

# Teoría de Lenguajes

# Teoría de la Programación

Clase 1: Introducción





# Leandro Ferrigno

- Ing en Informática
- Programador:
  - PHP
  - Javascript
  - C
  - Python
  - Java
  - C#
  - (y muy poco de muchos lenguajes más...)
- Algoritmos y Prog I, II, Lenguajes Formales

Contacto: [leanrafa\(at\)gmail.com](mailto:leanrafa(at)gmail.com)



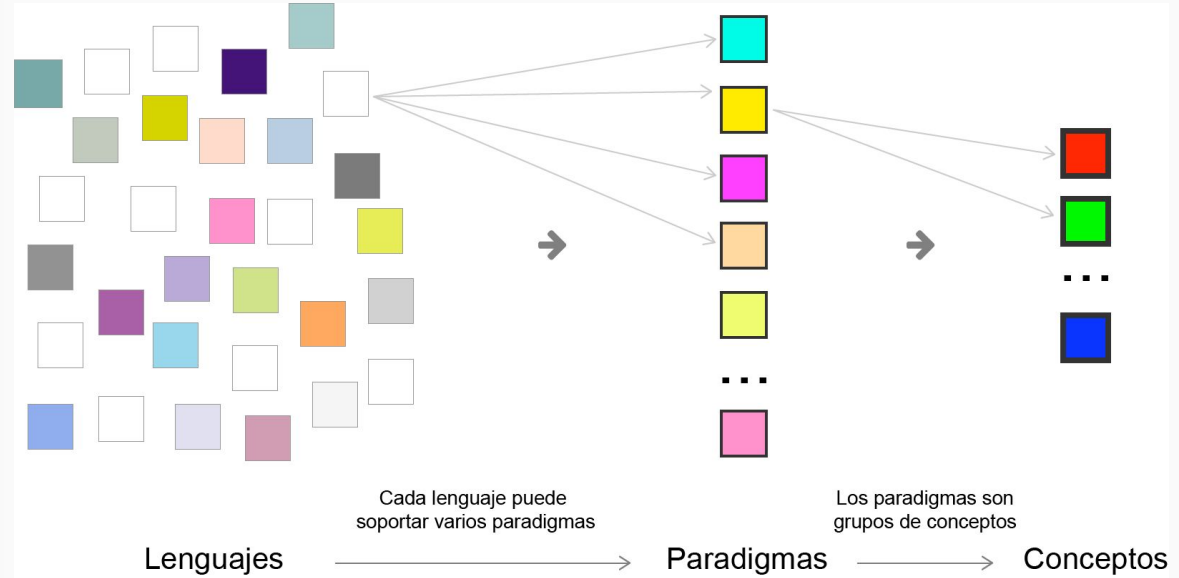
## Rosita Wachenchauzer

- Profesora FIUBA y UNTREF
- Secretaria de asuntos académicos FIUBA
- En FIUBA coordinadora de:
  - Algoritmos y programación I
  - Algoritmos y programación II
  - Teoría de lenguajes / de la programación
  - Teoría de algoritmos I
  - Teoría de algoritmos II

¿Qué saben hasta ahora?

# OBJETIVOS

## Conceptos de programación



# OBJETIVOS

Analizar  
lenguajes de  
programación

## Sintaxis:

Estructura o forma de los programas.  
¿Cómo se escribe?

## Semántica:


Significado. ¿Cómo se ejecuta?

## Pragmática:

¿Para qué propósitos es útil un lenguaje  
y para qué propósitos no lo es?



# Contenido

- Conceptos: Declarativo  Objetos
  - Variables
  - Funciones
  - Concurrencia
  - TDAs
  - Manejo de memoria
  - Entre otros
- Oz

## Bibliografía principal:

**Concepts, Techniques, and Models of Computer Programming** By Peter Van Roy and Seif Haridi



# Modalidad

- Clases
- Desarrollo de TP grupal
- Exposiciones grupales
- Ejercicios
- Final

# TP grupal + Exposiciones

- Grupo
- Lenguaje
- Contenido
- Presentismo en exposiciones (75%)
- Aprobación individual

# Información / Comunicación

**Slack de la materia**

<https://tdl-fiuba.slack.com/>

**Invitación: <https://goo.gl/dXrCu7>**

**Sitio Google: Contenido estático de la materia**

<https://sites.google.com/site/fiuba7531/>

# Conceptos básicos

# Modelo computacional

Sistema formal que define cómo se ejecutan los cálculos. Se define en términos de los conceptos que incluye.

Un modelo computacional es una definición más precisa de un paradigma de programación.

## Declarativo - Imperativo

*More is not better (or worse), just different*

# Modelo computacional

## Declarativo vs Imperativo



*Tiramisú funcional*

# Lenguaje de programación

- Sintaxis
- Semántica
- Pragmática

## Turing completo



Alan Turing (1912 -1954)  
1936: Máquina de Turing



Alonzo Church (1903-1995)  
1941:  $\lambda$  - Calculus

# Lenguaje de programación - Historia

- FORTRAN (1955), John Backus
- LISP (1958), McCarthy
- COBOL (1959), Grace Hopper.
- C (1972-1978), Kernighan & Ritchie
- Prolog (1972), Colmerauer & Roussel & Kowalski,
- Smalltalk (1972), Alan Kay & Dan Ingalls & Adele Goldberg
- Eiffel (1985), Meyer
- Erlang (1986), Armstrong
- Haskell (1990), Universidades Yale, Glasglow y Chalmers
- Python, Ruby, Java, Javascript, PHP, C#, Scala, CLojure, Go, Swift, ...



**m**  **art**

# Oz - Browse

```
{Browse 'Hola a todos y todas!'}
```

# Oz - Variables

```
A = 4  
B = 5  
C = A*B + 3  
{Browse C}
```

- **Nomenclatura**
- **Simple asignación**
- **Tipado**

## Oz - Variables II (scopes)

```
local A B C in  
  A = 4  
  B = 5  
  C = A*B + 3  
  {Browse C}  
end
```

```
declare A B C  
A = 4  
B = 5  
C = A*B + 3  
{Browse C}
```

## Oz - Variables III (Variables are just shortcuts for values)

```
local A B C in  
  A = 4  
  {Browse A}  
end
```

```
declare A B C  
  4 = A  
  {Browse A}
```

# Oz - Funciones

```
local Mayor A B M in  
  fun {Mayor X Y}  
    if (X > Y) then X else Y end  
  end
```

```
A = 30
```

```
B = 20
```

```
M = {Mayor A B}
```

```
{Browse M}
```

```
end
```

Definición

Aplicación

# Oz - Records

Estructura de datos para agrupar referencias

```
tree(key:I value:Y left:LT right:RT)
```

Etiqueta (label)

Características (features)

## Oz - Records II

```
local Alumno E in
  E = 80
  Alumno = persona(nombre: 'Roberto'
app: 'Sanchez' edad: E)
  {Browse Alumno.nombre}
  {Browse Alumno.edad}
end
```



# Oz - Tuplas

Una tupla es un Record en el cual sus características son enteros empezando por el 1

```
R= tree(key:2 value:15 left:A right:B)
L = tree(3 X A B)
{Browse {Arity R}}
{Browse {Arity L}}
```

# Oz - Binding / Partial values

## Binding

Asignarle valor a una variable.

## Partial Value:

Estructura de datos que puede tener unbound variables

## Variable - Variable binding

Cuando una variable se liga a otra variable

# Oz - Listas

Una lista es una tupla con simplemente 2 elementos, uno el primer elemento y el segundo el resto de la lista.

Construcción de una lista: [], |, o 'l'()

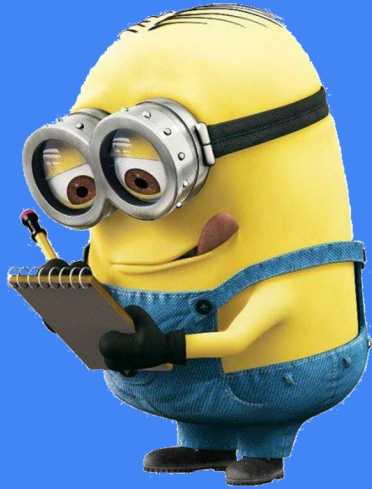
# Oz - Pattern Matching

Es una manera de acceder a los campos de una estructura de datos y obtener los valores

Un patrón “matchea” sobre un record cuando coincide:  
Width, Label & Features

Ejemplo: Length de una lista

# Ejercicios!



- 1) Devolver el máximo de una lista de enteros
- 2) A partir de una lista de números devolver una lista de los valores absolutos

# Oz - Evaluación perezosa (Lazy)

## Eager vs Lazy

En la evaluación perezosa los cálculos se hacen recién al momento en que se necesitan

```
fun lazy {Ints N}  
  N|{Ints N+1}  
end
```

# Oz - Alto orden (high order programming)

Funciones como ciudadanos de primera clase (first class)

Poder pasar funciones como parámetro a otras funciones se conoce como programación de alto orden

# Oz - Concurrency

## Threads

## Comunicación entre threads

```
local A B C in
  thread  A = B + C  end
  C = 4
  thread  B = 10  end
  {Browse A}
end
```



# Oz - Dataflow

¿Qué pasa si queremos usar una variable que todavía no está ligada a un valor?

- ¿Se lanza un error?
- ¿Se asume un valor por defecto?
- ¿Se detiene la ejecución?

# Oz - Estado explícito

## Explícito vs Implícito

Utilización de estados con celdas de memoria:

- NewCell
- @
- :=

```
C={NewCell 0} C:=@C+1 {Browse @C}
```

# Oz - Objetos

¿Con los conceptos que vimos hasta ahora, cómo definirían un objeto como lo conocen?

Al objeto lo vamos a definir por ahora como una función con memoria interna

# Oz - Clases

¿Cómo hacemos para tener más de una instancia de un objeto?

Definimos a la clase como una fábrica de objetos. Permite crear instancias.

# Oz - Indeterminismo y atomicidad

Concurrencia + estados  
=  
Indeterminismos



Atomicidad de operaciones con Locks

# Oz - Procedimientos

Los procedimientos son conceptos más básicos que las funciones.

Se utilizan variables no ligadas como parámetros

# Bibliografía

- **Concepts, Techniques, and Models of Computer Programming - Capítulo 1**, Peter Van Roy and Seif Haridi
- **Programming Paradigms for Dummies: What Every Programmer Should Know**, Peter Van Roy
- **Extras:**
  - [History of programming languages](#) - Wikipedia