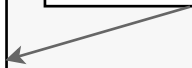


User Environment

new-sqrt

$\lambda(1)$



```
1. (define (new-sqrt x)
2. (define (good-enough? guess)
3. (< (abs (- (square guess) x)) 0.000001))
4. (define (average x y)
5. (/ (+ x y) 2))
6. (define (improve guess)
7. (average guess (/ x guess)))
8. (define (sqrt-iter guess)
9. (if (good-enough? guess)
10. guess
11. (sqrt-iter (improve guess))))
12. (sqrt-iter 1.0))
13. (new-sqrt 2.0)
```

Line 13

User Environment

new-sqrt $\lambda(1)$

new-sqrt

good-enough?	$\lambda(2)$
average	$\lambda(4)$
improve	$\lambda(7)$
sqrt-iter	$\lambda(8)$
x	2

Line 12

Line 13

```
1. (define (new-sqrt x)
2.   (define (good-enough? guess)
3.     (< (abs (- (square guess) x)) 0.000001))
4.   (define (average x y)
5.     (/ (+ x y) 2))
6.   (define (improve guess)
7.     (average guess (/ x guess)))
8.   (define (sqrt-iter guess)
9.     (if (good-enough? guess)
10.        guess
11.        (sqrt-iter (improve guess))))
12.   (sqrt-iter 1.0))
13. (new-sqrt 2.0)
```

User Environment

new-sqrt

$\lambda(1)$

new-sqrt

good-enough?

$\lambda(2)$

average

$\lambda(4)$

improve

$\lambda(7)$

sqrt-iter

$\lambda(8)$

x

2

good-enough

guess

1.0

Line 11

Line 12

Line 13

```
1. (define (new-sqrt x)
2.   (define (good-enough? guess)
3.     (< (abs (- (square guess) x)) 0.000001))
4.   (define (average x y)
5.     (/ (+ x y) 2))
6.   (define (improve guess)
7.     (average guess (/ x guess)))
8.   (define (sqrt-iter guess)
9.     (if (good-enough? guess)
10.        guess
11.        (sqrt-iter (improve guess))))
12.   (sqrt-iter 1.0))
13. (new-sqrt 2.0)
```