

```
1 #lang racket
2
3 (define (try a b)
4   (if (= a 0)
5       1
6       b))
7
8 ;;(try 0 (/ 1 0)) Errors in function arguments show
8 up immediately
9
10 (define numbers (list 1 2 3 4 5))
11
12 (define (trace op)
13   (printf (string-append op "!\\n")))
14
15 (define squares (map (lambda (x) (trace "Map") (* x
15 x)) (list 1 2 3)))
16 squares
17
18 (define (add x y)
19   (trace "Addition")
20   (+ x y))
21
22 (define (subtract x y)
23   (trace "Subtraction")
24   (- x y))
25
26 (define (multiply x y)
27   (trace "Multiply")
28   (* x y))
29
30 (define (fac n)
31   (if (= n 1)
32       1
33       (multiply n (fac (subtract n 1)))))
34
```

```
35 (fac 5)
36
37 (define (tail-fac n)
38   (letrec ((helper (lambda (x res)
39                     (if (= x 1)
40                         res
41                         (helper (subtract x
41 1)(multiply x res))))))
42     (helper n 1)))
43
44 (tail-fac 5)
45
46 (define (loop-forever)
47   (loop-forever))
48
49 ;;(loop-forever)
50
51 (define (three x)
52   3)
53
54 (three (loop-forever))
```