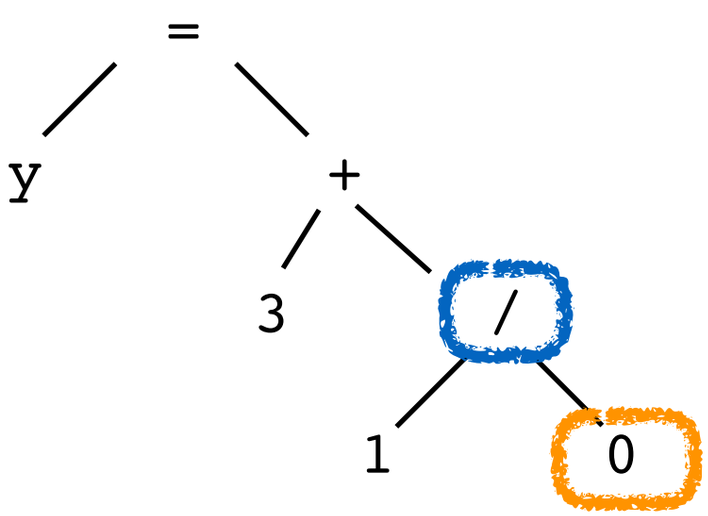
NodeKind	...	...	...	...	...
BinaryOp	...	...	...	...	...
IntLiteral	...	...	...	...	...

Static/Syntactic features

What kind of node is it?

What is the size of its AST subtree?

```
y = 3 + 1 / 0
```



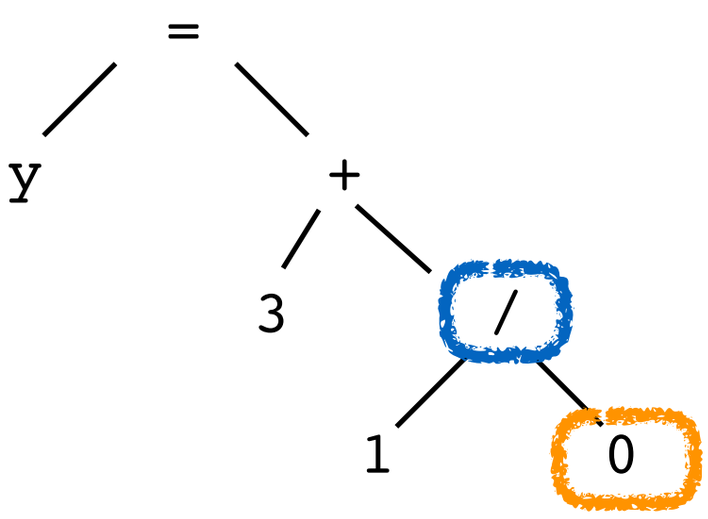
Terms

Features →

NodeKind	Size	...	...	...	...
BinaryOp	3	...	...	...	...
IntLiteral	1	...	...	...	...

Dynamic features

```
y = 3 + 1 / 0
```

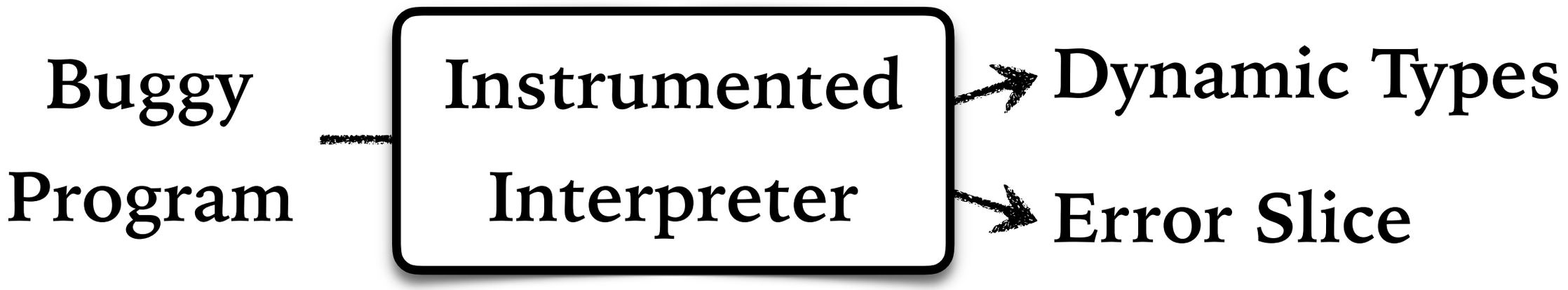


Terms

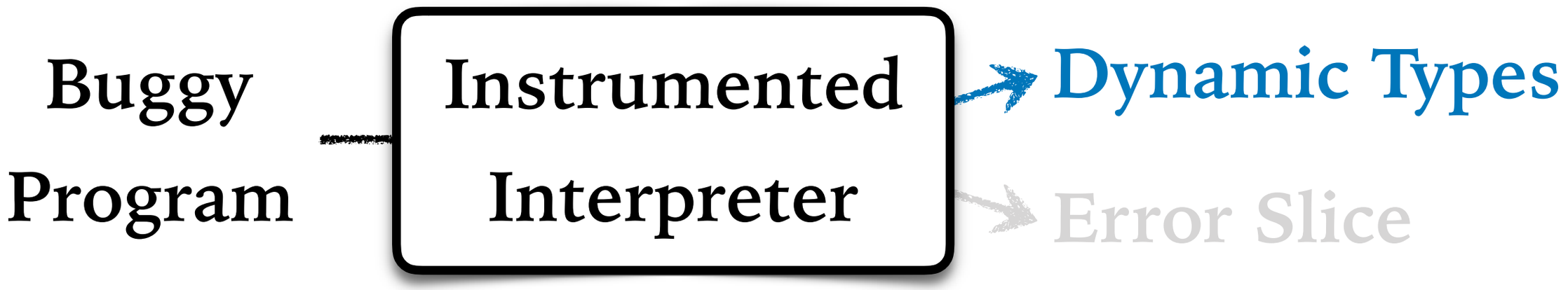
Features

NodeKind	Size	Type	Slice	Crash?	Msg
BinaryOp	3	…	…	…	…
IntLiteral	1	…	…	…	…

Dynamic features



Dynamic features



Code

```
x = "hi" * (3+2)
```

Code

```
x = "hi" * (3+2)
```

Environment

Variable	value	type

Code

Environment

Variable	value	type

```
x = "hi" * (3+2)
```

Code

```
x = "hi" * (3+2)
```

Environment

Variable	value	type
x	"hihihihihi"	string

Code

```
x = "hi" * (3+2)
```

Environment

Variable	value	type
x	"hihihihihi"	string

Problem: What's the type of (3+2)?

Code

```
x = "hi" * (3+2)
```

Environment

Variable	value	type
x	"hihihihihi"	string

Problem: What's the type of (3+2)?

Solution: Convert to ANF first

Code

Environment

```
a1 = 3
a2 = 2
a3 = (a1 + a2)
a4 = "hi"
x  = a4 * a3
```

```
x = "hi" * (1+2)
```

Code

Environment

Variable	value	type

```
a1 = 3
a2 = 2
a3 = (a1 + a2)
a4 = "hi"
x  = a4 * a3
```

```
x = "hi" * (1+2)
```

Code

```
a1 = 3
a2 = 2
a3 = (a1 + a2)
a4 = "hi"
x  = a4 * a3
```

Environment

Variable	value	type
a1	3	int

```
x = "hi" * (1+2)
```

Code

```
a1 = 3
a2 = 2
a3 = (a1 + a2)
a4 = "hi"
x  = a4 * a3
```

Environment

Variable	value	type
a1	3	int
a2	2	int

```
x = "hi" * (1+2)
```

Code

```
a1 = 3
a2 = 2
a3 = (a1 + a2)
a4 = "hi"
x  = a4 * a3
```

Environment

Variable	value	type
a1	3	int
a2	2	int
a3	5	int

```
x = "hi" * (1+2)
```

Code

```
a1 = 3
a2 = 2
a3 = (a1 + a2)
a4 = "hi"
x  = a4 * a3
```

Environment

Variable	value	type
a1	3	int
a2	2	int
a3	5	int
a4	"hi"	string

```
x = "hi" * (1+2)
```

Code

```
a1 = 3
a2 = 2
a3 = (a1 + a2)
a4 = "hi"
x  = a4 * a3
```

Environment

Variable	value	type
a1	3	int
a2	2	int
a3	5	int
a4	"hi"	string
x	"hihihihihi"	string

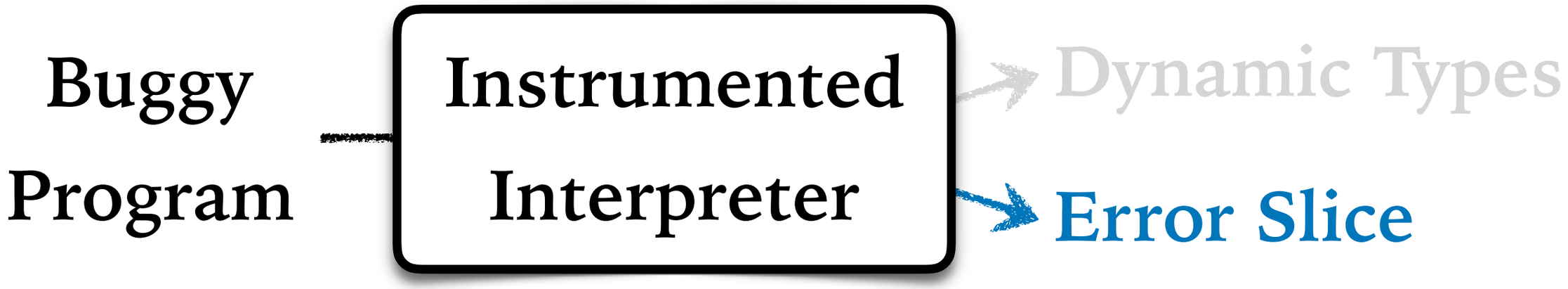
```
x = "hi" * (1+2)
```

Buggy Program

Instrumented Interpreter

Dynamic Types

Error Slice


Buggy
Program
Instrumented
Interpreter
Dynamic Types
Error Slice

```
w = 3

x = 0

y = w + 1

z = 6 / x
```

```
w = 3

x = 0

y = w + 1

z = 6 / x
```

```
ZeroDivisionError
```

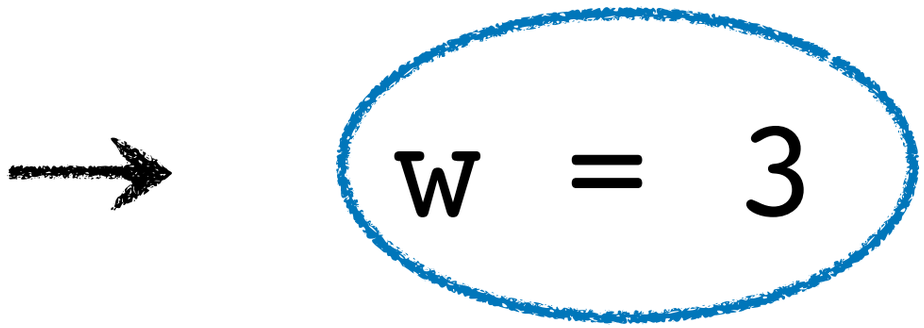
```
w = 3

x = 0

y = w + 1

z = 6 / x
```

```
ZeroDivisionError
```

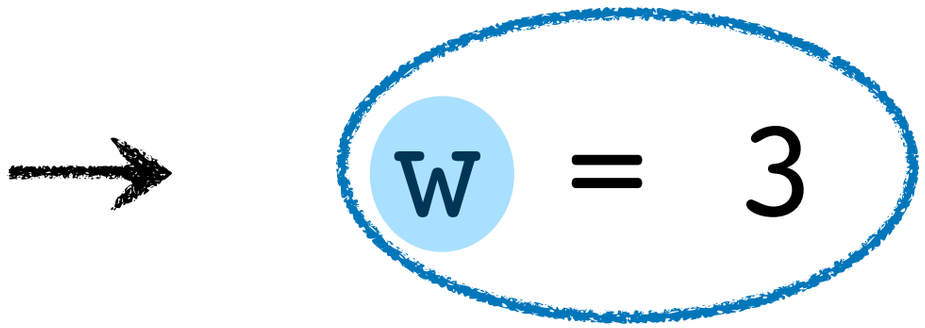
```
w = 3

x = 0

y = w + 1

z = 6 / x
```

```
w = 3

x = 0
```

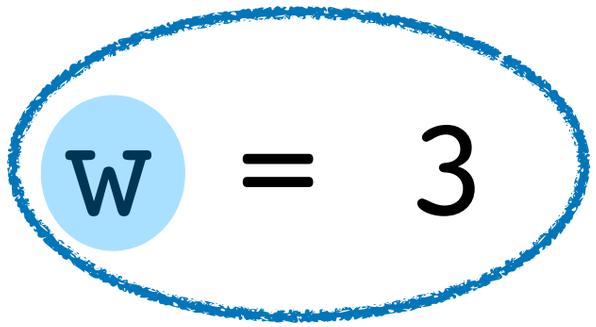
```
y = w + 1

z = 6 / x
```



```
w = 3

x = 0

y = w + 1

z = 6 / x
```

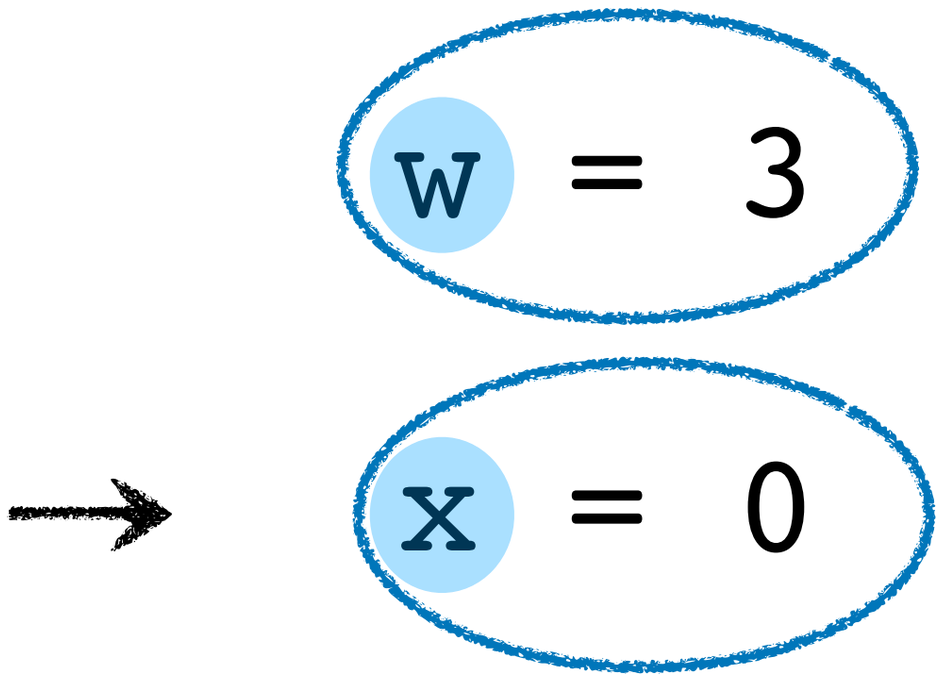



```
x = 0

y = w + 1

z = 6 / x
```

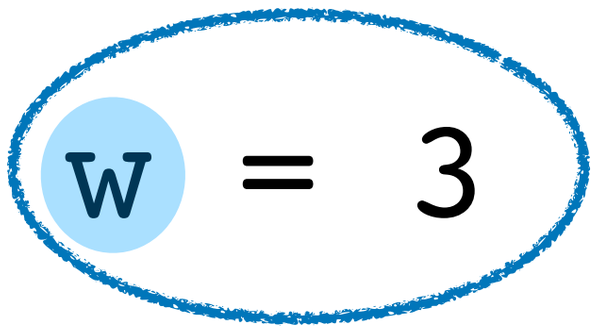


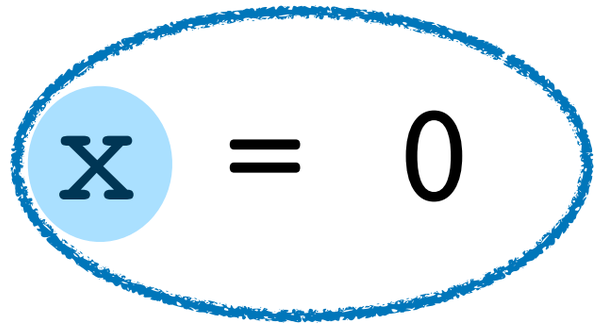
x = 0

```
y = w + 1

z = 6 / x
```




```
y = w + 1

z = 6 / x
```





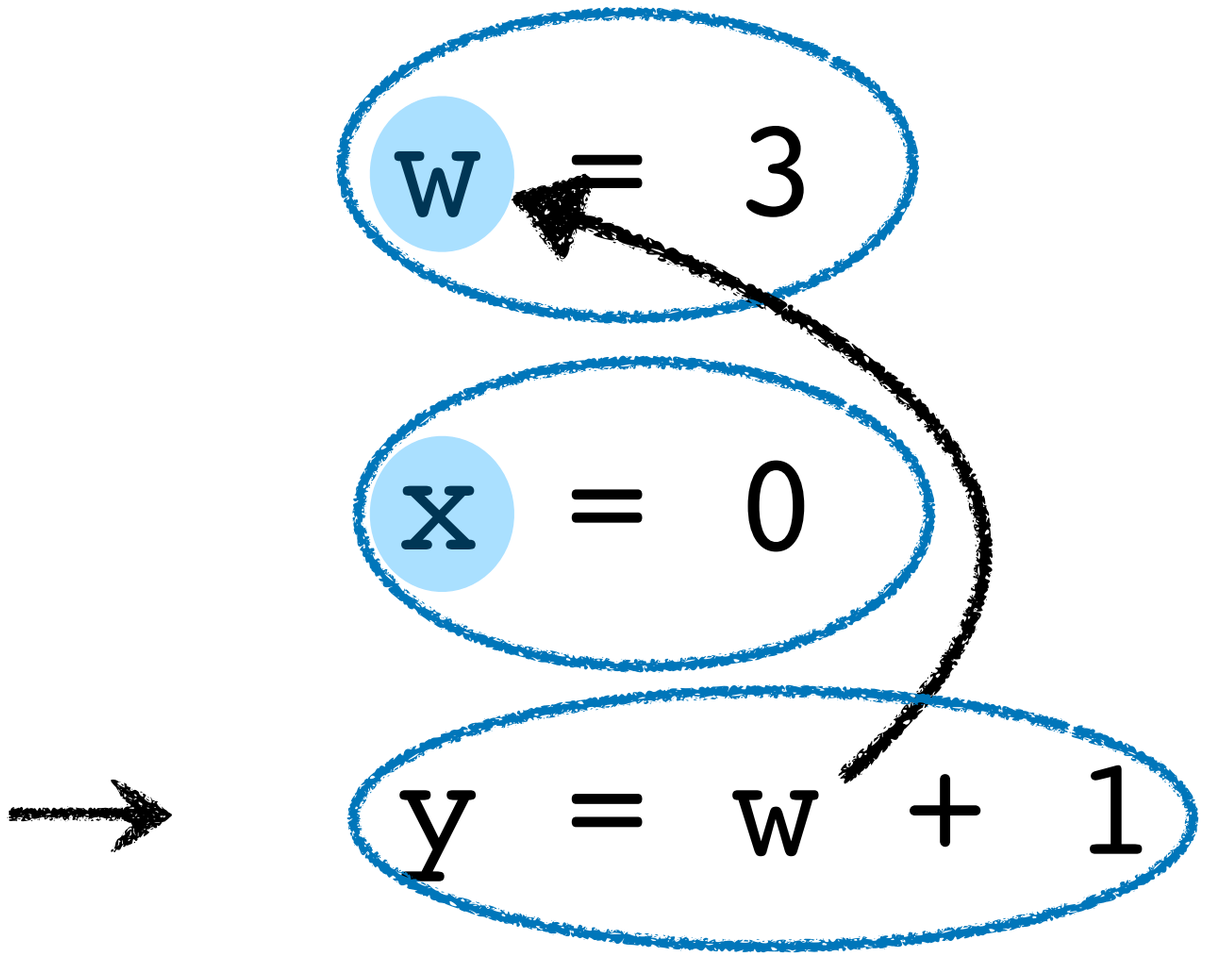
```
y = w + 1

z = 6 / x
```

```
w = 3

x = 0
```

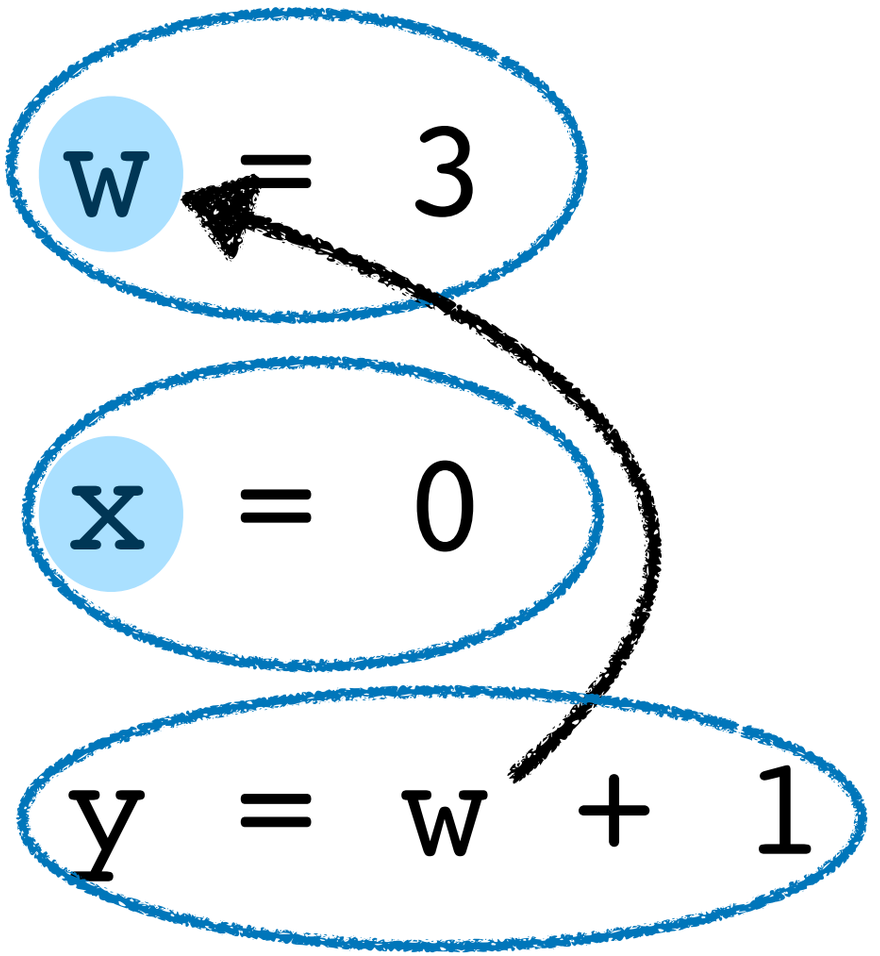
→

```
y = w + 1
```

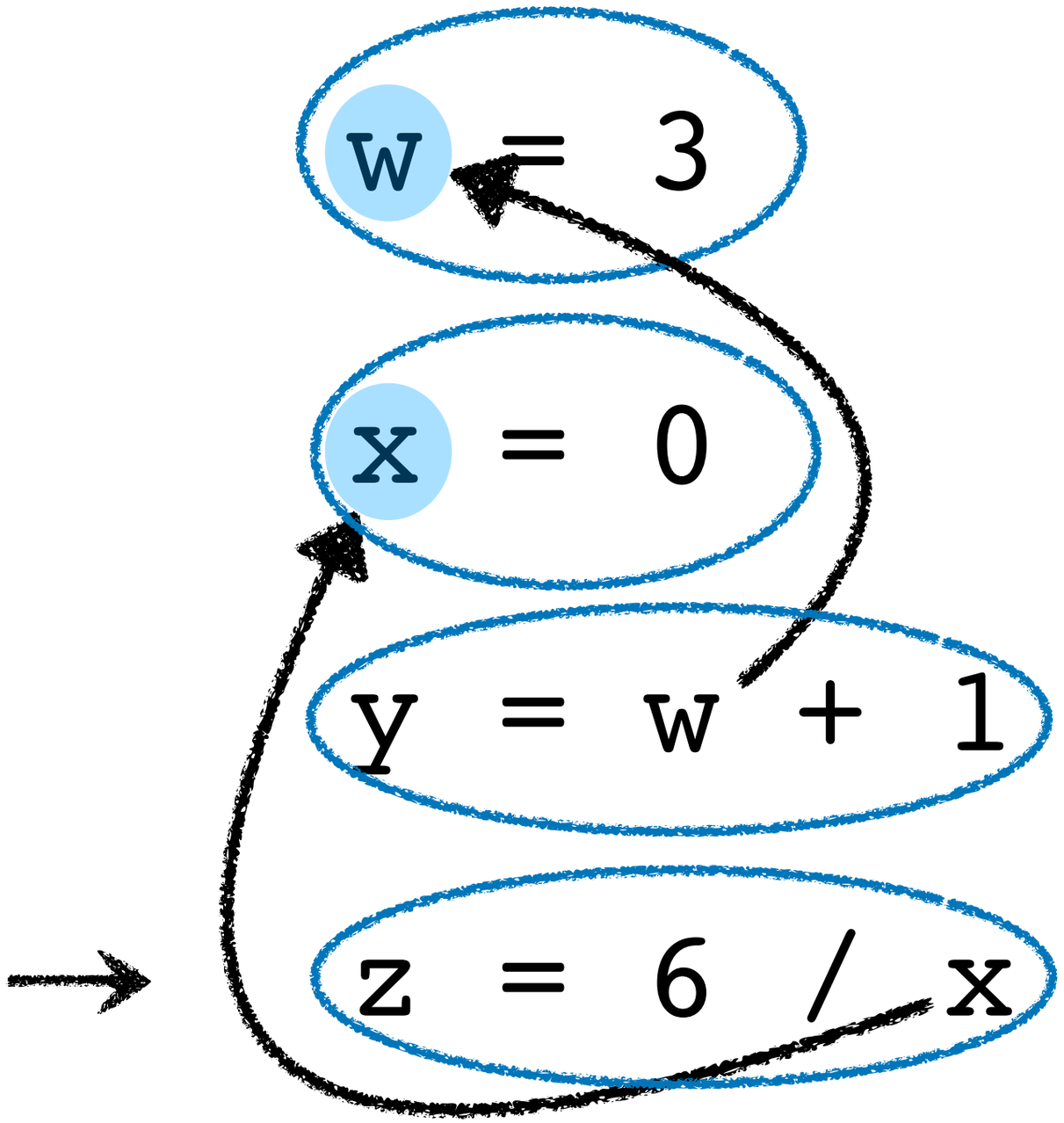
```
z = 6 / x
```

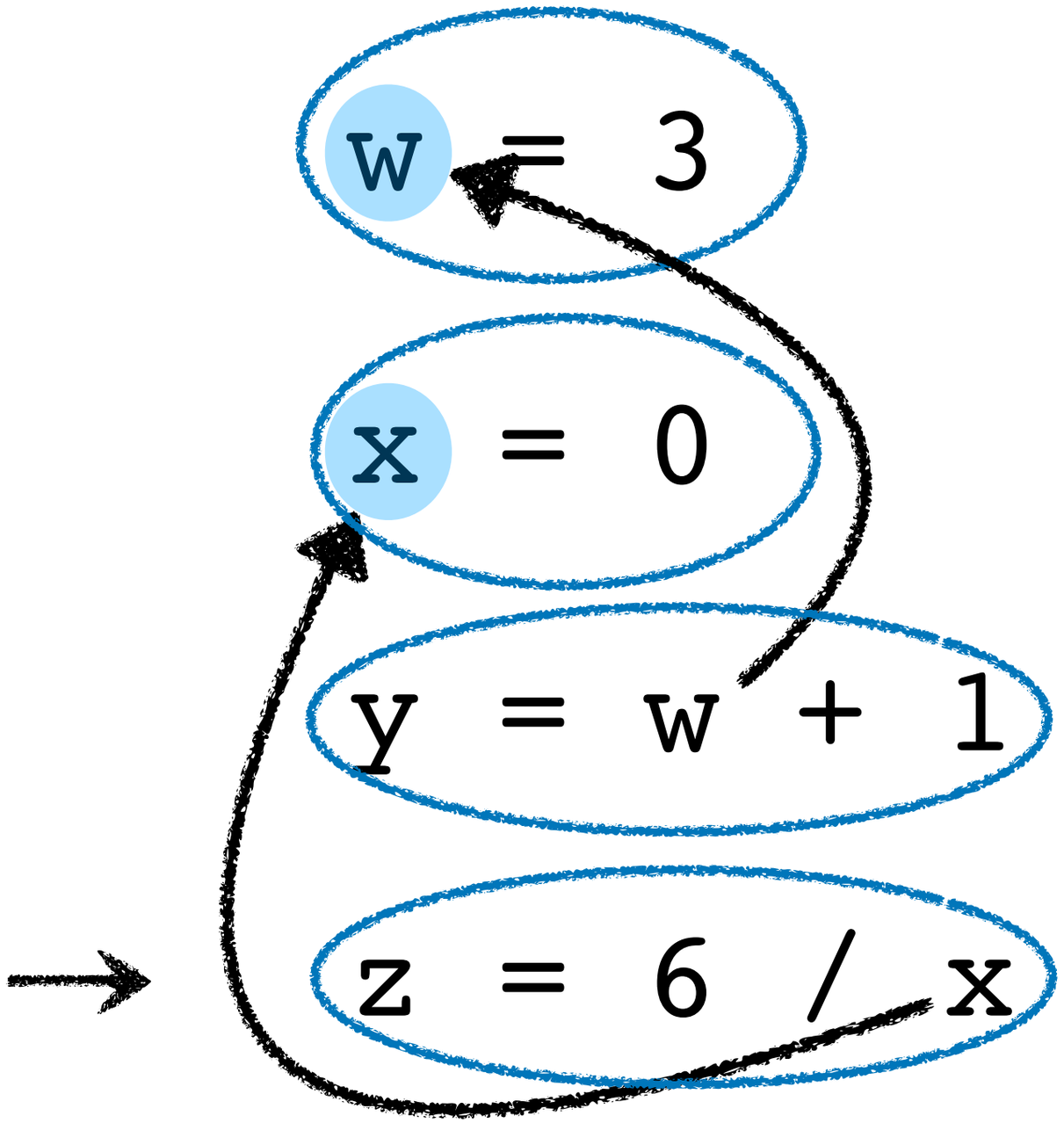



```
z = 6 / x
```

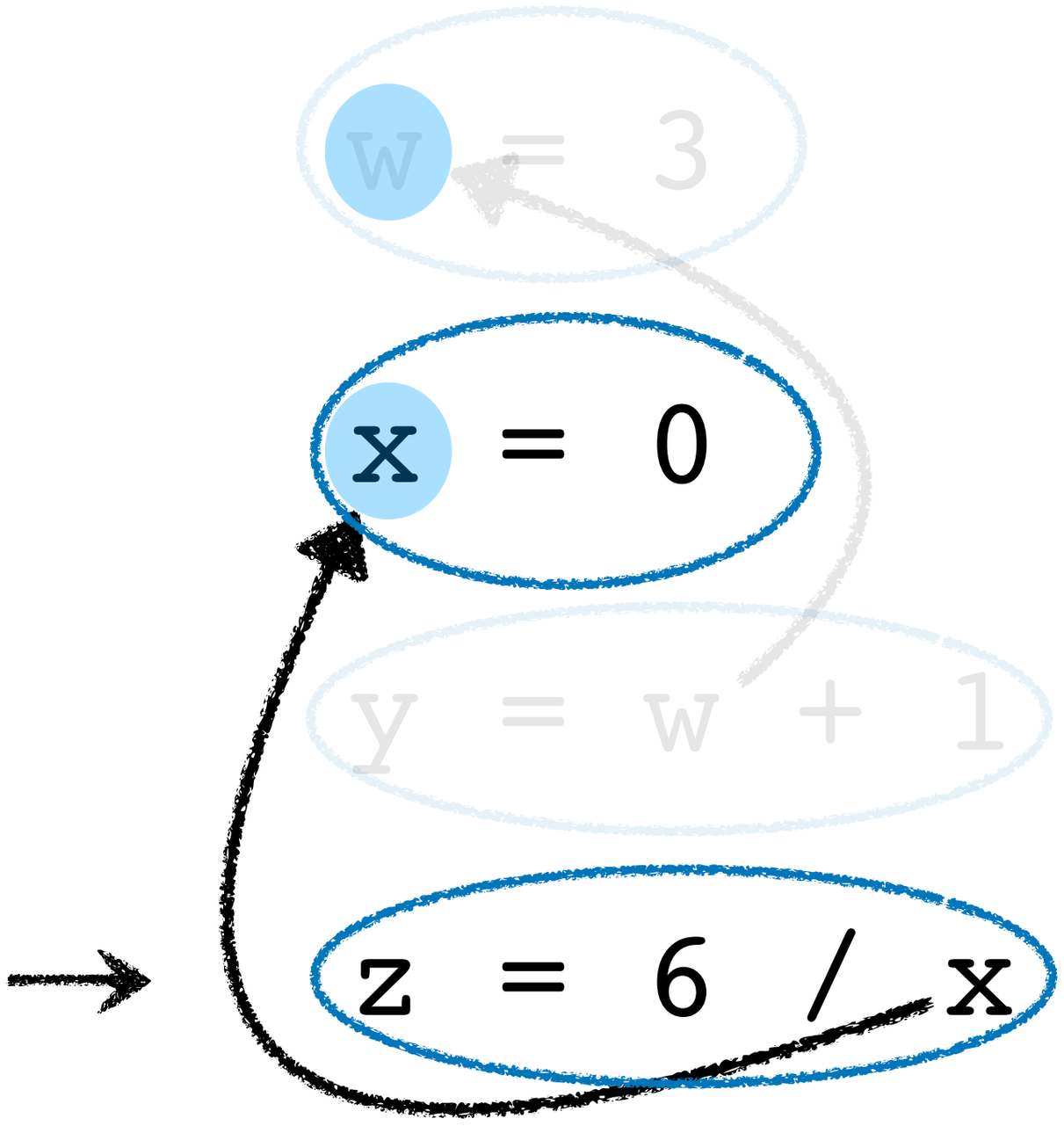


→ `z = 6 / x`


w = 3
x = 0
y = w + 1
z = 6 / x



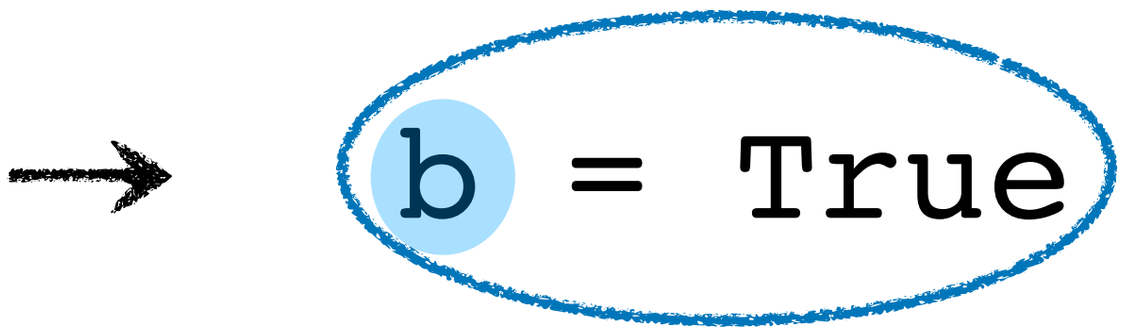
ZeroDivisionError




```
ZeroDivisionError
```

```
b = True

if b:
    x = 0
z = 6 / x
```

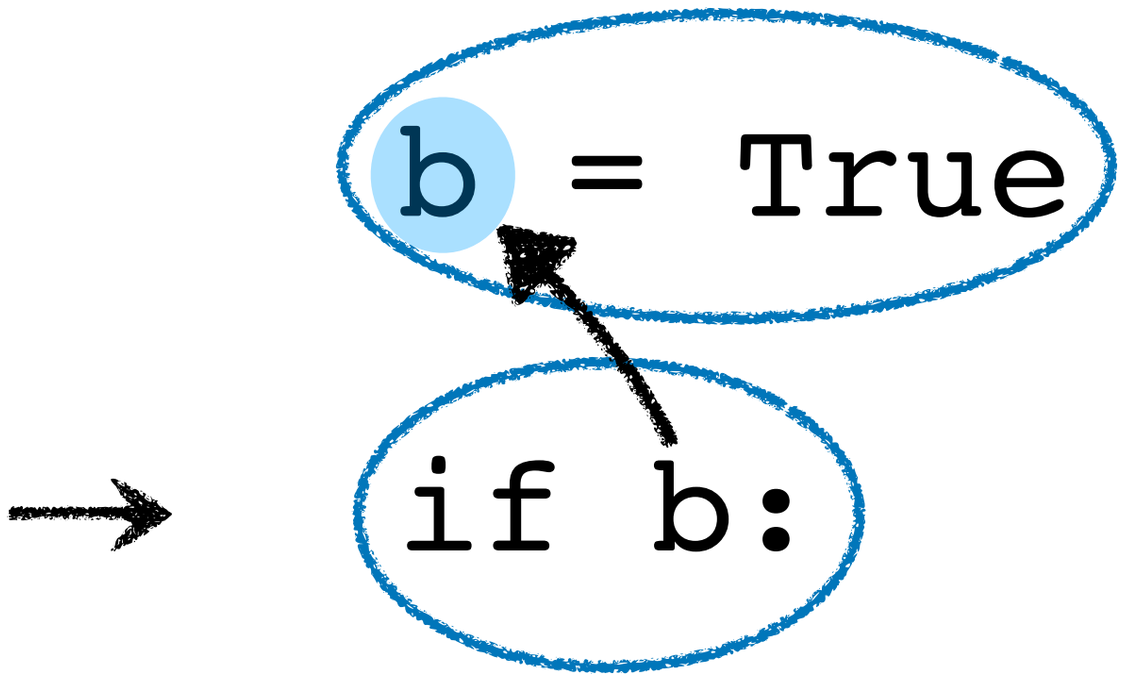
```
b = True

if b:
    x = 0
z = 6 / x
```



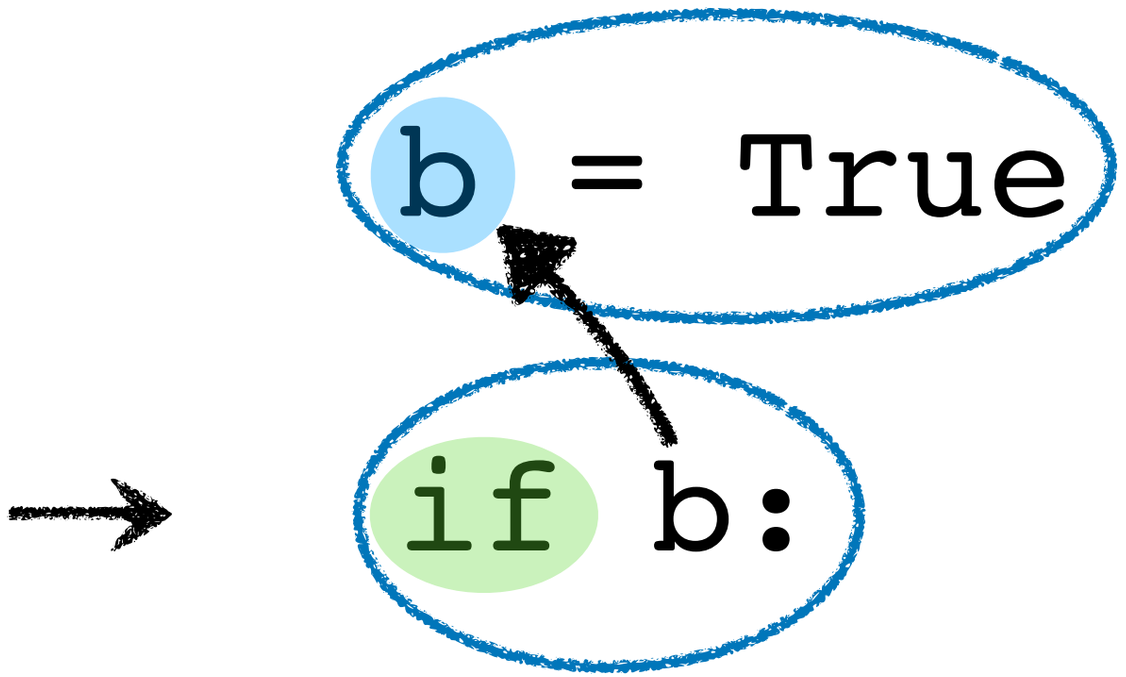

```
if b:
    x = 0
z = 6 / x
```

```
b = True
```

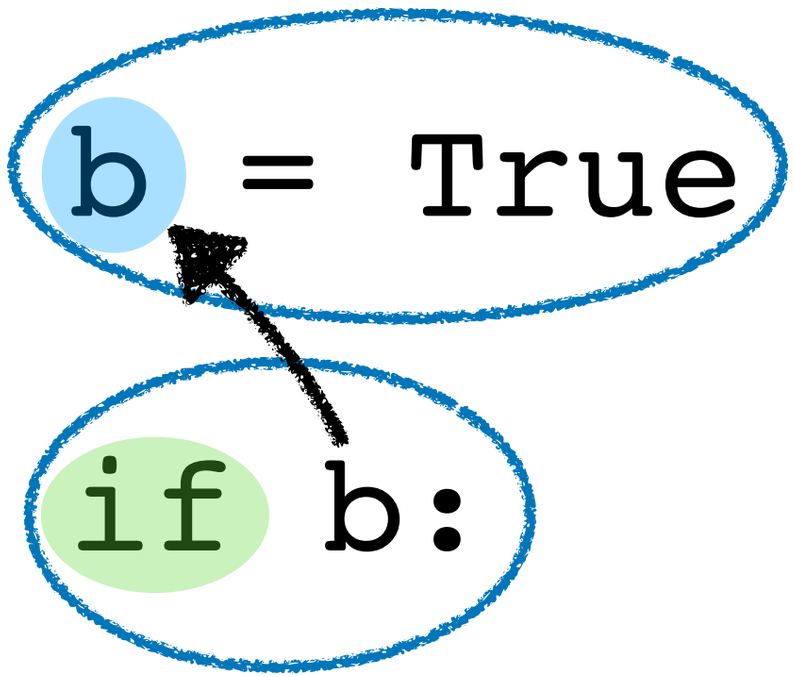
```
if b:
    x = 0
z = 6 / x
```




```
x = 0

z = 6 / x
```

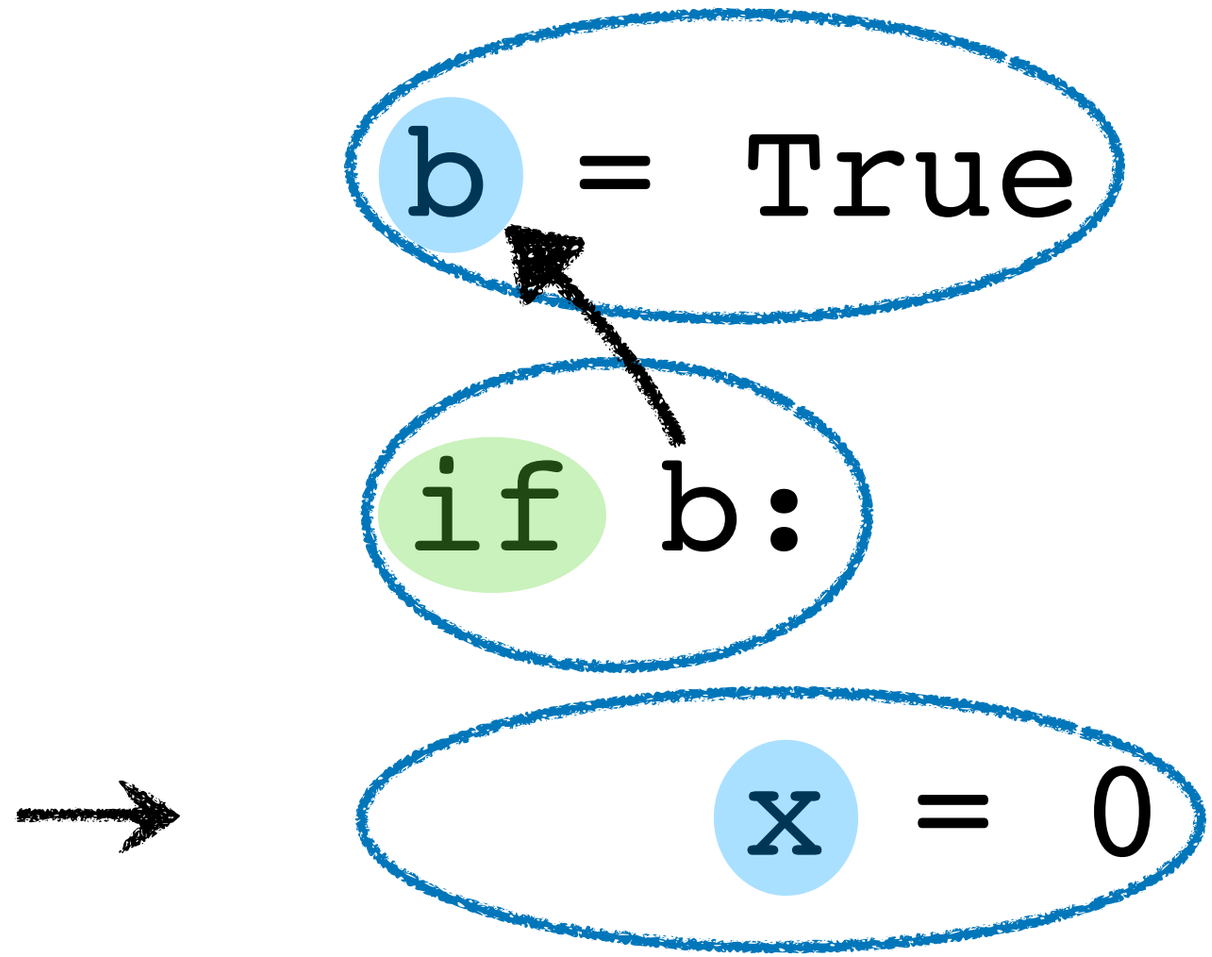



```
x = 0

z = 6 / x
```

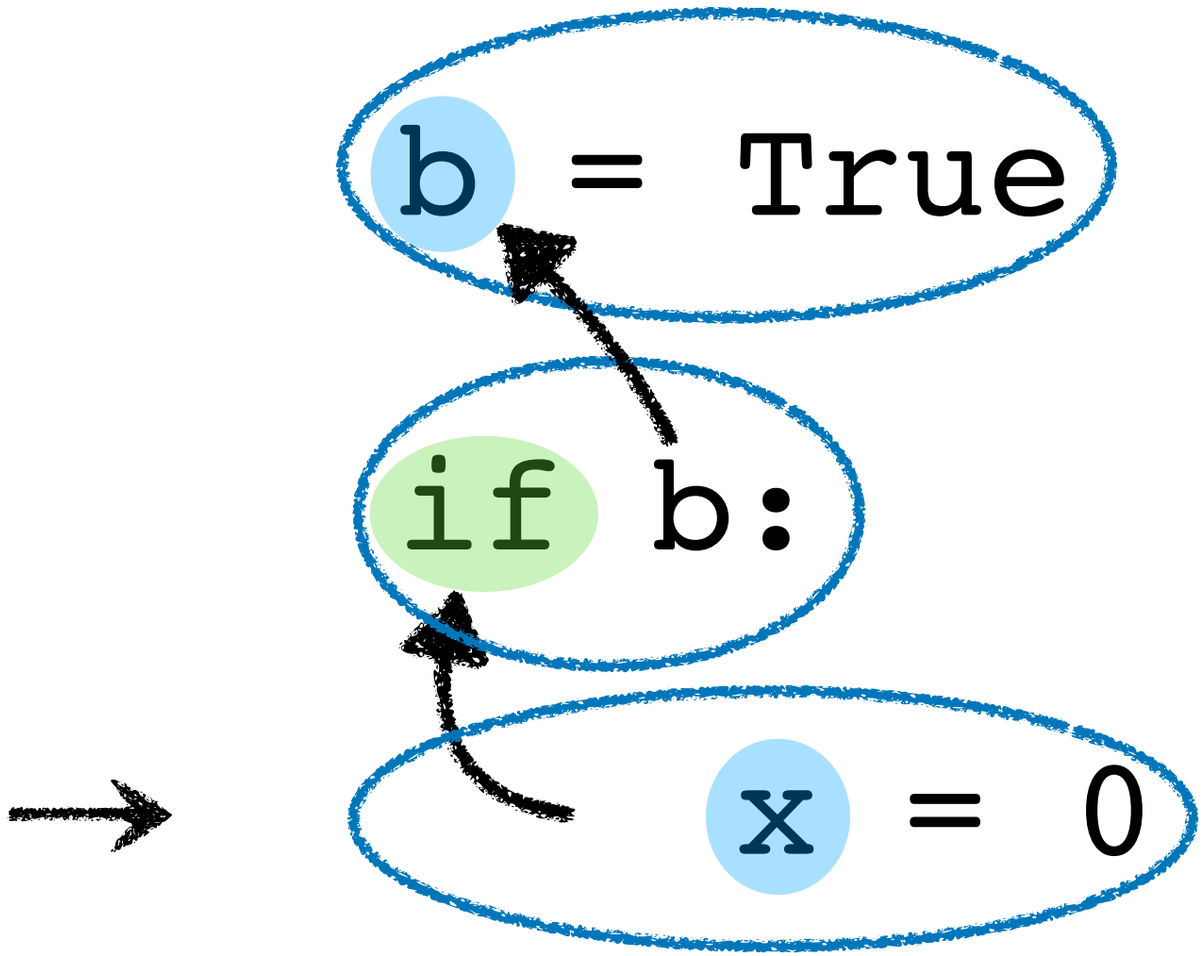



```
x = 0
```

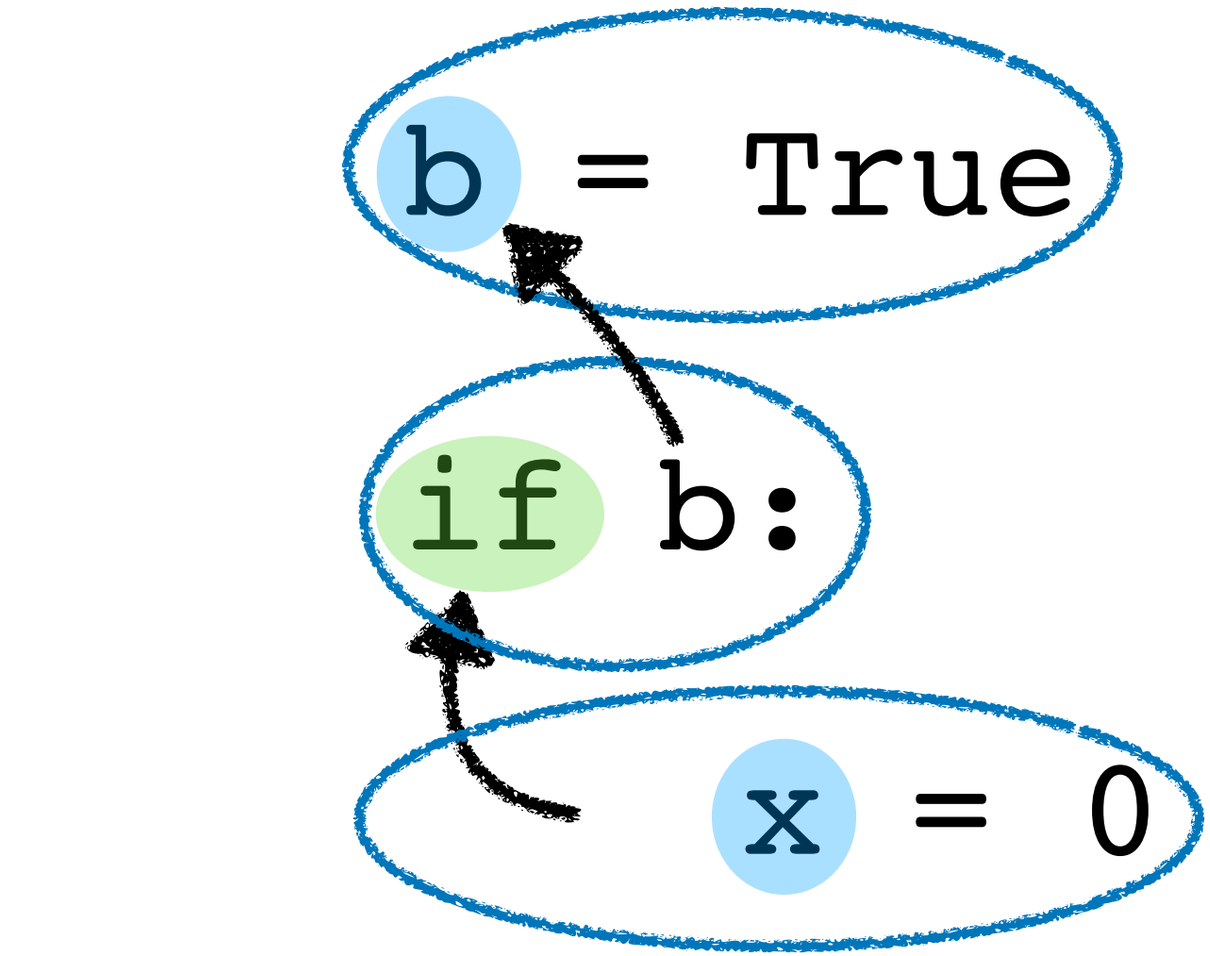
```
z = 6 / x
```



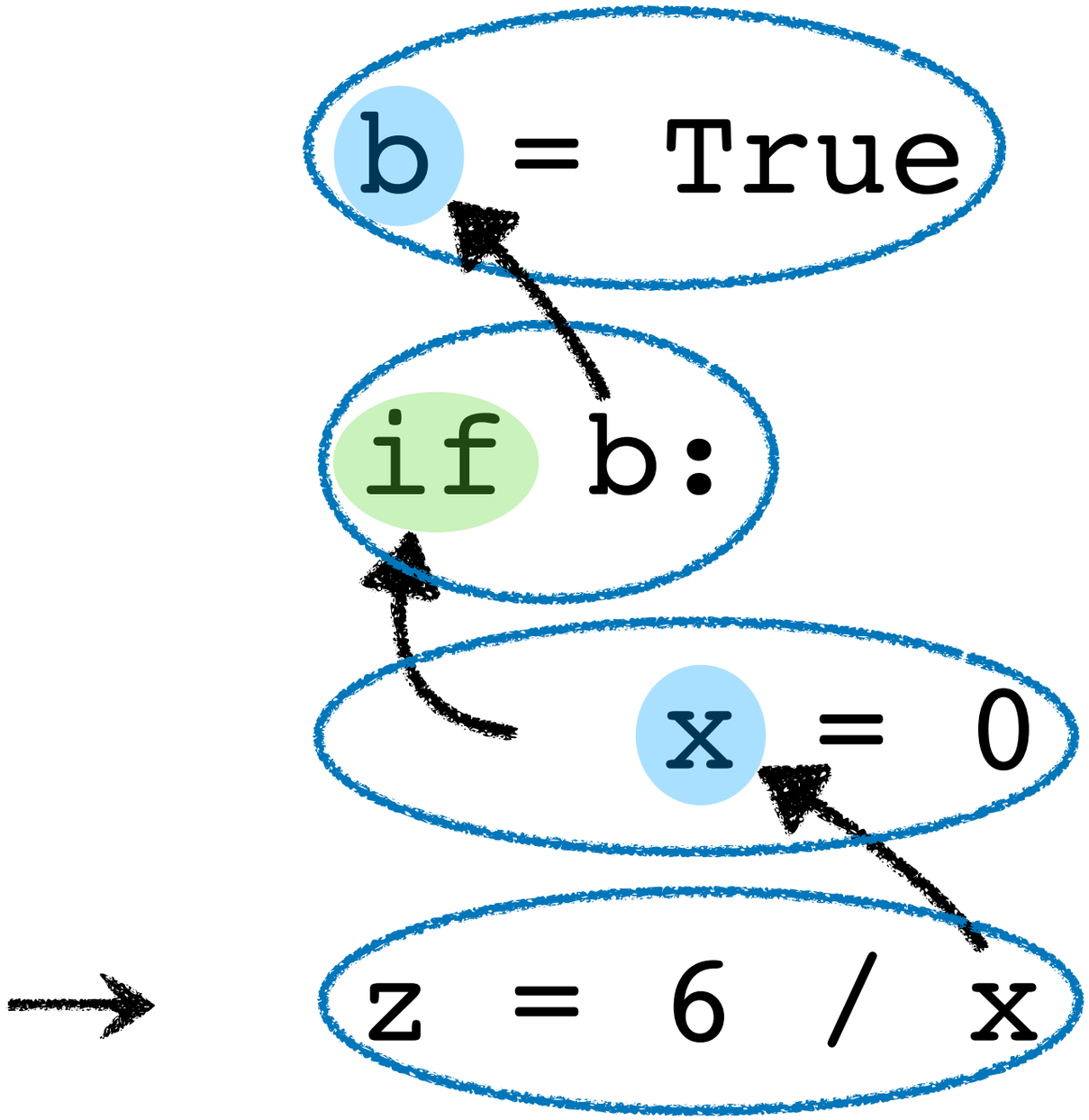

```
z = 6 / x
```

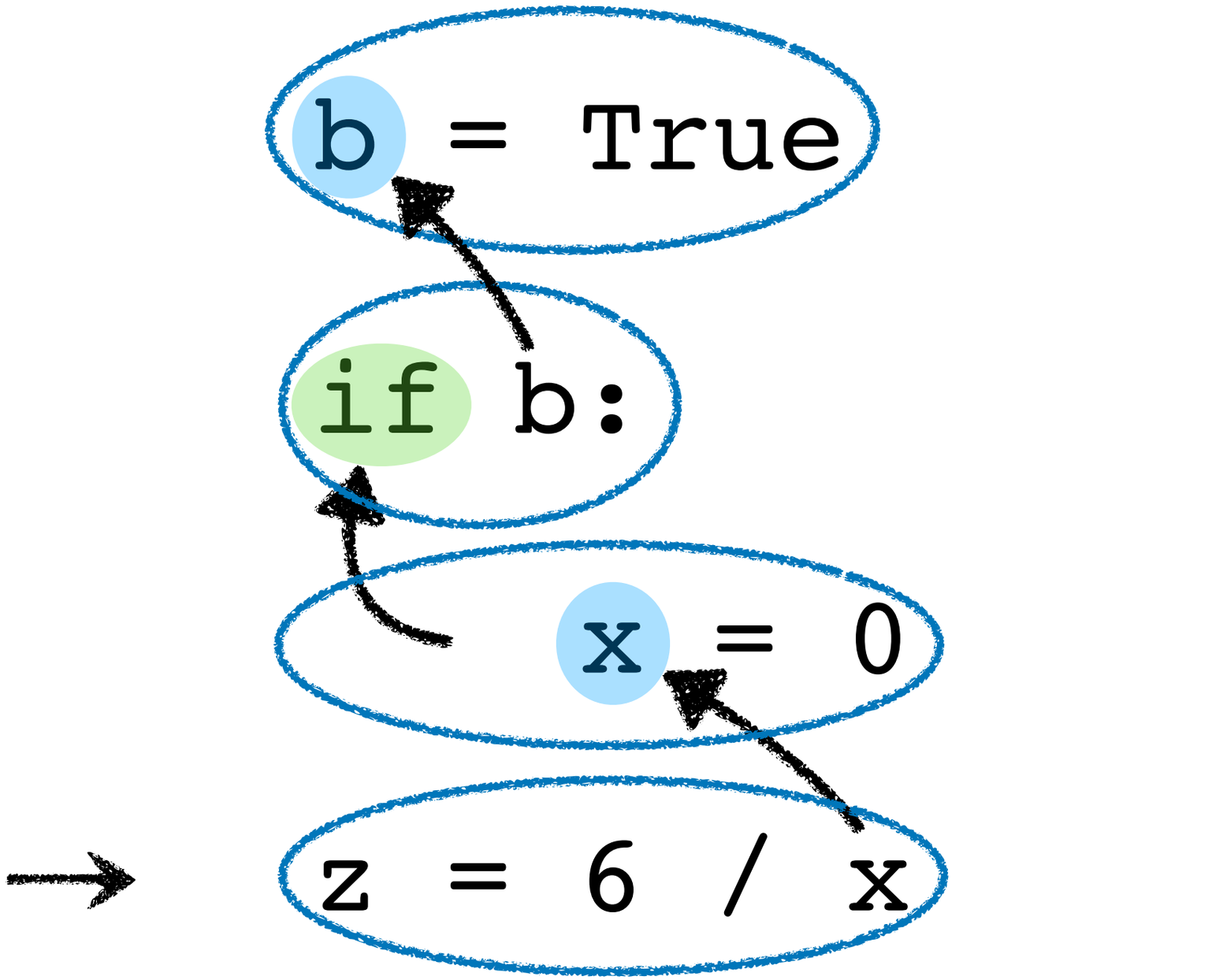



```
z = 6 / x
```




```
z = 6 / x
```

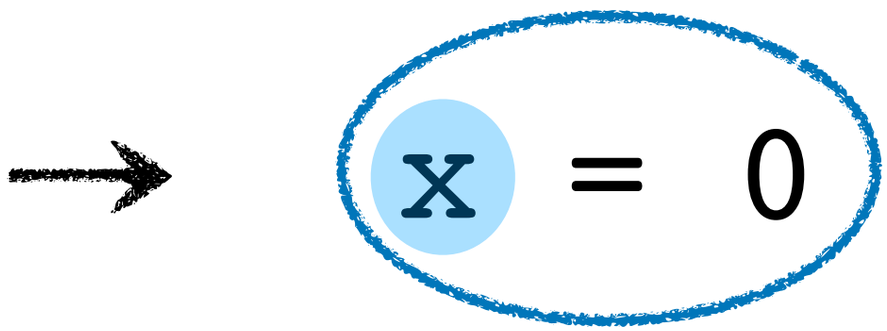

b = True
if b:
x = 0
z = 6 / x



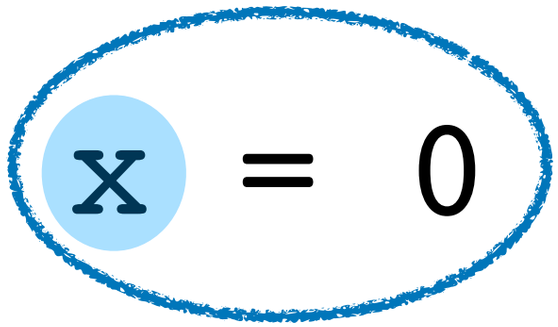
ZeroDivisionError

```
x = 0

if x != 0:
    x = 1
z = 6 / x
```

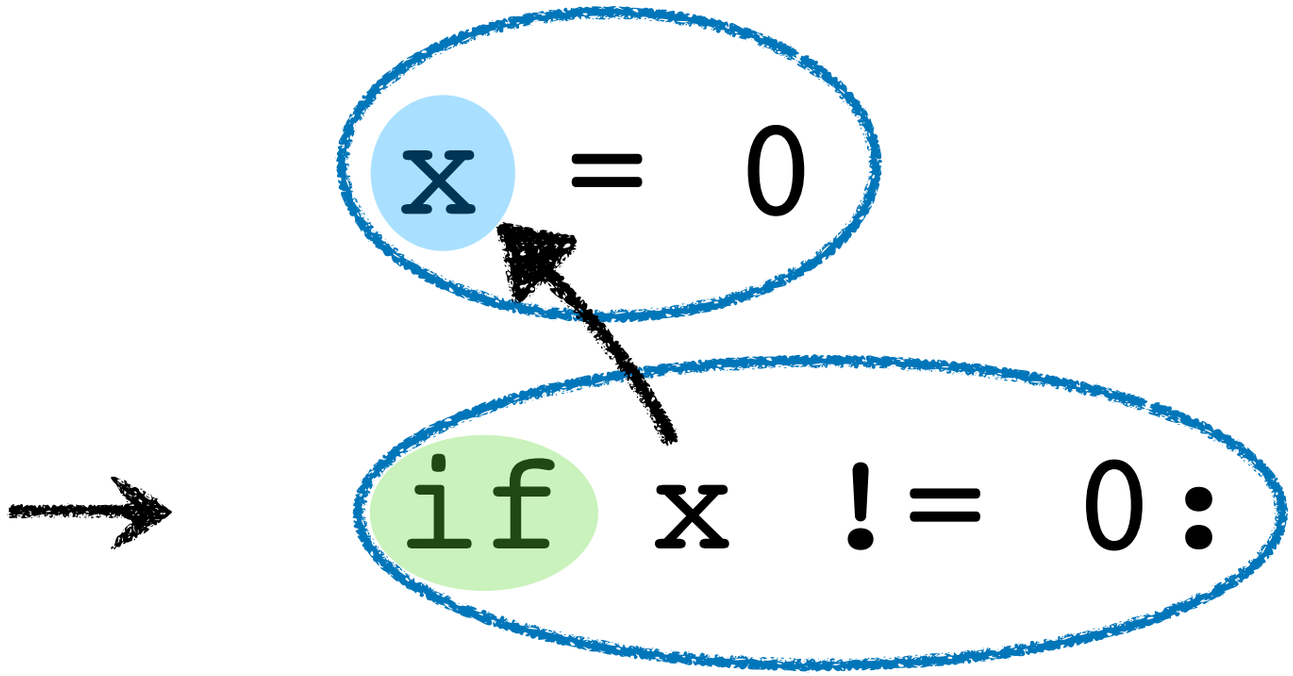
```
x = 0

if x != 0:
    x = 1

z = 6 / x
```




```
if x != 0:
    x = 1
z = 6 / x
```

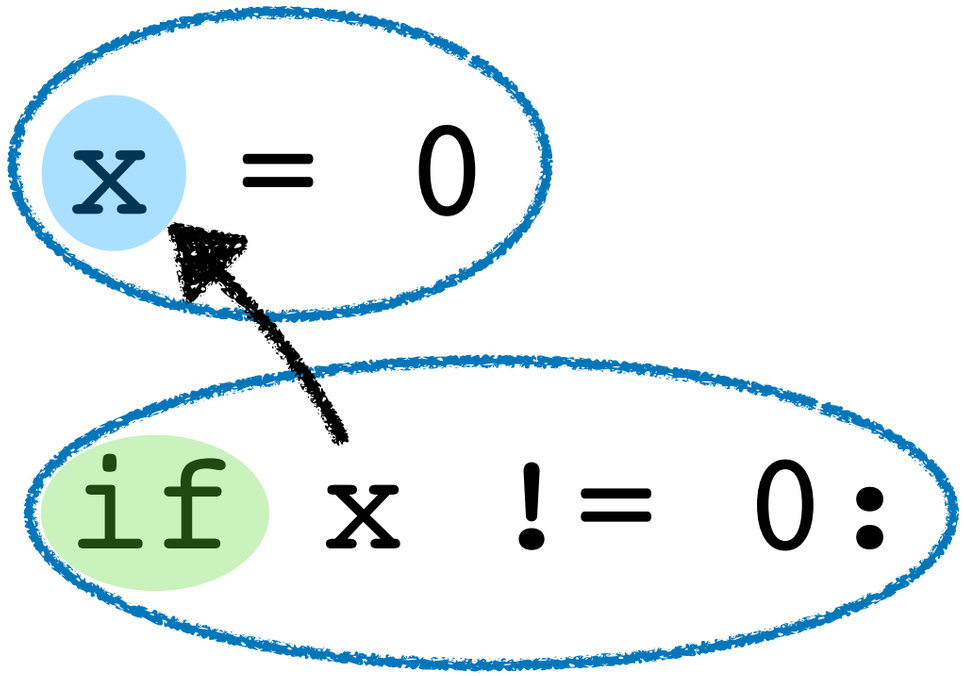


```
if x != 0:
    x = 1
z = 6 / x
```

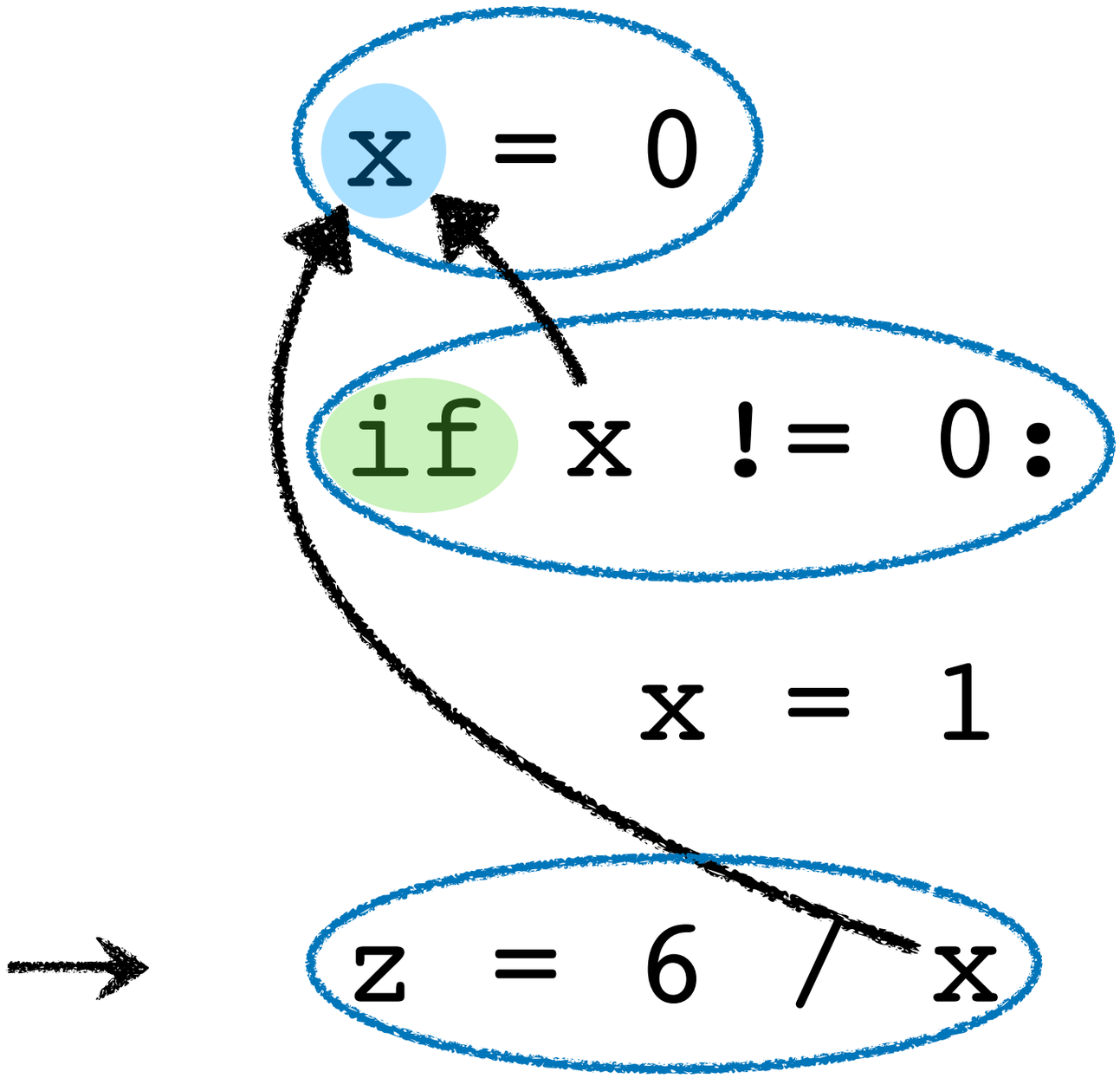


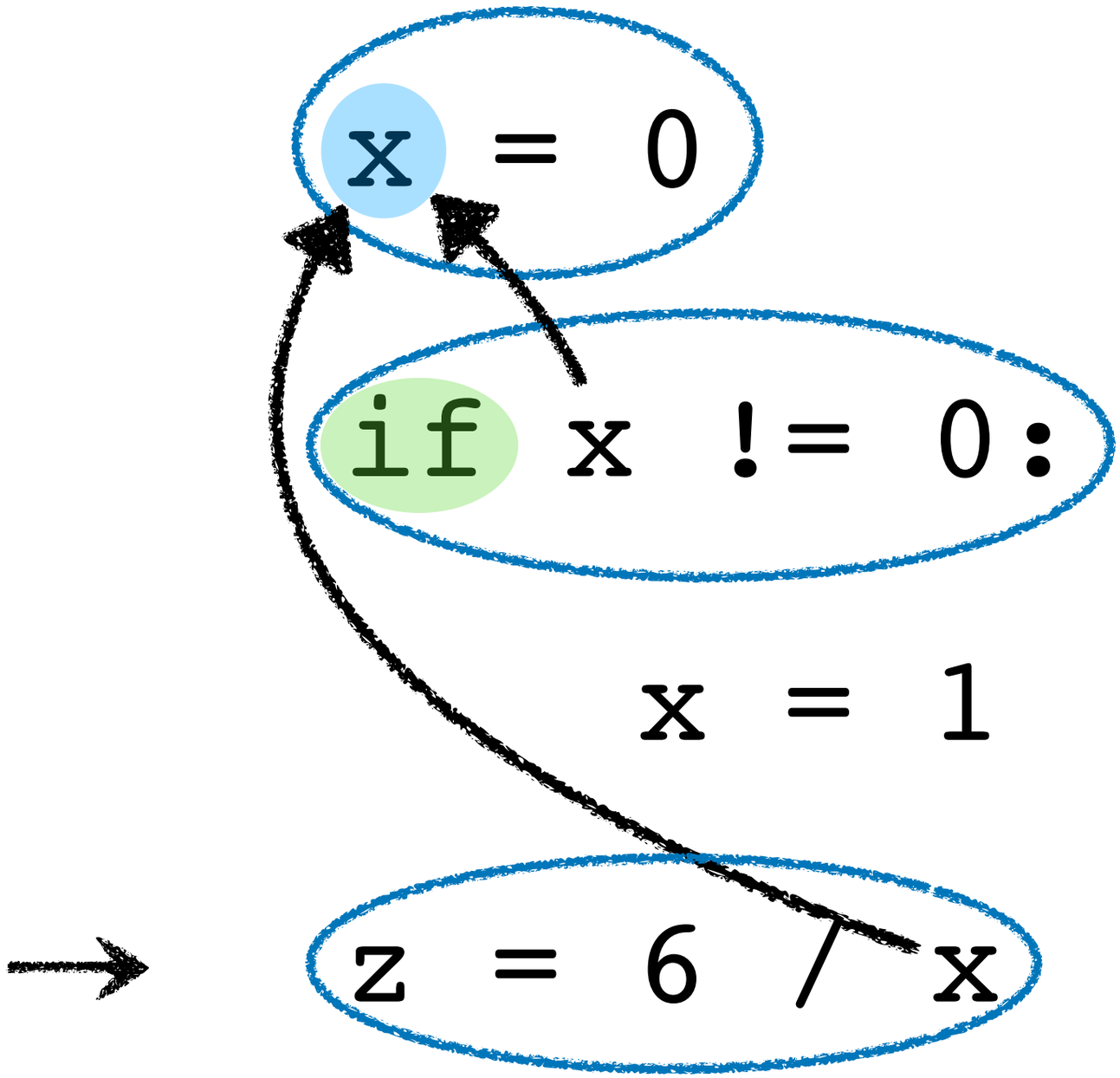

```
x = 1
```

```
z = 6 / x
```

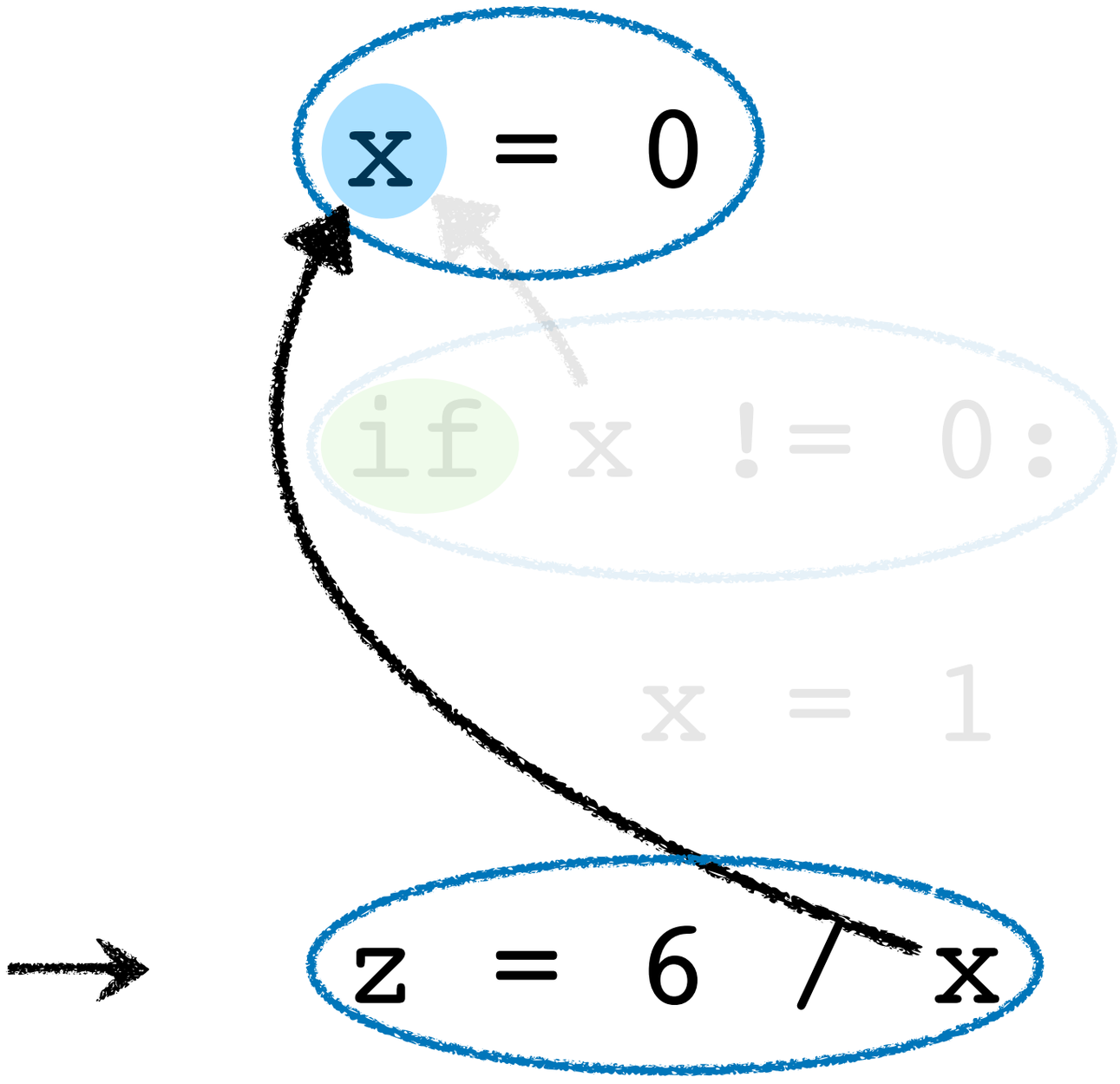



```
        x = 1

→       z = 6 / x
```


x = 0
if x != 0:
x = 1
z = 6 / x



ZeroDivisionError



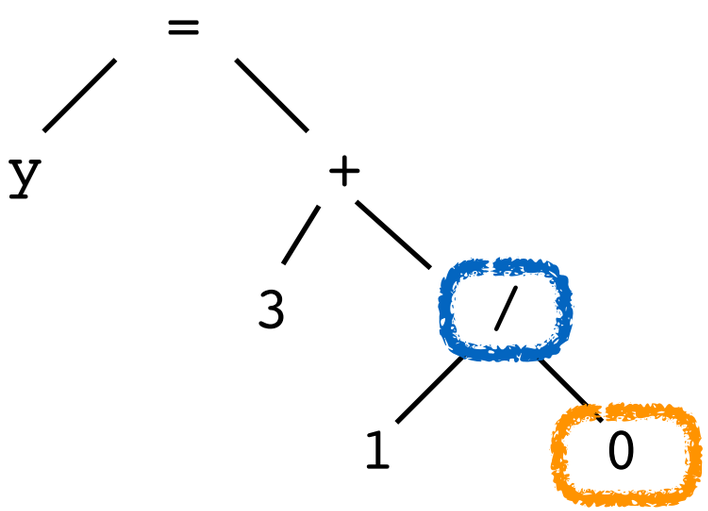
ZeroDivisionError

Dynamic features

What is the *type* of the expression?

Is it in the error *slice*?

```
y = 3 + 1 / 0
```



Terms

Features →

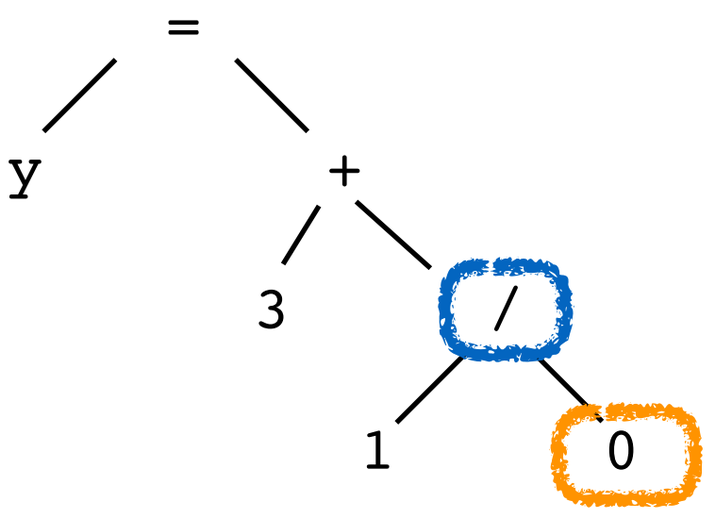
NodeKind	Size	Type	Slice	Crash?	Msg
BinaryOp	3	unknown	Yes	...	...
IntLiteral	1	int	Yes	...	...

Dynamic features

Where does the program actually *crash*?

What error *message* do we get?

```
y = 3 + 1 / 0
```

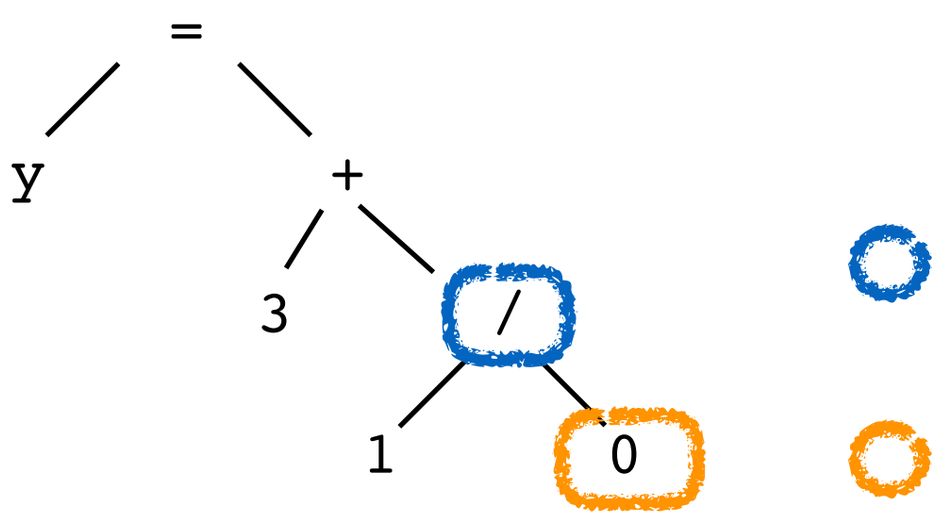


Terms

Features

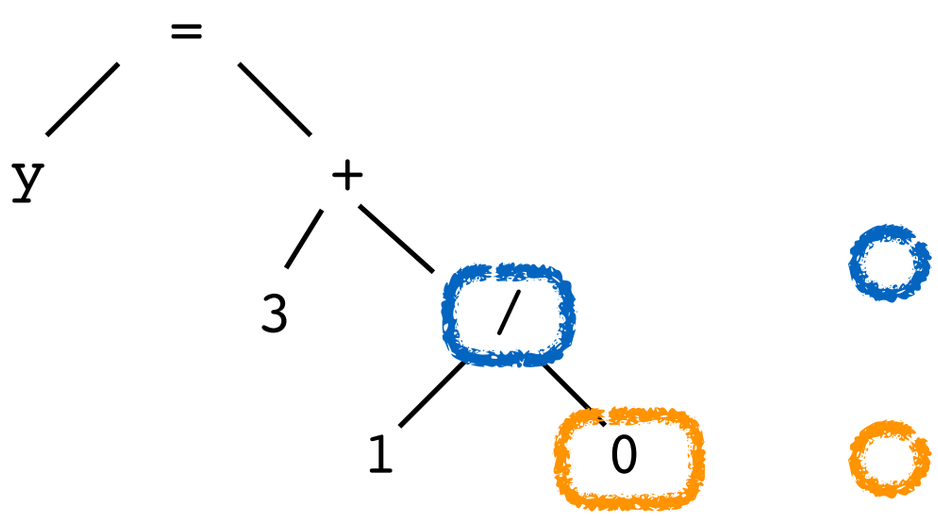
NodeKind	Size	Type	Slice	Crash?	Msg
BinaryOp	3	unknown	Yes	Yes	Div0
IntLiteral	1	int	Yes	No	Div0

Contextual features



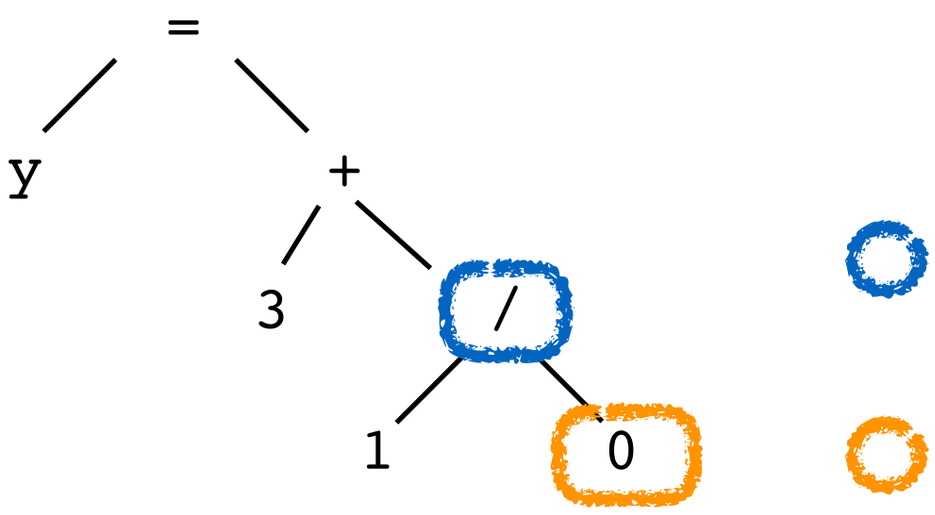
NodeKind	Size	Type	Slice	Crash?	Msg
BinaryOp	3	unknown	Yes	Yes	Div0
IntLiteral	1	int	Yes	No	Div0

Contextual features



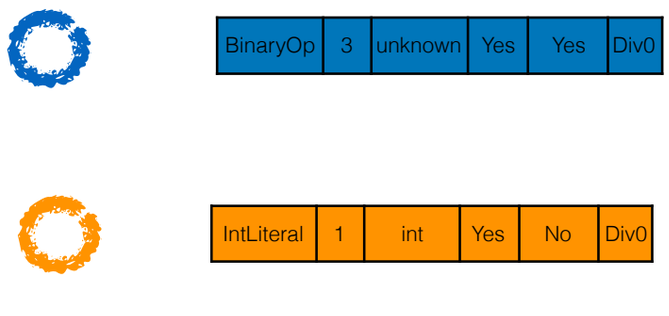
NodeKind	Size	Type	Slice	Crash?	Msg
BinaryOp	3	unknown	Yes	Yes	Div0
IntLiteral	1	int	Yes	No	Div0

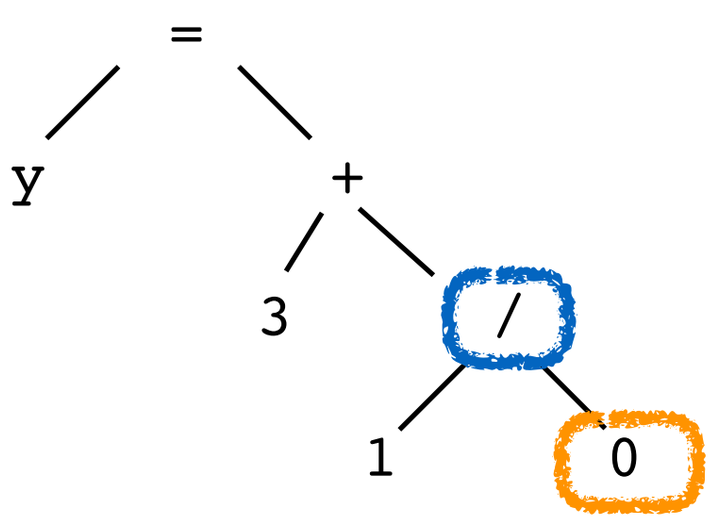
Contextual features



BinaryOp	3	unknown	Yes	Yes	Div0
IntLiteral	1	int	Yes	No	Div0

Contextual features



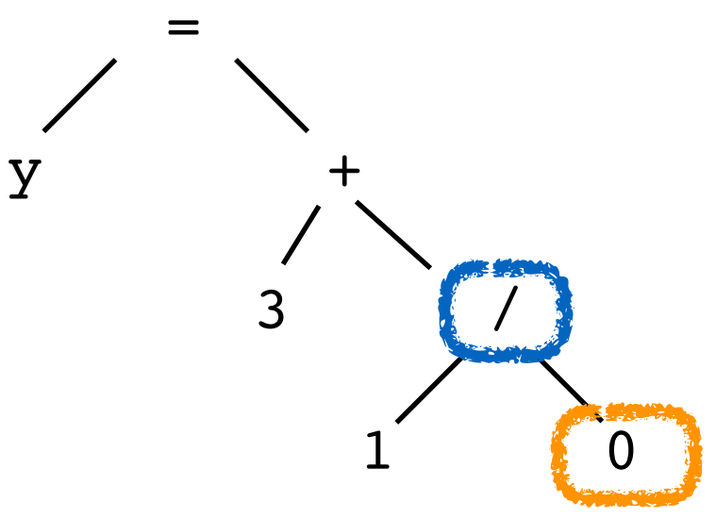


Contextual features

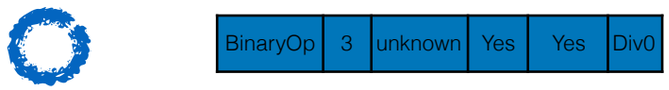
BinaryOp	3	unknown	Yes	Yes	Div0

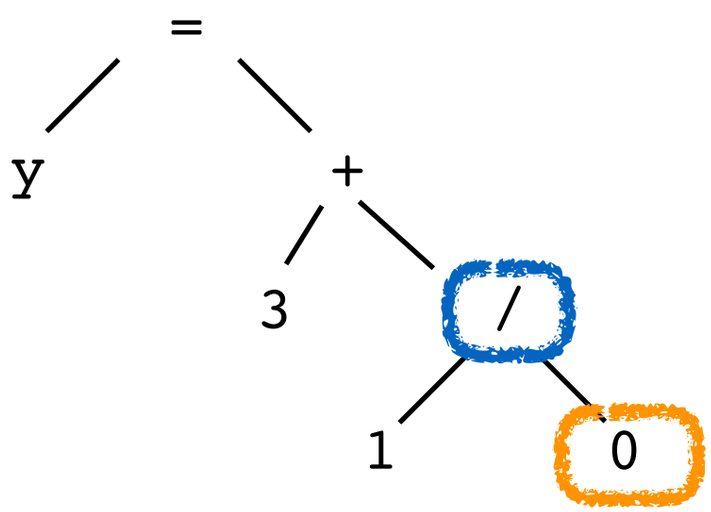
IntLiteral	1	int	Yes	No	Div0

Problem: we lost all **context**



Contextual features

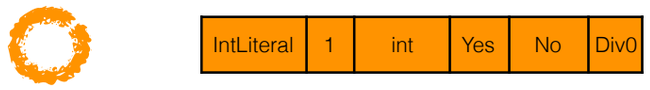




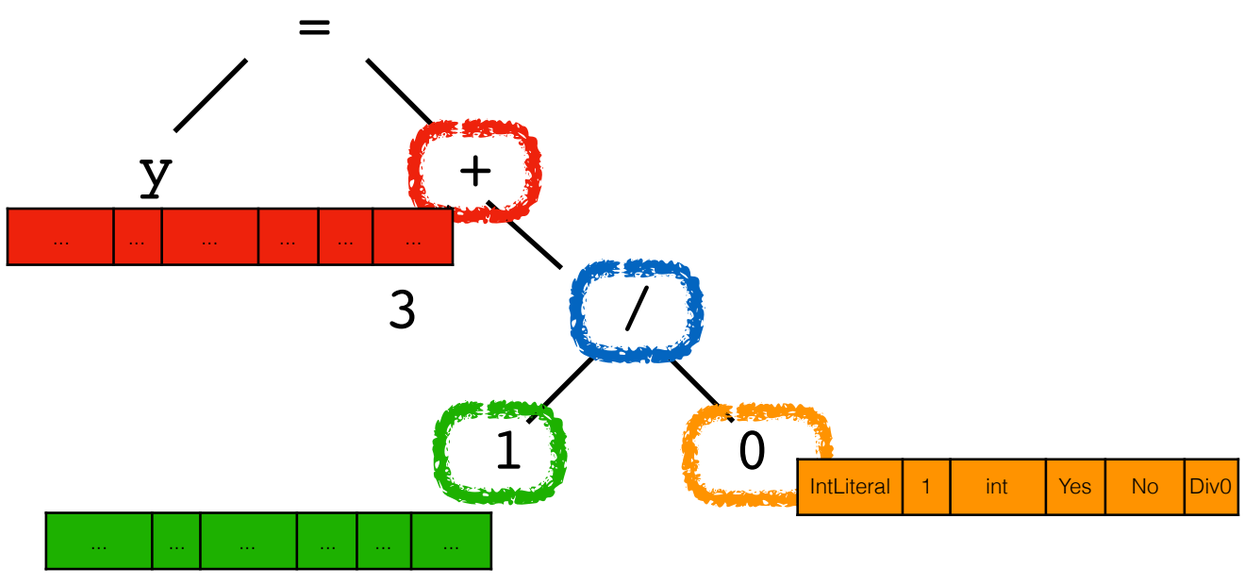
Problem: we lost all **context**

Regain context by concatenating parents and children

Contextual features

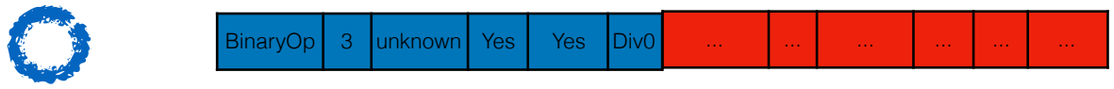


Problem: we lost all **context**

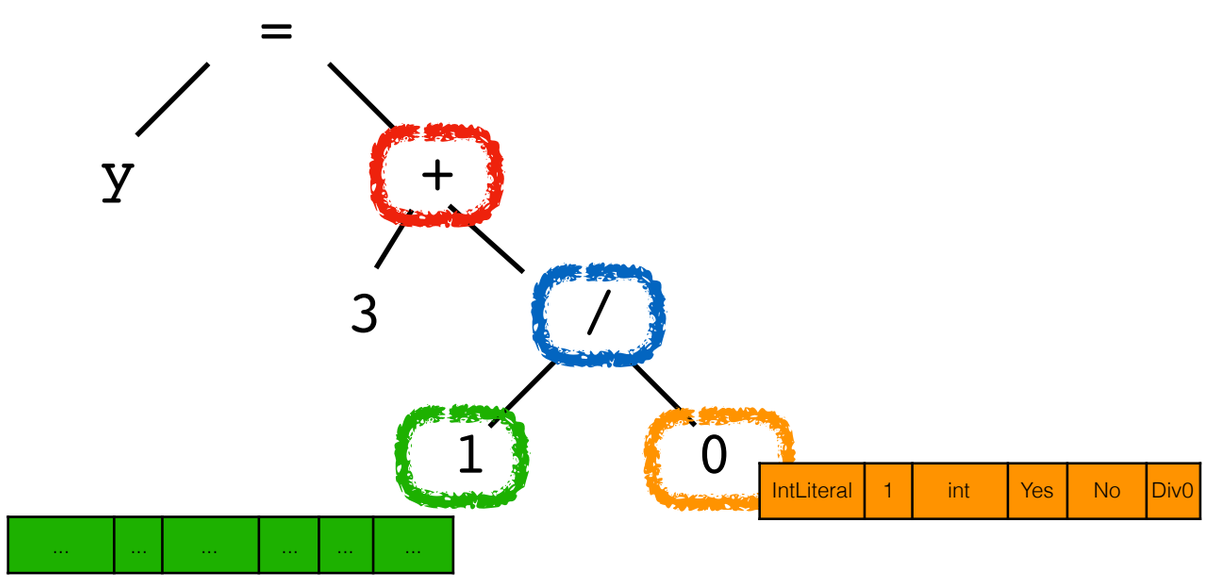


Regain context by concatenating parents and children

Contextual features

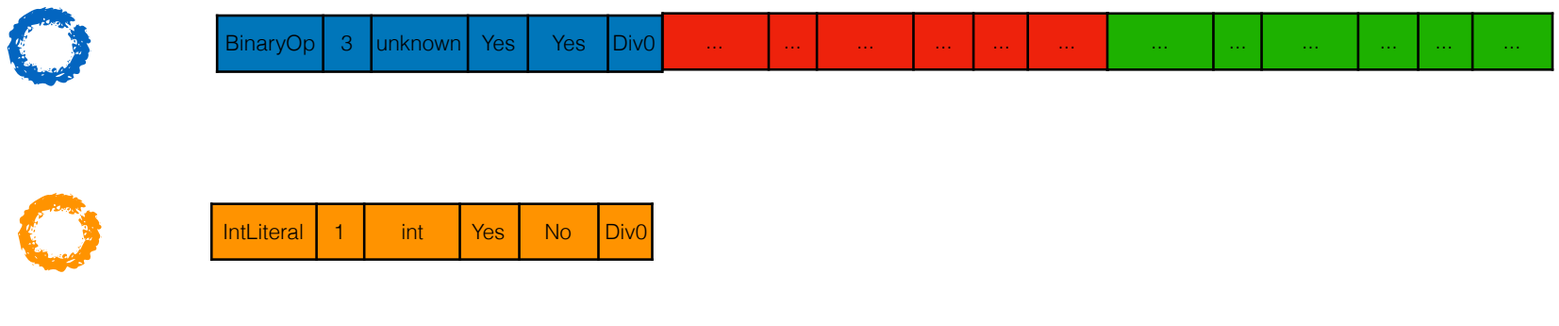


Problem: we lost all **context**

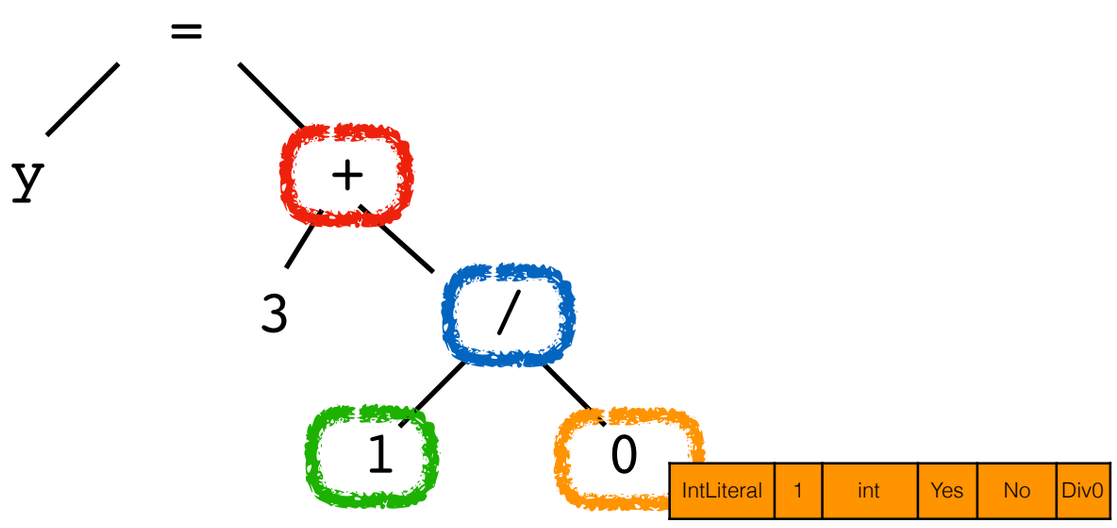


Regain context by concatenating parents and children

Contextual features

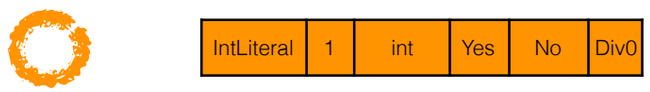


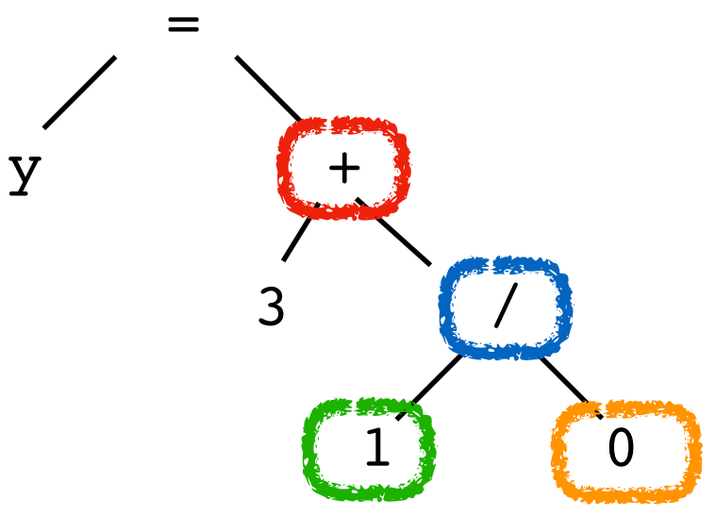
Problem: we lost all **context**



Regain context by concatenating parents and children

Contextual features

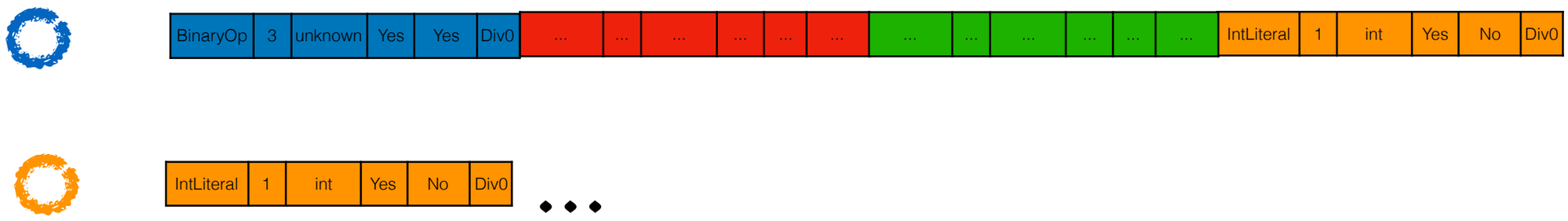


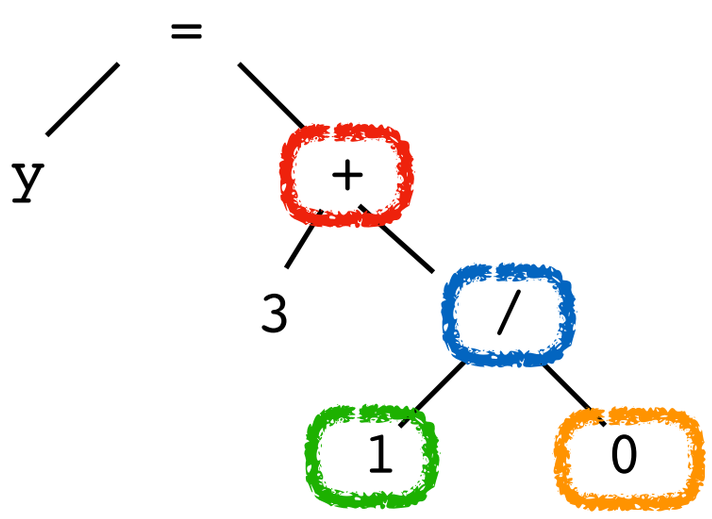


Problem: we lost all **context**

Regain context by concatenating parents and children

Contextual features





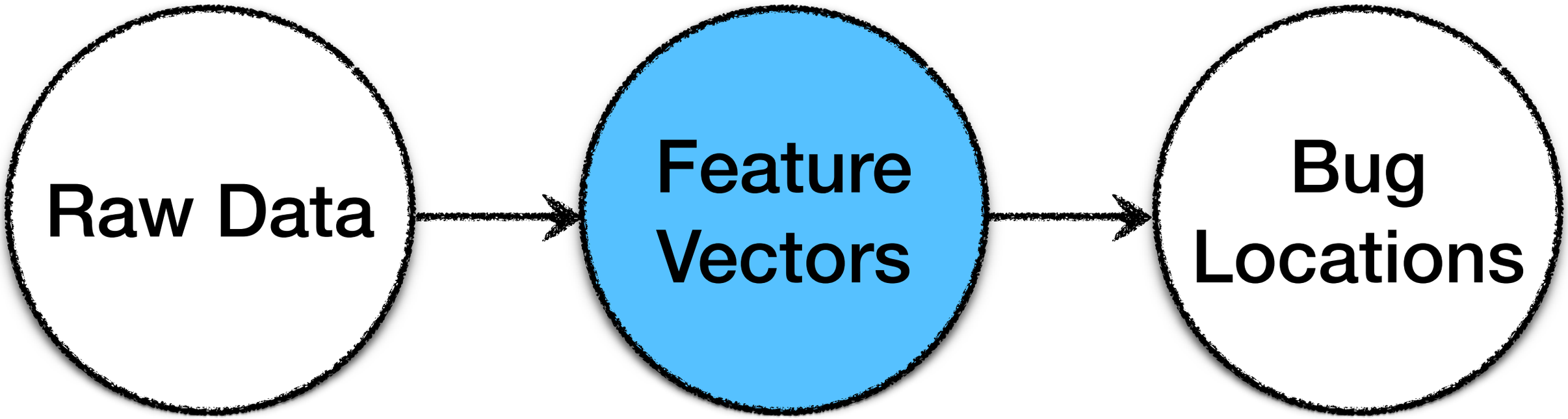
Problem: we lost all **context**

Regain context by concatenating parents and children

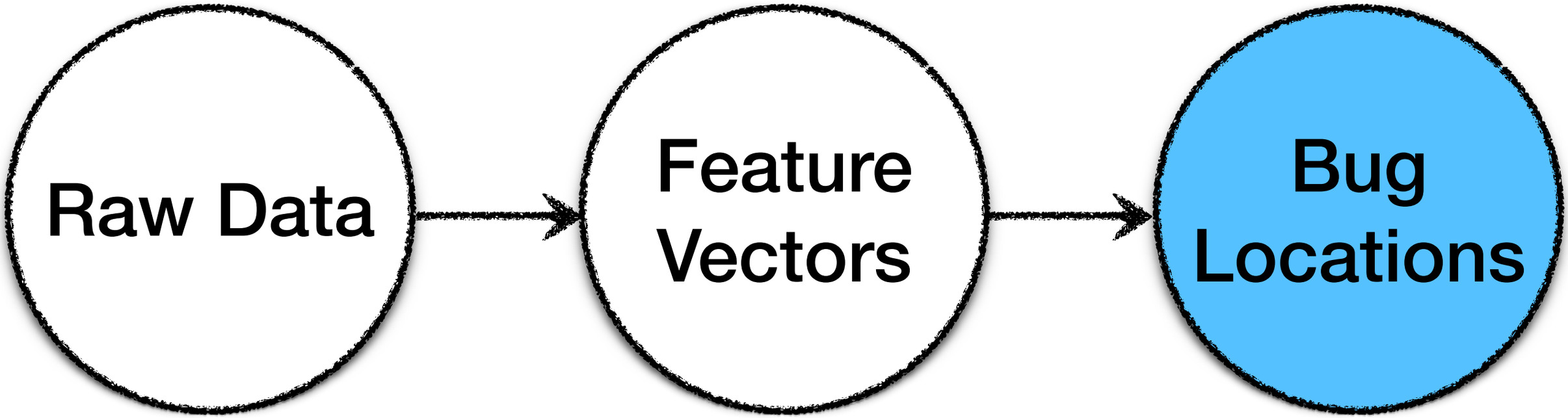
One-hot Encoding

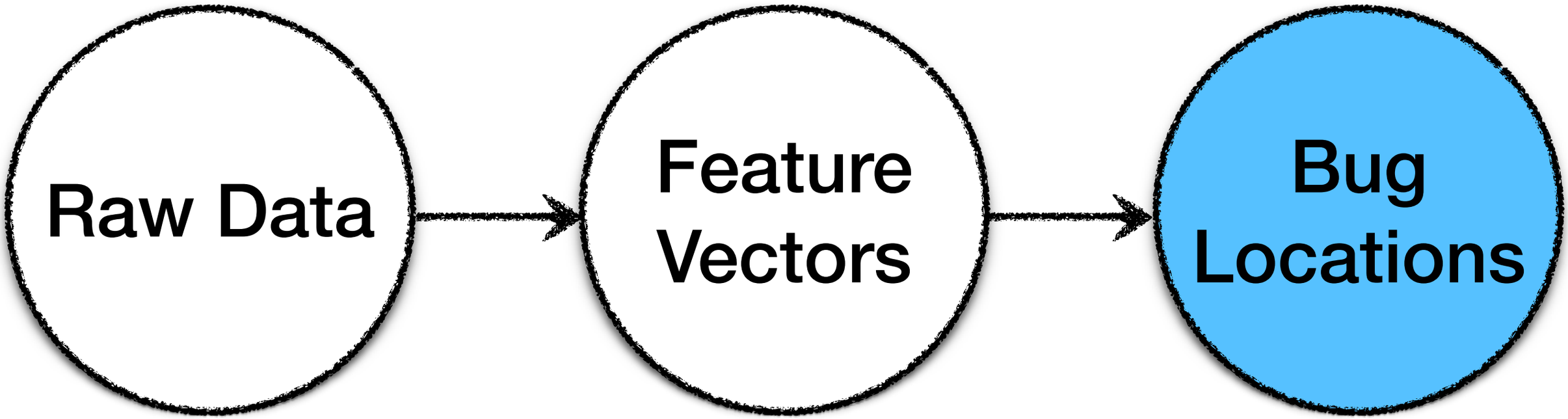
...	Type	...
...	int	...

...	Type-is-int	Type-is-str	Type-is-func	...
...	1	0	0	...

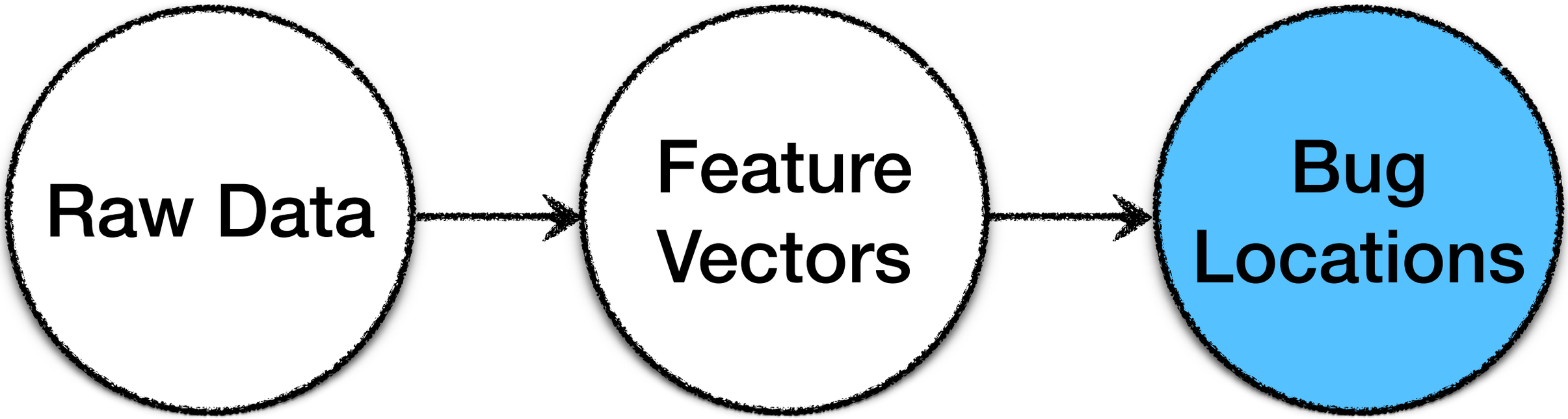


- Static/Syntactic
- Dynamic
- Contextual


Raw Data
Feature Vectors
Bug Locations

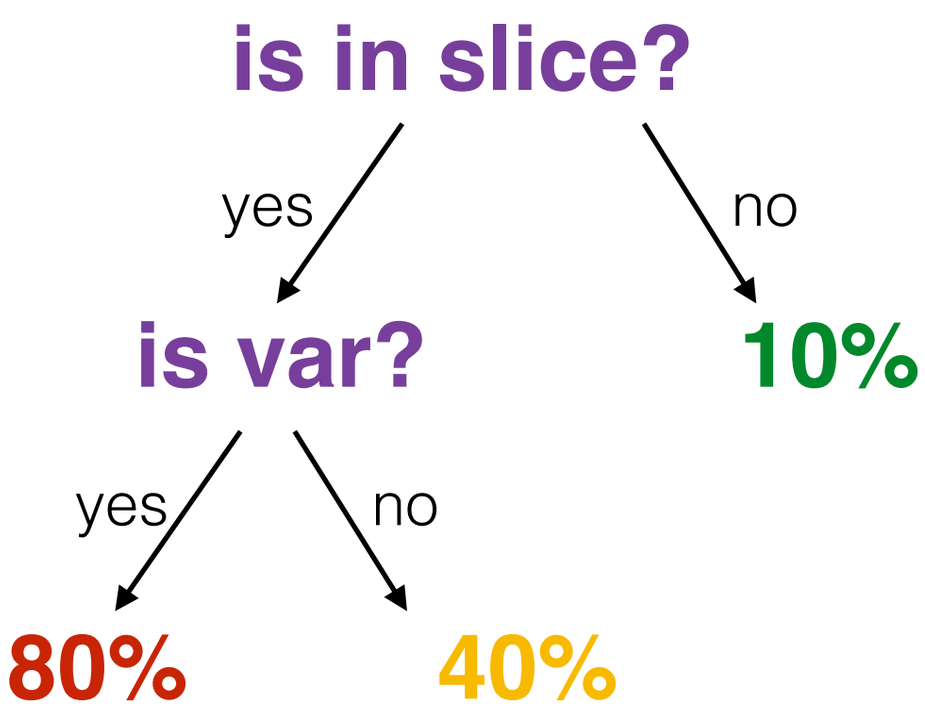


- Models



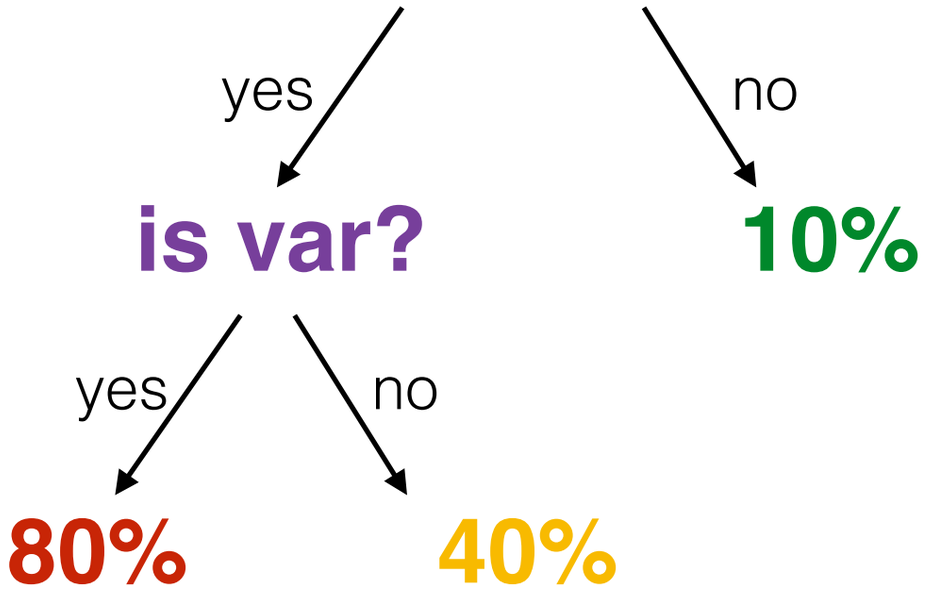
- Models
- Preliminary results

Decision Tree



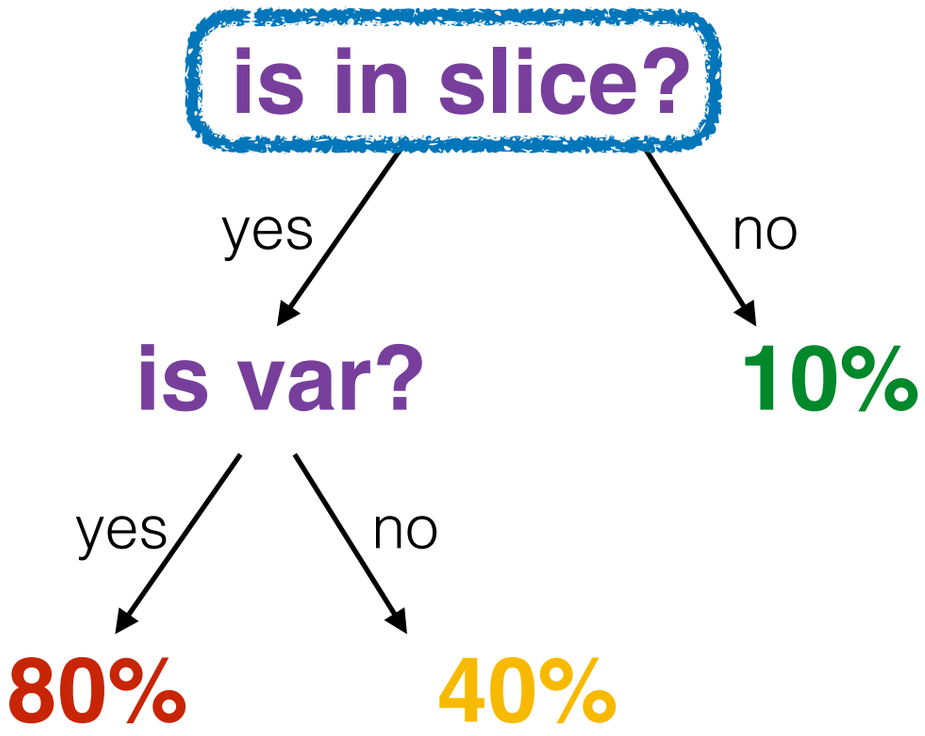
Decision Tree

NodeKind	...	Slice	...
BinaryOp	...	Yes	...



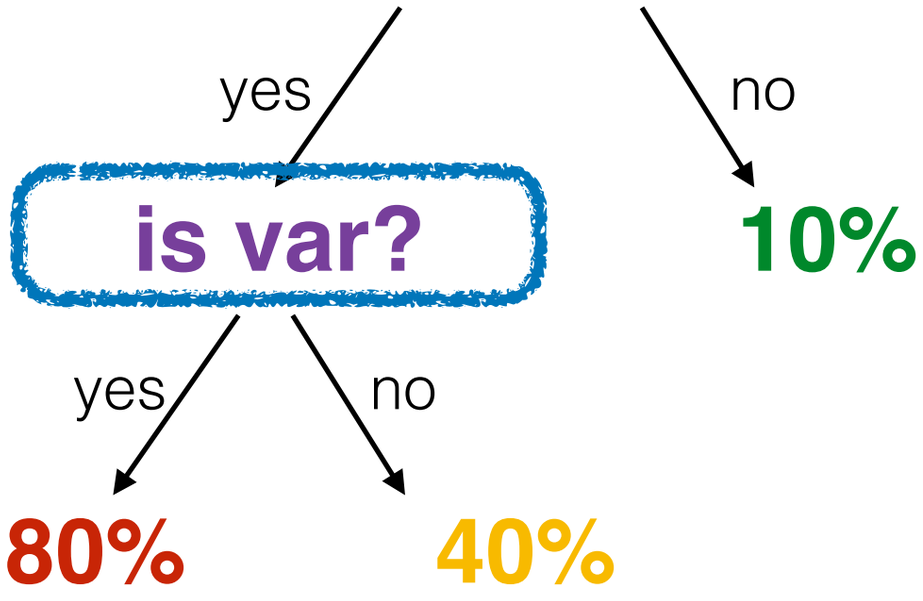
Decision Tree

NodeKind	...	Slice	...
BinaryOp	...	Yes	...



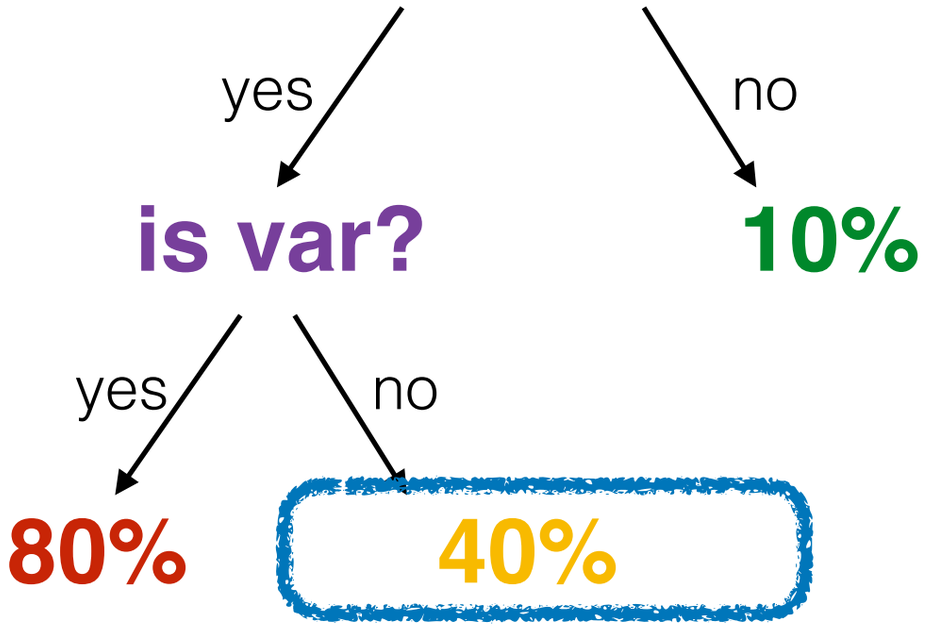
Decision Tree

NodeKind	...	Slice	...
BinaryOp	...	Yes	...

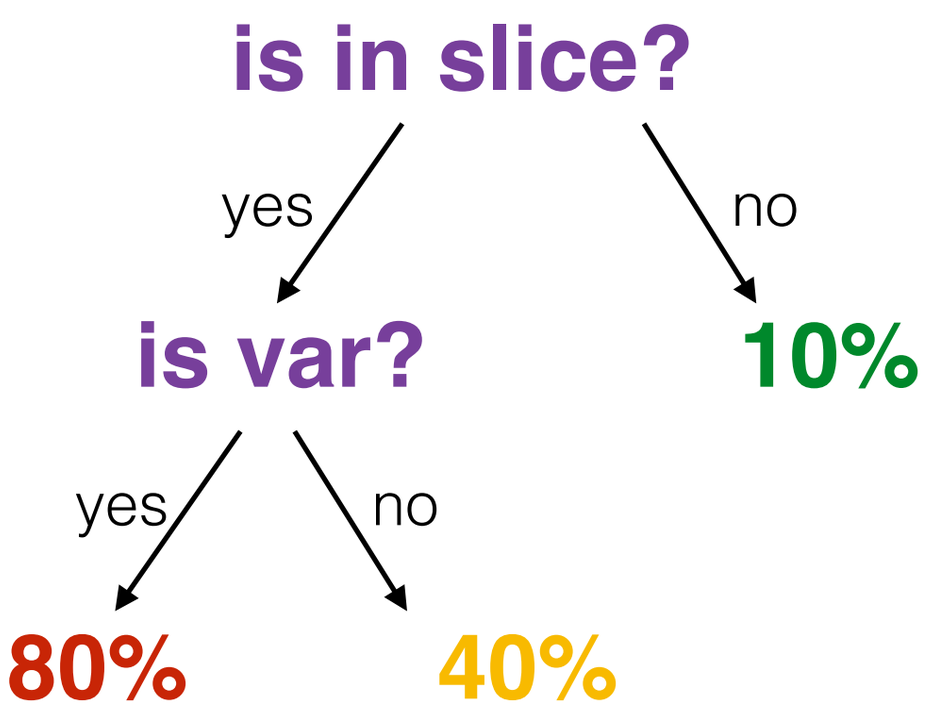


Decision Tree

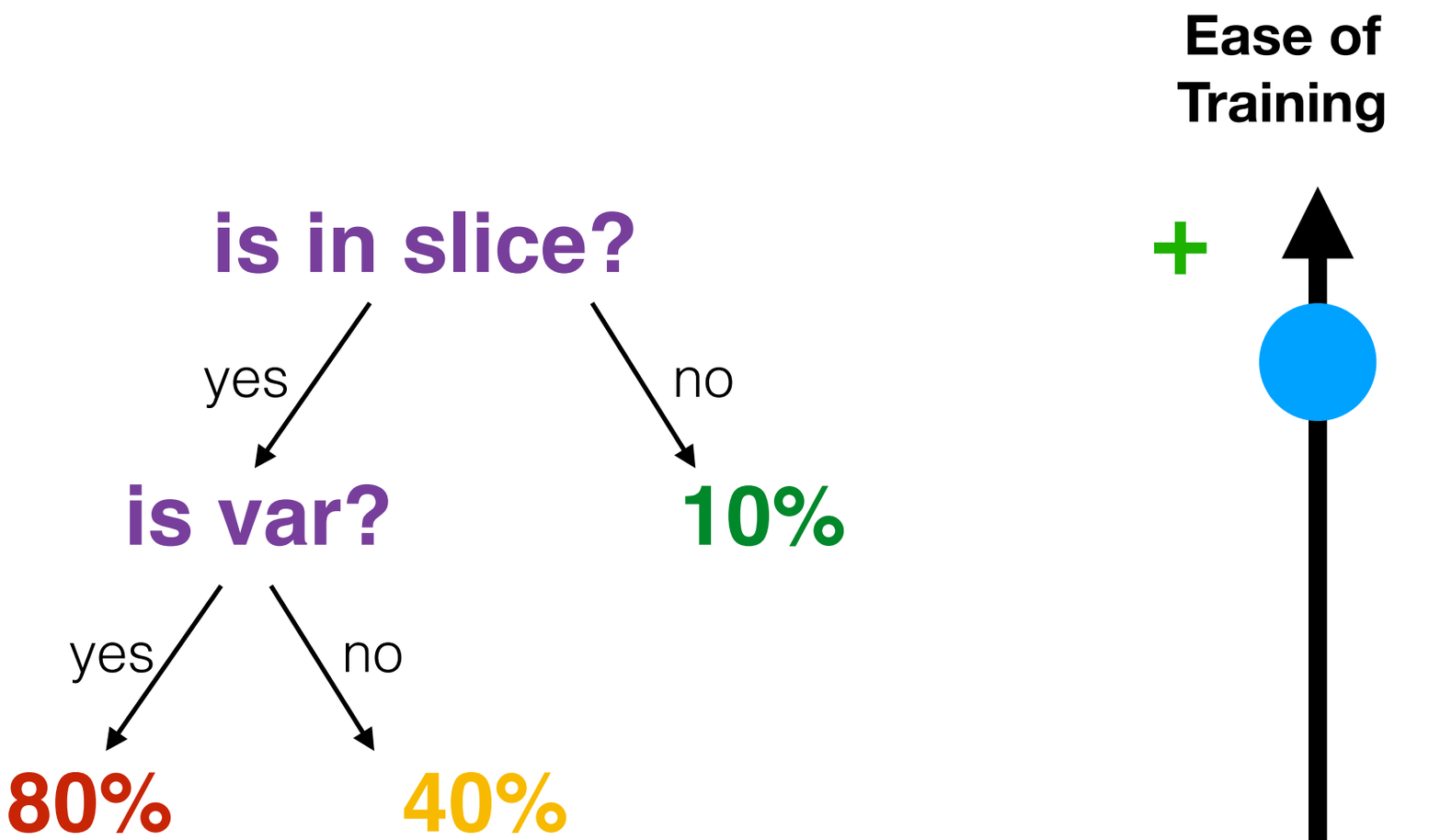
NodeKind	...	Slice	...
BinaryOp	...	Yes	...



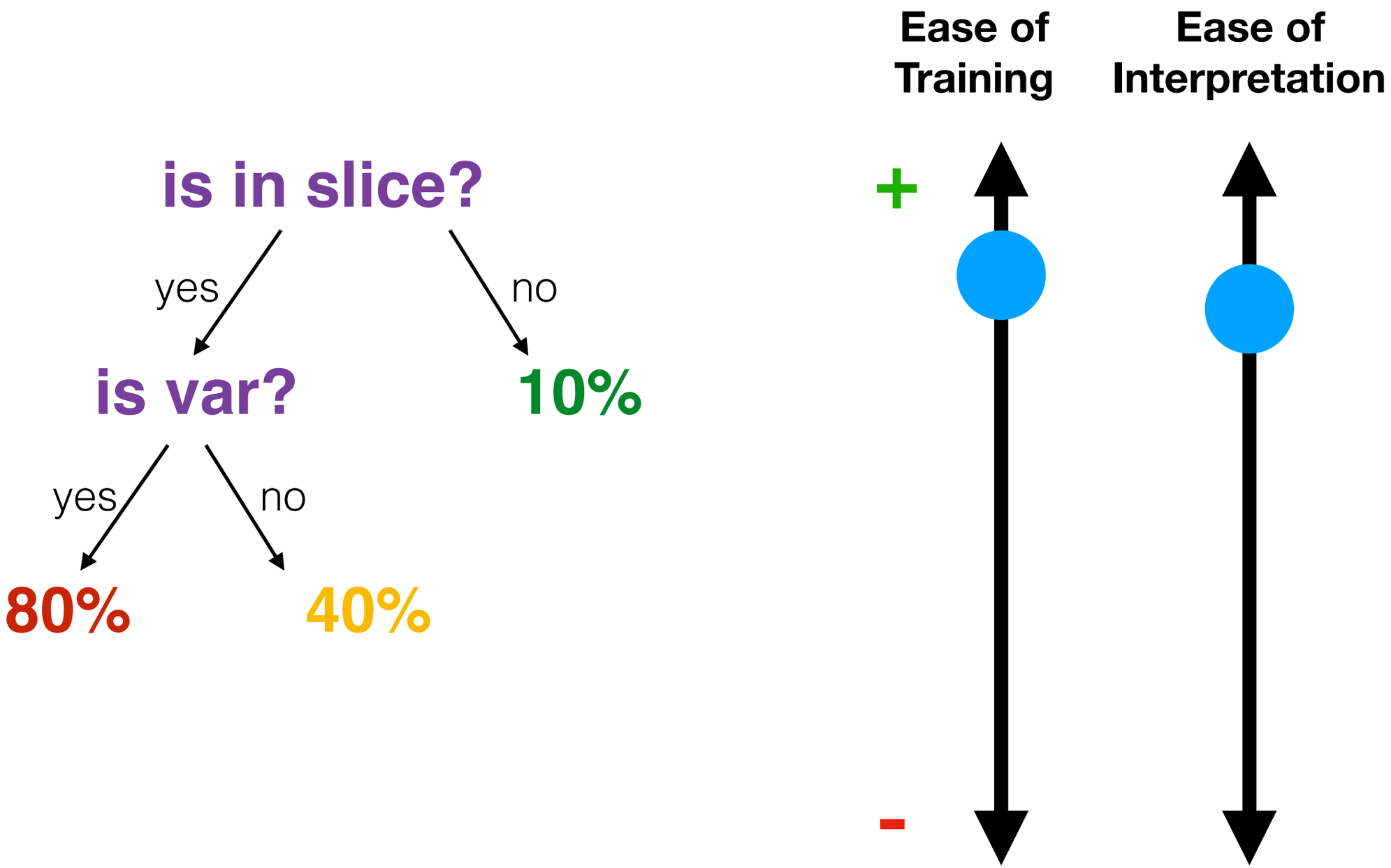
Decision Tree



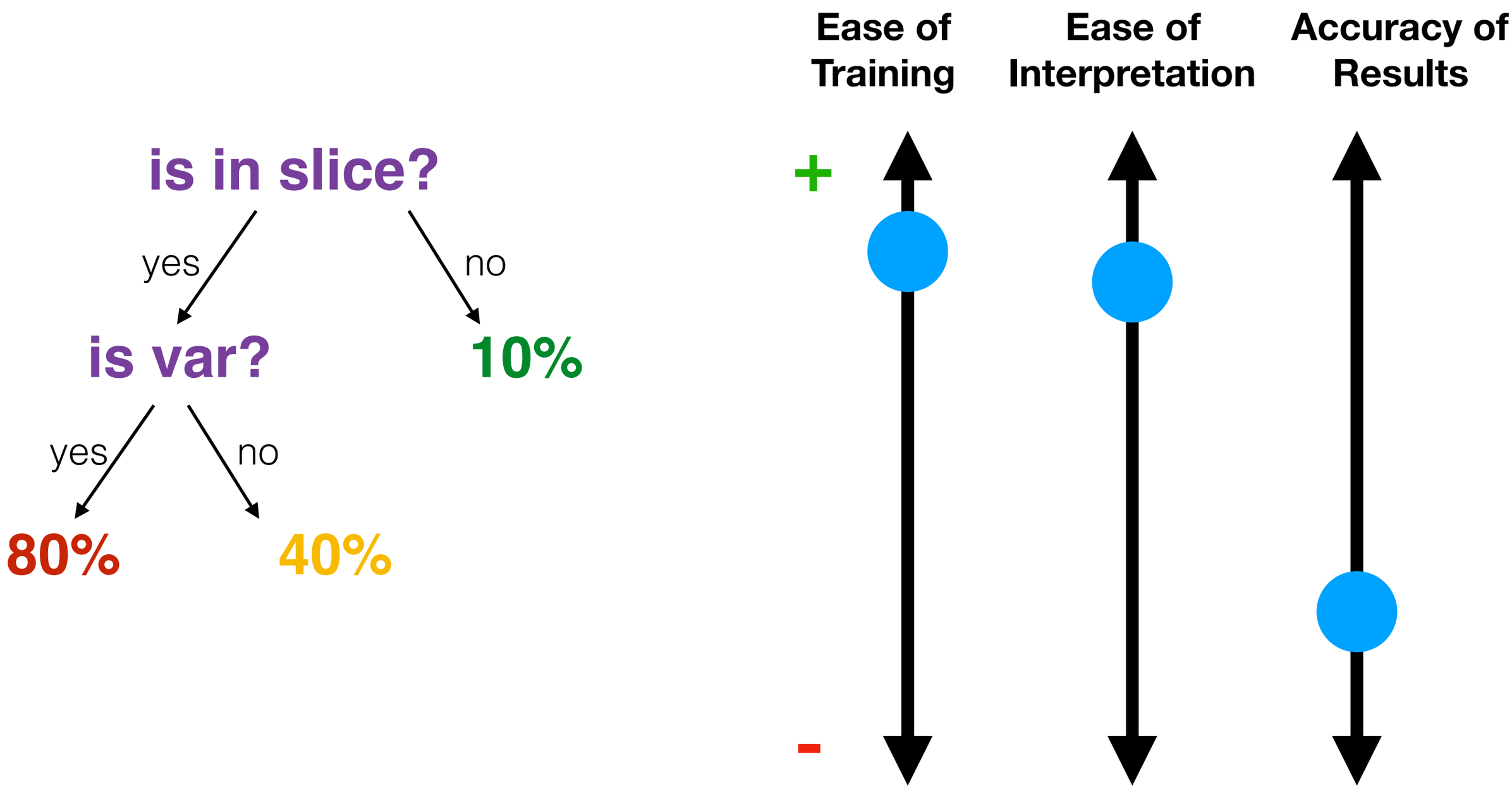
Decision Tree



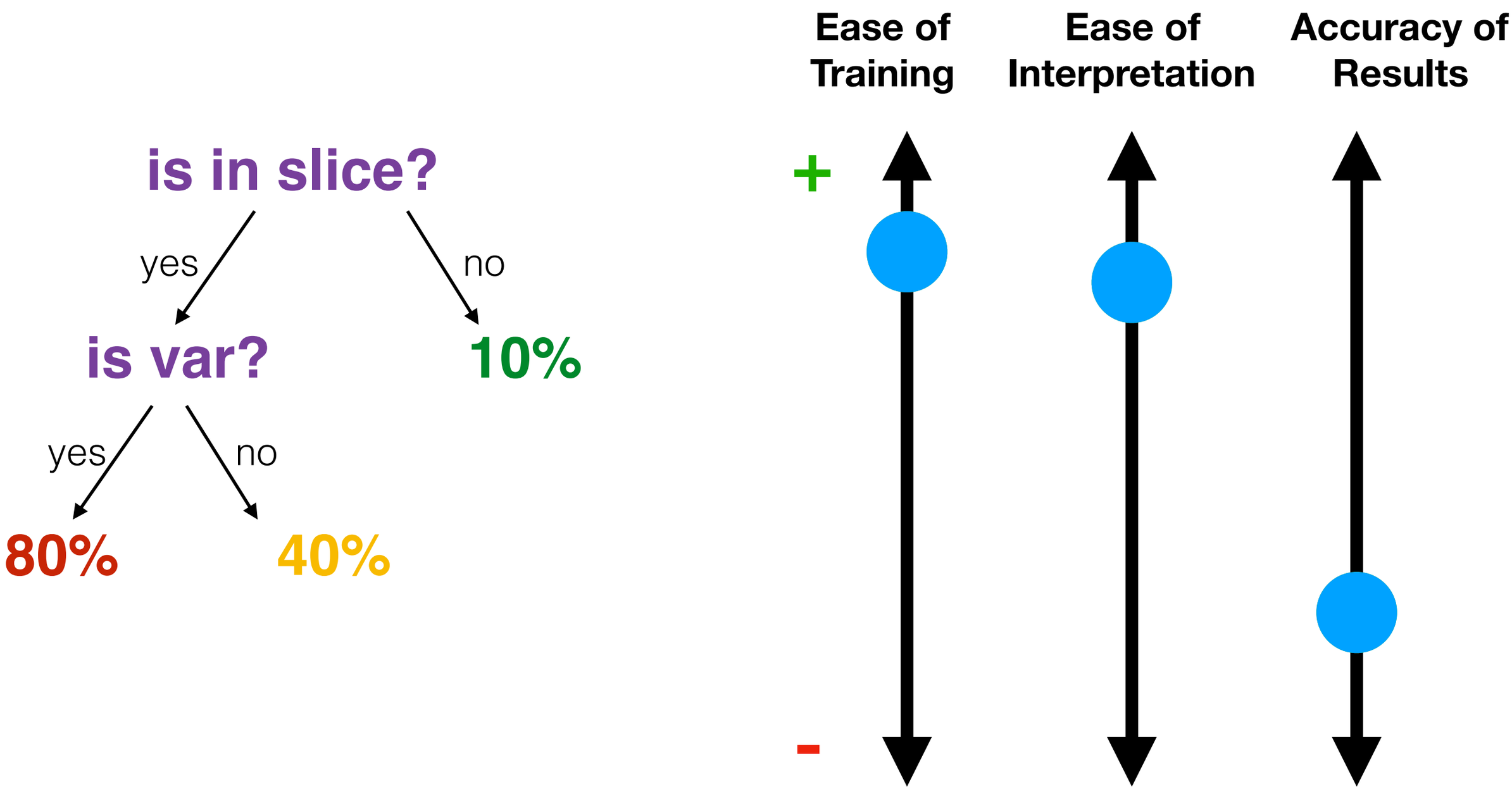
Decision Tree



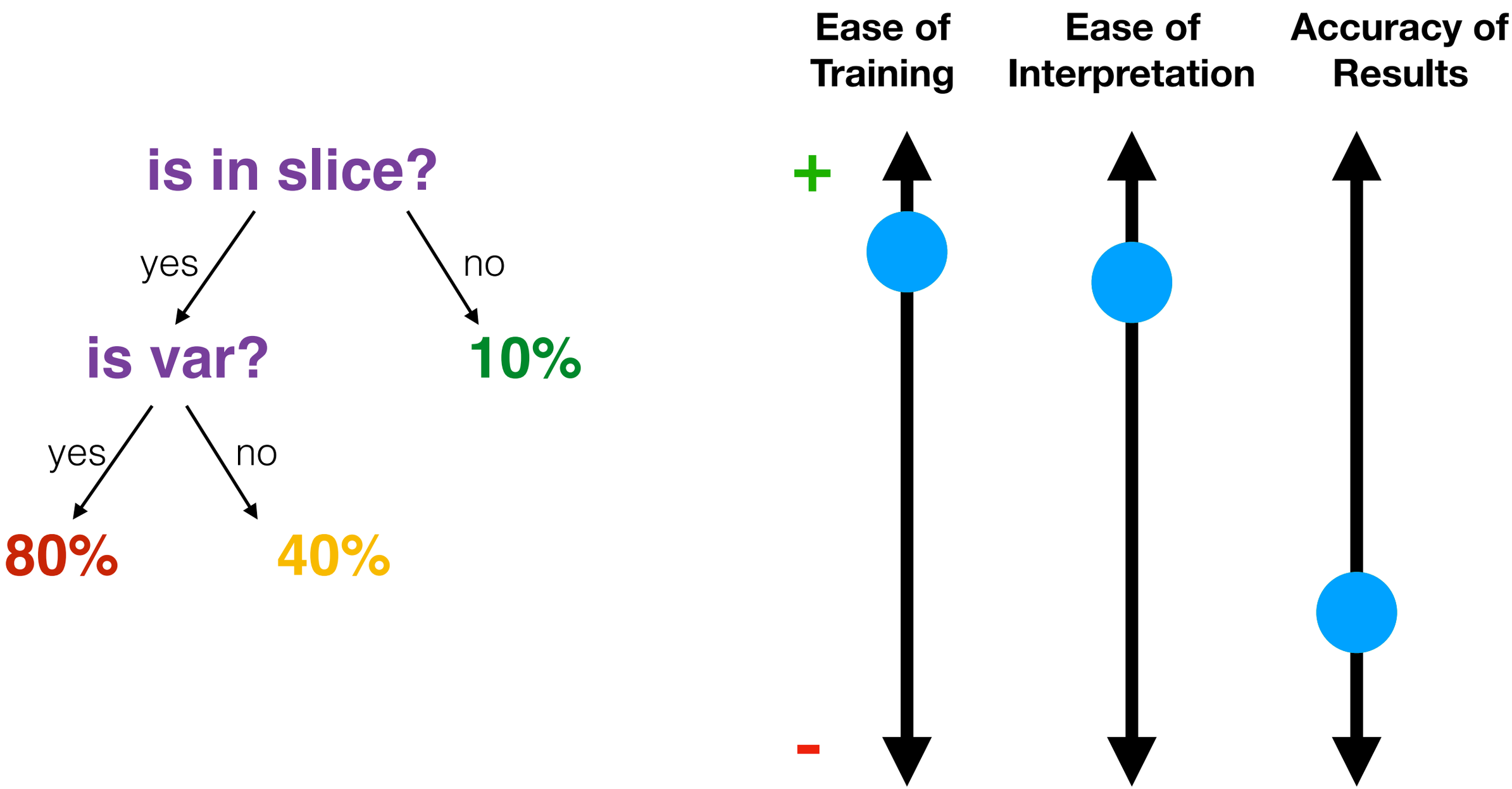
Decision Tree



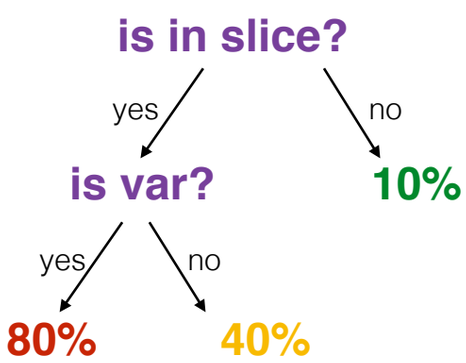
Decision Forests

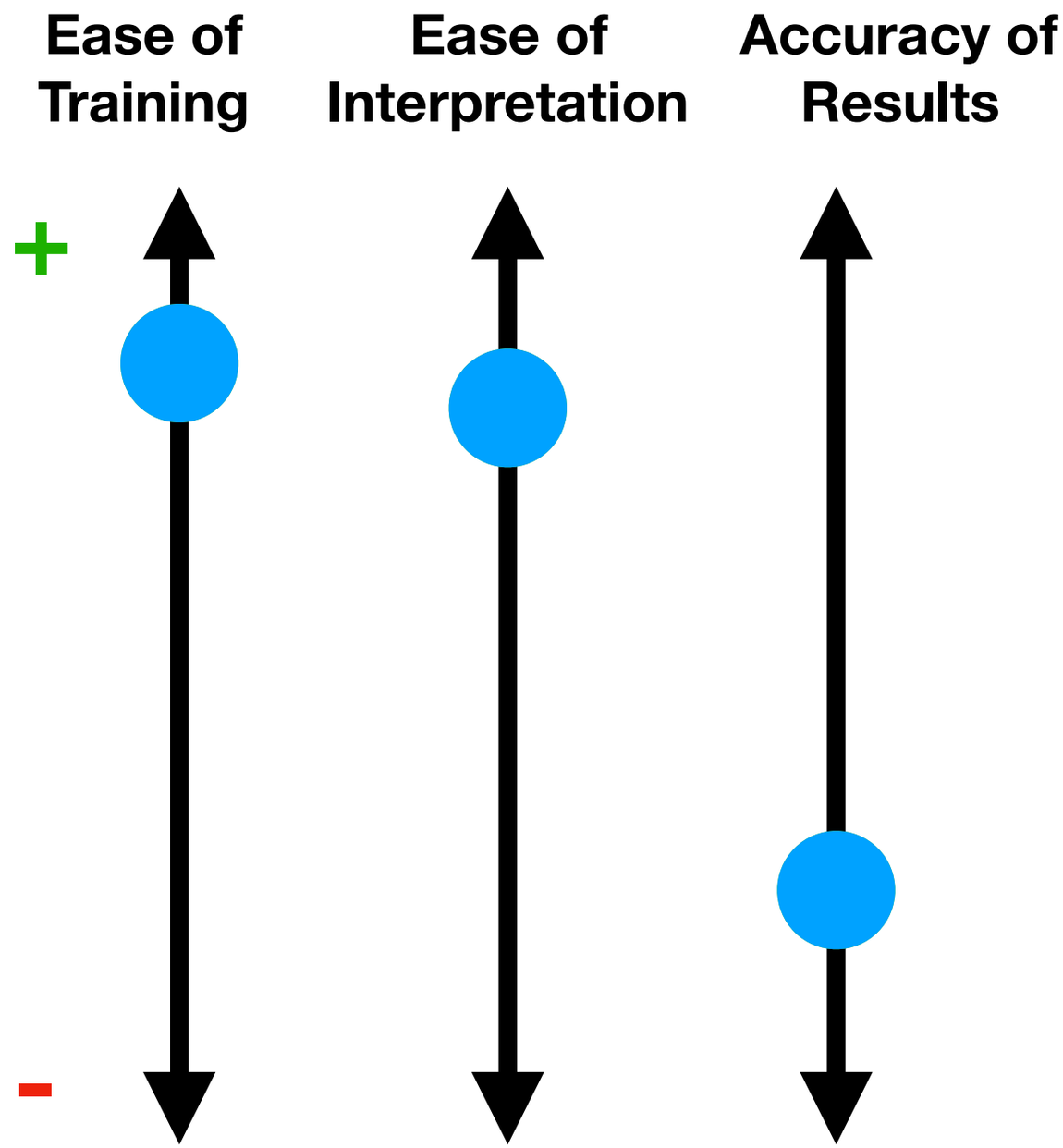


Decision Forests

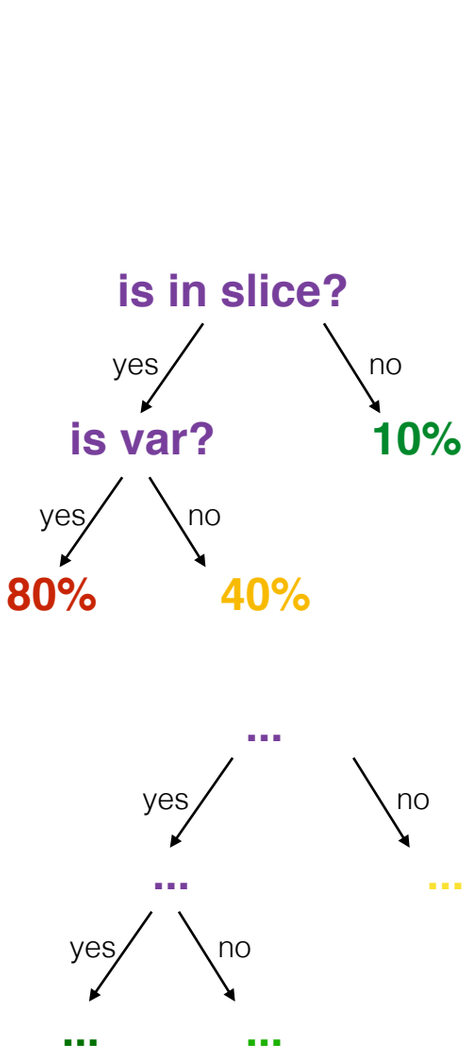


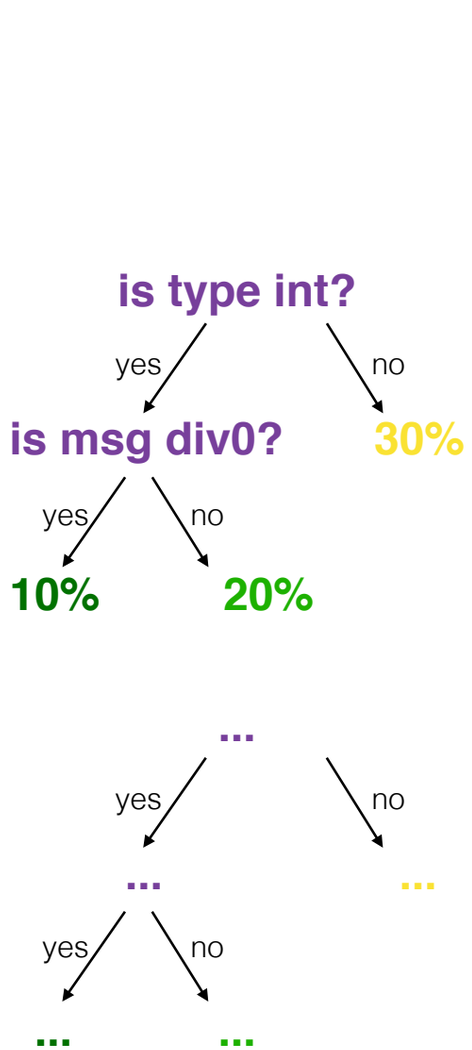
Decision Forests

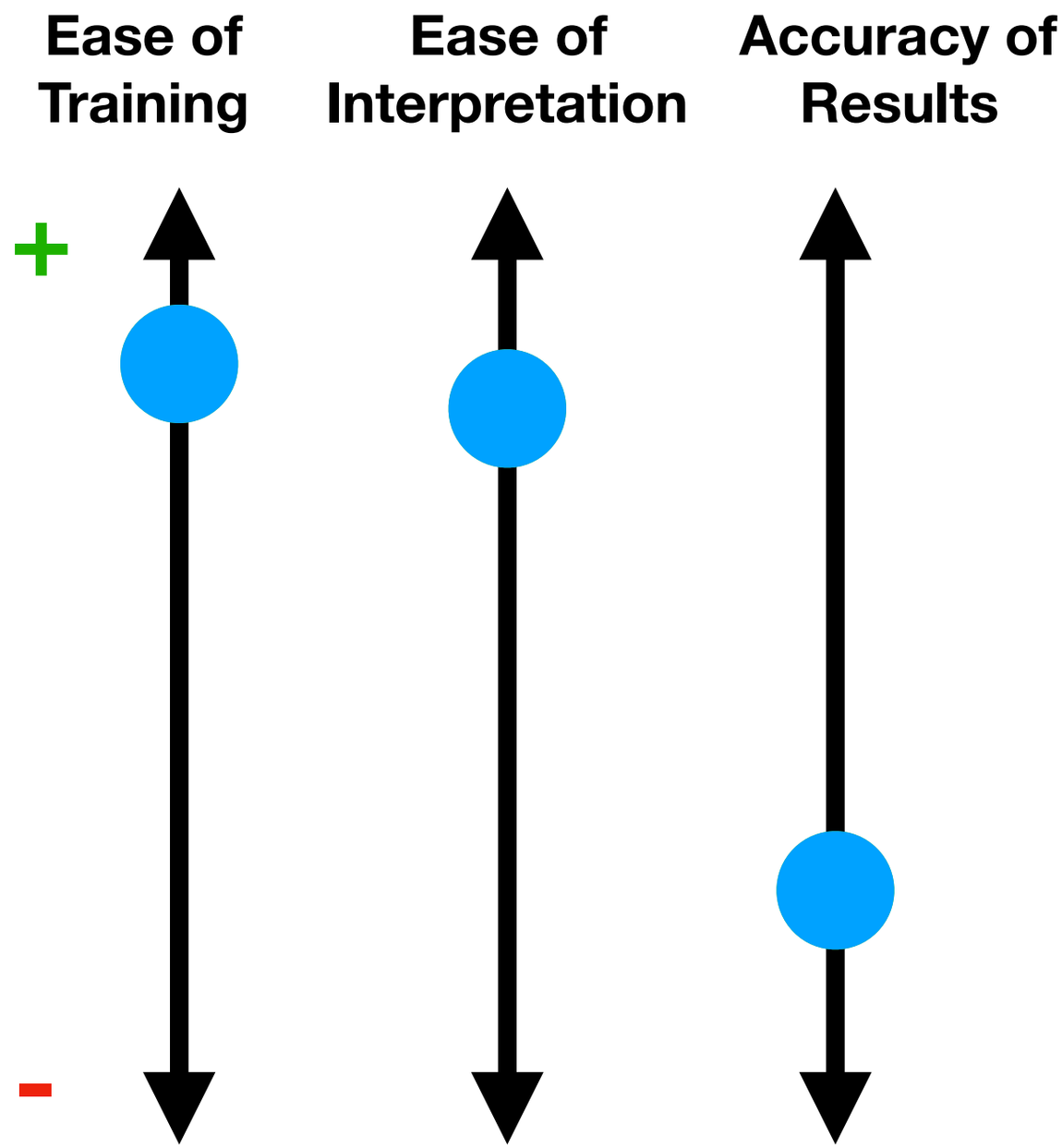




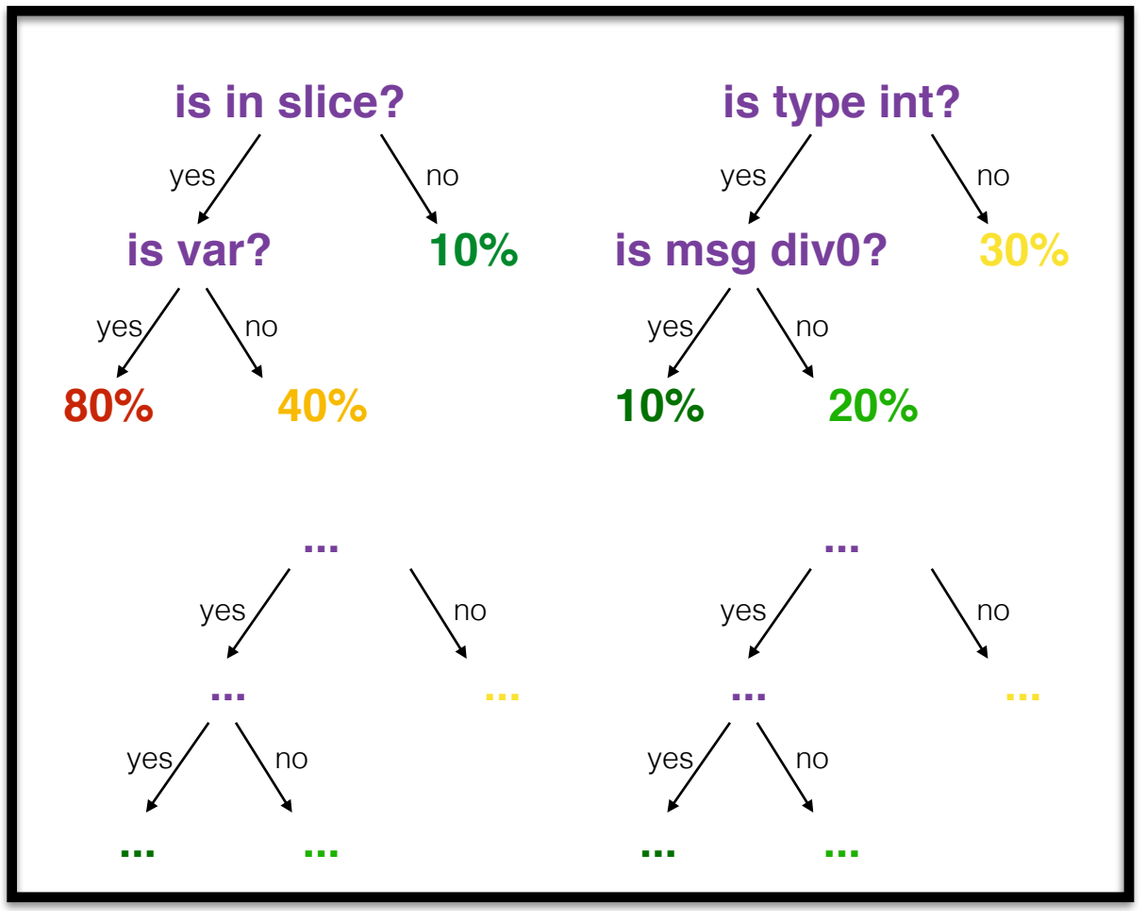
Decision Forests

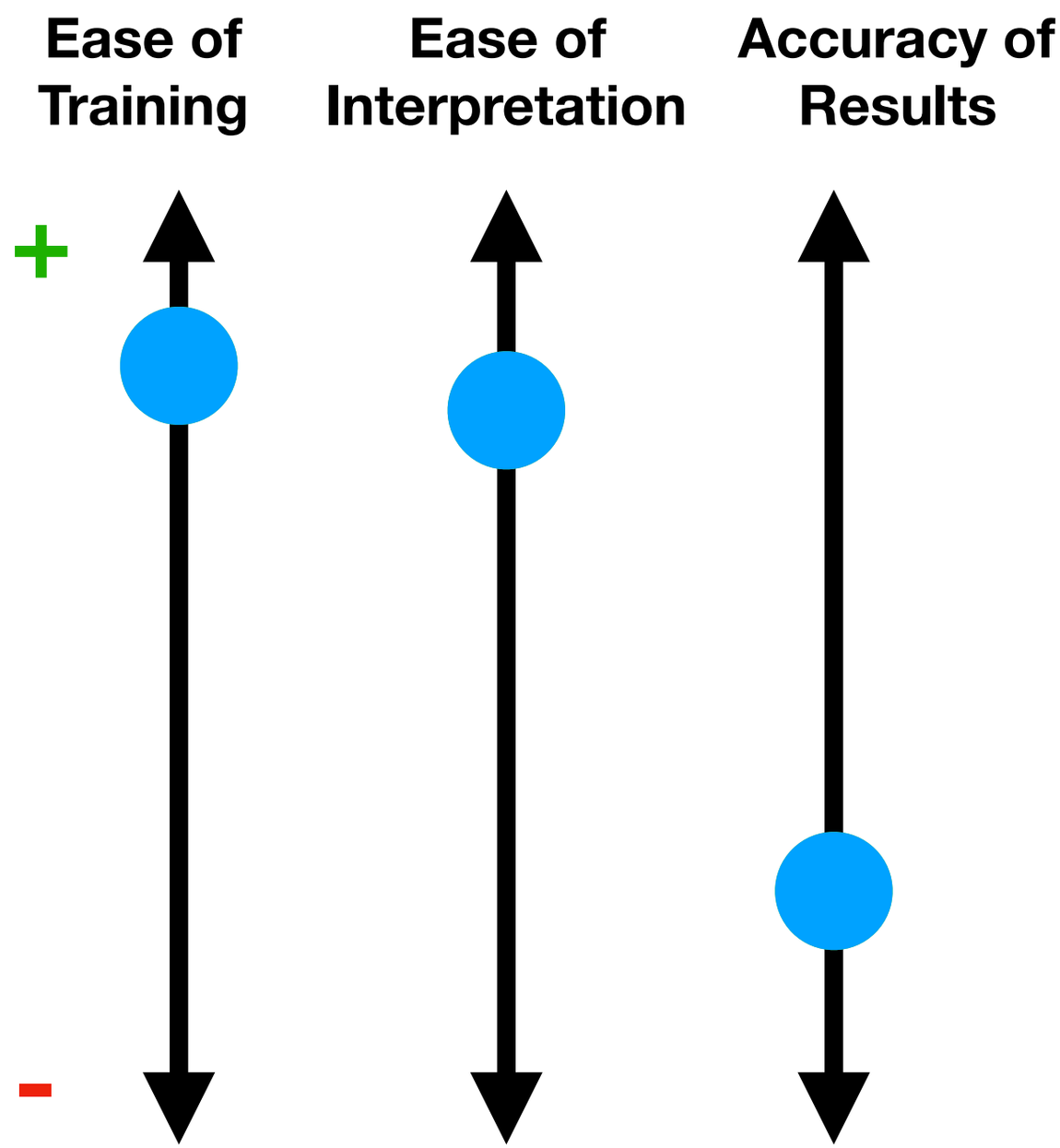




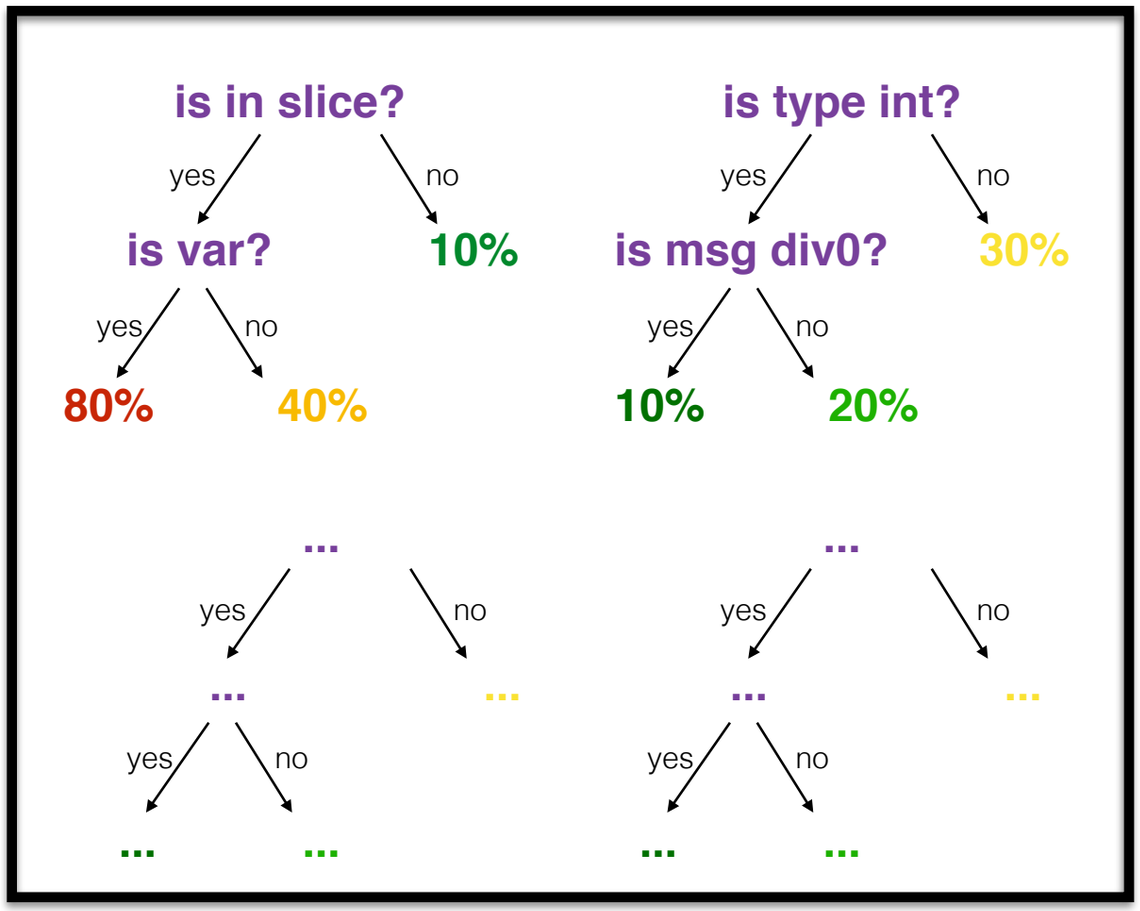


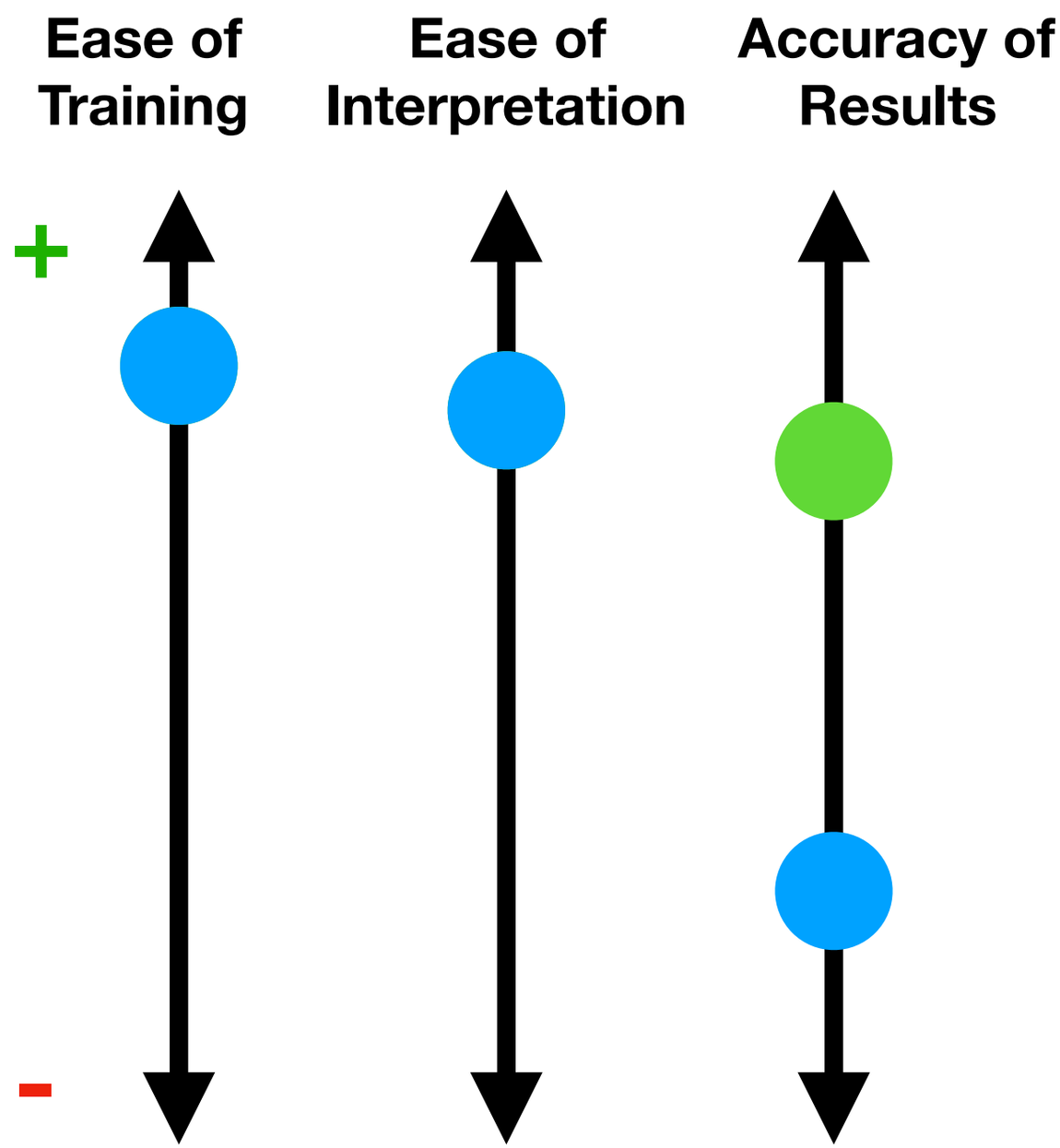
Decision Forests



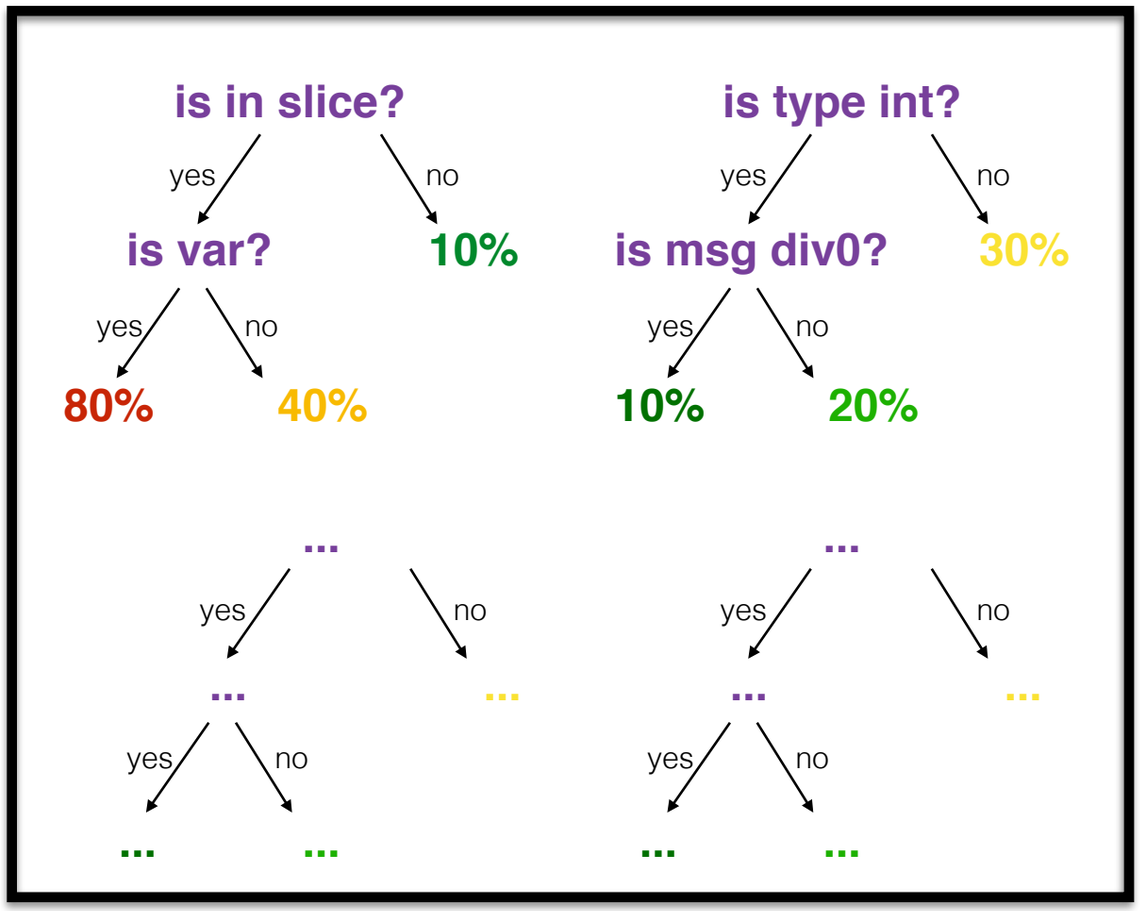


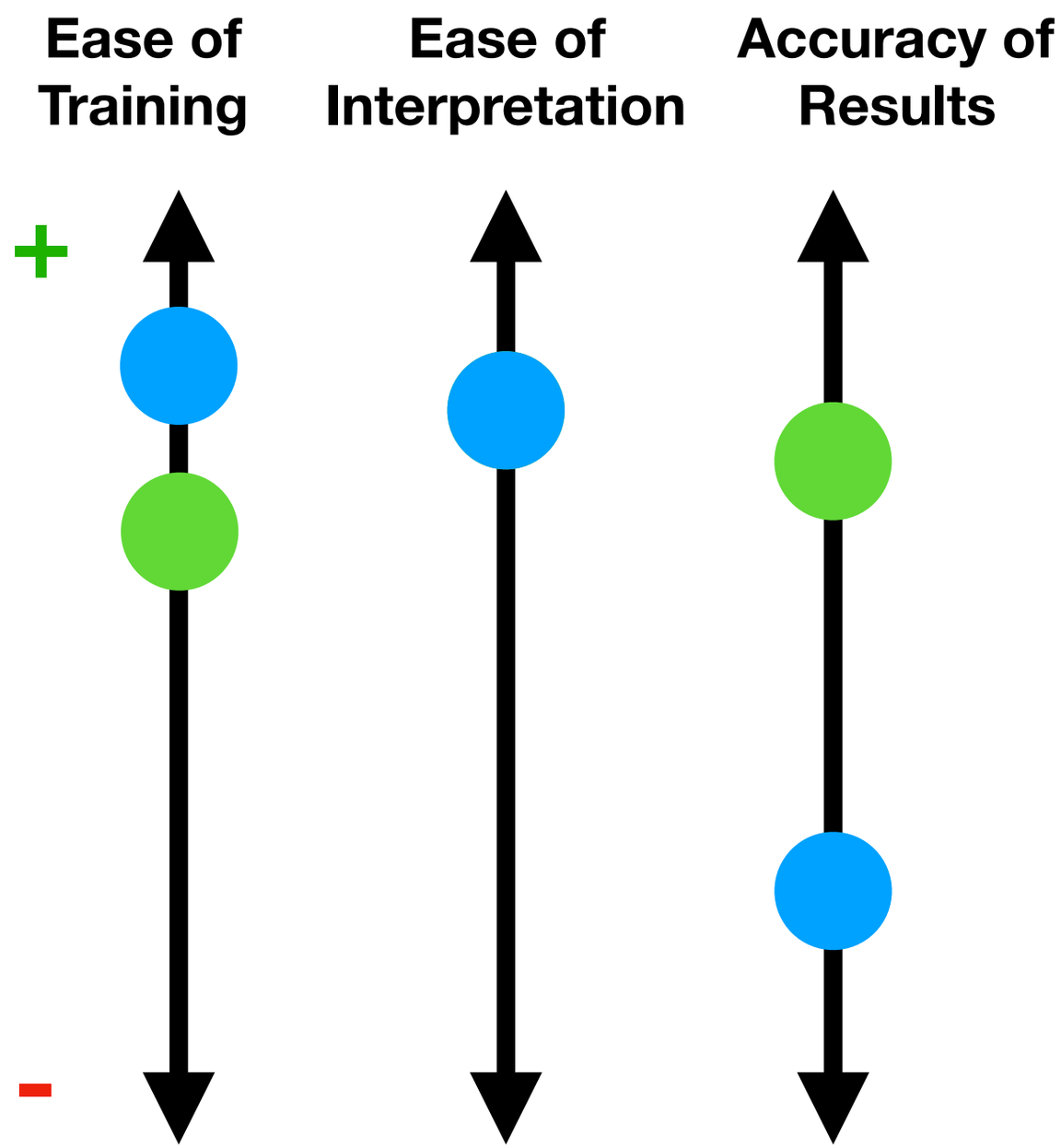
Decision Forests



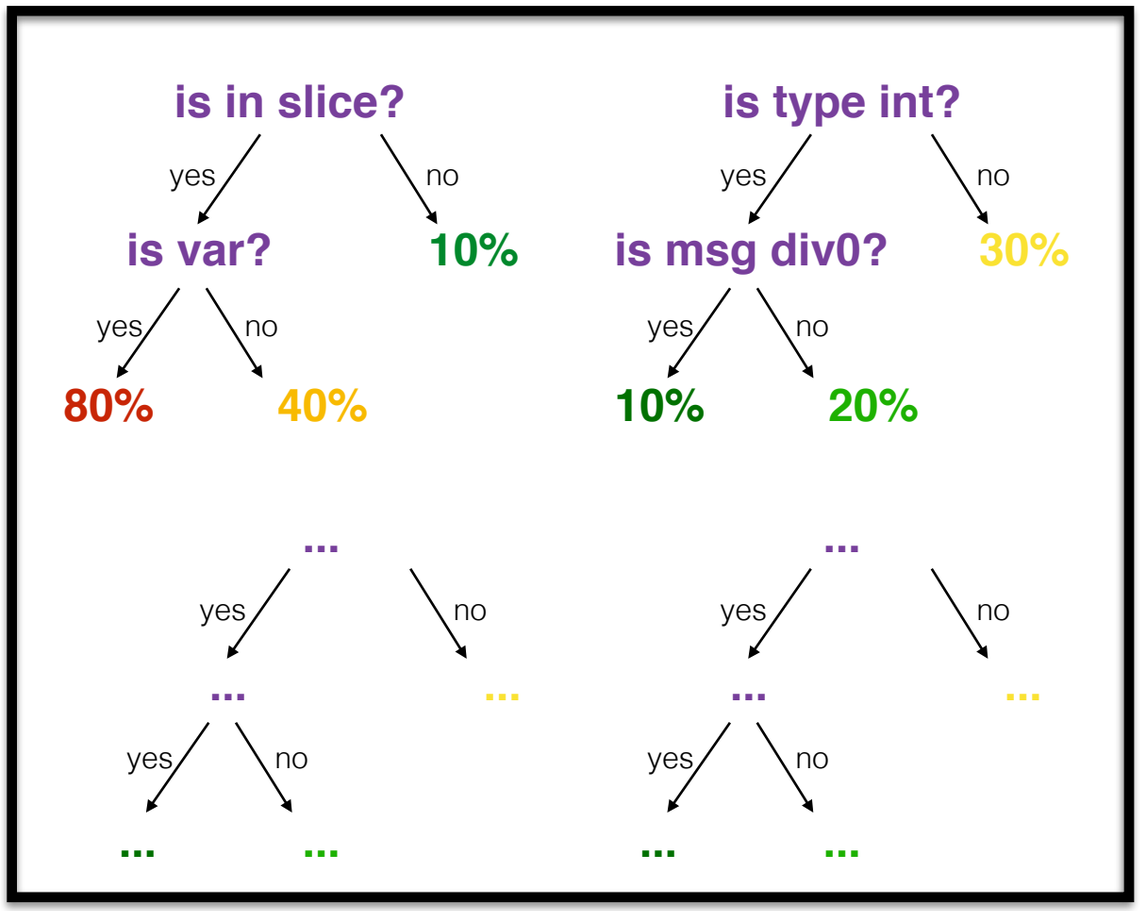


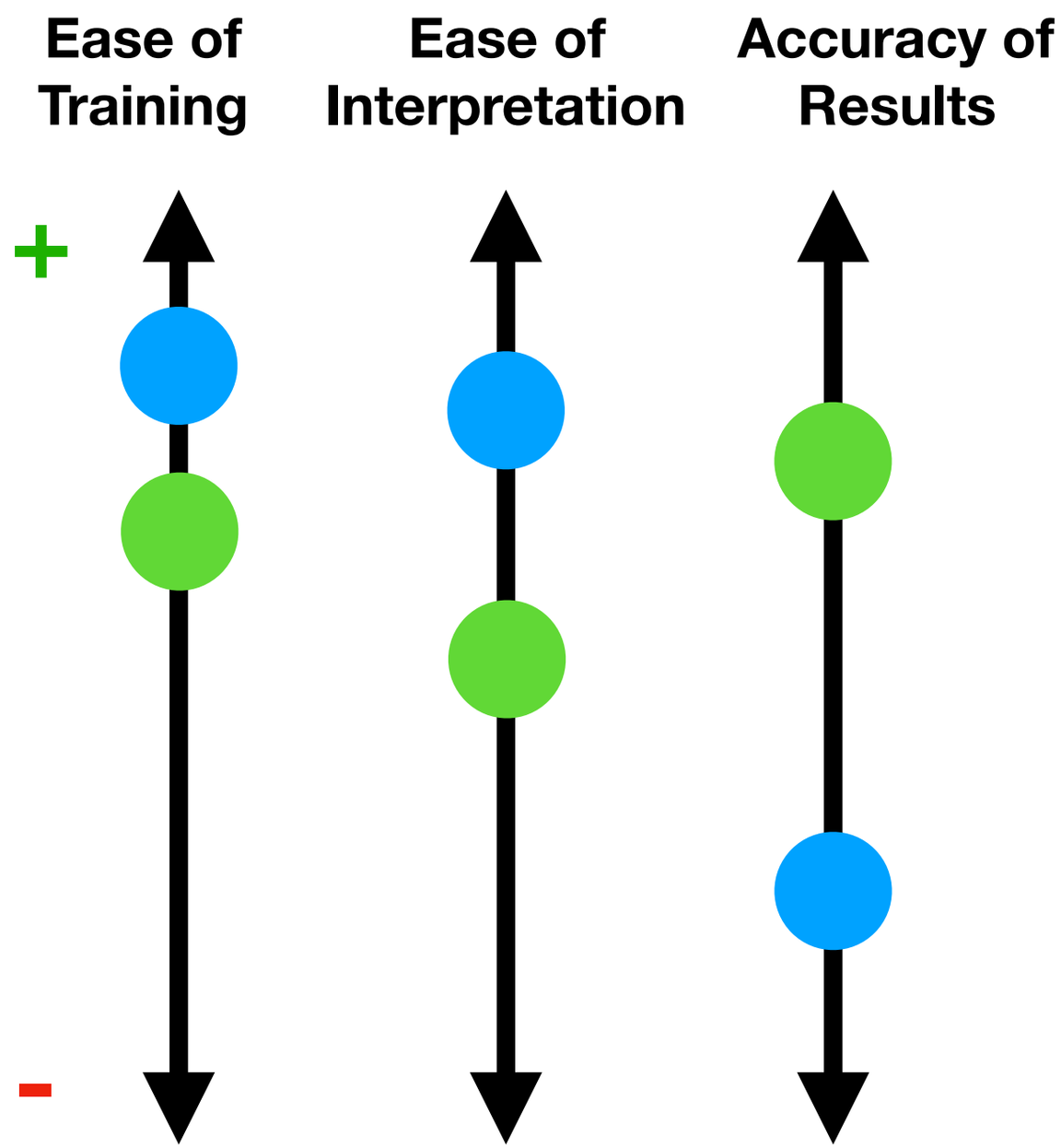
Decision Forests



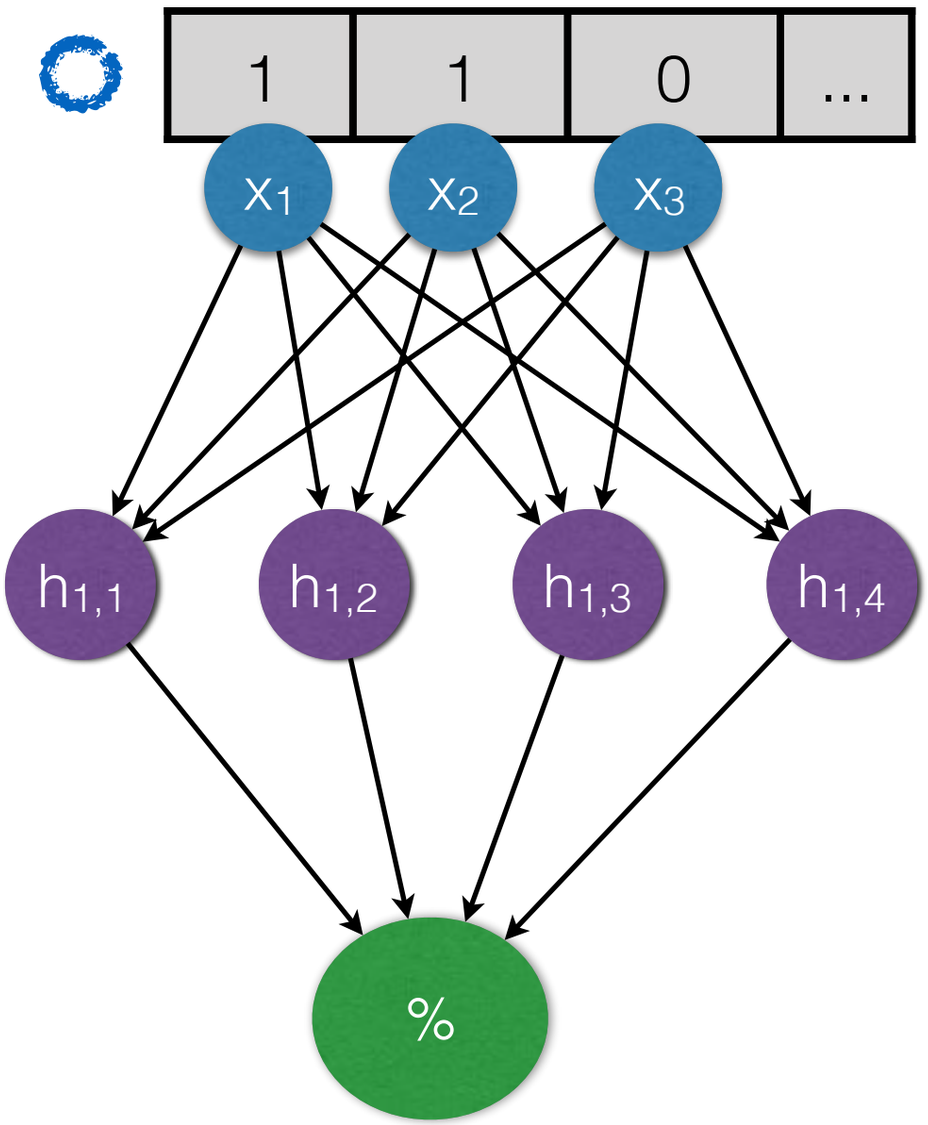


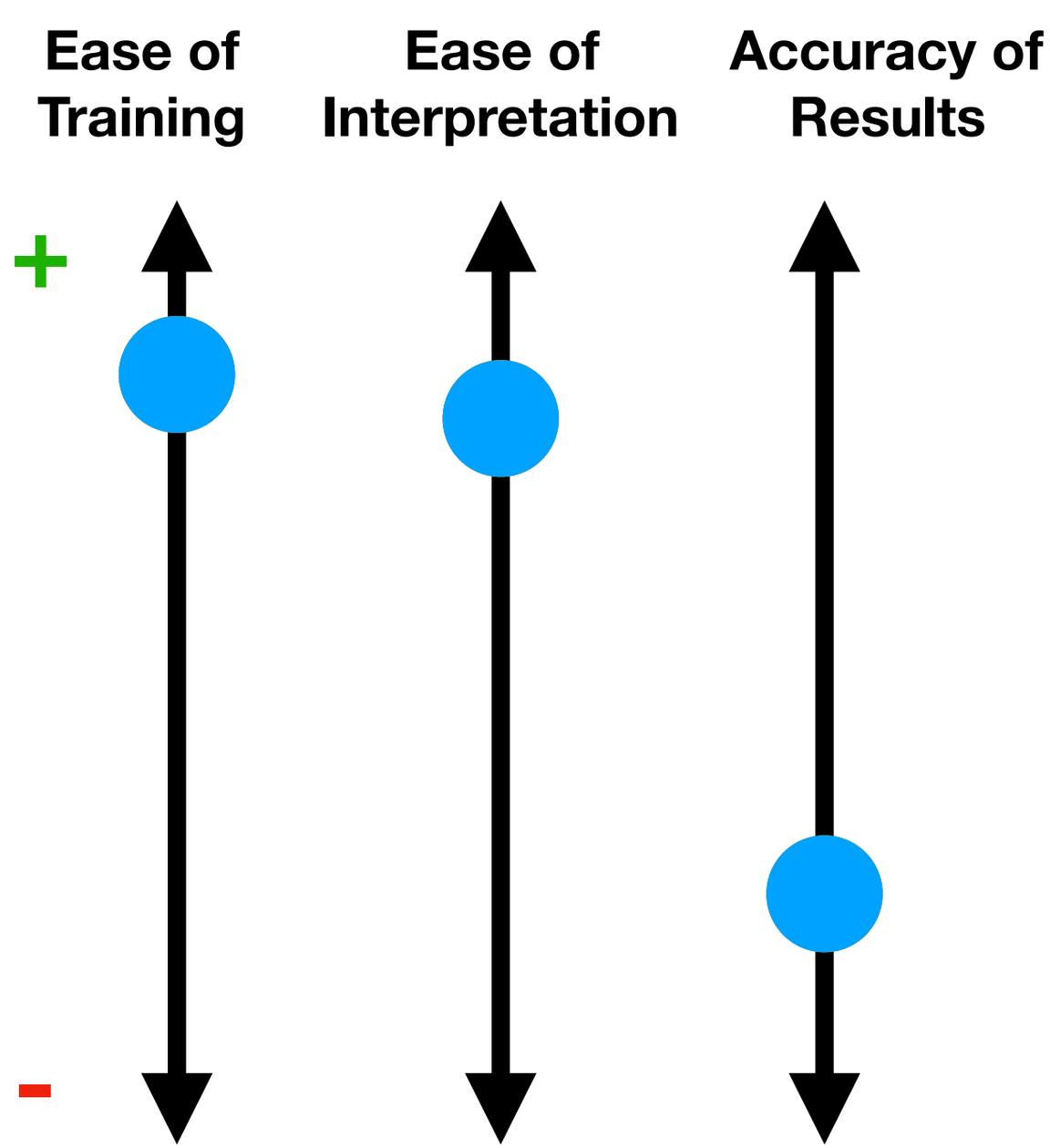
Decision Forests



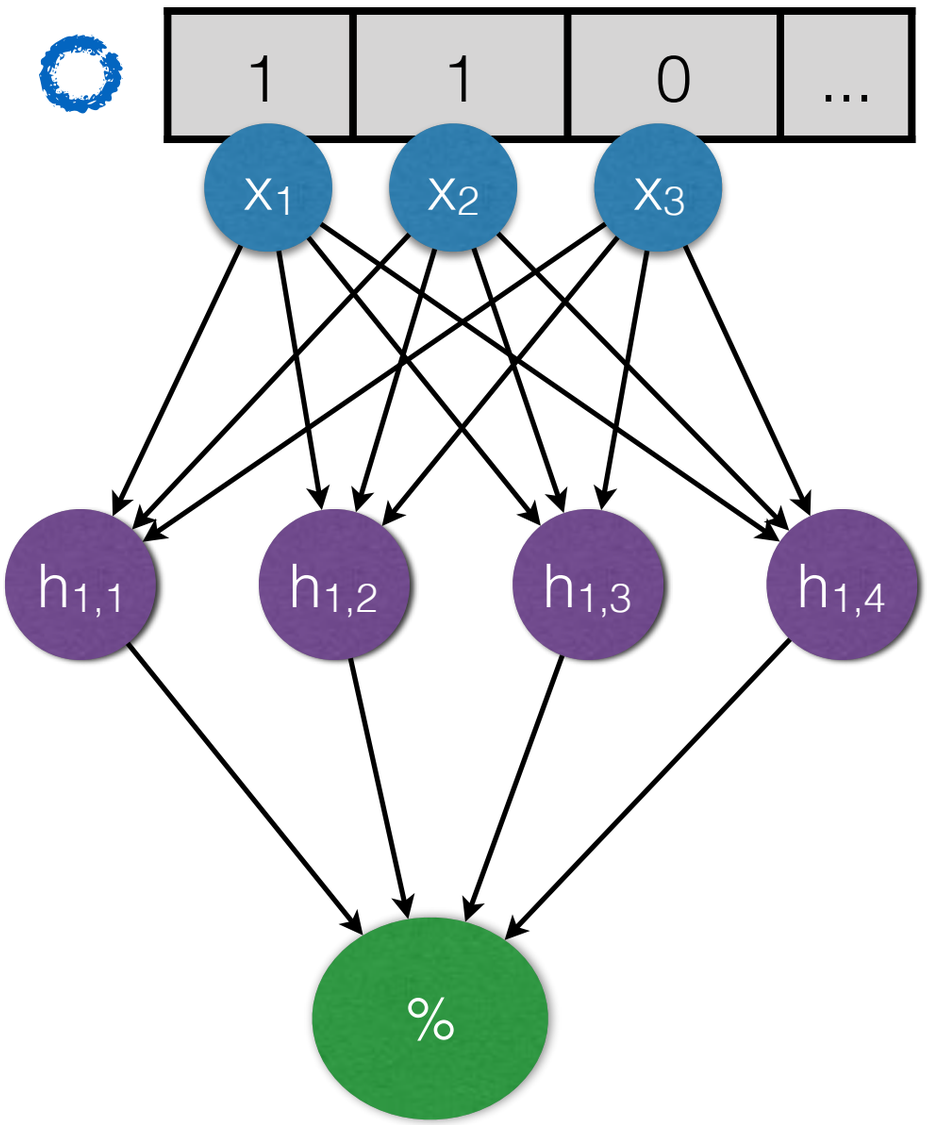


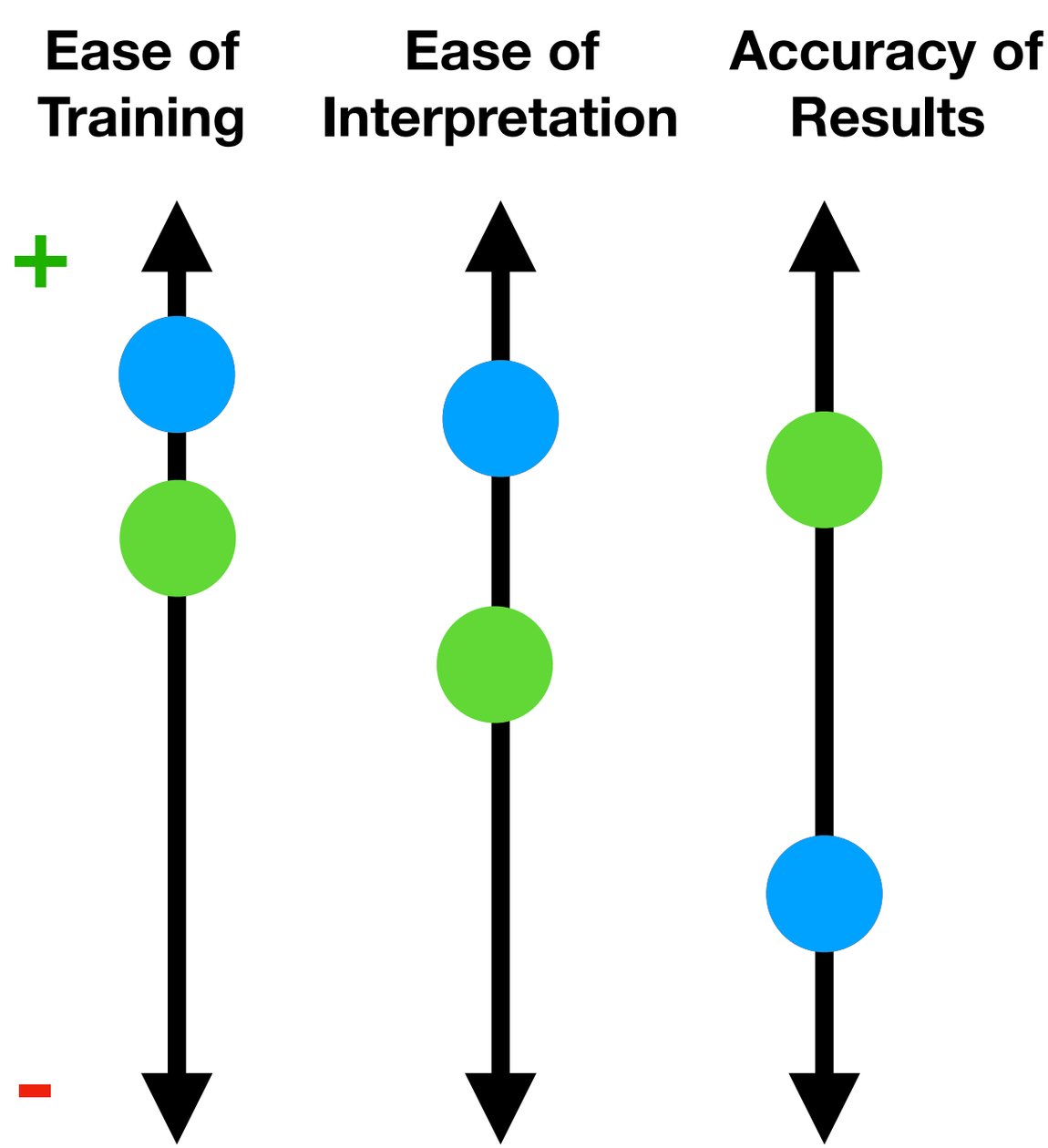
Multi-Layer Perceptron



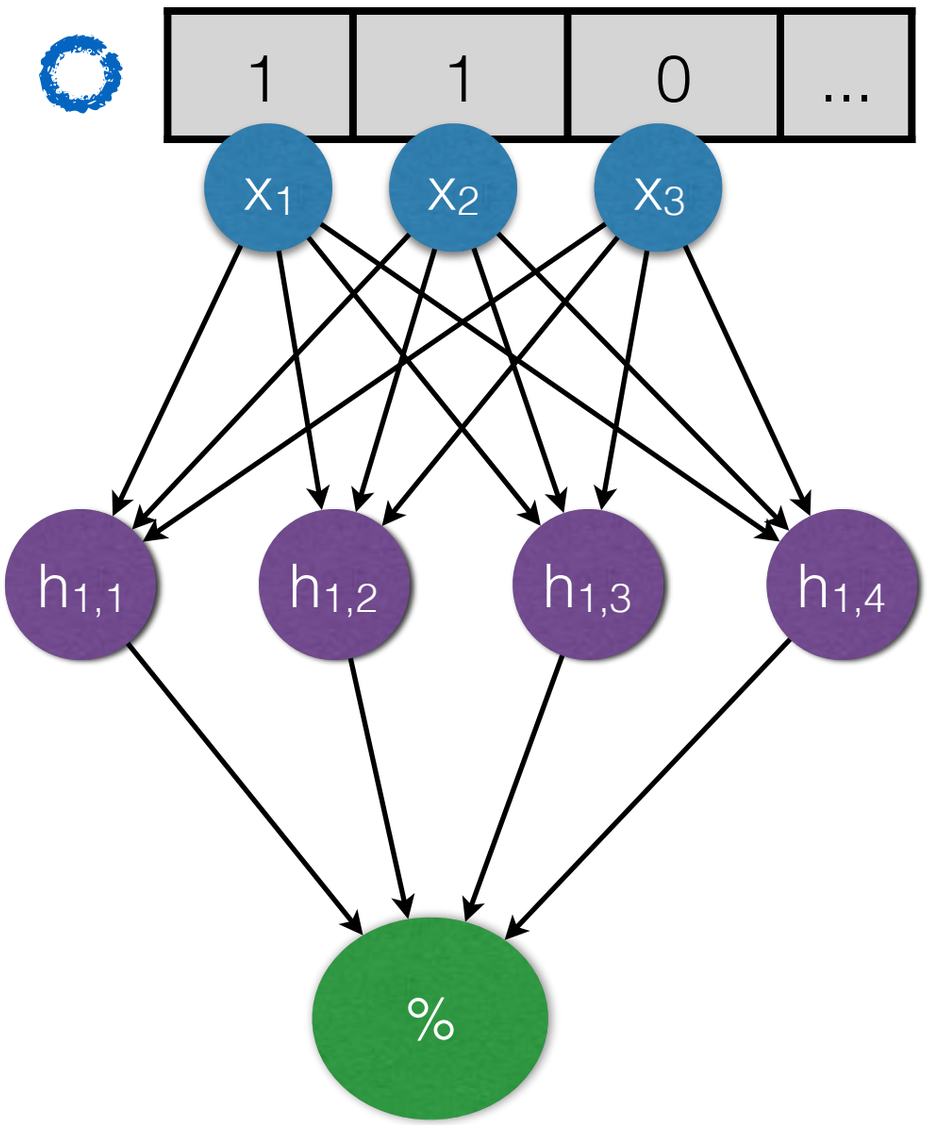


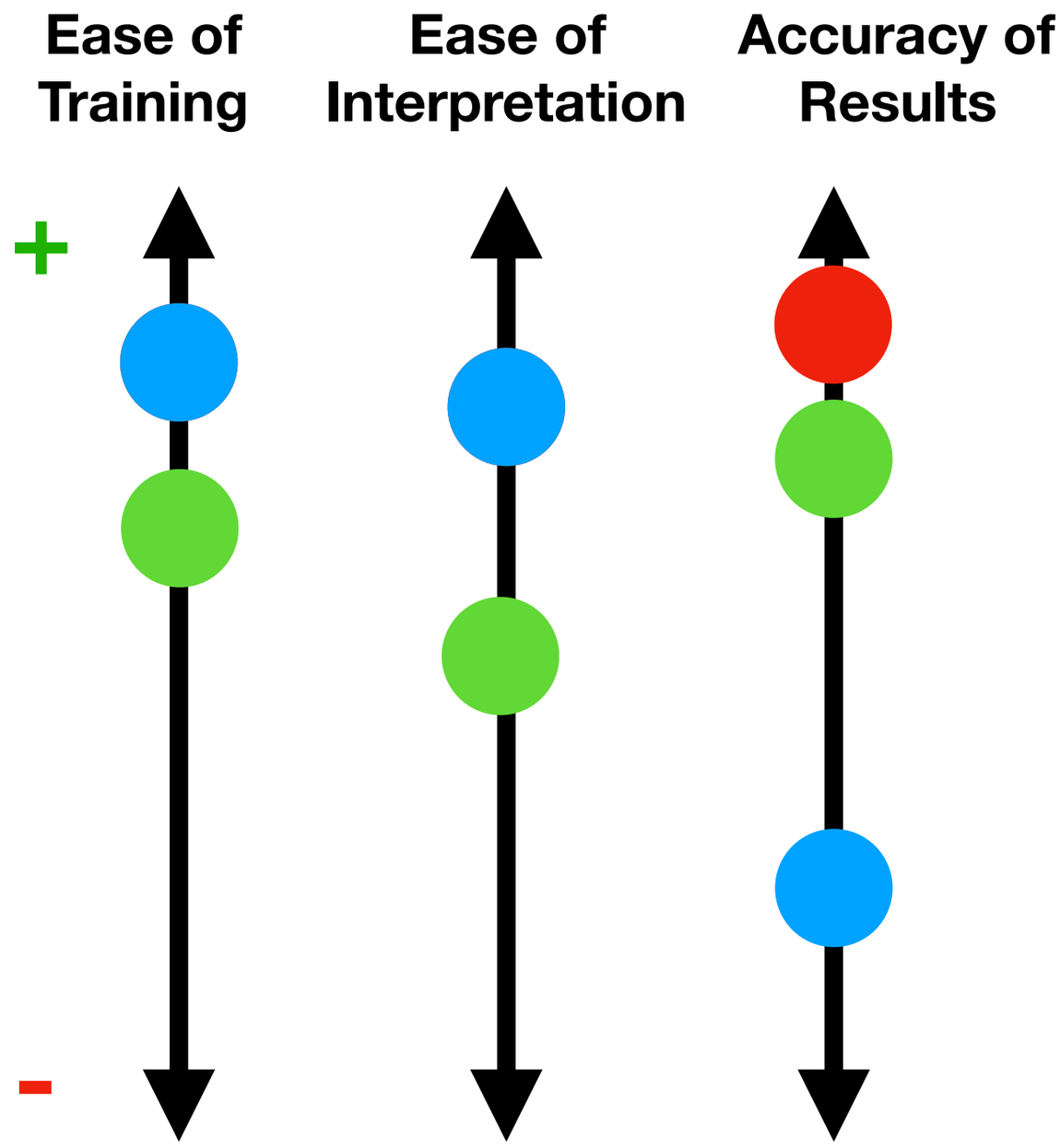
Multi-Layer Perceptron



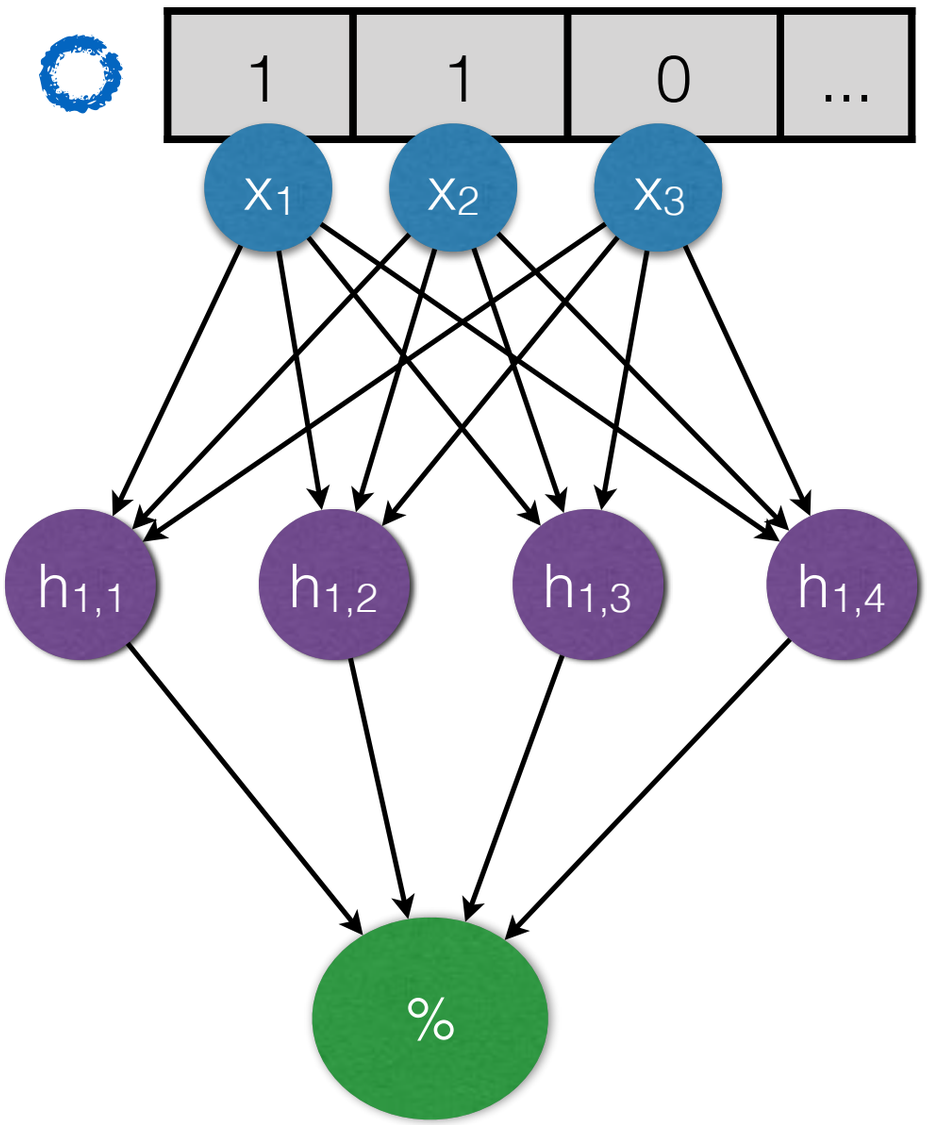


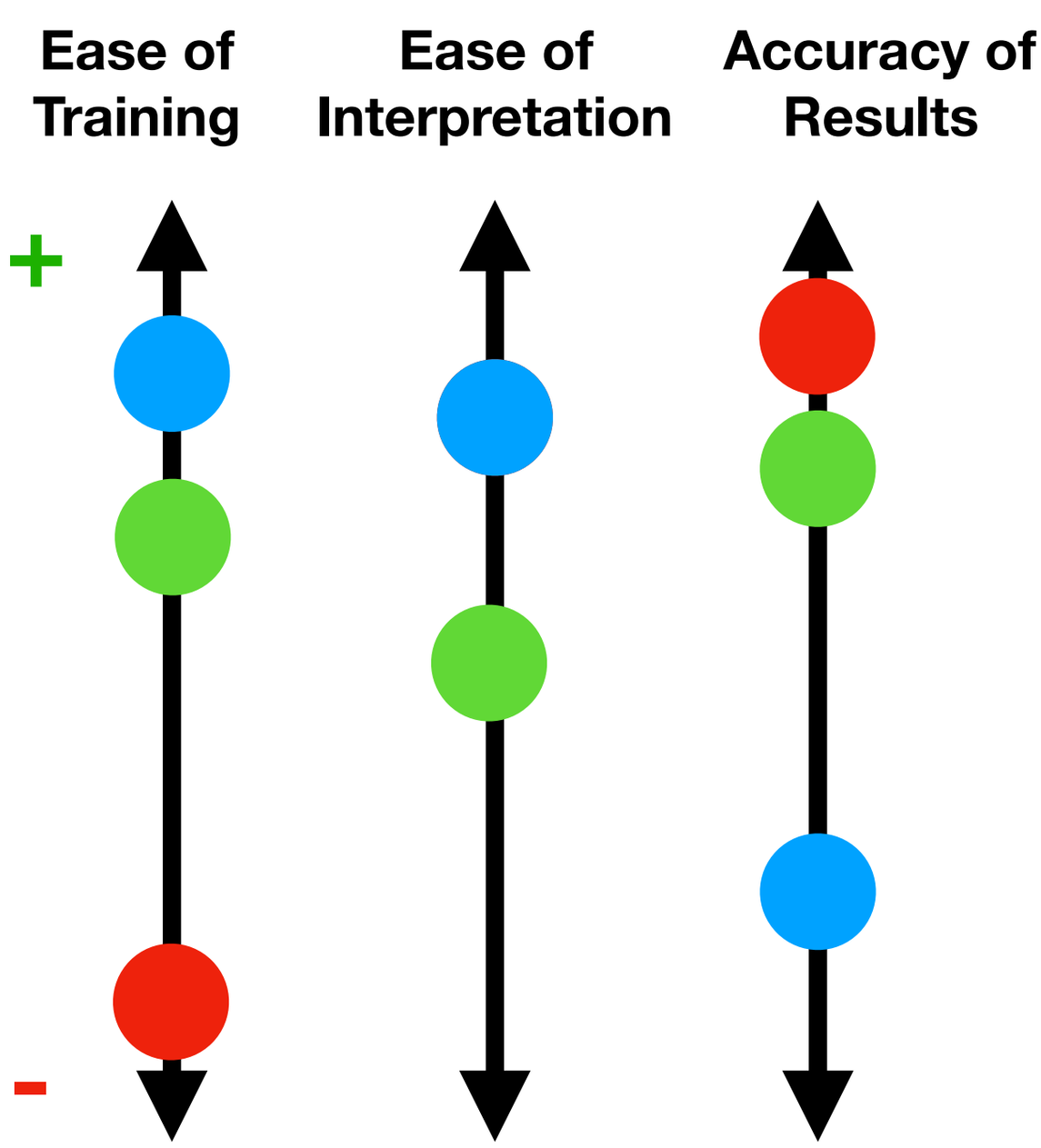
Multi-Layer Perceptron



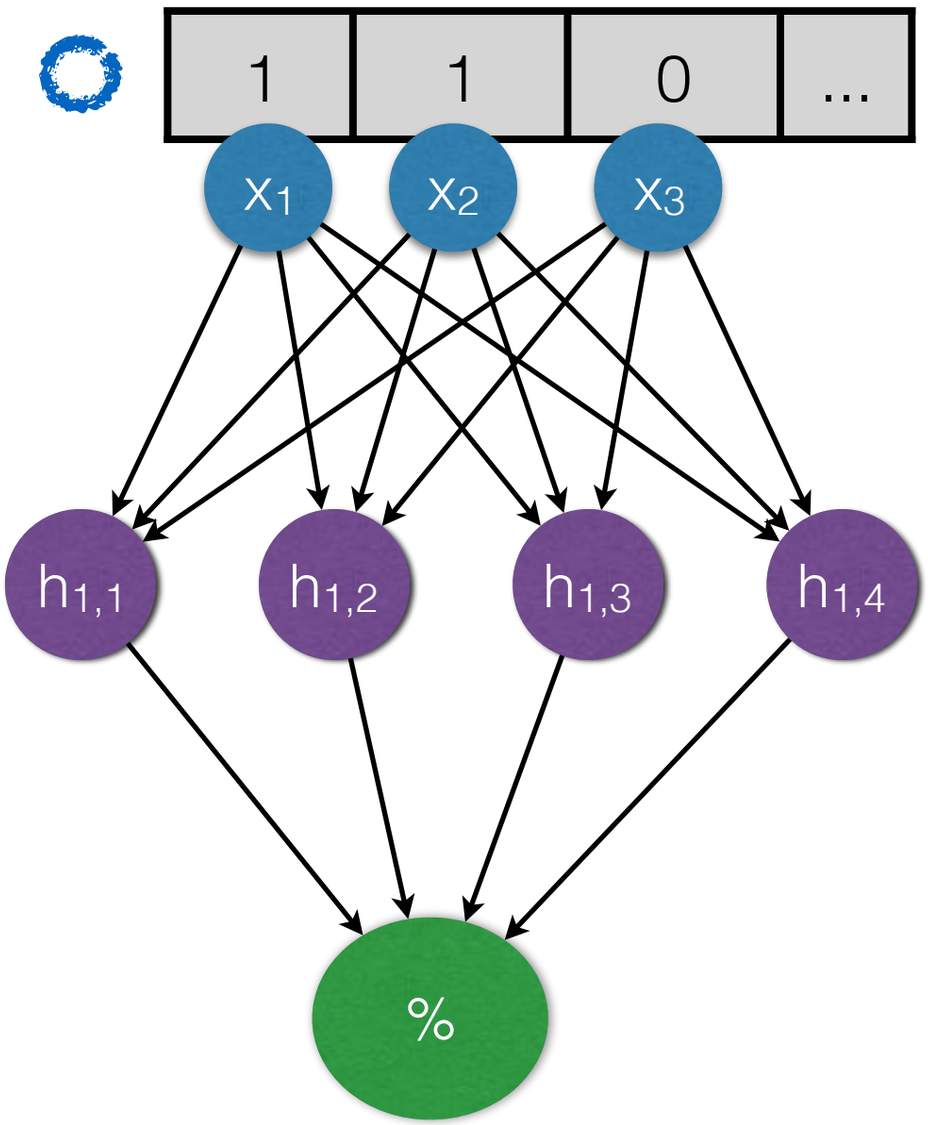


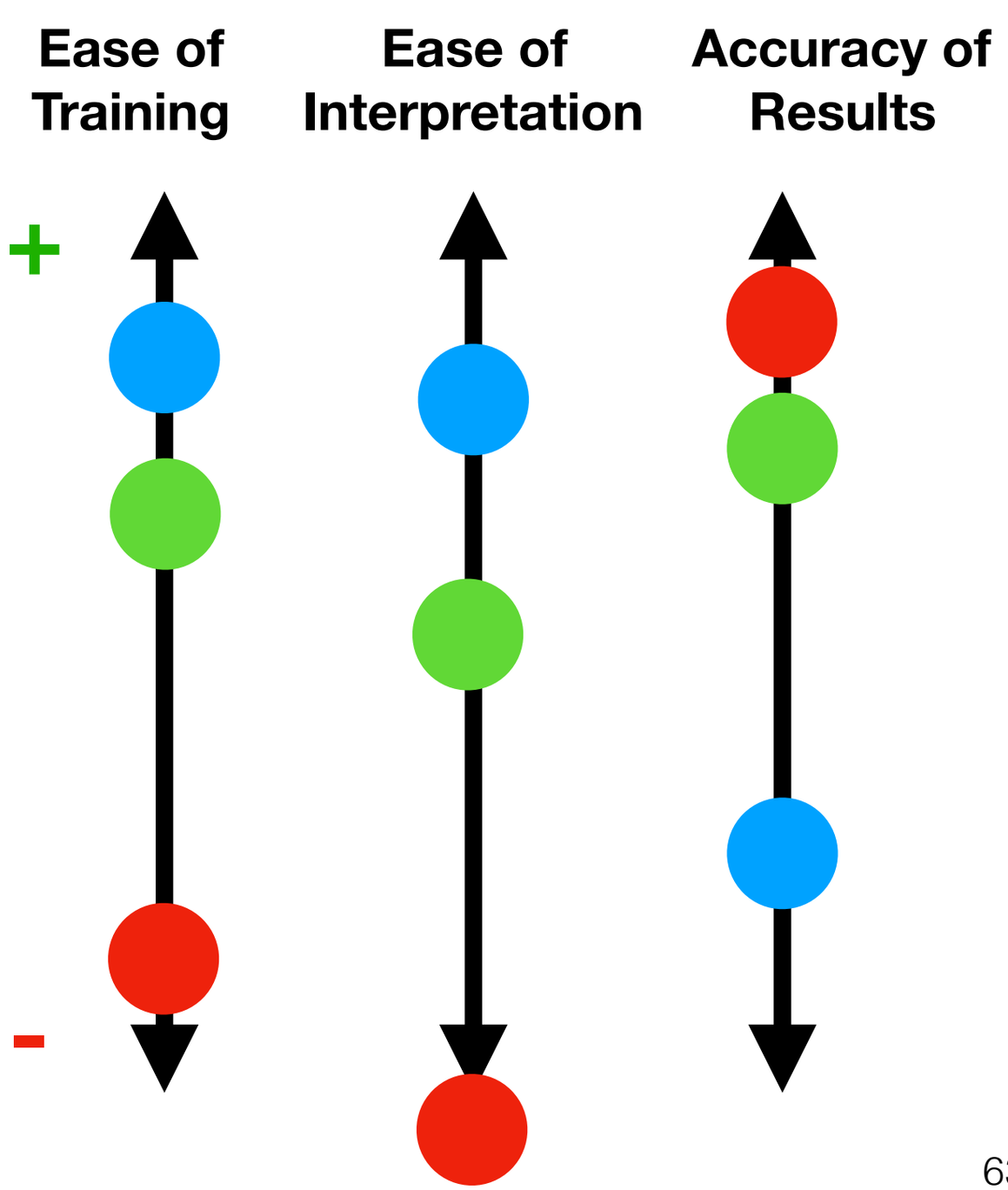
Multi-Layer Perceptron





Multi-Layer Perceptron





Preliminary results

How accurate are we?

Which features are most important?

Top-k Accuracy:

When classifier ranks expressions by likelihood, how often is one of the top k answers correct?

Accuracy

Our top-1 accuracy	63%
Baseline: blame line that crashes	43%
(other baseline: guess randomly)	14%

Accuracy

Our top-1 accuracy	63%
Baseline: blame line that crashes	43%
(other baseline: guess randomly)	14%
Our top-3 accuracy	82%

Which features are most important?

1. Is it where the crash happens?
2. Is the expression in the error slice?
3. Is the type of the *parent* unknown?

Example

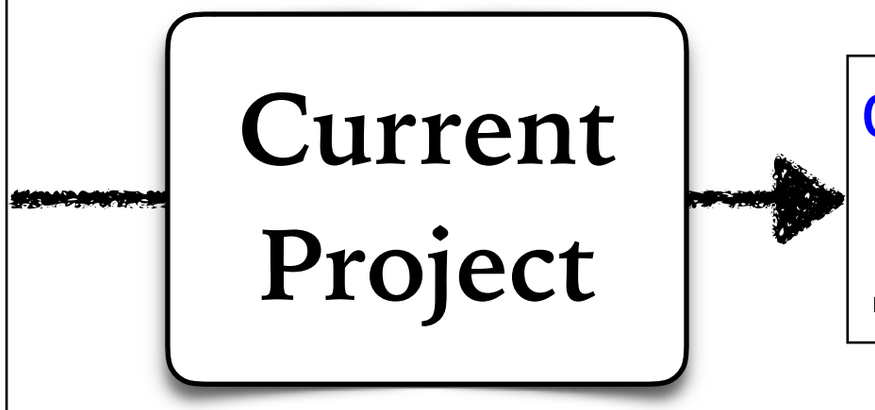
Future directions

Future directions

- Synthesis
- Feedback
- Malformed code

Synthesis

```
def double(x):
    result = x*2

y = double(3)
z = y + 4
```




```
def double(x):
    result = x*2
...
```

Synthesis

```
def double(x):
    result = x*2

y = double(3)
z = y + 4
```

Current Project

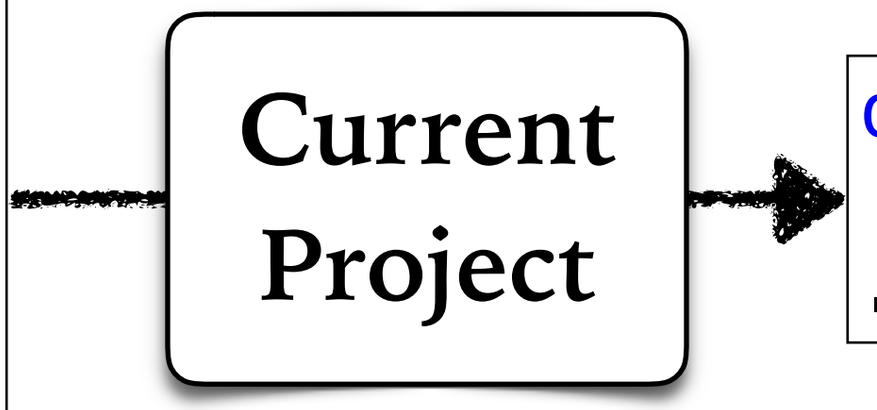
```
def double(x):
    result = x*2
...
```

???

```
def double(x):
    return x*2
...
```

Synthesis

```
def double(x):
    result = x*2

y = double(3)
z = y + 4
```




```
def double(x):
    result = x*2
...
```

```
def double(x):
    return x*2
...
```

Synthesis

```
def double(x):
    result = x*2

y = double(3)
z = y + 4
```

Current Project

```
def double(x):
    result = x*2
...
```

Rip out bug

```
def double(x):
    ???????????
...
```

```
def double(x):
    return x*2
...
```

Synthesis

```
def double(x):
    result = x*2

y = double(3)
z = y + 4
```

Current Project

```
def double(x):
    result = x*2
...
```

Rip out bug

```
def double(x):
    ???????????
...
```

Existing Synthesis Technique

```
def double(x):
    return x*2
...
```

Synthesis

```
def double(x):
    result = x*2

y = double(3)
z = y + 4
```

Current Project

```
def double(x):
    result = x*2
...
```

Rip out bug

```
def double(x):
    ???????????
...
```

hint:
x*2

Existing Synthesis Technique

```
def double(x):
    return x*2
...
```

Synthesis

```
def double(x):
    result = x*2

y = double(3)
z = y + 4
```

Current Project

```
def double(x):
    result = x*2
...
```

Rip out bug

```
def double(x):
    ???????????
...
```

hint:
x*2

Hybrid Synthesis Technique

```
def double(x):
    return x*2
...
```

Propagating feedback

Propagating feedback

Expert feedback is great but doesn't scale

Propagating feedback

Expert feedback is great but doesn't scale

Machine learning can cluster similar errors

Propagating feedback

Expert feedback is great but doesn't scale

Machine learning can cluster similar errors

Expert provides feedback for whole cluster at once

Malformed Code

Current project can't handle syntax errors

Need a way to handle damaged ASTs

```
print(len("bah")
print("Hello, world!")
```

```
SyntaxError: invalid syntax, line 2
```

Timeline

Timeline

- Localizing errors: August (ICSE)

Timeline

- Localizing errors: August (ICSE)
- Synthesizing fixes: March (FSE)

Timeline

- Localizing errors: August (ICSE)
- Synthesizing fixes: March (FSE)
- Propagating feedback: August (ICSE)

Timeline

- Localizing errors: August (ICSE)
- Synthesizing fixes: March (FSE)
- Propagating feedback: August (ICSE)
- Other options
  - ESOP - October
  - ECOOP - January
  - OOPSLA - April