```
ouzo2:code$ ghci
GHCi, version 8.2.2: http://www.haskell.org/ghc/  :? for help
Loaded GHCi configuration from /Users/hallgren/.ghci
Prelude> :load Intro.hs
[1 of 1] Compiling Main             ( Intro.hs, interpreted )
Ok, one module loaded.
*Main> toE
toEUR   toEnum
*Main> toEUR 85
8.23962776269872
*Main> toSEK 50
515.8000000000001
*Main> :r
[1 of 1] Compiling Main             ( Intro.hs, interpreted )
Ok, one module loaded.
*Main> :r
[1 of 1] Compiling Main             ( Intro.hs, interpreted )
Ok, one module loaded.
*Main> pro
product         prop_exchange   properFraction
*Main> prop
prop_exchange   properFraction
*Main> prop_exchange 50
True
*Main> :r
[1 of 1] Compiling Main             ( Intro.hs, interpreted )
Ok, one module loaded.
*Main> quickCheck prop
prop_exchange   properFraction  property
*Main> quickCheck prop_exchange
*** Failed! Falsifiable (after 13 tests and 2 shrinks):
51.0
*Main> prop_exchange 51
False
*Main> toEUR (toSEK 51)
50.99999999999999
*Main> :r
[1 of 1] Compiling Main             ( Intro.hs, interpreted )
Ok, one module loaded.
*Main> prop_exchange 51
True
*Main> quickCheck prop_exchange
+++ OK, passed 100 tests.
*Main> 5 ~== 4
```

```
False
*Main> 4 ~== 5
True
*Main> :r
[1 of 1] Compiling Main             ( Intro.hs, interpreted )
Ok, one module loaded.
*Main> quickCheck prop_exchange
+++ OK, passed 100 tests.
*Main> :r
[1 of 1] Compiling Main             ( Intro.hs, interpreted )
Ok, one module loaded.
*Main> abs
abs       absolute
*Main> abs
abs       absolute
*Main> absolute 10
10
*Main> absolute (-10)
10
*Main> :r
[1 of 1] Compiling Main             ( Intro.hs, interpreted )
Ok, one module loaded.
*Main> absolute 10
10
*Main> absolute (-10)
*** Exception: Intro.hs:35:1-21: Non-exhaustive patterns in fu
nction absolute

*Main> absolute 10
10
*Main> absolute (-10)
*** Exception: Intro.hs:35:1-21: Non-exhaustive patterns in fu
nction absolute

*Main> absolute 5 - 3
2
*Main> absolute (5 - 3)
2
*Main> absolute 5 - 7
-2
*Main> absolute (5 - 7)
*** Exception: Intro.hs:35:1-21: Non-exhaustive patterns in fu
nction absolute
```

```
*Main> :r
[1 of 1] Compiling Main             ( Intro.hs, interpreted )
Ok, one module loaded.
*Main> absolute (5 - 7)
2
*Main> :r
[1 of 1] Compiling Main             ( Intro.hs, interpreted )
Ok, one module loaded.
*Main> 2^5
32
*Main> :r
[1 of 1] Compiling Main             ( Intro.hs, interpreted )
Ok, one module loaded.
*Main> power 2 5
32
*Main> power 2 10
1024
*Main> power 10 10
10000000000
*Main> power 10 0
1
*Main> power 10 (-1)
*** Exception: Intro.hs:(47,1)-(48,36): Non-exhaustive pattern
s in function power

*Main> :r
[1 of 1] Compiling Main             ( Intro.hs, interpreted )
Ok, one module loaded.
*Main> power 10 (-1)
^C^C*** Exception: stack overflow
*Main>
*Main> power 10 4.5
^CInterrupted.
*Main> :r
[1 of 1] Compiling Main             ( Intro.hs, interpreted )
Ok, one module loaded.
*Main> quickCheck prop_power
*** Failed! (after 3 tests and 1 shrink):                    E
xception:
  Intro.hs:(47,1)-(48,36): Non-exhaustive patterns in function
 power
0
-1
*Main> :r
```

```
[1 of 1] Compiling Main             ( Intro.hs, interpreted )
Ok, one module loaded.
*Main> quickCheck prop_power
+++ OK, passed 100 tests.
*Main> verboseCheck prop_power
Passed:
0
0

Passed:
1
0

Passed:
1
-1

Passed:
2
-2

Passed:
-1
1

Passed:
3
0

Passed:
-2
-6

Passed:
-1
-2

Passed:
5
2

Passed:
-7
-1
```

Passed:
2
10

Passed:
-1
9

Passed:
8
5

Passed:
-7
-4

Passed:
10
-13

Passed:
-10
-14

Passed:
13
14

Passed:
-1
4

Passed:
6
17

Passed:
1
9

Passed:
-4
16

Passed:
−16
13

Passed:
−14
−3

Passed:
−18
9

Passed:
−22
−4

Passed:
−12
11

Passed:
21
0

Passed:
20
3

Passed:
−17
−12

Passed:
20
−6

Passed:
23
30

Passed:
−28
11

Passed:
20
−14

Passed:
−33
14

Passed:
10
−32

Passed:
28
−33

Passed:
29
−19

Passed:
27
−4

Passed:
−5
−27

Passed:
−38
12

Passed:
1
−23

Passed:
−7
−32

Passed:
−32
−8

Passed:
−17
6

Passed:
32
−34

Passed:
33
−26

Passed:
−14
−29

Passed:
44
22

Passed:
35
6

Passed:
7
29

Passed:
−29
47

Passed:
3
20

Passed:
−21
45

Passed:
−18
−41

Passed:
−26
45

Passed:
24
−14

Passed:
−35
−13

Passed:
18
47

Passed:
45
−3

Passed:
−17
20

Passed:
30
56

Passed:
−7
16

Passed:
9
−25

Passed:
14
−26

Passed:
60
41

Passed:
−52
9

Passed:
26
9

Passed:
13
23

Passed:
52
0

Passed:
34
32

Passed:
−10
54

Passed:
1
−32

Passed:
−54
−45

Passed:
−63
−18

Passed:
53
−1

Passed:
−53
26

Passed:
-29
-60

Passed:
43
28

Passed:
-22
54

Passed:
65
-23

Passed:
50
52

Passed:
24
76

Passed:
50
56

Passed:
26
13

Passed:
14
-22

Passed:
-45
-52

Passed:
-4
37

Passed:
-58
-40

Passed:
-39
-49

Passed:
-14
16

Passed:
-42
7

Passed:
86
-27

Passed:
-55
38

Passed:
-13
62

Passed:
-12
-35

Passed:
-47
-87

Passed:
87
-43

Passed:
83
-45

```
Passed:
86
19

Passed:
-50
93

+++ OK, passed 100 tests.
*Main>
*Main> :browse
exchangeRate :: Double
toEUR :: Double -> Double
toSEK :: Double -> Double
prop_exchange :: Double -> Bool
(~==) :: (Fractional a, Ord a) => a -> a -> Bool
absolute :: (Num p, Ord p) => p -> p
absolute' :: (Num p, Ord p) => p -> p
power :: (Ord t, Num p, Num t) => p -> t -> p
prop_power :: (Eq a, Integral p, Num a) => a -> p -> Bool
snacks :: [Char]
dinner :: [[Char]]
*Main> :r
[1 of 1] Compiling Main             ( Intro.hs, interpreted )
Ok, one module loaded.
*Main> :browse
exchangeRate :: Double
toEUR :: Double -> Double
toSEK :: Double -> Double
prop_exchange :: Double -> Bool
(~==) :: (Fractional a, Ord a) => a -> a -> Bool
absolute :: (Num p, Ord p) => p -> p
absolute' :: (Num p, Ord p) => p -> p
power :: Integer -> Integer -> Integer
prop_power :: Integer -> Integer -> Bool
snacks :: [Char]
dinner :: [[Char]]
*Main> power 2 32
4294967296
*Main> power 2 4.5

<interactive>:53:9: error:
    • No instance for (Fractional Integer)
```

```
        arising from the literal ‘4.5’
    • In the second argument of ‘power’, namely ‘4.5’
      In the expression: power 2 4.5
      In an equation for ‘it’: it = power 2 4.5
*Main> power 2 (-5)
*** Exception: Intro.hs:(48,1)-(49,36): Non-exhaustive pattern
s in function power

*Main> :r
[1 of 1] Compiling Main             ( Intro.hs, interpreted )
Ok, one module loaded.
*Main> power 2 5
32.0
*Main> improt

<interactive>:57:1: error: Variable not in scope: improt
*Main> :r
[1 of 1] Compiling Main             ( Intro.hs, interpreted )
Ok, one module loaded.
*Main> intersect 0
0
*Main> intersect 1
0
*Main> intersect 2
1
*Main> intersect 3
3
*Main> intersect 4
6
*Main> intersect 5
10
*Main> :r
[1 of 1] Compiling Main             ( Intro.hs, interpreted )
Ok, one module loaded.
*Main> intersect' 5
10
*Main> intersect' 1
0
*Main> intersect' 0
0
*Main> [0 .. 10]
[0,1,2,3,4,5,6,7,8,9,10]
*Main> [0 .. 100]
[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23
```

```
,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,4
4,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,
65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85
,86,87,88,89,90,91,92,93,94,95,96,97,98,99,100]
*Main> [0,2 .. 100]
[0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38,40,42,
44,46,48,50,52,54,56,58,60,62,64,66,68,70,72,74,76,78,80,82,84
,86,88,90,92,94,96,98,100]
*Main> [100,99 .. 1]
[100,99,98,97,96,95,94,93,92,91,90,89,88,87,86,85,84,83,82,81,
80,79,78,77,76,75,74,73,72,71,70,69,68,67,66,65,64,63,62,61,60
,59,58,57,56,55,54,53,52,51,50,49,48,47,46,45,44,43,42,41,40,3
9,38,37,36,35,34,33,32,31,30,29,28,27,26,25,24,23,22,21,20,19,
18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1]
*Main> :r
[1 of 1] Compiling Main             ( Intro.hs, interpreted )
Ok, one module loaded.
*Main> :type ex
exampleFunction  exchangeRate     exponent
examplePair      exp
exampleTriple    expectFailure
*Main> :type exampleFunction
exampleFunction :: Show a => (Bool, a, String) -> String
*Main> ex
exampleFunction  exchangeRate     exponent
examplePair      exp
exampleTriple    expectFailure
*Main> exampleFunction ex
exampleFunction  exchangeRate     exponent
examplePair      exp
exampleTriple    expectFailure
*Main> exampleFunction exampleTriple
"42"
*Main> :r
[1 of 1] Compiling Main             ( Intro.hs, interpreted )
Ok, one module loaded.
*Main> :t ex
exampleFunction  exchangeRate     exponent
examplePair      exp
exampleTriple    expectFailure
*Main> :t exampleTriple
exampleTriple :: (Integer, Integer, [Char])
*Main> :t exampleFunction
exampleFunction :: Show a => (Bool, a, String) -> String
```

```
*Main> exampleFunction exampleTriple

<interactive>:79:17: error:
    • Couldn't match type 'Integer' with 'Bool'
      Expected type: (Bool, Integer, String)
        Actual type: (Integer, Integer, [Char])
    • In the first argument of 'exampleFunction', namely
        'exampleTriple'
      In the expression: exampleFunction exampleTriple
      In an equation for 'it': it = exampleFunction exampleTri
ple
*Main> 'a'
'a'
*Main> :t 'a'
'a' :: Char
*Main> :t "abc"
"abc" :: [Char]
*Main> :browse
exchangeRate :: Double
toEUR :: Double -> Double
toSEK :: Double -> Double
prop_exchange :: Double -> Bool
(~==) :: (Fractional a, Ord a) => a -> a -> Bool
absolute :: (Num p, Ord p) => p -> p
absolute' :: (Num p, Ord p) => p -> p
power :: Double -> Integer -> Double
prop_power :: Double -> Integer -> Bool
intersect :: Integer -> Integer
intersect' :: Int -> Int
examplePair :: (Integer, [Char])
exampleTriple :: (Integer, Integer, [Char])
exampleFunction :: Show a => (Bool, a, String) -> String
snacks :: [Char]
dinner :: [[Char]]
*Main> :t reverse
reverse :: [a] -> [a]
*Main> reverse [1..10]
[10,9,8,7,6,5,4,3,2,1]
*Main> reverse "Haskell"
"lleksaH"
*Main> ex
exampleFunction  exchangeRate     exponent
examplePair      exp
exampleTriple    expectFailure
```

```
*Main> exampleFunction [1,2,3]

<interactive>:87:17: error:
    • Couldn't match expected type '(Bool, (), String)'
                  with actual type '[Integer]'
    • In the first argument of 'exampleFunction', namely '[1,
2, 3]'
      In the expression: exampleFunction [1, 2, 3]
      In an equation for 'it': it = exampleFunction [1, 2, 3]
*Main> :r
[1 of 1] Compiling Main             ( Intro.hs, interpreted )
Ok, one module loaded.
*Main> summary []
"Nothing"
*Main> summary ["Fish","Chips"]
"Fish and Chips"
*Main> dinner
["Spam","Fish","Chips","Spam","Spam","Pudding"]
*Main> summary dinner
"Spam then some more stuff, finally Pudding"
*Main> summary ["Fish"]
"Fish then some more stuff, finally *** Exception: Prelude.las
t: empty list
*Main> :r
[1 of 1] Compiling Main             ( Intro.hs, interpreted )
Ok, one module loaded.
*Main> summary ["Fish"]
"Only Fish"
*Main> :r
[1 of 1] Compiling Main             ( Intro.hs, interpreted )
Ok, one module loaded.
*Main> len []
0
*Main> len [5]
1
*Main> len [5,4]
2
*Main> len [5,4.3]
2
*Main> len [5,4,3]
3
*Main> len [1..100]
100
*Main> len [1..1000]
```

```
1000
*Main> :t len
len :: Num p => [a] -> p
*Main> :r
[1 of 1] Compiling Main             ( Intro.hs, interpreted )
Ok, one module loaded.
*Main> :t len
len :: [a] -> Integer
*Main> :r
[1 of 1] Compiling Main             ( Intro.hs, interpreted )
Ok, one module loaded.
*Main> last [2]
2
*Main> last [2,4,5]
5
*Main> last' [2,4,5]
5
*Main> last' [2]
2
*Main> last' []
*** Exception: Intro.hs:(91,1)-(92,23): Non-exhaustive pattern
s in function last'

*Main> last []
*** Exception: Prelude.last: empty list
*Main> :r
[1 of 1] Compiling Main             ( Intro.hs, interpreted )
Ok, one module loaded.
*Main> last' []
*** Exception: last': empty list
CallStack (from HasCallStack):
  error, called at Intro.hs:91:16 in main:Main
*Main> [1..10]
[1,2,3,4,5,6,7,8,9,10]
*Main> :r
[1 of 1] Compiling Main             ( Intro.hs, interpreted )
Ok, one module loaded.
*Main> ex1
[1,4,9,16,25,36,49,64,81,100]
*Main> :r
[1 of 1] Compiling Main             ( Intro.hs, interpreted )
Ok, one module loaded.
*Main> doubles [1..10]
[2,4,6,8,10,12,14,16,18,20]
```

```
*Main> :r
[1 of 1] Compiling Main             ( Intro.hs, interpreted )
Ok, one module loaded.
*Main> :r
Ok, one module loaded.
*Main> ex2
[1,3,5,7,9,11,13,15,17,19]
*Main> :r
[1 of 1] Compiling Main             ( Intro.hs, interpreted )
Ok, one module loaded.
*Main> ex3
[('A',1),('A',2),('A',3),('A',4),('B',1),('B',2),('B',3),('B',
4),('C',1),('C',2),('C',3),('C',4),('D',1),('D',2),('D',3),('D
',4)]
*Main> :fr
unknown command ':fr'
use :? for help.
*Main> :r
[1 of 1] Compiling Main             ( Intro.hs, interpreted )
Ok, one module loaded.
*Main> ex4
[('A',1),('B',2),('C',3),('D',4)]
*Main> :r
[1 of 1] Compiling Main             ( Intro.hs, interpreted )
Ok, one module loaded.
*Main> pythag 20
[(3,4,5),(4,3,5),(5,12,13),(6,8,10),(8,6,10),(8,15,17),(9,12,1
5),(12,5,13),(12,9,15),(12,16,20),(15,8,17),(16,12,20)]
*Main> :r
[1 of 1] Compiling Main             ( Intro.hs, interpreted )
Ok, one module loaded.
*Main> pythag 20
[(3,4,5),(5,12,13),(6,8,10),(8,15,17),(9,12,15),(12,16,20)]
*Main> pythag 100
[(3,4,5),(5,12,13),(6,8,10),(7,24,25),(8,15,17),(9,12,15),(9,4
0,41),(10,24,26),(11,60,61),(12,16,20),(12,35,37),(13,84,85),(
14,48,50),(15,20,25),(15,36,39),(16,30,34),(16,63,65),(18,24,3
0),(18,80,82),(20,21,29),(20,48,52),(21,28,35),(21,72,75),(24,
32,40),(24,45,51),(24,70,74),(25,60,65),(27,36,45),(28,45,53),
(28,96,100),(30,40,50),(30,72,78),(32,60,68),(33,44,55),(33,56
,65),(35,84,91),(36,48,60),(36,77,85),(39,52,65),(39,80,89),(4
0,42,58),(40,75,85),(42,56,70),(45,60,75),(48,55,73),(48,64,80
),(51,68,85),(54,72,90),(57,76,95),(60,63,87),(60,80,100),(65,
72,97)]
```