

Semana 2

Lenguajes de Programación

IIC1005

2018

PLAN SEMESTRAL

A	B	C	D	E	F
Week	Fecha semana	Clase Martes	Clase Jueves	Control	Enunciados
I	7 - 9 Ago	Introduccion	Github+terminal+Jupyter		
II	14 - 16 Ago	Leng. Prog + Jupyter 2	Visualización		TC1 Git + Shell
III	21 - 23 Ago	Tecn Web HTML + CSS	Tecn Web JS		TG1 Jupyter + Web
IV	28 - 30 Ago	Arquitectura	SO		
V	4 - 6 Sep	BD	BD		
VI	11 - 13 Sep	Algoritmos	Ingenieria de Sotware		TC2 BD
VII	18 - 20 Sep	FERIADO	ML		TG2 ML
VIII	25 - 27 Sep	ML	ML	I1: 28 Sept Web/Arg/SO/BD	
IX	2 - 4 Oct	ML	Guest: DL		
X	9 - 11 Oct	Computabilidad	Complejidad		TC3 Turing
XI	16 - 18 Oct	Prog Logica	Prog Logica		
XII	23 - 25 Oct	BPM	BPM II	I2: 24 Oct ML / Comp	TC4 BPM
XIII	30 - 1 Nov	Innovación Tecnológica	FERIADO		
XIV	6 - 8 Nov	Guest: CSCW	MOOC		
XV	13 -15 Nov	Criptografía y aplicaciones	Guest: Mobile & Cloud	I3: 14 Jun BPM / Prog Log	
XVI	20 - 22 Nov	Resumen Final	Guest: Miguel Nussb.		

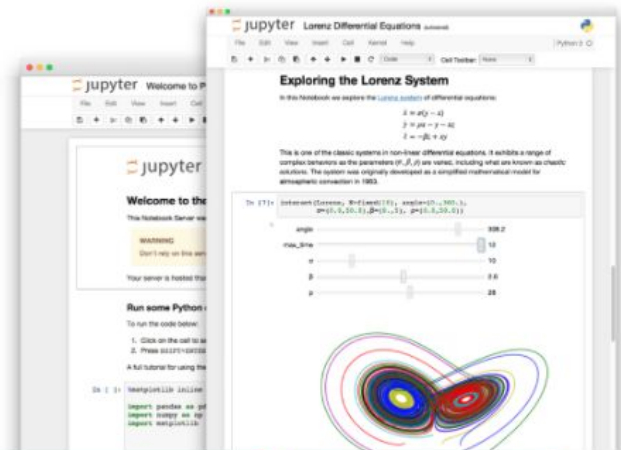
Avisos Varios

- No hay ayudantía mañana (es feriado)
- Subiremos un video tutorial sobre git de aquí al final de la semana
- Recibirán el enunciado de tarea chica 1 entre el domingo y el lunes
- Habemos Syllabus
<https://github.com/Exploratorio-DCC-PUC/Syllabus>
- 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

[Install](#)[About Us](#)[Community](#)[Documentation](#)[NBViewer](#)[Widgets](#)[Blog](#)[Try it in your browser](#)[Install the Notebook](#)

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.



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

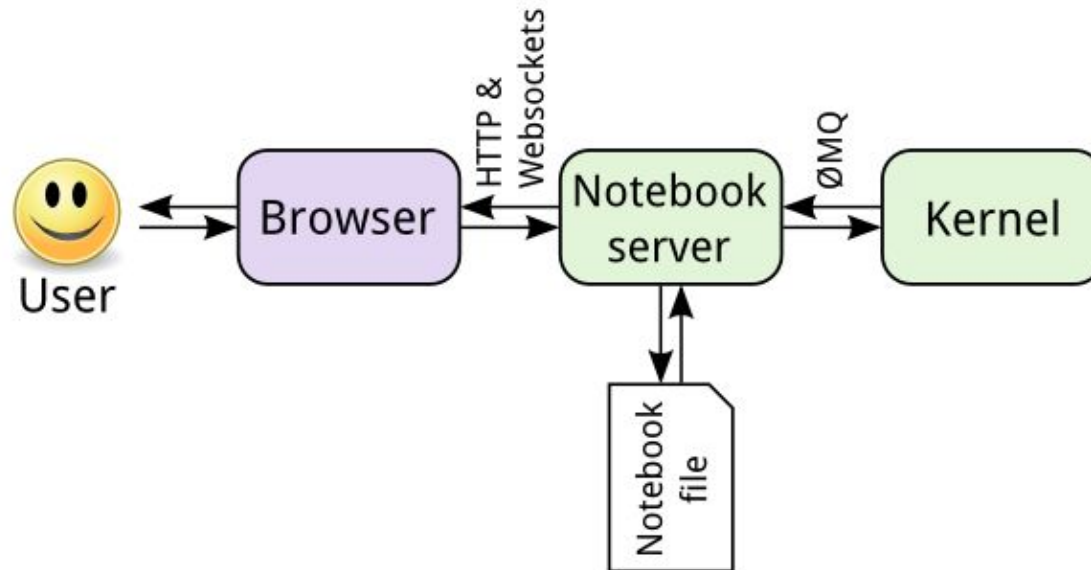
```
jupyter notebook
```

```
(py35) Deniss-MacBook-Pro-2:code denisparra$ jupyter notebook
[I 07:14:46.977 NotebookApp] Serving notebooks from local directory: /Volumes/GoogleDrive/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 kernels (twice to skip confirmation).
```

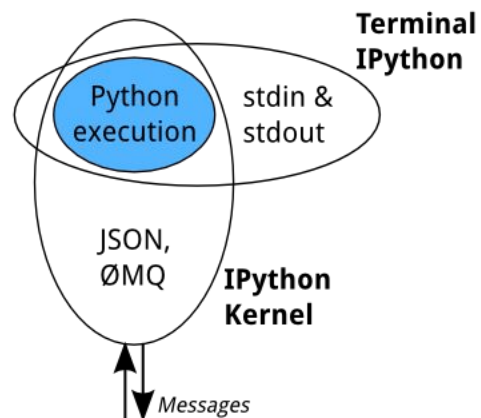
- Queda disponible en el puerto 8888 por defecto y lo accedemos con un cliente: el navegador web

¿Cómo funciona?

Jupyter
Notebooks



IPython



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


Fundamentos Básicos

[Logout](#)[Files](#)[Running](#)[Clusters](#)

Select items to perform actions on them.

[Upload](#)[New ▾](#)

<input type="checkbox"/>	<input type="text" value="examples"/>	Name ↑	Last Modified ↑
<input type="checkbox"/>	..		seconds ago
<input type="checkbox"/>	Builtin Extensions		5 months ago
<input type="checkbox"/>	Customization		5 months ago
<input type="checkbox"/>	Embedding		5 months ago
<input type="checkbox"/>	images		5 months ago
<input type="checkbox"/>	Interactive Widgets		5 months ago
<input type="checkbox"/>	IPython Kernel		5 months ago
<input type="checkbox"/>	Notebook		seconds ago
<input type="checkbox"/>	Parallel Computing		5 months ago
<input type="checkbox"/>	utils		5 months ago
<input type="checkbox"/>	Index.ipynb		5 months ago


Archivos .ipynb

- Cabecera (menú)

 jupyter Untitled (unsaved changes)



File Edit View Insert Cell Kernel Help

 Python 3 

          Code  Cell Toolbar: None 

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.

I'm a **markdown** cell.

In [2]: `print("I'm a code cell")`

I'm a code cell

I'm a ****raw**** cell

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

Hoy: Lenguajes de Programación

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

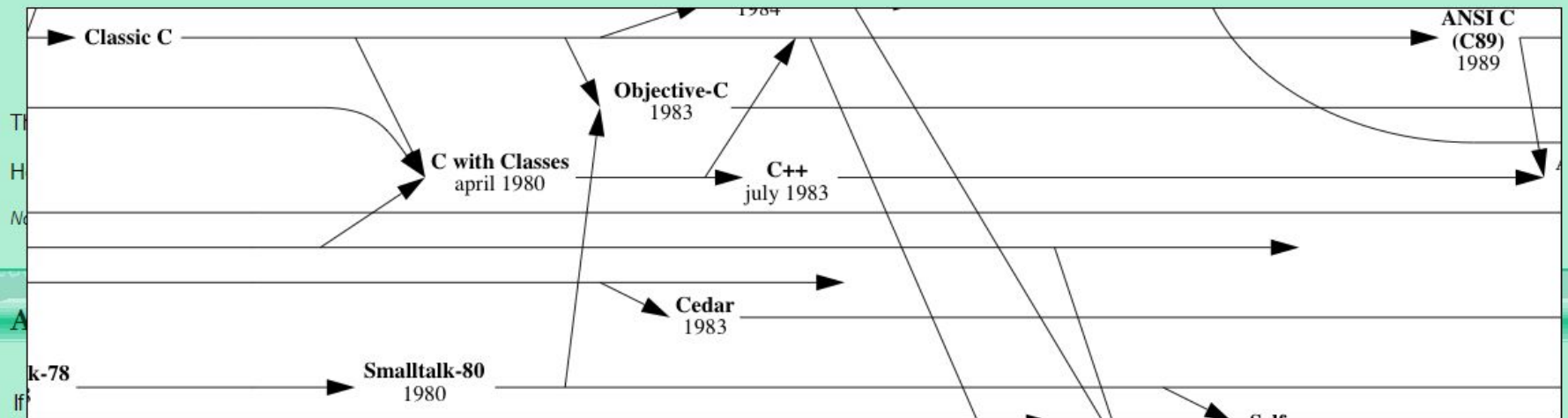
Un poquito de historia

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

Below, you can see the preview of the **Computer Languages History** (move on the white zone to get a bigger image):



If you want to print this timeline, you can **freely** download one of the following PDF files:



If you want a copy and I'll put it on this [page](#). ;-)

Lenguajes más populares

<https://octoverse.github.com/>



¿Cuánto importa la popularidad?

- Es importante, pero hay lenguajes que parecen poco populares y son muy usados en ciertas áreas:
- En Bancos y grandes compañías: 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).
```

PROLOG

- Logic programming language: asociado a lógica e inteligencia artificial.
- 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

LOGO

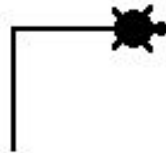
- Creado en 1969 por Seymour Papert, con propósito pedagógico



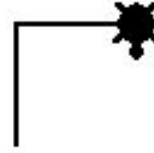
`forward 50`



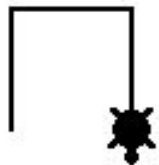
`right 90`



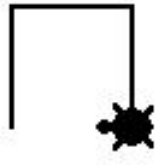
`forward 50`



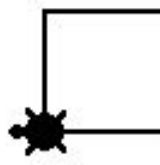
`right 90`



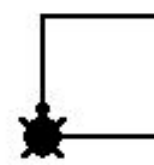
`forward 50`



`right 90`



`forward 50`



`right 90`



Por si alguien quiere probar

- Turtle academy <http://turtleacademy.com/>

[Lessons](#)[User programs](#)[Playground](#)[News](#)[About](#)[English](#)[Login/Sign Up](#)

Turtle Academy

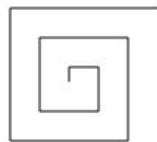
The easy way to learn programming

Turtle Academy makes it surprisingly easy to start creating amazing shapes using the LOGO language

Here are some examples for easy and fun programming

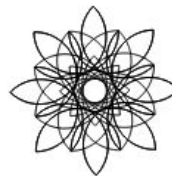
Create a Spiral

```
for [i 10 100 10] [fd :i rt 90]  
ht
```



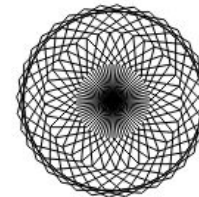
Cool flower

```
repeat 8 [rt 45 repeat 6  
[repeat 90 [fd 1 rt 2] rt 90]]  
ht
```



Crazy octagon

```
cs repeat 36 [ rt 10 repeat  
8 [ fd 25 lt 45]] ht
```



Cuándo fueron creados estos lenguajes?

- Assembly
- FORTRAN
- C
- C++
- Java
- Javascript
- Python
- C#



Cuándo fueron creados estos lenguajes?

- Assembly (1949)
- FORTRAN
- C
- C++
- Java
- Javascript
- Python
- C# (2001)



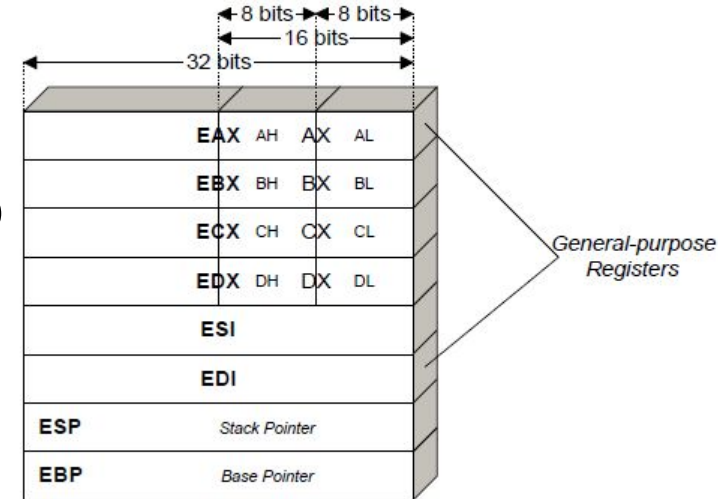
Cuándo fueron creados estos lenguajes?

- Assembly (1949)
- 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 ecx, [n2]
add eax, ecx
ret
```

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

Considerando la Arquitectura: Assembly

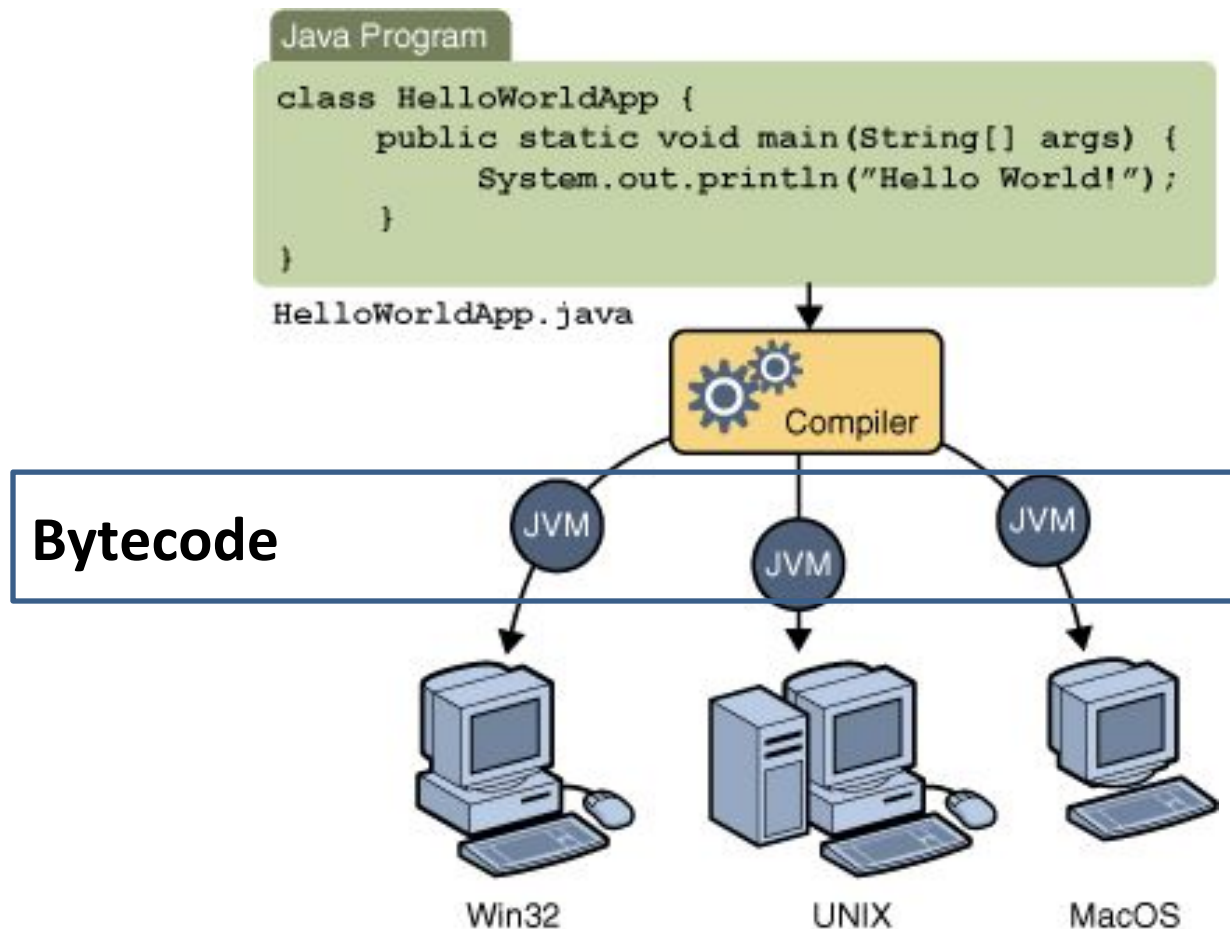
- Es un lenguaje de bajo nivel, y no porque sea de mala calidad ;-)
- Los lenguajes “Assembly” consideran directamente la arquitectura del equipo (número y tamaño de los registros) y por lo tanto no son portables a otras arquitecturas (como Java, por ejemplo)

Primera diferencia: Lenguaje Compilado versus Interpretado

- Compilado: FORTRAN, Pascal, C, C++
- Interpretado: Python, Ruby
- Compilado es generalmente más rápido 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 );
```

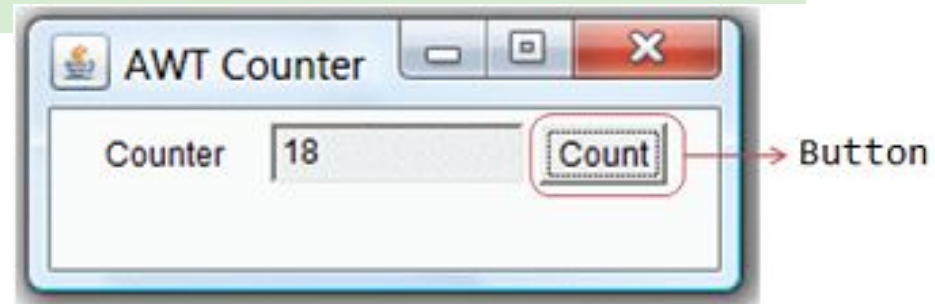
Manejo de memoria: C++

```
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();  
    }  
}
```



SQL

- **Structured Query Language** orientado especialmente para DBMS relacionales
- Cuando la información está almacenada de forma estructurada, SQL es el estándar 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 programación en SQL

- ¿Le dijeron en algún 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

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

Por que sería útil 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 escribiéramos 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 prepararme 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!