Programación I - Segundo Cuatrimestre

Unidad 1: Constantes y Funciones

Septiembre 2020

Natalia Colussi

Licenciatura en Ciencias de la Computación
Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Universidad Nacional de Rosario

Resumen

1. Las Constantes

2. Las Funciones

    Sintaxis en Racket

    Composición de Funciones

    Evaluación de Funciones

    Alcance de los Parámetros

Las Constantes

Constantes

- Asociamos un identificador con un valor durante todo el programa.
- Usamos siempre las mayúsculas para la constantes.
- En general las constantes las escribimos al ppio del archivo.
- La palabra clave de Racket para definir una constantes es **define**.
- La sintaxis es:

  (define < **IDENTIFICADOR** > <**EXPRESIÓN** >**)**
- Vamos a Racket!

Las Funciones

Funciones

- Las funciones generalizan expresiones similares.
- Ejemplo 1:

$$5^3 \times 2 + 1$$
$$7^3 \times 2 + 1$$
$$2^3 \times 2 + 1$$

- ¿Qué podemos observar? ¿Qué expresión podemos abstraer de forma general?
- ¿Podemos hacer algo más?

Funciones

- Matemáticamente podemos llamar a esta función:

$$f(x) = x^3 \times 2 + 1$$

- ¿Qué hicimos?

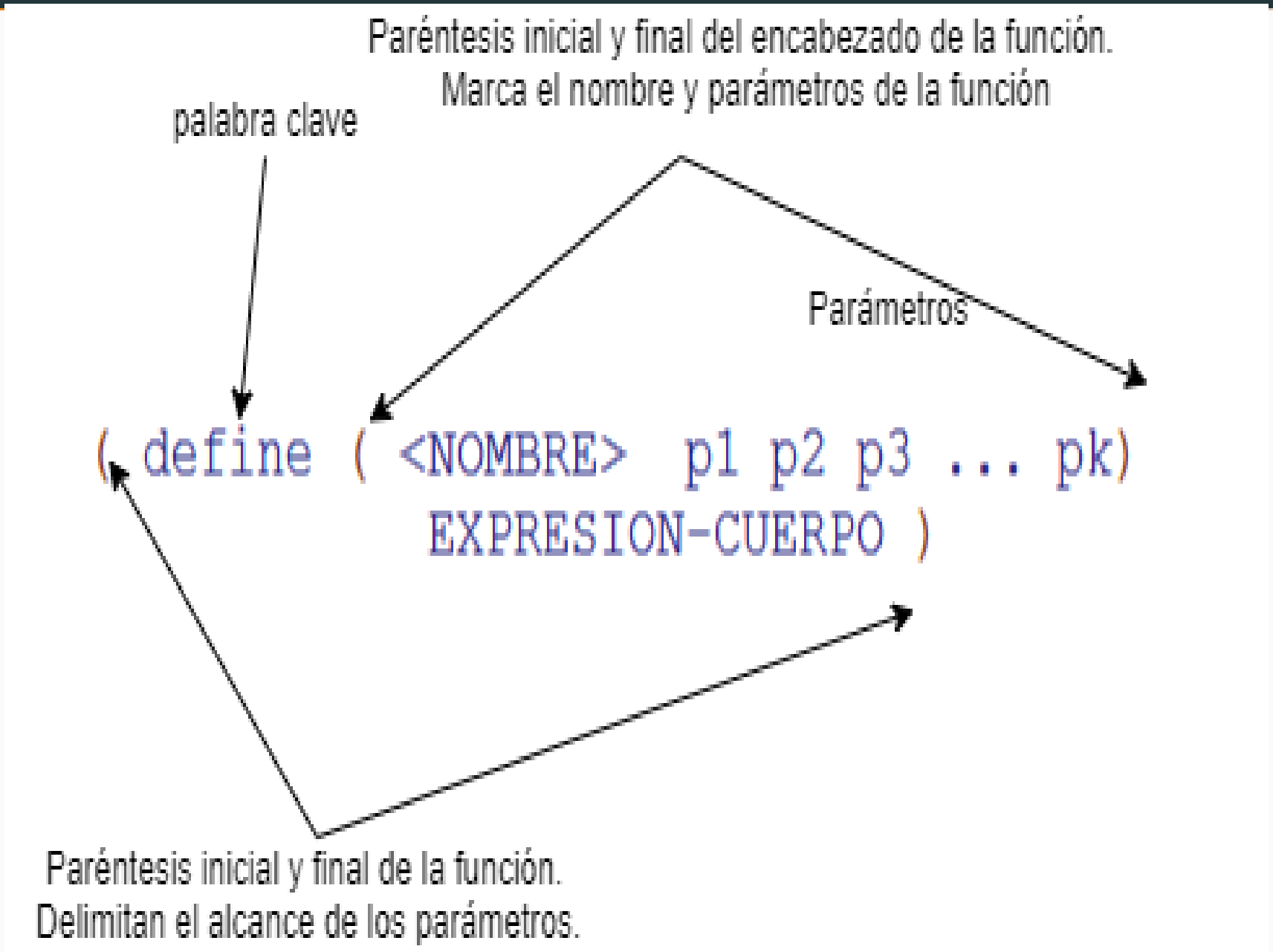
  Abstrajimos una expresión (valor) y lo reemplazamos por una variable.

- ¿Cómo escribimos las expresiones anteriores?

$$\begin{array}{lll} f(5) & \text{es igual a} & 5^3 \times 2 + 1 \\ f(7) & \text{es igual a} & 7^3 \times 2 + 1 \\ f(2) & \text{es igual a} & 2^3 \times 2 + 1 \end{array}$$

- ¿Cómo podemos expresar estas ideas en Racket?

Funciones

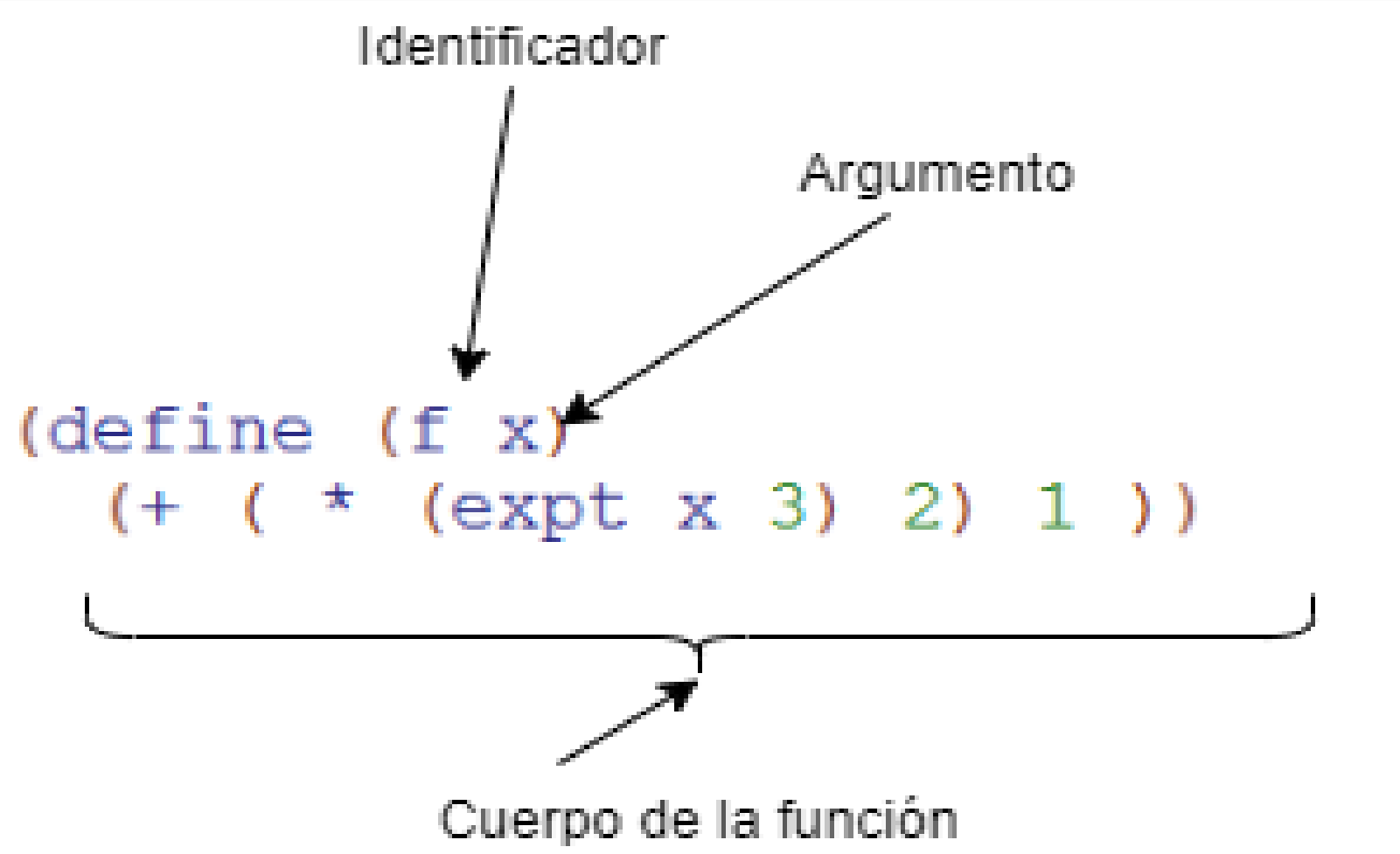

Paréntesis inicial y final del encabezado de la función.
Marca el nombre y parámetros de la función
palabra clave
Parámetros
(define (<NOMBRE> p1 p2 p3 ... pk)
EXPRESION-CUERPO)
Paréntesis inicial y final de la función.
Delimitan el alcance de los parámetros.

Funciones

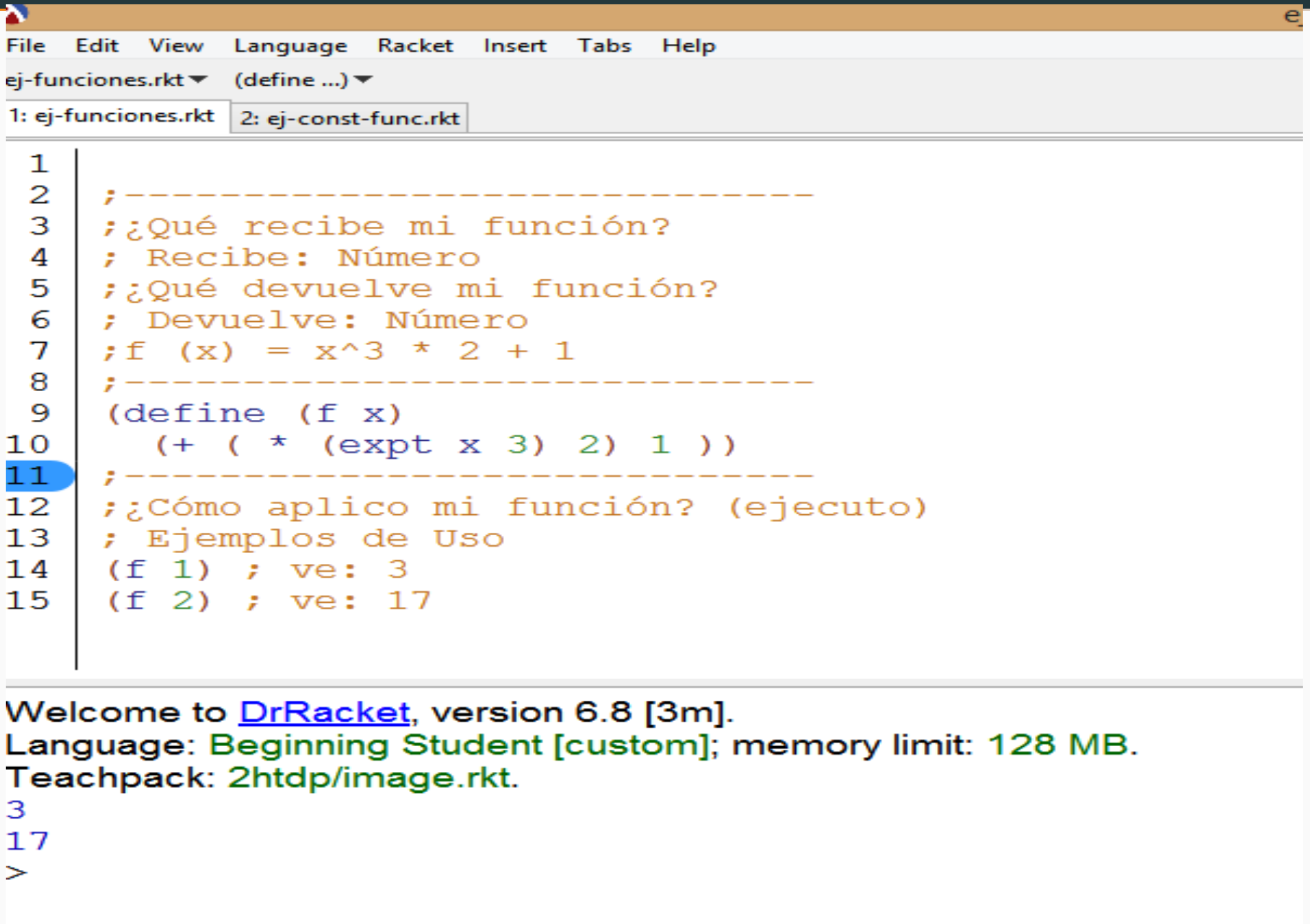
La definición de funciones consiste de:

- un **nombre** , es decir, un **identificador**
  - En general buscamos que sea significativo.
  - Nos permite aplicar la función, es decir, invocarla.
  - En un mismo archivo de trabajo no repetimos el nombre de las funciones.
- una lista de **parámetros** ;
  - Puede tener un parámetro o más, pero no cero.
  - Los parámetros son variables especiales, les damos nombre significativos.
  - Se asocian a los argumentos que le pasamos a la función cuando la invocamos.
  - Pueden pertenecer a distintos tipos (conjuntos de datos).
  - Tienen sentido en el cuerpo de la función.
- un **cuerpo** asociado al identificador y los parámetros.
  - El cuerpo puede ser una expresión o un valor constante.
  - La expresión asociada al cuerpo vinculará, relacionará de una forma particular (según lo que haga la función) a los argumentos.

Funciones


Identificador
Argumento
(define (f x)
(+ (* (expt x 3) 2) 1))
Cuerpo de la función

Funciones


File Edit View Language Racket Insert Tabs Help
ej-funciones.rkt (define ...)
1: ej-funciones.rkt
2: ej-const-func.rkt
1
2 ;------------------------------
3 ;¿Qué recibe mi función?
4 ; Recibe: Número
5 ;¿Qué devuelve mi función?
6 ; Devuelve: Número
7 ;f (x) = x^3 * 2 + 1
8 ;------------------------------
9 (define (f x)
10 (+ (* (expt x 3) 2) 1))
11 ;------------------------------
12 ;¿Cómo aplico mi función? (ejecuto)
13 ; Ejemplos de Uso
14 (f 1) ; ve: 3
15 (f 2) ; ve: 17
Welcome to DrRacket, version 6.8 [3m].
Language: Beginning Student [custom]; memory limit: 128 MB.
Teachpack: 2htdp/image.rkt.
3
17
>

Funciones

- ¿Podemos agregar algo más a esta función?

Funciones

- ¿Podemos agregar algo más a esta función?
- ¿Qué sucedería si tuviéramos las siguientes funciones ya definidas?

```
1  ;al-cubo: Número -> Número
2  ;-----------------------------------
3  (define (al-cubo num)
4                (expt num 3) )
5  ;-----------------------------------
6
7  ;doble: Número -> Número
8  ;-----------------------------------
9  (define (doble num)
10               (* 2 num) )
11
12 ;suma-uno: Número -> Número
13 ;-----------------------------------
14 (define (suma-uno num)
15           (+ num 1) )
```

Funciones

- ¿Podemos agregar algo más a esta función?
- ¿Qué sucedería si tuviéramos las siguientes funciones ya definidas?

```
1  ;al-cubo: Número -> Número
2  ;-----------------------------------
3  (define (al-cubo num)
4                (expt num 3) )
5  ;-----------------------------------
6
7  ;doble: Número -> Número
8  ;-----------------------------------
9  (define (doble num)
10               (* 2 num) )
11
12 ;suma-uno: Número -> Número
13 ;-----------------------------------
14 (define (suma-uno num)
15           (+ num 1) )
```

Funciones

- Redefinimos la función original, inicial,

$$f(x) = x^3 \times 2 + 1$$

- ... utilizando las tres funciones que ya teníamos hecha.

```
1  ;f: Número -> Número
2  ;-----------------------------------
3  ( define (f num)
4             (suma-uno (doble (al-cubo num))) )
5  ;-----------------------------------
6  (f 5) ; ve: 251
7  (f 7) ; ve: 687
8  (f 2) ; ve: 17
```

Funciones

- ¿Cómo se llega a la **composición** profesora?

Funciones

- ¿Cómo se llega a la **composición** profesora?
- A partir del principio o la **estrategia** de resolución de problemas basada en la **división de tareas**, o la dividir un problema en sub-problemas.

Evaluación de Funciones

- De igual modo que evaluamos las expresiones numéricas y booleanas, evaluaremos las expresiones que tienen asociadas funciones.
- Dadas las siguientes definiciones:

```
;fun: Número -> Número
;-----------------------------------
( define (fun num)
           (* num  num))
;-----------------------------------
(define (gee x y)
           (- x y))
-------------------------------------
```

- Vamos a calcular esta evaluación

  `(gee (fun 2) (gee 3 1))`

  y luego lo chequeamos con Racket.

Evaluación de Funciones

```
    (gee (fun 2) (gee 3 1))
=   {Def. fun }
    (gee (* 2 2) (gee 3 1))
=   {Def. * }
    (gee 4 (gee 3 1))
=   {Def. gee }
    (gee 4 (- 3 1))
=   {Def. - }
    (gee 4 2)
=   {Def. gee }
    (- 4 2)
=   {Def. - }
    2.
```

Evaluación de Funciones

- En las evaluaciones siempre elegimos la **expresión que esta más a la izquierda**.
- Resolvemos las **sub-expresiones en los argumentos primero**, y luego aplicamos la función.
- Si hay **varias sub-expresiones** por resolver en los argumentos, tomamos siempre la que esta más a la izquierda.
- En cada paso de evaluación analizamos siempre primero qué debemos resolver sobre **totalidad de la expresión**.

Alcance de los Parámetros

- Nos preguntamos ¿Hasta dónde tienen sentido **referenciar** un parámetro utilizado en una función?
- Los parámetros tienen "sentido" dentro del cuerpo de la función, fuera de ella no, asumen otro "sentido".
- Sentido = valor asociado al invocar a la función.
- Ejemplo:

```
1  (define x 10)  ; Constante en mi programa
2
3  ;fun: Number -> Number
4  (define (fun x)
5          (* x x 100) )
6  ;-----------------------------------------
7  (fun 5) ; x (parámetro) se asoció al valor 5.
8
9  (+ x 10) ; x referencia  a la constante 10.
```

Alcance de los Parámetros

Alcance de una variable/constante
Entorno (expresión o programa completo) en donde la variable/constante tiene asociado (conserva) su último valor.

- En nuestros programas **Racket** las *constantes* tienen el alcance (conservar su valor asociado) en todo el código del programa.
- En las funciones definidas en **Racket**, los *parámetros* (variables) su alcance llega sólo sobre el cuerpo de la función.

¿Preguntas?

- **Pueden dejar las consultas en el foro del capítulo, aclaren qué son sobre cuestiones teóricas.**
- **Pueden consultar también sobre sus dudas en los horarios de clases teóricos síncronos.**