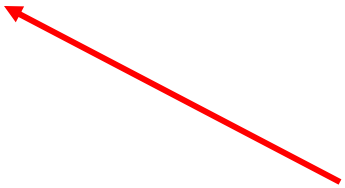
Tuples & Lists

- Tuples have almost the same syntax as Rust.

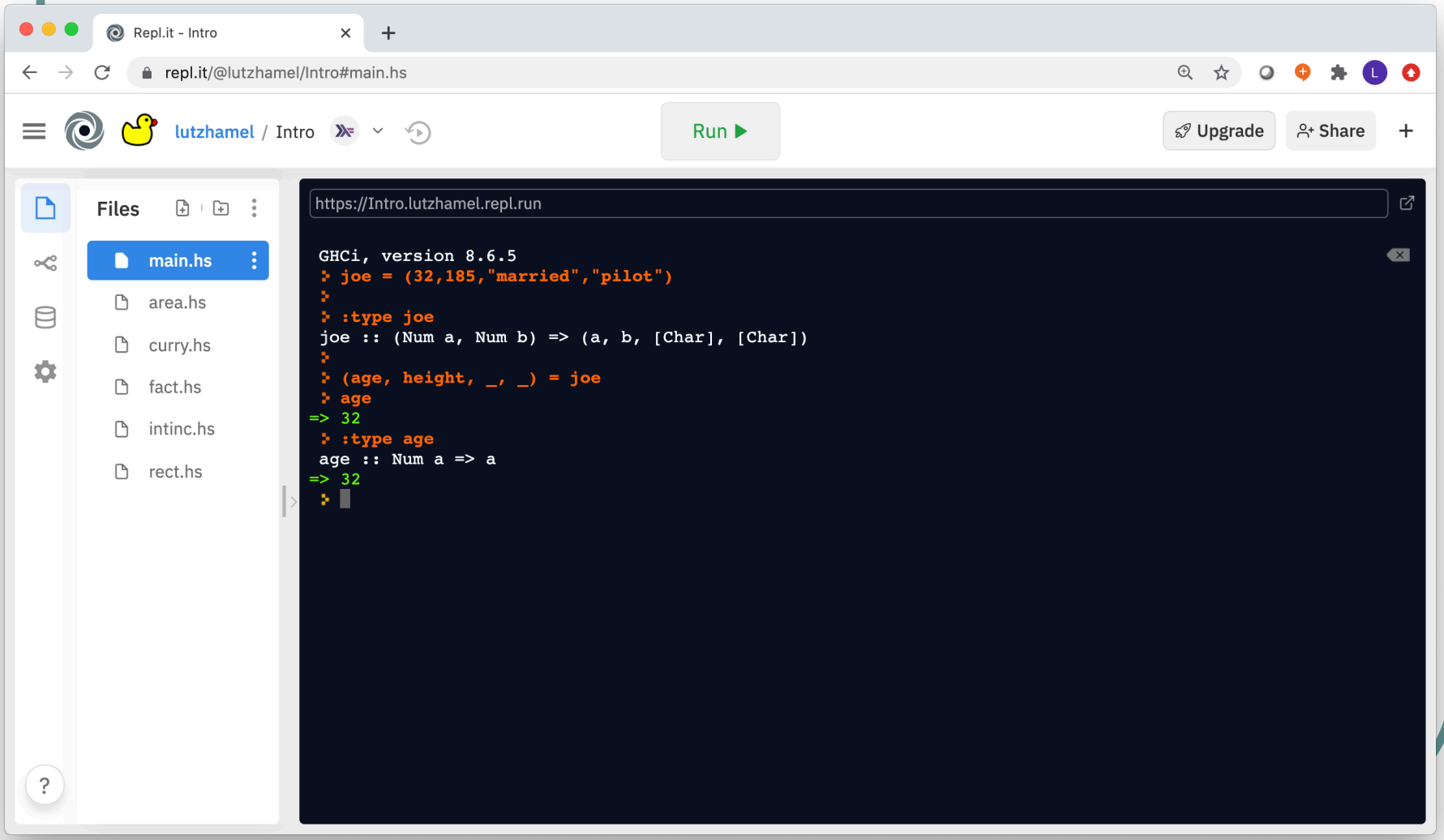
```
> joe = (32,185,”married”,”pilot”)
> :type joe
joe :: (Num a, Num b) => (a, b, [Char], [Char])
> (age, height, _, _) = joe
> age
=> 32
> :type age
age :: Num a => a
```

Type



Pattern Matching

Tuples & Lists

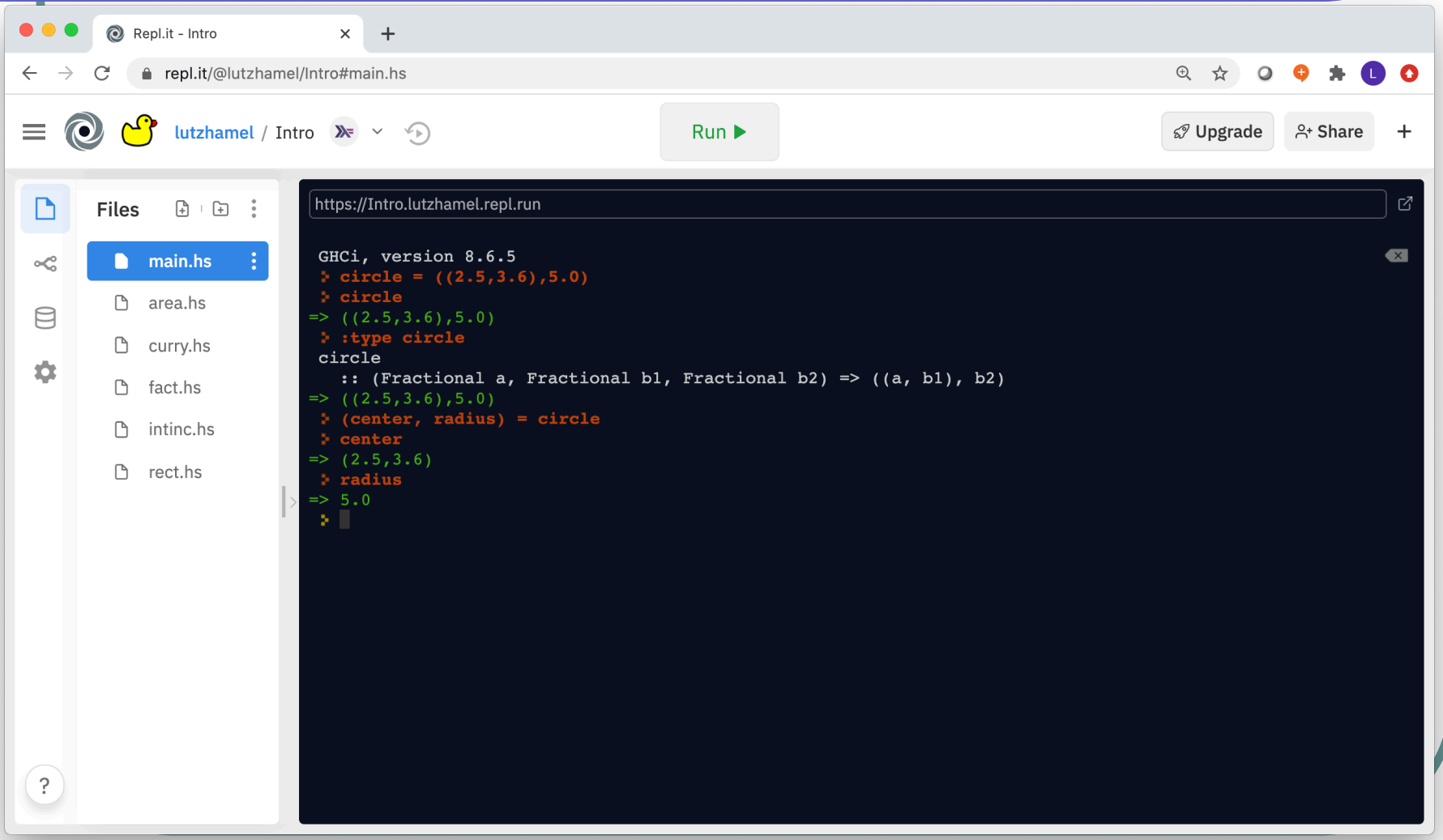


Tuples & Lists

- Nested tuples – say we want to specify a circle with a center and a radius…

```
> circle = ((2.5,3.6),5.0)
> circle
=> ((2.5,3.6),5.0)
> :type circle
circle  :: (Fractional a, Fractional b1, Fractional b2) => ((a, b1), b2)
=> ((2.5,3.6),5.0)
> center, radius) = circle
➢  center
=> (2.5,3.6)
> radius=> 5.0
```

Tuples & Lists



Tuples & Lists

- In a list all elements are of the same type

```
> oddlist = [ 1, 3, 5, 7, 9 ]
> :type oddlist
oddlist :: Num a => [a]
>
```

different from tuples!

nested = [(1,2),(3,4)]
nested = [[1,2],[3,4]]
nested = [[1,2],[3,4,5]]
nested = [(1,2),(3,4,5)]

what is the type of these constructions?

Tuples & Lists

- There exists a special list → the empty list: [ ]

```
> mylist = []
> :type mylist
mylist :: [a]
>
```

Polymorphic type!

List Operators

- ++ - concatenates two lists
  - (++) :: [a] -> [a] -> [a]

Recall, any infix operator can be viewed as a function by putting parentheses around it.

```
 > [1,2,3]++[4,5,6]
=> [1,2,3,4,5,6]
 > ["not"]++["married"]
=> ["not","married"]
 > ['n','o','t']++"married"
=> "notmarried"
 >
```

But…

List Operators

- : - (cons operator) glue elements together to form a list
- the last elements has to always be a list
  - (:) :: a -> [a] -> [a]

```
 ‣ 1 : [2,3]
=> [1,2,3]
 ‣ 1 : 2 : 3 : [] == [1,2,3]
=> True
 ‣ 1 : rest = [1,2,3]
 ‣ rest
=> [2,3]
 ‣ 1 : rest = [4,5,6]
 ‣ rest
*** Exception: <interactive>:26:1-18: Non-exhaustive patterns in 1 : rest
 ‣ a : 2 : rest = [3,2,1,0]
 ‣ a
=> 3
 ‣ rest
=> [1,0]
 ‣
```

List Operators

In all the examples below, what are the types of the variables?

(a) x = ["hello"] ++ ["there"]

(b) x = ["hello" ++ "there"]

(c) joe = (32, 185, "married", "pilot")

    jack = (29, 160, "not married", "cook")

    people = [joe, jack]

(d) l = [[1,2,3],[4,5,6,7,8]]

(e) y = 1:2:3

(f) nested = [(1,2),(3,4,5)]