**Mumuki**

Terminal:

* Puedo abrir mi archivo con "stack ghci archivo.hs"
* Luego de guardar un cambio (ctrl + s), debo recargar en la terminal ghci: “:r”
* Salir de la terminal: ":q”
* Ver el tipo de una función: “:t función”

Comentarios:

“-- comment“ comentario de una línea  
“{- comment -}” comentario de líneas múltiples

Alias:

Le puedo dar un **alias** a un valor (los alias son únicos):

Ej. edadFranco = 24

Tipos de dato:

* Números:
  + 14
  + 14.9
  + (-14)
  + (-14.9)
* Booleanos (primera letra mayúscula):
  + True
  + False
* Strings (van entre comillas):

Ej. nombreHobbit = “Frodo”

Funciones:

Se usan de manera “fija”, ya que van siempre antes del parámetro.

* Definición: siguiente numero = numero + 1
* Aplicación: siguiente 4
* Resultado: 5

“siguiente” es el nombre de la función

“numero” es el parámetro

“numero + 1” es el retorno

Si quiero pasar dos parámetros en una función, debo separarlos con un espacio (tanto en la definición como en la aplicación):

Ej. perimetroRectangulo lado1 lado2 = 2 \* lado1 + 2 \* lado2

Composición de funciones:

Ej. dobleDelSiguiente numero = doble (siguiente numero)

Es lo mismo que dobleDelSiguiente = doble.siguiente

Restricción de dominio e imagen:

funcion :: Integer -> Integer

Operadores:

Son como funciones usadas de manera “infija”, ya que pueden ir delante del parámetro o entre varios.

Ej. siguiente numero = (+) numero 1

Ej. numero + 4

Funciones de strings:

* length nos dice el largo de un string.

Ej. length "seru giran"  
10

* head nos devuelve el primer caracter (la primer letra) de un string.

Ej. head "xilofon"  
'x'

* ++ nos permite concatenar (unir) elementos del mismo tipo (listas, strings, ints, etc.) y es infija.

Ej. "guarda" ++ "polvos"  
"guardapolvos"

* elem nos dice si una letra (o cualquier caracter) forma parte de una palabra.

Ej. elem 'a' "moto"  
False

Ej. elem 't' "pintura"  
True

* words separa un String en una lista de strings, donde cada elemento es una palabra del String original.

\*Main> words "hola, como andas"

["hola,","como","andas"]

Funciones varias:

* rem devuelve el resto de la división entera:

Ej. rem 12 5  
2

Guardas:

Sirve para hacer una función por partes. En el siguiente caso, si el número ingresado es par, devuelve su siguiente, sino devuelve el doble.

calcular nro | even nro = siguiente nro  
| otherwise = nro \*2

Variable anónima: “\_”

Tipar un alias:

Ej. edad :: Int  
edad = 24

Tipar una función:

Hay tantas flechas como parámetros que toma la función

siguiente :: Int -> Int

Clases de tipos (Typeclasses):

Son tipos de dato más generales que agrupan otros más específicos.

Por ejemplo, Num es una clase de tipo que agrupa a Int y a Float. Otras clases de tipos son:

* Eq: las cosas que se pueden comparar con == o /=.
* Ord: las cosas que se pueden comparar y ordenar.
* Show:  las cosas que se pueden convertir en un string y mostrar en una consola, mediante la función show

Tipar una función con clases de tipos:

Ej. doble :: Num a => a -> a  
doble numero = numero \* 2

Ej. funcion :: (Ord a, Show b) => a -> a -> b -> Bool  
funcion x y z = x > y || show z == "hola"

Lo que está a la izquierda de “ => “ son restricciones. Ahí se establece que “a” (variable de tipo) es del tipo Num.  
A la derecha de “ => “ se establece lo que toma y lo que devuelve la función, al igual que antes.