

# Introducción a la programación

Python un lenguaje de programación  
Edición, compilación y ejecución de programa  
Entornos de programación con Python

# Objetivos

- ▶ Distinguir entre las actividades de edición y compilación para ejecutar un programa en un lenguaje de programación
- ▶ Depurar un programa
- ▶ Diferenciar entre el lenguaje de programación Python, el interpretador y las herramientas de desarrollo (IDE)

# Conceptos básicos de lenguajes de programación

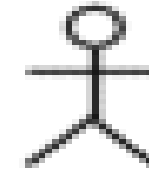
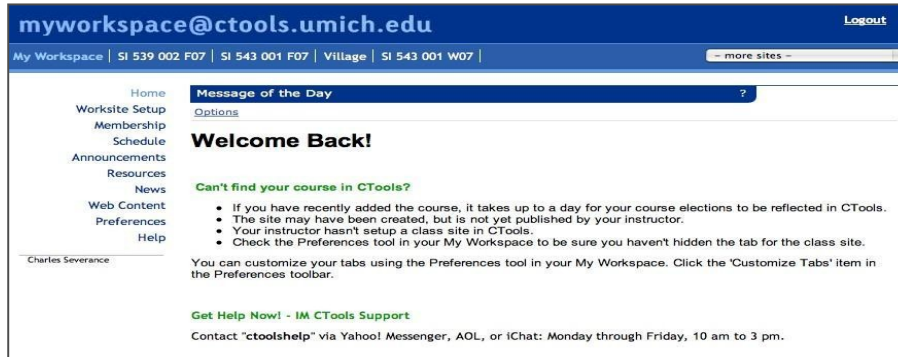
# ¿Qué es una computadora?

- ▶ 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 computadora construyen esos programas

# Usuarios vs. Programadores



**Usuario**



**Programador**



....



# ¿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**



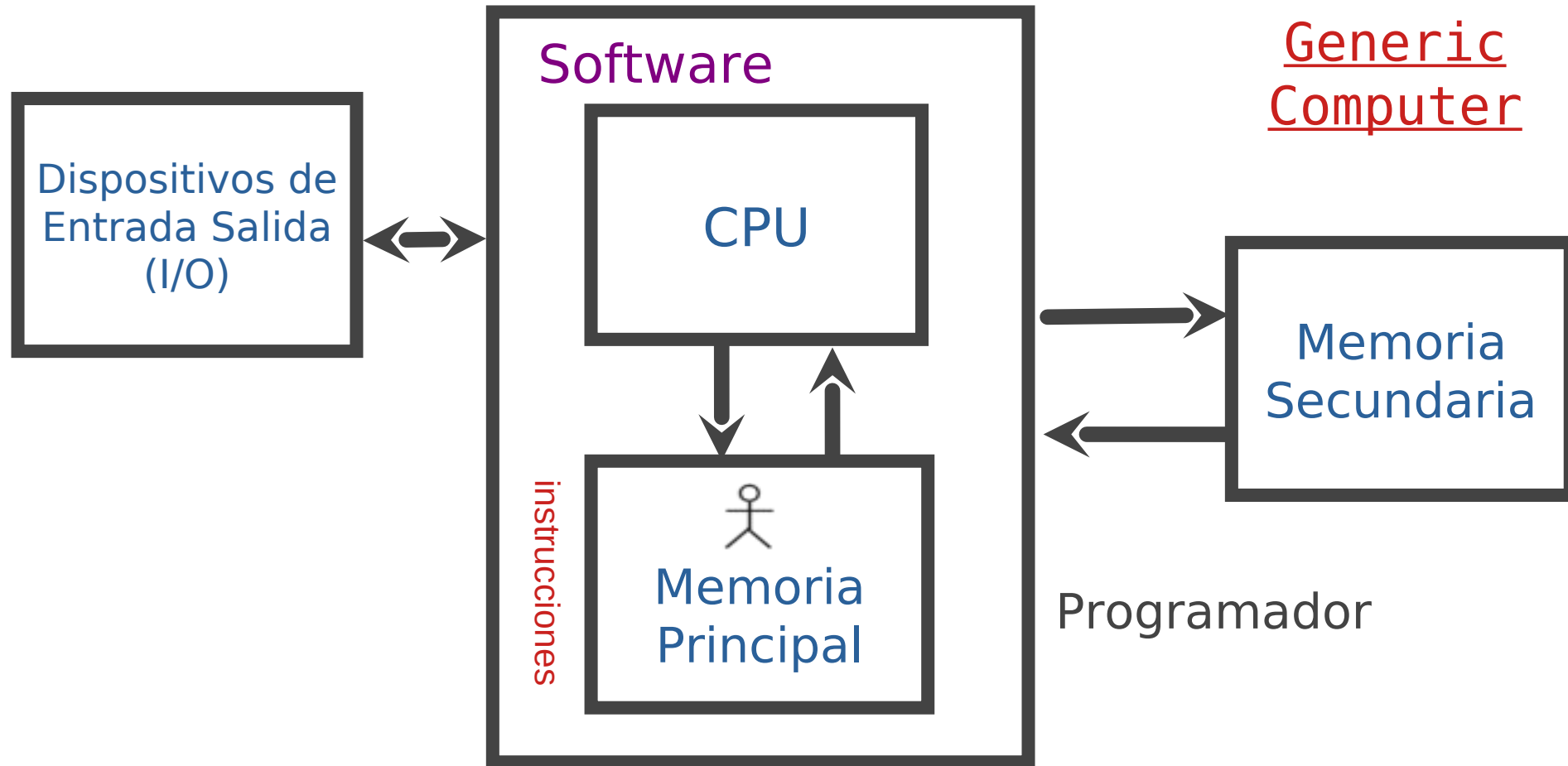
# ¿Qué es un programa?

- ▶ Es una **secuencia ordenada de instrucciones**
- ▶ Es un conjunto de líneas de código de nuestra inteligencia dentro de una computadora
- ▶ Es una obra de arte creativa en especial si es de utilidad para quien la utiliza
- ▶ Otras formas de referirse a un programa: Código Fuente, Scripts, Software.

¿Cómo es que un programa le dice a una computadora qué hacer?



# Arquitectura de Hardware



# Interpretadores y compiladores

# Lenguajes de Programación

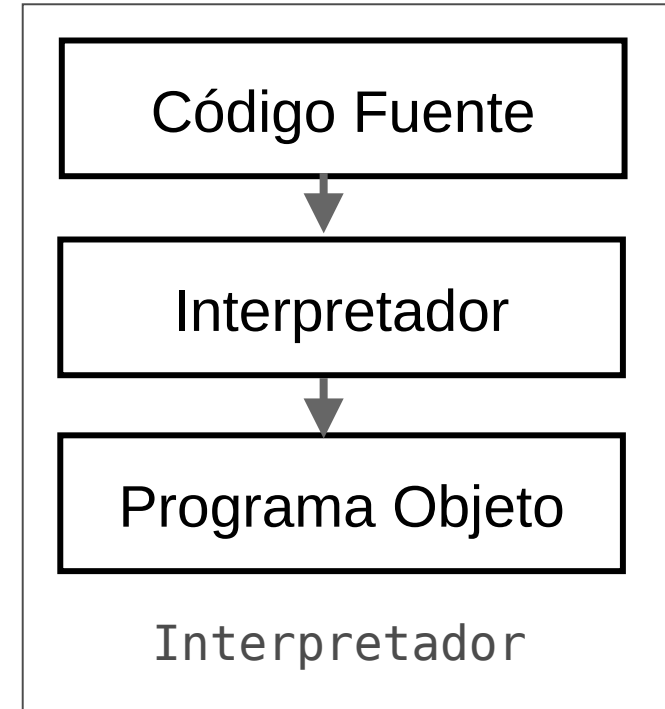
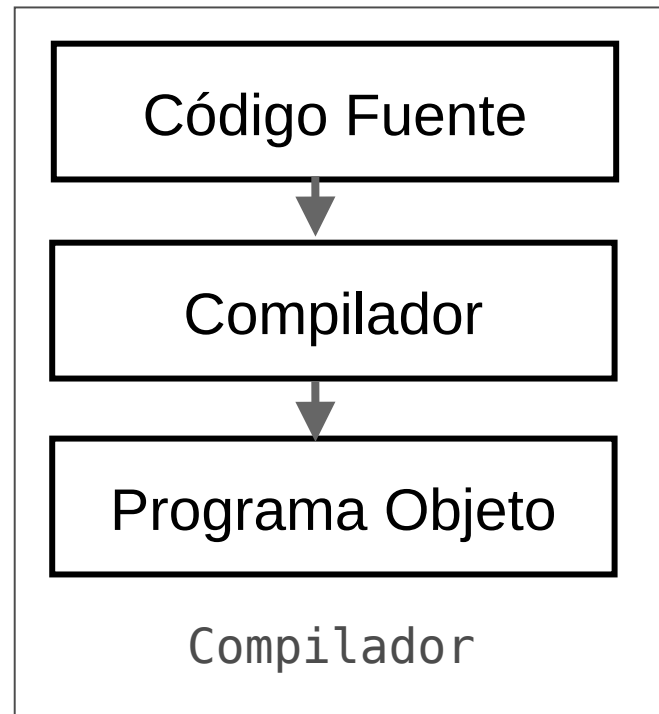
Colocar las instrucciones para que el CPU las entienda significará **poner esas instrucciones en algún lenguaje de computadora.**

Existen muchos lenguajes y clasificaciones, entre ellos:

- ▶ **Lenguaje de alto nivel:** con representación simbólica parecido al inglés y notación matemática. Python, C, Java, Scilab, etc
- ▶ **Lenguaje de bajo nivel:** ejerce un control directo sobre el hardware y está condicionado a la arquitectura de la computadora. Ensamblador.
- ▶ **Lenguaje máquina:** que entienden los circuitos microprogramables tales como un microprocesador

# Traducción a lenguaje máquina

En algún momento para **ejecutar un programa escrito** en algún lenguaje de alto o bajo nivel debe realizarse **el paso de traducción a lenguaje máquina**.



# 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

## Interpretador

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

# Ejemplos

```
HolaMundoC.c x
#include <stdio.h>
int main()
{
    printf("Hola Mundo\n");
    printf("Hello World\n");
    printf("Ciao a tutti\n");
    return 0;
}
```

C

```
HolaMundoPy.py x
print "Hello World in Python"
print "Hola mundo"
print "Ciao a tutti"
```

Python

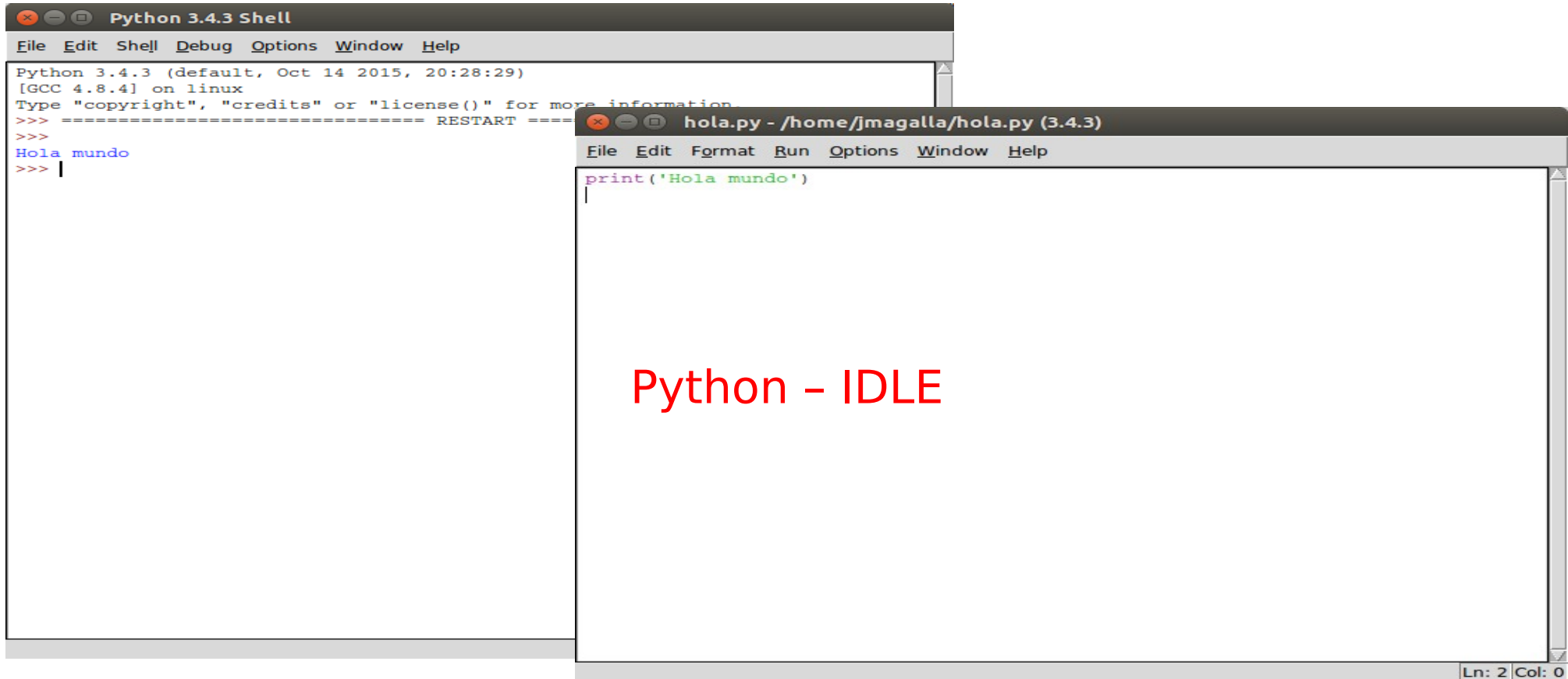


# Entornos de programación

Editores de texto  
Entorno integrado (IDE)

```
Python 2.7.6 (default, Mar 22 2014, 22:59:38)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

## Python – Consola interactiva

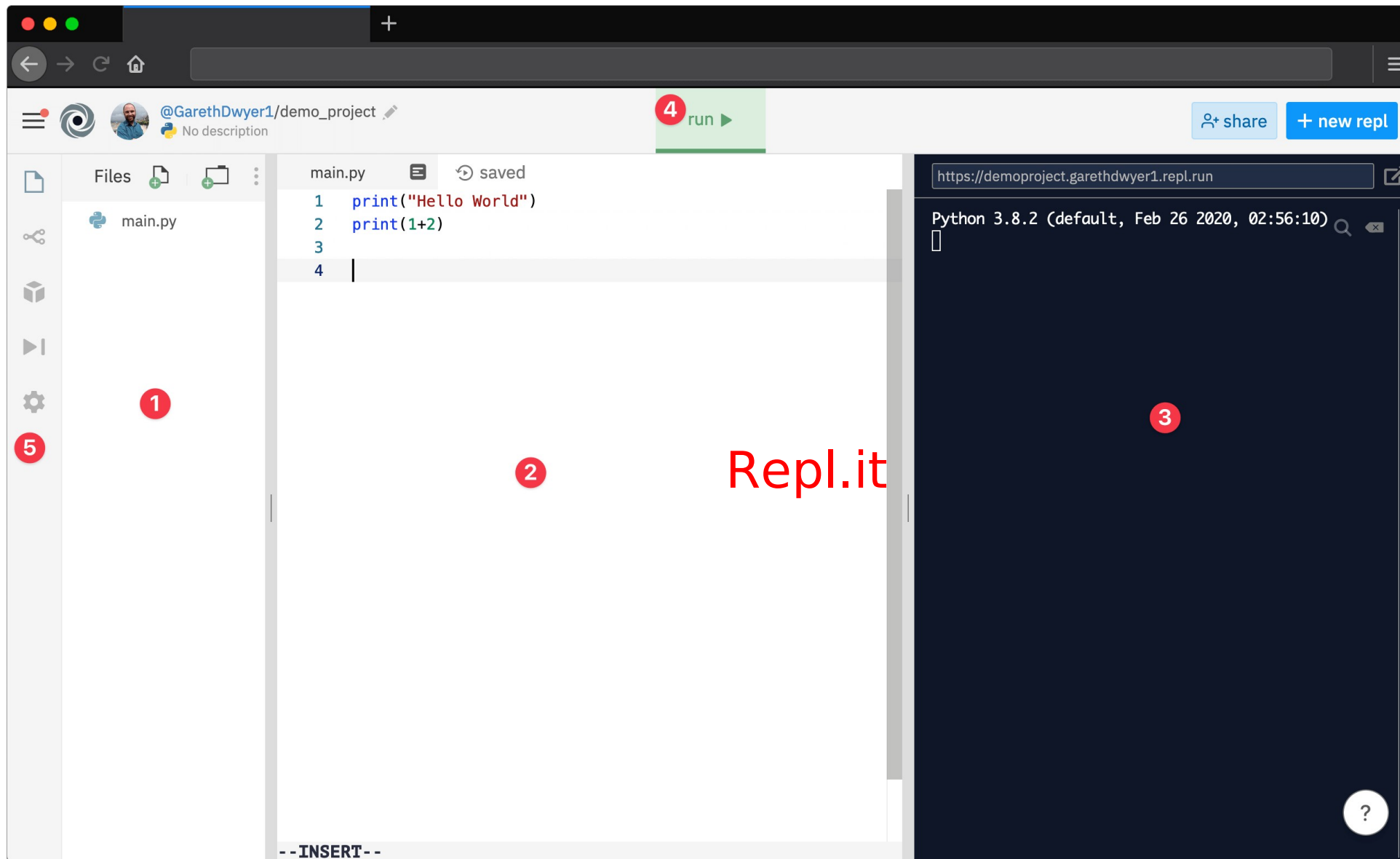


## Python – IDLE



# ¿Qué es un IDE?

- ▶ IDE: Entorno de desarrollo integrado.
- ▶ Normalmente consta de editor de código fuente, herramientas de construcción automáticas y un depurador.
- ▶ La de mayor parte de estos IDE tienen autocompletado inteligente.



**Left pane:** files and configuration.  
**Middle pane:** code editor.  
**Right pane:** output sandbox.  
**Run button**  
**Menu bar:** this lets you control what you see in the main left pane (pane 1).

SPYDER

<https://www.spyder-ide.org/>

The screenshot displays the Spyder IDE interface with the following components:

- Left Panel (File Explorer):** Shows a project structure with files like `App.py`, `template.py`, `plot_example.py`, and `plugin.py`. The `plugin.py` file is currently selected.
- Editor:** Displays the code for `plot_example.py`. The code includes imports for `numpy`, `matplotlib.pyplot`, `matplotlib.cm`, and `matplotlib.colors`. It defines two functions: `generate_polar_plot()` and `generate_dem_plot()`. The `generate_polar_plot()` function generates a polar plot with 20 slices, and the `generate_dem_plot()` function generates a 3D representation of a terrain DEM.
- Right Panel (Variable Explorer):** Shows a table of variables and their values.
 

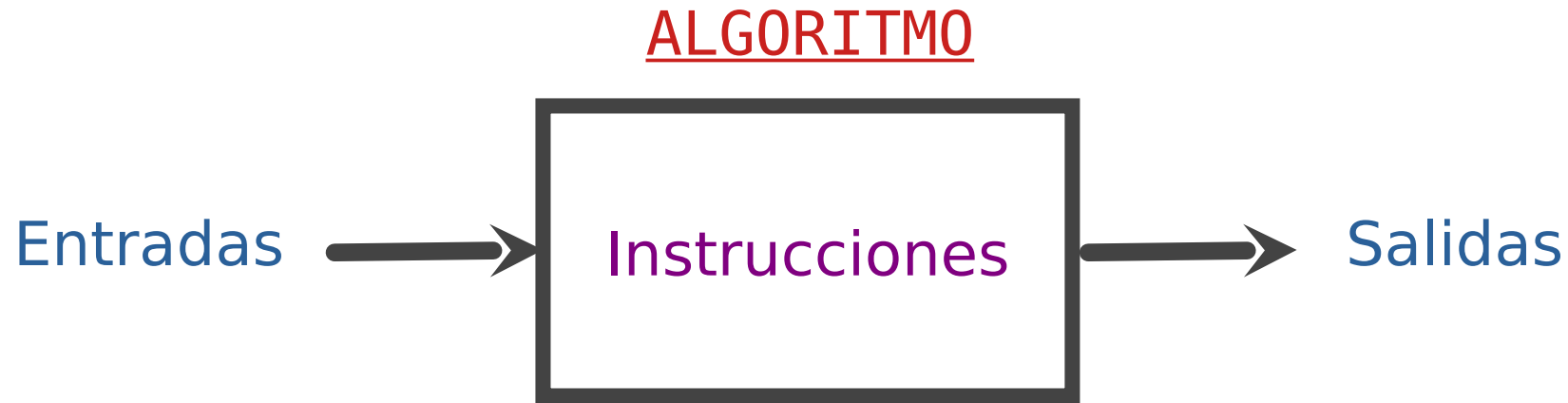
Name	Type	Size	Value
data	Array of str128	(3, 3)	ndarray object of numpy module
df	DataFrame	(2, 2)	Column names: Col1, Col2
filename	str	1	/Users/Documents/spyder/spyder/tests/test_dont_use.py
i	Array of uint32	(10, 10)	[[0 0 0 ... 0 0 0] [0 0 0 ... 0 0 0]
li	list	5	['abcd', 745, 2.23, 'efgh', 70.2]
r	float	1	6.46567886443
t	tuple	5	('abcd', 745, 2.23, 'efgh', 70.2)
tinylst	list	2	[123, 'efgh']
- Bottom Panel:** Contains two plots. The left plot is a 3D surface plot showing a terrain DEM. The right plot is a polar plot showing data distribution across different angles.
- Status Bar:** Shows the current state of the IDE, including the Python interpreter (conda: base (Python 3.7.4)), the current line and column (Line 1, Col 1), the encoding (ASCII), the line feed (LF), the read/write status (RW), and the memory usage (Mem 50%).

# Conceptos y propiedades de los algoritmos

# ¿Qué es un Algoritmo?

- ▶ Un algoritmo es una **descripción ordenada de las instrucciones** que deben realizarse para resolver un problema en un tiempo finito
- ▶ Un **algoritmo de computador** debe orientarse a la implementación computacional, pero para problemas simples puede omitirse e ir directo a la implementación
- ▶ Es importante **desarrollar el pensamiento algorítmico** progresivamente

# De la estructura de un algoritmo



- ▶ Un algoritmo necesita comunicarse con el entorno, por lo tanto tendrá entradas de datos y salida de resultados
- ▶ Un algoritmo será un procedimiento de instrucciones que transformará las entradas y producirá los resultados esperados

# Características de los algoritmos

Los algoritmos se pueden representar en diversas notaciones, lenguaje natural, gráfico, simbólico, etc

Los algoritmos deben ser:

- fáciles de construir y entender: instrucciones simples, claras y precisas
- exactos: con suficientes instrucciones para resolver el problema
- finitos: tener principio y fin
- reproducibles: deben entregar los mismos resultados si se utilizan los mismos datos de entrada

# Construcción de algoritmos

Es importante asegurarnos de que el problema que intentamos resolver está en nuestro ámbito de conocimiento.

Pasos para la construcción de un algoritmo:

- Debe definirse el objetivo al que se desea llegar
- Debe identificarse las variables
- Escribir las instrucciones necesarias para llegar al objetivo propuesto.
- Validar el algoritmo mediante pruebas y corregir errores



# Ejemplo

Escriba un algoritmo para prepararse un sándwich de jamón y queso.

Recuerde: descripción ordenada de instrucciones (pasos numerados) y dentro de nuestras competencias