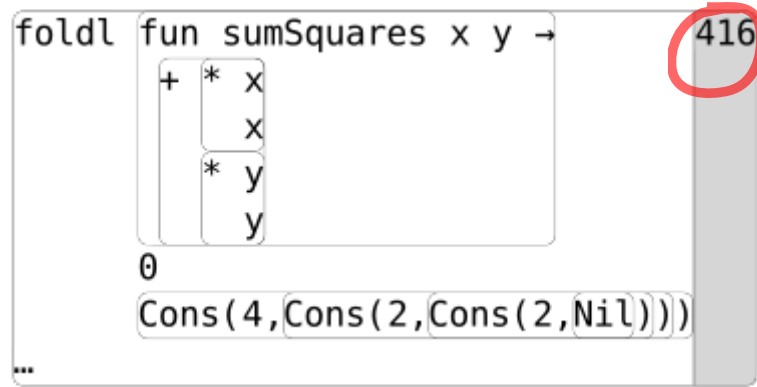


# Execution browsing

```
foldl fun sumSquares x y → 416
  + * x
  x
  * y
  y
0
Cons(4,Cons(2,Cons(2,Nil)))
...
```

# Execution browsing

```
foldl fun sumSquares x y →  
  + * x  
    x  
  * y  
    y  
0  
Cons(4,Cons(2,Cons(2,Nil)))  
...
```



grey **value pane**  
shows computed result

# Execution browsing

```
foldl fun sumSquares x y → 416
  + * x
    x
  * y
    y
0
Cons(4,Cons(2,Cons(2,Nil)))
...
```

there is an **account** of how this value was computed which is currently hidden

# Execution browsing

```
foldl op:fun sumSquares x y → 416
+ * x
  x
  * y
  y
z:0
xs:Cons(4,Cons(2,Cons(2,Nil)))
→ case xs of
  Nil → z
  Cons(x,xs') →
    op x
    foldl op 20
      z
      xs'
  ...
  ...
```

expanding reveals part of the explanation

# Execution browsing

```
foldl op: fun sumSquares x y → 416
  + * x
  x
  * y
  y
z: 0
xs: Cons(4, Cons(2, Cons(2, Nil)))
→ case xs of
  Nil → z
  Cons(x, xs') →
    op x
    foldl op 20
      z
      xs'
  ...
  ...
```

parameter bindings are shown as “keyword arguments”

# Execution browsing

```
foldl op:fun sumSquares x y → 416
+ * x
  x
  * y
  y
z:0
xs:Cons(4,Cons(2,Cons(2,Nil)))
→ case xs of
  Nil → z
  Cons(x, xs') →
  op x
  foldl op 20
  z
  xs'
  ...
  ...
```

matched patterns/  
live variables in **bold**

# Execution browsing

```
foldl op:fun sumSquares x y → 416
  + * x
    x
  * y
    y
z:0
xs:Cons(4,Cons(2,Cons(2,Nil)))
→ case xs of
  Nil → z
  Cons(x,xs') →
    op x
      foldl op 20
        z
        xs'
      ...
    ...
```

intermediate values have their own explanations

# Execution browsing

```
foldl op:fun sumSquares x y → 416
  + * x
    x
  * y
    y
z:0
xs:Cons(4,Cons(2,Cons(2,Nil)))
→ case xs of
  Nil → z
  Cons(x,xs') →
    op x
      foldl op 20
        z
        xs'
      ...
    ...
```

explanation are made of smaller explanations



# Execution browsing

```
foldl op:fun sumSquares x y → 416
  + * x
    x
  * y
    y
z:0
xs:Cons(4,Cons(2,Cons(2,Nil)))
→ case xs of
  Nil → z
  Cons(x,xs') →
    op x
      foldl op:op 20
        z:z
        xs:xs'
      → case xs of
        Nil → z
        Cons(x,xs') →
          op x
            foldl op 4
              z
              xs'
            ...
          ...
        ...
      ...
    ...
  ...
```

explanation are made of smaller explanations

# Execution browsing

```
foldl op:fun sumSquares x y → 416
  + * x
    x
  * y
    y
  z:0
  xs:Cons(4,Cons(2,Cons(2,Nil)))
→ case xs of
  Nil → z
  Cons(x,xs') →
    op x:x
    y:foldl op:op 20
      z:z
      xs:xs'
      → case xs of
        Nil → z
        Cons(x,xs') →
          op x
          foldl op 4
            z
            xs'
            ...
          ...
      → + * x 16
        x
        * y 400
        y
```

explanation are made of smaller explanations

# Execution browsing

syntax all  
the way down

```
foldl op:fun sumSquares x y → 416
  + * x
    x
  * y
    y
z:0
xs:Cons(4,Cons(2,Cons(2,Nil)))
→ case xs of
  Nil → z
  Cons(x,xs') →
    op x:x
    y:foldl op:op 20
      z:z
      xs:xs'
      → case xs of
        Nil → z
        Cons(x,xs') →
          op x
          foldl op 4
            z
            xs'
            ...
          ...
      → + * x16
        x
        * y400
        y
```