

# Introducción a la Programación

Fundamentos de Programación  
CCPG1001

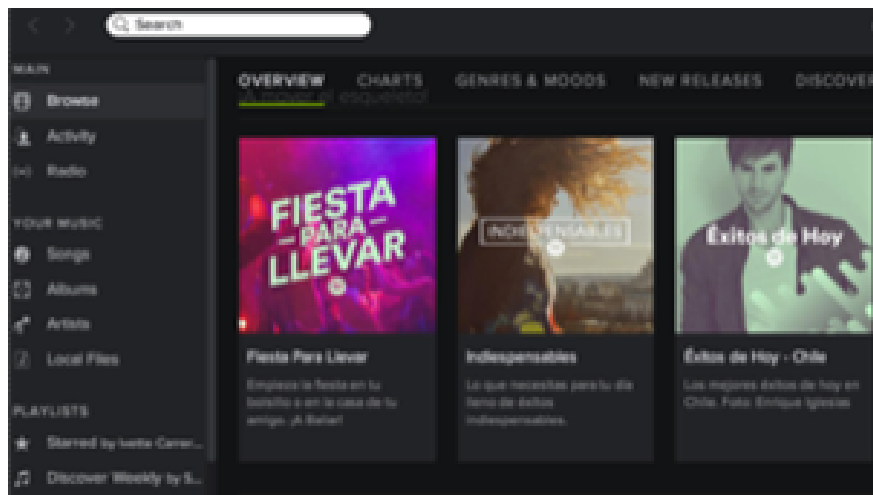
# **Introducción**

# Qué es una computadora?

- Es un dispositivo electrónico que sirve para manipular datos. Una computadora permite almacenar, recuperar y procesar datos.
- Es un dispositivo capaz de realizar cálculos y tomar decisiones lógicas mucho más rápido que los humanos.

# Cómo le decimos a la computadora qué tiene que hacer?

- Los usuarios a través de diferentes ***programas*** (instrucciones) le dicen a la computadora que hacer.



- Los programadores a través de un **lenguaje de programación** construyen esos programas

# Por qué aprender a programar?

Entre otras cosas me permite:

- **Automatizar tareas repetitivas y ser más productivo**
- Crear herramientas que otros usan (trabajo de programador)
- Ganar dinero

Otras razones:

- Fomenta la creatividad
- Crear cosas de interés personal
- Es divertido

“Todos en este país deberían aprender a programar una computadora porque te enseña a pensar”

– Steve Jobs





# **Conceptos básicos de lenguajes de programación**

# Lenguaje de programación

- Un lenguaje de programación es un **lenguaje formal** diseñado para realizar procesos que pueden ser llevados a cabo por máquinas como las computadoras.
- Pueden usarse para crear **programas** que controlen el comportamiento físico y lógico de una máquina, para expresar **algoritmos** con precisión, o como modo de comunicación humana.

# Lenguaje de programación

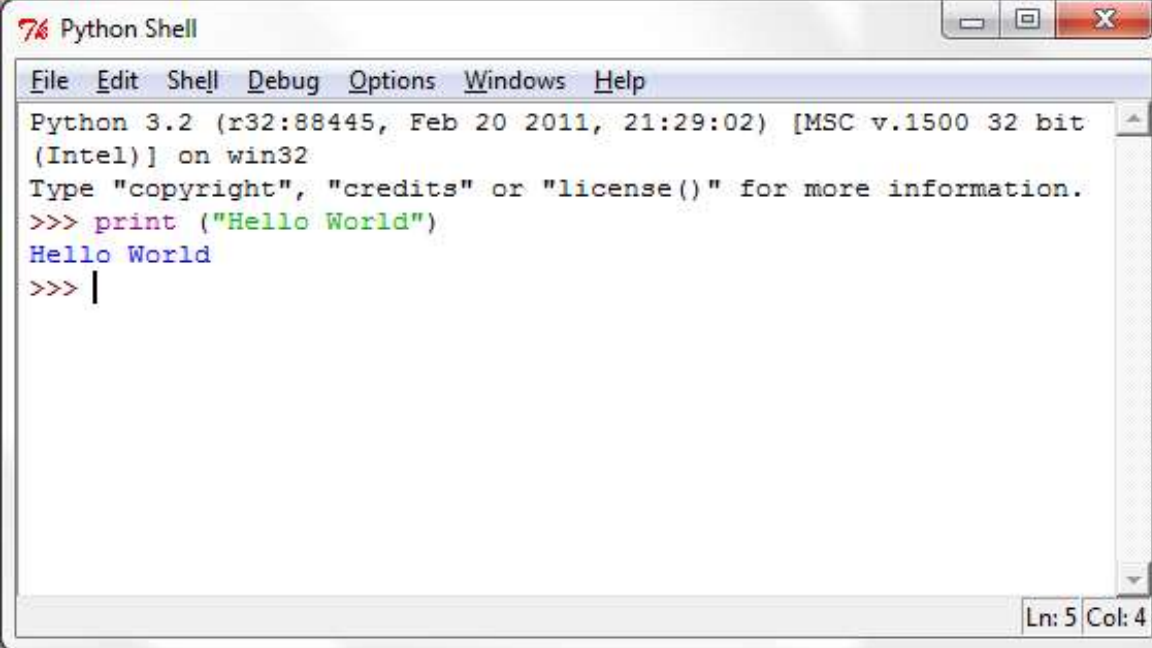
- Un lenguaje de programación es un **lenguaje formal** diseñado para realizar procesos que pueden ser llevados a cabo por máquinas como las computadoras.
- Pueden usarse para crear **programas** que controlen el comportamiento físico y lógico de una máquina, para expresar **algoritmos** con precisión, o como modo de comunicación humana.

# Programas!

# Qué es un programa?

- Secuencia ordenada de instrucciones para resolver un problema
- Es un conjunto de líneas de código de nuestra inteligencia dentro de una computadora

# Hello World!

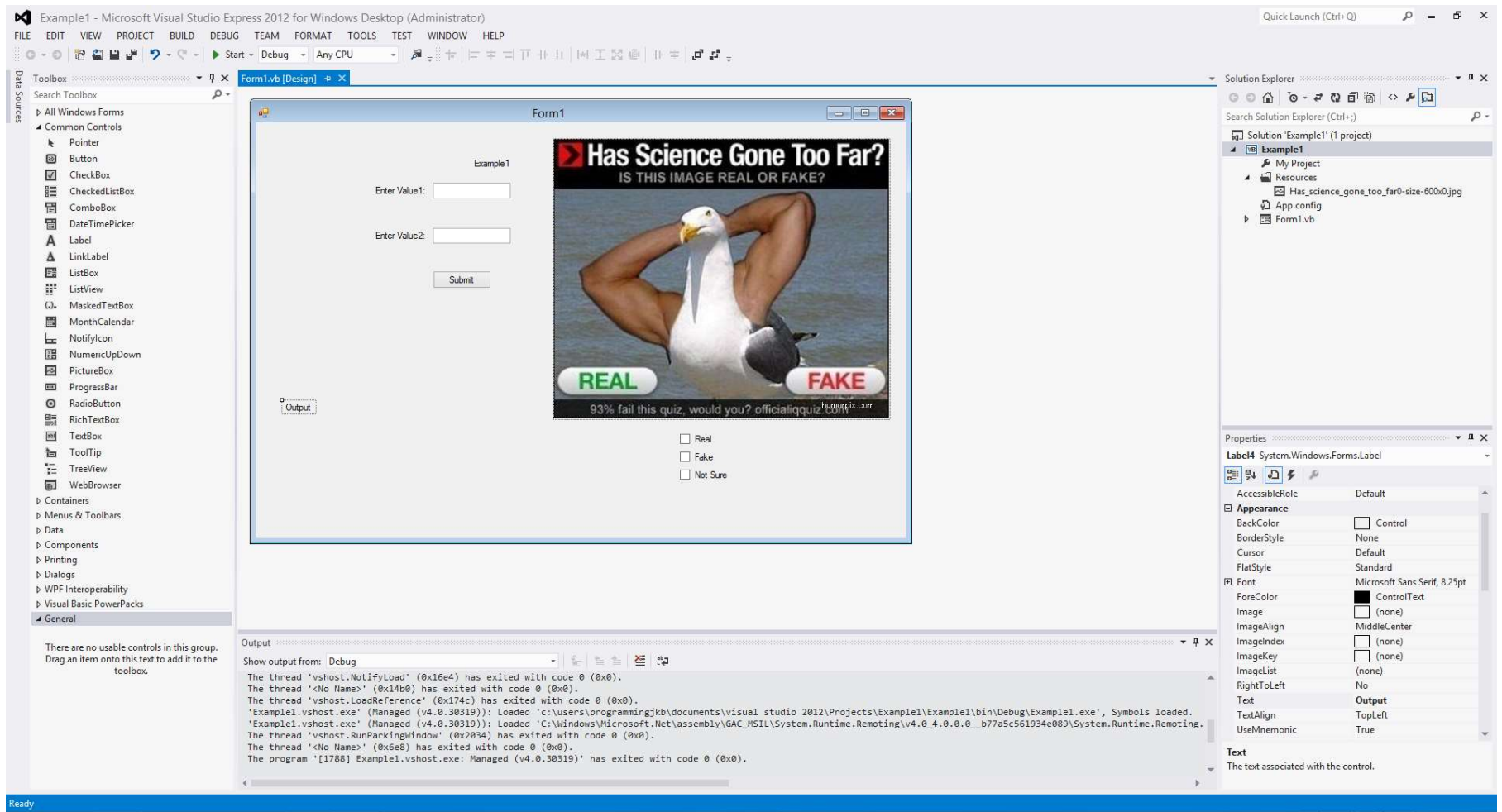


A screenshot of a Python Shell window. The window has a title bar that says "Python Shell" with standard minimize, maximize, and close buttons. Below the title bar is a menu bar with options: File, Edit, Shell, Debug, Options, Windows, and Help. The main text area contains the following text: "Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit (Intel)] on win32", "Type 'copyright', 'credits' or 'license()' for more information.", and a prompt ">>>". The user has entered the command "print ('Hello World')", and the shell has responded with "Hello World". The prompt ">>>" is followed by a vertical bar cursor. At the bottom right of the window, a status bar shows "Ln: 5 Col: 4".

```
Python Shell
File Edit Shell Debug Options Windows Help
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500 32 bit
(Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print ("Hello World")
Hello World
>>> |
```

Ln: 5 Col: 4

# Programas por diversion!

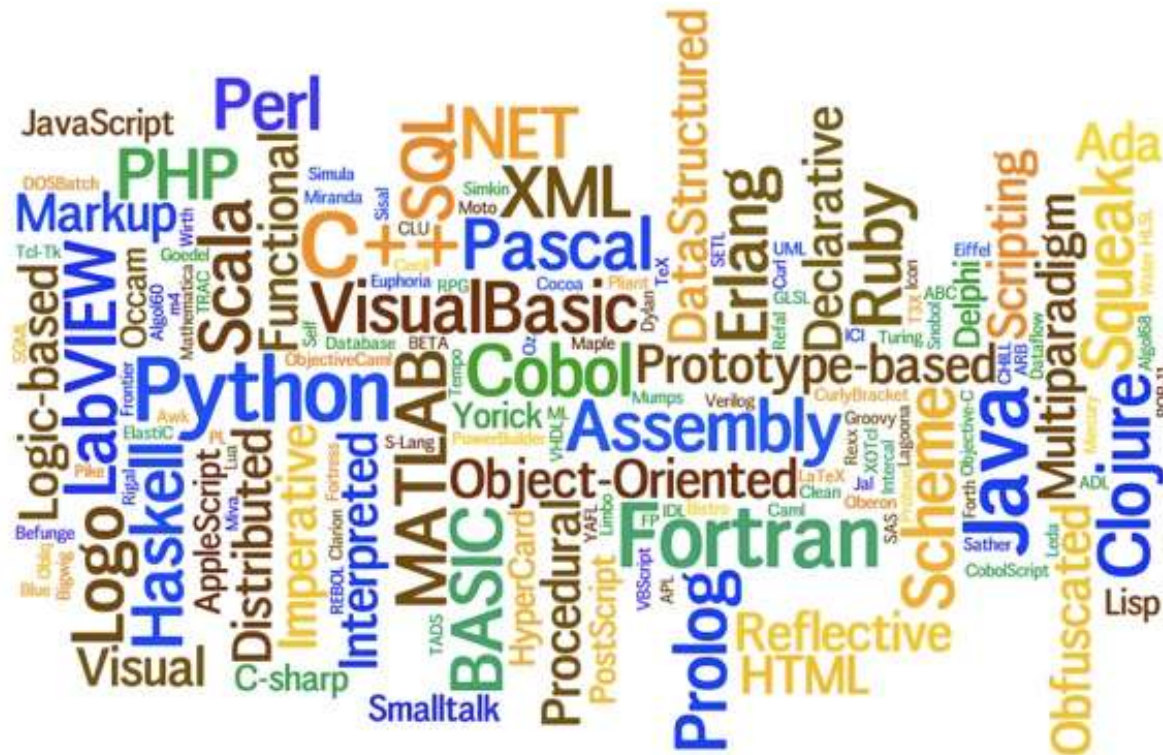




# Programas mas complejos



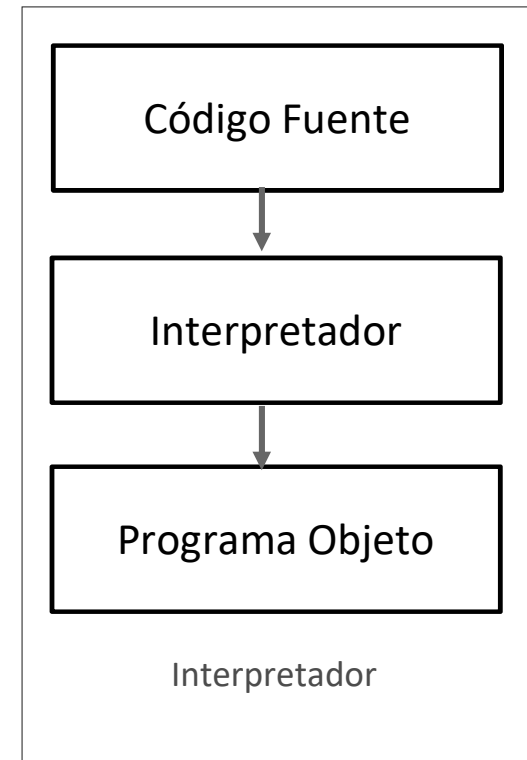
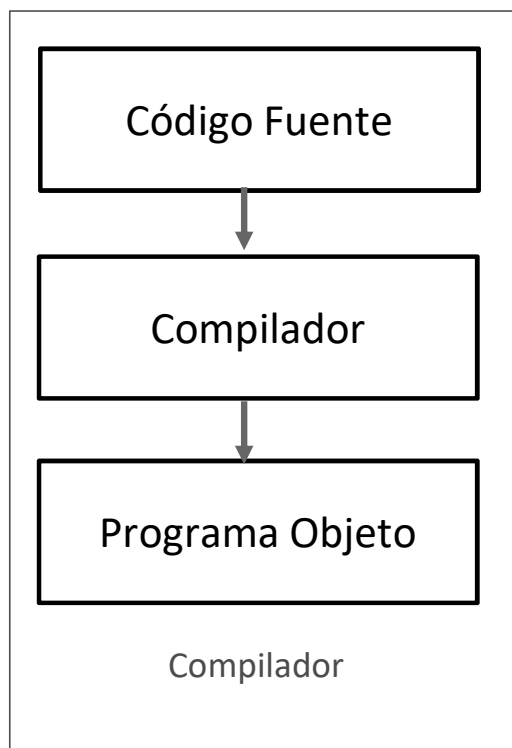
# Lenguajes de programación



# Tipos de lenguajes

- Lenguaje de Alto Nivel: representación simbólica, parecido al inglés y notación matemática
- Lenguaje de Bajo Nivel: control directo sobre el hardware y condicionado a la arquitectura de la computadora
- Lenguaje de Maquina: circuitos micro programables tales como un microprocesador

# Traducción a lenguaje maquina

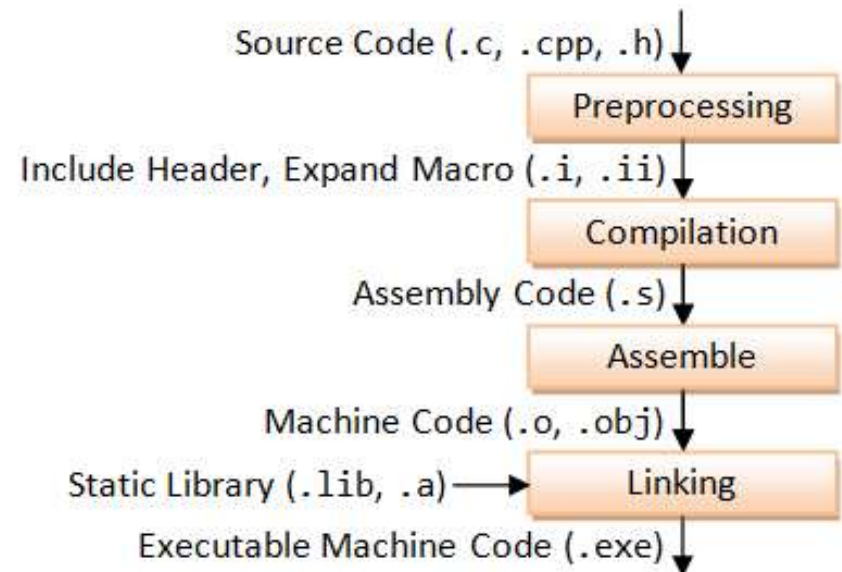


# **Interpretadores y Compiladores**

# Compilador vs Interpretador

## Compilador

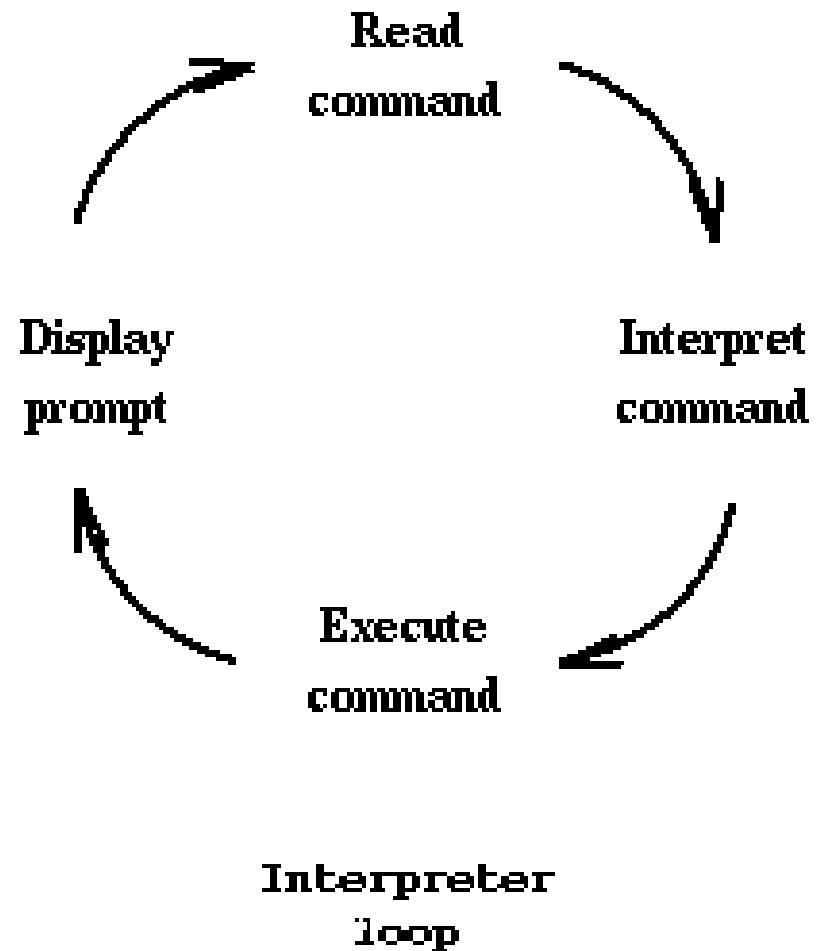
- Compilador toma como entrada todo el código fuente
- Genera un código intermedio de objeto independiente del compilador
- Es más rápido de ejecutar
- Los programas no necesitan ser compilados cada vez que se ejecutan
- Los errores son mostrados después que se verifica todo el programa



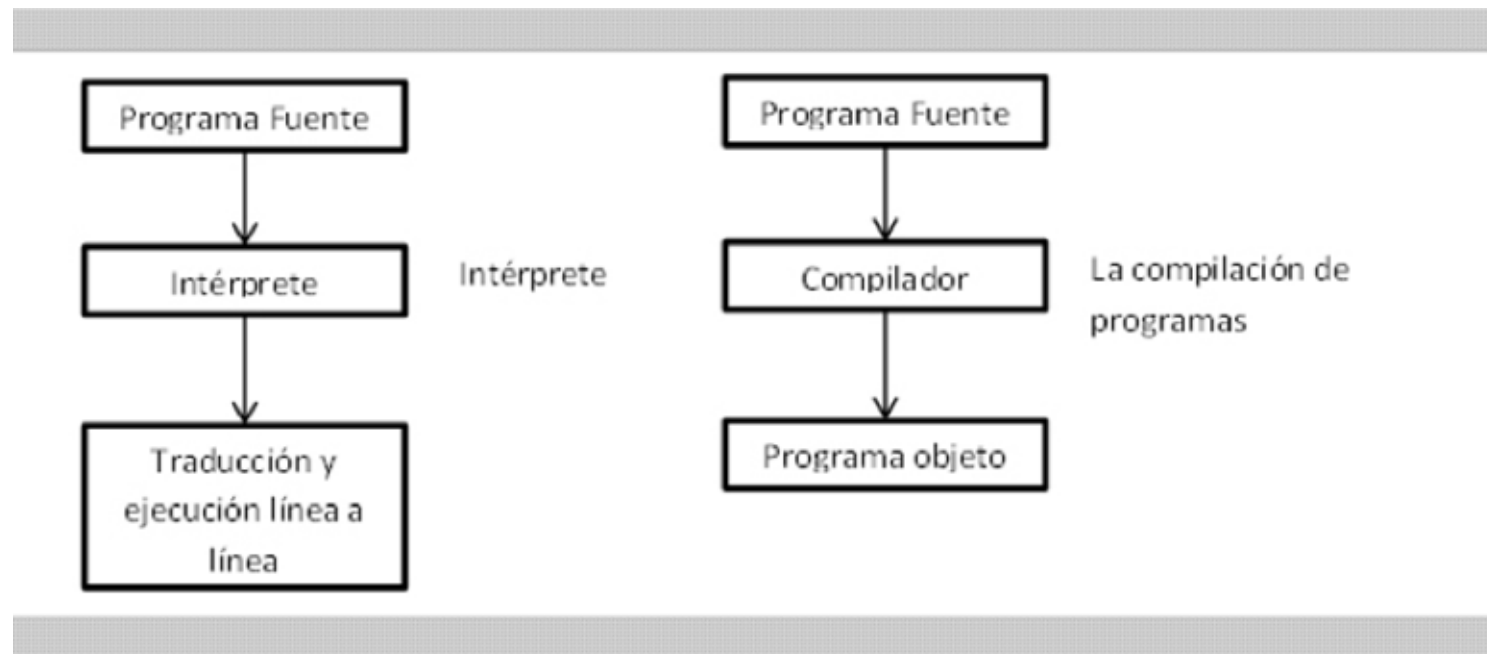
# Compilador vs Interpretador

## Interpretador

- Interpretador toma como entrada una simple instrucción (shell interactivo)
- No se genera código intermedio
- Es mas lento de ejecutar
- Los programas necesitan ser interpretados cada vez que se ejecutan
- Los errores son mostrados por cada instrucción interpretada



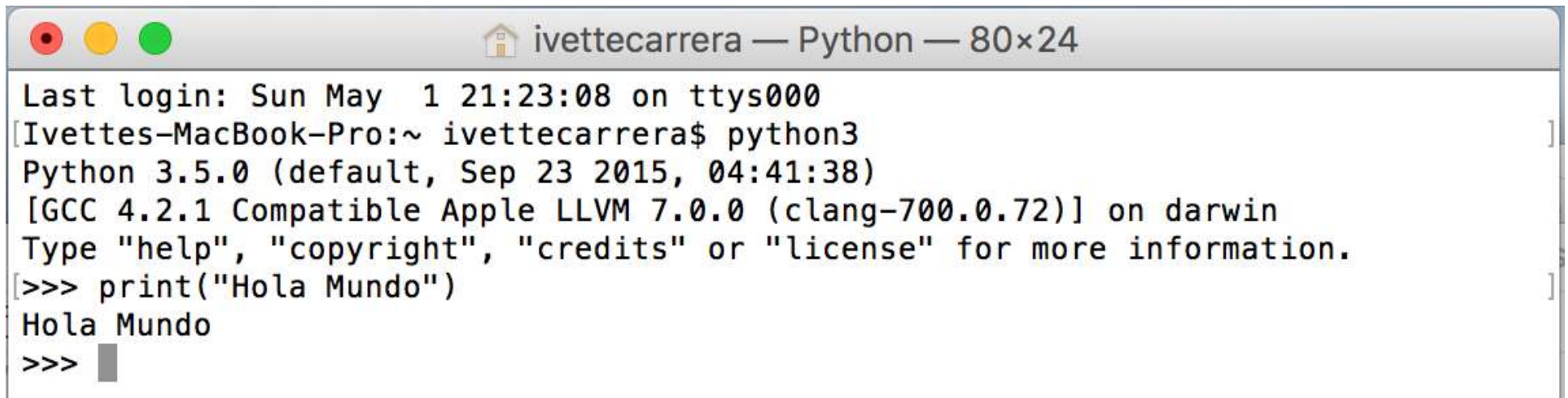
# Compilador vs Interpretador





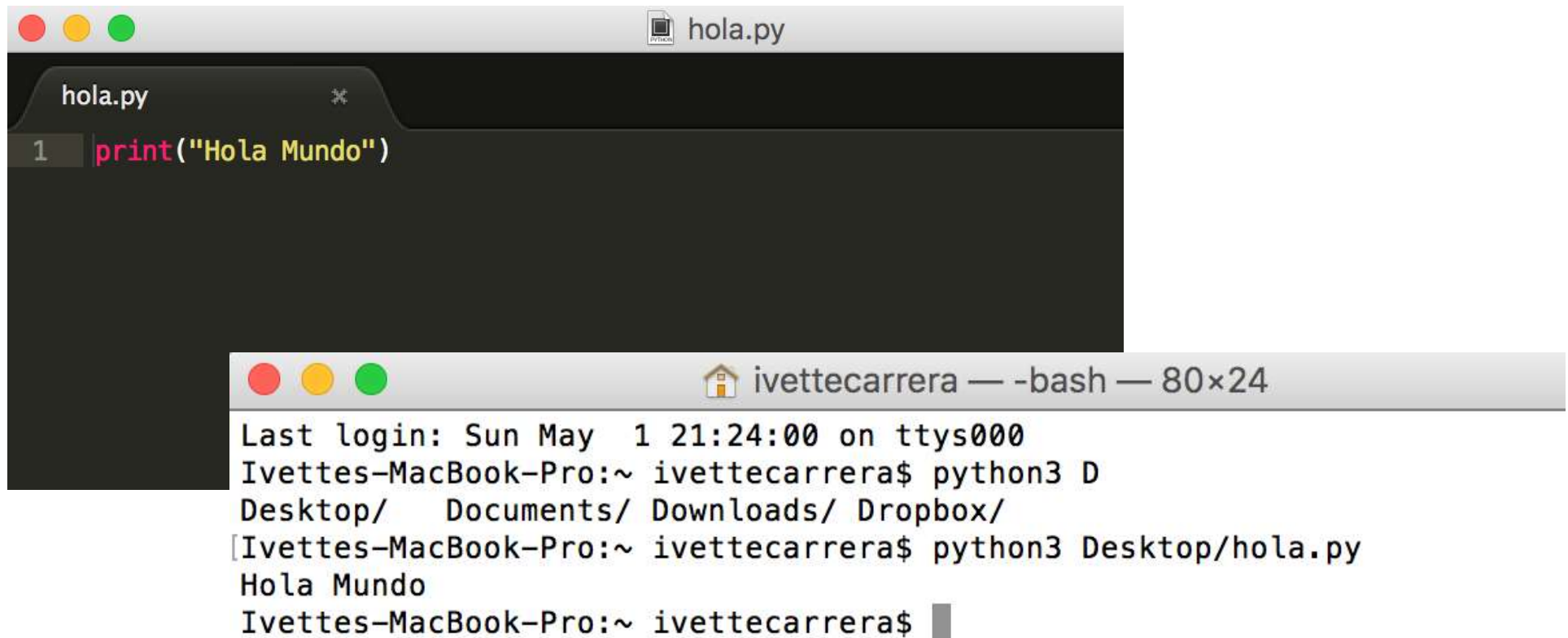
# **Ambientes de Programación**

# Console

A screenshot of a macOS terminal window. The title bar at the top shows three colored window control buttons (red, yellow, green) on the left, a home icon followed by the text 'ivettecarrera — Python — 80x24' in the center, and a vertical scrollbar on the right. The terminal content shows a login message, the execution of 'python3', version and compiler information, and a successful Python command to print 'Hola Mundo'.

```
Last login: Sun May  1 21:23:08 on ttys000
[Ivettes-MacBook-Pro:~ ivettecarrera$ python3
Python 3.5.0 (default, Sep 23 2015, 04:41:38)
[GCC 4.2.1 Compatible Apple LLVM 7.0.0 (clang-700.0.72)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
[>>> print("Hola Mundo")
Hola Mundo
>>> █
```

# Text-Editor + Console



The image shows a screenshot of a Mac desktop with two windows. The top window is a text editor titled 'hola.py' with a dark background. It contains a single line of Python code: `print("Hola Mundo")`. The bottom window is a terminal titled 'ivettecarrera — -bash — 80x24'. It shows the output of running the Python script, displaying 'Hola Mundo'.

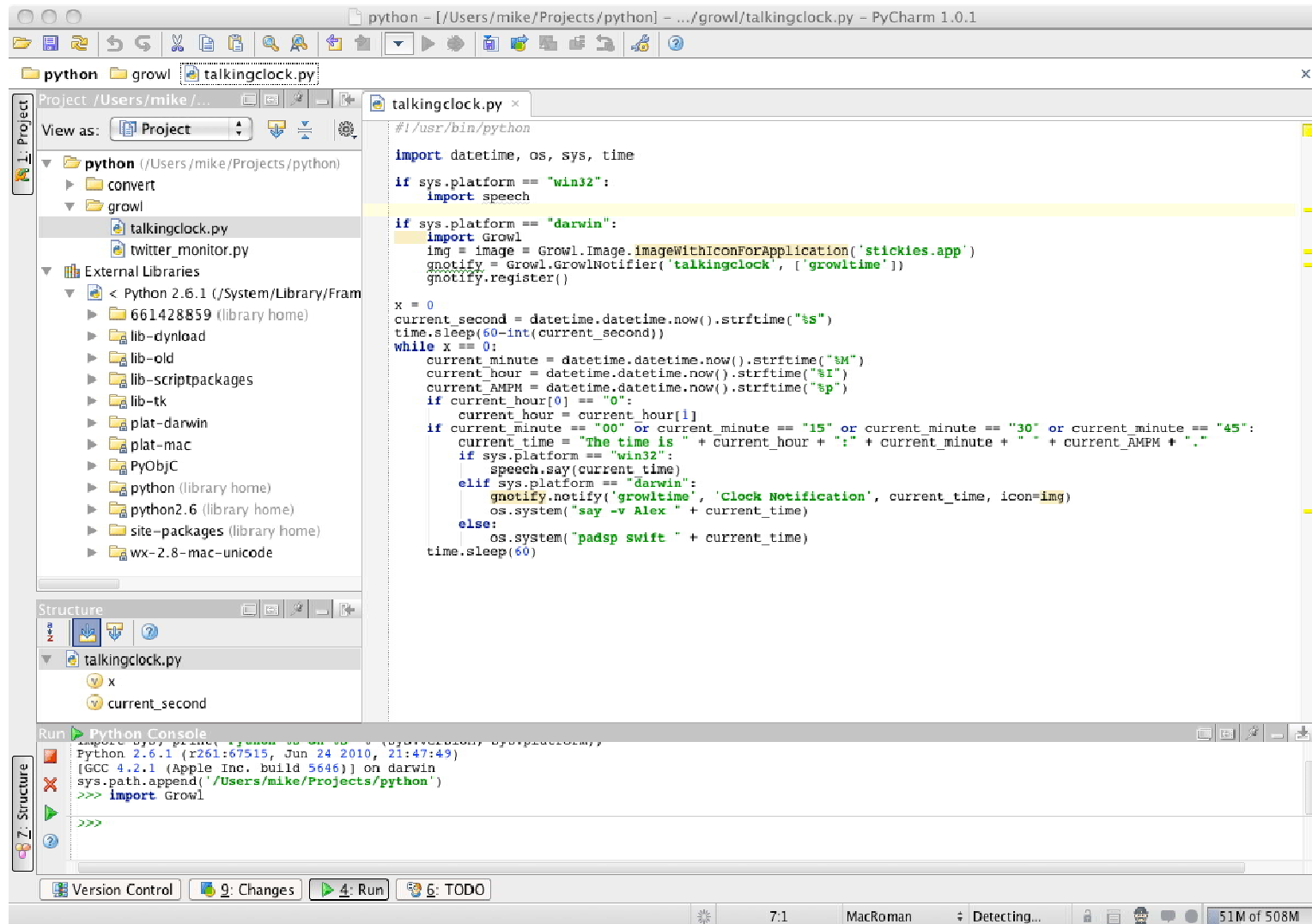
```
hola.py
1 print("Hola Mundo")

ivettecarrera — -bash — 80x24
Last login: Sun May  1 21:24:00 on ttys000
Ivettes-MacBook-Pro:~ ivettecarrera$ python3 D
Desktop/  Documents/ Downloads/ Dropbox/
[Ivettes-MacBook-Pro:~ ivettecarrera$ python3 Desktop/hola.py
Hola Mundo
Ivettes-MacBook-Pro:~ ivettecarrera$
```

# IDE

- Integrated Development Environment
- Editor de código fuente con herramientas de construcción automáticas y un depurador.
- Poseen características como: autocompletar, arrastrar y soltar, break points, etc.
- Existen IDEs que son dirigidos a un lenguaje de programación específicos otros no.

# IDE

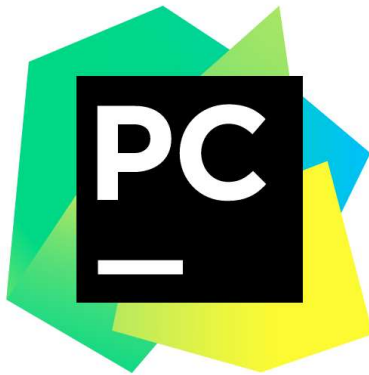


# PyCharm Community



PyCharm

[What's New](#) [Features](#) [Docs & Demos](#) [Buy](#) [Download](#)



Version: 2016.1.2

Build: 145.844

Released: April 7, 2016

[System requirements](#)

[Installation](#)

## Download PyCharm

OS X

WINDOWS

LINUX

### Professional

Full-featured IDE  
for Python & Web  
development

DOWNLOAD

246 MB

### Community

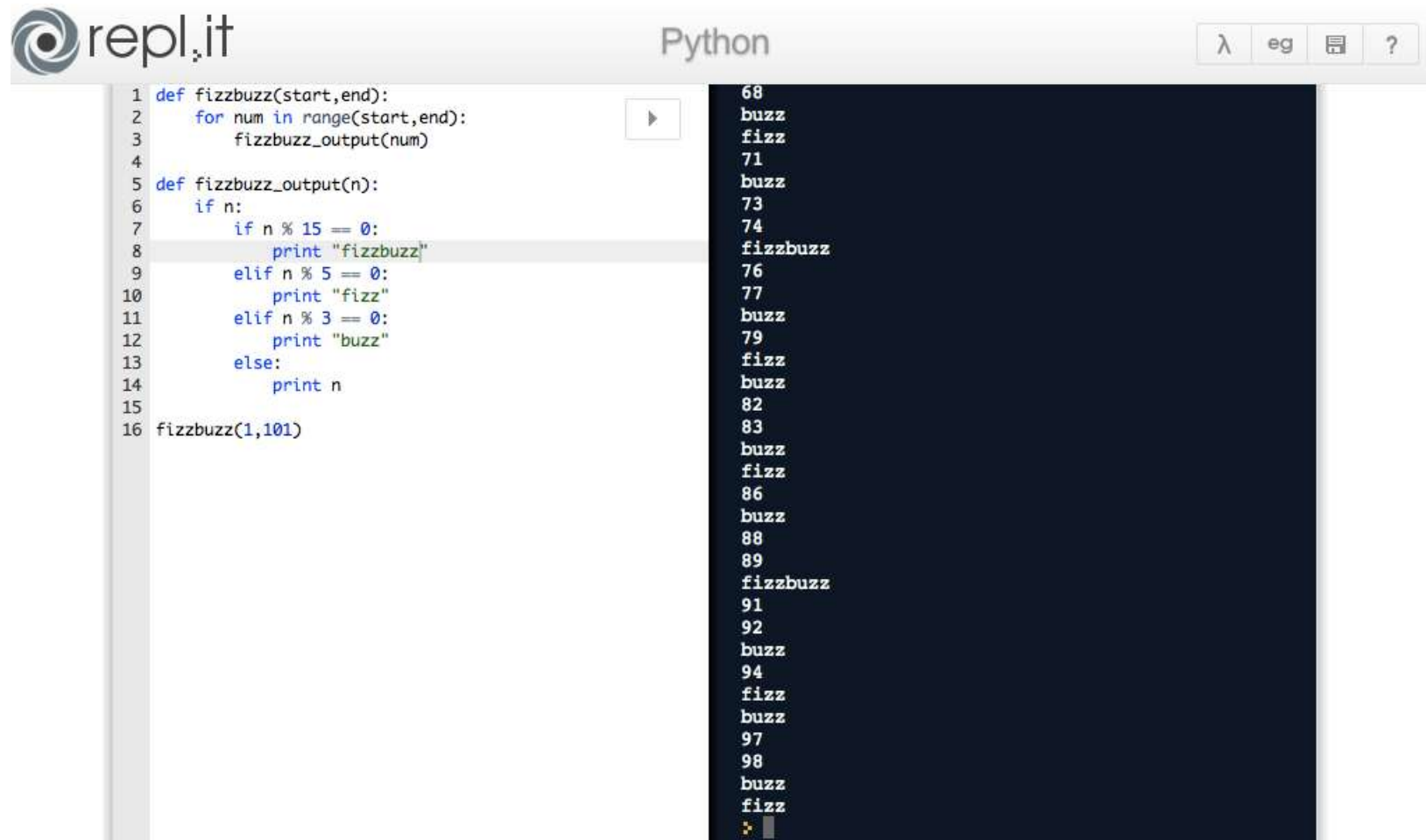
Lightweight IDE  
for Python & Scientific  
development

DOWNLOAD

173 MB

<https://www.jetbrains.com/pycharm/download/>

# Online shells



The screenshot shows the repl.it online shell interface. The top bar includes the repl.it logo, the language 'Python', and utility icons for lambda, eg, a file icon, and a help icon. The left pane contains the following Python code:

```
1 def fizzbuzz(start,end):
2     for num in range(start,end):
3         fizzbuzz_output(num)
4
5 def fizzbuzz_output(n):
6     if n:
7         if n % 15 == 0:
8             print "fizzbuzz"
9         elif n % 5 == 0:
10            print "fizz"
11        elif n % 3 == 0:
12            print "buzz"
13        else:
14            print n
15
16 fizzbuzz(1,101)
```

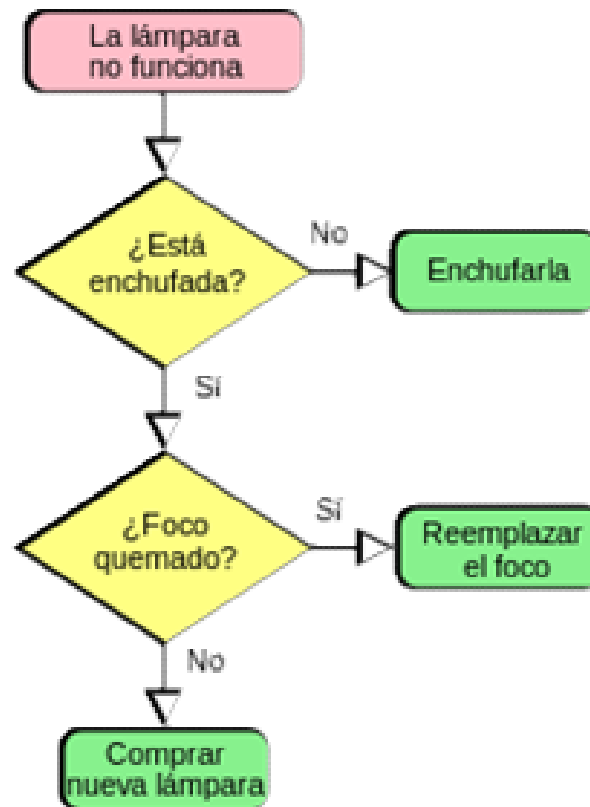
The right pane displays the output of the program, showing numbers from 1 to 100 with 'fizz' or 'buzz' substitutions for multiples of 3 or 5, and 'fizzbuzz' for multiples of 15. The output ends with a cursor on line 101.

```
68
buzz
fizz
71
buzz
73
74
fizzbuzz
76
77
buzz
79
fizz
buzz
82
83
buzz
fizz
86
buzz
88
89
fizzbuzz
91
92
buzz
94
fizz
buzz
97
98
buzz
fizz
```

# **El rol de los algoritmos en el proceso de resolución de problemas**

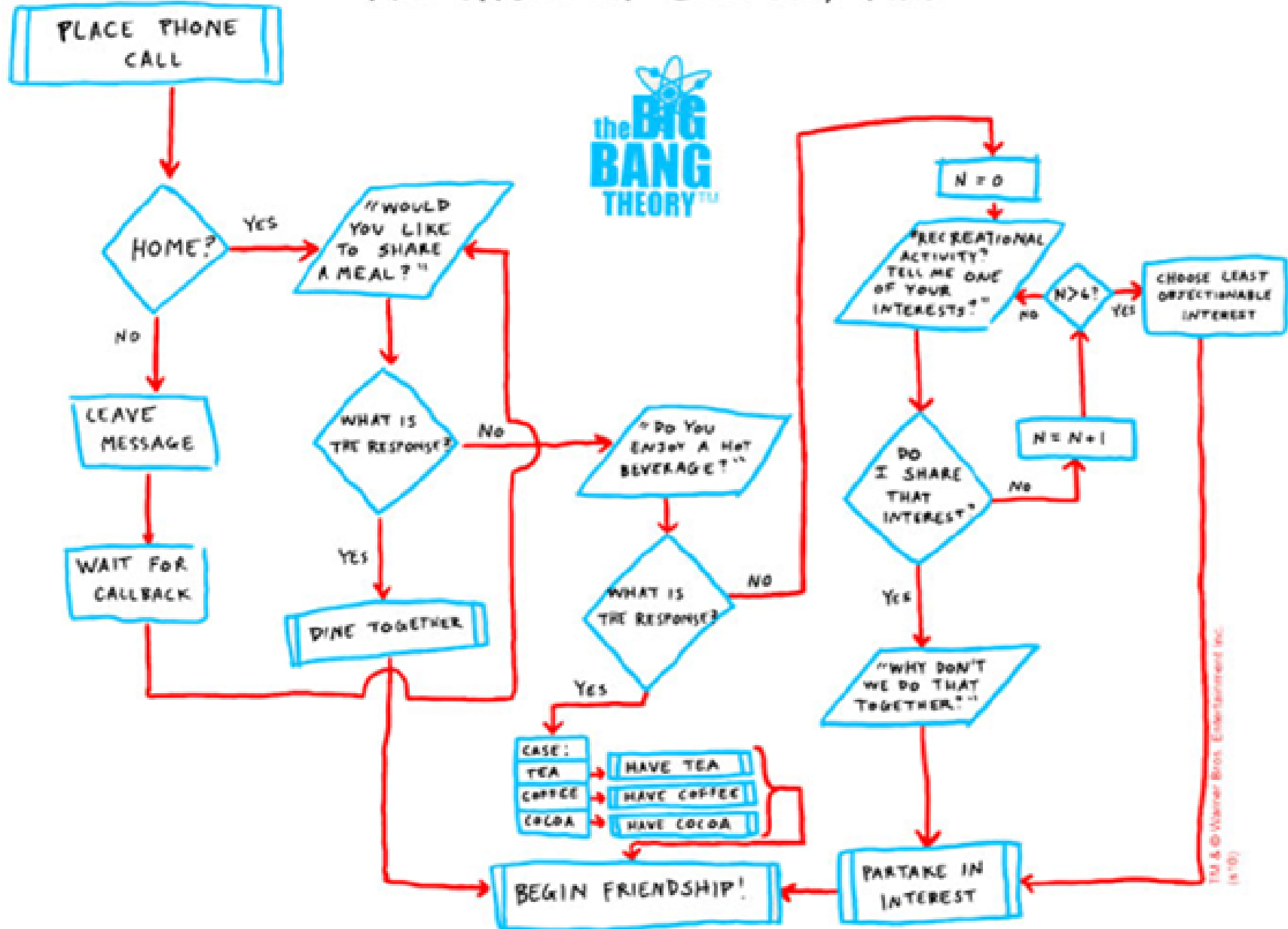


# Algoritmos



# THE FRIENDSHIP ALGORITHM

DR. SHELDON COOPER, Ph.D



# Problemas en el diario vivir

- Cuál es la mejor ruta para ir de un punto A a un punto B?
- Cálculo del IVA en alguna transacción.
- La compra de entradas para un evento.
- Ordenar comida a domicilio.

# Metodología para resolver un problema

## Análisis

- Definir y entender el problema.
- Conocer las variables de entrada, los procesos y las salidas.
- Cuál es el objetivo esperado?

## Diseño

- Cómo se va a resolver el problema?
- Algoritmo que lo resuelve
- Técnicas de representación de la solución

## Implementación

- Implementar la solución en un lenguaje formal que el computador entienda

## Revisión

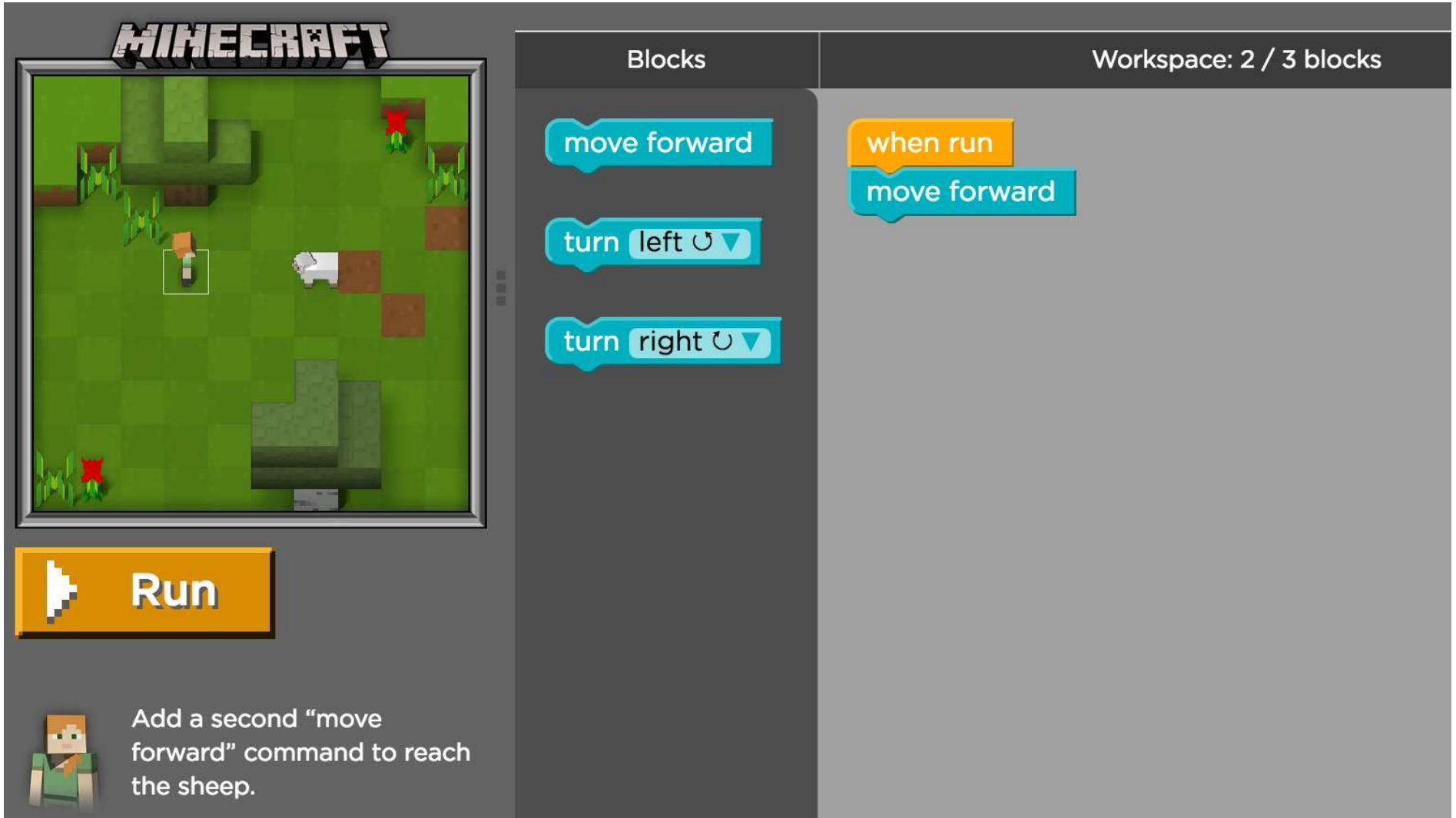
- Pruebas en cada etapa del proceso
- La solución es correcta?
- Se puede optimizar la solución?

# En computación ...

- El proceso de resolución de un problema culmina en la generación de un **Algoritmo**

# Conceptos y Propiedades de los algoritmos

# Qué es un Algoritmo?



The screenshot displays the Minecraft Education Edition interface. On the left is a 3D game world with a player character, a sheep, and some trees. Below the game view is a large orange 'Run' button. At the bottom left, a small character icon is next to the text: 'Add a second “move forward” command to reach the sheep.' The right side of the interface is a coding workspace. It has a header with 'Blocks' and 'Workspace: 2 / 3 blocks'. The workspace contains a single script: an orange 'when run' event block connected to a blue 'move forward' action block. The 'Blocks' panel on the left lists three available blocks: 'move forward', 'turn left' (with a left-turn icon), and 'turn right' (with a right-turn icon).

**MINECRAFT**

Workspace: 2 / 3 blocks

move forward

turn left ↶

turn right ↷

**Run**

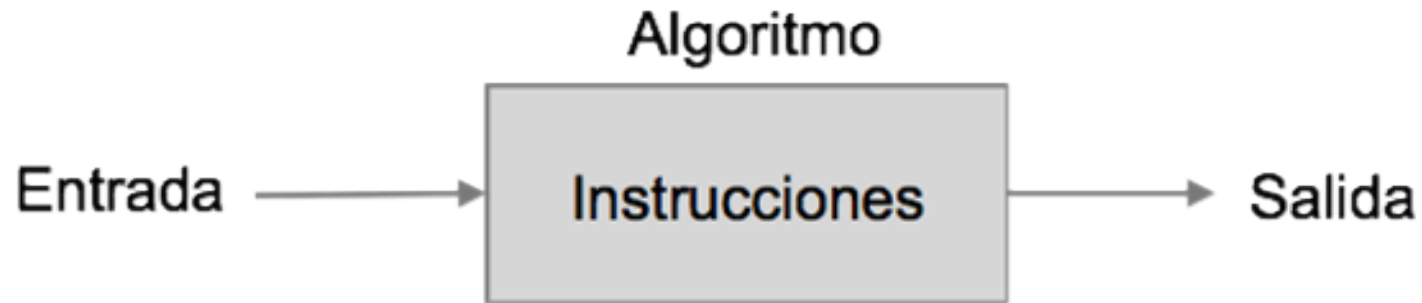
Add a second “move forward” command to reach the sheep.

# Definición de un algoritmo

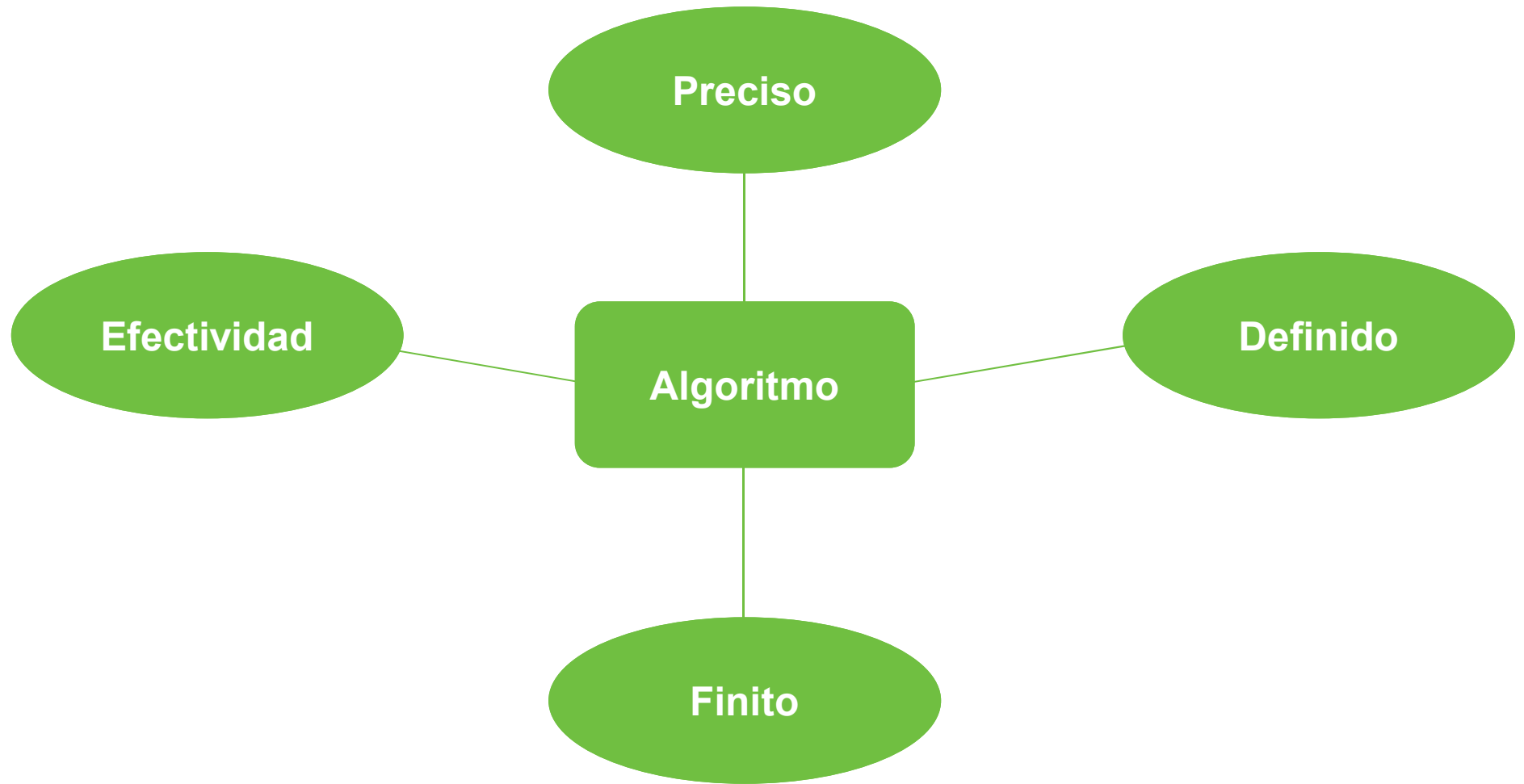
Un algoritmo es una descripción ordenada de las instrucciones que deben realizarse para resolver un problema en un tiempo finito.



# Estructura de un Algoritmo



# Propiedades de los algoritmos



# Representar un algoritmo

- Lenguaje Natural
- Diagrama de flujos
- Pseudocódigo

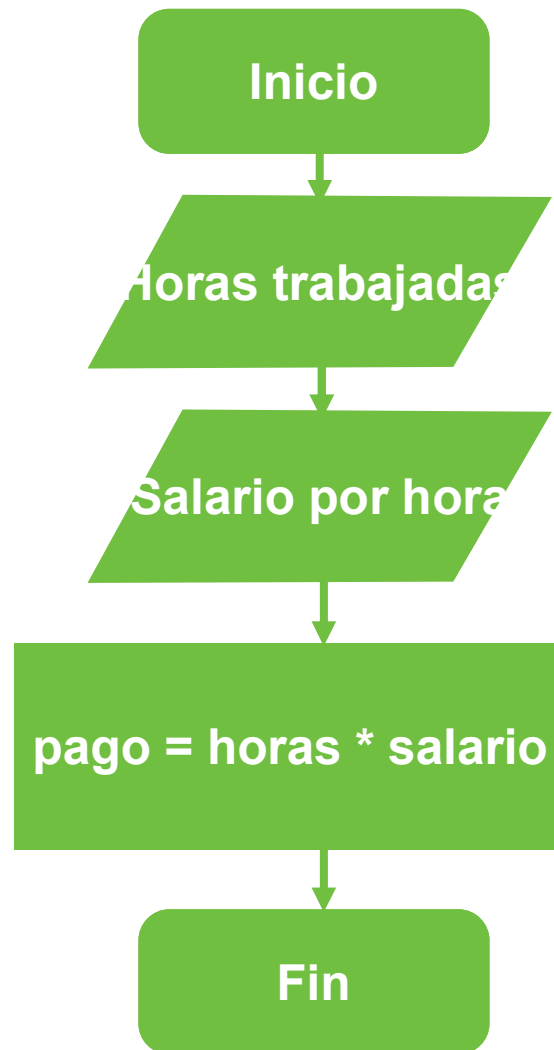
Ok, entonces, un  
problema...

Calcular el pago de un  
trabajador por horas?

# Lenguaje Natural

- Inicio
- Pedir el numero de horas trabajadas
- Pedir el salario por hora
- Multiplicar el numero de horas trabajadas por el salario
- Presentar el pago total
- Fin

# Diagrama de Flujo



# Pseudocódigo

BEGIN

INPUT horas

INPUT salariohora

pago = horas\*salariohora

OUTPUT pago

END





# Tarea

- Escribir un Programa en Python que muestre:

Nombre:

Matrícula:

Equipo de Fútbol Favorito:

- Escribir un algoritmo en lenguaje natural que calcule la edad de una persona