

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

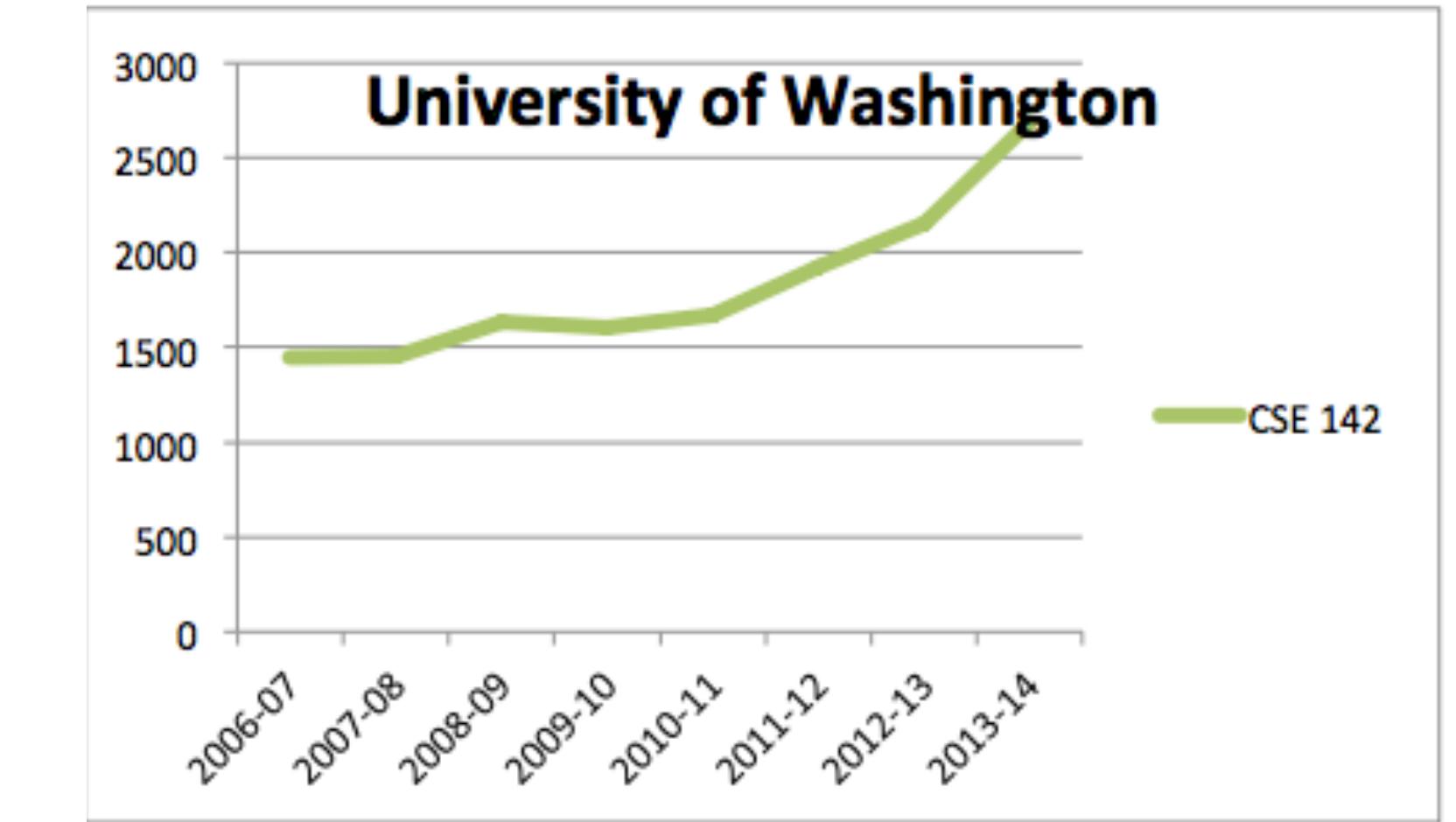
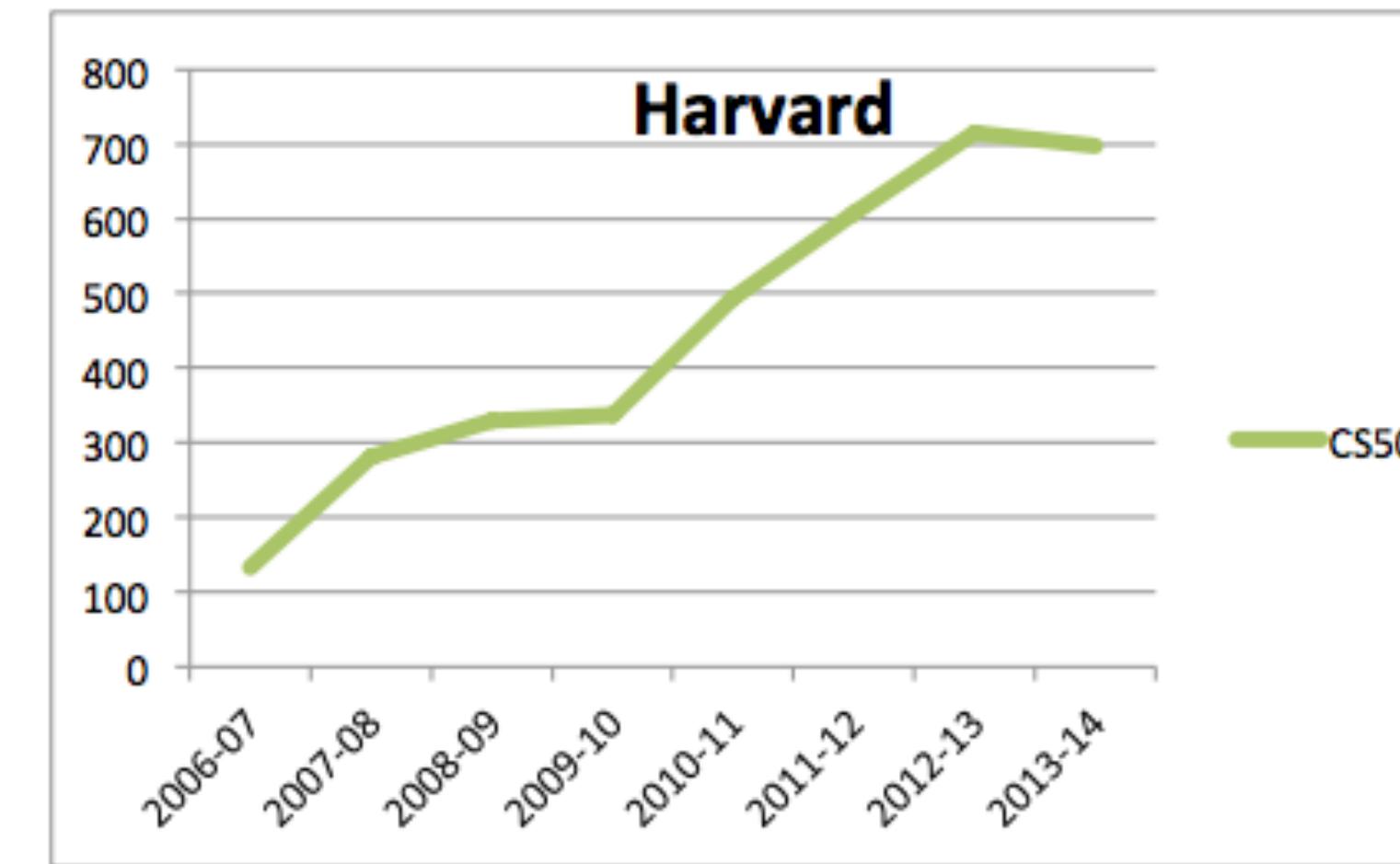
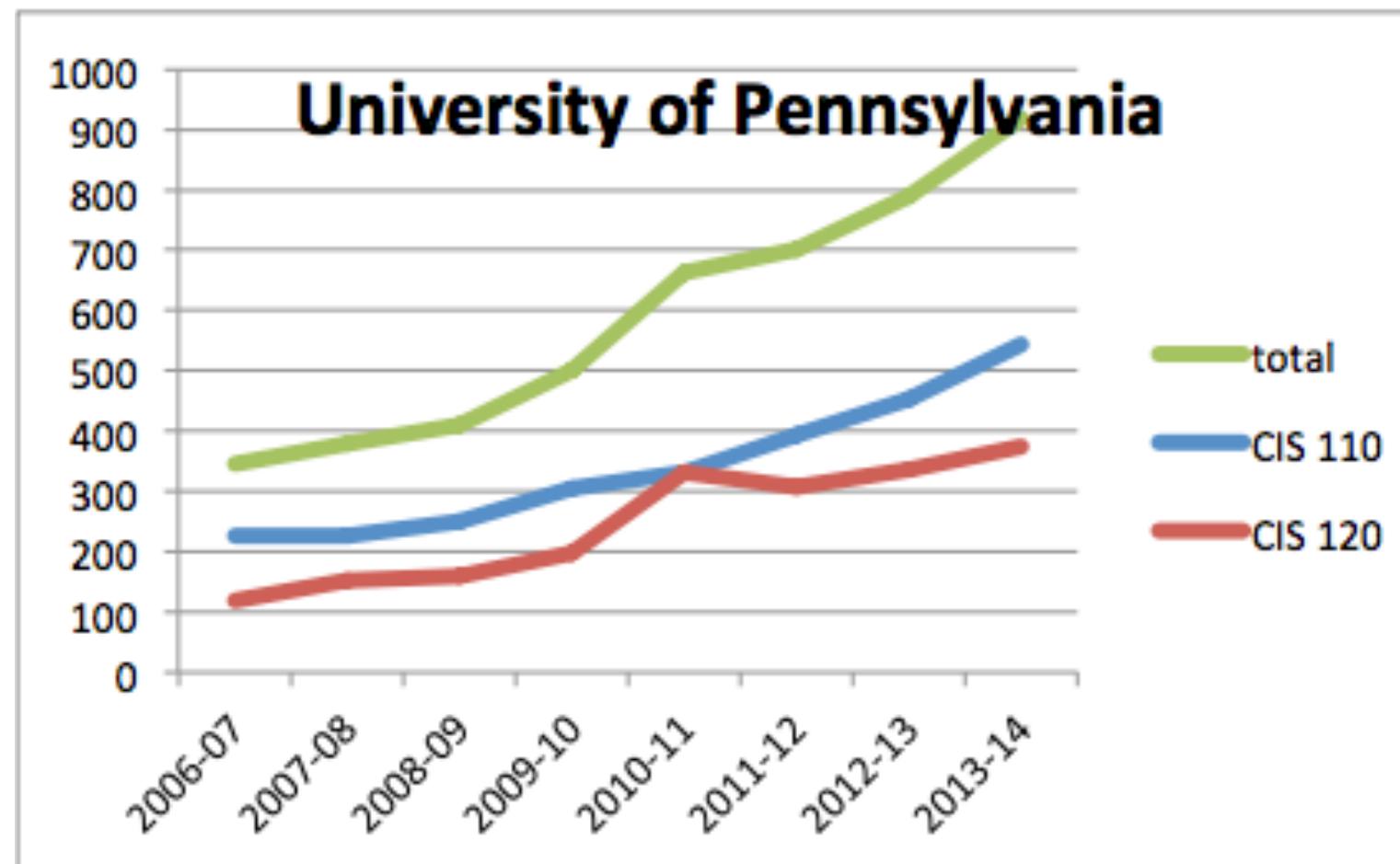
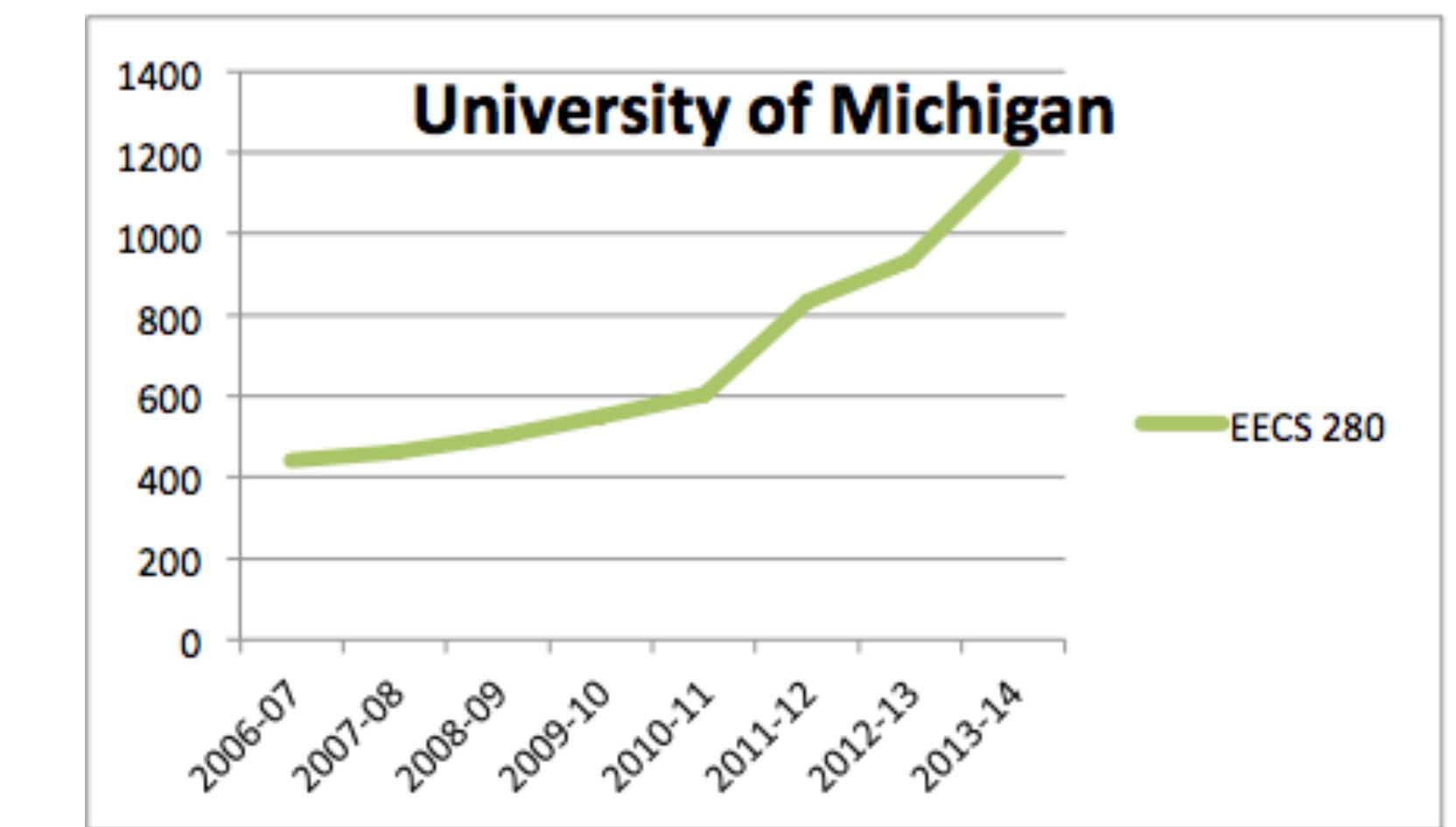
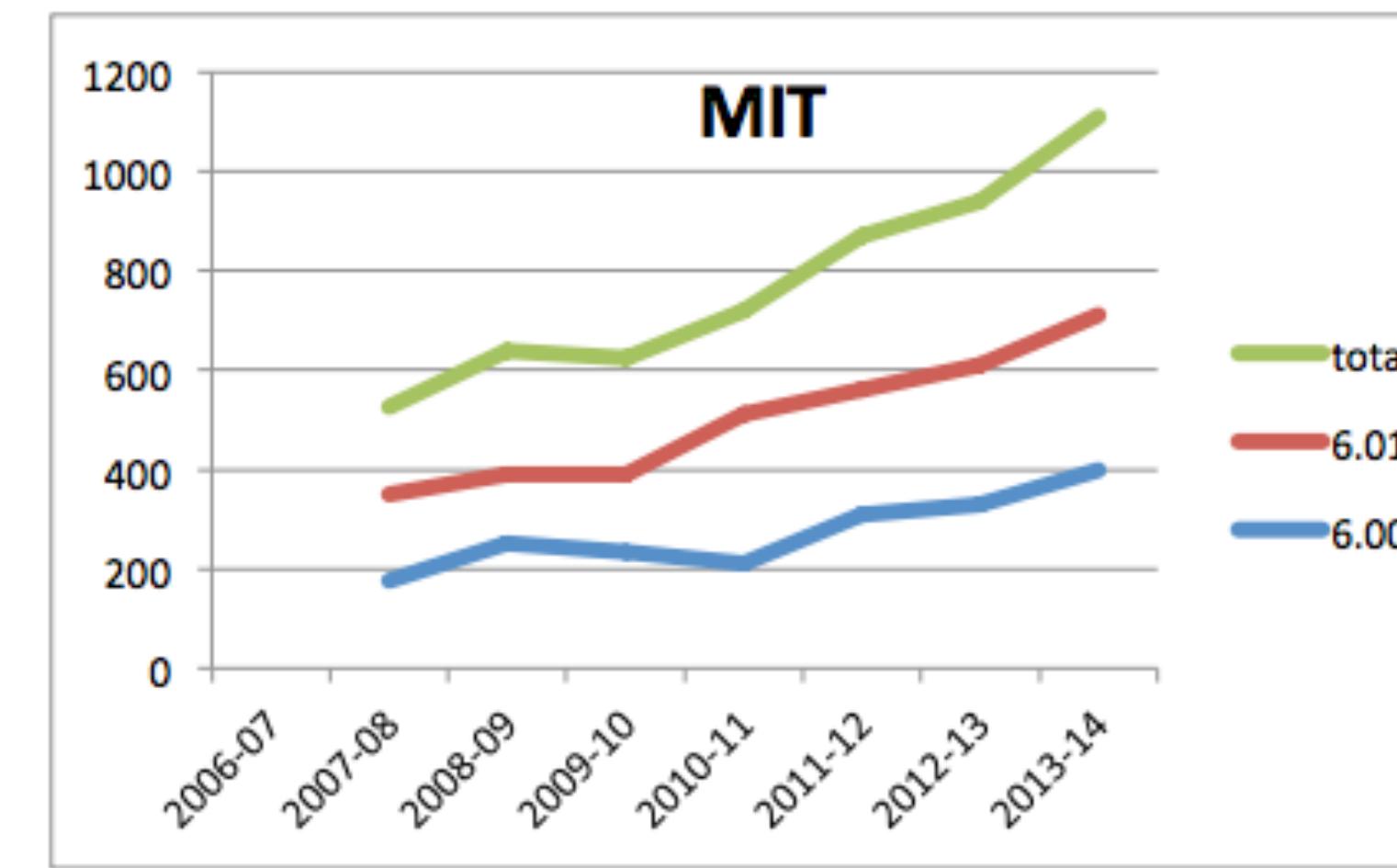
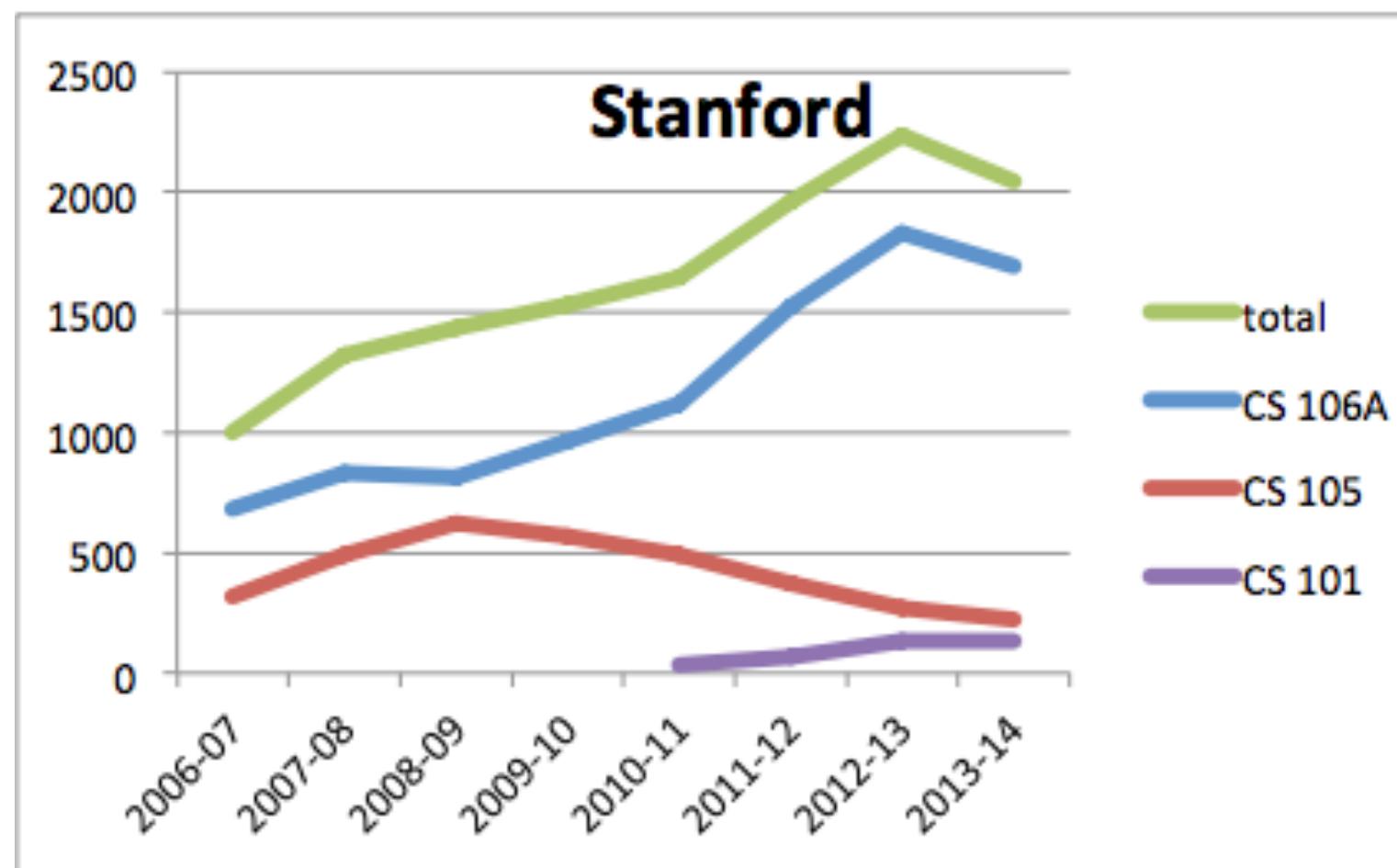


Universidade Federal
de Campina Grande

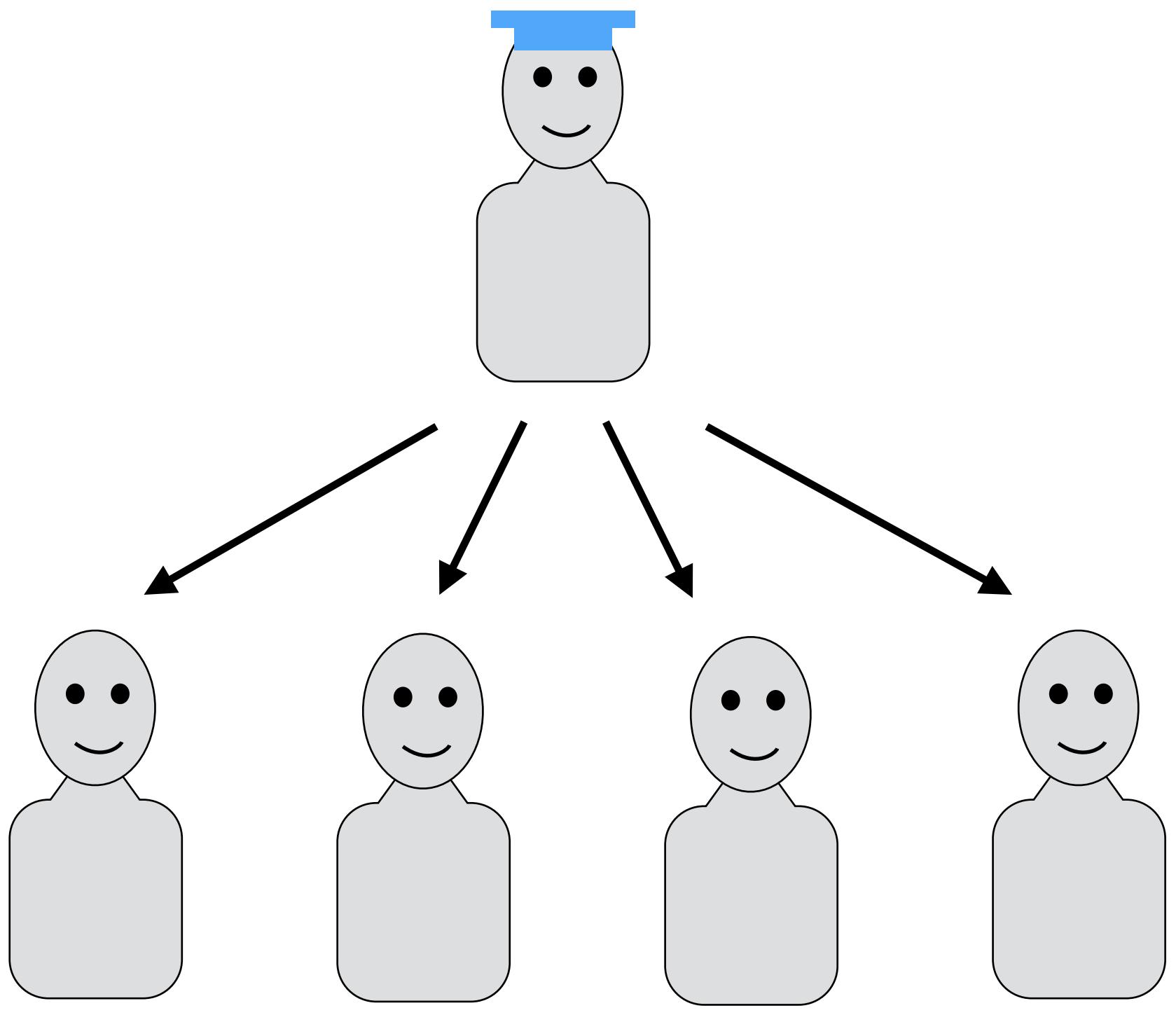


WISCONSIN
UNIVERSITY OF WISCONSIN-MADISON

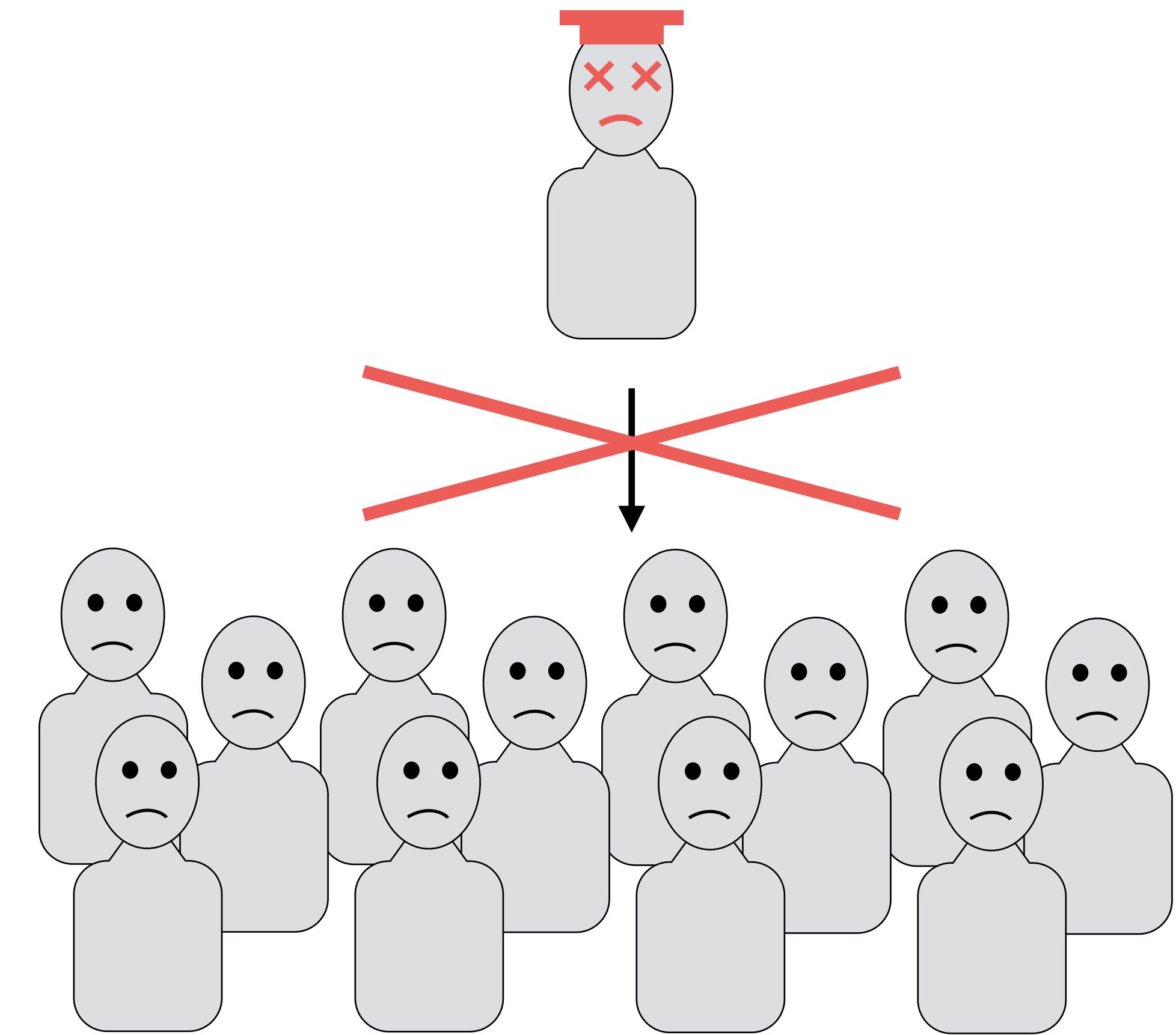
Increasing Demand for Introductory CS Courses



Teachers' Personalized Attention **Does Not Scale**

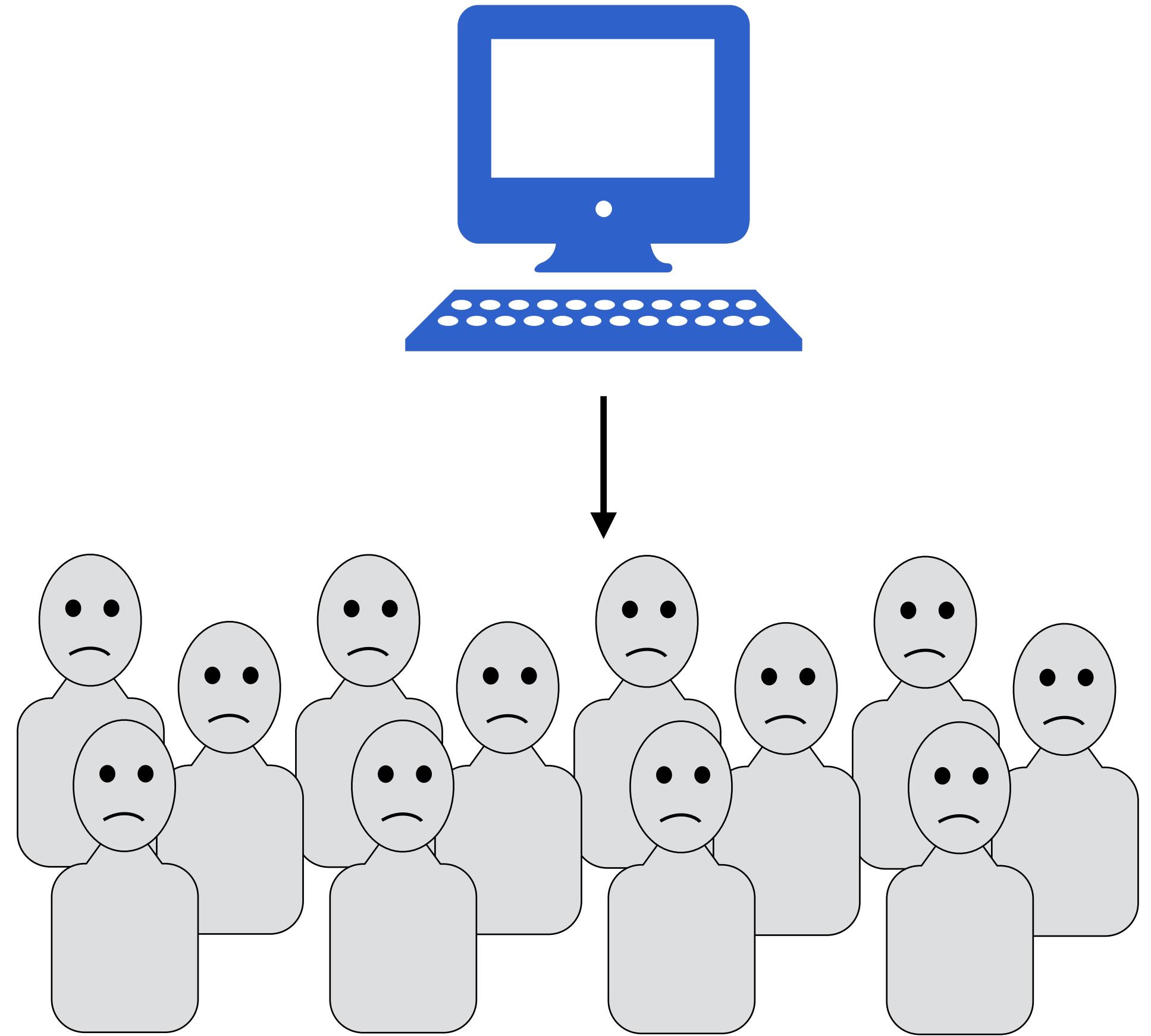


Traditional Classroom



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(
```



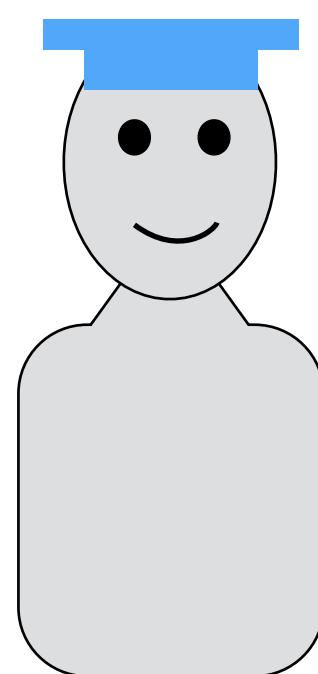
e(..., n-1, ...))

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 + 1 + 2 + 3 + 4 + 5  
o 15 # 0 + 1 + 2 + 3 + 4 + 5
```

Python 2.7

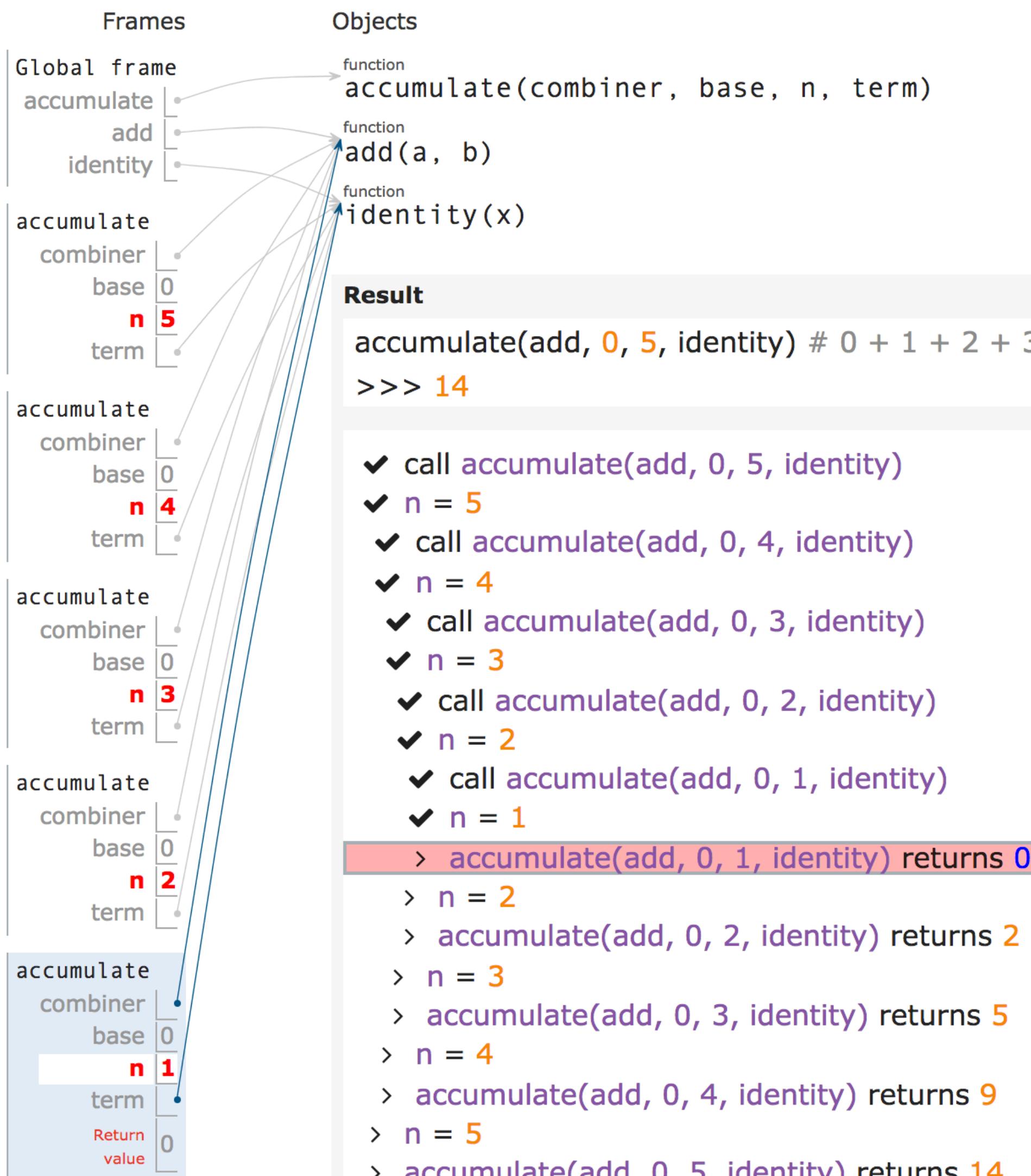
```

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 >



TraceDiff

Python 2.7

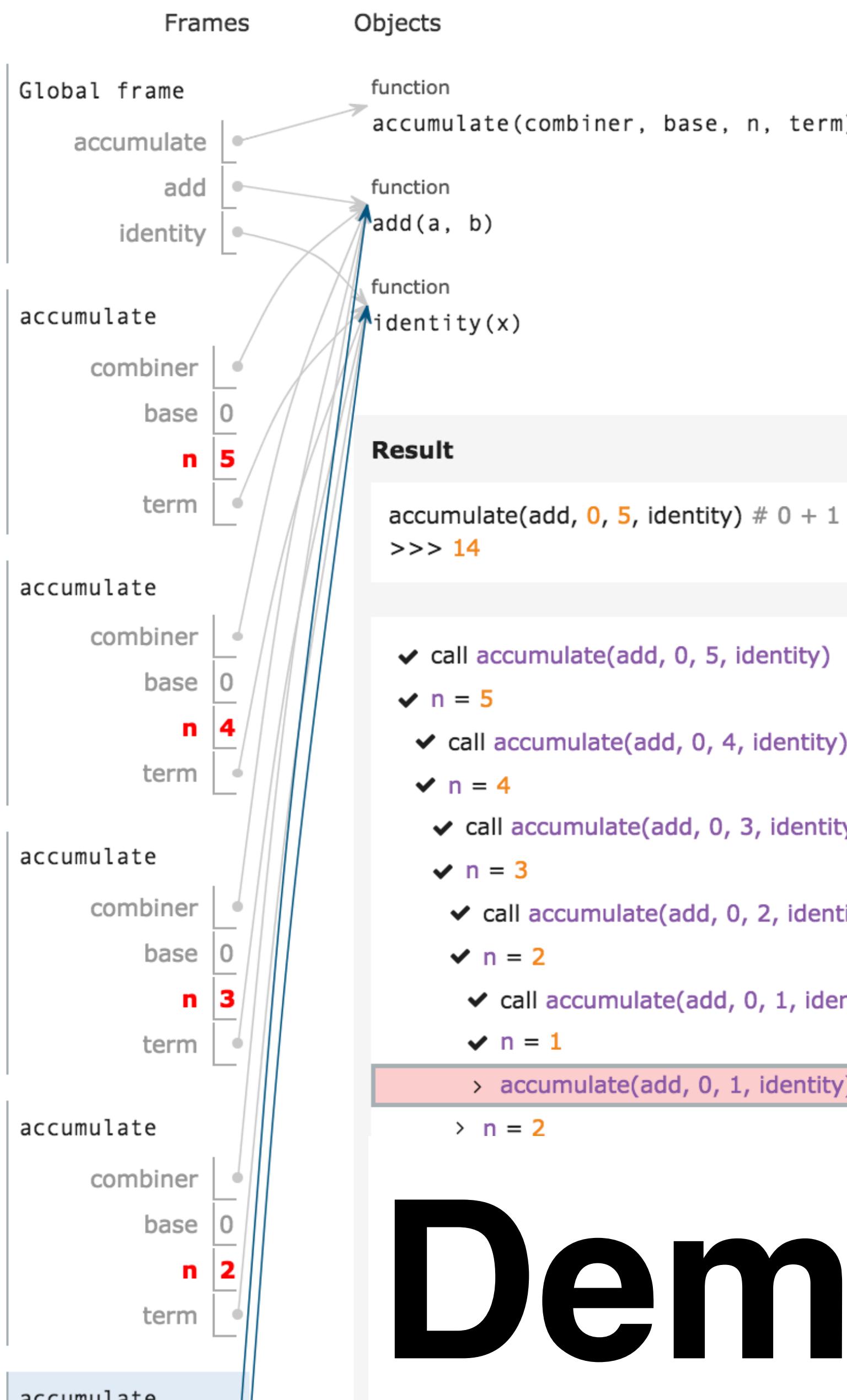
```
1 def accumulate(combiner, base, n, term):  
2     if n==1:  
3         return base  
4     else:  
5         return combiner(term(n), accumulate(  
6  
7 def add(a, b):  
8     return a + b  
9  
10 def identity(x):  
11    return x  
12  
13 accumulate(add, 0, 5, identity)
```

green arrow: line that has just executed

red arrow: next line to execute

< Back Step 32 of 48

Forward >



<https://ryosuzuki.github.io/trace-diff/?type=accumulate&id=10> rns 6

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

1. Highlight Differences

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
```

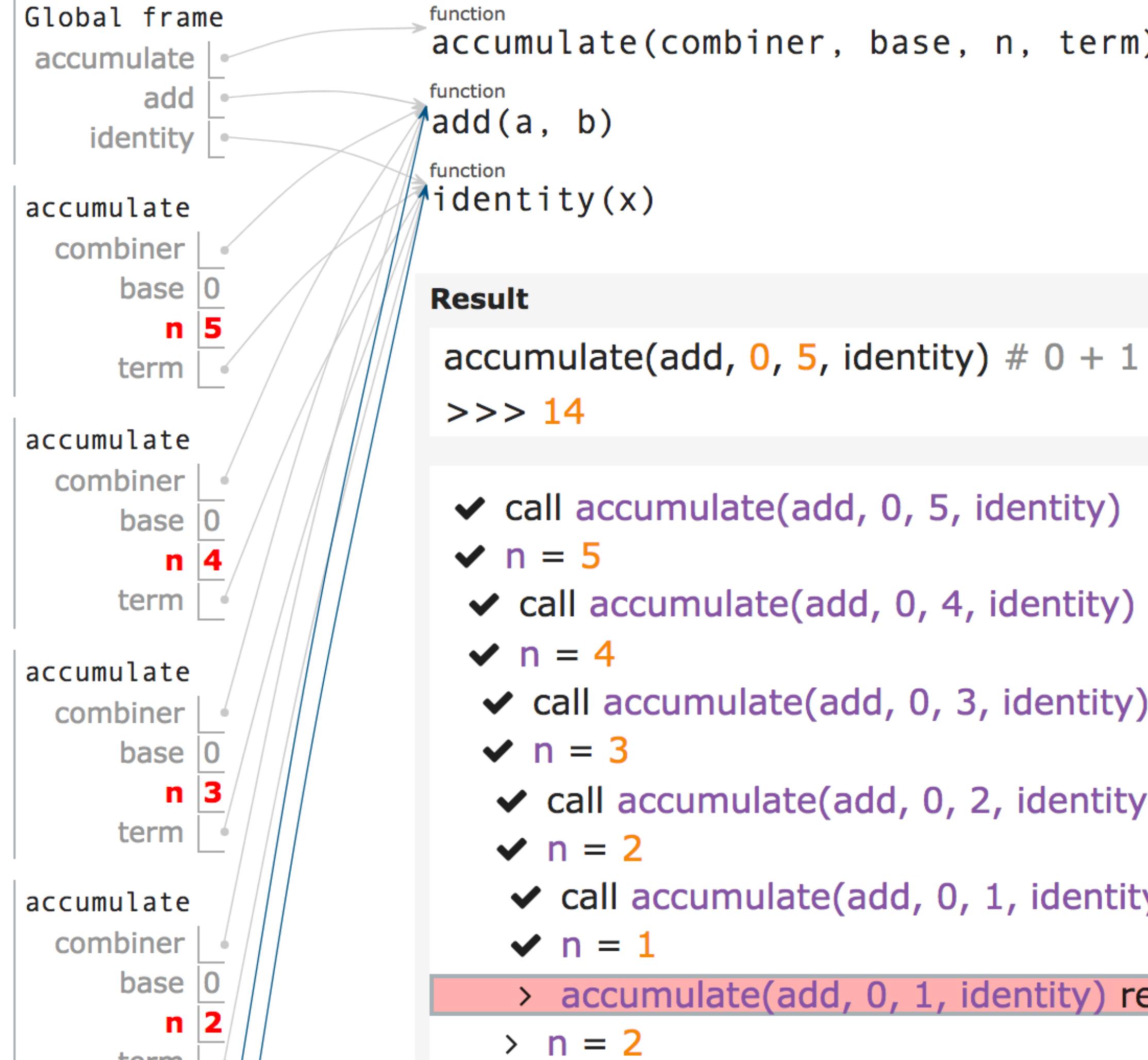
▶ line that has just executed

▶ next line to execute

< Back Step 32 of 48 Forward >

Frames

Objects



Expected

```
accumulate(ac
>>> 15
```

- ✓ call accumu
- ✓ n = 5
- ✓ call accumu
- ✓ n = 4
- ✓ call accumu
- ✓ n = 3
- ✓ call accu
- ✓ n = 2
- ✓ call acc
- ✓ n = 1

> call accu
> n = 0
cum
1
umu
2
mul

2. Interactively Explore

Return

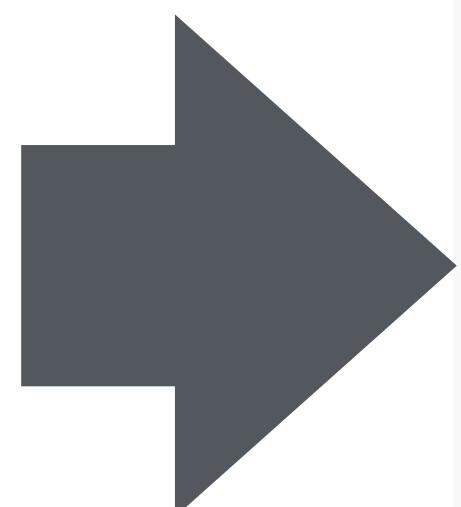
> n = 5

> n = 3

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
```

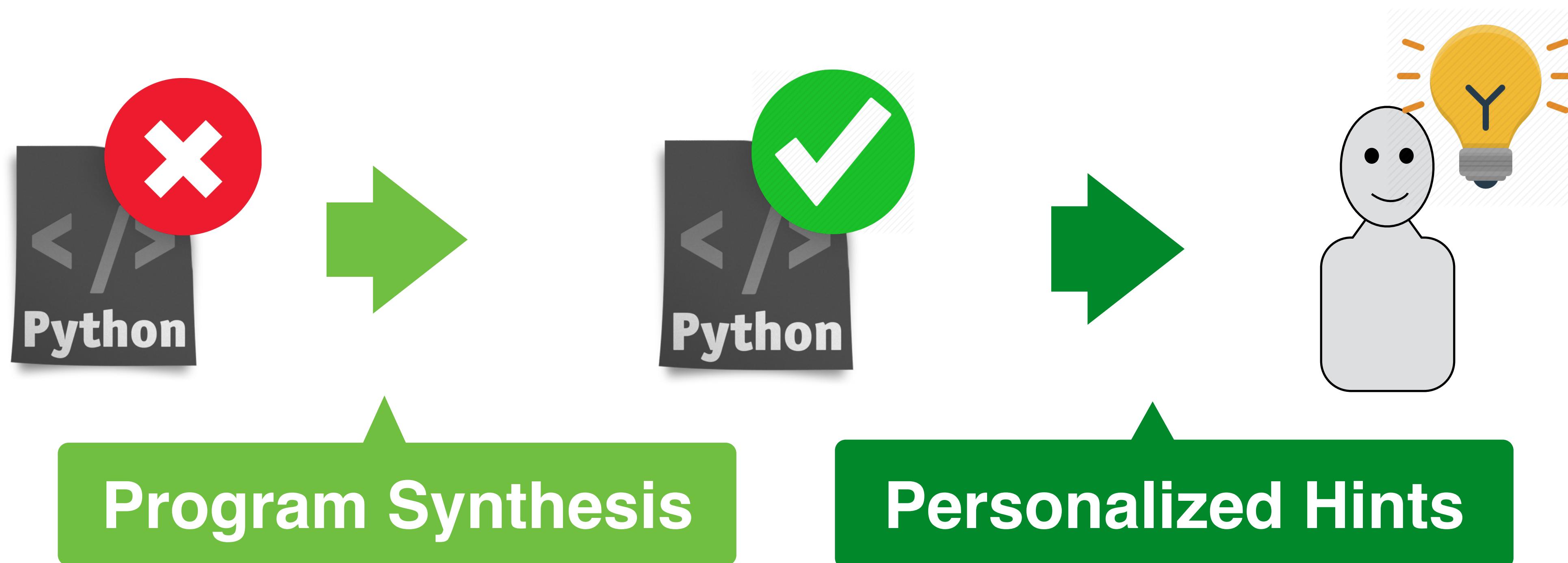
- ✓ call `accumulate(add, 0, 5, identity)`
- ✓ `total = base`
- > `total = add(1, 1)`
- > `total = add(2, 2)`
- > `total = add(3, 3)`
- > `total = add(4, 4)`
- > `total = add(5, 5)`
- > `accumulate(add, 0, 5, identity)` returns `total`

3. Abstract Values Into Expressions

Motivation

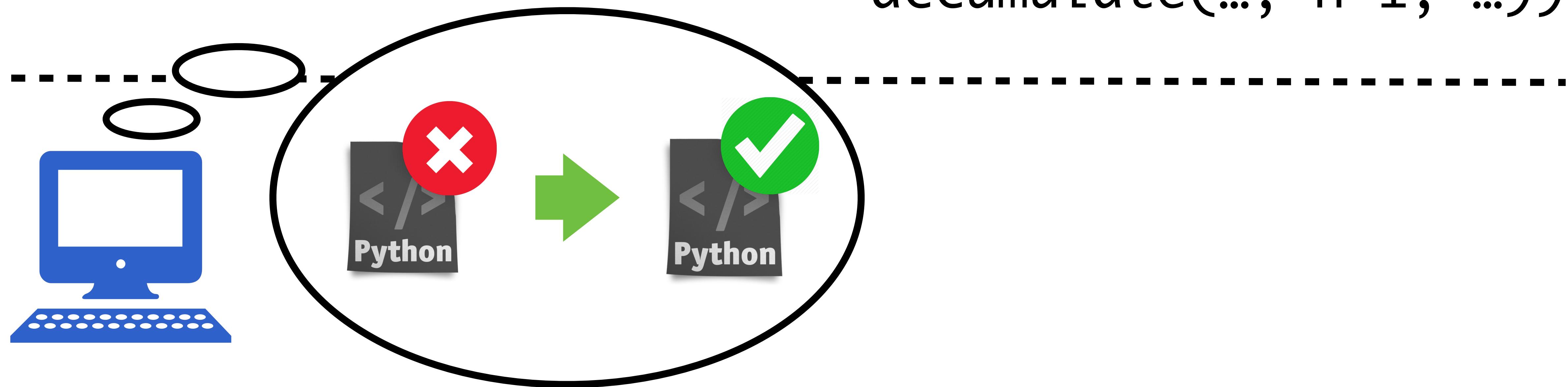
Program Synthesis Feedback

(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

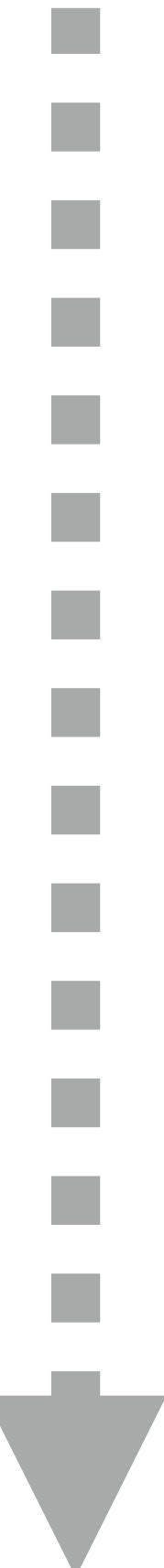
In line 2, change $n = 1$ to $n = 0$

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

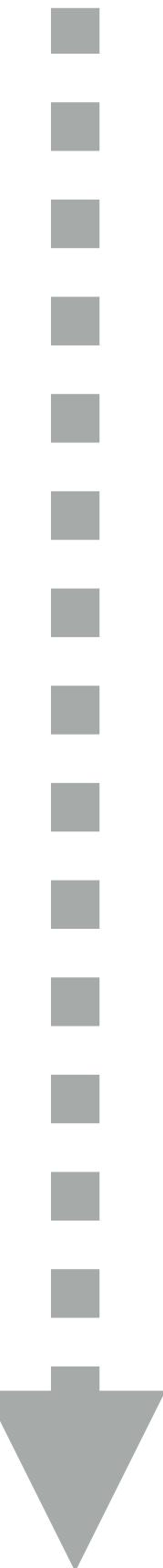
Line 2 needs to be changed
In line 2, check “n”

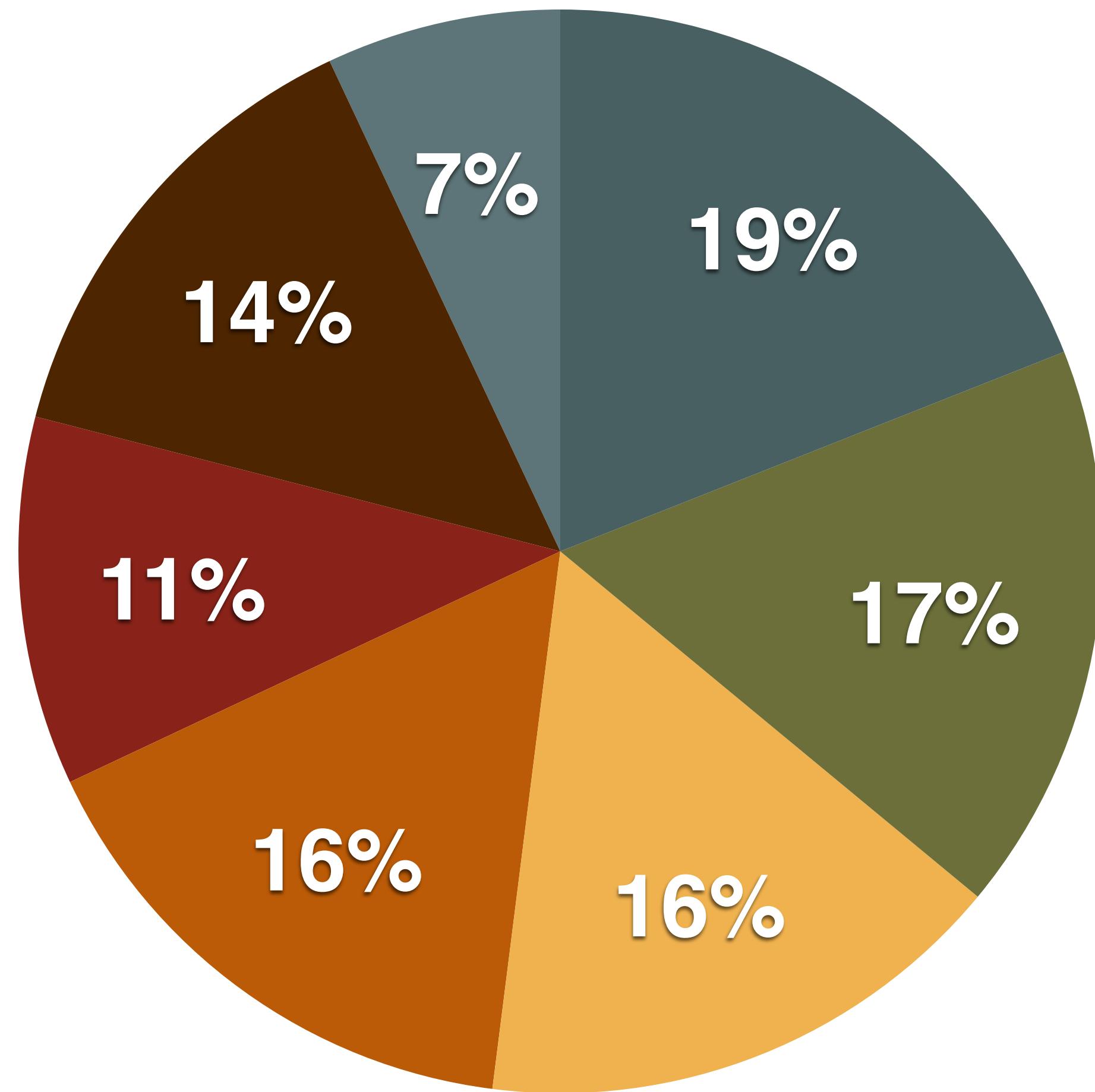
High-level
Hints



Bottom-out

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

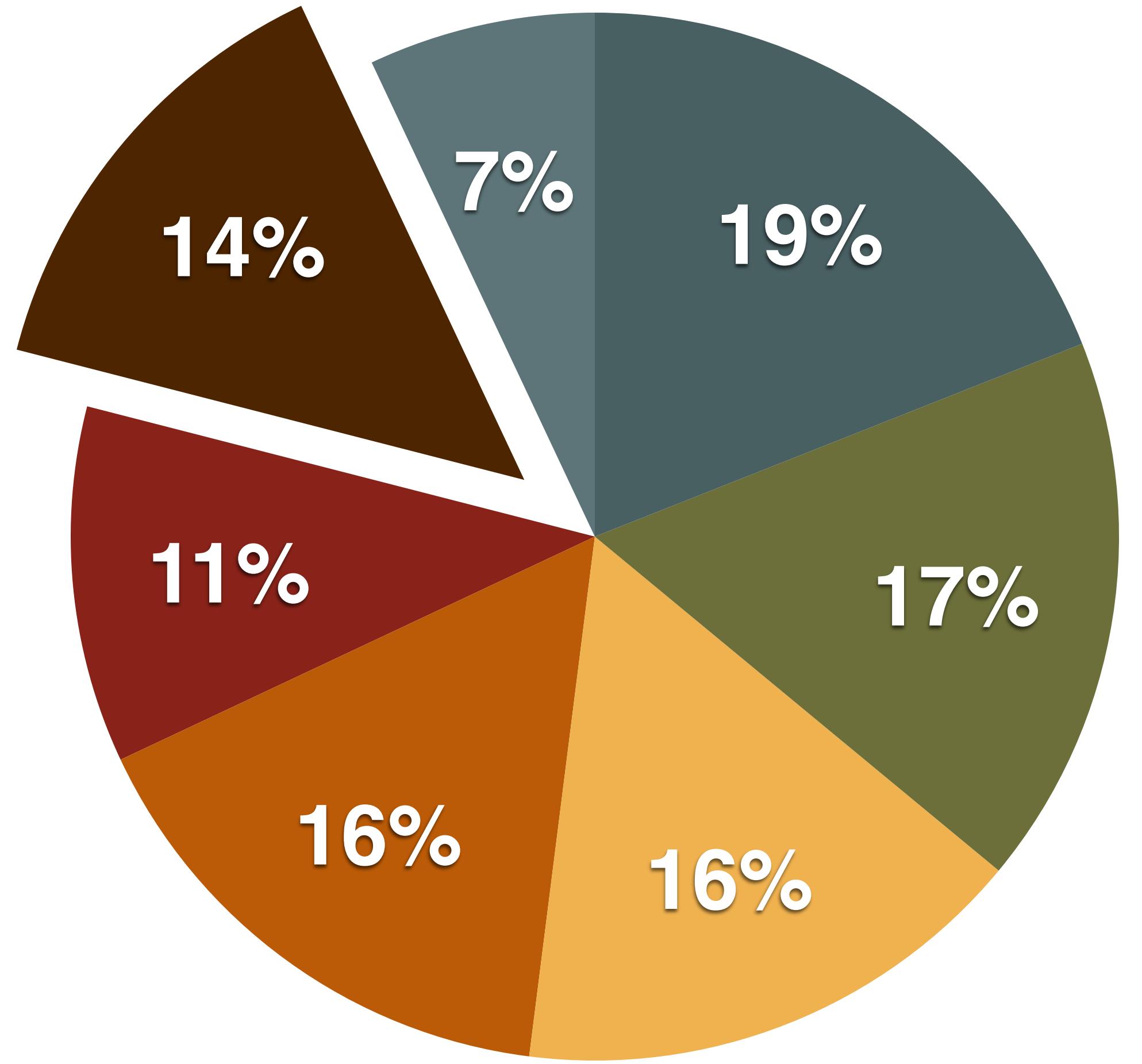




- Diagnose the cause of error: 19%
- Suggest to run code in PythonTutor: 17%
- Explain incorrect behavior: 16%
- Remind relevant resources: 16%
- Provide example usage: 11%
- Point out location: 14%
- Suggest concrete fix: 7%

132 posts on piazza

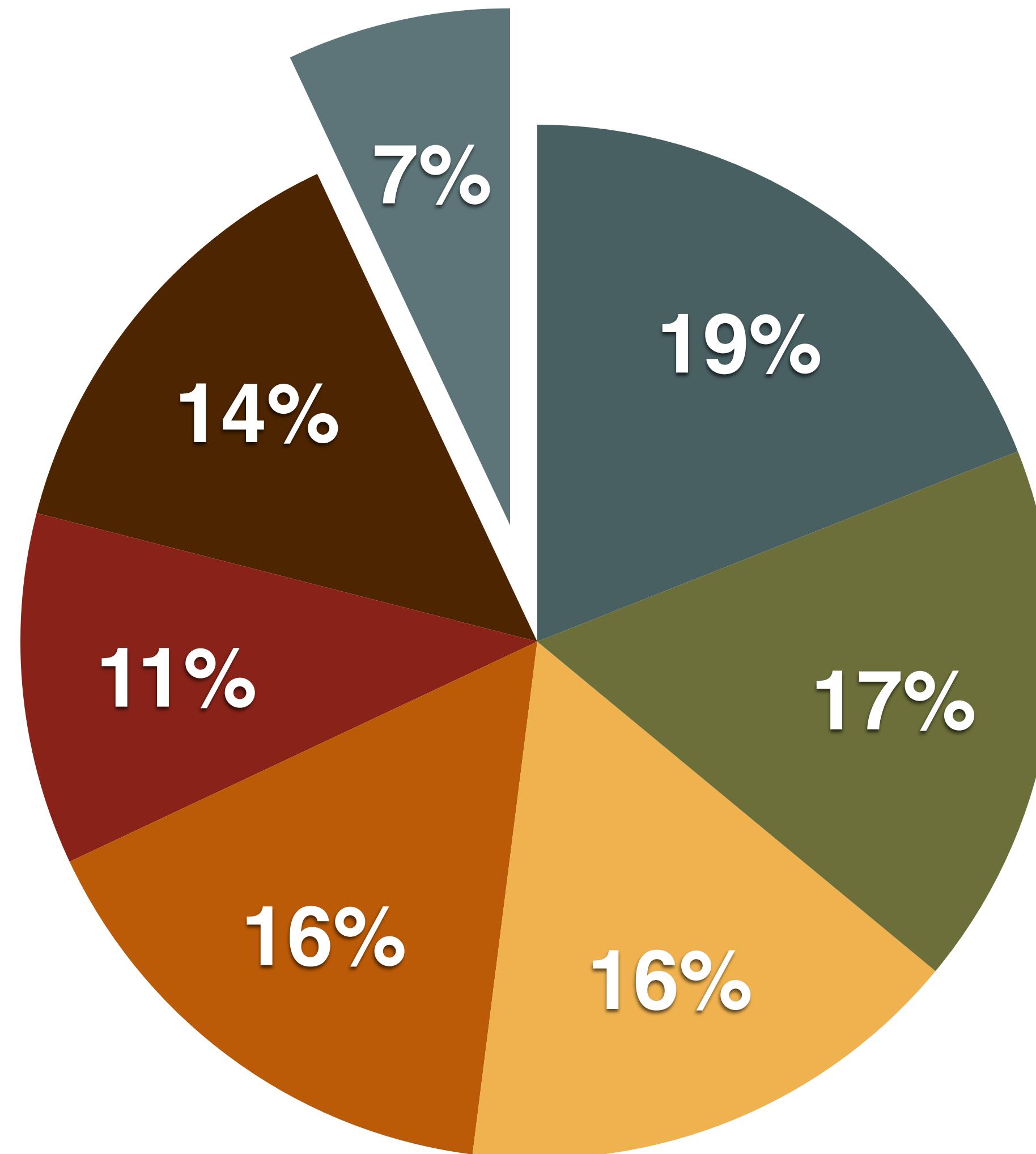




- Diagnose the cause of error: 19%
- Suggest to run code in PythonTutor: 17%
- Explain incorrect behavior: 16%
- Remind relevant resources: 16%
- Provide example usage: 11%
- **Point out location: 14%**
- Suggest concrete fix: 7%

132 posts on piazza

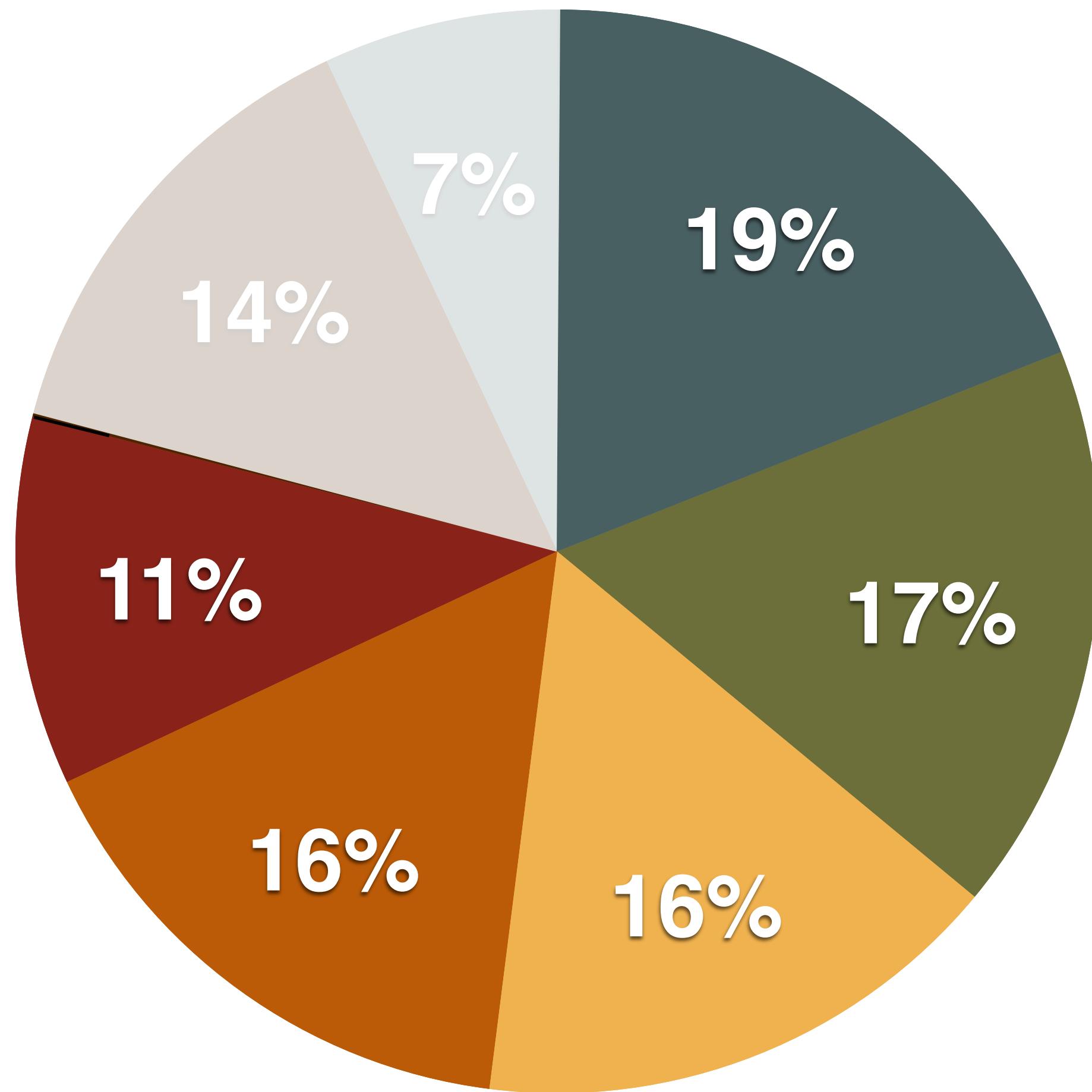




132 posts on piazza

- 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%**





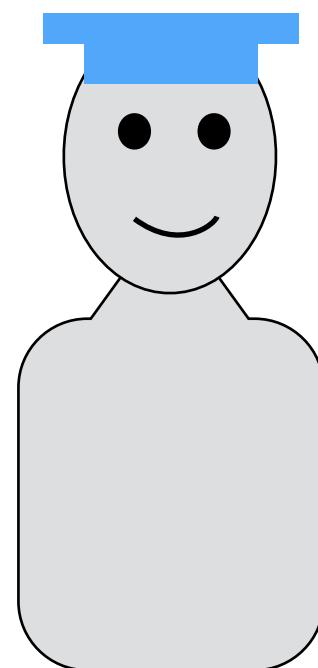
- **Diagnose the cause of error: 19%**
- **Suggest to run in PythonTutor: 17%**
- **Explain incorrect behavior: 16%**
- **Remind relevant resources: 16%**
- **Provide example usage: 11%**
- Point out location: 14%
- Suggest concrete fix: 7%

132 posts on piazza



Code

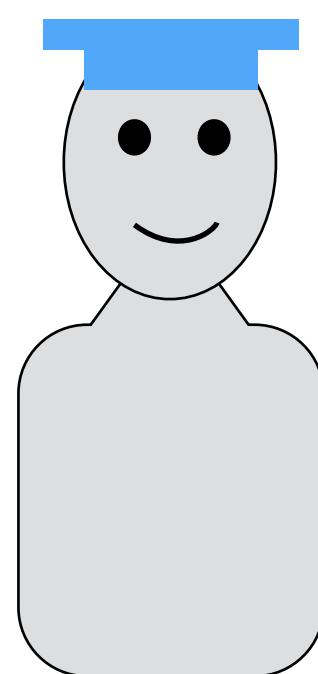
```
def accumulate(combiner, base, n, term):  
    if n == 1:  
        if n == 0:  
            return base  
    else:  
        return combiner(term(n),  
                         accumulate(..., n-1, ...))
```



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)

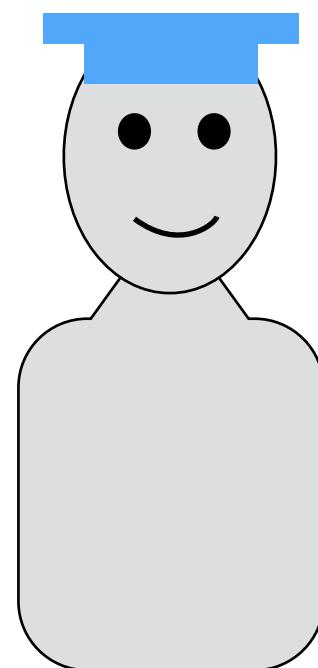
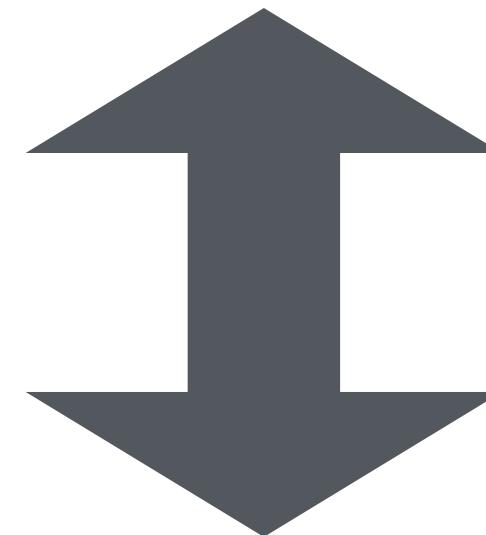
x 14 # 0 + 0 + 2 + 3 + 4 + 5

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 + + 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



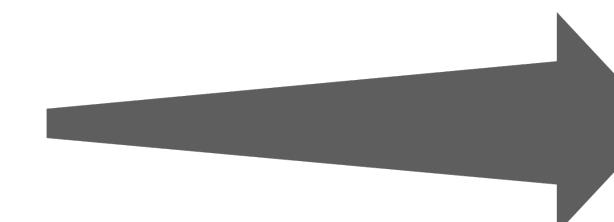
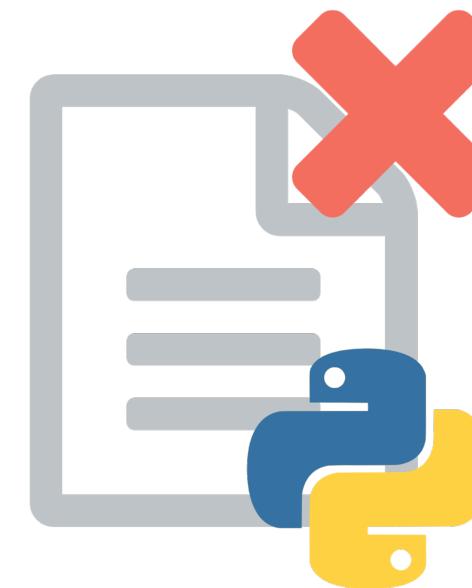
accumulate(add, 0, 5, identity)

x 14 # 0 + + 2 + 3 + 4 + 5

o 15 # 0 + 1 + 2 + 3 + 4 + 5

Implementation

Incorrect submission

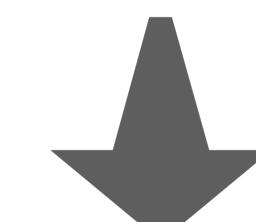


Result

```
accumulate(add, 0, 5, identity) # 0 +  
>>> 10  
  
✓ call accumulate(add, 0, 5, identity)  
✓ total = 0  
> total = 2
```

① Synthesis

Closest correct program



② Execute & Record Trace

③

Filter & Highlight Trace Difference

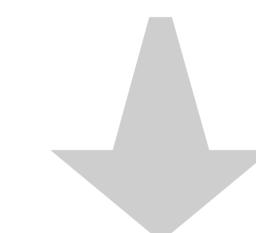
Expected

```
accumulate(add, 0, 5, identity) # 0 +  
>>> 15  
  
✓ call accumulate(add, 0, 5, identity)  
✓ total = 0  
> total = 1
```

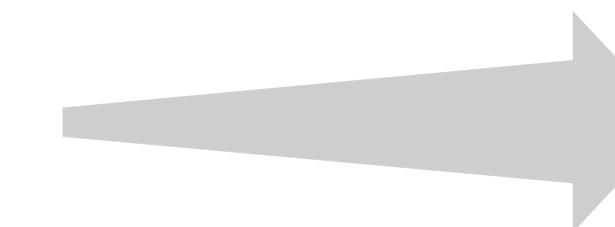
Incorrect submission

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
```

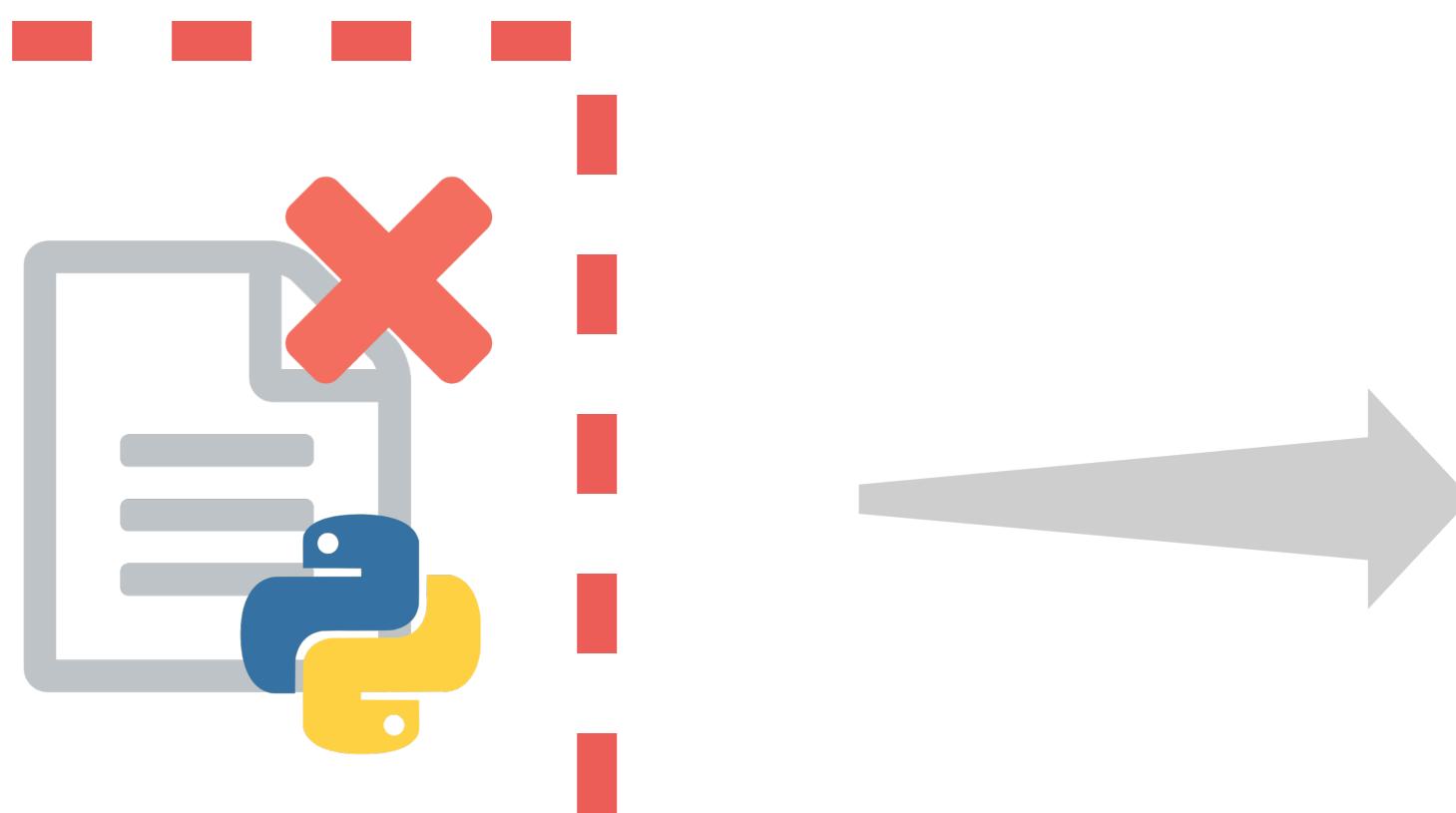
③

Filter & Highlight Trace Difference

Expected

```
accumulate(add, 0, 5, identity) # 0 + :  
>>> 15  
  
✓ call accumulate(add, 0, 5, identity)  
✓ total = 0  
> total = 1
```

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)

Learn code transformation
from examples

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

Insert

$\langle \text{exp} \rangle * \langle \text{name} \rangle \rightarrow \langle \text{exp} \rangle * \boxed{\text{term}(\langle \text{name} \rangle)}$

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

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

Learn code transformation
from examples

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

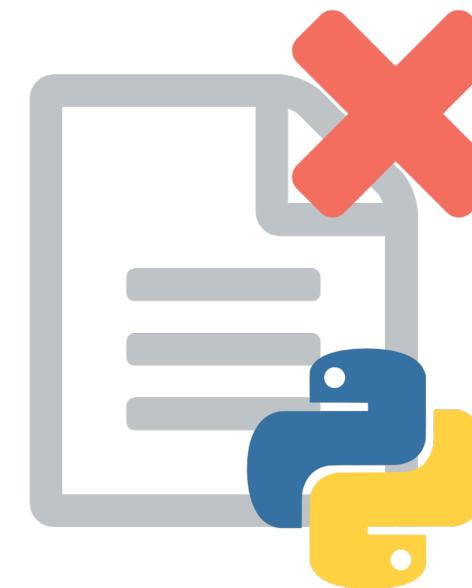
```
def product(n, term):
    total, k = 1, 1
    while k<=n:
        - total = total*k
        + total = total*term(k)
        k = k+1
    return total
```

Insert

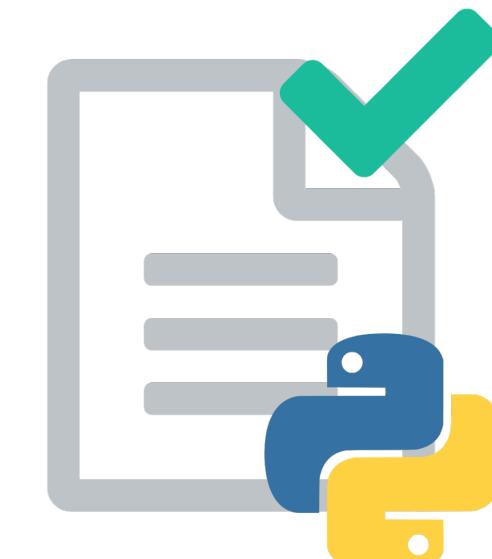
$\langle \text{exp} \rangle * \langle \text{name} \rangle \rightarrow \langle \text{exp} \rangle * \boxed{\text{term}(\langle \text{name} \rangle)}$

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

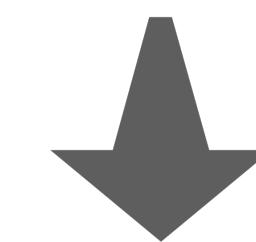
Incorrect submission



Closest correct program



① Synthesis



② Execute & Record Trace



Result

```
accumulate(add, 0, 5, identity) # 0 +  
>>> 10  
  
✓ call accumulate(add, 0, 5, identity)  
✓ total = 0  
> total = 2
```

③

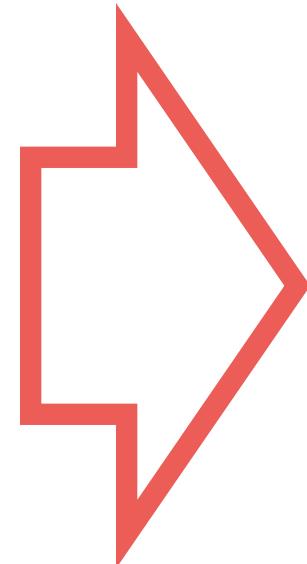
Filter & Highlight Trace Difference

Expected

```
accumulate(add, 0, 5, identity) # 0 + :  
>>> 15  
  
✓ call accumulate(add, 0, 5, identity)  
✓ total = 0  
> total = 1
```

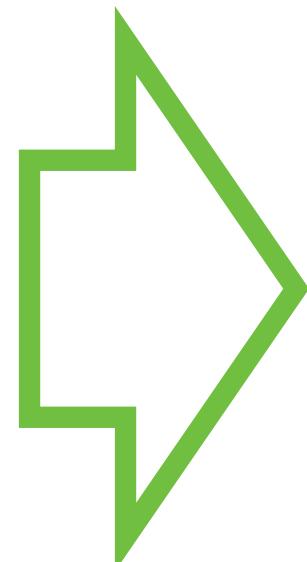
→ def accumulate(n):
 total = 0
 i = 1
 while i <= n:
 k = square(i)
 total = total + k
 i = i + 1

n: 3



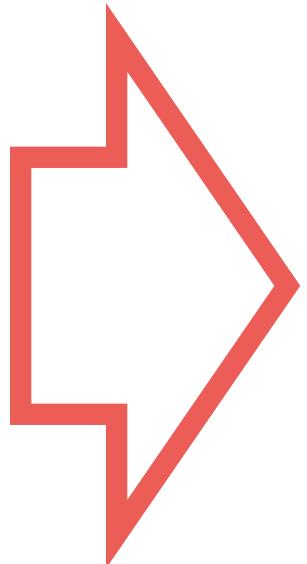
→ def accumulate(n):
 total = 10
 i = 1
 while i <= n:
 k = square(i)
 total = total + i
 i = i + 1

n: 3



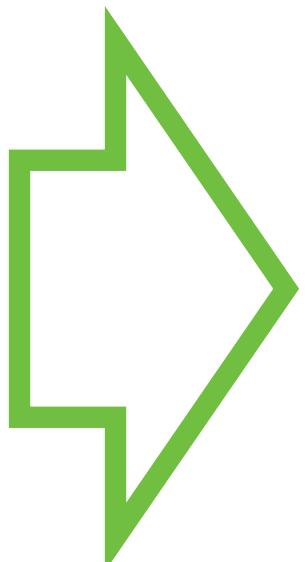
```
def accumulate(n):  
    total = 0  
    i = 1  
    while i <= n:  
        k = square(i)  
        total = total + k  
        i = i + 1
```

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



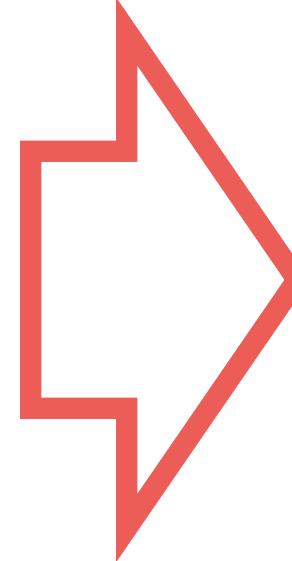
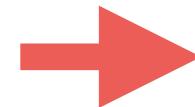
```
def accumulate(n):  
    total = 0  
    → i = 1  
    while i <= n:  
        k = square(i)  
        total = total + k  
        i = i + 1
```

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
```



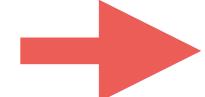
```
n: 3  
total: 0  
i: 1  
k: 1
```

```
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
```



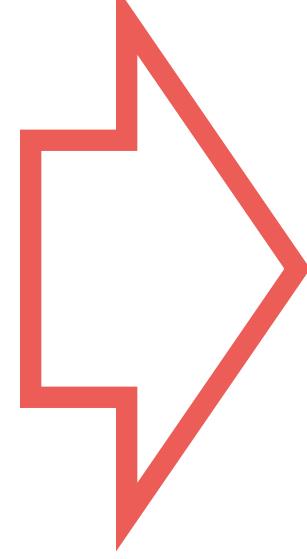
n: 3
total: 0, 1
i: 1
k: 1

```
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
```



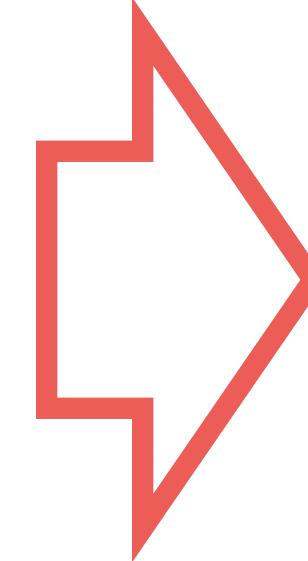
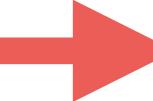
n: 3
total: 0, 1
i: 1, 2
k: 1

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



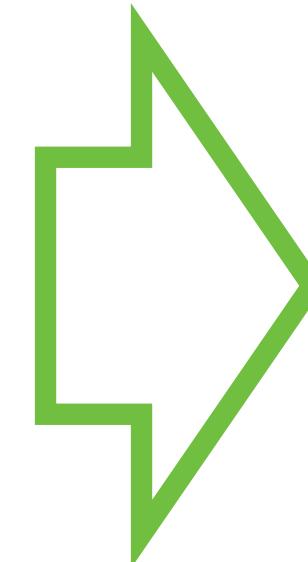
n: 3
total: 10, 11
i: 1, 2
k: 1

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



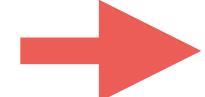
```
n: 3  
total: 0, 1  
i: 1, 2  
k: 1, 4
```

```
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
```



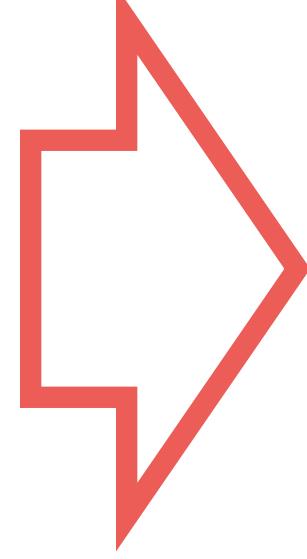
n: 3
total: 0, 1, 5
i: 1, 2
k: 1, 4

```
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
```



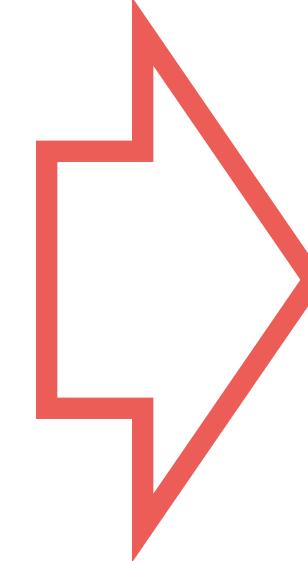
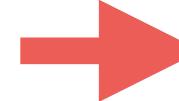
n: 3
total: 0, 1, 5
i: 1, 2, 3
k: 1, 4

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



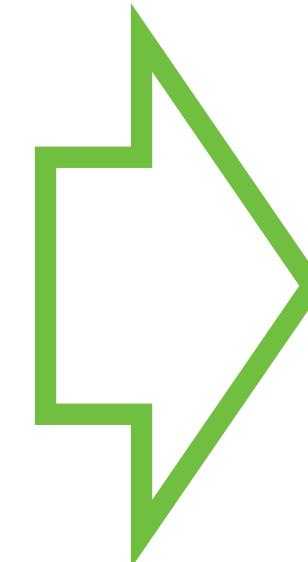
n: 3
total: 10, 11, 15
i: 1, 2, 3
k: 1, 4

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



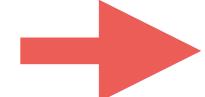
n: 3
total: 0, 1, 5
i: 1, 2, 3
k: 1, 4, 9

```
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
```



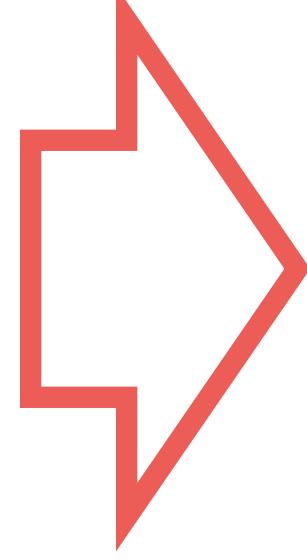
n: 3
total: 0, 1, 5, 14
i: 1, 2, 3
k: 1, 4, 9

```
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
```



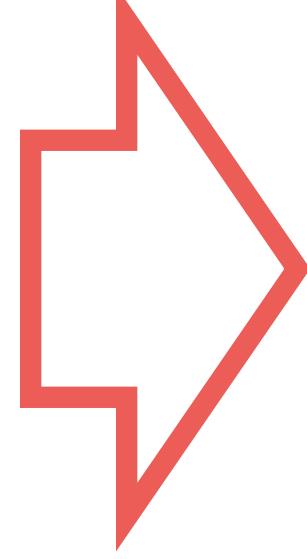
n: 3
total: 0, 1, 5, 14
i: 1, 2, 3, 4
k: 1, 4, 9

```
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

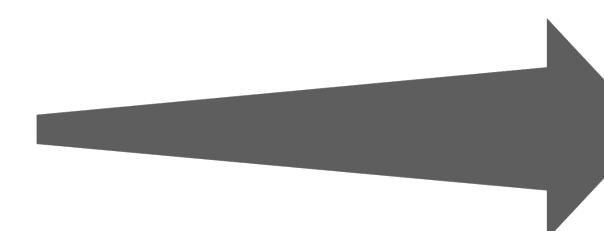
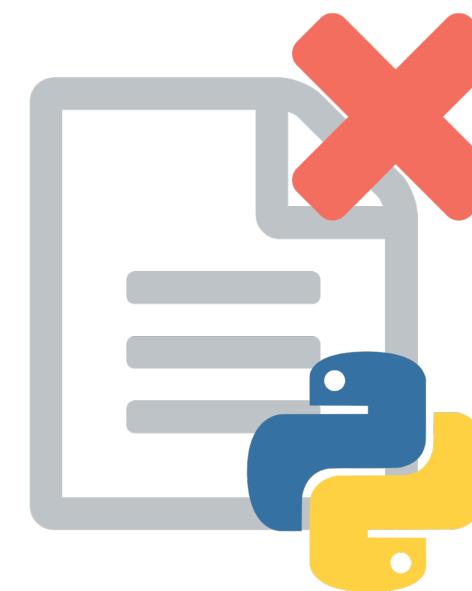
Result

```
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)
```

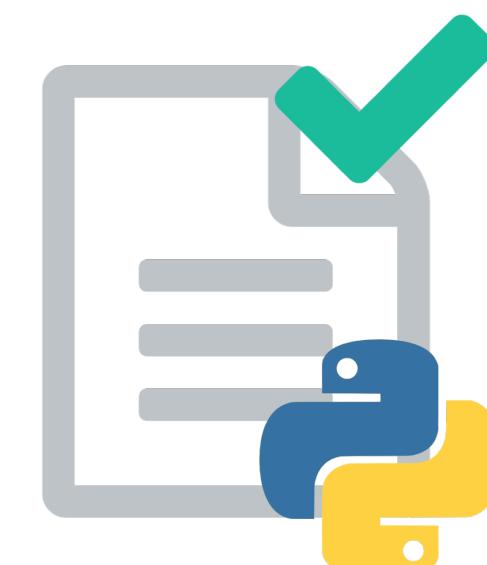
: + 24 lines

Incorrect submission

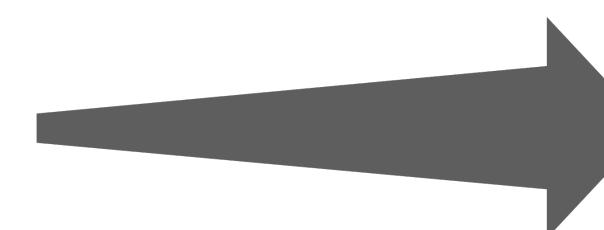


① Synthesis

Closest correct program



② Execute & Record Trace



Result

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

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

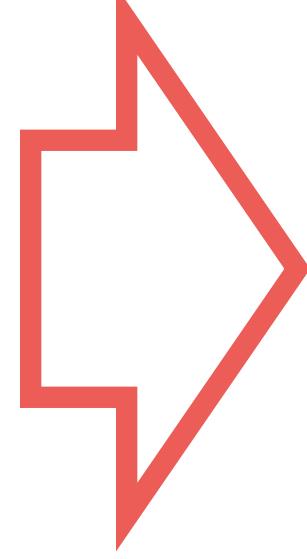
Expected

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

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

A screenshot of a debugger interface showing two execution traces side-by-side. The top trace is labeled 'Result' and the bottom trace is labeled 'Expected'. Both traces show the same initial steps: an 'accumulate' call with arguments 'add', '0', '5', and 'identity', followed by an input of '10'. The results diverge at the final step: the 'Result' trace shows 'total = 2' (highlighted with a pink bar), while the 'Expected' trace shows 'total = 1' (also highlighted with a pink bar). The Python logo is visible at the bottom of both traces.

```
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

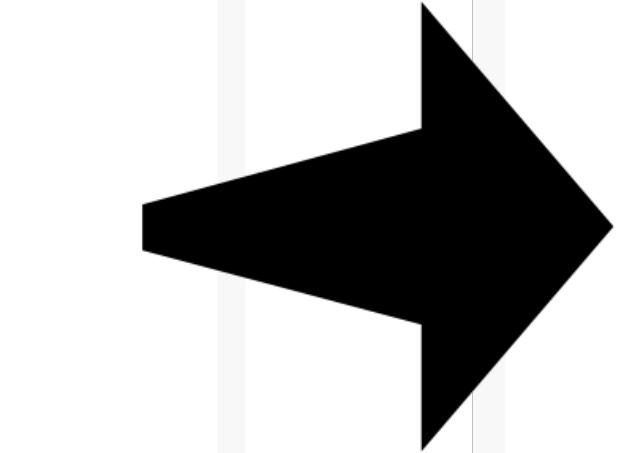
k: 1, 4, 9

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

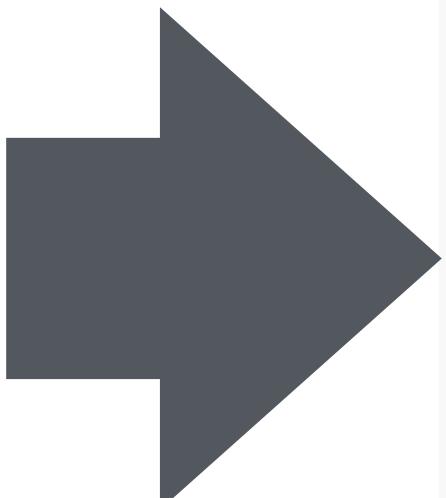
Value Abstraction

Abstract Values into Expressions

Result

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

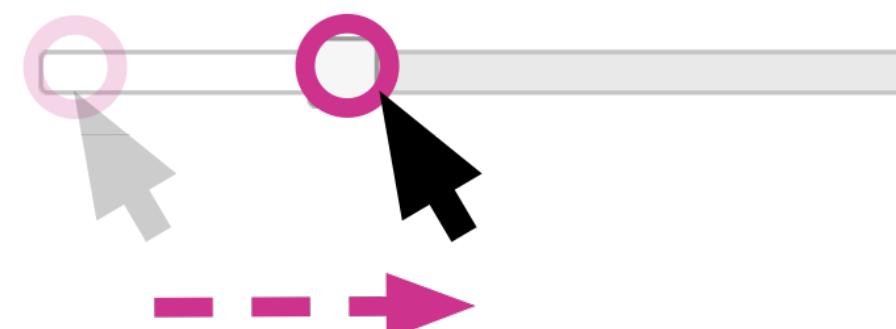
- ✓ call `accumulate(add, 0, 5, identity)`
- ✓ `total = 0`
- › `total = 2`
- › `total = 4`
- › `total = 6`
- › `total = 8`
- › `total = 10`
- › `accumulate(add, 0, 5, identity)` returns `10`



Result

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

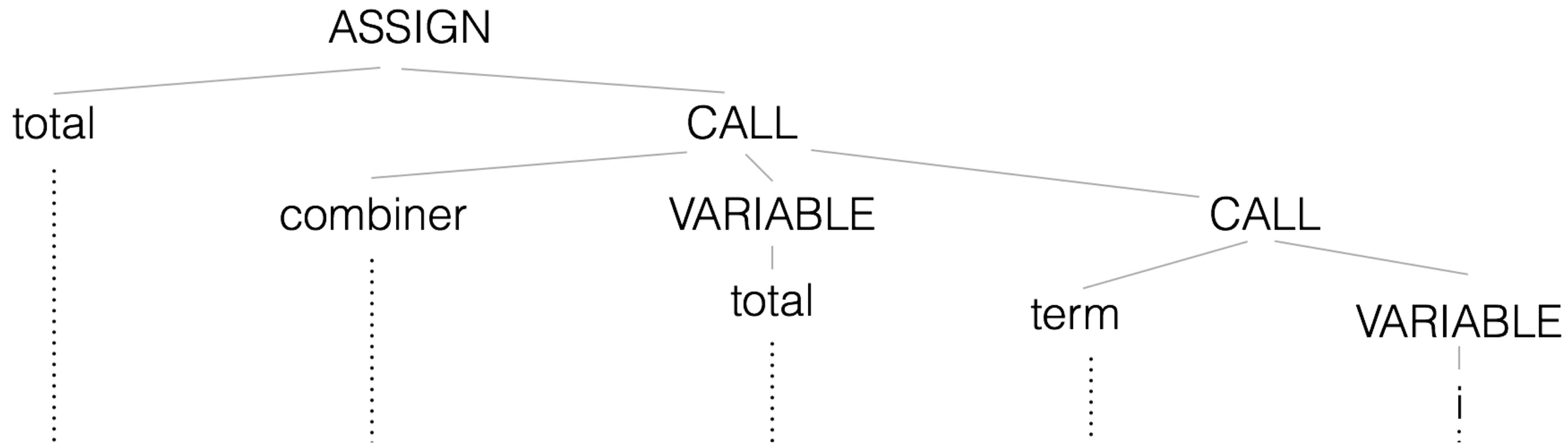
- ✓ call `accumulate(add, 0, 5, identity)`
- ✓ `total = base`
- › `total = add(1, 1)`
- › `total = add(2, 2)`
- › `total = add(3, 3)`
- › `total = add(4, 4)`
- › `total = add(5, 5)`
- › `accumulate(add, 0, 5, identity)` returns `total`



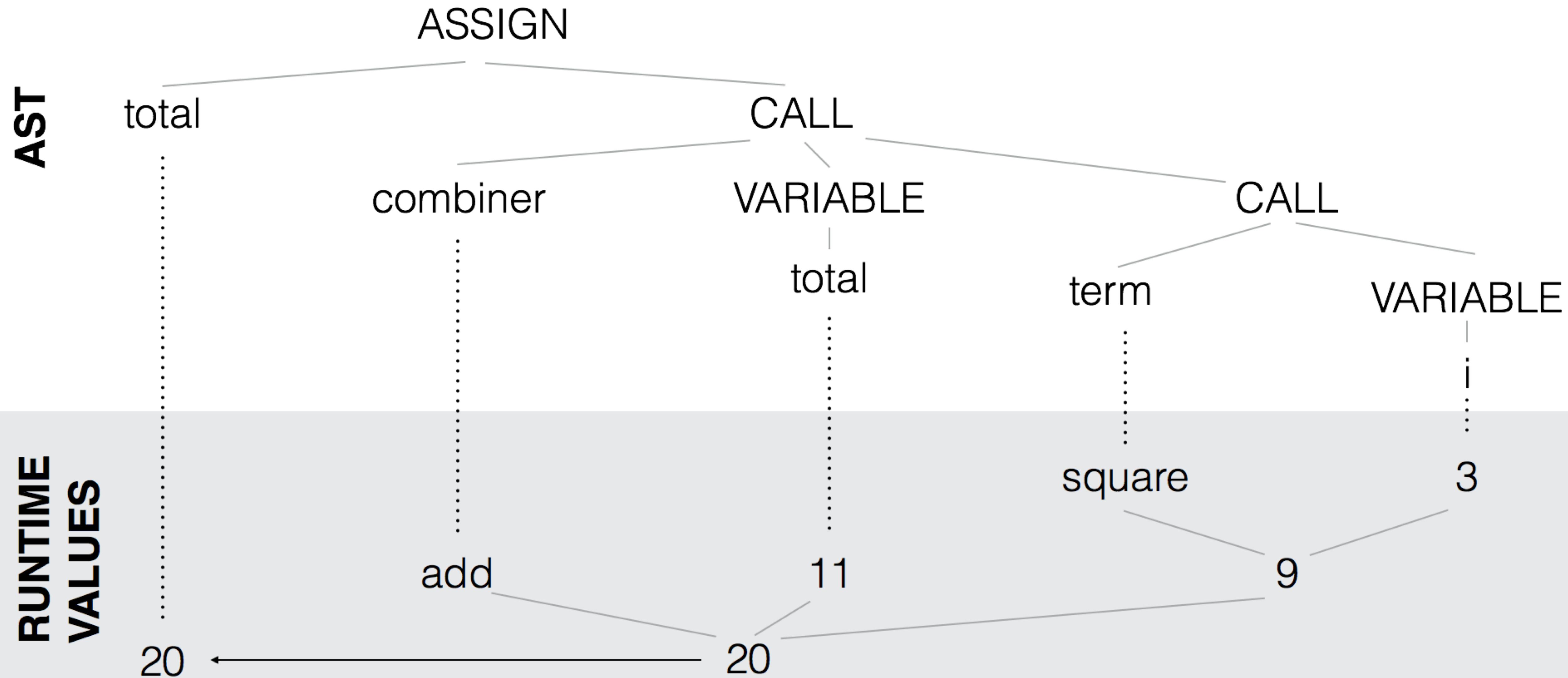
```
total = combiner(i, term(i))
```

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

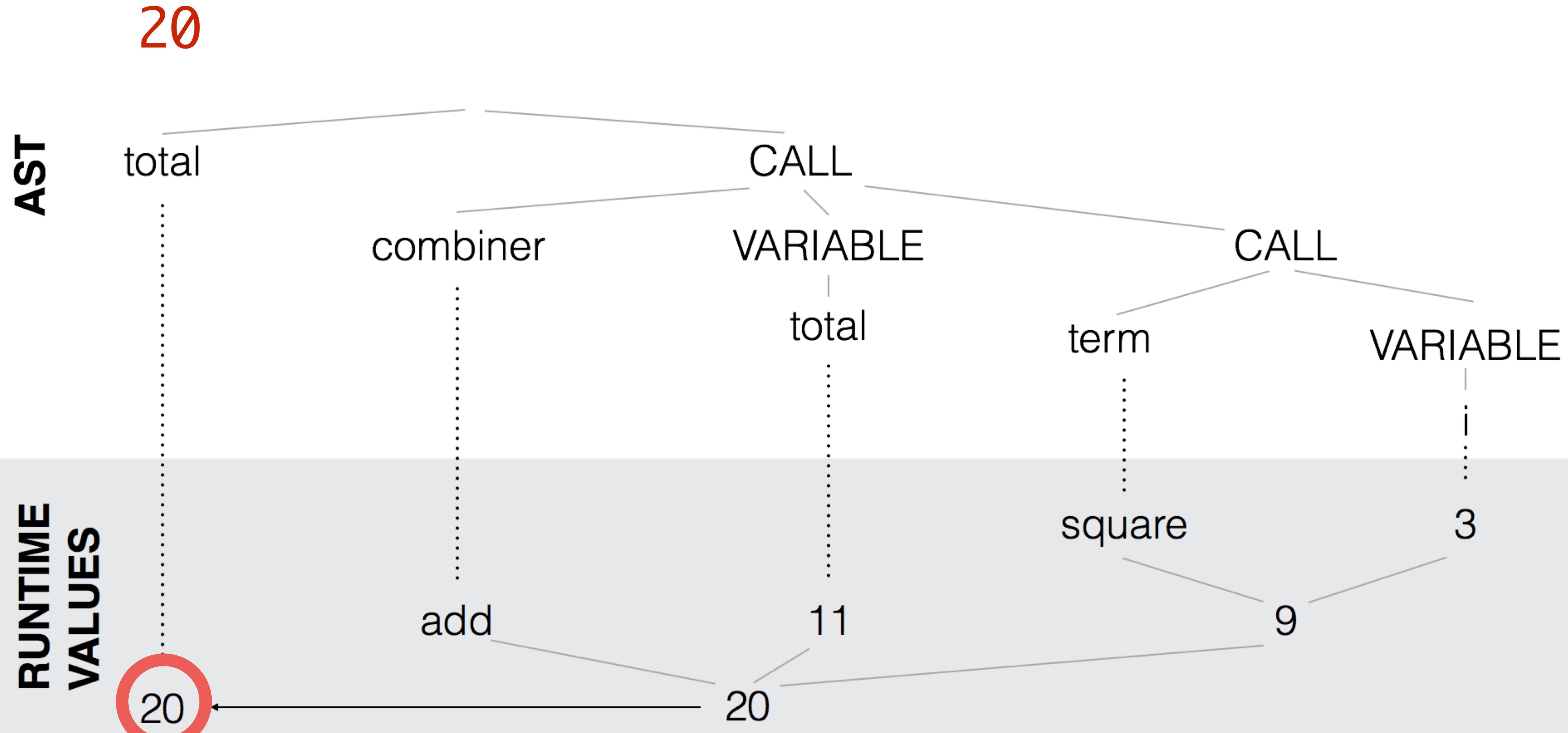
AST



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



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



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

20 → add(11, 9)

AST

total

combiner

CALL

VARIABLE

total

CALL

term

VARIABLE

add

11

9

RUNTIME
VALUES

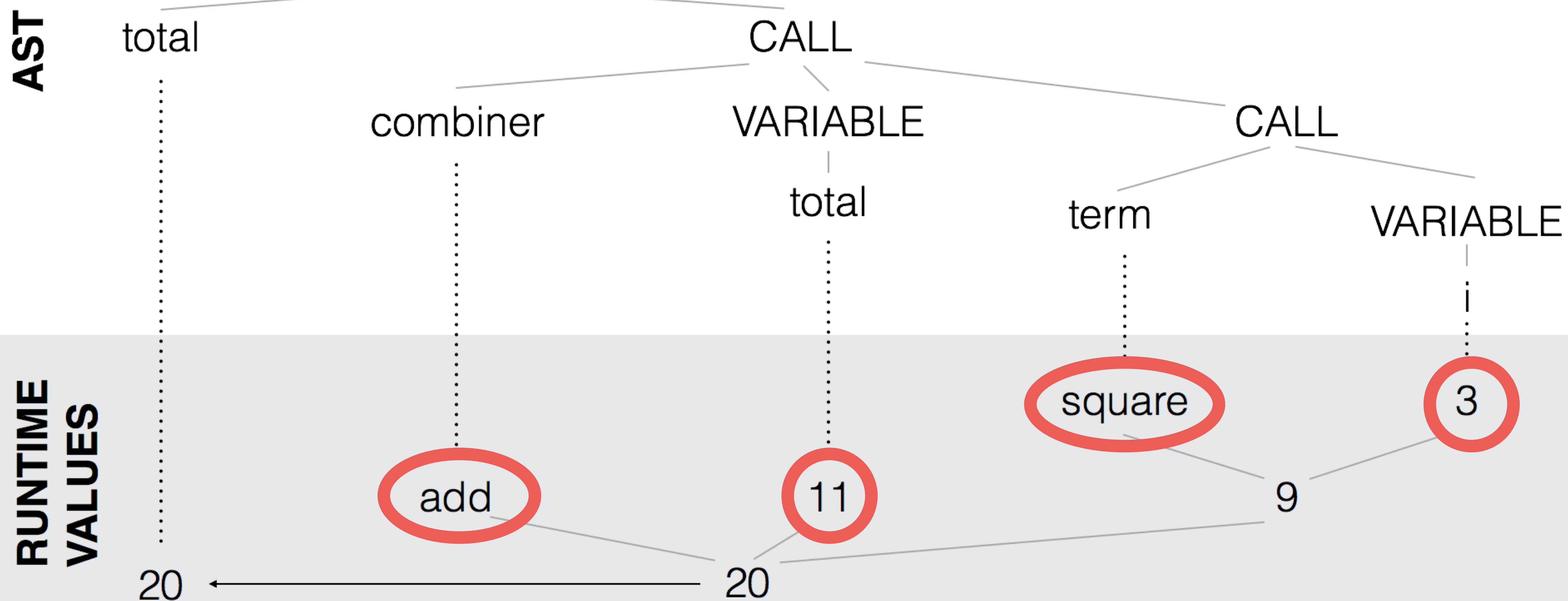
20

20

3

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

20 → add(11, 9) → add(11, square(3)) → ...

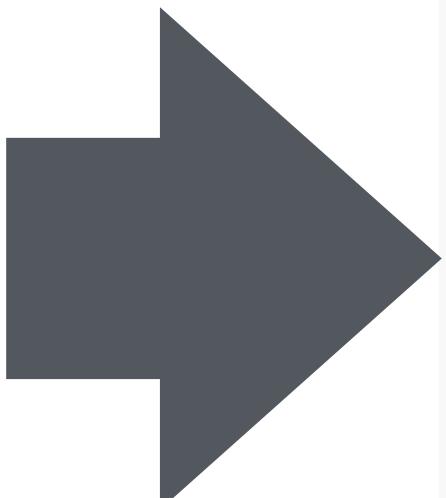


Abstract Values into Expressions

Result

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

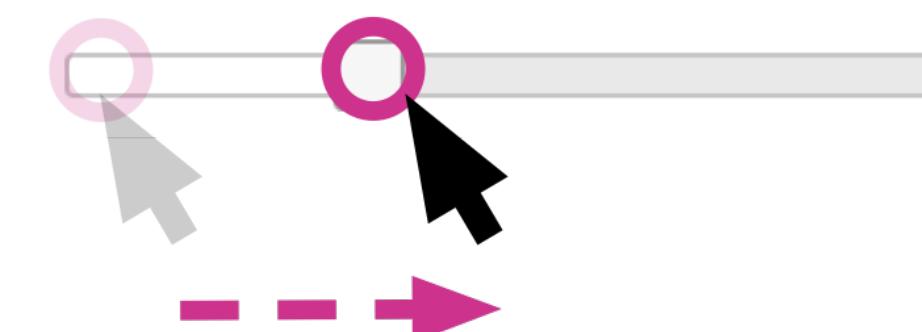
- ✓ call `accumulate(add, 0, 5, identity)`
- ✓ `total = 0`
- › `total = 2`
- › `total = 4`
- › `total = 6`
- › `total = 8`
- › `total = 10`
- › `accumulate(add, 0, 5, identity)` returns `10`



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:

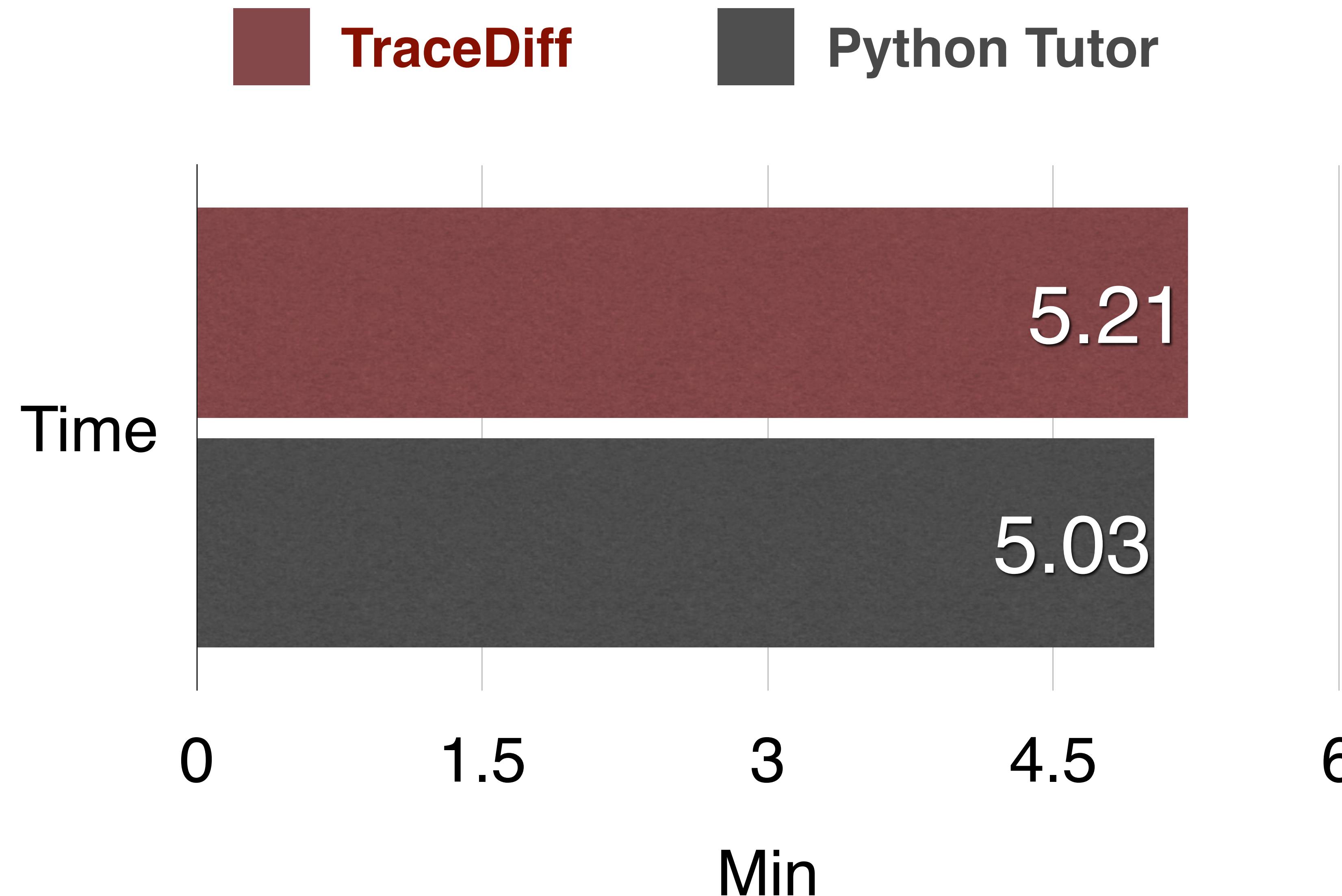
- **Identify** the bug
- **Fix** the bug

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

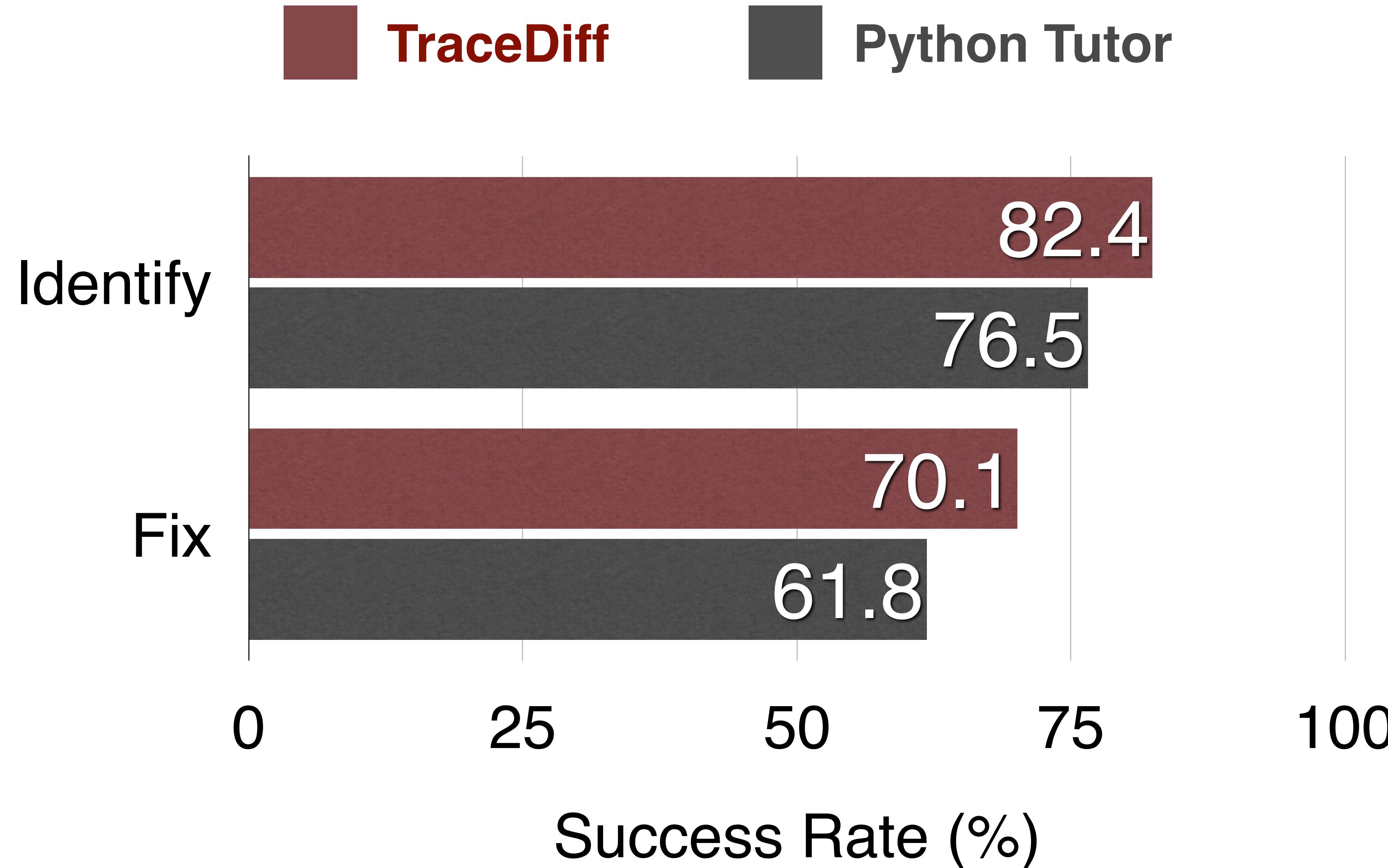
RQ2: Can TraceDiff help students identify and fix **more bugs** than Python Tutor?

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**?



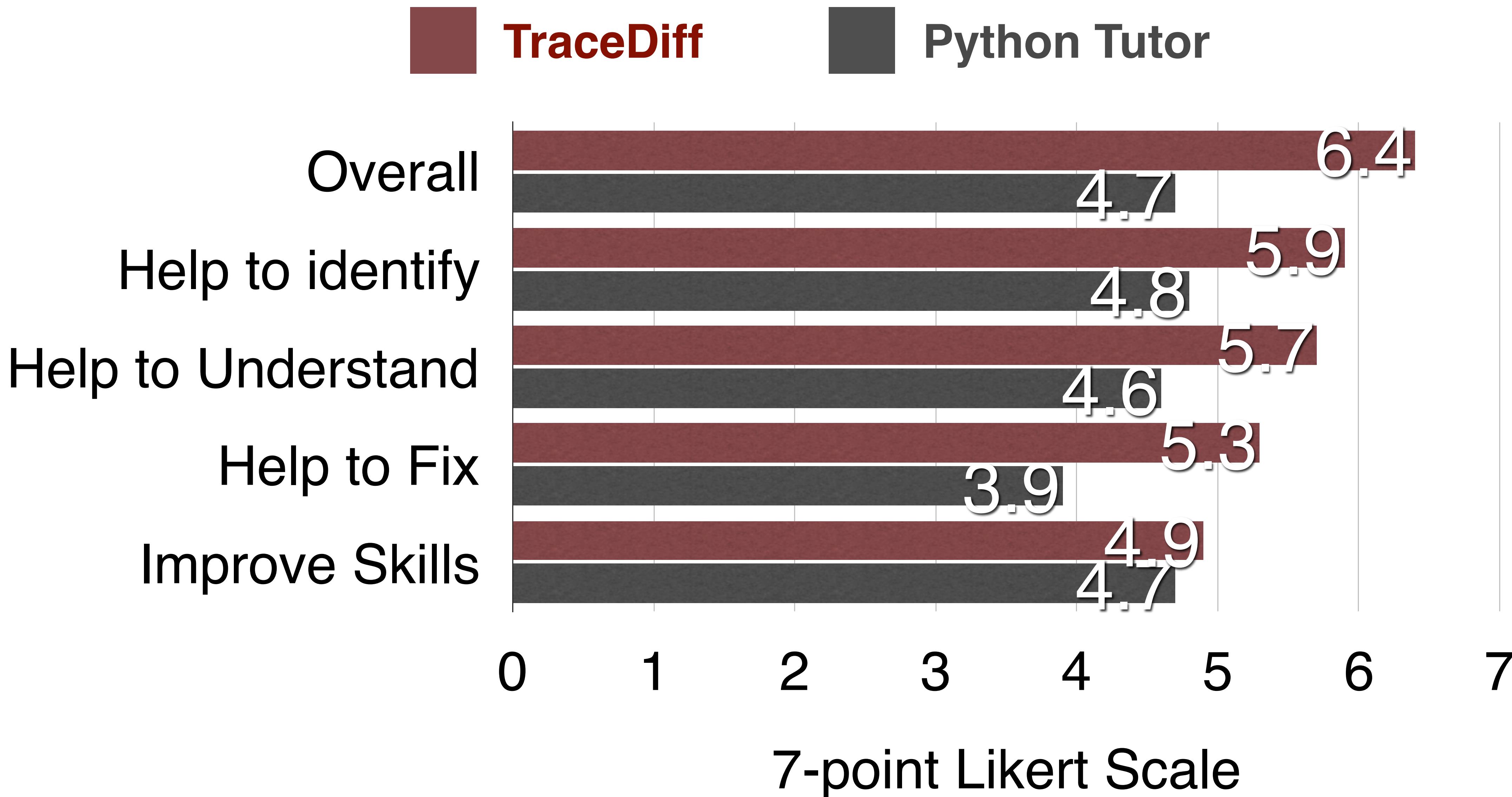
Discussion

Task Selection: **Simple ↔ Complex**

Discussion

Task Setting: **Single** \leftrightarrow **Multiple** Attempts

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



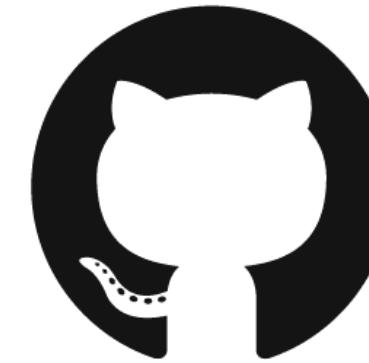
Future Work

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/>

[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