Semana 2
Lenguajes de Programación
IIC1005
2018

Denis Parra
dparra@ing.puc.cl
Profesor Asistente
Departamento de Ciencia de la Computación
Escuela de Ingeniería, PUC Chile

Habemos Ayudantes

- Dalal Chahuan dcchahuan@uc.cl
- Vicente Dominguez vidominguez@uc.cl
- Daniela Flores diflores@uc.cl
- Felipe Garrido figarrido@uc.cl
- Antonio Ossa aaossa@uc.cl
- Florencia Valladares fvalladares1@uc.cl

PLAN SEMESTRAL

A	B	C	D	G	H	J	K
Week	Fecha semana	Clase Martes	Clase Jueves	Ayudantía	Control	Tarea Chica	Tarea Grande
I	6 - 8 Mar	Introduccion+terminal	Github+Jupyter				
II	13 - 15 Mar	Leng. Prog + Jupyter 2	Visualizacion + HCI	Jupyter Pandas			
III	20 - 22 Mar	Tecn Web HTML + CSS	Tecn Web JS	Jupyter Plots		TC1 Git+Shell	
IV	27 - 29 Mar	Arquitectura	SO+Redes	Web			
V	3 - 5 Abr	BD	BD	Web			TG1 Jupyter + Web
VI	10 - 12 Abr	Algoritmos	Ingenieria de Sotware		I1: 12Abr Web/HCI		
VII	17 - 19 Abr	ML	ML			TC2 BD (SQL+Mongo)	
VIII	24 - 26 Abr	ML	ML				
IX	3 may.	FERIADO	Guest: DL				TG2 ML
X	8 - 10 May	Computabilidad	Complejidad				
XI	15 - 17 May	Prog Logica	Prog Logica		I2: 16May IngSoft		
XII	22 - 24 Ma	BPM	BPM			TC3 Maq de Turing	
XIII	29 - 31 Ma	Guest: Criptomonedas	Guest: VR/AR				
XIV	5 - 7 Jun	Guest: CSCW	Guest: MOOC			TC4 BPM	
XV	12 -14 Jun	Guest: Miguel Nussb.	Guest: TBA		I3: 14Jun ML+IA		
XVI	19 - 21 Jun	Resumen Final					

Esta semana

- Ayudantía mañana: Python y jupyter notebooks
- Recibirán enunciado de tarea chica 1 y control 1
- Yo: Pasar lista: mínimo 70% de asistencia

Clase pasada: Jupyter notebooks

- Jupyter Notebook es una aplicación web que permite crear y compartir documentos que contienen código fuente, ecuaciones, visualizaciones y texto explicativo.

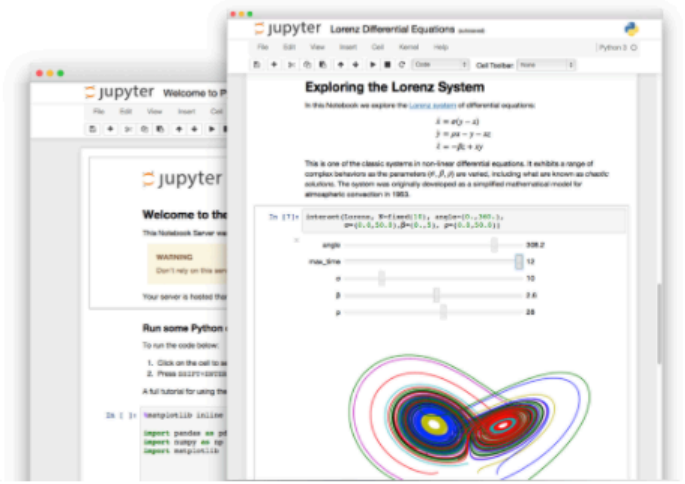
- Nos permite interactuar con código python.

- http://jupyter.org/

http://jupyter.org

jupyter

Install About Us Community Documentation NBViewer Widgets Blog



The Jupyter Notebook

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

Try it in your browser | Install the Notebook

Language of choice

The Notebook has support for over 40 programming languages, including Python, R, Julia, and Scala.

Share notebooks

Notebooks can be shared with others using email, Dropbox, GitHub and the Jupyter Notebook Viewer.

Interactive output

Your code can produce rich, interactive output: HTML, images, videos, LaTeX, and custom MIME types.

Big data integration

Leverage big data tools, such as Apache Spark, from Python, R and Scala. Explore that same data with pandas, scikit-learn, ggplot2, TensorFlow.

Instalación

- Para newbies: Anaconda

https://www.anaconda.com/download/

- Para más avanzados: Pip

First, ensure that you have the latest pip; older versions may have trouble with some dependencies:

```
pip3 install --upgrade pip
```

Then install the Jupyter Notebook using:

```
pip3 install jupyter
```

(Use `pip` if using legacy Python 2.)

Levantar servidor jupyter

- Tipear en terminal

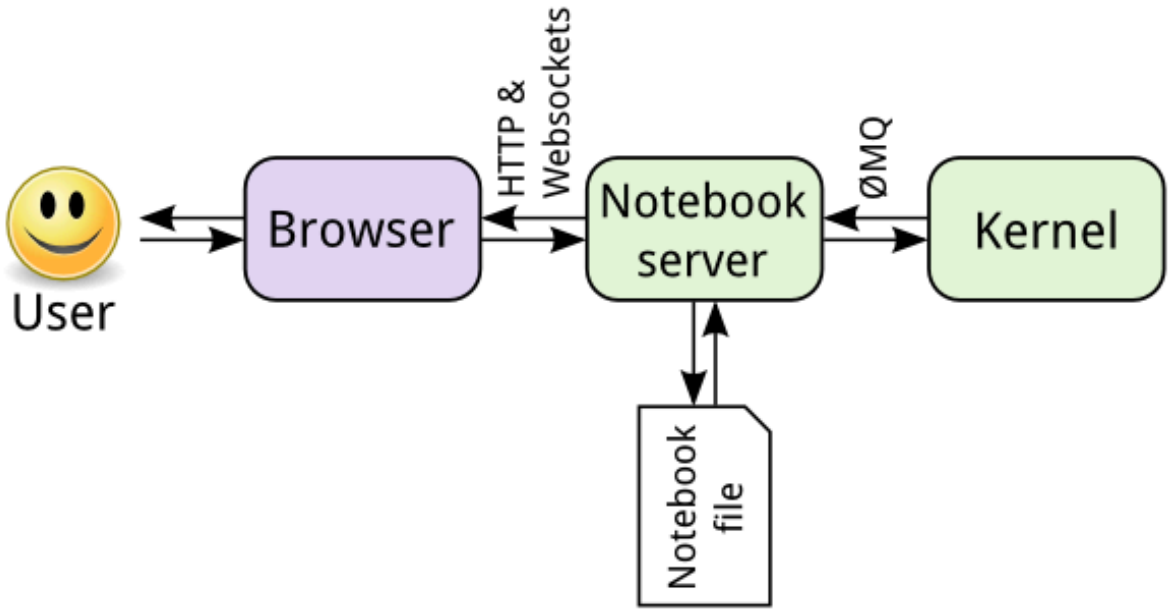
```
jupyter notebook
```

```
(py35) Deniss-MacBook-Pro-2:code denisparra$ jupyter notebook
[I 07:14:46.977 NotebookApp] Serving notebooks from local directory: /Volumes/Goog
leDrive/My Drive/PUC/IIC1005-2018-1/code
[I 07:14:46.977 NotebookApp] 0 active kernels
[I 07:14:46.977 NotebookApp] The Jupyter Notebook is running at: http://localhost:
8888/
[I 07:14:46.978 NotebookApp] Use Control-C to stop this server and shut down all k
ernels (twice to skip confirmation).
```

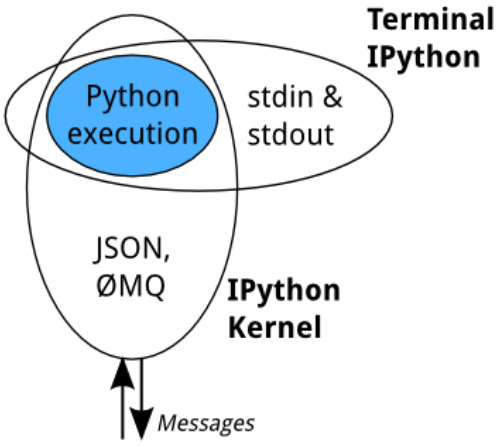
- Queda disponible en el puerto 8888 por defecto y lo accedemos con un cliente: el navegador web

¿Cómo funciona?

Jupyter Notebooks

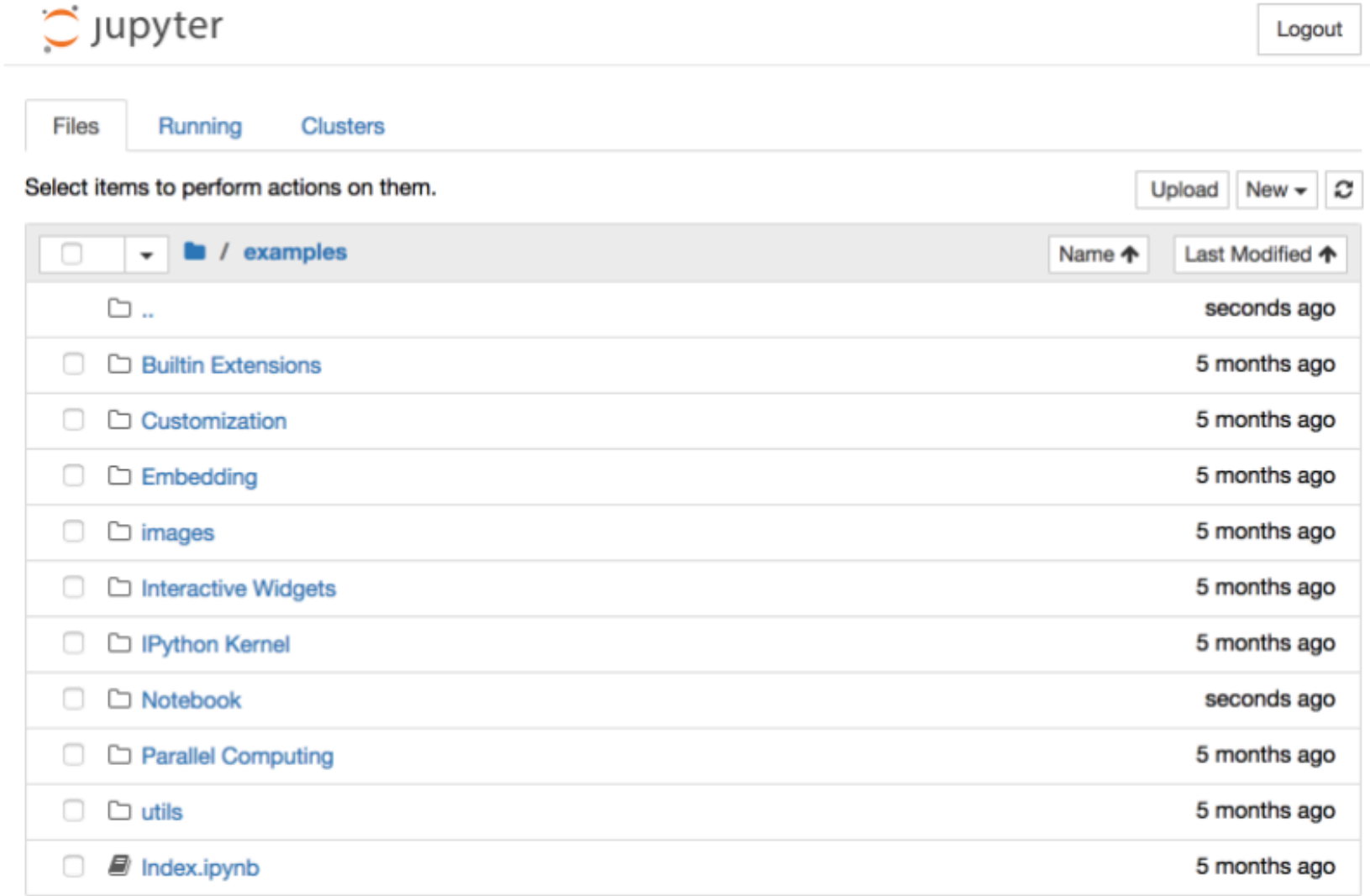


IPython




```
while True:
    code = input(">>> ")
    exec(code)
```

Fundamentos Básicos



Archivos .ipynb

- Cabecera (menú)

- **Markdown cells** - These are used to build a nicely formatted narrative around the code in the document. The majority of this lesson is composed of markdown cells.
- **Code cells** - These are used to define the computational code in the document. They come in two forms: the *input cell* where the user types the code to be executed, and the *output cell* which is the representation of the executed code. Depending on the code, this representation may be a simple scalar value, or something more complex like a plot or an interactive widget.
- **Raw cells** - These are used when text needs to be included in raw form, without execution or transformation.

Archivos .ipynb y Notebooks

- Celdas

- **Markdown cells** - These are used to build a nicely formatted narrative around the code in the document. The majority of this lesson is composed of markdown cells.
- **Code cells** - These are used to define the computational code in the document. They come in two forms: the *input cell* where the user types the code to be executed, and the *output cell* which is the representation of the executed code. Depending on the code, this representation may be a simple scalar value, or something more complex like a plot or an interactive widget.
- **Raw cells** - These are used when text needs to be included in raw form, without execution or transformation.

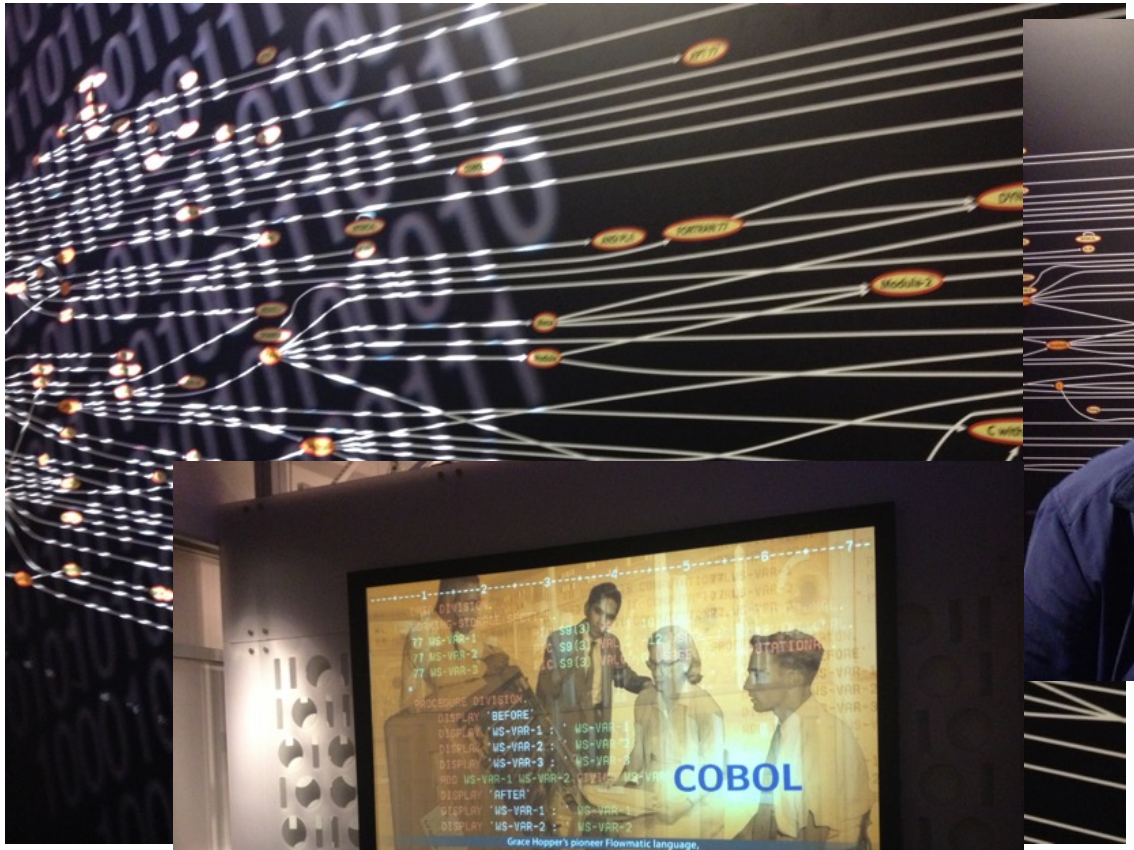
Ejemplo en vivo

```
In [1]: import numpy as np
        import pandas as pd

In [2]: myiris = pd.read_csv('iris-iic1005.csv', header=0)

In [3]: myiris.head()
```

Out[3]:

	sepallength	sepalwidth	petallength	petalwidth	class	class_numeric
0	5.1	3.5	1.4	0.2	Iris-setosa	0
1	4.9	3.0	1.4	0.2	Iris-setosa	0
2	4.7	3.2	1.3	0.2	Iris-setosa	0
3	4.6	3.1	1.5	0.2	Iris-setosa	0
4	5.0	3.6	1.4	0.2	Iris-setosa	0

...



COBOL
Grace Hopper's pioneer Flowmatic language,

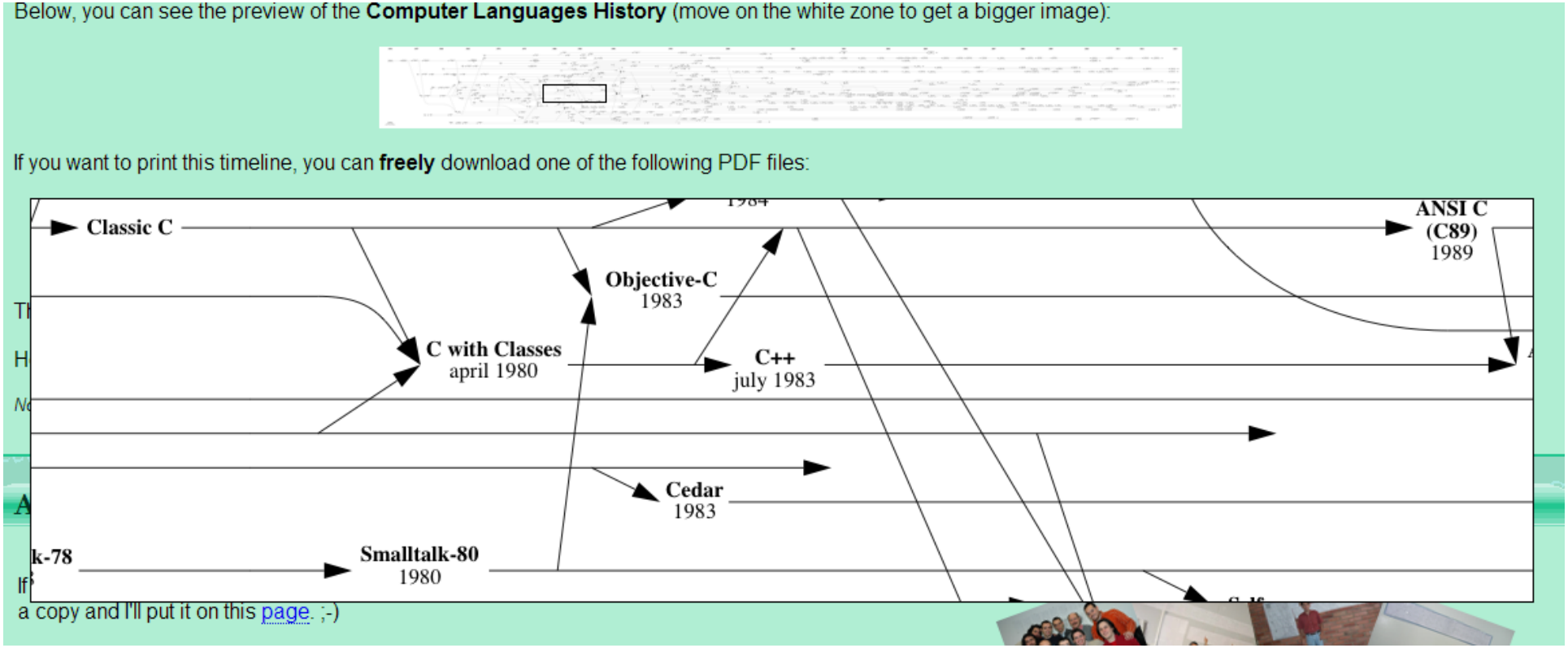

KISS
MY BITS
COMPUTERWORLD

Hoy: Lenguajes de Programacion

- Algo de Historia, y luego …
  - Assembly
  - Fortran
  - C
  - Prolog
  - C++
  - Java / C#
  - Objective-C
  - Python
  - Ruby
  - PHP
  - Javascript
  - R/Matlab (Computacion Cientifica)
  - SQL

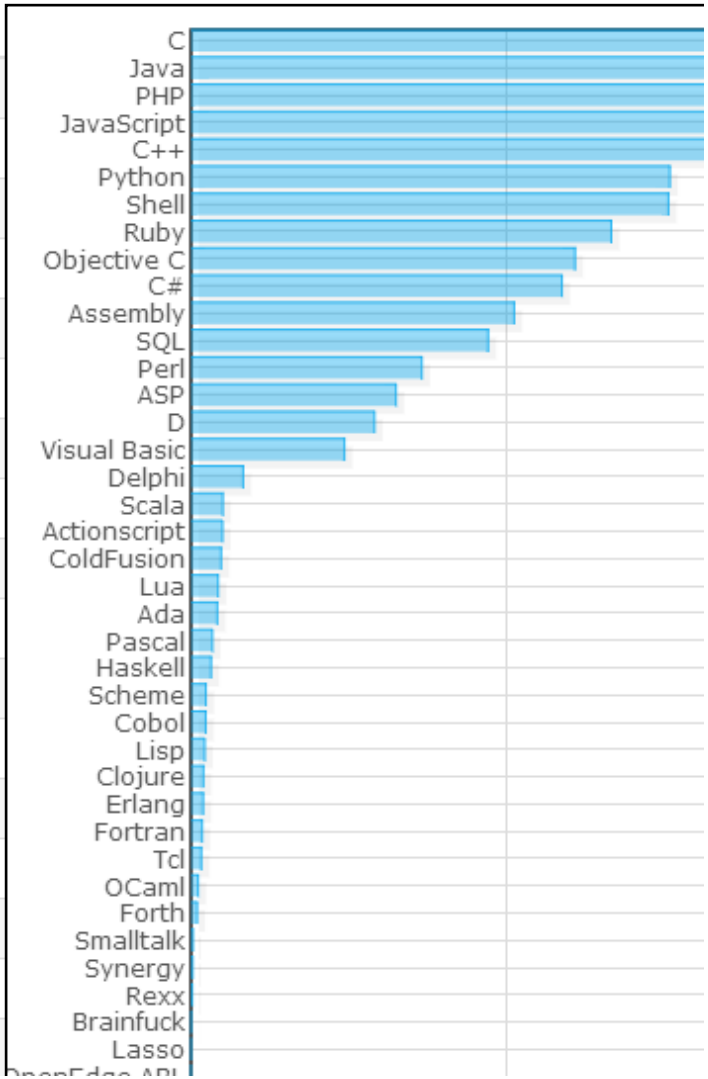
Un poquito de historia

http://www.levenez.com/lang/



TIOBE vs. LangPop

Oct 2015	Oct 2014	Change	Programming Language	Ratings	Change
1	2	↑	Java	19.543%	+6.04%
2	1	↓	C	16.190%	-1.47%
3	4	↑	C++	5.749%	+0.88%
4	5	↑	C#	4.825%	+0.08%
5	8	↑	Python	4.512%	+2.18%
6	7	↑	PHP	2.561%	-0.38%
7	13	⇈	Visual Basic .NET	2.462%	+0.71%
8	12	⇈	JavaScript	2.292%	+0.52%
9	9		Perl	2.247%	+0.13%
10	16	⇈	Ruby	1.825%	+0.70%
11	11		Delphi/Object Pascal	1.637%	-0.18%
12	31	⇈	Assembly language	1.573%	+1.16%
13	14	↑	Visual Basic	1.515%	-0.05%
14	3	⇊	Objective-C	1.419%	-8.68%
15	19	⇈	Swift	1.277%	+0.52%
16	20	⇈	Pascal	1.194%	+0.47%
17	27	⇈	MATLAB	1.159%	+0.55%



¿Cuánto importa la popularidad?

- Es importante, pero hay lenguajes que parecen poco populares y son muy usados en ciertas áreas:
- En Bancos y grandes compañíias: COBOL
- Aplicaciones matemáticas: FORTRAN
- Etc…

Supongan este requerimiento

<<If somebody came to me and wanted to pay me a lot of money to build a large scale message handling system that really had to be up all the time, could never afford to go down for years at a time, I would unhesitatingly choose …. to build it in.>>

¿Qué lenguaje elegirían?

Supongan este requerimiento

<<If somebody came to me and wanted to pay me a lot of money to build a large scale message handling system that really had to be up all the time, could never afford to go down for years at a time, I would unhesitatingly choose **ERLANG** to build it in.>>

Tim Bray, director of Web Technologies at Sun Microsystems, keynote at OSCON in July 2008

ERLANG

- Primera versión de Erlang implementada en ProLog
- ¿Quién lo usa?
  - Amazon.com: Para su BD SimpleDB
  - WhatsApp: Para soportar el servicio de mensajería, con 2 millones de usuarios conectados por servidor
  - Bet365: Para el servicio de apuestas inPlay
  - ….

Erlang <-> Relación con ProLog

- ¿Qué tiene de especial Prolog que no tienen otros lenguajes que han usado antes?

```
-module(count_to_ten).
-export([count_to_ten/0]).

count_to_ten() -> do_count(0).

do_count(10) -> 10;
do_count(Value) -> do_count(Value + 1).
```

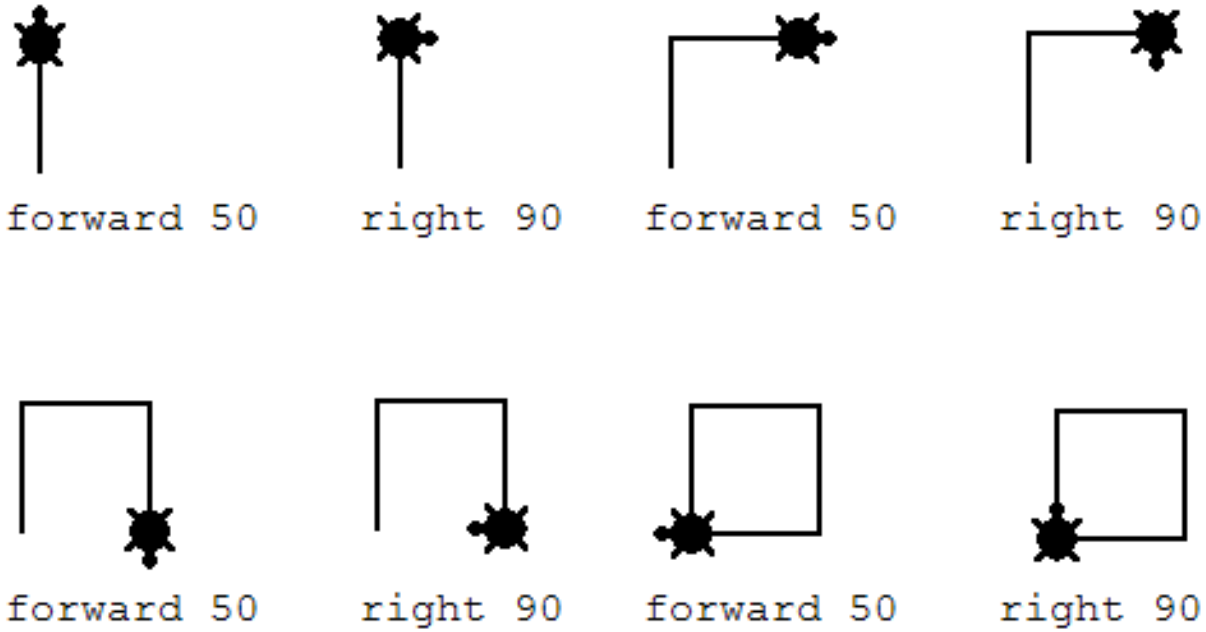
Relación con ProLog

- Hot Code Loading (Hot Swaping)

```
%% A process whose only job is to keep a counter.
%% First version
-module(counter).
-export([start/0, codeswitch/1]).

start() -> loop(0).

loop(Sum) ->
  receive
     {increment, Count} ->
        loop(Sum+Count);
     {counter, Pid} ->
        Pid ! {counter, Sum},
        loop(Sum);
     code_switch ->
        ?MODULE:codeswitch(Sum)
        % Force the use of 'codeswitch/1' from the latest MODULE version
  end.

codeswitch(Sum) -> loop(Sum).
```

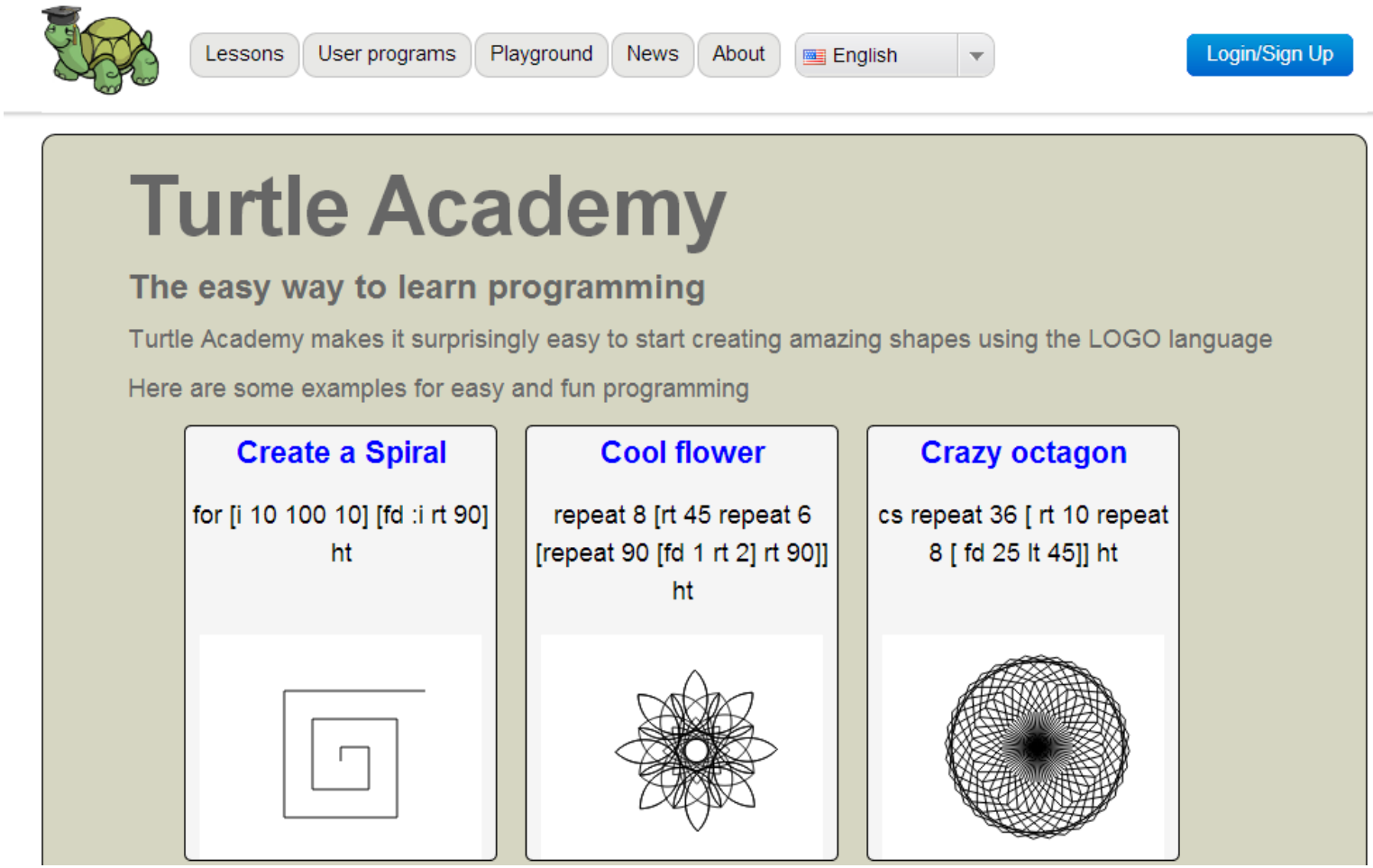
LOGO

- Creado en 1969 por Seymour Papert, con propósito pedagógico (falleció esta semana)



Por si alguien quiere probar

- Turtle academy http://turtleacademy.com/



Cuándo fueron creados estos lenguajes?

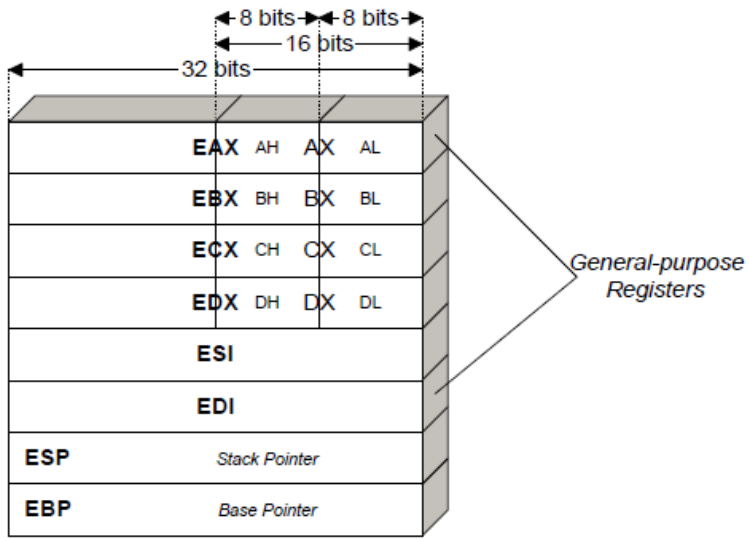
- Assembly
- FORTRAN
- C
- C++
- Java
- Javascript
- Python
- C# (2001)

Cuándo fueron creados estos lenguajes?

- Assembly (195x)
- FORTRAN (1953)
- C (1969)
- C++ (1980)
- Java (1995)
- Javascript (1995)
- Python (1991)
- C# (2001)

Ejemplo de Assembly x86

- **Así es como sumas dos números:**
- Poner primer numero en registro
- Poner segundo numero en registro
- Sumar los registros
- Retornar resultado




```
;n1 db 3; n2 db 7
mov eax, 3 ; podria ser mov eax, [n1]
mov ecx, 7; podria ser mov eax, [n2]
add eax, ecx
ret
```

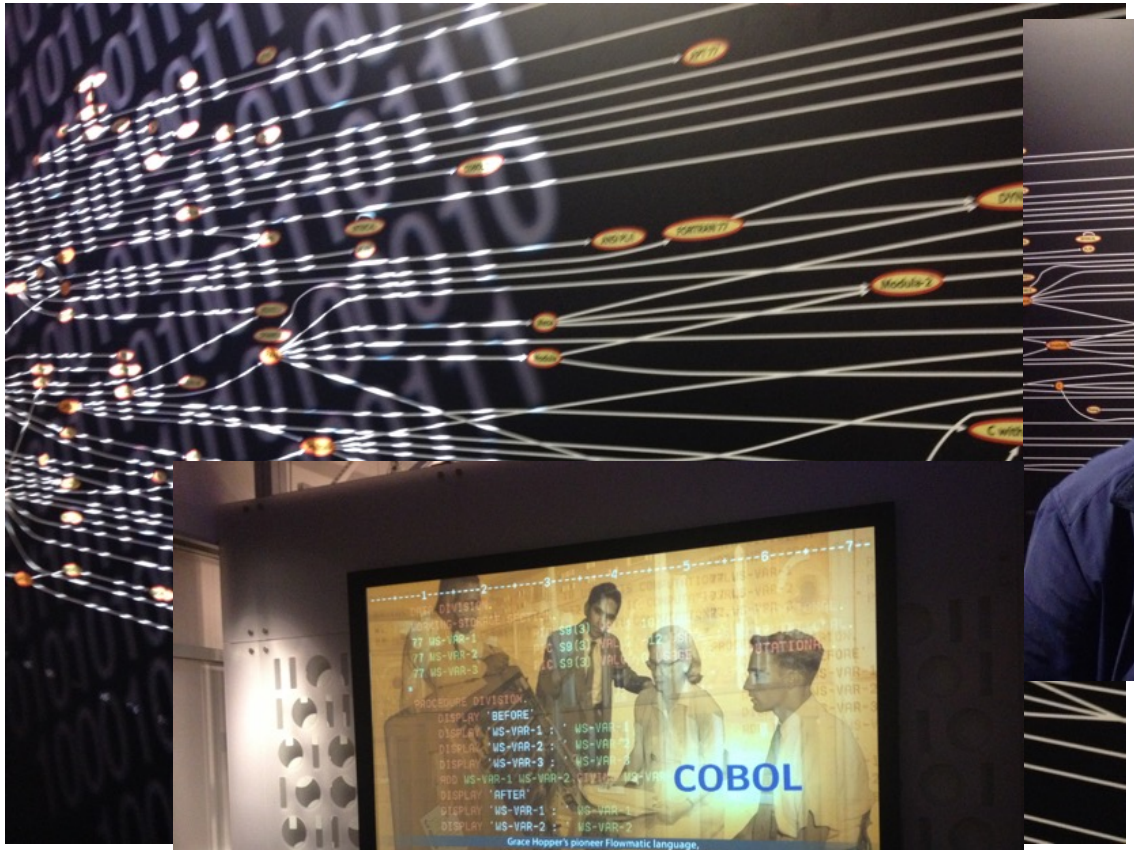
```
// equivalente en C
int a = 3;
int c = 7;
a += c;
return a;
```

Considerando la Arquitectura: Assembly

- Es un lenguaje de bajo nivel, y no porque sea de mala calidad ;-)

- Los lenguajes “Assembly” consideran directamente la arquitectura del equipo (numero y tamaño de los registros) y por lo tanto no son portables a otras arquitecturas (como Java, por ejemplo)

...



COBOL
Grace Hopper's pioneer Flowmatic language,

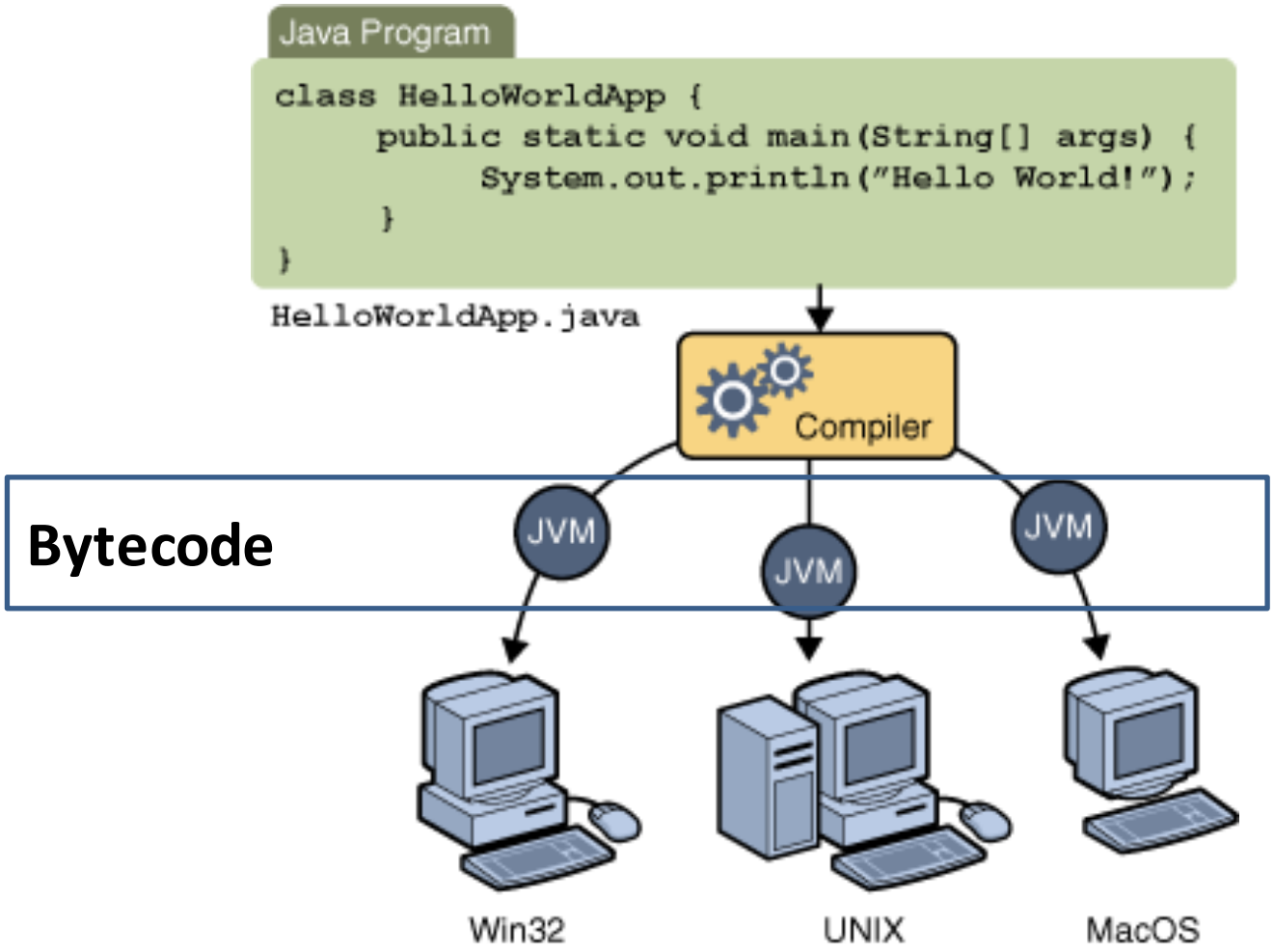

KISS
MY BITS
COMPUTERWORLD

Primera diferencia: Lenguaje Compilado versus Interpretado

- Compilado: FORTRAN, Pascal, C, C++
- Interpretado: Python, Ruby
- Compilado es generalmente más rapido porque apuntan directamente a la máquina/arquitectura en la cual se ejecutan.
- Interpretado tiende a ser más portable.
- Versión de lenguajes “interpretados” es más fáciles de crear porque escribir compiladores es algo difícil.

¿Cómo funciona JAVA?

- Con la Java Virtual Machine



POO: Hello World en C++ y Java

========== C++ ==========

```
#include <iostream>
int main() {
        //comentario
        std::cout << "Hello World!";
}
```

========== JAVA ==========

```
public class HelloWorld {       //comentario
        public static void main(String[] args) {
        System.out.println("Hello,  World");
        }
}
```

C / C++ / Java

Manejo de memoria: C

```
1#include <stdlib.h> // needed for malloc and free!
2int *p_int = malloc(sizeof(*p_int));
3// use p_int
4free( p_int );
```

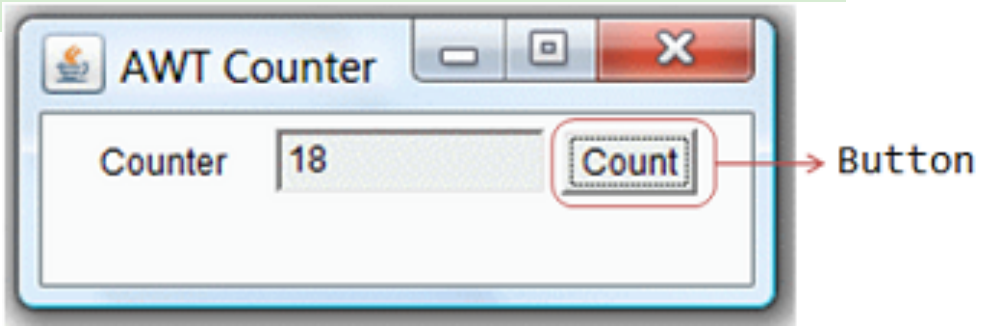
Manejo de memoria: C++

```
1int *p_int = new int;
2// use p_int
3delete p_int;
```

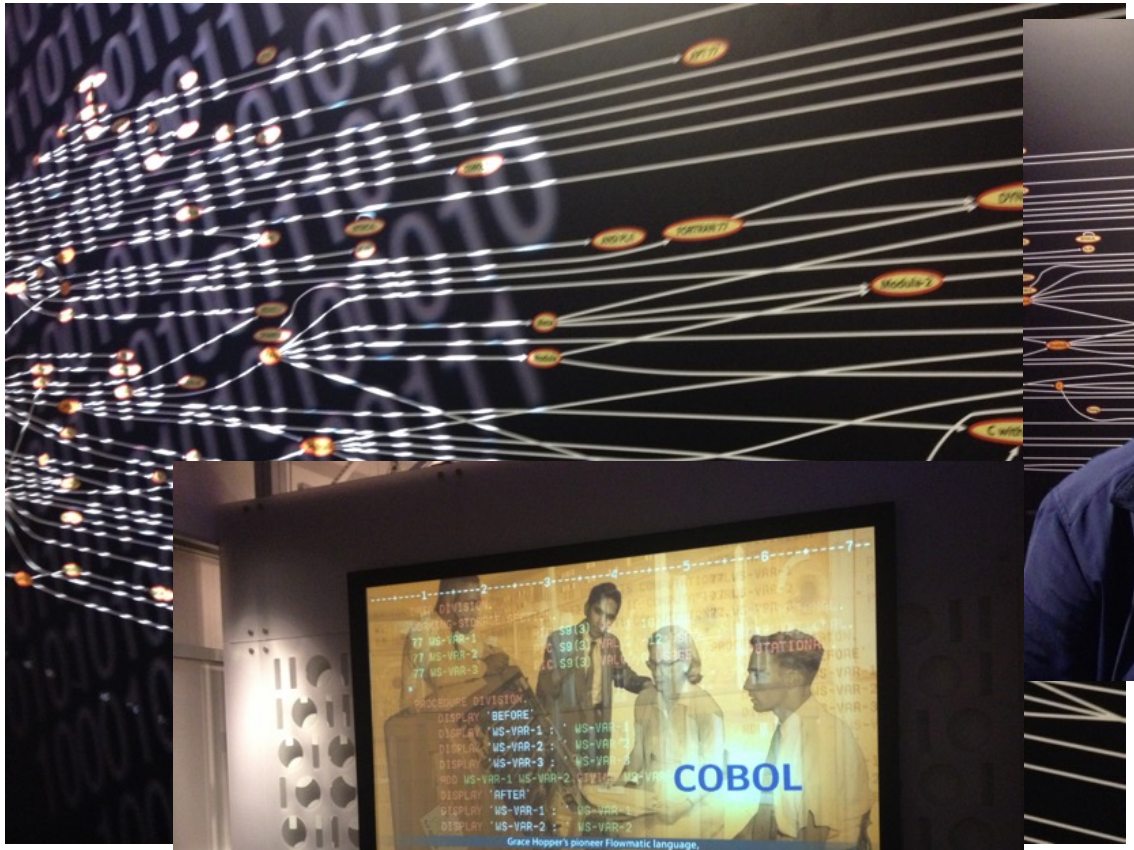
Manejo de memoria Java: Garbage Collector!

Java

```
import java.awt.Frame;
// Using Frame class in package java.awt
// A GUI program is written as a subclass of Frame - the top-level container
// This subclass inherits all properties from Frame, e.g., title, icon, buttons, content-pane
public class MyGUIProgram extends Frame {
// Constructor to setup the GUI components
public MyGUIProgram() {
......
}

// Other methods ...... ......
// The entry main() method
public static void main(String[] args) {
// Invoke the constructor (to setup the GUI) by allocating an instance
new MyGUIProgram();
}

}
```



...



COBOL
Grace Hopper's pioneer Flowmatic language,


KISS
MY BITS
COMPUTERWORLD

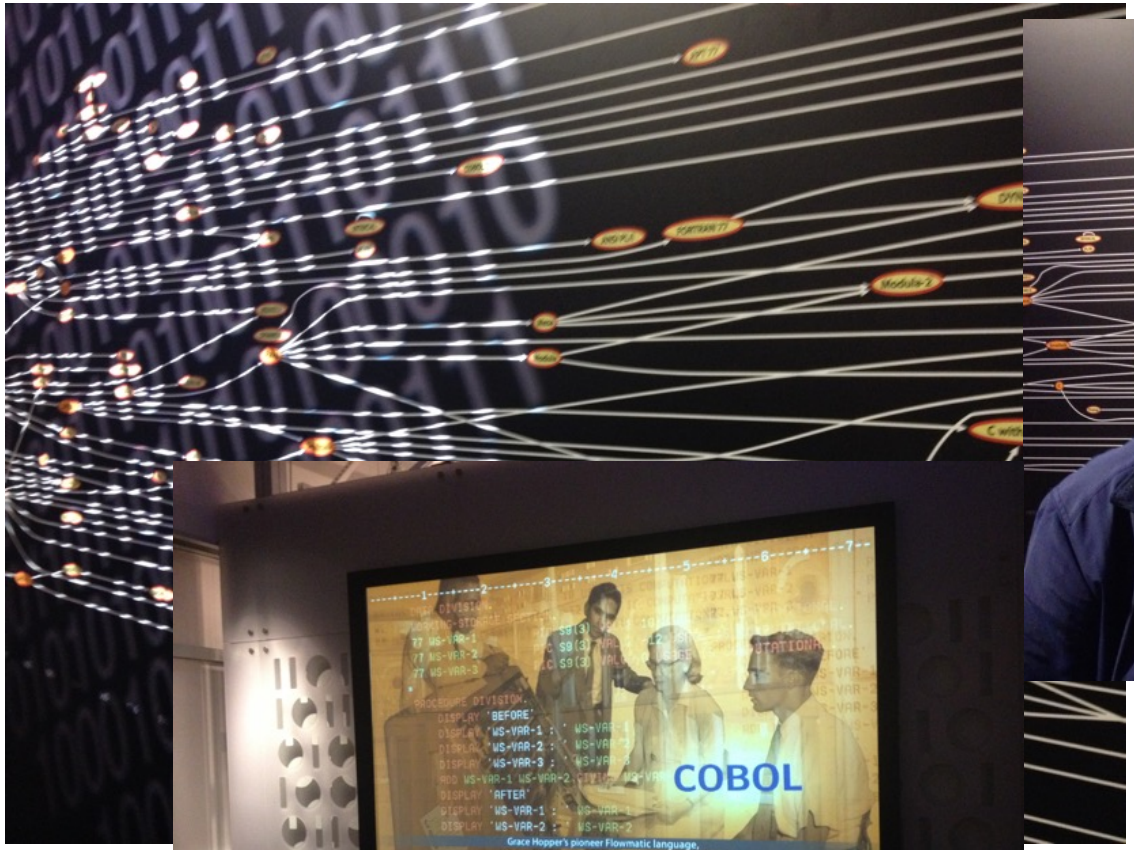
PROLOG

- Logic programming language: asociado a lógica e inteligencia artifical.
- Lenguaje declarativo que expresa relaciones y permite realizar inferencias

```
mother_child(trude, sally).
father_child(tom, sally).
father_child(tom, erica).
parent_child(X, Y) :- father_child(X, Y).
parent_child(X, Y) :- mother_child(X, Y).
sibling(X, Y) :- parent_child(Z, X), parent_child(Z, Y).
```

?- sibling(sally, erica). Yes

...



COBOL
Grace Hopper's pioneer Flowmatic language,


KISS
MY BITS
COMPUTERWORLD

SQL

- **Structured Query Language** orientado especialmente para DBMS relacionales
- En la clase de Information Retrieval vimos algunas procedimientos simples (bag-of-words model, TF-IDF) para buscar informacion no estructurada, pero cuando la informacion esta almacenada de forma estructurada, SQL es el estandar para consultas.

SQL – 3 ejemplos

Considerando estas tablas:

TABLA CUSTOMERS

CustomerID	CustomerName	ContactName	Country
1	Alfreds Futterkiste	Maria Anders	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mexico

TABLA ORDERS

OrderID	CustomerID	OrderDate
10308	2	1996-09-18
10309	37	1996-09-19
10310	77	1996-09-20

SQL – EJEMPLO 1: SELECT

- SELECT [campos] FROM [tabla] WHERE [condiciones]

SELECT CustomerID, ContactName

FROM Orders

WHERE Country = “Mexico”

Paradigma de programacion en SQL

- ¿Le dijeron en algun momento a la instrucción de SQL cómo ir a buscar la información?
- NO -> **SQL es un tipo de lenguaje declarativo**
- ¿Qué otros paradigmas de lenguajes de programación hay?
- (y por que nos interesa saber…)

Paradigmas de Programación

- Lenguajes más comunes responden a varios paradigmas

Paradigm	Description	Main characteristics	Related paradigm(s)	Critics	Examples
Imperative	Computation as statements that *directly* change a program state(data fields)	Direct assignments, common data structures,global variables		Edsger W. Dijkstra,Michael A. Jackson	C, C++,Java, PHP,Python
Structured	A style of imperative programming with more logical program structure	Structograms, indentation, either no, or limited use of, goto statements	Imperative		C, C++, Java
Procedural	Derived from structured programming, based on the concept of modular programming or the *procedure call*	Local variables, sequence, selection,iteration, and modularization	Structured, imperative		C, C++, Lisp,PHP, Python
Functional	Treats computation as the evaluation of mathematical functions avoiding state and mutable data	Lambda calculus, compositionality, formula, recursion, referential transparency, no side effects			Erlang, Haskell, Lisp, Clojure, Scala, F#

Por que sería util forzar programación funcional? rt(x) = rt(y) if x = y

```
globalValue = 0;

integer function rq(integer x)
begin
     globalValue = globalValue + 1;
     return x + globalValue;
end

integer function rt(integer x)
begin
     return x + 1;
end
```

Si escribiésemos un loop con la F(x)

- Un compilador podría detectar una función dentro de un loop y optimizar la forma de referenciarla y escribirla en código de máquina SOLAMENTE si la función no depende del estado de ejecución del programa

```
While (i < 1000)
    rq(i)
```

Depende de variable global definida en tiempo de ejecución

```
While (i < 1000)
    rt(i)
```

Este sí cumple con transparencia referencial

Otros lenguajes - LoLCode

-
-

Now with doge!

LOLCODE is an esoteric programming language inspired by the funny things that cats say on the Internet.

Learn more about the language

lci is a correct, portable, fast, and precisely documented interpreter for LOLCODE written in C.

Download source

GitHub project page

Problems? Check out the mailing list or file a bug report.

LOLCODE

Hagamos un “Hola Mundo” en LOLCODE

LOLCODE 1

```
HAI 1.2

VISIBLE "Hai world"

KTHXBYE
```

LOLCODE 2

- Declarar e inicializar una variable
- Mostrarla en Pantalla

I HAS A VARIABLE ITZ <var>

LOLCODE 3

- Agregar comentarios

BTW

LOLCODE 4

- Solicitar al usuario input desde teclado

GIMMEH

LOLCODE 5

- Una bifurcación (IF)

```
…, O RLY?
        YA RLY
                …
        NO WAI
                …
OIC
```

LOLCODE 6

- CONTADOR

IM IN YR LOOP

…

IM OUTTA YR LOOP

…otros

- Switch… case

<expression>
WTF?
  OMG <value literal>
    <code block>
 [OMG <value literal>
    <code block> …]
 [OMGWTF
    <code block>]
OIC

Volviendo a la realidad

- La tarea 1 grande incluye
- Incluye programación en:
  - Python
  - HTML
  - CSS
  - Javascript

¿Cómo preparme rápidamente?

- Venir a ayudantías
- CodeCademy: Basic Web Projects
  - https://www.codecademy.com/en/tracks/projects
- W3Schools: tutorial javascript
  - https://www.w3schools.com/js/default.asp
- Libros de referencia:
  - Guia paso a paso: You Don't Know Javascript
  - Tradicional: Javascript The Definitive Guide

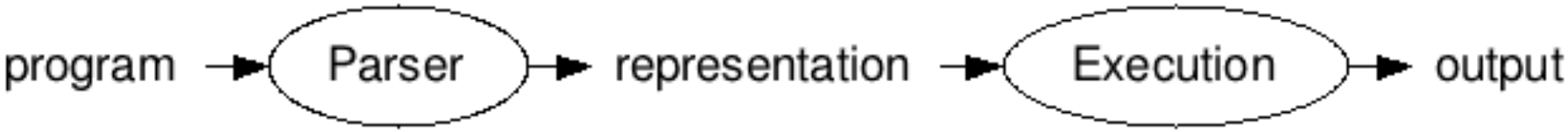
Gracias!

¿Pero cómo empezar?

- El mejor lugar: la consola del navegador que me permite ejecutar código en javascript.
- Click con el botón derecho sobre cualquier parte de la página y seleccionar “Inspect”
  - Hola Mundo
  - alert
  - Cambiar estilos

¿Y cómo escribo mi propio lenguage?

- Necesitas un parser (sintaxis) y luego otro programa que implementa la ejecución




```
>> program = "(begin (define r 3) (* 3.141592653 (* r r)))"

>>> parse(program)
['begin', ['define', 'r', 3], ['*', 3.141592653, ['*', 'r', 'r']]]


>>> eval(parse(program))
28.274333877
```

code from (How to Write a (Lisp) Interpreter (in Python))
http://norvig.com/lispy.html

¿Y cómo escribo mi propio lenguage?

- Necesitas un parser (sintaxis) y luego otro programa que implementa la ejecución
- Debes partir definiendo tu lenguaje: símbolos, reglas… y debe ser Context-free

$$G = (\{S\}, \{a, b\}, P, S)$$

$S \rightarrow aSb$

$S \rightarrow ab$

Equivalente a $\{a^n b^n : n \geq 1\}$

Si el tiempo da

- Ver ejemplo en

http://en.wikipedia.org/wiki/Context-free_grammar

Algebraic expressions [edit]

Here is a context-free grammar for syntactically correct infix algebraic expressions in the variables x, y and z:

1. S → x
2. S → y
3. S → z
4. S → S + S
5. S → S - S
6. S → S * S
7. S → S / S
8. S → (S)

This grammar can, for example, generate the string

(x + y) * x - z * y / (x + x)

as follows:

S (the start symbol)
→ S - S (by rule 5)
→ S * S - S (by rule 6, applied to the leftmost S)
→ S * S - S / S (by rule 7, applied to the rightmost S)

Paso de ejecución

Here is the definition of `eval`. Each of the nine cases in the table above has a line or two or three here, and the definition of `eval` needs nothing but those nine cases. The `eval` function takes two arguments: an expression, `x`, and an *environment*, `env`. An environment is a mapping from variable names to their values and will be covered in depth in the next section.

```
def eval(x, env=global_env):
    "Evaluate an expression in an environment."
    if isa(x, Symbol):              # variable reference
        return env.find(x)[x]
    elif not isa(x, list):          # constant literal
        return x
    elif x[0] == 'quote':           # (quote exp)
        (_, exp) = x
        return exp
    elif x[0] == 'if':              # (if test conseq alt)
        (_, test, conseq, alt) = x
        return eval((conseq if eval(test, env) else alt), env)
    elif x[0] == 'set!':            # (set! var exp)
        (_, var, exp) = x
        env.find(var)[var] = eval(exp, env)
    elif x[0] == 'define':          # (define var exp)
        (_, var, exp) = x
        env[var] = eval(exp, env)
    elif x[0] == 'lambda':          # (lambda (var*) exp)
        (_, vars, exp) = x
        return lambda *args: eval(exp, Env(vars, args, env))
    elif x[0] == 'begin':           # (begin exp*)
        for exp in x[1:]:
            val = eval(exp, env)
        return val
    else:                           # (proc exp*)
```

Sugiero chequerar tambien: https://sites.google.com/a/bodik.org/cs164/directory/pa1

¡Gracias!