Programación Funcional

Introducción al paradigma funcional y Haskell

Mariano Rean

Universidad Nacional de Hurlingham

28 de Marzo, 2020

Introducción

Figure 1: Alonzo Church desarrollador del λ-cálculo en los '30.

Programación

¿Qué es un programa?

Programación

¿Qué es un programa? No sé, pero seguro tiene algo que ver con...

Programación

¿Qué es un programa? No sé, pero seguro tiene algo que ver con...

- variables

Programación

¿Qué es un programa? No sé, pero seguro tiene algo que ver con...

- variables
- ciclos

Programación

¿Qué es un programa? No sé, pero seguro tiene algo que ver con...

- variables
- ciclos
- condicionales

Programación

¿Qué es un programa? No sé, pero seguro tiene algo que ver con...

- variables
- ciclos
- condicionales
- punteros

Programación

¿Qué es un programa? No sé, pero seguro tiene algo que ver con...

- variables
- ciclos
- condicionales
- punteros

Esto es cierto en el paradigma imperativo. Se piensa a la programación de un modo operacional. En esta materia no va a haber nada de esto (!).

Programación Funcional

¿Qué es un programa funcional?

Programación Funcional

¿Qué es un programa funcional? Cualquier programa que no usa variables (mutables), asignaciones, ciclos ni cualquier otra estructura de control imperativa.

Programación Funcional

¿Qué es un programa funcional? Cualquier programa que no usa variables (mutables), asignaciones, ciclos ni cualquier otra estructura de control imperativa.

Y entonces qué nos queda...?

Programación Funcional

¿Qué es un programa funcional? Cualquier programa que no usa variables (mutables), asignaciones, ciclos ni cualquier otra estructura de control imperativa.

Y entonces qué nos queda...? Funciones!

Programación Funcional

¿Qué es un programa funcional? Cualquier programa que no usa variables (mutables), asignaciones, ciclos ni cualquier otra estructura de control imperativa.

Y entonces qué nos queda...? Funciones!
Las funciones (en un sentido puro) describen una relación entre argumentos y resultados.

Programación Funcional

¿Qué es un programa funcional? Cualquier programa que no usa variables (mutables), asignaciones, ciclos ni cualquier otra estructura de control imperativa.

Y entonces qué nos queda...? Funciones!
Las funciones (en un sentido puro) describen una relación entre argumentos y resultados.

En el paradigma funcional se piensa a la programación de un modo denotacional. Describe el *qué* en lugar del *cómo*.

Evolución

Lenguajes Funcionales

- Lisp (1959)

Evolución

Lenguajes Funcionales

- Lisp (1959)
- ML (1973)

Evolución

Lenguajes Funcionales

- Lisp (1959)
- ML (1973)
- Haskell (1990)

Evolución

Lenguajes Funcionales

- Lisp (1959)
- ML (1973)
- Haskell (1990)
- Javascript (1995)

Evolución

Lenguajes Funcionales

- Lisp (1959)
- ML (1973)
- Haskell (1990)
- Javascript (1995)
- OCaml (1996)

Evolución

Lenguajes Funcionales

- Lisp (1959)
- ML (1973)
- Haskell (1990)
- Javascript (1995)
- OCaml (1996)
- Scala (2003)

Evolución

Lenguajes Funcionales

- Lisp (1959)
- ML (1973)
- Haskell (1990)
- Javascript (1995)
- OCaml (1996)
- Scala (2003)

De todos estos, el único lenguaje funcional puro es Haskell.

Evolución

Lenguajes Funcionales

- Lisp (1959)
- ML (1973)
- Haskell (1990)
- Javascript (1995)
- OCaml (1996)
- Scala (2003)

De todos estos, el único lenguaje funcional puro es Haskell.

Otros lenguajes de programación incorporaron características de los lenguajes funcionales:

Evolución

Lenguajes Funcionales

- Lisp (1959)
- ML (1973)
- Haskell (1990)
- Javascript (1995)
- OCaml (1996)
- Scala (2003)

De todos estos, el único lenguaje funcional puro es Haskell.

Otros lenguajes de programación incorporaron características de los lenguajes funcionales:

- C++11 (2011)

Evolución

Lenguajes Funcionales

- Lisp (1959)
- ML (1973)
- Haskell (1990)
- Javascript (1995)
- OCaml (1996)
- Scala (2003)

De todos estos, el único lenguaje funcional puro es Haskell.

Otros lenguajes de programación incorporaron características de los lenguajes funcionales:

- C++11 (2011)
- Java 8 (2014)

Motivación

¿Por qué estudiar programación funcional?

Motivación

¿Por qué estudiar programación funcional?

- Nos da otra forma de pensar la programación

Motivación

¿Por qué estudiar programación funcional?

- Nos da otra forma de pensar la programación
- Escribir código declarativo da lugar a un código:

Motivación

¿Por qué estudiar programación funcional?

- Nos da otra forma de pensar la programación
- Escribir código declarativo da lugar a un código:
  - legible

Motivación

¿Por qué estudiar programación funcional?

- Nos da otra forma de pensar la programación
- Escribir código declarativo da lugar a un código:
  - legible
  - modificable/extensible

Motivación

¿Por qué estudiar programación funcional?

- Nos da otra forma de pensar la programación
- Escribir código declarativo da lugar a un código:
  - legible
  - modificable/extensible
  - escalable

Motivación

¿Por qué estudiar programación funcional?

- Nos da otra forma de pensar la programación
- Escribir código declarativo da lugar a un código:
  - legible
  - modificable/extensible
  - escalable
  - reutilizable

Motivación

¿Por qué estudiar programación funcional?

- Nos da otra forma de pensar la programación
- Escribir código declarativo da lugar a un código:
  - legible
  - modificable/extensible
  - escalable
  - reutilizable
- facilita el testing

Motivación

¿Por qué estudiar programación funcional?

- Nos da otra forma de pensar la programación
- Escribir código declarativo da lugar a un código:
  - legible
  - modificable/extensible
  - escalable
  - reutilizable
- facilita el testing
- evita problemas en la programación concurrente

Motivación

¿Por qué estudiar programación funcional?

- Nos da otra forma de pensar la programación
- Escribir código declarativo da lugar a un código:
  - legible
  - modificable/extensible
  - escalable
  - reutilizable
- facilita el testing
- evita problemas en la programación concurrente

Conclusión: nos hace mejores programadores!

Haskell

Figure 2: Haskell B. Curry fue un matemático y lógico que hizo importantes contribuciones a la lógica combinatoria, una variante del λ-cálculo.

Programación en Haskell

Haskell es un lenguaje funcional puro.

Programación en Haskell

Haskell es un lenguaje funcional puro.

Es decir, toda función en Haskell al evaluarse en los mismos valores devuelve el mismo resultado.

Programación en Haskell

Haskell es un lenguaje funcional puro.

Es decir, toda función en Haskell al evaluarse en los mismos valores devuelve el mismo resultado.

Esta propiedad hace que probar la correctitud de una función sea mucho más sencillo.

Programación en Haskell

Haskell es un lenguaje funcional puro.

Es decir, toda función en Haskell al evaluarse en los mismos valores devuelve el mismo resultado.

Esta propiedad hace que probar la correctitud de una función sea mucho más sencillo.

Por ejemplo, analicemos la siguiente función en Python:

Funciones impuras

```
1     v = 0
2     def f(x):
3       v += x
4       return v
5     print(f(1))
6     print(f(1))
```

Funciones impuras

```
1     v = 0
2     def f(x):
3       v += x
4       return v
5     print(f(1))
6     print(f(1))
```

Notemos que la primer llamada a $f(1)$ devuelve 1, pero la segunda llamada devuelve 2. En este caso decimos que la función f es impura.

Funciones impuras

¿Qué problema trae esto?

Funciones impuras

¿Qué problema trae esto?

Si tenemos las expresiones $f(1) + f(1)$ y $2 * f(1)$. No podemos intercambiarlas ya que denotan diferentes valores!

Funciones impuras

¿Qué problema trae esto?

Si tenemos las expresiones $f(1) + f(1)$ y $2 * f(1)$. No podemos intercambiarlas ya que denotan diferentes valores!

$f(1) + f(1) => 1 + 2 = 3$

$2 * f(1) => 2 * 1 = 2$

Programación en Haskell

¿Qué es un programa en Haskell?

Programación en Haskell

¿Qué es un programa en Haskell?

- Conjunto de ecuaciones orientadas.

Programación en Haskell

¿Qué es un programa en Haskell?

- Conjunto de ecuaciones orientadas.
- La ejecución del programa consiste en evaluar una expresión.

Programación en Haskell

¿Qué es un programa en Haskell?

- Conjunto de ecuaciones orientadas.
- La ejecución del programa consiste en evaluar una expresión.

```
1  doble x = 2 * x
2  suma x y = x + y
```

Programación en Haskell

¿Qué es un programa en Haskell?

- Conjunto de ecuaciones orientadas.
- La ejecución del programa consiste en evaluar una expresión.

```
1  doble x = 2 * x
2  suma x y = x + y
```

Podemos evaluar expresiones de forma interactiva por medio del entorno interactivo **GHCi** (`https://www.haskell.org/ghc/download`).

Programación en Haskell

¿Qué es un programa en Haskell?

- Conjunto de ecuaciones orientadas.
- La ejecución del programa consiste en evaluar una expresión.

```
1  doble x = 2 * x
2  suma x y = x + y
```

Podemos evaluar expresiones de forma interactiva por medio del entorno interactivo **GHCi** (`https://www.haskell.org/ghc/download`).

```
1  Prelude> suma (doble 10) 5
2  25
```

A programar!

1. Crear un directorio propio y trabajar toda la clase ahí adentro.
2. Abrir una terminal, ir al directorio propio y ejecutar **ghci** (abre el intérprete de Haskell).
3. Ejecutar alguna operación simple, por ejemplo **8 * 7**.
4. Escribir en un archivo con nombre Clase01.hs las funciones *doble* y *suma*.
5. Cargar el archivo en el entorno:

```
Prelude> :l Clase01.hs
```

6. Evaluar expresiones con *doble* y *suma*.
7. Agregar al código la función *producto* que toma dos paramétros y los multiplica.
8. En GHCi, recargar el programa:

```
*Main> :r
```

9. Evaluar expresiones con *producto*
10. Pueden cerrar el intérprete ejecutando:

```
*Main> :q
```

Ejercicios

Programar las siguientes funciones

- $cuadrado(x) = x^2$
- $||(x_1, x_2)|| = \sqrt{x_1^2 + x_2^2}$
- $constante8(x) = 8$

Guardas

En matemática definimos funciones partidas de la forma:

$$|x| = \begin{cases} x & x \geq 0 \\ -x & x < 0 \end{cases}$$

Guardas

En matemática definimos funciones partidas de la forma:

$$|x| = \begin{cases} x & x \geq 0 \\ -x & x < 0 \end{cases}$$

En haskell podemos hacer algo similar usando guardas:

```
1  absoluto x | x >= 0 = x
2  absoluto x | x < 0  = -x
```

Guardas

En matemática definimos funciones partidas de la forma:

$$|x| = \begin{cases} x & x \geq 0 \\ -x & x < 0 \end{cases}$$

En haskell podemos hacer algo similar usando guardas:

```
1  absoluto x | x >= 0 = x
2  absoluto x | x < 0  = -x
```

Otra forma:

```
1  absoluto x | x >= 0     = x
2             | otherwise = -x
```