

```
sum [] = 0
```




```
sum (x:xs) = x + sum xs
```

```
isEven x = x `div` 2 == 0
```

```
sumEvens l =
```

```
  let evens = filter isEven
```

```
  in sum evens
```

 is identical to  (step )

⌵ The expression **evens** can have two conflicting types



sum

1 2 3 4

evens

5 6 7



## Conflicting types

Possible type 1

**evens** :: **[Int]**

*Inferred from the orange highlights on the left side*

Possible type 2

**evens** :: **[Int] → [Int]**

*Inferred from the blue highlights on the left side*

## Relevant type information

**sum** :: **[Int] → Int**

*Inferred from the orange highlights on the left side*

**filter** :: **(a → Bool) → [a] → [a]**

*Imported from prelude*