Another simple example

- Suppose we want to compute powers of a number by successive multiplication
- Idea would be to keep track of number of multiplications, plus intermediate product
- Stop when have multiplied number x by itself p times, and return final product
- Here is simple code

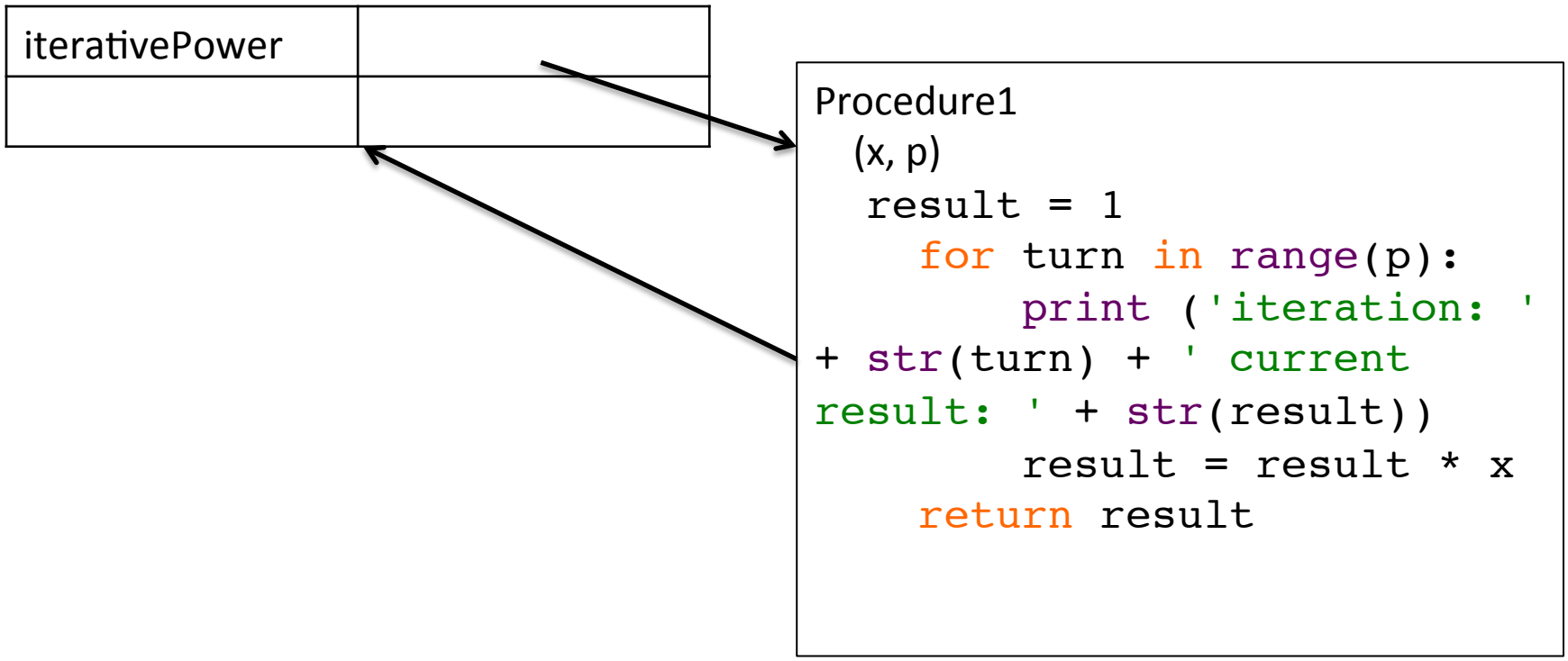
Computing powers

```
x = float(raw_nput('Enter a number: ')
p = int(raw_input('Enter an integer power: '))

result = 1

for turn in range(p):
    print('iteration: ' + str(turn) + \
          'current result: ' + str(result))
    result = result * x
```

Let's define our procedure

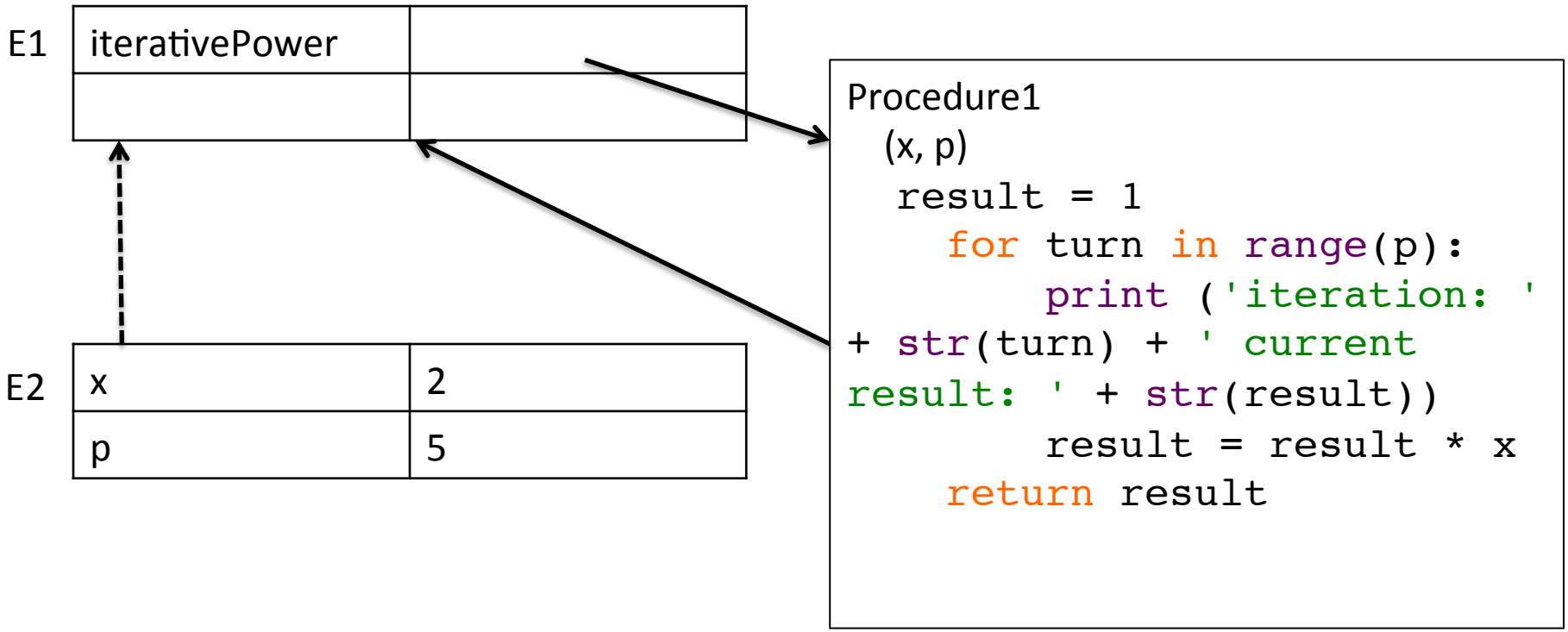
```
def iterativePower(x, p):
    result = 1
    for turn in range(p):
        print ('iteration: ' +
  str(turn) + ' current result: '
  + str(result))
        result = result * x
    return result
```

And this creates



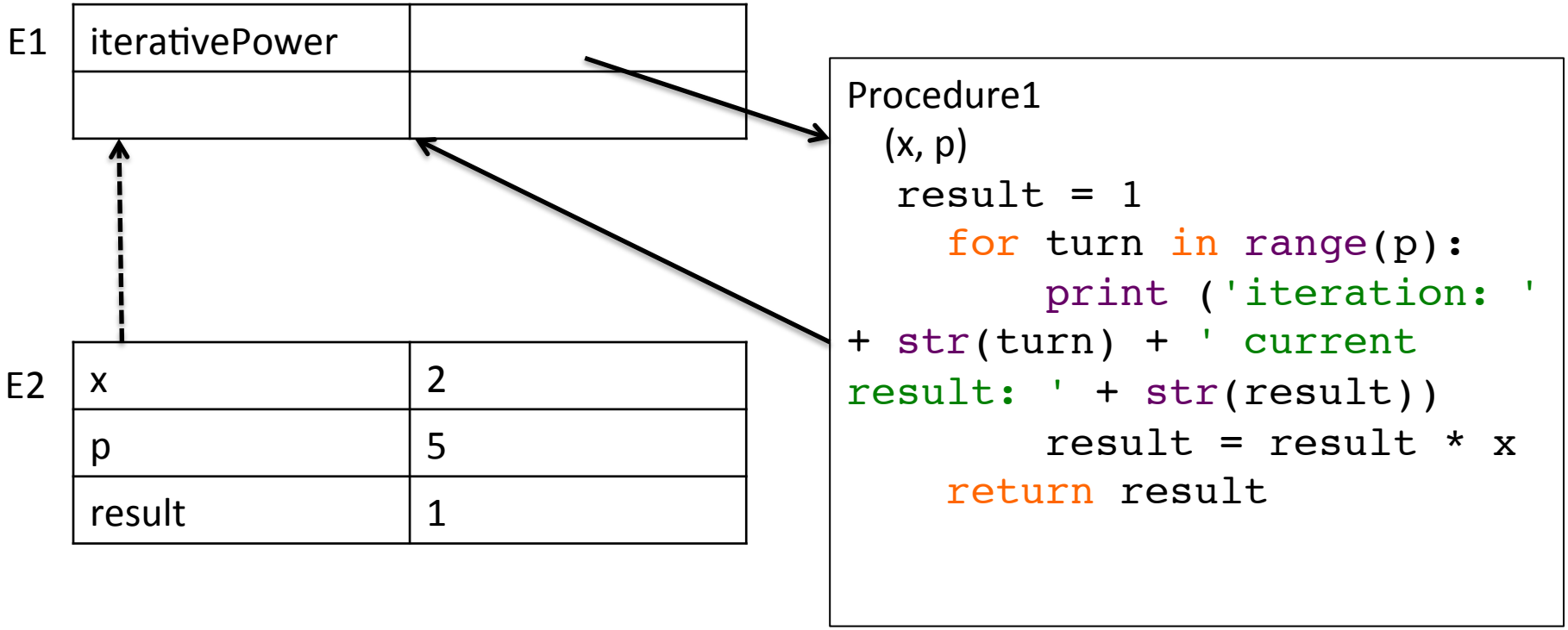
Example

- Call iterativePower(2, 5)



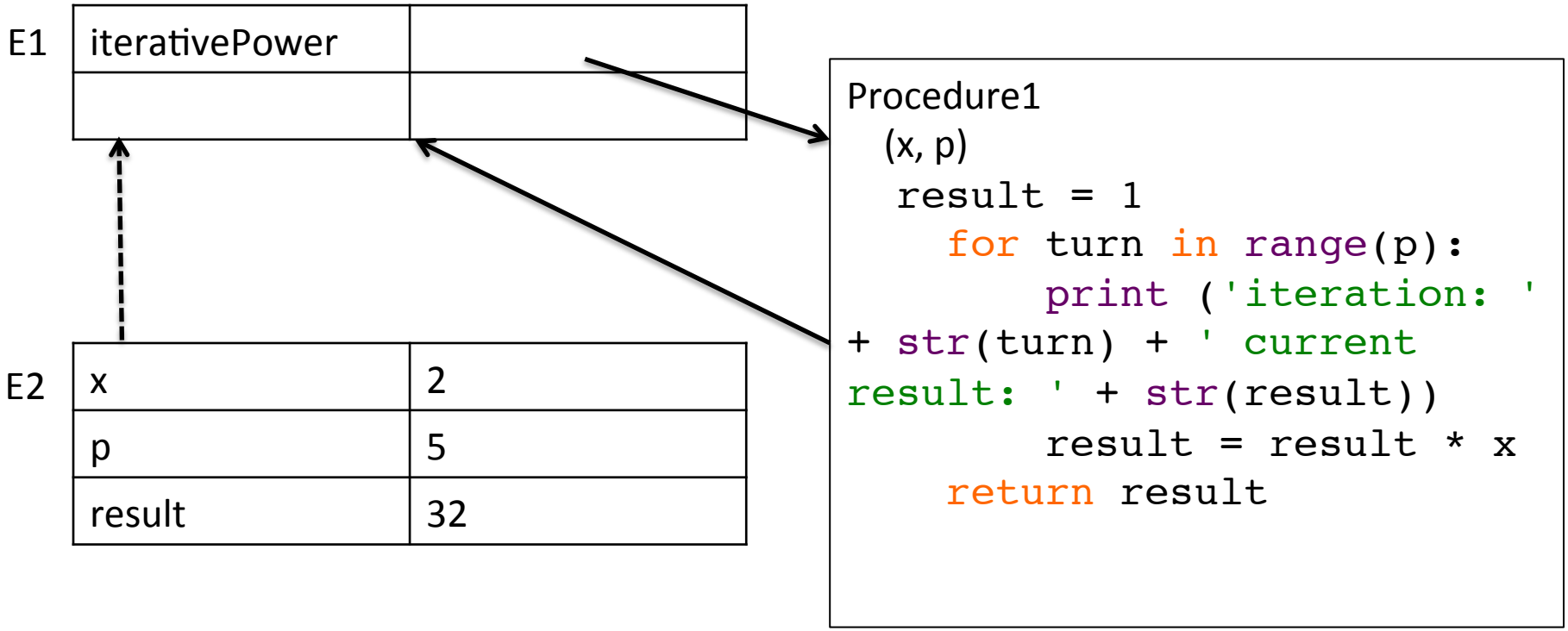
Example

- Evaluating body of procedure initially causes…



Example

- and for loop rebinds local variable until exit, when return statement returns value of result



Scoping is local

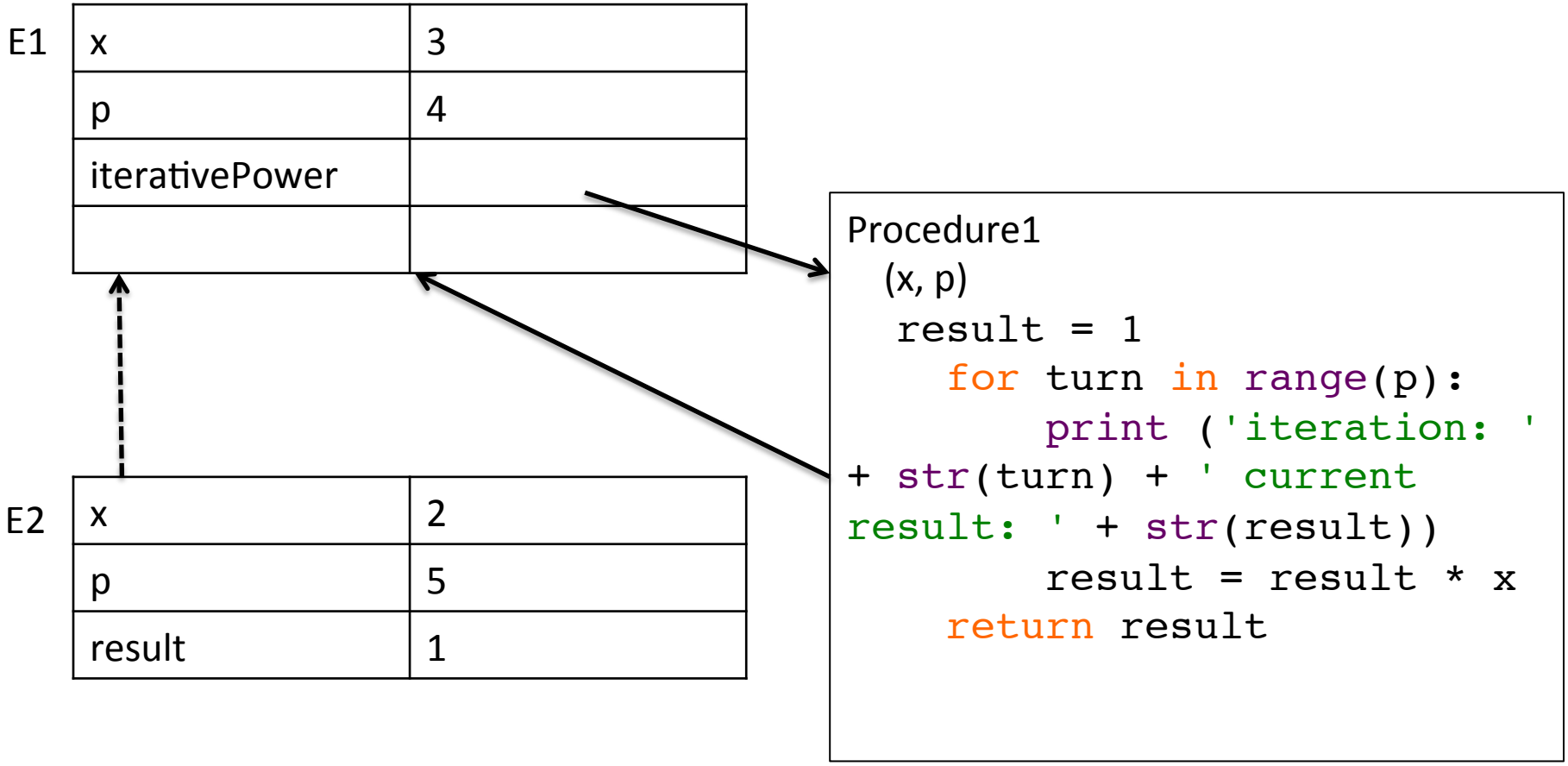
- Imagine we had evaluated

```
x = 3
p = 4
print(iterativePower(2, 5))
```

- Then our local environment would have separate bindings for x and p, which would not be visible to the environment of the function call

Example

- Evaluating body of procedure initially causes…



Example

- But now evaluation of body only sees bindings in E2