Синтаксис Haskell, как функционального языка, ориентирован на исполнение парадигм ФП. В связи с этим человеку, плохо знакомому с ФП, некоторые конструкции языка могут показаться сложными для понимания. В процессе разбора синтаксиса я буду сравнивать с языками Java и Python.

Во-первых, так как Haskell все же функциональный язык, я хотел бы обратить внимание на объявление функций. Haskell имеет автоматический вывод типов, однако, при объявлении функций требуется указывать все входные и возвращаемый типы. В этом Haskell похож на Java. В Puthon же и вовсе отсутствует объявление типов в функции - сначала идет определяющая функцию конструкция def, после которой идет название, принимаемые аргументы (нетипизированные) и сразу реализация. Вот примеры объявления функций:

1. Haskell:

**fun :: Int -> String -> Float**

(объявляется фукция fun, принимающая 2 аргумента с типами Int и String b и возвращающая значение типа Float)

1. Java:

**Float fun (Integer a, String b) {** … **}**

(та же самая функция)

1. Python:

**def fun(a, b):**

...

При этом в Haskell это лишь объявление функции, а Java и Python требуют сразу реализовать ее. В Haskell реализация может идти после других функций и даже частями. Например, с фиксированными входными аргументами:

**fun :: Int -> Int -> Double**

**fun n 0 =** ..реализация 1..

**fun 1 k =** ..реализация 2..

**fun n k =** ..реализация 3..

Порядок выполнения такой:

Если второй аргумент равен 0, то выполняется реализация 1, иначе если первый аргумент равен 1 - реализация 2, иначе - реализация 3.

Также в Haskell вместо самих типов можно указывать холдеры:

**fun :: a -> a -> Double**

(это значит, что оба принимаемых аргумента должны быть одного типа).

На первый взгляд объявление входных и возвращаемого типов в Haskell может привести в замешательство - они разделены одним и тем же знаком - “стрелкой”, однако потом становится логичным объяснением то, что последний тип в этом перечислении является типом возвращаемого значения.

Отступы и переносы строк. В Haskell как и в Python наличие и количество отступов и переносов играют важную роль. Они являются не только визуальным разделителем, но и влияют на структуру конструкций кода. В Java эту функцию выполняют скобки **{}** и **;** . С одной стороны влияние отступов имеет положительные эффекты - стиль написания кода мало отличается у разных людей, что повышает читаемость и понимание чужого кода, еще такой код кажется более чистым без дополнительных символов. Однако все же некоторые конструкции Haskell можно писать по-разному (не влияют на компиляцию), но, я думаю, нежелательно. В Java

хотя и есть стандарты - они не всегда выполняются. При желании весь код можно написать в одну строку, соблюдая скобки и разделители, что не есть хорошо.

С другой стороны отсутствие скобок в Haskell может навести на какую-то незавершенность, наличие закрывающей скобки - однозначно показывает конец конструкции.

Объявление и присваивание переменных, а точнее их отсутствие отличает Haskell от Java и Python. Сначала это может озадачить. Но Haskell - чистый функциональный язык программирования, это значит, что “оператор присваивания отсутствует, объекты нельзя изменять и уничтожать, можно только создавать новые путём декомпозиции и синтеза существующих”. Для этого применяется рекурсии. Исходя из этого, можно сделать вывод, что без объявления и присваивания переменных можно обойтись.

К синтаксису Haskell, как и к синтаксису любого другого языку программирования можно привыкнуть путем практики. Haskell - типичный функциональный ЯП, и зная концепции функционального программирования к языку будет проще привыкнуть.