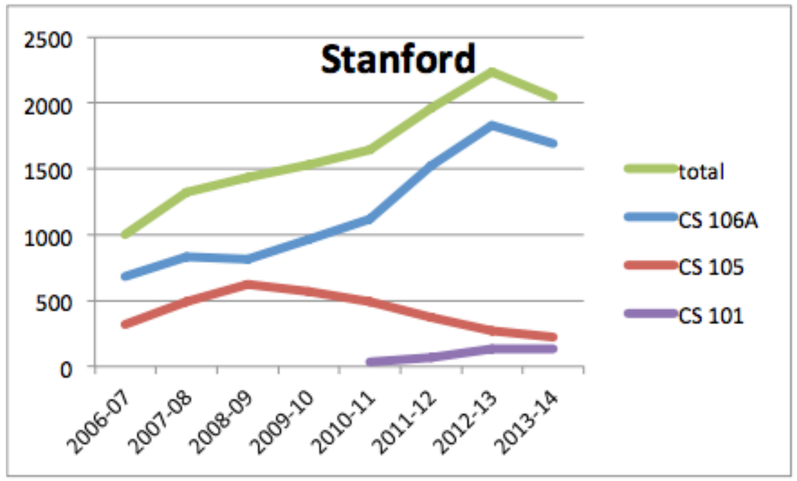
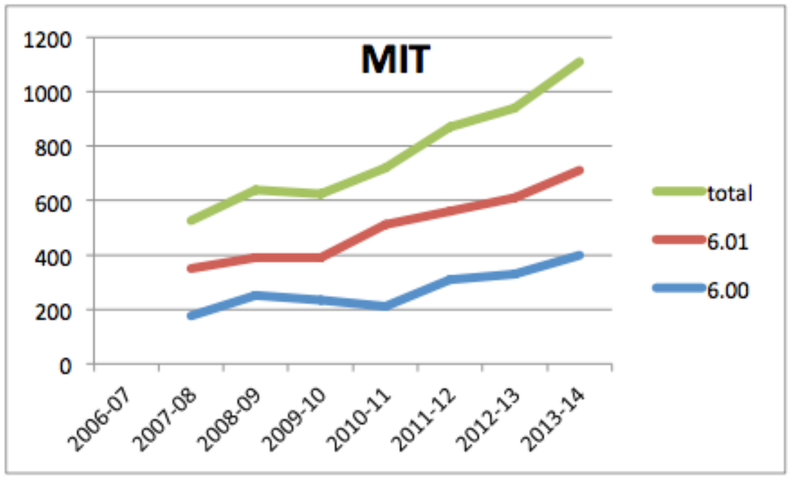
TraceDiff: Debugging Unexpected Code Behavior Using Trace Divergences

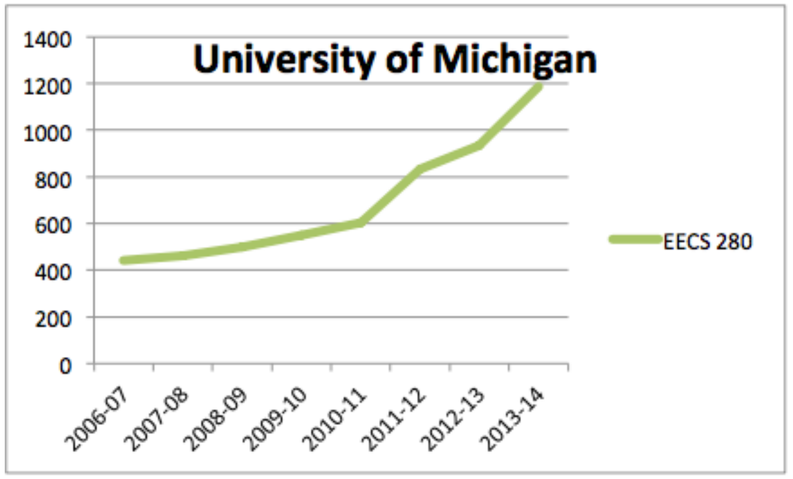
Ryo Suzuki, Gustavo Soares, Andrew Head, Elena Glassman,
Ruan Reis, Melina Mongiovi, Loris D'Antoni, Björn Hartmann

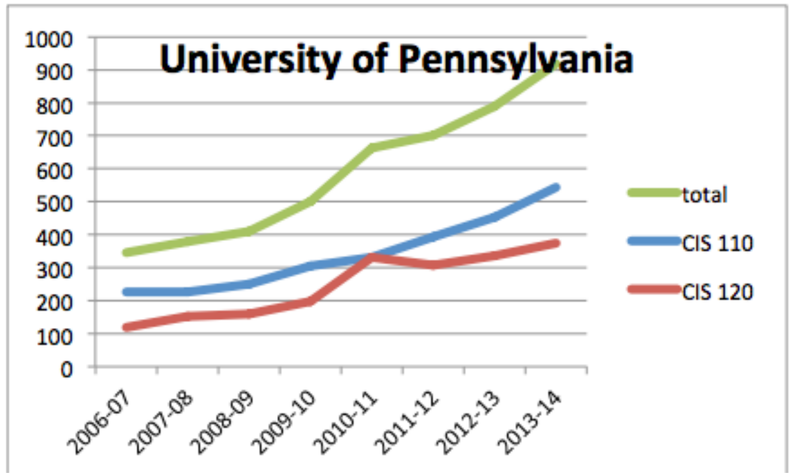
University of Colorado Boulder
Berkeley UNIVERSITY OF CALIFORNIA
Universidade Federal de Campina Grande
WISCONSIN UNIVERSITY OF WISCONSIN–MADISON

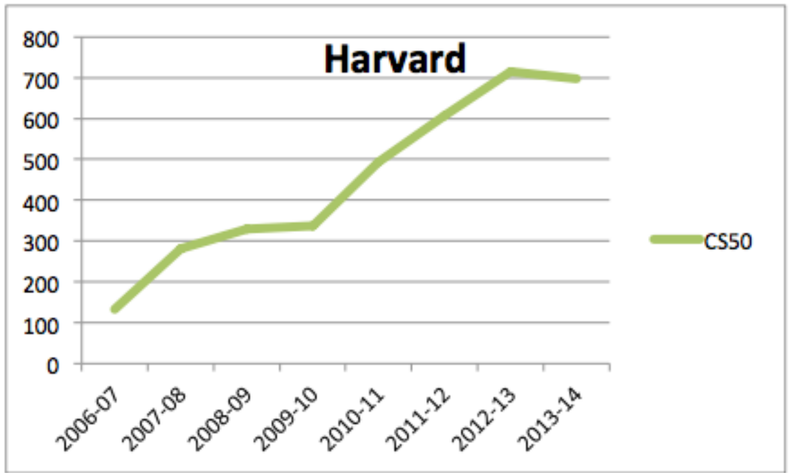
Increasing Demand for Introductory CS Courses

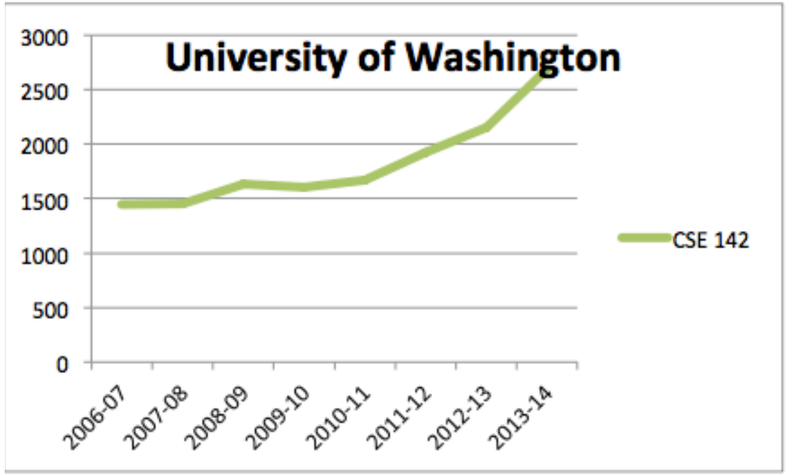






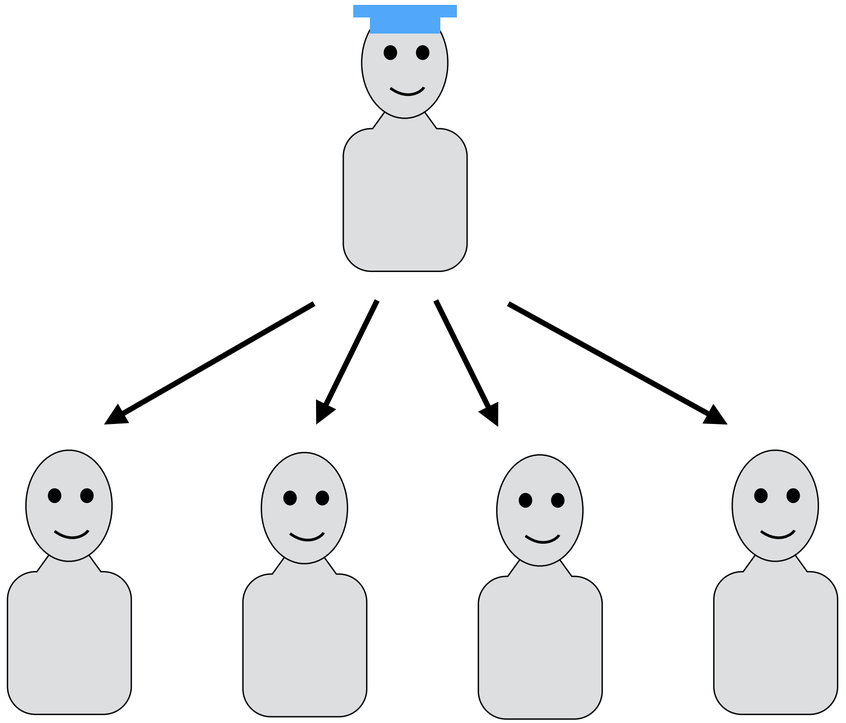




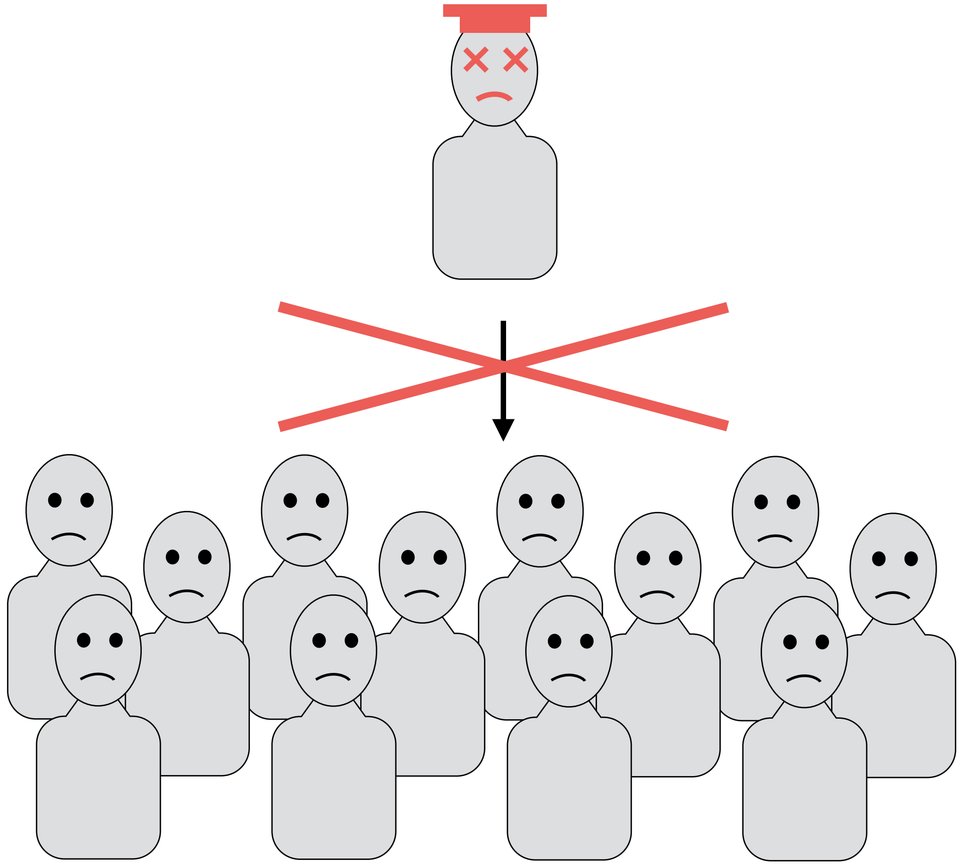


source: https://www.geekwire.com/2014/analysis-examining-computer-science-education-explosion/

Teachers' Personalized Attention **Does Not Scale**



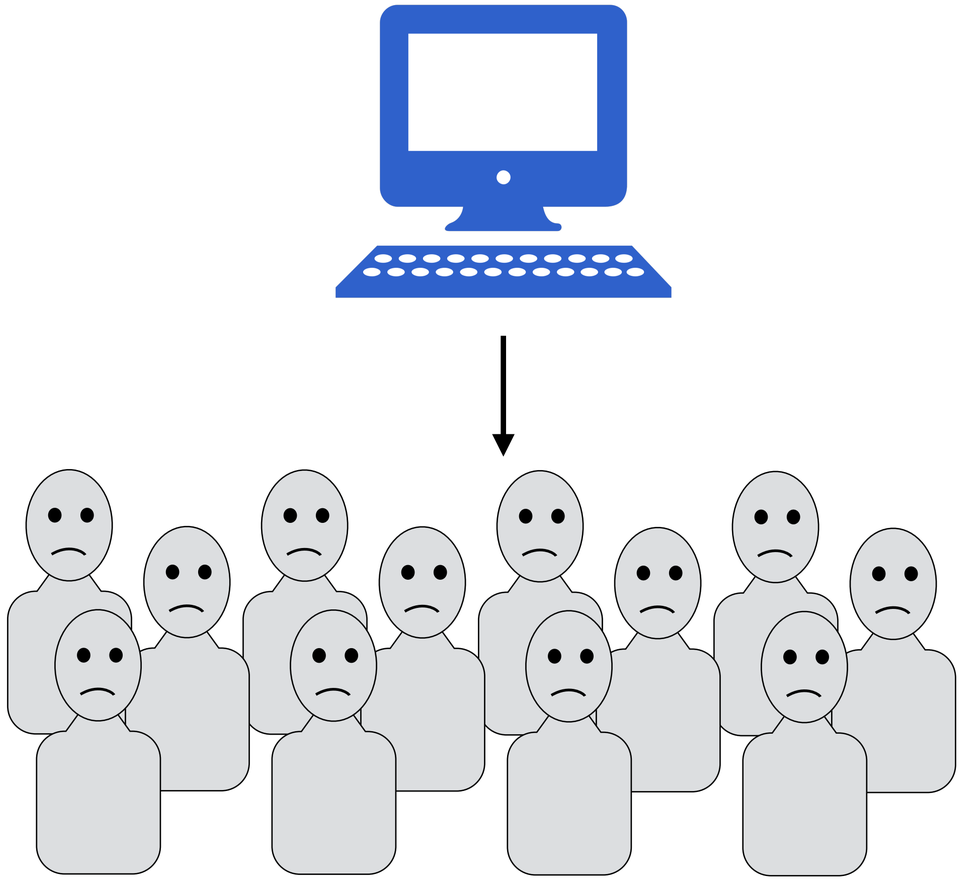
Traditional Classroom



Massive Classroom

(1,000-2,000 students)

Automatic Programming Feedback



Massive Classroom

(1,000-2,000 students)

Code

```
def accumulate(combiner, base, n, term):
    if n == 1:
    if n == 0:
        return base
    else:
     return combiner(term(n),
                     accumulate(…, n-1, …))
```

```
>>> accumulate(add, 0, 5, identity)
15 # 0 + 1 + 2 + 3 + 4 + 5

>>> accumulate(mul, 2, 3, square)
72 # 2 * 1^2 * 2^2 * 3^2
```

Code

```
def accumulate(combiner, base, n, term):
    if n == 1:
    if n == 0:
        return base
    else:
     return combiner(term(n),
                     accumulate(…, n-1, …))
```

Test Case Feedback

```
>>> accumulate(add, 0, 5, identity)
x 14
o 15
```

Code

```
def accumulate(combiner, base, n, term):
    if n == 1:
    if n == 0:
        return base
    else:
     return combiner(
```

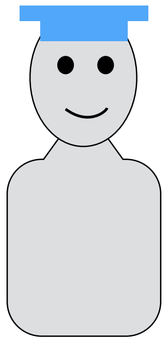
```
e(…, n-1, …))
```

Test Case Feedback

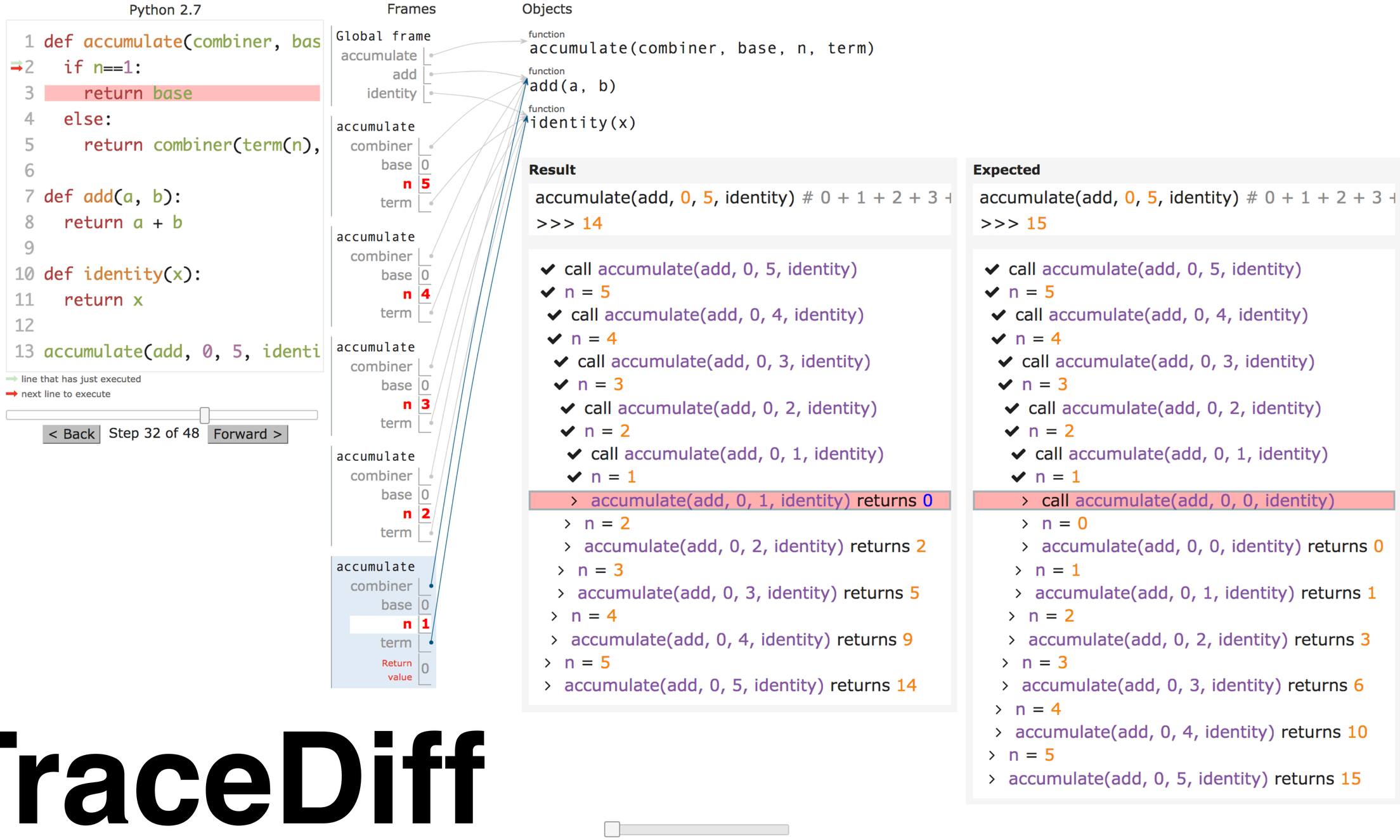
```
>>> accumulate(add, 0, 5, identity)
x 14
o 15
```

Code

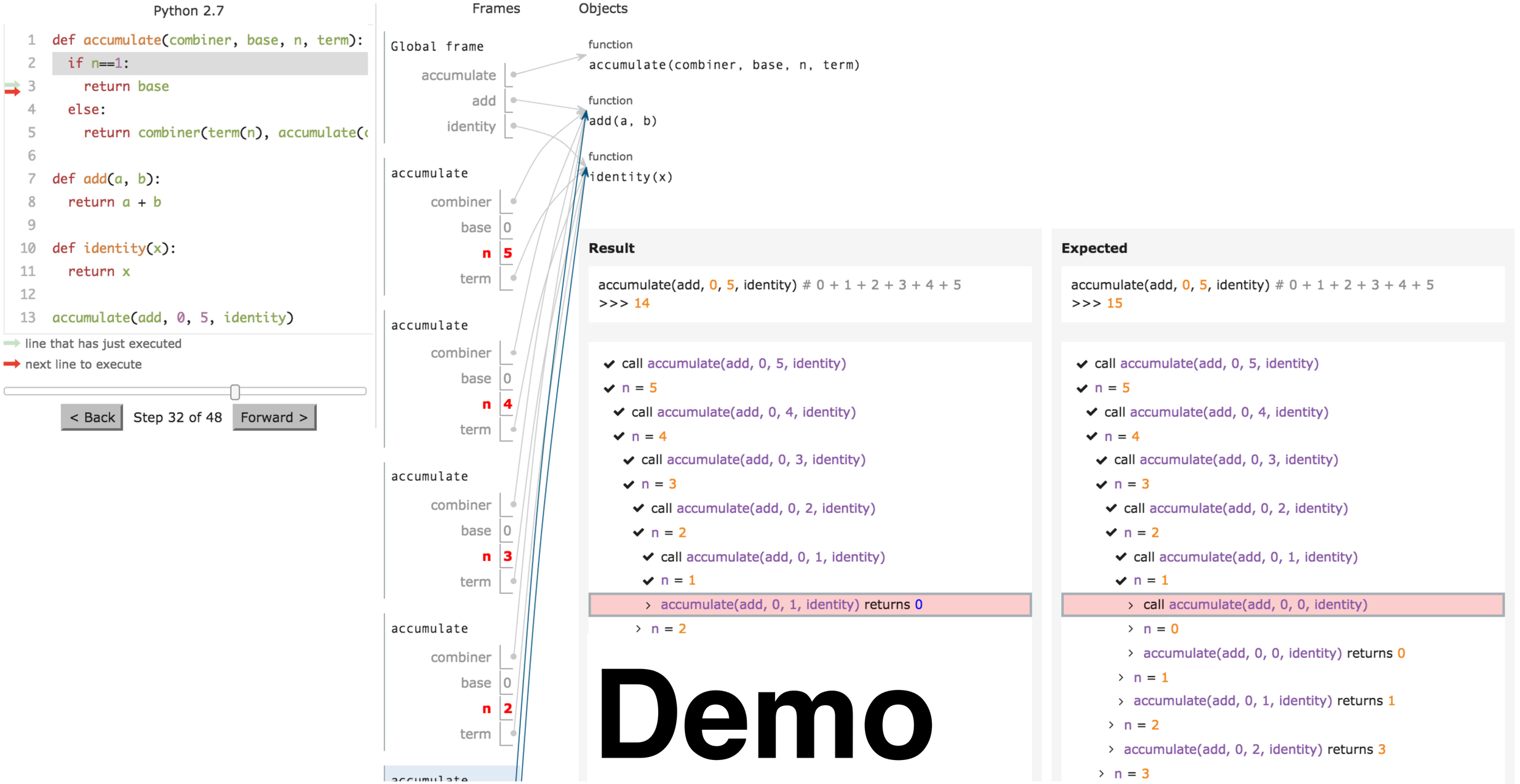
```
def accumulate(combiner, base, n, term):
    if n == 1:
    if n == 0:
        return base
    else:
     return combiner(term(n),
                     accumulate(…, n-1, …))
```

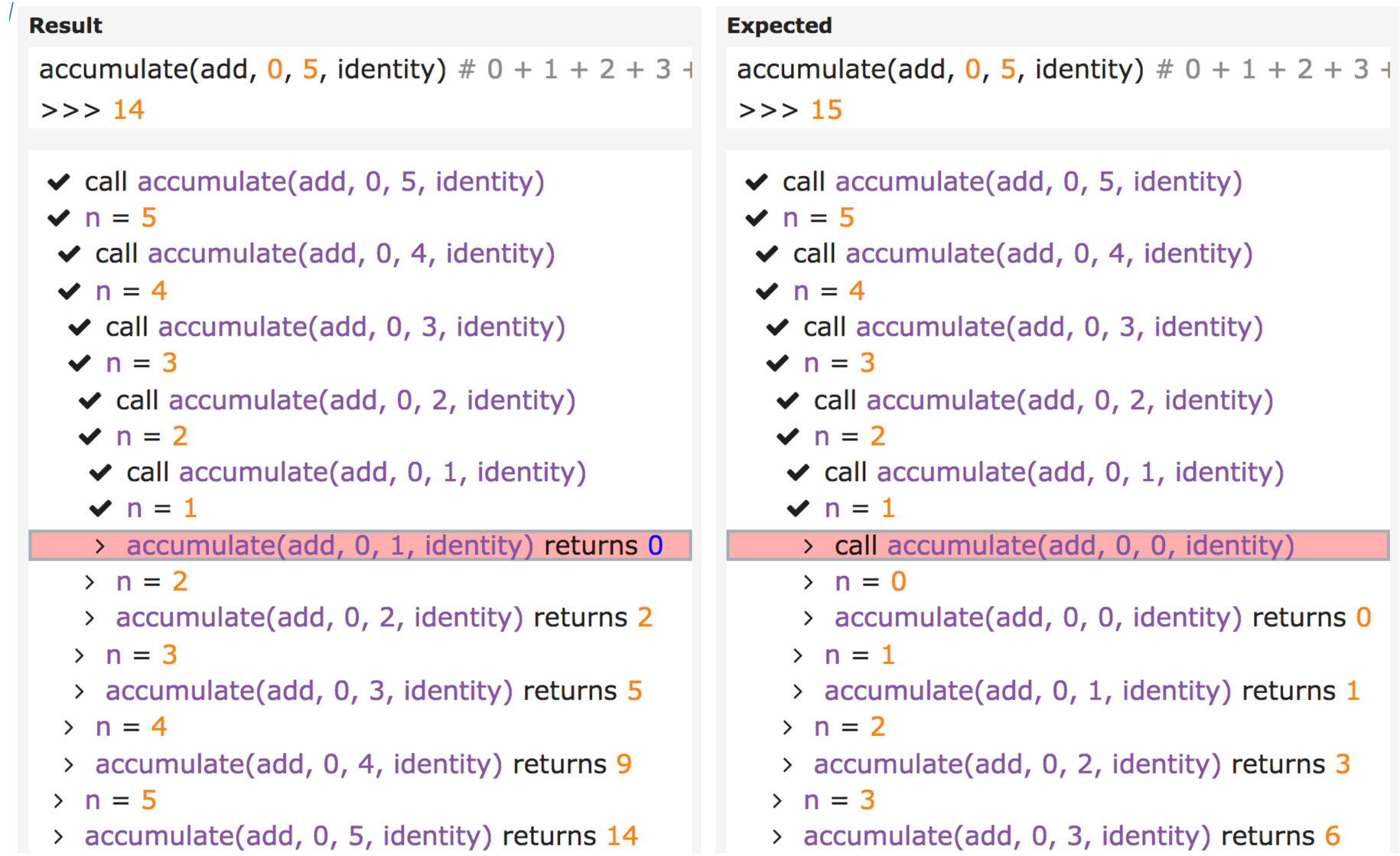


```
accumulate(add, 0, 5, identity)
x 14 # 0 +    + 2 + 3 + 4 + 5
o 15 # 0 + 1 + 2 + 3 + 4 + 5
```

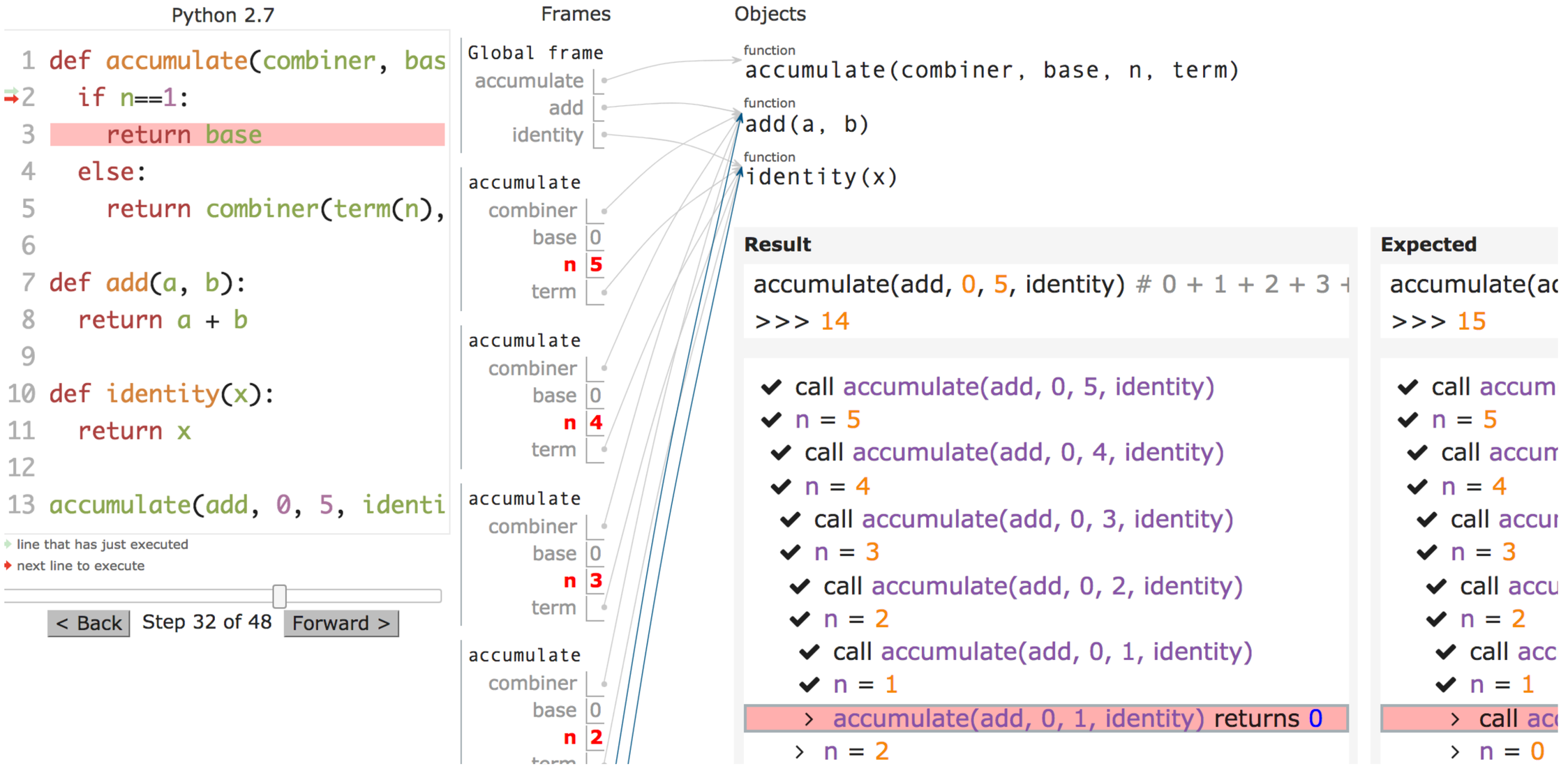

Python 2.7
1 def accumulate(combiner, bas
2 if n==1:
3 return base
4 else:
5 return combiner(term(n),
6
7 def add(a, b):
8 return a + b
9
10 def identity(x):
11 return x
12
13 accumulate(add, 0, 5, identi
line that has just executed
next line to execute
< Back
Step 32 of 48
Forward >
Frames
Global frame
accumulate
add
identity
accumulate
combiner
base 0
n 5
term
accumulate
combiner
base 0
n 4
term
accumulate
combiner
base 0
n 3
term
accumulate
combiner
base 0
n 2
term
accumulate
combiner
base 0
n 1
term
Return value 0
Objects
function
accumulate(combiner, base, n, term)
function
add(a, b)
function
identity(x)
Result
accumulate(add, 0, 5, identity) # 0 + 1 + 2 + 3 +
>>> 14
call accumulate(add, 0, 5, identity)
n = 5
call accumulate(add, 0, 4, identity)
n = 4
call accumulate(add, 0, 3, identity)
n = 3
call accumulate(add, 0, 2, identity)
n = 2
call accumulate(add, 0, 1, identity)
n = 1
accumulate(add, 0, 1, identity) returns 0
n = 2
accumulate(add, 0, 2, identity) returns 2
n = 3
accumulate(add, 0, 3, identity) returns 5
n = 4
accumulate(add, 0, 4, identity) returns 9
n = 5
accumulate(add, 0, 5, identity) returns 14
Expected
accumulate(add, 0, 5, identity) # 0 + 1 + 2 + 3 +
>>> 15
call accumulate(add, 0, 5, identity)
n = 5
call accumulate(add, 0, 4, identity)
n = 4
call accumulate(add, 0, 3, identity)
n = 3
call accumulate(add, 0, 2, identity)
n = 2
call accumulate(add, 0, 1, identity)
n = 1
call accumulate(add, 0, 0, identity)
n = 0
accumulate(add, 0, 0, identity) returns 0
n = 1
accumulate(add, 0, 1, identity) returns 1
n = 2
accumulate(add, 0, 2, identity) returns 3
n = 3
accumulate(add, 0, 3, identity) returns 6
n = 4
accumulate(add, 0, 4, identity) returns 10
n = 5
accumulate(add, 0, 5, identity) returns 15

TraceDiff

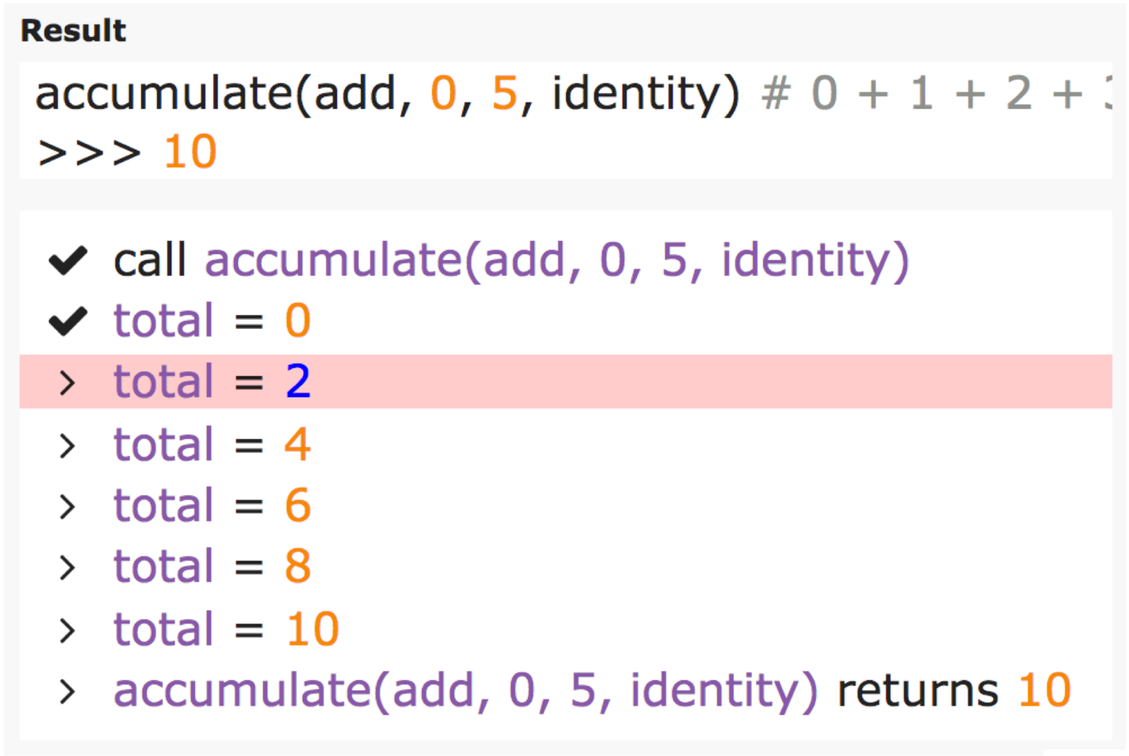

Python 2.7
1 def accumulate(combiner, base, n, term):
2 if n==1:
3 return base
4 else:
5 return combiner(term(n), accumulate(
6
7 def add(a, b):
8 return a + b
9
10 def identity(x):
11 return x
12
13 accumulate(add, 0, 5, identity)
line that has just executed
next line to execute
< Back
Step 32 of 48
Forward >
Frames
Objects
Global frame
accumulate
add
identity
function
accumulate(combiner, base, n, term)
function
add(a, b)
function
identity(x)
accumulate
combiner
base 0
n 5
term
accumulate
combiner
base 0
n 4
term
accumulate
combiner
base 0
n 3
term
accumulate
combiner
base 0
n 2
term
Result
accumulate(add, 0, 5, identity) # 0 + 1 + 2 + 3 + 4 + 5
>>> 14
call accumulate(add, 0, 5, identity)
n = 5
call accumulate(add, 0, 4, identity)
n = 4
call accumulate(add, 0, 3, identity)
n = 3
call accumulate(add, 0, 2, identity)
n = 2
call accumulate(add, 0, 1, identity)
n = 1
accumulate(add, 0, 1, identity) returns 0
n = 2
Expected
accumulate(add, 0, 5, identity) # 0 + 1 + 2 + 3 + 4 + 5
>>> 15
call accumulate(add, 0, 5, identity)
n = 5
call accumulate(add, 0, 4, identity)
n = 4
call accumulate(add, 0, 3, identity)
n = 3
call accumulate(add, 0, 2, identity)
n = 2
call accumulate(add, 0, 1, identity)
n = 1
call accumulate(add, 0, 0, identity)
n = 0
accumulate(add, 0, 0, identity) returns 0
n = 1
accumulate(add, 0, 1, identity) returns 1
n = 2
accumulate(add, 0, 2, identity) returns 3
n = 3
Demo
https://ryosuzuki.github.io/trace-diff/?type=accumulate&id=10

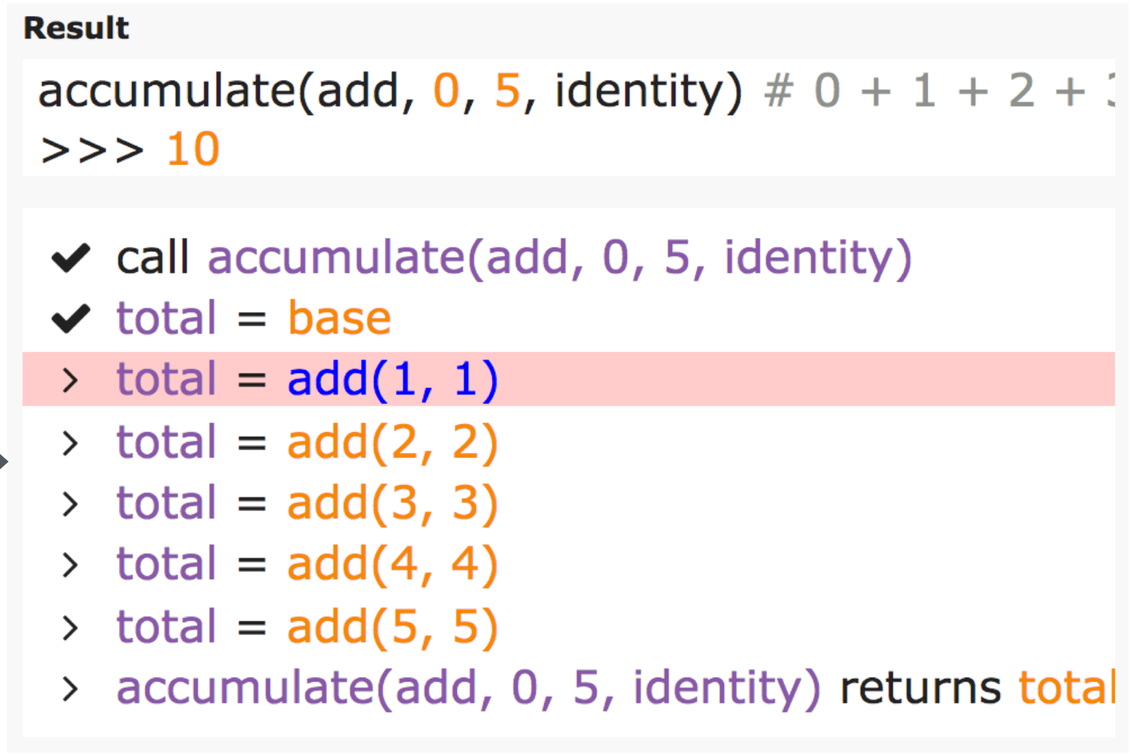


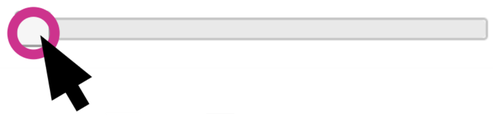
1. Highlight Differences

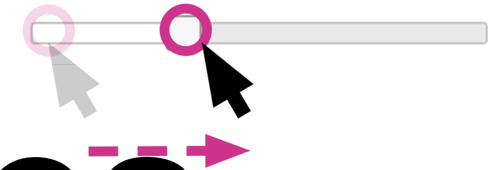


2. Interactively Explore







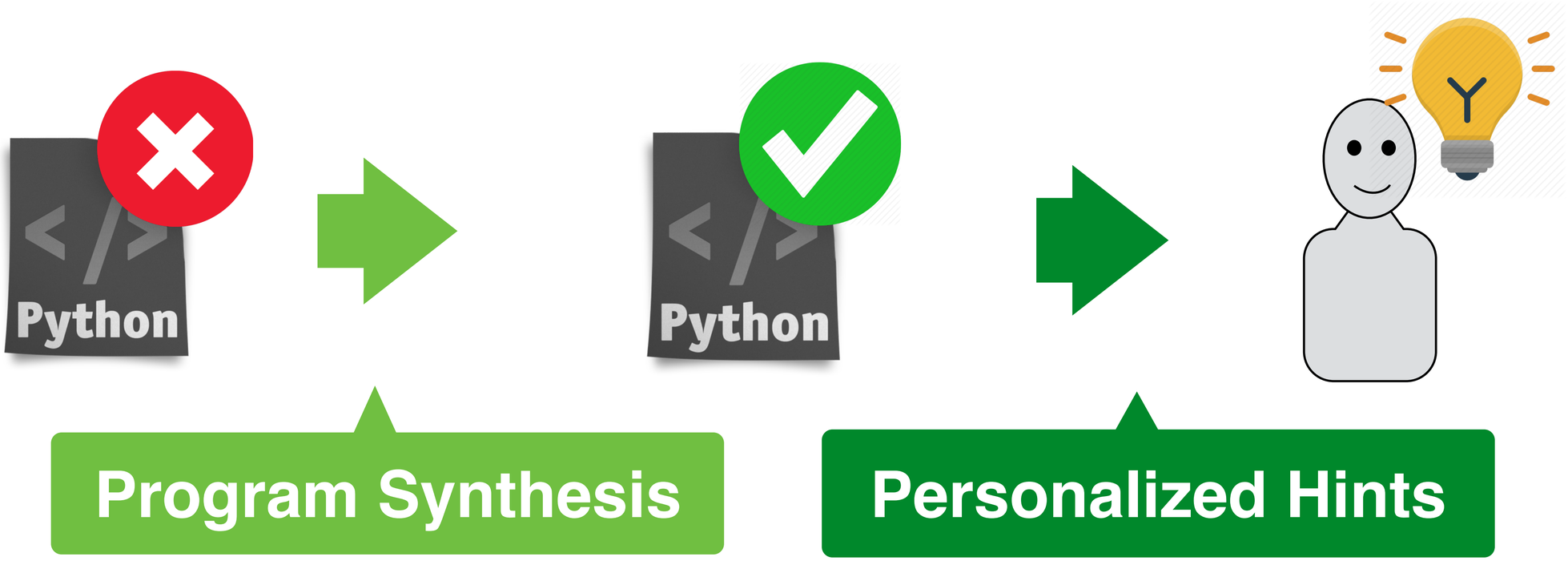


3. Abstract Values Into Expressions

Motivation

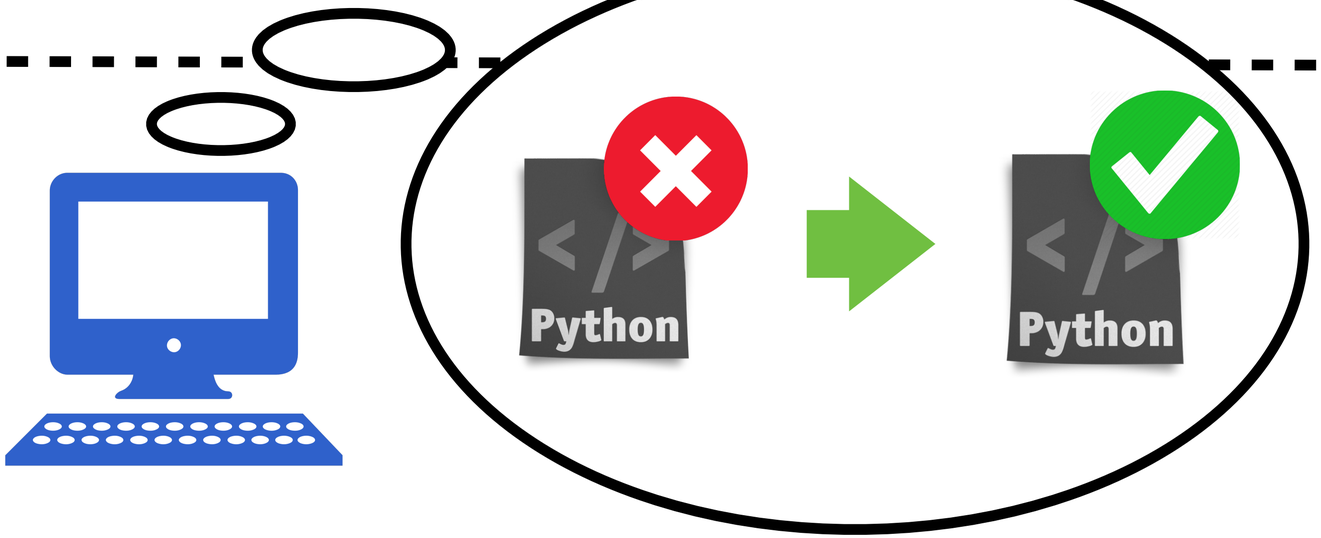
Program Synthesis Feedback

(e.g. Singh [PLDI'13], D'Antoni [TOCHI'15], Rolim [ICSE'17])



Code

```
def accumulate(combiner, base, n, term):
    if n == 1:
    if n == 0:
        return base
    else:
     return combiner(term(n),
                     accumulate(…, n-1, …))
```



Code

```
def accumulate(combiner, base, n, term):
    if n == 1:
    if n == 0:
        return base
    else:
     return combiner(term(n),
                     accumulate(…, n-1, …))
```

Line 2 needs to be changed

Code

```
def accumulate(combiner, base, n, term):
    if n == 1:
    if n == 0:
        return base
    else:
     return combiner(term(n),
                     accumulate(…, n-1, …))
```

Line 2 needs to be changed

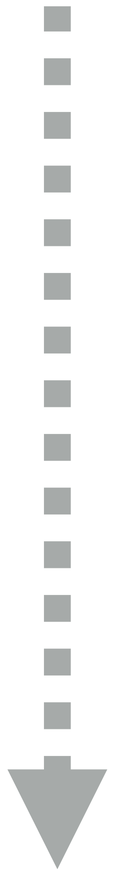
In line 2, change n = 1 to n = 0

Program Synthesis Feedback

Line 2 needs to be changed

In line 2, check "n"

In line 2, change n = 1 to n = 0



Program Synthesis Feedback

Pointing

Line 2 needs to be changed

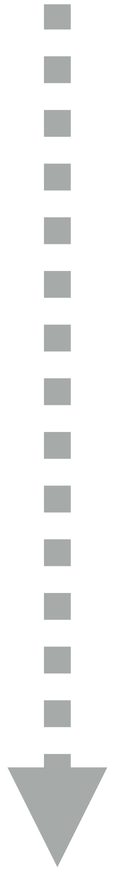
In line 2, check "n"

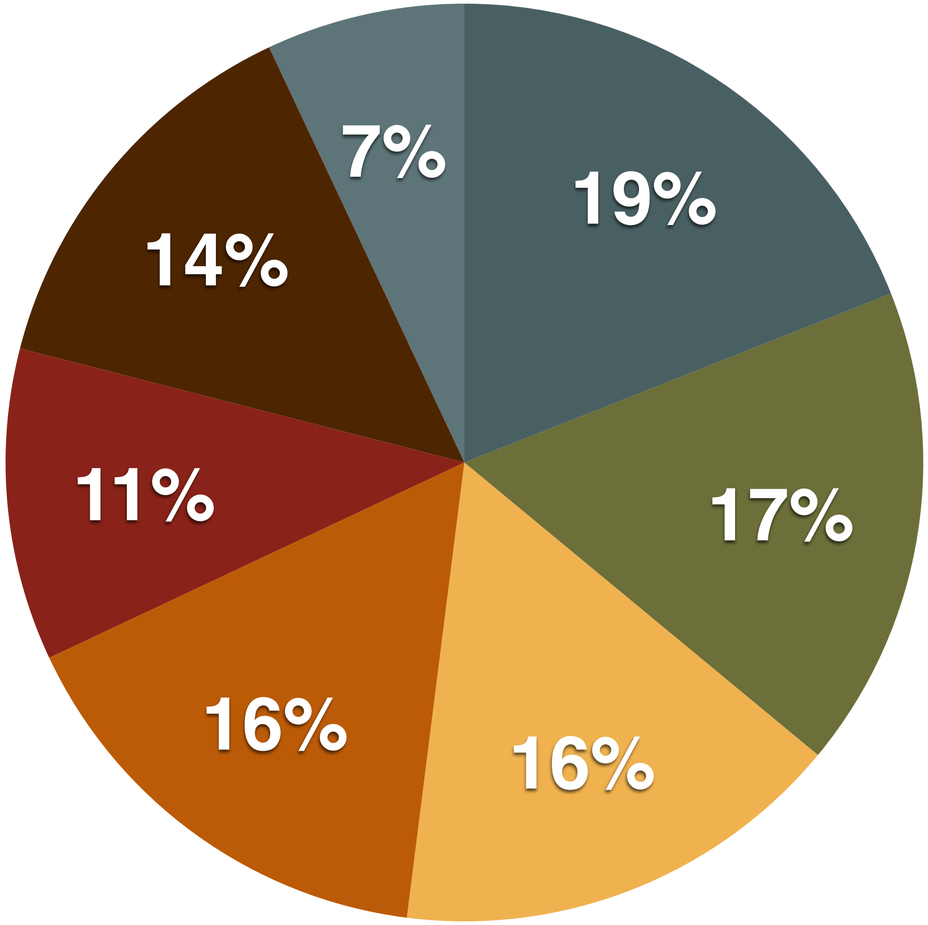
High-level Hints

?

Bottom-out

In line 2, change n = 1 to n = 0

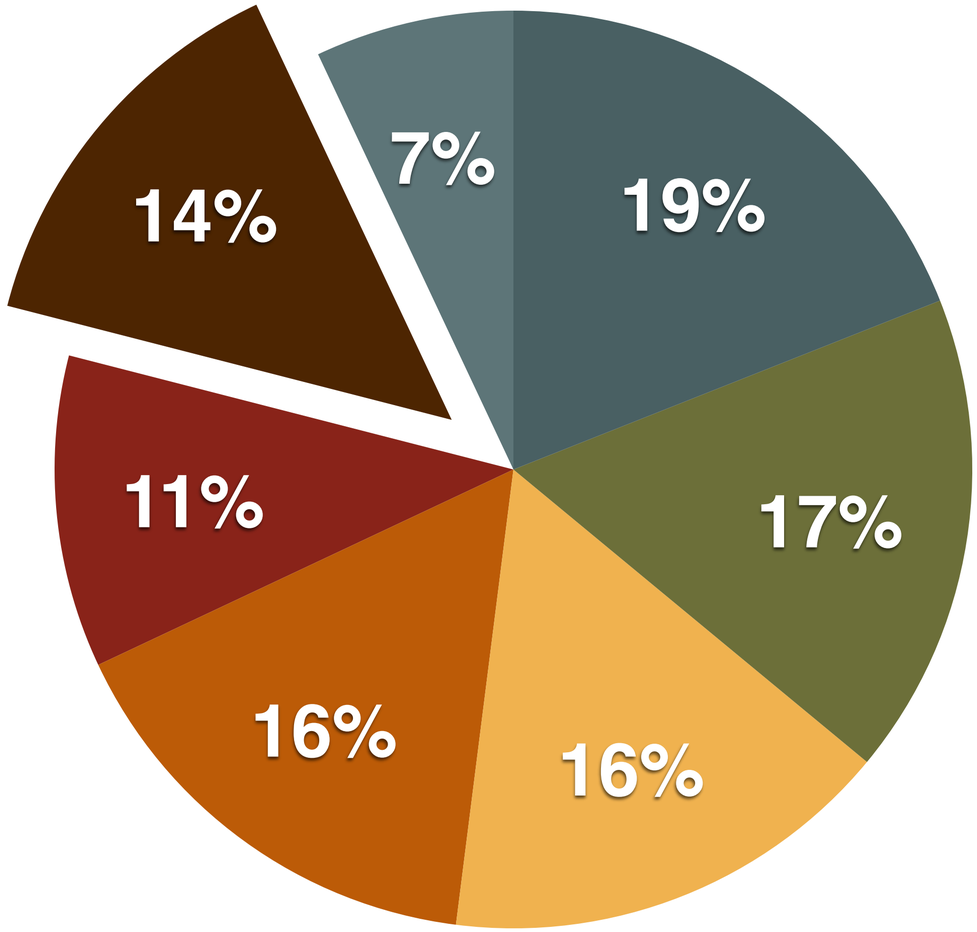




- Diagnose the cause of error:19%
- Suggest to run code in PythonTutor: 17%
- Explain incorrect behavior: 16%
- Remind relevant resources: 16%
- Provide example usage: 11%

- Point out location: 14%
- Suggest concrete fix: 7%

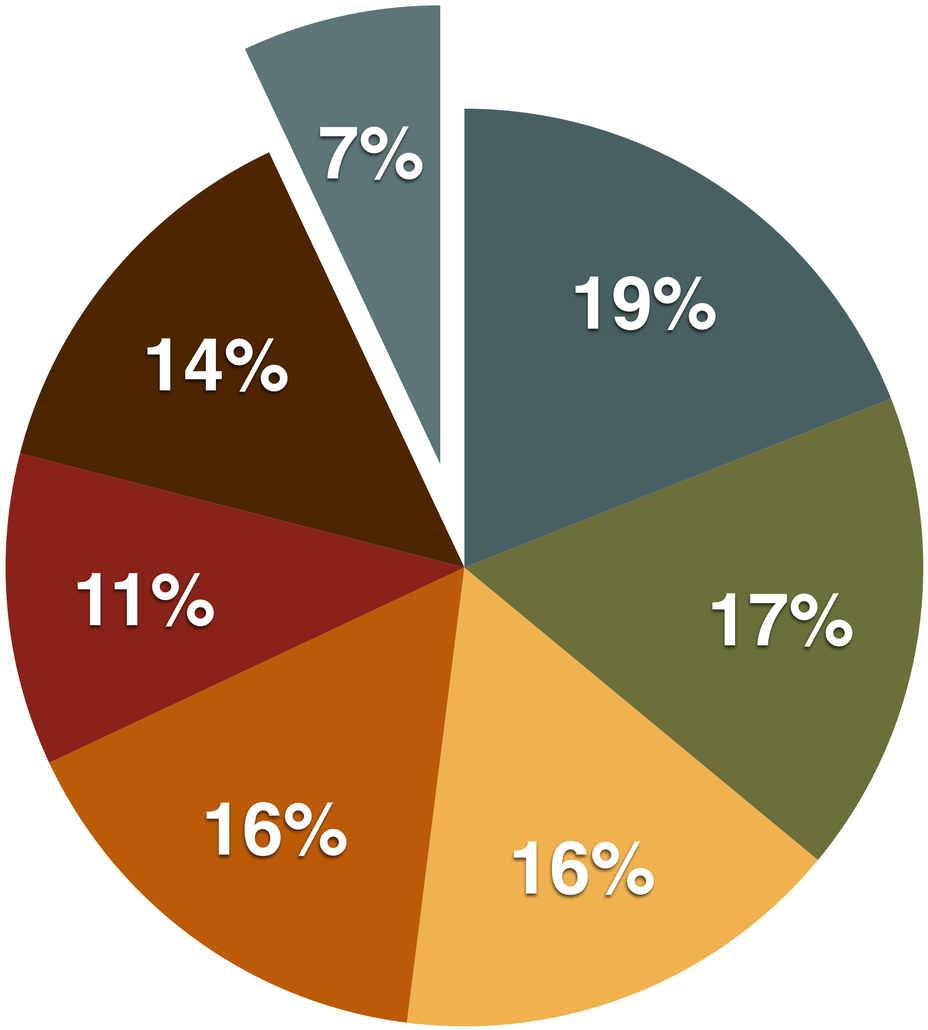
132 posts on piazza



- Diagnose the cause of error:19%
- Suggest to run code in PythonTutor: 17%
- Explain incorrect behavior: 16%
- Remind relevant resources: 16%
- Provide example usage: 11%

- **Point out location: 14%**
- Suggest concrete fix: 7%

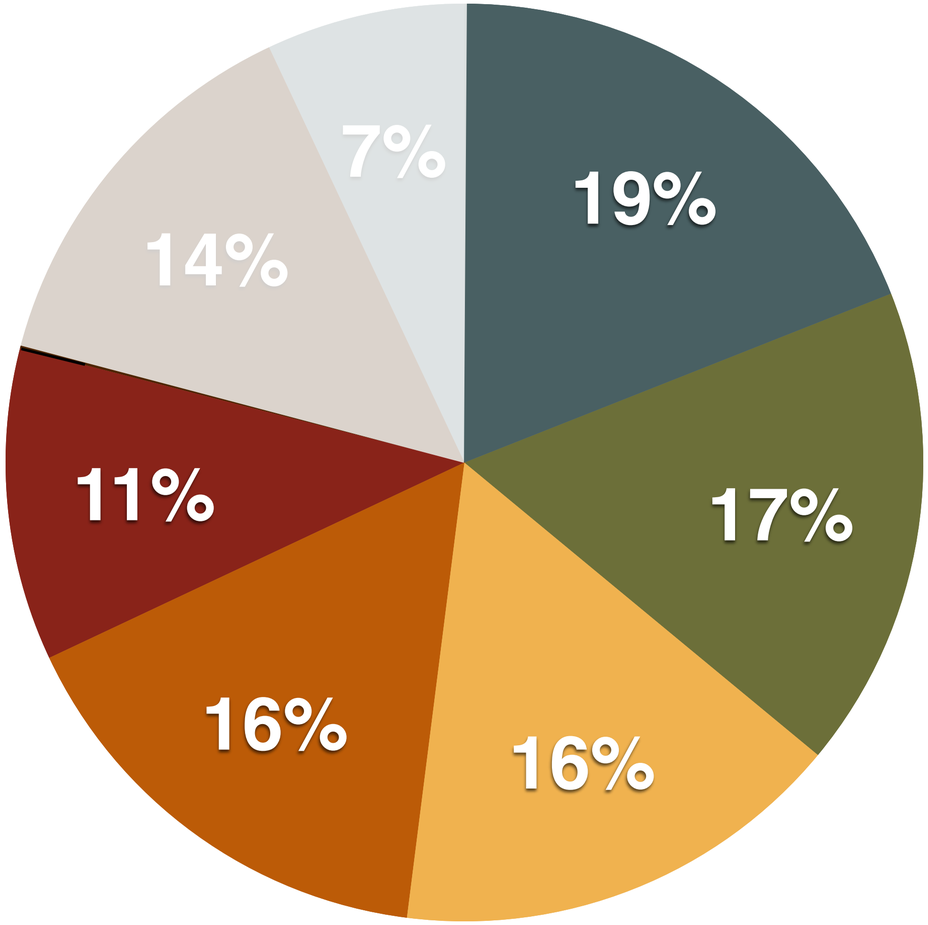
132 posts on piazza



- Diagnose the cause of error:19%
- Suggest to run code in PythonTutor: 17%
- Explain incorrect behavior: 16%
- Remind relevant resources: 16%
- Provide example usage: 11%

- Point out location: 14%
- **Suggest concrete fix: 7%**

132 posts on piazza



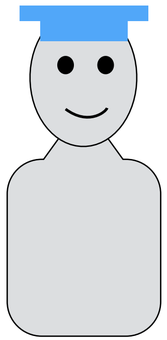
- **Diagnose the cause of error:19%**
- **Suggest to run in PythonTutor: 17%**
- **Explain incorrect behavior: 16%**
- **Remind relevant resources: 16%**
- **Provide example usage: 11%**

- Point out location: 14%
- Suggest concrete fix: 7%

132 posts on piazza

Code

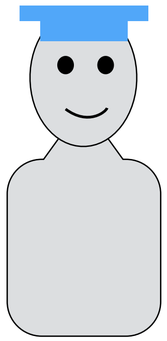
```
def accumulate(combiner, base, n, term):
    if n == 1:
    if n == 0:
        return base
    else:
     return combiner(term(n),
                     accumulate(…, n-1, …))
```



Check the base condition
in the recursive function

Code

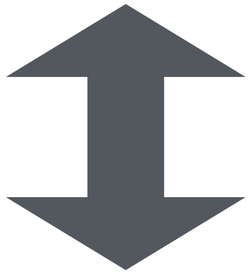
```
def accumulate(combiner, base, n, term):
    if n == 1:
    if n == 0:
        return base
    else:
     return combiner(term(n),
                     accumulate(…, n-1, …))
```

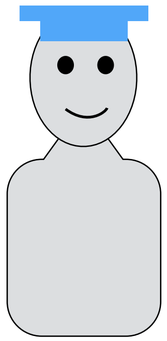


```
accumulate(add, 0, 5, identity)
x 14 # 0 +    + 2 + 3 + 4 + 5
o 15 # 0 + 1 + 2 + 3 + 4 + 5
```

Line 2 needs to be changed

In line 2, change n = 1 to n = 0



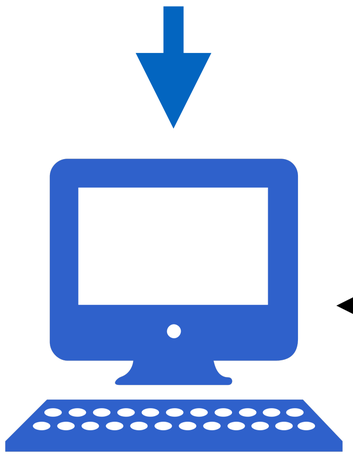


```
accumulate(add, 0, 5, identity)
x 14 # 0 + 0 + 2 + 3 + 4 + 5
o 15 # 0 + 1 + 2 + 3 + 4 + 5
```

Code

```
def accumulate(combiner, base, n, term):
    if n == 1:
    if n == 0:
        return base
    else:
     return combiner(term(n),
                     accumulate(…, n-1, …))
```

Our Goal



```
accumulate(add, 0, 5, identity)
x 14 # 0 +  + 2 + 3 + 4 + 5
o 15 # 0 + 1 + 2 + 3 + 4 + 5
```

Implementation

Incorrect submission

Closest correct program

1 Synthesis

2 Execute & Record Trace

3 Filter & Highlight Trace Difference

Result

```
accumulate(add, 0, 5, identity) # 0 +
>>> 10

✔ call accumulate(add, 0, 5, identity)
✔ total = 0
› total = 2
```

Expected

```
accumulate(add, 0, 5, identity) # 0 +
>>> 15

✔ call accumulate(add, 0, 5, identity)
✔ total = 0
› total = 1
```

Incorrect submission

1 Synthesis

2 Execute & Record Trace

3 Filter & Highlight Trace Difference

Result

```
accumulate(add, 0, 5, identity) # 0 +
>>> 10
```

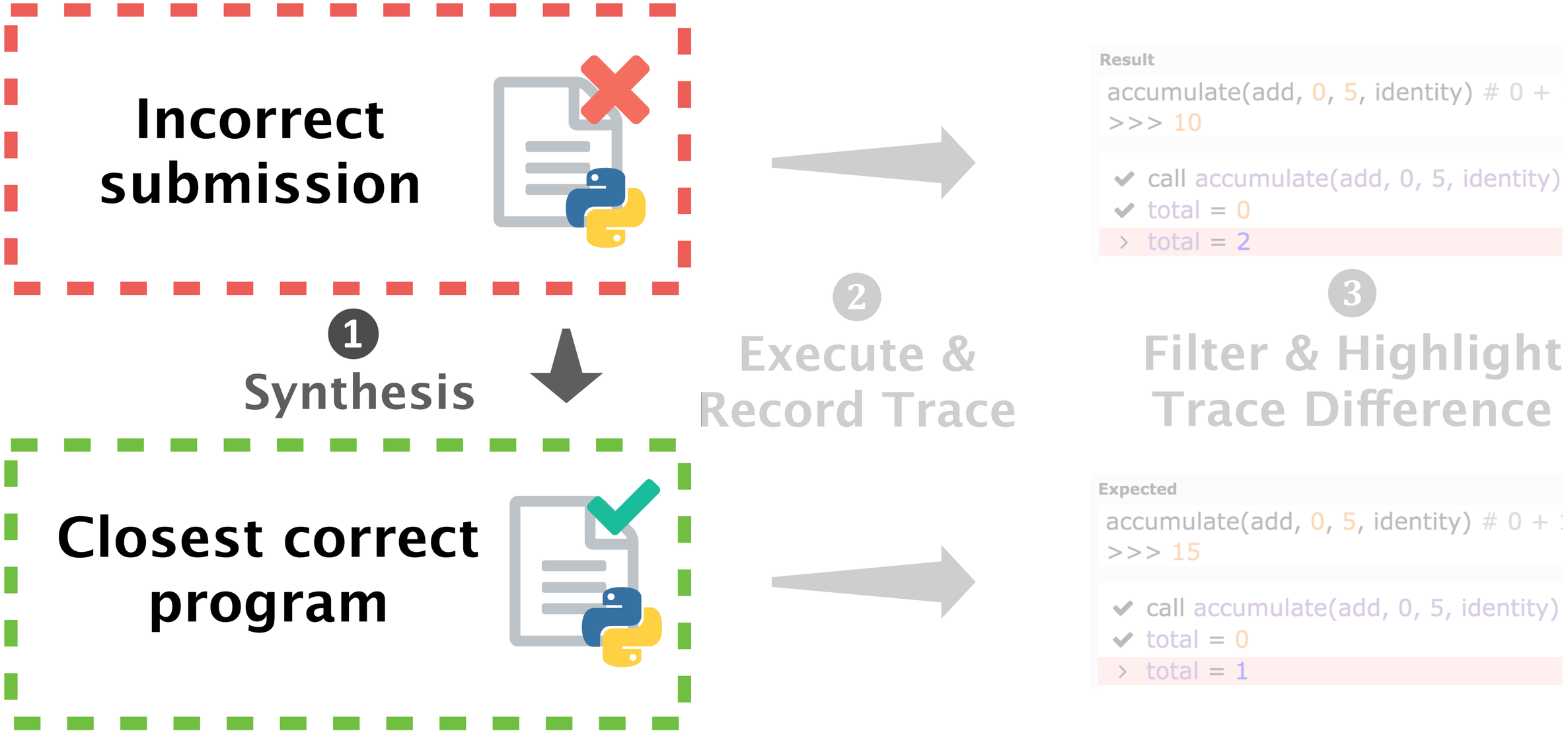
✔ call accumulate(add, 0, 5, identity)
✔ total = 0
› total = 2

Closest correct program

Expected

```
accumulate(add, 0, 5, identity) # 0 +
>>> 15
```

✔ call accumulate(add, 0, 5, identity)
✔ total = 0
› total = 1


Incorrect submission
1
Synthesis
Closest correct program
2
Execute & Record Trace
Result
accumulate(add, 0, 5, identity) # 0 +
>>> 10
call accumulate(add, 0, 5, identity)
total = 0
total = 2
3
Filter & Highlight Trace Difference
Expected
accumulate(add, 0, 5, identity) # 0 +
>>> 15
call accumulate(add, 0, 5, identity)
total = 0
total = 1

Learning Code Transformation (e.g., Refazer [ICSE'17])

Example from student submissions (e.g. Student 1)

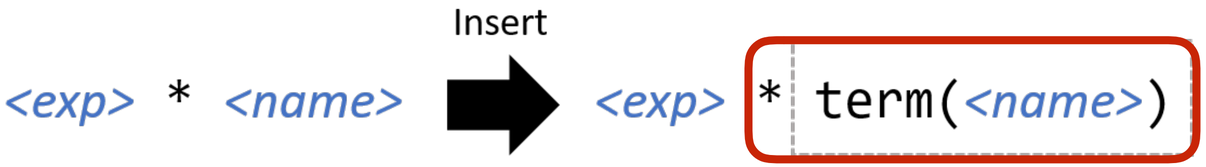
```
 def product(n, term):
   total, k = 1, 1
   while k<=n:
-    total = total*k
+    total = total*term(k)
     k = k+1
   return total
```

Learning Code Transformation (e.g., Refazer [ICSE'17])

Example from student submissions (e.g. Student 1)

```
 def product(n, term):
   total, k = 1, 1
   while k<=n:
-    total = total*k
+    total = total*term(k)
     k = k+1
   return total
```

Learn code transformation from examples

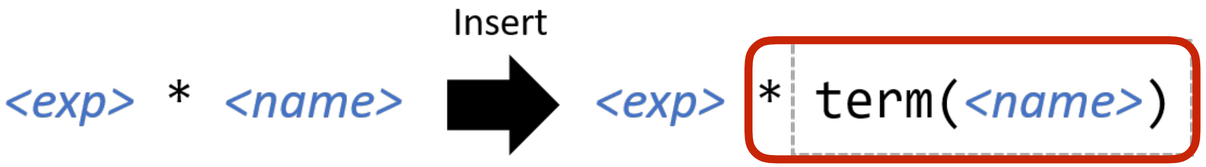


Learning Code Transformation (e.g., Refazer [ICSE'17])

Example from student submissions (e.g. Student 1)

```
 def product(n, term):
   total, k = 1, 1
   while k<=n:
-    total = total*k
+    total = total*term(k)
     k = k+1
   return total
```

Learn code transformation from examples



Apply code transformation (e.g., Student 2)

```
 def product(n, term):
   if (n==1):
     return 1
-  return product(n-1, term)*n
+  return product(n-1, term)*term(n)
```

Incorrect submission

1 Synthesis

Closest correct program

2 Execute & Record Trace

Result

```
accumulate(add, 0, 5, identity) # 0 +
>>> 10
```

call accumulate(add, 0, 5, identity)
total = 0
› total = 2

3 Filter & Highlight Trace Difference

Expected

```
accumulate(add, 0, 5, identity) # 0 +
>>> 15
```

call accumulate(add, 0, 5, identity)
total = 0
› total = 1

```
def accumulate(n):
    total = 0
    i = 1
    while i <= n:
        k = square(i)
        total = total + k
        i = i + 1
```

n: 3

```
def accumulate(n):
    total = 10
    i = 1
    while i <= n:
        k = square(i)
        total = total + i
        i = i + 1
```

n: 3

```
def accumulate(n):
    total = 0
    i = 1
    while i <= n:
        k = square(i)
        total = total + k
        i = i + 1
```

n: 3

total: 0

```
def accumulate(n):
    total = 10
    i = 1
    while i <= n:
        k = square(i)
        total = total + i
        i = i + 1
```

n: 3

total: 3

```
def accumulate(n):
    total = 0
    i = 1
    while i <= n:
        k = square(i)
        total = total + k
        i = i + 1
```

```
n: 3

total: 0

i: 1
```

```
def accumulate(n):
    total = 10
    i = 1
    while i <= n:
        k = square(i)
        total = total + i
        i = i + 1
```

```
n: 3

total: 10

i: 1
```

```
def accumulate(n):
    total = 0
    i = 1
    while i <= n:
      k = square(i)
      total = total + k
      i = i + 1
```

```
n: 3

total: 0

i: 1

k: 1
```

```
def accumulate(n):
    total = 10
    i = 1
    while i <= n:
      k = square(i)
      total = total + i
      i = i + 1
```

```
n: 3

total: 10

i: 1

k: 1
```

```
def accumulate(n):
    total = 0
    i = 1
    while i <= n:
        k = square(i)
        total = total + k
        i = i + 1
```

```
n: 3

total: 0, 1

i: 1

k: 1
```

```
def accumulate(n):
    total = 10
    i = 1
    while i <= n:
        k = square(i)
        total = total + i
        i = i + 1
```

```
n: 3

total: 10, 11

i: 1

k: 1
```

```
def accumulate(n):
    total = 0
    i = 1
    while i <= n:
        k = square(i)
        total = total + k
        i = i + 1
```

```
n: 3

total: 0, 1

i: 1, 2

k: 1
```

```
def accumulate(n):
    total = 10
    i = 1
    while i <= n:
        k = square(i)
        total = total + i
        i = i + 1
```

```
n: 3

total: 10, 11

i: 1, 2

k: 1
```

```
def accumulate(n):
    total = 0
    i = 1
    while i <= n:
        k = square(i)
        total = total + k
        i = i + 1
```

```
n: 3

total: 0, 1

i: 1, 2

k: 1, 4
```

```
def accumulate(n):
    total = 10
    i = 1
    while i <= n:
        k = square(i)
        total = total + i
        i = i + 1
```

```
n: 3

total: 10, 11

i: 1, 2

k: 1, 4
```

```
def accumulate(n):
    total = 0
    i = 1
    while i <= n:
        k = square(i)
        total = total + k
        i = i + 1
```

```
n: 3

total: 0, 1, 5

i: 1, 2

k: 1, 4
```

```
def accumulate(n):
    total = 10
    i = 1
    while i <= n:
        k = square(i)
        total = total + i
        i = i + 1
```

```
n: 3

total: 10, 11, 15

i: 1, 2

k: 1, 4
```

```
def accumulate(n):
    total = 0
    i = 1
    while i <= n:
        k = square(i)
        total = total + k
        i = i + 1
```

n: 3

total: 0, 1, 5

i: 1, 2, 3

k: 1, 4

```
def accumulate(n):
    total = 10
    i = 1
    while i <= n:
        k = square(i)
        total = total + i
        i = i + 1
```

n: 3

total: 10, 11, 15

i: 1, 2, 3

k: 1, 4

```
def accumulate(n):
    total = 0
    i = 1
    while i <= n:
      k = square(i)
      total = total + k
      i = i + 1
```

```
n: 3

total: 0, 1, 5

i: 1, 2, 3

k: 1, 4, 9
```

```
def accumulate(n):
    total = 10
    i = 1
    while i <= n:
      k = square(i)
      total = total + i
      i = i + 1
```

```
n: 3

total: 10, 11, 15

i: 1, 2, 3

k: 1, 4, 9
```

```
def accumulate(n):
    total = 0
    i = 1
    while i <= n:
        k = square(i)
        total = total + k
        i = i + 1
```

```
n: 3

total: 0, 1, 5, 14

i: 1, 2, 3

k: 1, 4, 9
```

```
def accumulate(n):
    total = 10
    i = 1
    while i <= n:
        k = square(i)
        total = total + i
        i = i + 1
```

```
n: 3

total: 10, 11, 15, 24

i: 1, 2, 3

k: 1, 4, 9
```

```
def accumulate(n):
    total = 0
    i = 1
    while i <= n:
        k = square(i)
        total = total + k
        i = i + 1
```

n: 3

total: 0, 1, 5, 14

i: 1, 2, 3, 4

k: 1, 4, 9

```
def accumulate(n):
    total = 10
    i = 1
    while i <= n:
        k = square(i)
        total = total + i
        i = i + 1
```

n: 3

total: 10, 11, 15, 24

i: 1, 2, 3, 4

k: 1, 4, 9

```
def accumulate(n):
    total = 0
    i = 1
    while i <= n:
        k = square(i)
        total = total + k
        i = i + 1
```

n: 3

total: 0, 1, 5, 14

i: 1, 2, 3, 4

k: 1, 4, 9

```
def accumulate(n):
    total = 10
    i = 1
    while i <= n:
        k = square(i)
        total = total + i
        i = i + 1
```

n: 3

total: 10, 11, 15, 24

i: 1, 2, 3, 4

k: 1, 4, 9

Result

```
accumulate(add, 11, 5, identity) # 11 + 1 + 2 + 3 + 4 + 5
>>> 81
```

```
> call accumulate(add, 11, 5, identity)
> term = identity
> combiner = add
> base = 11
> n = 5
> k = 5
  > call accumulate(add, 11, 4, identity)
  > n = 4
  > k = 4
    > call accumulate(add, 11, 3, identity)
    > n = 3
    > k = 3
      > call accumulate(add, 11, 2, identity)
```

However,

showing all of traces
can overwhelm
students with
too much information

and make it **difficult to grasp an overview** of the behavior

Result

```
accumulate(add, 11, 5, identity) # 11 + 1 + 2 + 3 + 4 + 5
>>> 81
```

```
› call accumulate(add, 11, 5, identity)
› term = identity
› combiner = add
› base = 11
› n = 5
› k = 5
  › call accumulate(add, 11, 4, identity)
  › n = 4
  › k = 4
    › call accumulate(add, 11, 3, identity)
    › n = 3
    › k = 3
      › call accumulate(add, 11, 2, identity)
```

⋮ **+ 24 lines**

Incorrect submission

1 Synthesis

Closest correct program

2 Execute & Record Trace

3 Filter & Highlight Trace Difference

Result

```
accumulate(add, 0, 5, identity) # 0 +
>>> 10

✔ call accumulate(add, 0, 5, identity)
✔ total = 0
› total = 2
```

Expected

```
accumulate(add, 0, 5, identity) # 0 +
>>> 15

✔ call accumulate(add, 0, 5, identity)
✔ total = 0
› total = 1
```

```
def accumulate(n):
    total = 0
    i = 1
    while i <= n:
        k = square(i)
        total = total + k
        i = i + 1
```

n: 3

total: 0, 1, 5, 14

i: 1, 2, 3, 4

k: 1, 4, 9

```
def accumulate(n):
    total = 10
    i = 1
    while i <= n:
        k = square(i)
        total = total + i
        i = i + 1
```

n: 3

total: 10, 11, 15, 24

i: 1, 2, 3, 4

k: 1, 4, 9

```
def accumulate(n):
    total = 0
    i = 1
    while i <= n:
        k = square(i)
        total = total + k
        i = i + 1
```

n: 3

total: 0, 1, 5, 14

i: 1, 2, 3, 4

k: 1, 4, 9

```
def accumulate(n):
    total = 10
    i = 1
    while i <= n:
        k = square(i)
        total = total + i
        i = i + 1
```

n: 3

total: 10, 11, 15, 24

i: 1, 2, 3, 4

k: 1, 4, 9

Without filtering

Result

```
accumulate(add, 11, 5, identity) # 11 + 1 + 2 + 3 + 4 +
>>> 81
```

- ✔ call accumulate(add, 11, 5, identity)
- ✔ term = identity
- ✔ combiner = add
- ✔ base = 11
- ✔ n = 5
- ✔ k = 5
  - ✔ call accumulate(add, 11, 4, identity)
  - ✔ n = 4
  - ✔ k = 4
    - ✔ call accumulate(add, 11, 3, identity)
    - ✔ n = 3
    - ✔ k = 3
      - ✔ call accumulate(add, 11, 2, identity)

⋮ + 24 lines

A

With filtering

Result

```
accumulate(add, 11, 5, identity) # 11 + 1 + 2 + 3
>>> 81
```

- ✔ call accumulate(add, 11, 5, identity)
  - ✔ call accumulate(add, 11, 4, identity)
    - ✔ call accumulate(add, 11, 3, identity)
      - ✔ call accumulate(add, 11, 2, identity)
        - ✔ call accumulate(add, 11, 1, identity)
          - ✔ call accumulate(add, 11, 0, identity)
          - ✔ accumulate(add, 11, 0, identity) returns 11
        - › accumulate(add, 11, 1, identity) returns 23
      - › accumulate(add, 11, 2, identity) returns 36
    - › accumulate(add, 11, 3, identity) returns 50
  - › accumulate(add, 11, 4, identity) returns 65
- › accumulate(add, 11, 5, identity) returns 81

B

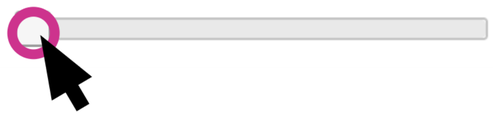
Value Abstraction

Abstract Values into Expressions

Result

```
accumulate(add, 0, 5, identity) # 0 + 1 + 2 + 
>>> 10
```

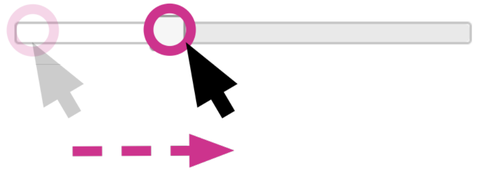
```
✔ call accumulate(add, 0, 5, identity)
✔ total = 0
› total = 2
› total = 4
› total = 6
› total = 8
› total = 10
› accumulate(add, 0, 5, identity) returns 10
```



Result

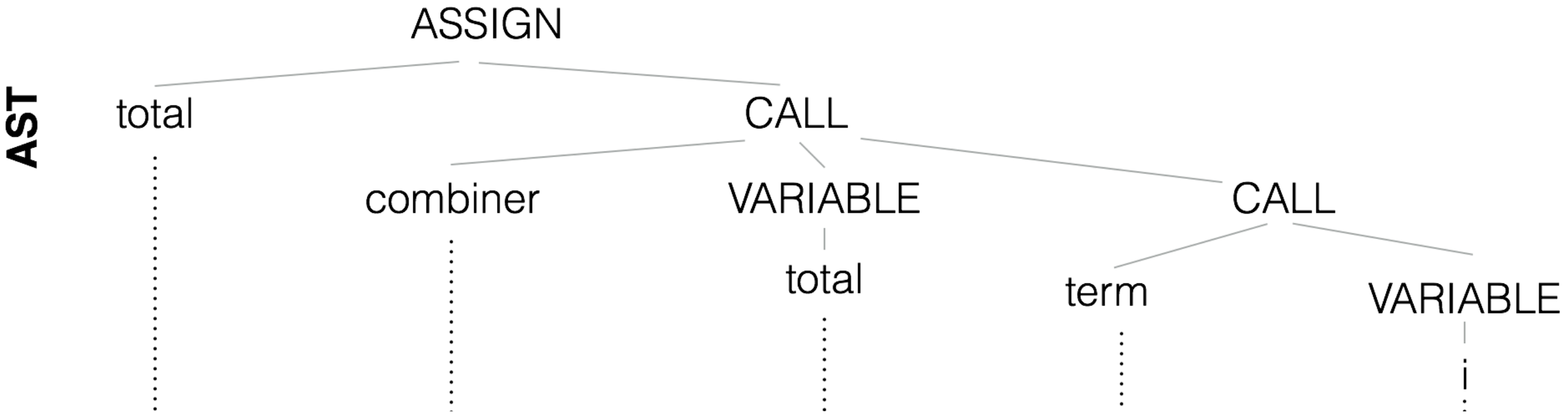
```
accumulate(add, 0, 5, identity) # 0 + 1 + 2 + 
>>> 10
```

```
✔ call accumulate(add, 0, 5, identity)
✔ total = base
› total = add(1, 1)
› total = add(2, 2)
› total = add(3, 3)
› total = add(4, 4)
› total = add(5, 5)
› accumulate(add, 0, 5, identity) returns total
```

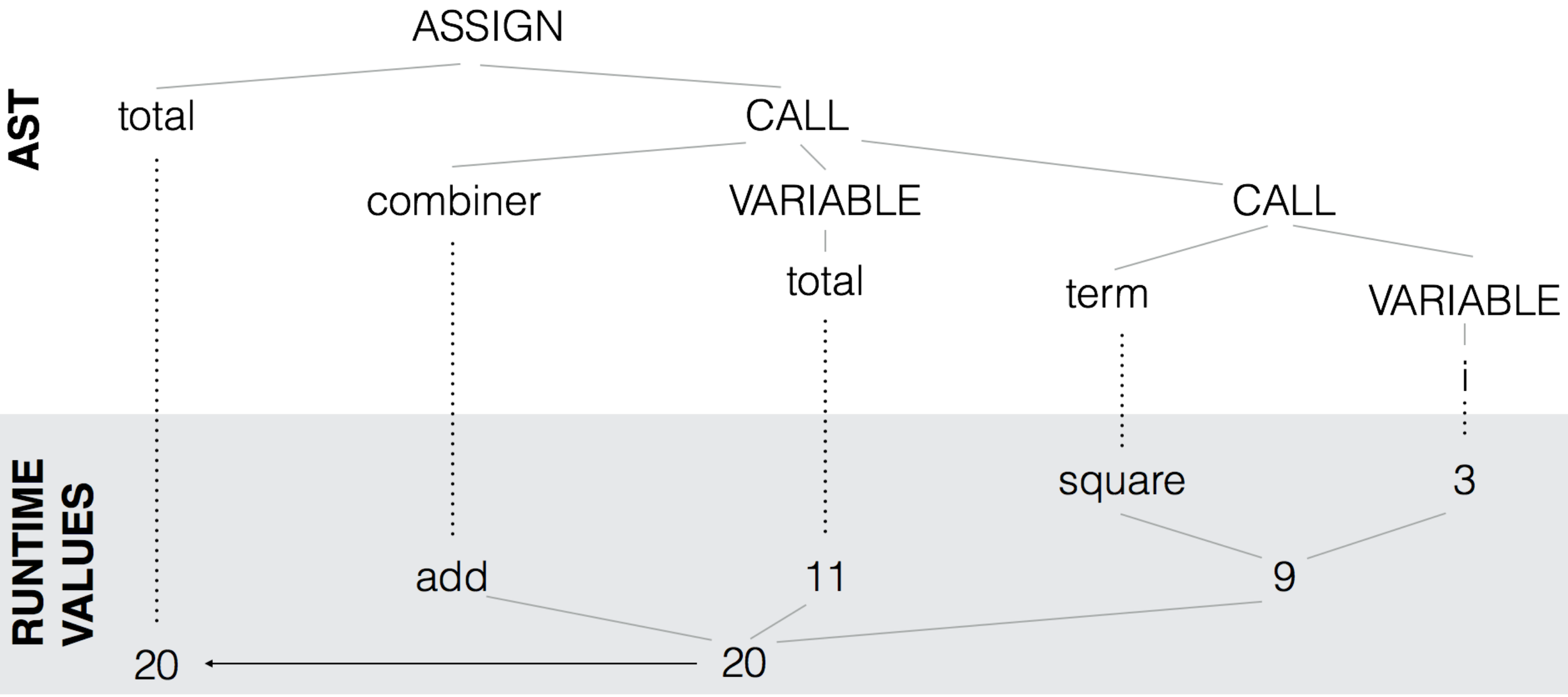


```
total = combiner(i, term(i))
```

```
total = combiner(i, term(i))
```

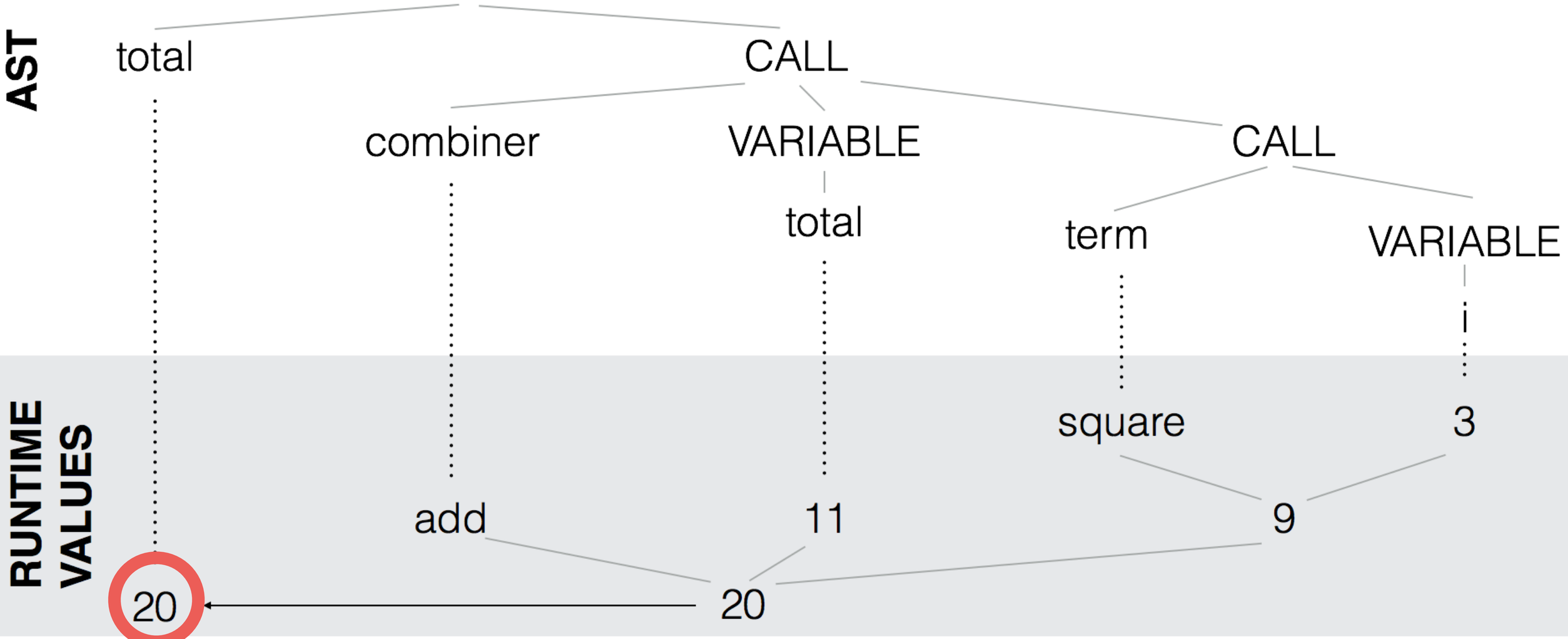


```
total = combiner(i, term(i))
```



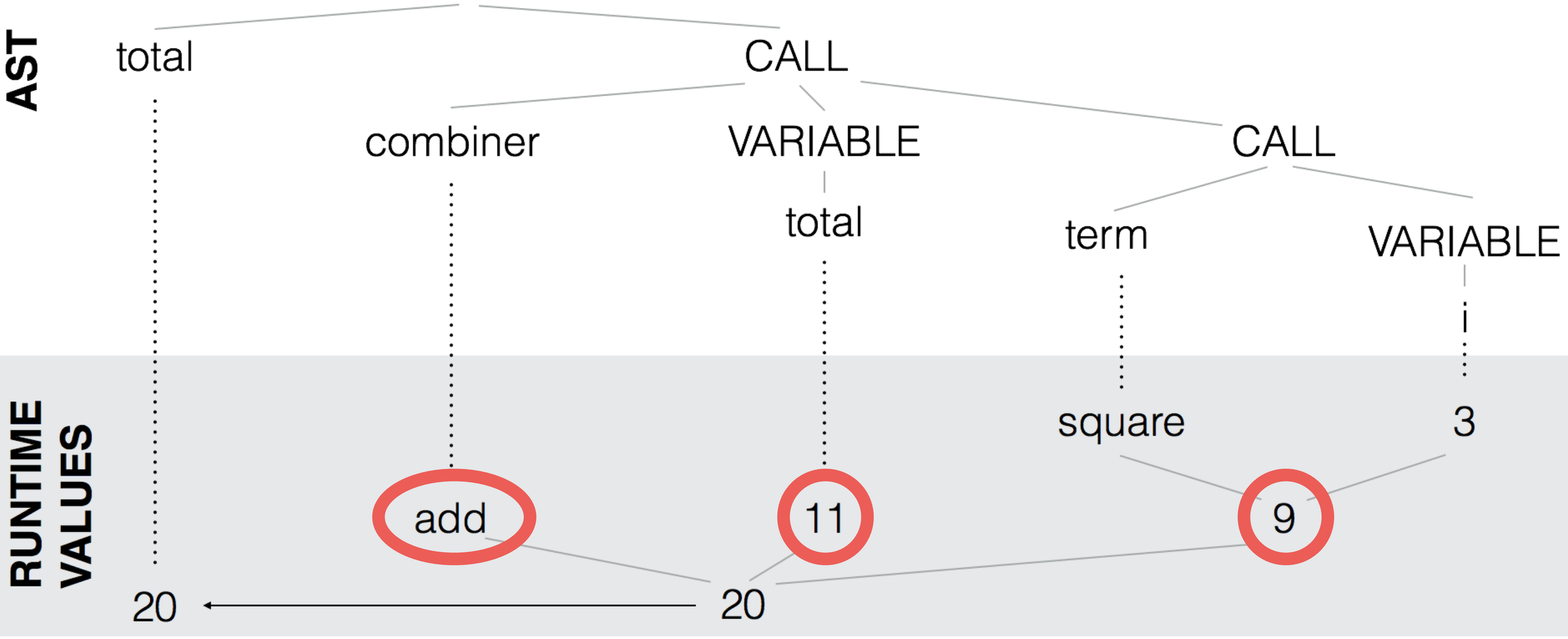
```
total = combiner(i, term(i))
```

20



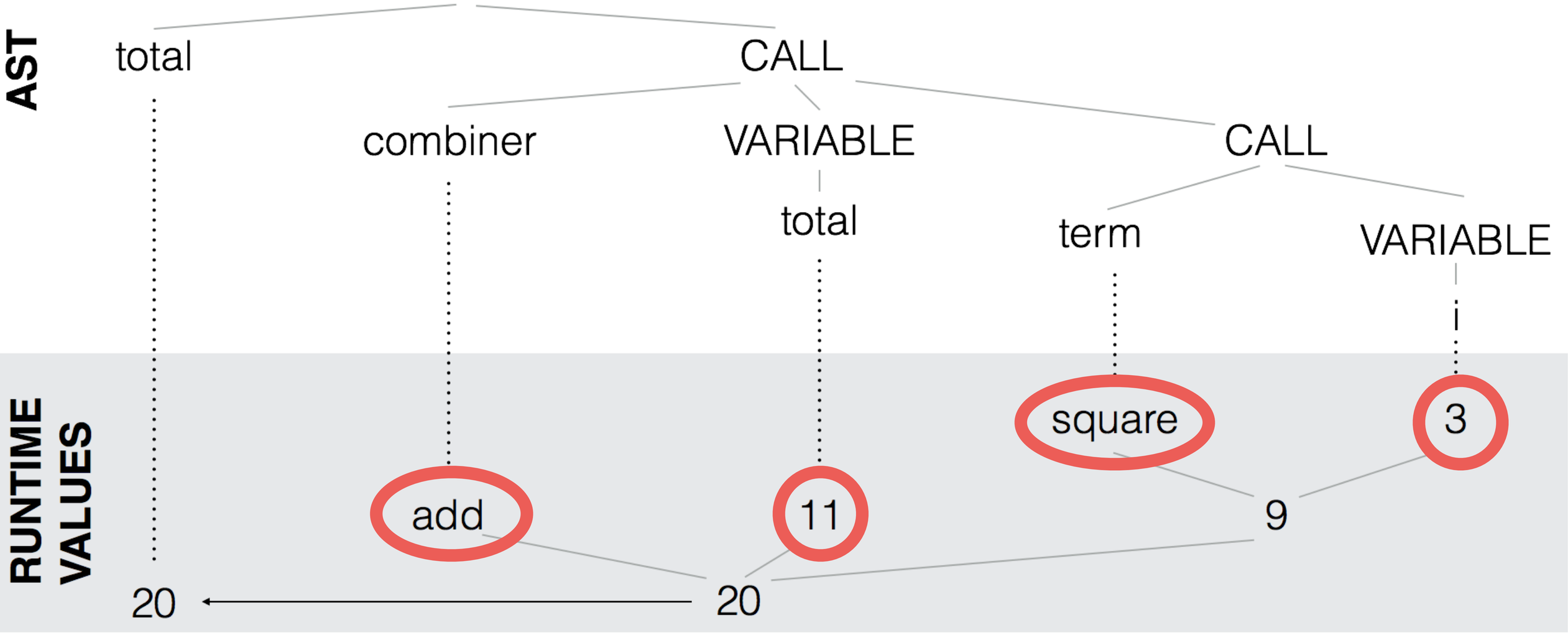
```
total = combiner(i, term(i))
```

```
20 → add(11, 9)
```



```
total = combiner(i, term(i))
```

```
20 → add(11, 9) → add(11, square(3)) → …
```

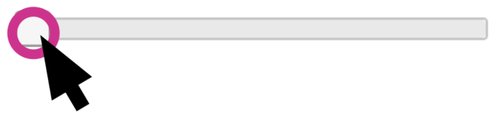


Abstract Values into Expressions

Result

```
accumulate(add, 0, 5, identity) # 0 + 1 + 2 + 
>>> 10
```

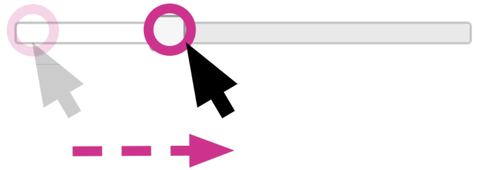
```
✔ call accumulate(add, 0, 5, identity)
✔ total = 0
› total = 2
› total = 4
› total = 6
› total = 8
› total = 10
› accumulate(add, 0, 5, identity) returns 10
```



Result

```
accumulate(add, 0, 5, identity) # 0 + 1 + 2 + 
>>> 10
```

```
✔ call accumulate(add, 0, 5, identity)
✔ total = base
› total = add(1, 1)
› total = add(2, 2)
› total = add(3, 3)
› total = add(4, 4)
› total = add(5, 5)
› accumulate(add, 0, 5, identity) returns total
```



Evaluation

TraceDiff ↔ PythonTutor

17 participants

4 problems (2: TraceDIff, 2: PythonTutor)

2 tasks for each problem:

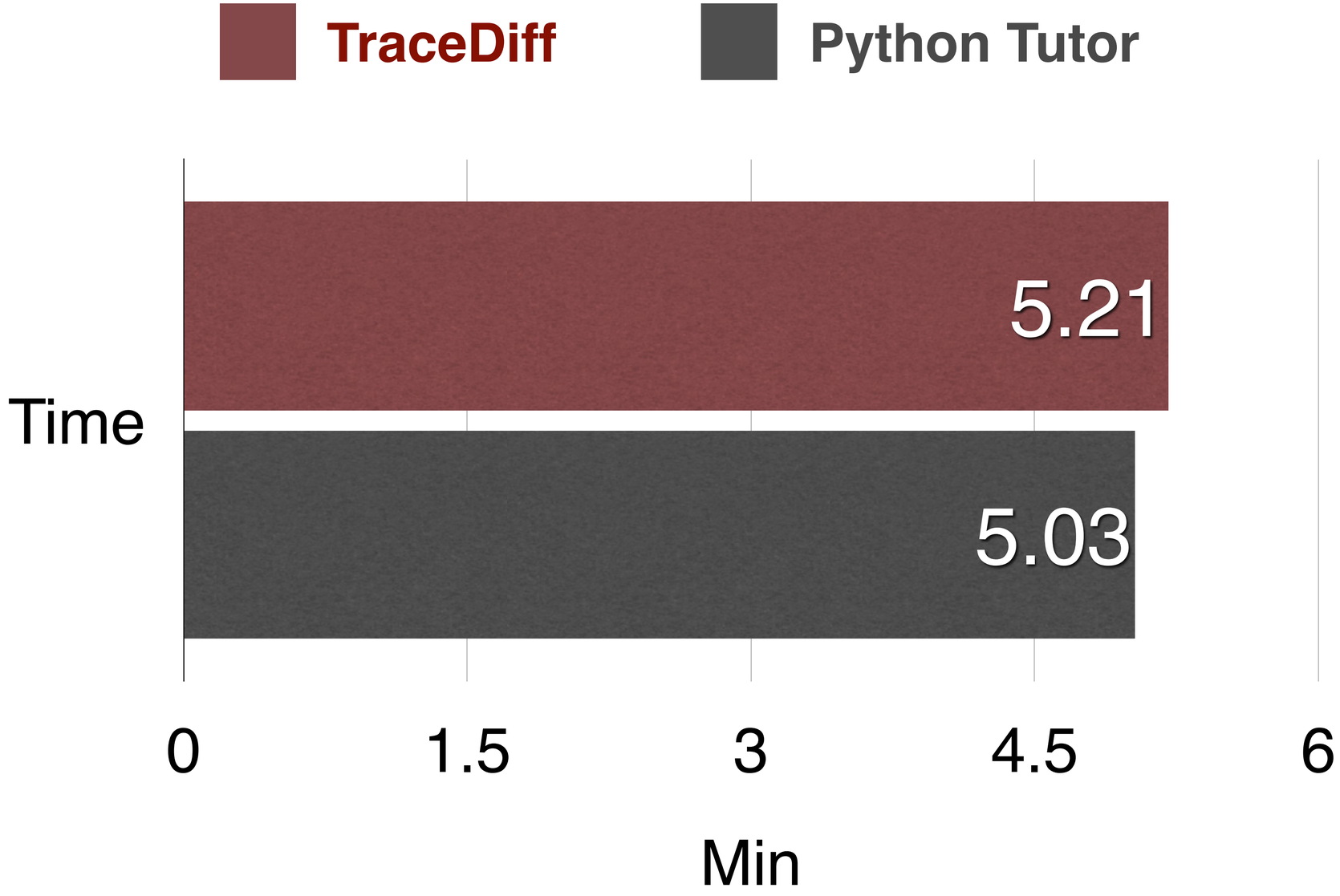
- **Identify** the bug
- **Fix** the bug

RQ1: Can TraceDiff help students
fix bugs **faster** than Python Tutor?

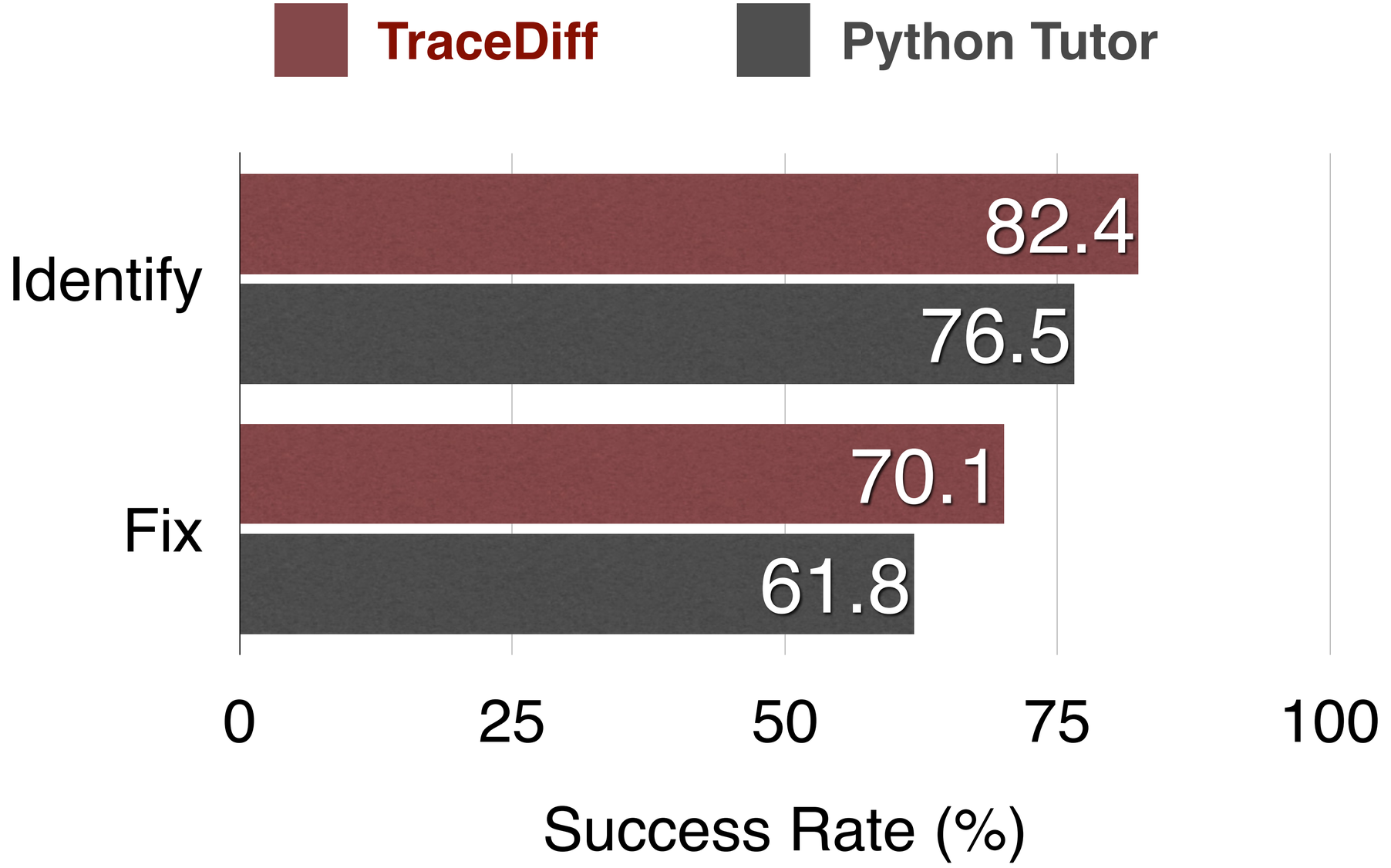
RQ2: Can TraceDiff help students
identify and fix **more bugs** than Python Tutor?

RQ3: Do students perceive TraceDiff
to be **more useful** for debugging tasks?

RQ1: Can students fix bugs **faster**?



RQ2: Can students identify and fix **more bugs**?



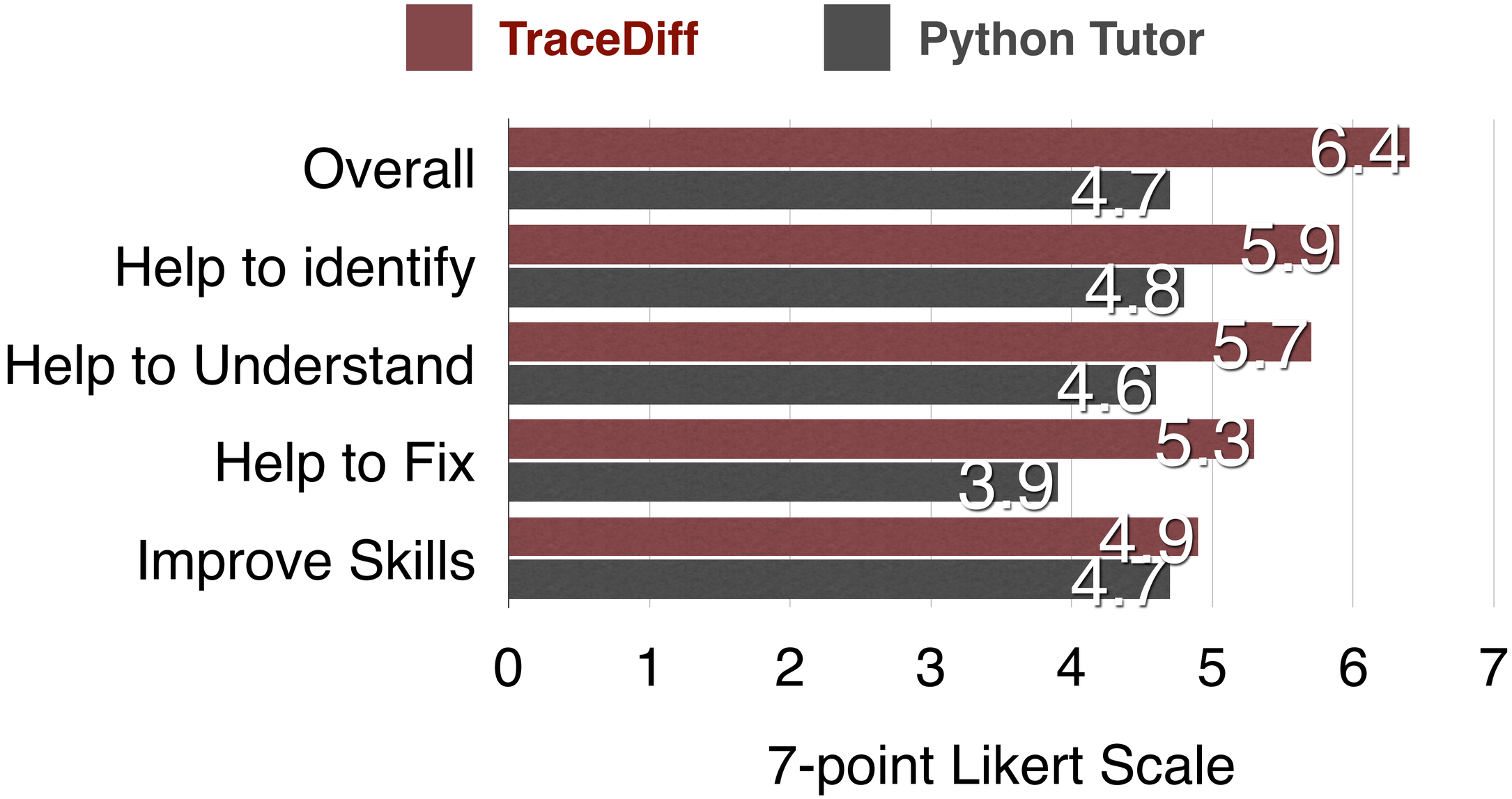
Discussion

Task Selection: **Simple** ↔ **Complex**

Discussion

Task Setting: **Single** ↔ **Multiple** Attempts

RQ3: Do students think the tool is **more useful**?



Future Work

Future Work

Deploy to actual programming courses
and
Evaluate in more realistic situation

Contributions:

1. **A characterization of key design guidelines** for effective programming feedback that can be generated by state-of-the-art synthesis techniques, informed by a formative study.
2. **The implementation of hints in an interactive debugging interface**, appropriate for deployment and evaluation in a massive programming classroom.
3. **Quantitative and qualitative results of a controlled experiment with 17 students** where we compare TraceDiff with Python Tutor interface.

GitHub ryosuzuki/trace-diff

This repository Search Pull requests Issues Marketplace Explore

ryosuzuki / trace-diff

Watch 1 Unstar 1 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

TraceDiff: Debugging Unexpected Code Behavior Using Trace Divergences [VL/HCC 2017]
https://ryosuzuki.github.io/trace-diff/

Edit

Add topics

219 commits 2 branches 2 releases 1 contributor MIT

Branch: master New pull request Create new file Upload files Find file Clone or download

ryosuzuki Update README.md Latest commit 11f24e0 on Aug 14

data	Rebuild	2 months ago
resources	Add resources	2 months ago

Thank you

How can we turn
the synthesized information into
more pedagogically useful
higher-level feedback?

```
accumulate(add, 0, 5, identity)
x 14 # 0 + 0 + 2 + 3 + 4 + 5
o 15 # 0 + 1 + 2 + 3 + 4 + 5
```

Test Case Feedback is **Scalable**

Student Code

```
1 def product(n, term):
2     total, k = 0, 1
3     while k <= n:
4         total, k = total * term(k), k + 1
5     return total
6
```

Test Results

Test	Input		Result	Expected	Output
1	(3, lambda x: x),	→	0	6	
2	(5, lambda x: x),	→	0	120	
3	(3, lambda x: x * x),	→	0	36	
4	(5, lambda x: x * x),	→	0	14400	

Test Case Feedback **does not Provide Scaffolds**

Student Code

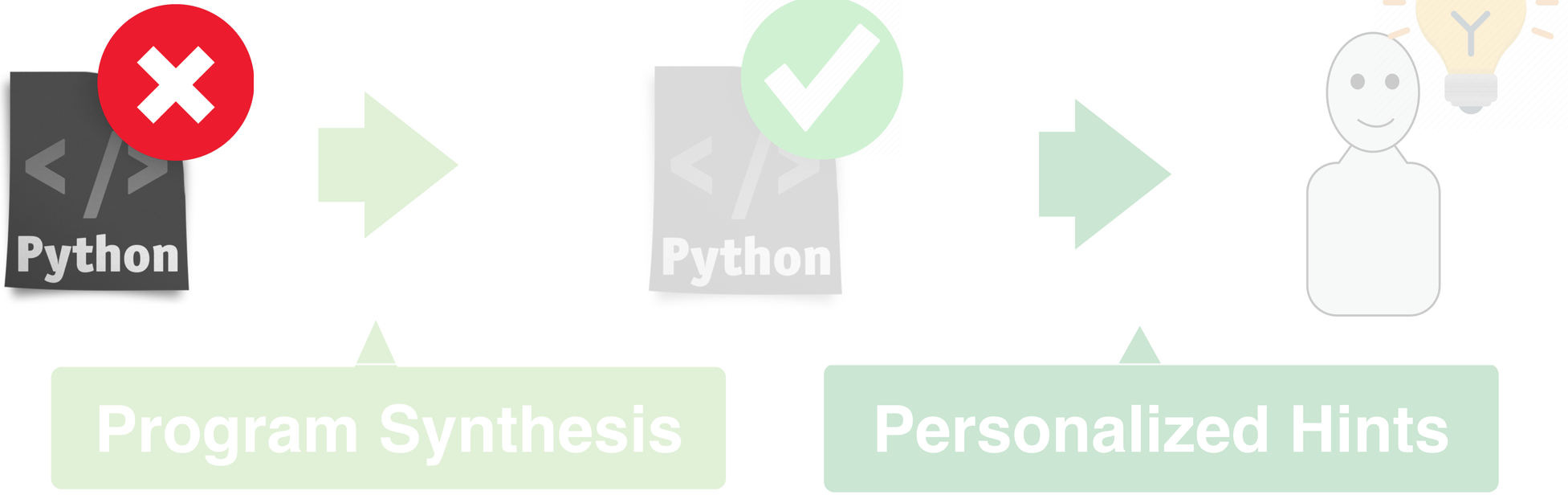
```
1 def product(n, term):
2     total, k = 0, 1
3     while k <= n:
4         total, k = total * term(k),
5     return total
6
```

Test Results

Test	Input		R	Output
1	(3, lambda x: x),	→	0	6
2	(5, lambda x: x),	→	0	120
3	(3, lambda x: x * x),	→	0	36
4	(5, lambda x: x * x),	→	0	14400

Program Synthesis Feedback

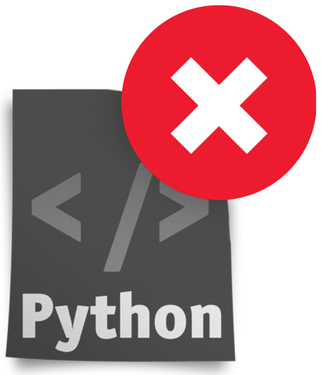
```
1 def product(n, term):
2     total, k = 0, 1
3     while k <= n:
4         total, k = total * term(k), k + 1
5     return total
6
```

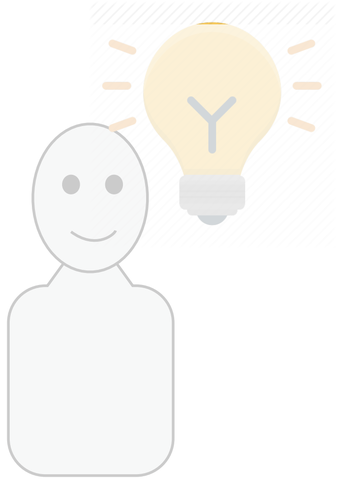


Program Synthesis Feedback

```
1 def product(n, term):
2     total, k = 0, 1
3     while k <= n:
4         total, k = total * term(k), k + 1
5     return total
6
```

```
total, k = 1, 0
```





Program Synthesis

Personalized Hints

Program Synthesis Feedback

```
1 def product(n, term):
2     total, k = 0, 1
3     while k <= n:
4         total, k =
5     return total
6
```

```
total, k = 1, 0
```

Line 2 needs to be changed

Program Synthesis

Personalized Hints

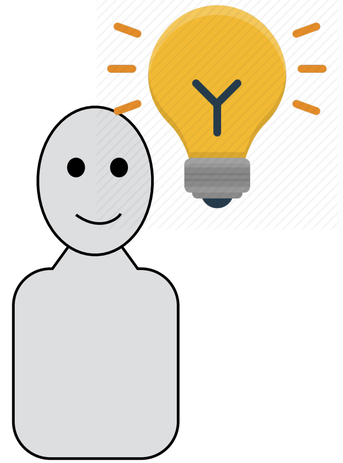
Program Synthesis Feedback

```
1 def product(n, term):
2     total, k = 0, 1
3     while k <= n:
4         total, k =
5     return total
6
```

```
total, k = 1, 0
```

Line 2 needs to be changed

In line 2, change total = 0 to total = 1



Program Synthesis

Personalized Hints

Program Synthesis Feedback

Pointing

Line 2 needs to be changed

In line 2, check total

High-level Hints

?

Bottom-out

In line 2, change total = 0 to total = 1

Program Synthesis Feedback

Pointing

Line 2 needs to be changed

In line 2, check total

High-level Hints

How can we turn the synthesized information into **more pedagogically useful, higher-level** feedback?

Bottom-out

In line 2, change total = 0 to total = 1

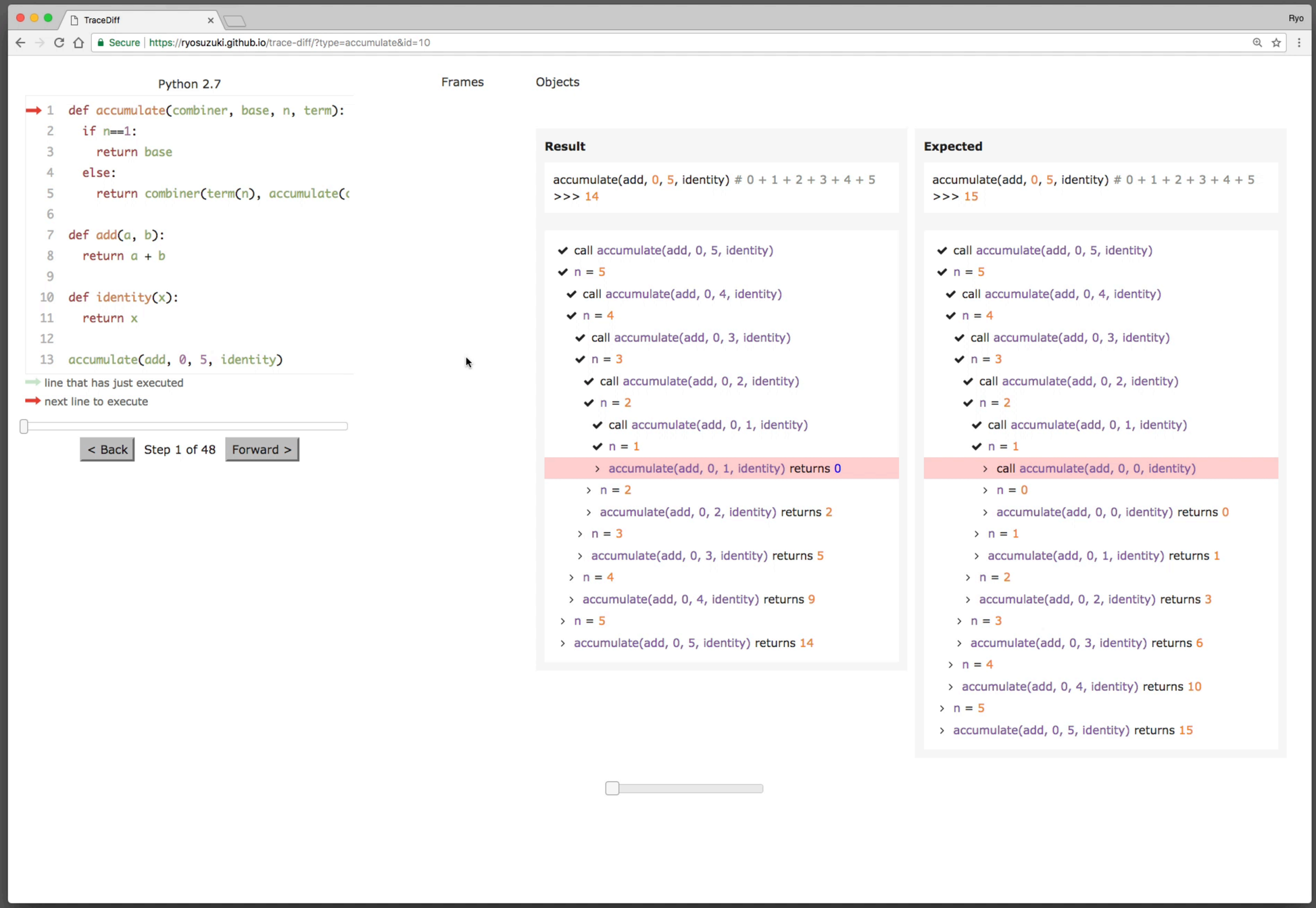
```
def accumulate(combiner, base, n, term):
    total = base
    i = 1
    while i<=n:
        total = combiner(i, term(i))
        total = combiner(total, term(i))
        i = i+1
    return total
```

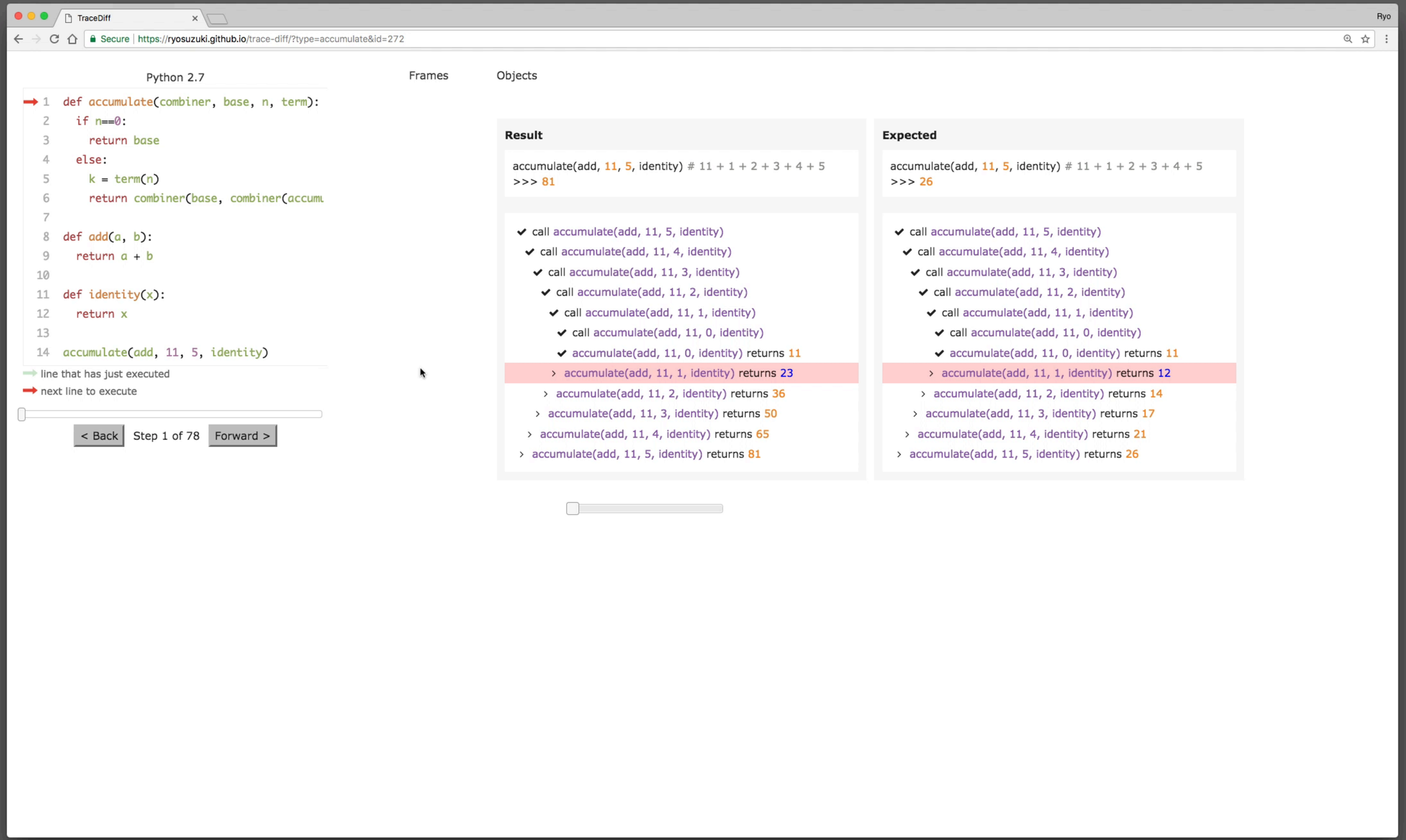
Test

```
accumulate(add, 0, 5, identity)
x 10 # 5 + 5
o 15 # 0 + 1 + 2 + 3 + 4 + 5
```

Record and Filter Code Traces

1. Function Call
2. Variable Assignment
3. Return Statement


TraceDiff
Ryo
Secure | https://ryosuzuki.github.io/trace-diff/?type=accumulate&id=10
Python 2.7
1 def accumulate(combiner, base, n, term):
2 if n==1:
3 return base
4 else:
5 return combiner(term(n), accumulate(
6
7 def add(a, b):
8 return a + b
9
10 def identity(x):
11 return x
12
13 accumulate(add, 0, 5, identity)
line that has just executed
next line to execute
< Back
Step 1 of 48
Forward >
Frames
Objects
Result
accumulate(add, 0, 5, identity) # 0 + 1 + 2 + 3 + 4 + 5
>>> 14
call accumulate(add, 0, 5, identity)
n = 5
call accumulate(add, 0, 4, identity)
n = 4
call accumulate(add, 0, 3, identity)
n = 3
call accumulate(add, 0, 2, identity)
n = 2
call accumulate(add, 0, 1, identity)
n = 1
accumulate(add, 0, 1, identity) returns 0
n = 2
accumulate(add, 0, 2, identity) returns 2
n = 3
accumulate(add, 0, 3, identity) returns 5
n = 4
accumulate(add, 0, 4, identity) returns 9
n = 5
accumulate(add, 0, 5, identity) returns 14
Expected
accumulate(add, 0, 5, identity) # 0 + 1 + 2 + 3 + 4 + 5
>>> 15
call accumulate(add, 0, 5, identity)
n = 5
call accumulate(add, 0, 4, identity)
n = 4
call accumulate(add, 0, 3, identity)
n = 3
call accumulate(add, 0, 2, identity)
n = 2
call accumulate(add, 0, 1, identity)
n = 1
call accumulate(add, 0, 0, identity)
n = 0
accumulate(add, 0, 0, identity) returns 0
n = 1
accumulate(add, 0, 1, identity) returns 1
n = 2
accumulate(add, 0, 2, identity) returns 3
n = 3
accumulate(add, 0, 3, identity) returns 6
n = 4
accumulate(add, 0, 4, identity) returns 10
n = 5
accumulate(add, 0, 5, identity) returns 15


TraceDiff
Ryo
Secure | https://ryosuzuki.github.io/trace-diff/?type=accumulate&id=272
Python 2.7
1 def accumulate(combiner, base, n, term):
2 if n==0:
3 return base
4 else:
5 k = term(n)
6 return combiner(base, combiner(accumu
7
8 def add(a, b):
9 return a + b
10
11 def identity(x):
12 return x
13
14 accumulate(add, 11, 5, identity)
line that has just executed
next line to execute
< Back
Step 1 of 78
Forward >
Frames
Objects
Result
accumulate(add, 11, 5, identity) # 11 + 1 + 2 + 3 + 4 + 5
>>> 81
call accumulate(add, 11, 5, identity)
call accumulate(add, 11, 4, identity)
call accumulate(add, 11, 3, identity)
call accumulate(add, 11, 2, identity)
call accumulate(add, 11, 1, identity)
call accumulate(add, 11, 0, identity)
accumulate(add, 11, 0, identity) returns 11
accumulate(add, 11, 1, identity) returns 23
accumulate(add, 11, 2, identity) returns 36
accumulate(add, 11, 3, identity) returns 50
accumulate(add, 11, 4, identity) returns 65
accumulate(add, 11, 5, identity) returns 81
Expected
accumulate(add, 11, 5, identity) # 11 + 1 + 2 + 3 + 4 + 5
>>> 26
call accumulate(add, 11, 5, identity)
call accumulate(add, 11, 4, identity)
call accumulate(add, 11, 3, identity)
call accumulate(add, 11, 2, identity)
call accumulate(add, 11, 1, identity)
call accumulate(add, 11, 0, identity)
accumulate(add, 11, 0, identity) returns 11
accumulate(add, 11, 1, identity) returns 12
accumulate(add, 11, 2, identity) returns 14
accumulate(add, 11, 3, identity) returns 17
accumulate(add, 11, 4, identity) returns 21
accumulate(add, 11, 5, identity) returns 26

Result

```
accumulate(add, 11, 5, identity) # 11 + 1 + 2 + 3
>>> 81
```

- ✔ call accumulate(add, 11, 5, identity)
  - ✔ call accumulate(add, 11, 4, identity)
    - ✔ call accumulate(add, 11, 3, identity)
      - ✔ call accumulate(add, 11, 2, identity)
        - ✔ call accumulate(add, 11, 1, identity)
          - ✔ call accumulate(add, 11, 0, identity)
          - ✔ accumulate(add, 11, 0, identity) returns 11
        - › accumulate(add, 11, 1, identity) returns 23
      - › accumulate(add, 11, 2, identity) returns 36
    - › accumulate(add, 11, 3, identity) returns 50
  - › accumulate(add, 11, 4, identity) returns 65
- › accumulate(add, 11, 5, identity) returns 81

A

Expected

```
accumulate(add, 11, 5, identity) # 11 + 1 + 2 + 3
>>> 26
```

- ✔ call accumulate(add, 11, 5, identity)
  - ✔ call accumulate(add, 11, 4, identity)
    - ✔ call accumulate(add, 11, 3, identity)
      - ✔ call accumulate(add, 11, 2, identity)
        - ✔ call accumulate(add, 11, 1, identity)
          - ✔ call accumulate(add, 11, 0, identity)
          - ✔ accumulate(add, 11, 0, identity) returns 11
        - › accumulate(add, 11, 1, identity) returns 12
      - › accumulate(add, 11, 2, identity) returns 14
    - › accumulate(add, 11, 3, identity) returns 17
  - › accumulate(add, 11, 4, identity) returns 21
- › accumulate(add, 11, 5, identity) returns 26

B

Python 2.7

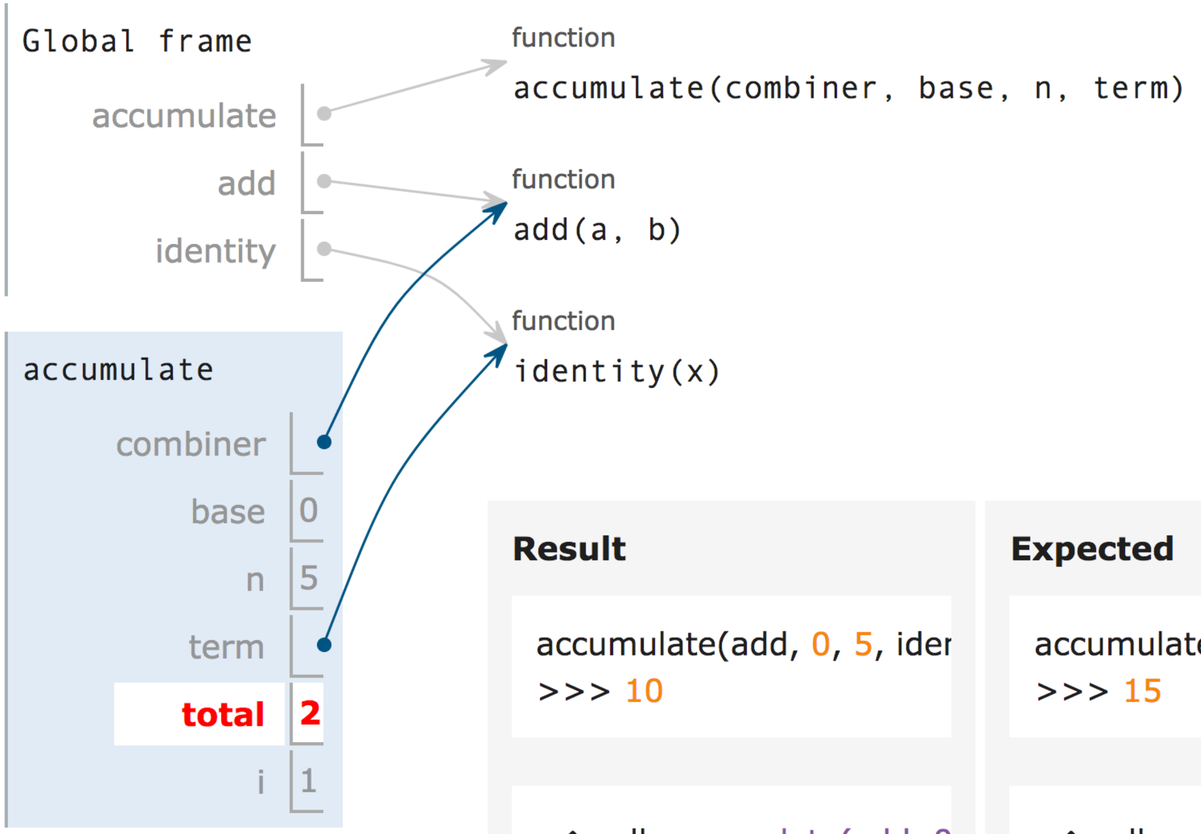
```
1  def accumulate(combiner, base, n, term):
2    total = base
3    i = 1
4    while i<=n:
5      total = combiner(i, term(i))
6      i = i+1
7    return total
8
9  def add(a, b):
10   return a + b
11
12 def identity(x):
13   return x
14
15 accumulate(add, 0, 5, identity)
```

line that has just executed

next line to execute

< Back | Step 16 of 55 | Forward >

Frames Objects



Result

accumulate(add, 0, 5, ider
>>> 10

- ✔ call accumulate(add, 0
- ✔ total = 0
- › total = 2
- › total = 4
- › total = 6
- › total = 8
- › total = 10

Expected

accumulate(add, 0, 5,
>>> 15

- ✔ call accumulate(ac
- ✔ total = 0
- › total = 1
- › total = 3
- › total = 6
- › total = 10
- › total = 15

Code

```
def accumulate(combiner, base, n, term):
    total = base
    i = 1
    while i<=n:
        total = combiner(i, term(i))
        total = combiner(total, term(i))
        i = i+1
    return total
```

Test

```
accumulate(add, 0, 5, identity)
x 10
o 15
```

D1: Encourage students to explore code execution with a visual debugger

D2: Describe how actual behavior differs from expected behavior

D3: Refer to concrete code locations and behavior to provide a starting point for exploration

Result

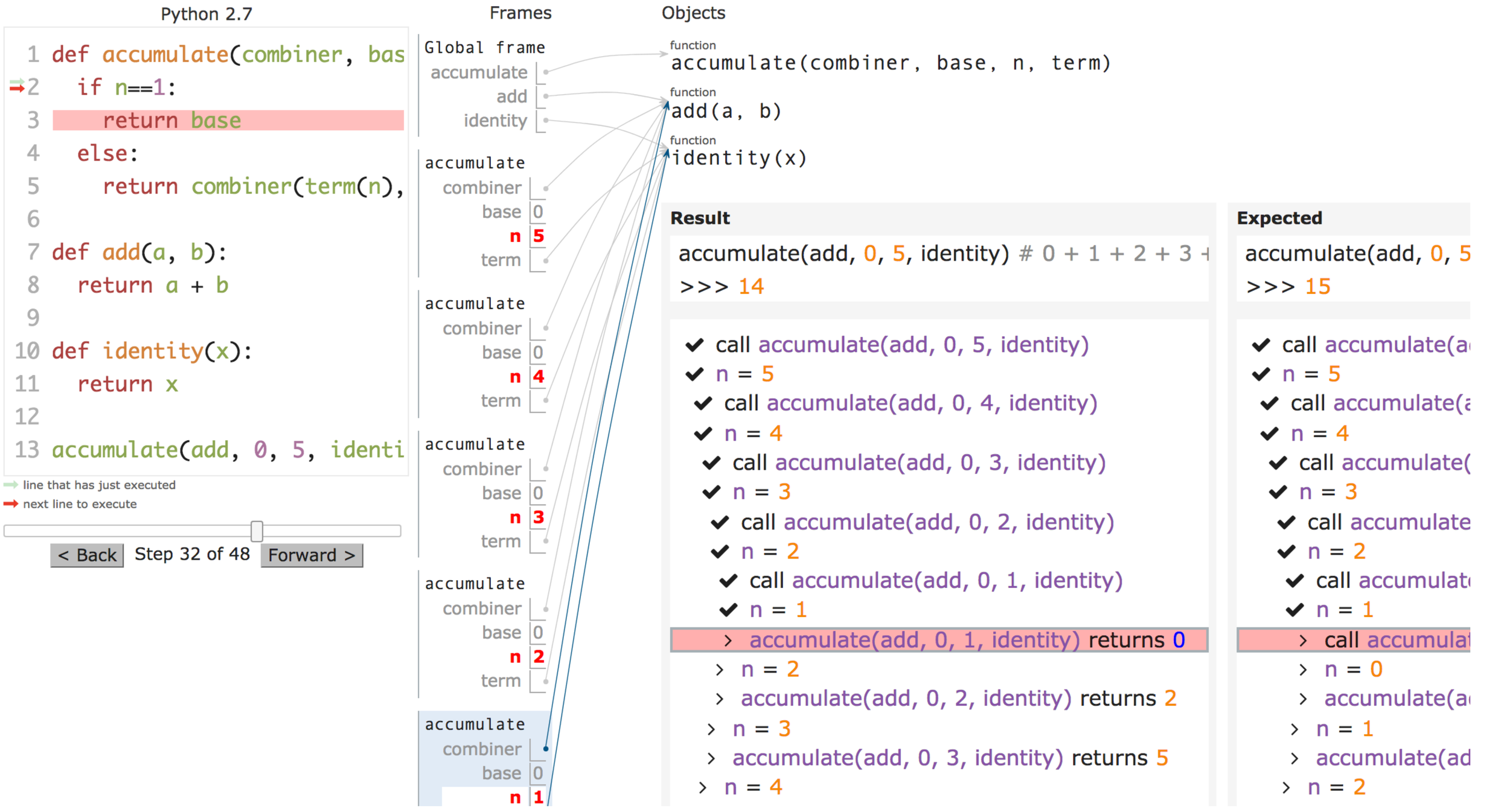
```
accumulate(add, 0, 5, identity) # 0 + 1 + 2 + 3 +
>>> 14
```

```
✔ call accumulate(add, 0, 5, identity)
✔ n = 5
 ✔ call accumulate(add, 0, 4, identity)
 ✔ n = 4
  ✔ call accumulate(add, 0, 3, identity)
  ✔ n = 3
   ✔ call accumulate(add, 0, 2, identity)
   ✔ n = 2
    ✔ call accumulate(add, 0, 1, identity)
    ✔ n = 1
    › accumulate(add, 0, 1, identity) returns 0
   › n = 2
   › accumulate(add, 0, 2, identity) returns 2
  › n = 3
  › accumulate(add, 0, 3, identity) returns 5
 › n = 4
 › accumulate(add, 0, 4, identity) returns 9
› n = 5
› accumulate(add, 0, 5, identity) returns 14
```

Expected

```
accumulate(add, 0, 5, identity) # 0 + 1 + 2 + 3 +
>>> 15
```

```
✔ call accumulate(add, 0, 5, identity)
✔ n = 5
 ✔ call accumulate(add, 0, 4, identity)
 ✔ n = 4
  ✔ call accumulate(add, 0, 3, identity)
  ✔ n = 3
   ✔ call accumulate(add, 0, 2, identity)
   ✔ n = 2
    ✔ call accumulate(add, 0, 1, identity)
    ✔ n = 1
    › call accumulate(add, 0, 0, identity)
    › n = 0
    › accumulate(add, 0, 0, identity) returns 0
   › n = 1
   › accumulate(add, 0, 1, identity) returns 1
  › n = 2
  › accumulate(add, 0, 2, identity) returns 3
 › n = 3
 › accumulate(add, 0, 3, identity) returns 6
› n = 4
› accumulate(add, 0, 4, identity) returns 10
› n = 5
› accumulate(add, 0, 5, identity) returns 15
```


Python 2.7
1 def accumulate(combiner, bas
2 if n==1:
3 return base
4 else:
5 return combiner(term(n),
6
7 def add(a, b):
8 return a + b
9
10 def identity(x):
11 return x
12
13 accumulate(add, 0, 5, identi
line that has just executed
next line to execute
< Back
Step 32 of 48
Forward >
Frames
Global frame
accumulate
add
identity
accumulate
combiner
base 0
n 5
term
accumulate
combiner
base 0
n 4
term
accumulate
combiner
base 0
n 3
term
accumulate
combiner
base 0
n 2
term
accumulate
combiner
base 0
n 1
Objects
function
accumulate(combiner, base, n, term)
function
add(a, b)
function
identity(x)
Result
accumulate(add, 0, 5, identity) # 0 + 1 + 2 + 3 +
>>> 14
✔ call accumulate(add, 0, 5, identity)
✔ n = 5
✔ call accumulate(add, 0, 4, identity)
✔ n = 4
✔ call accumulate(add, 0, 3, identity)
✔ n = 3
✔ call accumulate(add, 0, 2, identity)
✔ n = 2
✔ call accumulate(add, 0, 1, identity)
✔ n = 1
› accumulate(add, 0, 1, identity) returns 0
› n = 2
› accumulate(add, 0, 2, identity) returns 2
› n = 3
› accumulate(add, 0, 3, identity) returns 5
› n = 4
Expected
accumulate(add, 0, 5
>>> 15
✔ call accumulate(a
✔ n = 5
✔ call accumulate(a
✔ n = 4
✔ call accumulate(
✔ n = 3
✔ call accumulate
✔ n = 2
✔ call accumulat
✔ n = 1
› call accumulat
› n = 0
› accumulate(a
› n = 1
› accumulate(ad
› n = 2

Result

```
accumulate(add, 0, 5, identity) # 0 + 1 + 2 + 
>>> 10
```

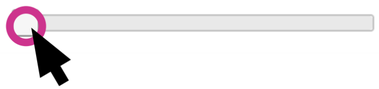
- ✔ call accumulate(add, 0, 5, identity)
- ✔ total = 0
- › total = 2
- › total = 4
- › total = 6
- › total = 8
- › total = 10
- › accumulate(add, 0, 5, identity) returns 10

Expected

```
accumulate(add, 0, 5, identity) # 0 + 1 + 2 + 
>>> 15
```

- ✔ call accumulate(add, 0, 5, identity)
- ✔ total = 0
- › total = 1
- › total = 3
- › total = 6
- › total = 10
- › total = 15
- › accumulate(add, 0, 5, identity) returns 15

A



Result

```
accumulate(add, 0, 5, identity) # 0 + 1 + 2 + 
>>> 10
```

- ✔ call accumulate(add, 0, 5, identity)
- ✔ total = base
- › total = add(1, 1)
- › total = add(2, 2)
- › total = add(3, 3)
- › total = add(4, 4)
- › total = add(5, 5)
- › accumulate(add, 0, 5, identity) returns total

Expected

```
accumulate(add, 0, 5, identity) # 0 + 1 + 2 + 
>>> 15
```

- ✔ call accumulate(add, 0, 5, identity)
- ✔ total = base
- › total = add(0, 1)
- › total = add(1, 2)
- › total = add(3, 3)
- › total = add(6, 4)
- › total = add(10, 5)
- › accumulate(add, 0, 5, identity) returns total

B