### Haskell-ის ინტერაქტიული კოდი. REPL-ის გამოყენება

Haskell-ი გვაძლევს კოდთან მუშაობის ორ გზას: პირველია, კოდის შეტანა უშუალოდ GHCi-ის ინტერაქტიულ გარემოში ანუ REPL; მეორეა, კოდის შეტანა საწყის ფაილში, შენახვა და შემდეგ საწყისი ფაილის ჩატვირთვა GHCi-ში. ჩვენ შევეხებით ორივე მეთოდს.

REPL არის შემოკლება შემდეგი ფრაზისა: *read-eval-print loop* (ციკლი წაკითხვა-მნიშვნელობის გამოთვლა - დაბეჭდვა). REPL არის ინტერაქტიული პროგრამირების გარემო, სადაც შეგვიძლია შევიტანოთ კოდი, რომლის მნიშვნელობაც გამოითვლება და დავინახავთ შედეგს. ეს პროცესი პირველად გაჩნდა ენა ლისპში, თუმცა ამჟამად საერთოა პროგრამირების თანამედროვე ენებისთვის, მათ შორის Haskell-ისთვის.

ინტერპრეტატორის დაყენებისა და გაშვების შემდეგ ჩვენ დავინახავთ შეტყობინებას ვერსიის შესახებ, განცხადებას (თუ გსურთ დახმარება, აკრიფეთ სიმბოლო ?) და „მოწვევას სამუშაოდ“ ( სტრიქონი Prelude> ):

GHCi, version 7.10.3: http://www.haskell.org/ghc/ :? for help

Prelude>

რა თქმა უნდა, ვერსია შეიძლება იყოს სხვა.

ბრძანებათა სტრიქონში სიტყვა Prelude მიუთითებს, რომ ჩატვირთულია და მზად არის გამოსაყენებლად სტანდარტული ბიბლიოთეკა Prelude. როცა ჩავტვირთავთ სხვა მოდულებს ან საწყის ფაილებს, მაშინ ისინი აისახება დიალოგურ ფანჯარაში. შესაძლებელია Prelude-ის გამორთვაც, რასაც შემდეგ შევეხებით.

მოდულს Prelude ზოგჯერ უწოდებენ „სტანდარტულ პრელუდიას“, ვინაიდან მისი შინაარსი განისაზღვრება Haskell 98–ის სტანდარტით.

მოსაწვევის პასუხად შევიტანოთ რამდენიმე მარტივი არითმეტიკული გამოსახულება:

Prelude> 2 + 2

4

Prelude> 7 < 9

True

Prelude> 10 ^ 2

100

დავწეროთ ჩვენი პირველი პროგრამა Haskell-ზე, რომელსაც ეკრანზე გამოაქვს მისასალმებელი სტრიქონი. ფუნქციას დავარქვათ sayHello და გადავცეთ ერთი სტრიქონული არგუმენტი. ფუნქციის სახელთან მისი განმარტების დასაკავშირებლად გამოვიყენოთ ოპერატორი let (სახელთან დაკავშირების ოპერატორი). REPL გარემოში ავკრიფოთ ფუნქციის განმარტება:

Prelude> let sayHello x = putStrLn ("Hello, " ++ x ++ "!")

მაგალითში გამოყენებული გვაქვს putStrLn, რომელიც არის სტრიქონის ეკრანზე გამოტანის ფორმატირებული ოპერატორი. მაგალითად,

Prelude> putStrLn "Here's a newline -->\n <-- See?"

Here's a newline -->

<-- See?

ეხლა მოვახდინოთ ფუნქციის გამოძახება არგუმენტით "Haskell" და ვნახოთ შედეგი:

Prelude> sayHello "Haskell"

Hello, Haskell!

აქ გამოყენებულია ოპერატორი ++ კონკატენაცია, რომელიც ახდენს სტრიქონების შერწყმას.

GHCi-დან გამოსასვლელად გამოიყენეთ ბრძანება :quit ან :q.

### მუშაობა საწყის ფაილებთან

მუშაობა საწყის ფაილებთან ისევე მოსახერხებელია, როგორც მუშაობა REPL-ის გარემოში. ფაილის გაფართოება სავალდებულოა რომ იყოს .hs, რომელიც მიუთითებს, რომ ეს არის საწყისი კოდი Haskell-ზე. ამ რეჟიმს მაშინ ვიყენებთ, როცა საჭიროა კოდის თანდათანობით აგება. კოდის შექმნის შემდეგ ამ ფაილის ჩავტვირთავთ REPL-ში და შემდეგ ძირითადი პროცესი დადის ინტერაქტიული პროცესზე: კოდის გაშვება, მოდიფიკაცია და ტესტირება.

საწყისი ფაილის შესრულებაც შესაძლებელია ორი გზით: ან გავუშვათ საწყისი კოდის ფაილი, ან REPL-ში გახსნათ იგი. პირველ შემთხვევაში გაიხსნება gnci.exe ინტერპრეტატორის ფანჯარა, სადაც უკვე ჩატვირთული იქნება ფაილის შინაარსი. მეორე შემთხვევაში კი, გავხსნით REPL-ს და სპეციალური დირექტივის (:load) გამოყენებით ჩავტვირთავთ ფაილს.

გავხსნათ ახალი Text Documen. საჭიროა ფაილს დაერქვას სახელი, მაგალითად, test.hs. შევიტანოთ ამ ფაილში შემდეგი კოდი და შევინახოთ იგი იგივე სახელით (test.hs):

sayHello :: String -> IO ()

sayHello x = putStrLn ("Hello, " ++ x ++ "!")

აქ ნიშანი „::“ აღნიშნავს ტიპს ანუ ჩვენ ვამბობთ, რომ „აქვს ტიპი“. ასე, რომ sayHello-ს აქვს ტიპი String -> IO (). ტიპებზე ჩვენ შემდგომ ვისაუბრებთ. შევნიშნოთ, რომ ეს სტრიქონის არ არის აუცილებელი, ვინაიდან Haskell თვითონ შეძლებს ამოიცნოს ჩვენს მიერ განსაზღვრული ფუნქციის ტიპი.

გავხსნათ WinGHCi და ჩავწეროთ შემდეგი ბრძანებები:

Prelude> :load test.hs

\*Main> sayHello "Haskell"

Hello, Haskell!

\*Main>

პირველი ბრძანების :load ჩატვირთავს ფაილს test.hs, რის შემდეგად ფუნქცია sayHello უკვე ჩანს REPL-ში, ამიტომაც შეგიძლიათ გადასცეთ ნებისმიერი სტრიქონი-არგუმენტი (მაგალითად,"Haskell") და ნახოთ შედეგი. დააკვირდით, რომ ფაილის ჩატვირთვის შემდეგ მოსაწვევი სტრიქონი შეიცვალა. თუ გამოიყენებთ GHCi-ის ბრძანებას :m-ს, რომელიც არის ბრძანება :module-ის შემოკლება, მაშინ დაბრუნდება Prelude> და ჩატვირთული ფაილი უკვე აღარ იქნება REPL-ის ხედვის, წვდომის არეში:

\*Main> :m

Prelude> sayHello "Haskell"

<interactive>:12:1: Not in scope: ‘sayHello’

(0.00 secs, 0 bytes)

Prelude>

### სავარჯიშო

1. მოცემულია შემდეგი საწყისი კოდი:

half x = x / 2

square x = x \* x

როგორ უნდა შეცვალოთ იგი, რათა გამოიყენოთ ეს კოდი REPL-ში? რა ტიპისაა თითოეული ფუნქცია (გამოიყენეთ :type დირექტივა!)

1. დაწერეთ ფუნქცია add a b, რომელიც გადაამრავლებს a და b რიცხვებს. ფუნქცია განმარტეთ როგორც REPL გარემოში, ასევე საწყისი კოდის სახით.
2. დაწერეთ ერთი ფუნქცია, რომელსაც ექნება ერთი არგუმენტი და იმუშავებს ყველა გამოსახულებისთვის. დაარქვით ფუნქციას სახელი.

3.14 **\*** (5 **\*** 5)

3.14 **\*** (10 **\*** 10)

3.14 **\*** (2 **\*** 2)

3.14 **\*** (4 **\*** 4)

აქ გამოყენებულია მნიშვნელობა pi.