


Haskell,

Tracks / Haskell / Exercises / Hello World



Hello World

In-progress Tutorial Exercise

Overview

Your iterations 1

Continue in editor

Iteration 1 Latest

Submitted via Editor, a few seconds ago

Passed

src/HelloWorld.hs

package.yaml

```
1 module HelloWorld (hello) where
2
3 hello :: String
4 hello = "Hello, World!"
5
```

Analysis

Tests

...

Matthijs gave this feedback on a solution very similar to yours:

Welcome to Haskell!

It looks like you're all set to proceed. Good luck 😊

pig latin
igpay atinlay

Pig Latin

In-progress Medium

Overview

Your iterations 1

Community Solutions

Code Review

Continue in editor

Iteration 1 Latest

Submitted via Editor, a few seconds ago

Passed

src/pigLatin.hs

package.yaml

```
1 module PigLatin (translate) where
2 import Data.List
3 translate = unwords . map ((++"ay") . translateW) . words
4 translateW :: String -> String
5 translateW "" = ""
6 translateW xs@(x:xs')
7   | x`elem`vowels = xs
8   | take 2 xs`elem`["xr", "yt"] = xs
9   | x == 'y' = xs' ++ "y"
10  | take 2 xs' == "qu" = drop 2 xs' ++ take 3 xs
11  | take 2 xs' == "qu" = drop 2 xs' ++ "qu"
12  | otherwise = dropWhile (`elem` consonants) xs ++ takeWhile (`elem` consonants) xs
13 vowels = "aeiou"
14 consonants = ['a'..'z'] \\ ('y':vowels)
15
```

Analysis


Tests

...

No auto suggestions? Try human mentoring.

Get real 1-to-1 human mentoring on the Pig Latin exercise and start writing better Haskell.

Get mentoring



House

In-progressMedium

OverviewYour iterations 1Community SolutionsCode Review

Continue in editor

Iteration 1

Latest

Submitted via Editor, a few seconds ago


src/House.hs

package.yaml

```
1 module House (rhyme) where
2
3 rhyme :: String
4 rhyme = unlines [
5     "This is the house that Jack built.",
6     "",
7     "This is the malt",
8     "that lay in the house that Jack built.",
9     "",
10    "This is the rat",
11    "that ate the malt",
12    "that lay in the house that Jack built.",
13    "",
14    "This is the cat",
15    "that killed the rat",
16    "that ate the malt",
17    "that lay in the house that Jack built.",
18    "",
19    "This is the dog",
20    "that worried the cat",
21    "that killed the rat",
22    "that ate the malt",
23    "that lay in the house that Jack built.",
24    "",
25    "This is the farmer with a horse",
26    "that bought the dog",
27    "that worried the cat",
28    "that killed the rat",
29    "that ate the malt",
30    "that lay in the house that Jack built."
31 ]
```

Analysis

Tests



No auto suggestions? Try human mentoring.

Get real 1-to-1 human mentoring on the House exercise and start writing better Haskell.

Get mentoring

OverviewYour iterations 1Dig DeeperCommunity SolutionsCode Review

Continue in editor

Iteration 1

Latest

Submitted via Editor, a few seconds ago

src/SpaceAge.hs

package.yaml

```
1 module SpaceAge (Planet(..), ageOn) where
2
3 data Planet = Mercury
4             | Venus
5             | Earth
6             | Mars
7             | Jupiter
8             | Saturn
9             | Uranus
10            | Neptune
11
12 ageOn :: Planet -> Float -> Float
13 ageOn planet seconds = ageInEarthYears / orbitalPeriod planet
14 where
15     ageInEarthYears = seconds / 31557600
16
17 orbitalPeriod :: Planet -> Float
18 orbitalPeriod Mercury = 0.2408467
19 orbitalPeriod Venus   = 0.61519726
20 orbitalPeriod Earth   = 1.0
21 orbitalPeriod Mars    = 1.8808158
22 orbitalPeriod Jupiter = 11.862615
23 orbitalPeriod Saturn  = 29.447498
24 orbitalPeriod Uranus  = 84.016846
25 orbitalPeriod Neptune = 164.79132
26
```

Analysis

Tests



No auto suggestions? Try human mentoring.

Get real 1-to-1 human mentoring on the Space Age exercise and start writing better Haskell.

Get mentoring

Overview

Your iterations 1

Community Solutions

Code Review

Continue in editor

Iteration 1

Latest

Submitted via Editor, a few seconds ago

Passed 1


src/ReverseString.hs

package.yaml

```
1 module ReverseString (reverseString) where
2
3 reverseString :: String -> String
4 reverseString str = reverse str
5
```

Analysis

Tests

 **Matthijs** gave this feedback on a solution very similar to yours:

Since the functions `f` and `\x -> f x` each produce exactly the same results when given the same arguments, they are deemed equivalent. That is: whenever you write `\x -> f x` you might as well write only `f` instead. This simplification is called **η -reduction**.

η -equivalence is relevant to this solution because

`f x = body`

is syntactic sugar for (and therefore equivalent to)

`f = \x -> body`