

Evaluation in Traditional REPLs

```
>>> sum(8./x/(x+2) for x in range(1,10000,4))
```

```
3.141392653591789
```

Python's REPL (read-eval-print-loop) lets us play with our code interactively to verify our mental model of it.

Evaluation in Lambda/Live Debugging

- Shows results for *all* subexpressions
- Code is evaluated while being typed
- Browse between different invocations of the same function


```
>>> sum map 1 .. 10000
```

```
step 4
```

```
[1, ...]
```

```
mapping  $x \rightarrow 8 / x / (x + 2)$ 
```

```
1
```

```
8
```

```
3
```



```
2.6666666666666665
```

```
[2.6666666666666665, ...]
```

```
3.141392653591789
```




```
def __init__(self):
```

```
    self.velocity = 0
```

```
    self.set_speed(0, 0, 0) #set speed
```

```
    self.set_pos(0, 0, 0) #set pos
```

```
    self.set_rot(0, 0, 0) #set rot
```

```
    self.set_accel(0, 0, 0)
```

```
    self.set_force(0, 0, 0)
```

```
    self.set_vel(0, 0, 0)
```

```
    self.set_pos(0, 0, 0)
```

```
    self.set_rot(0, 0, 0)
```

```
    self.set_acc(0, 0, 0)
```

```
    self.set_force(0, 0, 0)
```

```
    self.set_vel(0, 0, 0)
```

```
    self.set_pos(0, 0, 0)
```

```
    self.set_rot(0, 0, 0)
```

```
    self.set_acc(0, 0, 0)
```

```
    self.set_force(0, 0, 0)
```

```
    self.set_vel(0, 0, 0) #set vel
```

```
    self.set_pos(0, 0, 0) #set pos
```

```
    self.set_rot(0, 0, 0) #set rot
```

```
    self.set_acc(0, 0, 0)
```



FAQ - Continuous Evaluation

Q: How do we prevent unsafe code execution?
When code buys and sells stocks or launches rockets, should such code be automatically executed?

A:

Lamdu is a pure language (similar to Haskell).
Execution of code with effects is performed only when the user explicitly chooses so.



Evaluation in Traditional REPLs

```
>>> sum(8./x/(x+2) for x in range(1,10000,4))  
3.141392653591789
```



Python's REPL (read-eval-print-loop) lets us play with our code interactively to verify our mental model of it.

Evaluation in Lamdu / Live Debugging

- Shows results for *all* subexpressions
- Code is evaluated while being typed
- Browse between different invocations of the same function

Annotations Evaluation Theme light

```
>>> sum map 1 .. 10000  
      step 4  
      [1, ...]  
mapping x → 8 / x / (x + 2)  
        1    8    3  
        ◀ ▶ 2.6666666666666665  
[2.6666666666666665, ...]  
3.141392653591789
```