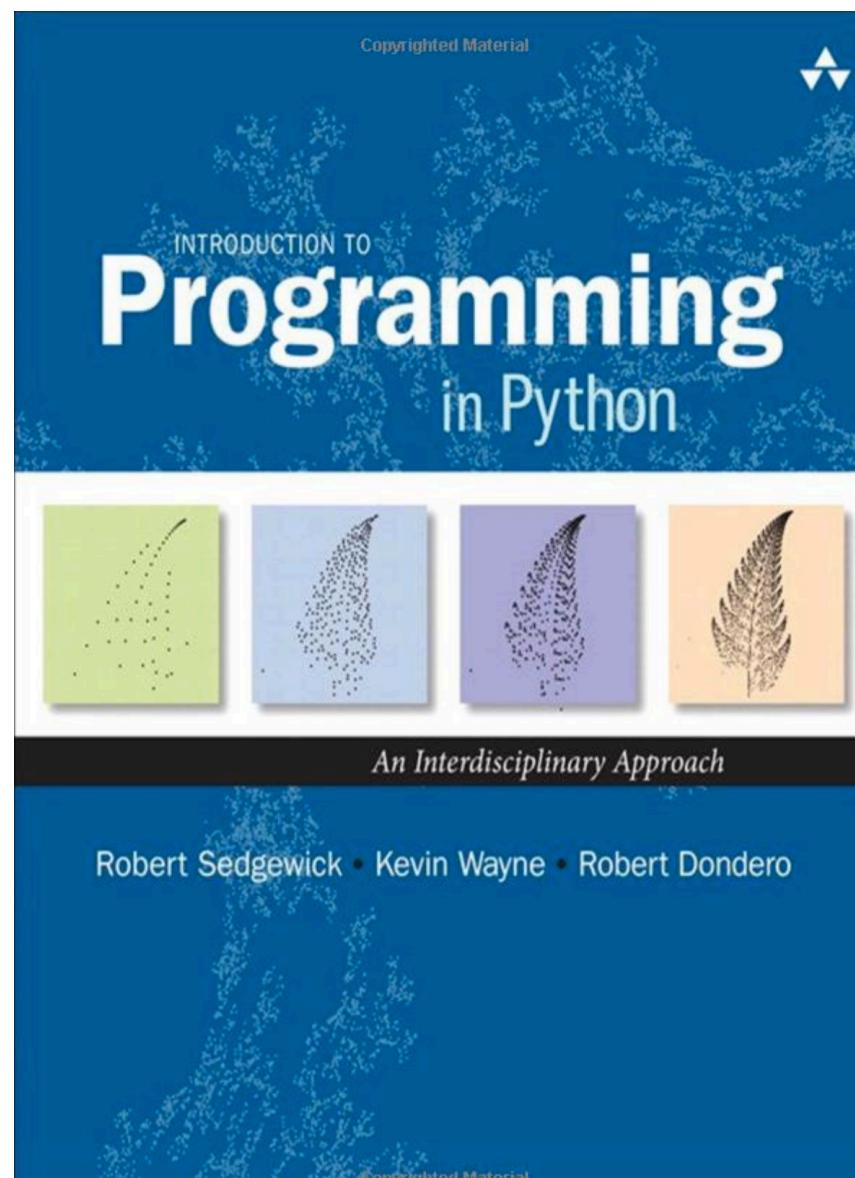


# Taller de Programación

## Clase 01: Introducción a Python

Daniela Opitz  
dopitz@udd.cl

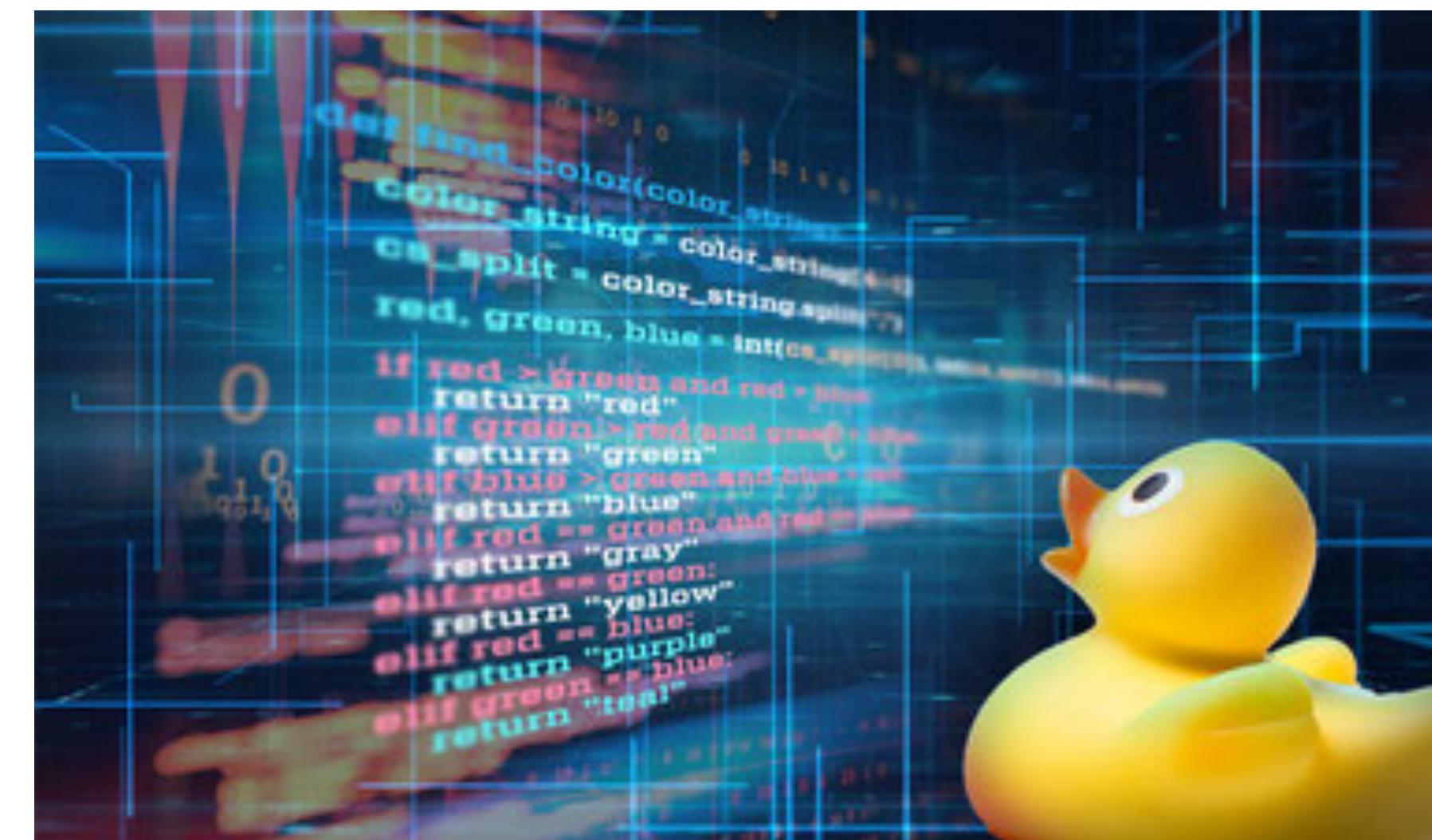


Basada en presentaciones oficiales de libro *Introduction to Programming in Python* (Sedgewick, Wayne, Dondero).

Disponible en <https://introcs.cs.princeton.edu/python>

# Outline Clase 1

- Anaconda (Spyder & Jupyter), Repl
- Shells
- Primer programa ('Hola Mundo')
- Sintaxis
- Variable y tipo de datos
- Operadores de aritmética básicos
- Entrada/salida de datos



# Anaconda



## ANACONDA®



Windows	MacOS	Linux
Python 3.8	Python 3.8	Python 3.8
<a href="#">64-Bit Graphical Installer (466 MB)</a>	<a href="#">64-Bit Graphical Installer (462 MB)</a>	<a href="#">64-Bit (x86) Installer (550 MB)</a>
<a href="#">32-Bit Graphical Installer (397 MB)</a>	<a href="#">64-Bit Command Line Installer (454 MB)</a>	<a href="#">64-Bit (Power8 and Power9) Installer (290 MB)</a>

# Anaconda

 ANACONDA NAVIGATOR Sign in to Anaconda Cloud

Home Applications on base (root) Channels Refresh

**JupyterLab** 1.2.6 An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture. [Launch](#)

**jupyter Notebook** 6.0.3 Web-based, interactive computing notebook environment; Edit and run human-readable docs while describing the data analysis. [Launch](#)

**IPy[ty]** 4.6.0 PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more. [Launch](#)

**Spyder** 4.0.1 Scientific Python Development Environment; Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features. [Launch](#)

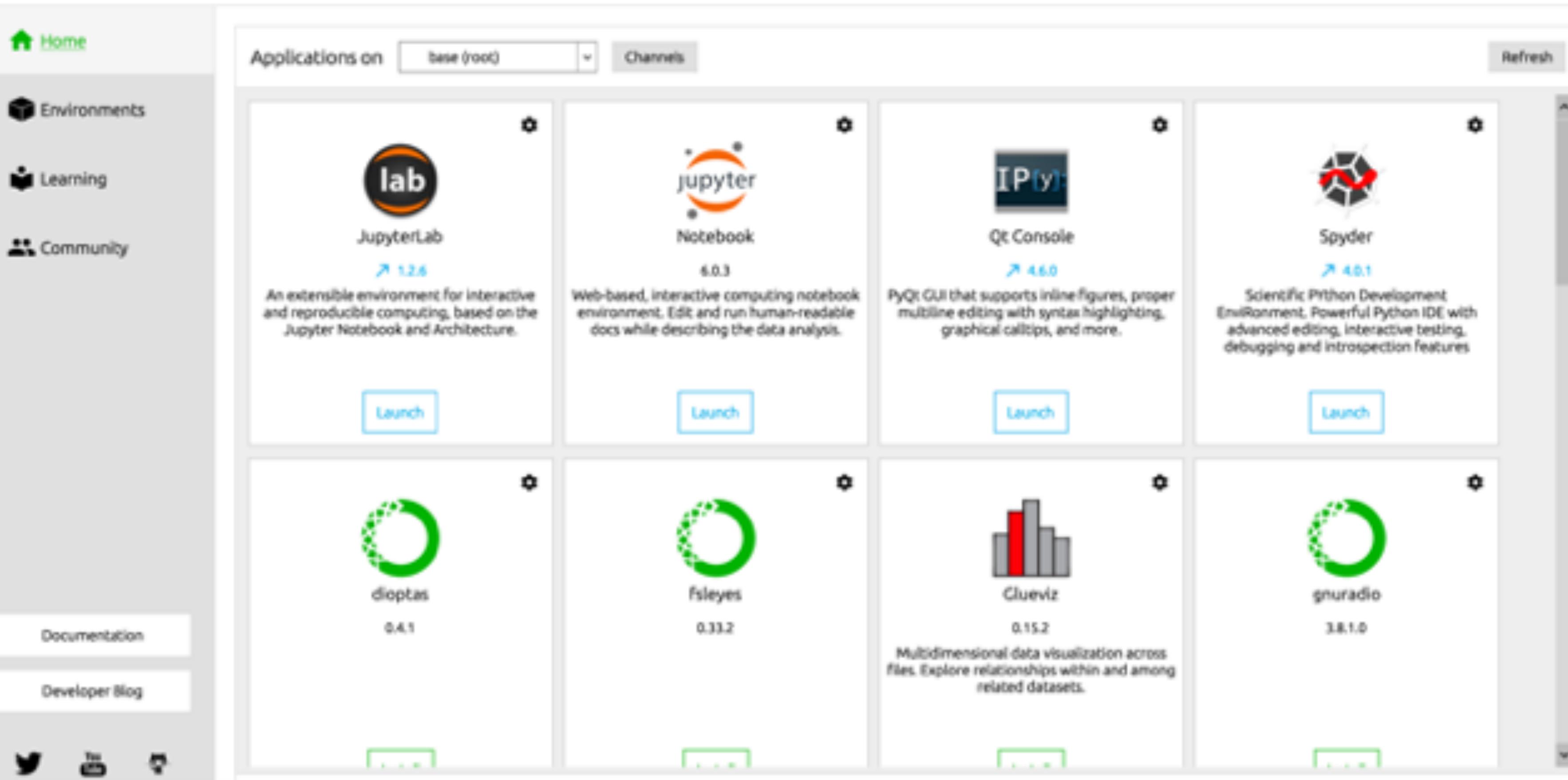
**dioptas** 0.4.1 [Documentation](#)

**fsleyes** 0.33.2 [Documentation](#)

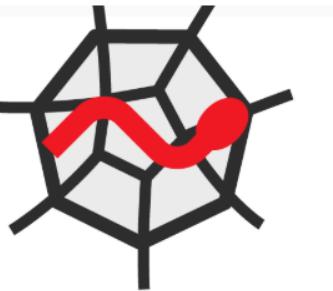
**Glueviz** 0.15.2 Multidimensional data visualization across files. Explore relationships within and among related datasets. [Documentation](#)

**gnuradio** 3.8.1.0 [Documentation](#)

Documentation Developer Blog [Twitter](#) [YouTube](#) [GitHub](#)



# Spyder & Python



## SPYDER

The Scientific Python Development Environment

A screenshot of the Spyder IDE interface. On the left is the file browser showing various Python files and modules. The central area contains a code editor with the following Python script:

```
6 import pylab
7 from numpy import cos, linspace, pi, sin, random
8 from scipy.interpolate import splprep, splev
9
10
11 # %% Generate data for analysis
12 t = linspace(0, 1.75 * 2 * pi, 100)
13
14 x = sin(t)
15 y = cos(t)
16 z = t
17
18 # Add noise
19 x += random.normal(scale=0.1, size=x.shape)
20 y += random.normal(scale=0.1, size=y.shape)
21 z += random.normal(scale=0.1, size=z.shape)
22
23 # %% Perform calculations
24
25
26 # %% Smoothness parameter
27
28 # Spline parameters
29 smoothness = 3.0 # Smoothness parameter
30 k_param = 2 # Spline order
31 nests = -1 # Estimate of number of knots needed (-1 = maximal)
32
33 # Find the knot points
34 knot_points, u = splprep([x, y, z], s=smoothness, k=k_param, nests=-1)
35
36 # Evaluate spline, including interpolated points
37 xnew, ynew, znew = splev(linspace(0, 1, 400), knot_points)
38
39
40 # %% Plot results
41
42 # TODO: Rewrite to avoid code smell
43 pylab.subplot(2, 2, 1)
44 data, = pylab.plot(x, y, 'bo-', label='Data with X-Y Cross Section')
45 fit, = pylab.plot(xnew, ynew, 'r--', label='Fit with X-Y Cross Section')
46 pylab.legend()
47 pylab.xlabel('x')
48 pylab.ylabel('y')
```

To the right of the code editor is the Variable Explorer window displaying data structures like 'bars' (BarContainer object of matplotlib.container), 'df' (Dataframe), 'filename' (str), 'list\_test' (list), 'radii' (float64), 'region' (tuple), 'rgb' (float64), and 'series' (Series object of pandas.core.series). Below the code editor is the IPython console showing a 3D surface plot of the data.

- Scientific Python Development Environment
- Entorno de desarrollo interactivo para el lenguaje Python



- Jupyter Notebook es un entorno informático interactivo para programar.

# repl.it

The screenshot shows the repl.it web-based development environment. At the top, there are icons for user profile, workspace name 'dopitz / Clase01', Python logo, and a refresh arrow. A 'Run ▶' button is located in the top right. On the left is a sidebar titled 'Files' containing three files: 'main.py' (selected), '01\_holamundi.py', and '02\_variables.py'. The main area is a code editor for 'main.py' with the following text:

```
1 Not sure what to do? Run some examples (start typing to dismiss)
```

- Entorno de desarrollo interactivo gratuito para el varios lenguajes incluido Python

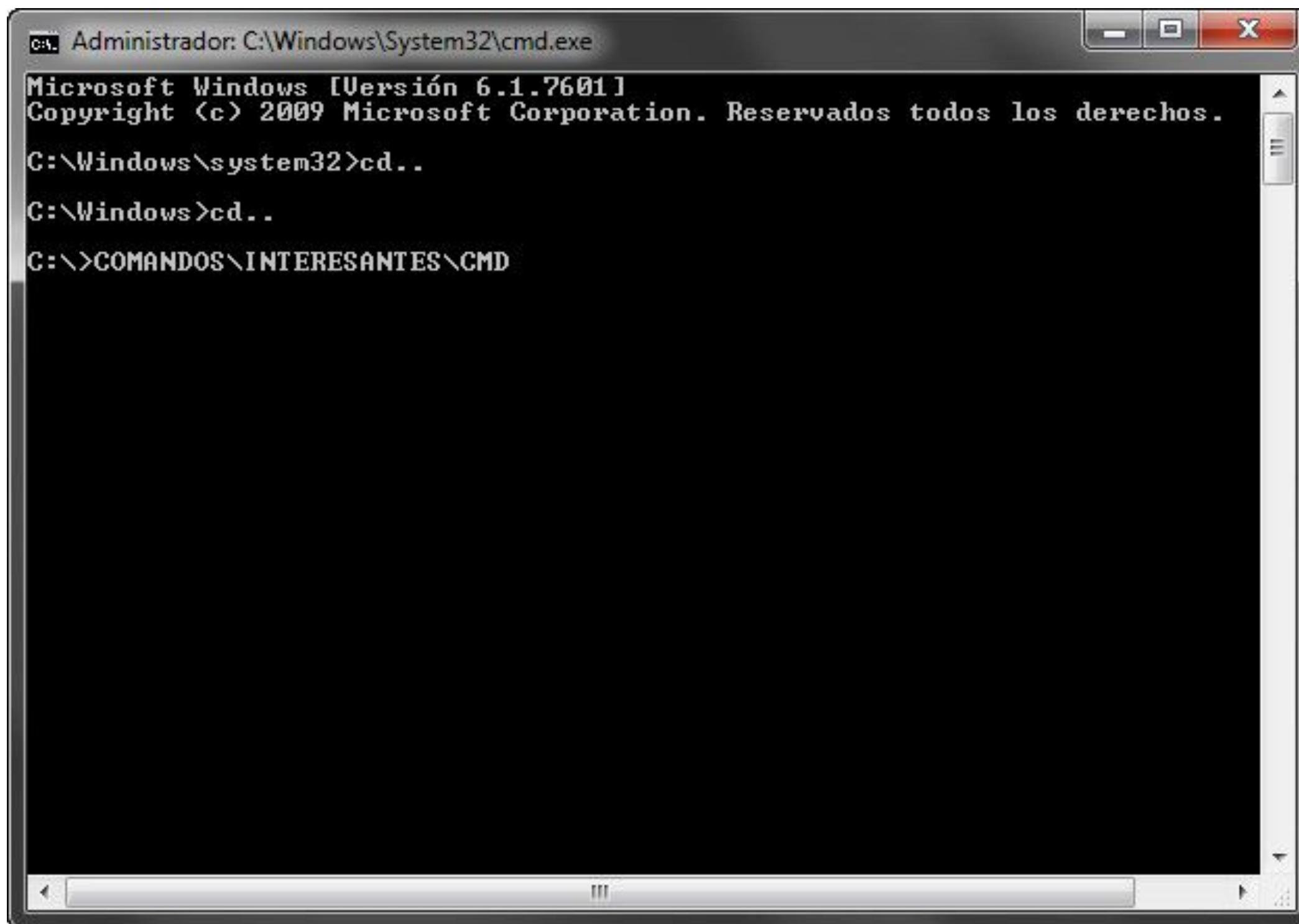
# Shells

Shell: intérprete de órdenes o intérprete de comandos. Provee una interfaz de usuario para acceder a los servicios del sistema operativo.

- De líneas texto (CLI)
- Gráficos (GIU)
- De lenguaje natural (NIU)

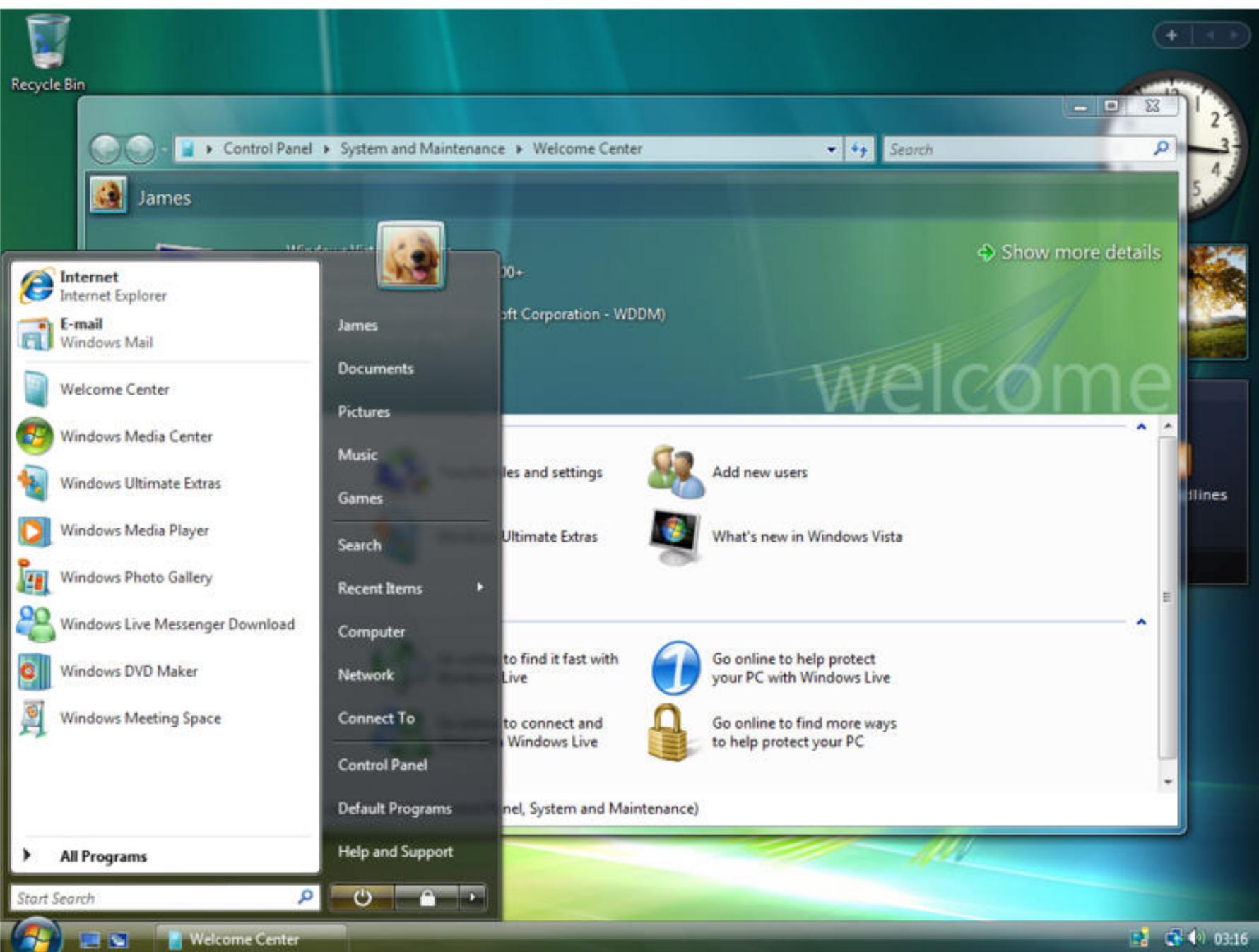
# CLI

Command interface: permite a los usuarios dar instrucciones por medio de una línea de texto simple.



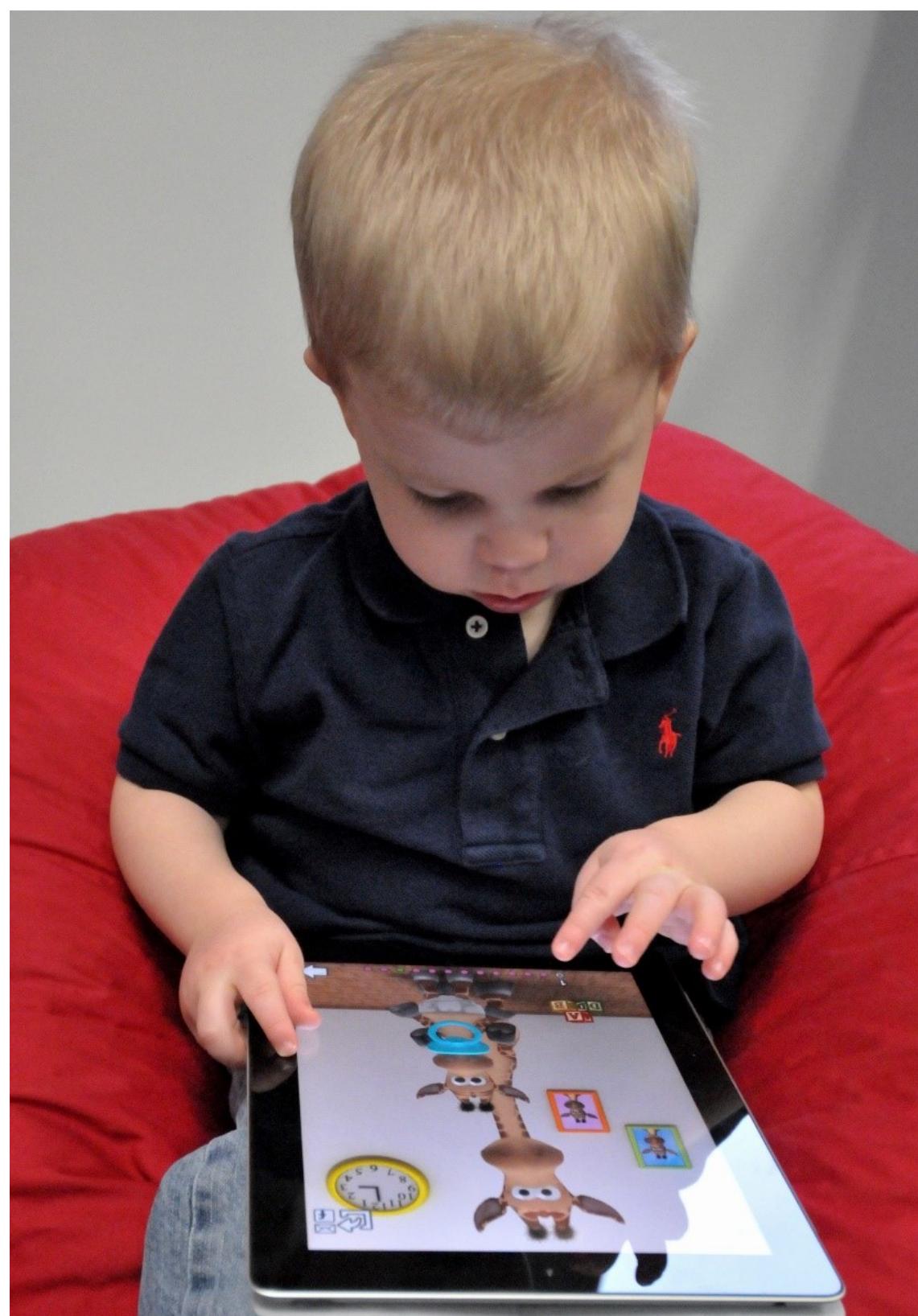
# GUI

Graphical user interface: utiliza imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz. Proporcionar un entorno visual sencillo para permitir la comunicación con el sistema operativo.



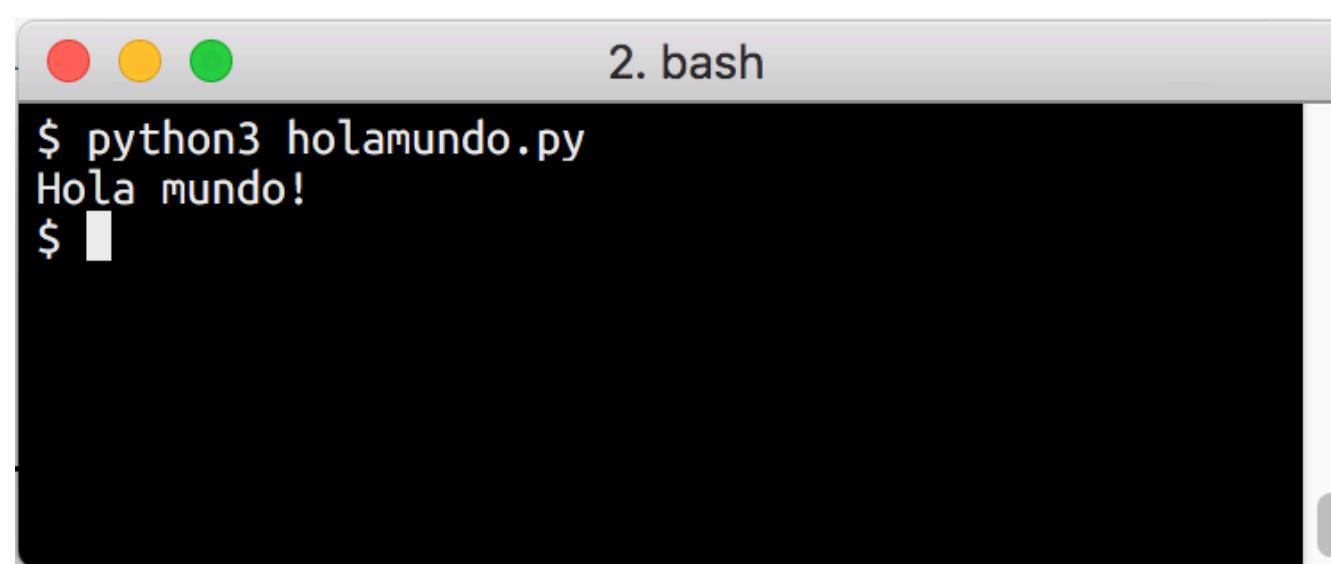
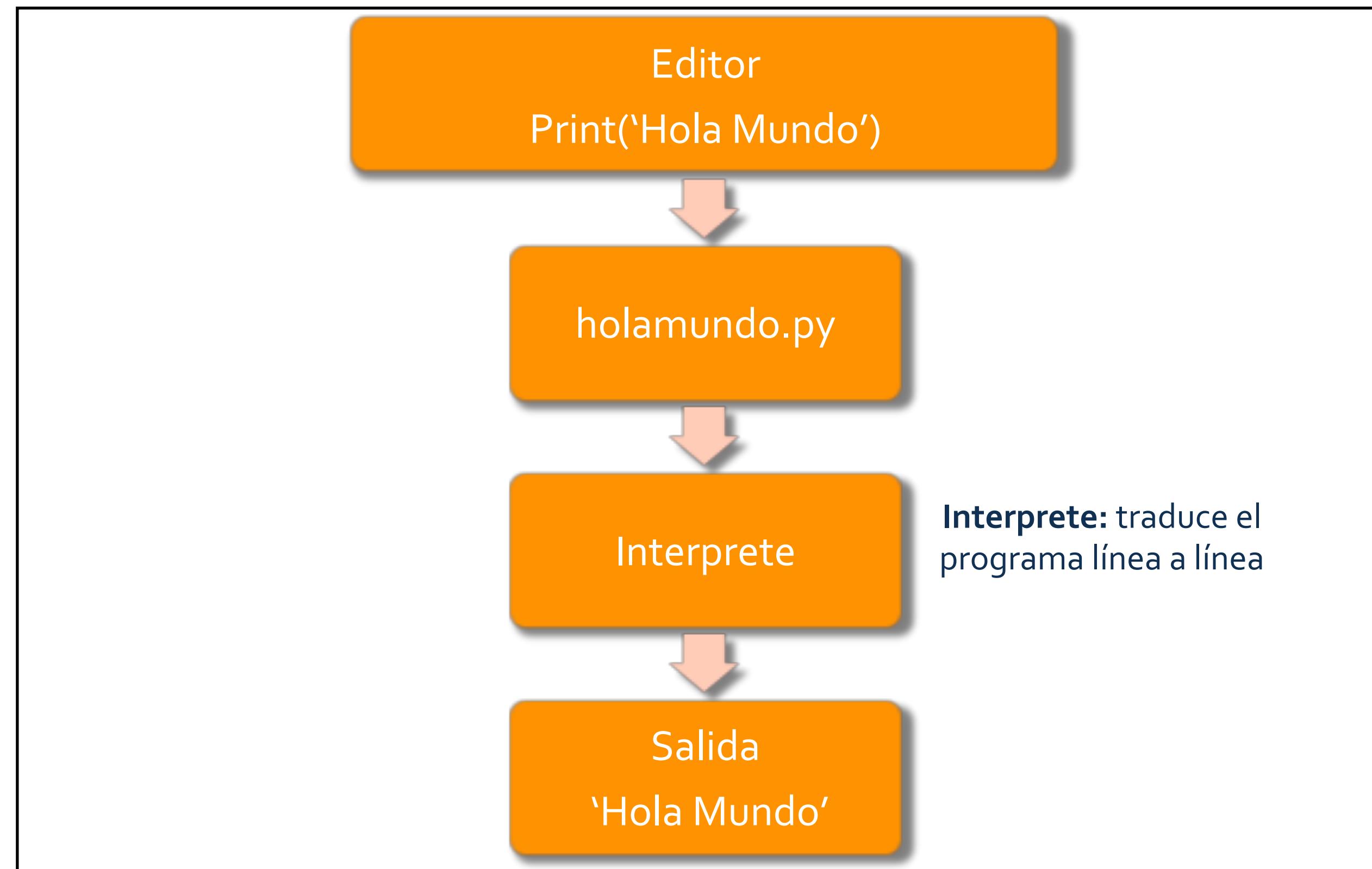
# NIU

Natural user interface: utiliza movimientos gestuales del cuerpo o de alguna de sus partes tales como las manos o los pies.



# Hola mundo

```
# holamundo.py  
print('Hola mundo!')
```

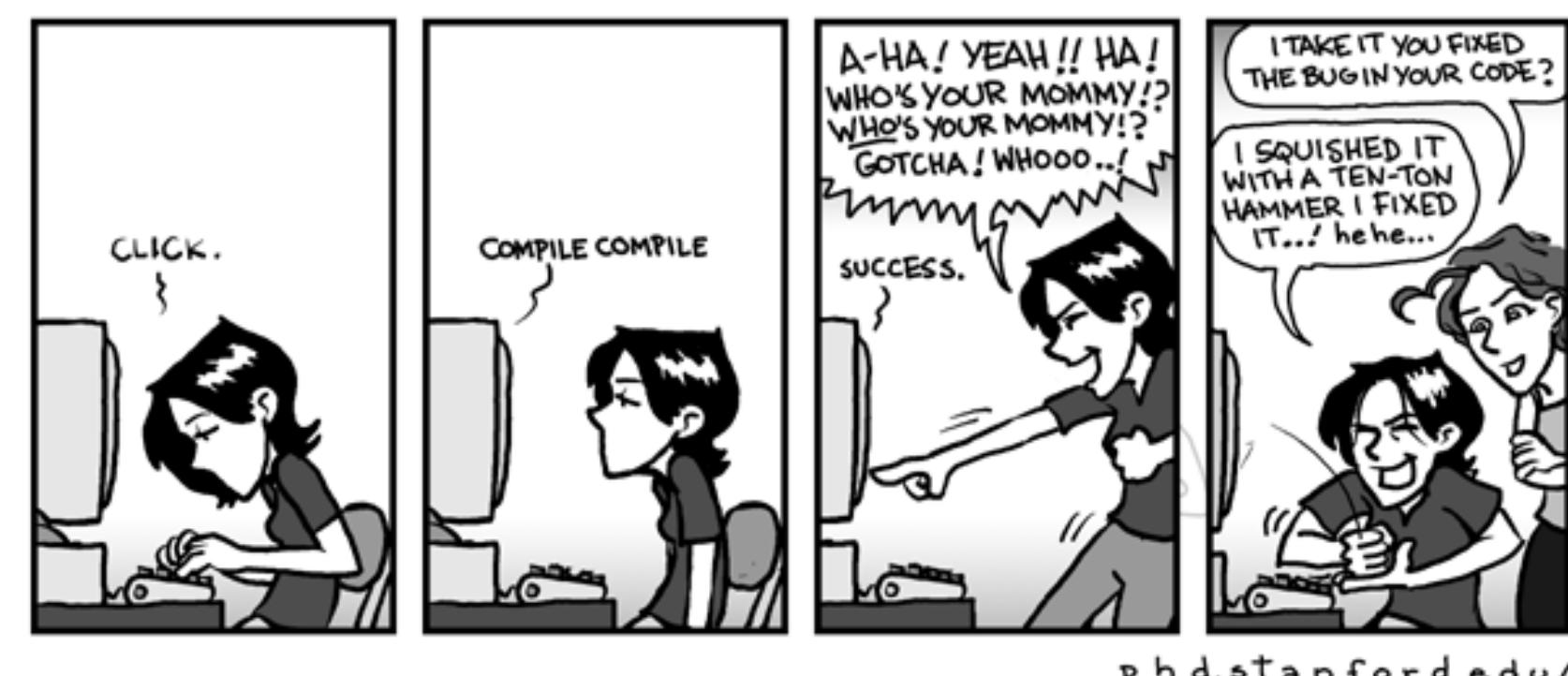


# Sintaxis

Conjunto de reglas que regulan el lenguaje

**Español:** "Juan perro gato" → No es sintácticamente válido  
"Juan abraza al gato" → Sintácticamente válido

**Leng. de programación:** "3.2"5 → No es sintácticamente válido  
3.2\*5 → Sintácticamente válido



# Variables

## ¿Qué es una variable?

Memoria que permite almacenar datos

## Ejemplo

```
nombre = 'Cameron Howe'  
print(nombre)  
'Cameron Howe'
```

02\_variables.py

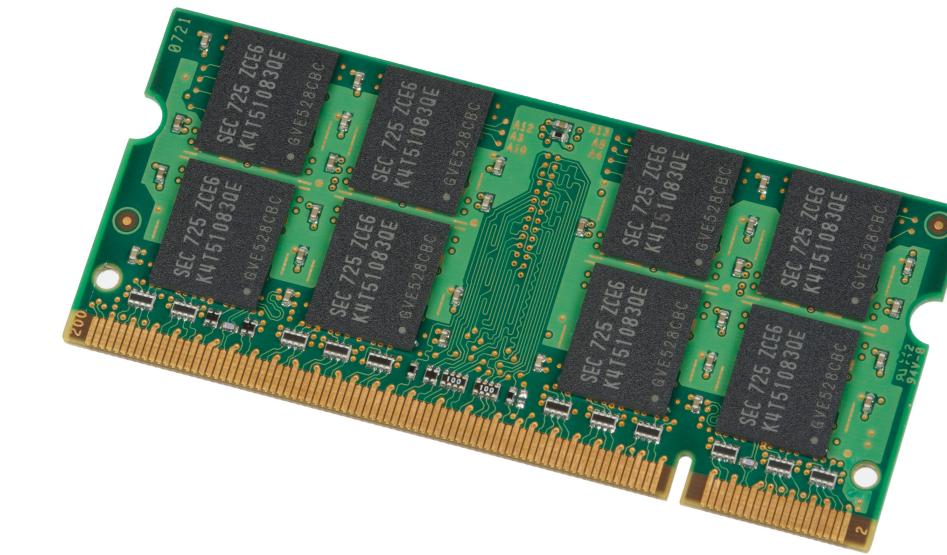


Cameron Howe, programadora protagonista de la serie Halt and Catch Fire.

# Memoria

## ¿Qué entendemos por memoria?

Dispositivo electrónico que permite almacenar datos durante un intervalo de tiempo



1	3	4	4	6	7	0	1
---	---	---	---	---	---	---	---

# Tipos de Datos

- int: representa números **enteros** (ej. 3)
- float: representa números **reales** (ej. 3.27)
- bool: representa valores **boolean** (**True** and **False**)
- str: representa valores **strings** (texto)
- NoneType – representa el valor **None**

Puedes usar **type()** para ver el tipo

# Conversión de Tipos de Datos

Podemos convertir str en números usando:

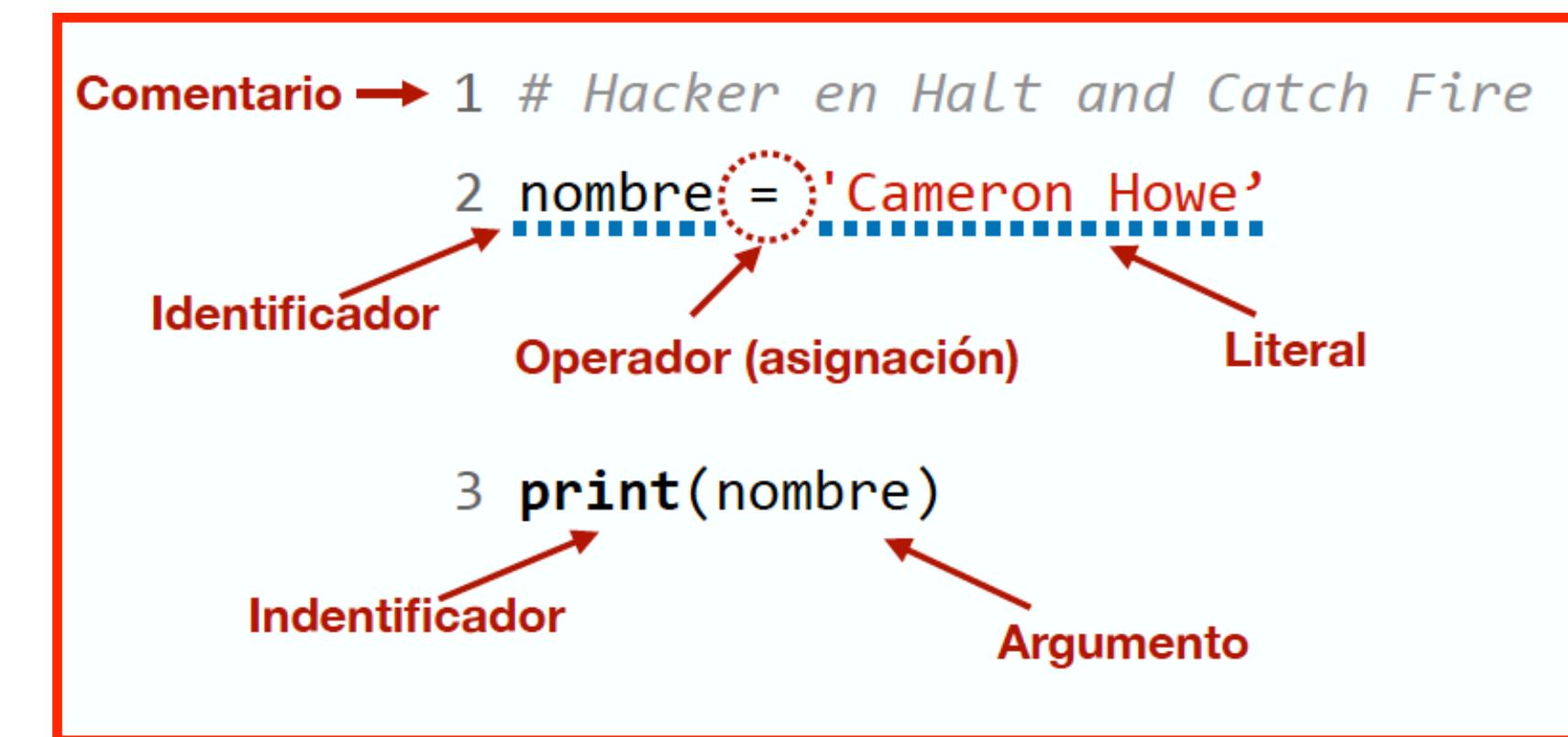
- `int(...)` para números enteros
- `float(...)` para números decimales

Y también podemos convertir números en texto usando `str(...)`

```
nombre = 'Olivia'  
edad1 = '35'  
edad2 = int(edad1)  
print('Hola', nombre, 'veo que tienes', str(edad2), 'años')  
decenas = edad2 // 10  
print('Y al menos tienes', decenas, 'décadas')
```

# Sintaxis

- **Literal:** es una forma sencilla de escribir un valor.
  - "hola" es un texto (tipo de dato **str**)
  - 3.14 es un número **float**
- **Identificador:** nombre que se le puede dar una variable o función.
  - **nombre, print**
- **Variables:** es una referencia a una parte de la memoria que representa algún dato. La usamos para guardar resultados parciales a medida que la computación se realiza.
- **Expresiones:** combinación de variables, literales e identificadores.
- **Traza:** evolución de variables a medida que se ejecuta un programa.



# Identificadores

- Deben comenzar con una letra o con un guión bajo
- Pueden contener dígitos, y guión bajo
  - x, nombre\_apellido, y55, \_holahola
- Pero no pueden comenzar con números o contener caracteres de operacion:
  - 12, 1x, -55, 33mineros, micro\$oft
- Tampoco pueden ser **palabras reservadas**:
  - Ejemplo: **int**, **if**, **return**, **float**...

# Identificadores (Palabras Reservadas)

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

# Identificadores

- Deben ser informativos!
  - Siglas son confusas: `LAX`, `CCP`, `myvar`
  - No use nombres largos: `mi_variable_de_tipo_string`
  - Recuerde que su código será leído por otra persona... incluido su “futuro yo”
- Buenos ejemplos:
  - `edad`, `suma_parcial`, `temperatura`, `process_type`, `current_time`

# Operaciones Sobre Datos

Operador	Significado
+	Suma (o adición)
-	Resta (o sustracción)
*	Multiplicación
**	Exponente
%	Módulo (resto)
/	División
//	División entera

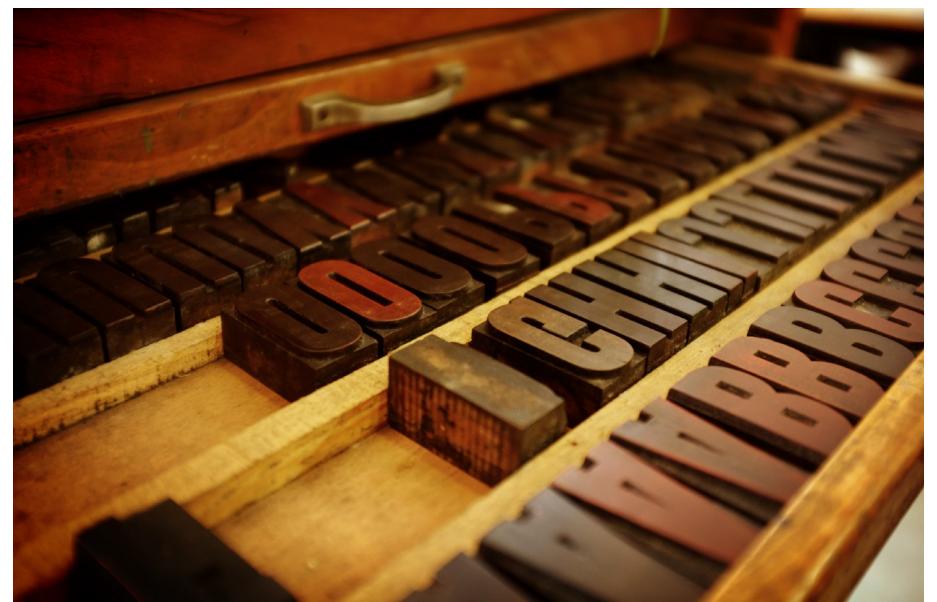
Texto	Números
+ (concatenar)	+ , - , / , * , % , //
+= (agregar al final)	+=, -=, *=, etc ...

Fuente: [https://es.wikibooks.org/wiki/Python/Generalidades/  
Palabras reservadas, operadores y simbolos del lenguaje](https://es.wikibooks.org/wiki/Python/Generalidades/Palabras_reservadas,_operadores_y_simbolos_del_lenguaje)

# Operaciones Datos

```
1 primero = 'Daniela'  
2 segundo = 'Opitz'  
3 nombre = primero + ' ' + segundo  
4 print('Tu nombre es', nombre)
```

Operaciones sobre **strings**



Daniela Opitz

04\_operaciones.py

# Entrada y Salida de Datos

Funciones

de entrada/salida:

`input(...)`

recibe datos por teclado

`print(...)`

imprime datos en la pantalla

```
nombre = input('Indica tu nombre: ')
print('Tu nombre es', nombre)
```

La función `input` por defecto importa variables de tipo `str` (texto)

# Entrada y Salida de Datos

## Entrada de números

```
nombre = input('Indica tu nombre: ')
edad = int(input('Indica tu edad: '))
print('Hola', nombre, 'veo que tienes', edad, 'años')
decenas = edad // 10
print('Y al menos tienes', decenas, 'decadas')
```

**P:** ¿Qué sucede si eliminamos el comando `int` en la línea 2 de la entrada de números?

**R:** Error en línea 4, no puedes hacer una división con un texto!



**07\_enteros\_in\_out.py**

# Entrada y Salida de Datos

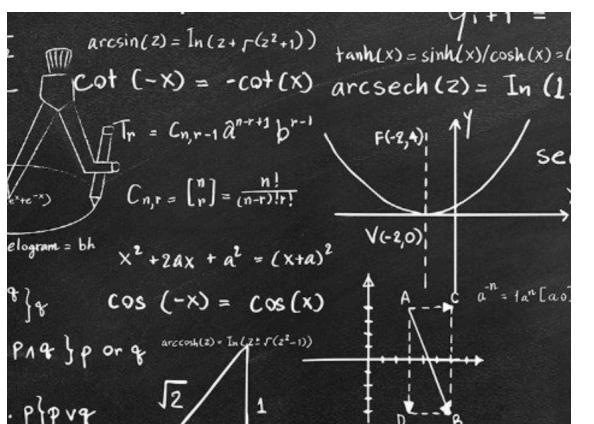
```
amigos = input('¿Cuántos amigues tienes en facebook?')
comun  = input('¿Cuántos amigues tienes en comun con tu mejor amigue?')

num_amigos = int(amigos)
num_comun  = int(comun)
porcentaje = num_comun/num_amigos*100

print('Wow, tienes', porcentaje, '% de amigues en común con tu mejor amigue!')

print('tipo variable amigos:', type(amigos))
print('tipo variable num_amigos:', type(num_amigos))
```

# Entrada y Salida de Datos



**math:** modulo con métodos matemáticos que importamos con el comando **import**



```
1 # En la siguiente linea importamos la funcion raiz cuadrada.  
2 # Revisa otras funciones matemáticas  
3 # en https://docs.python.org/3/library/math.html  
4 from math import sqrt  
5  
6 n = float(input('Ingrese un numero decimal: '))  
7 print('n = ', n)  
8 print('5 por n = ', 5*n)  
9 print('n dividido dos = ', n/2)  
10 print('raiz cuadrada de n = ', sqrt(n))
```

Revisar

<https://docs.python.org/2/library/math.html>

# Uso de la Consola

- **Windows**

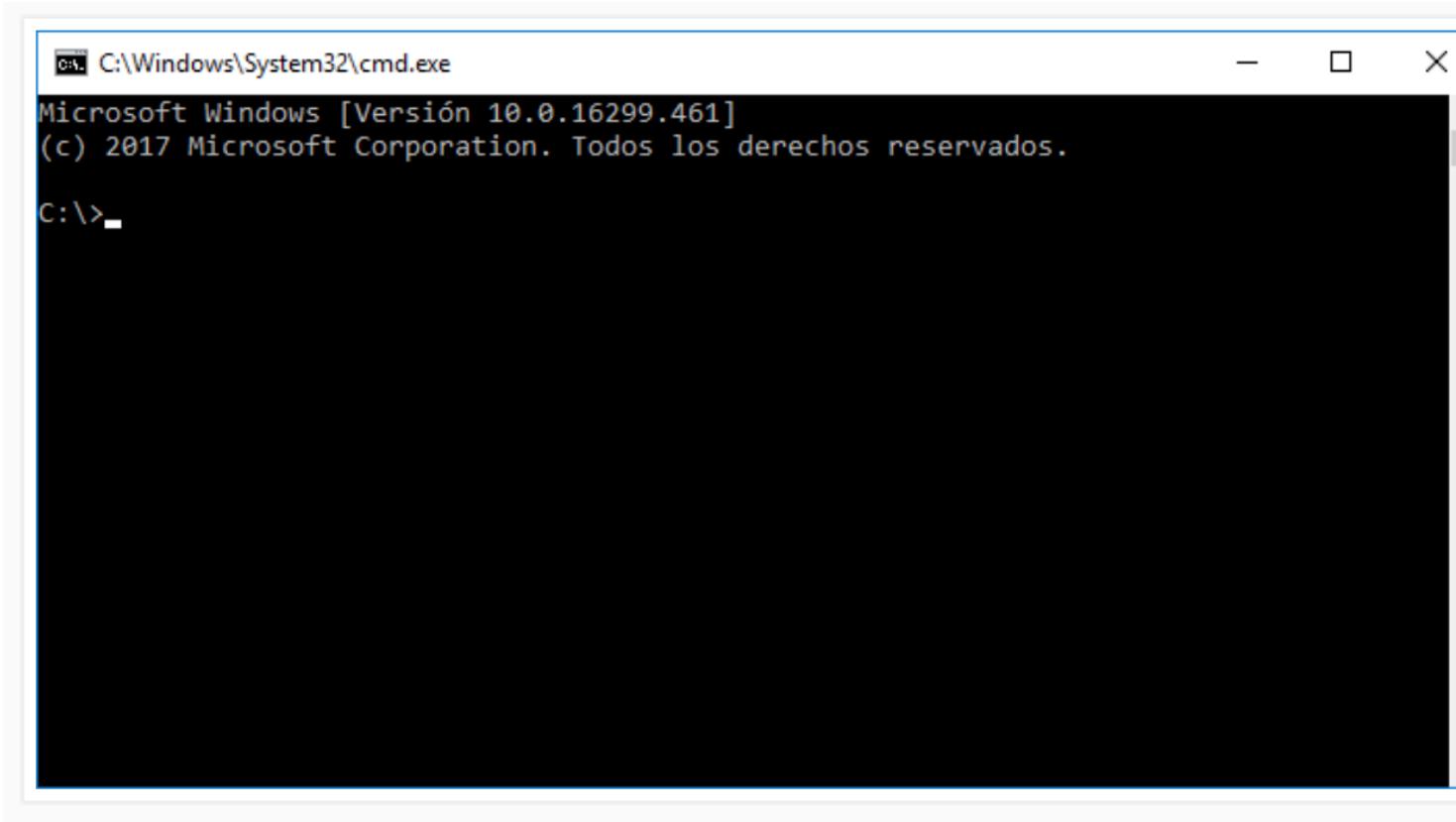
>>**cd**: Cambia a otro directorio

>>**dir**: Muestra una lista de archivos y subdirectorios en un directorio

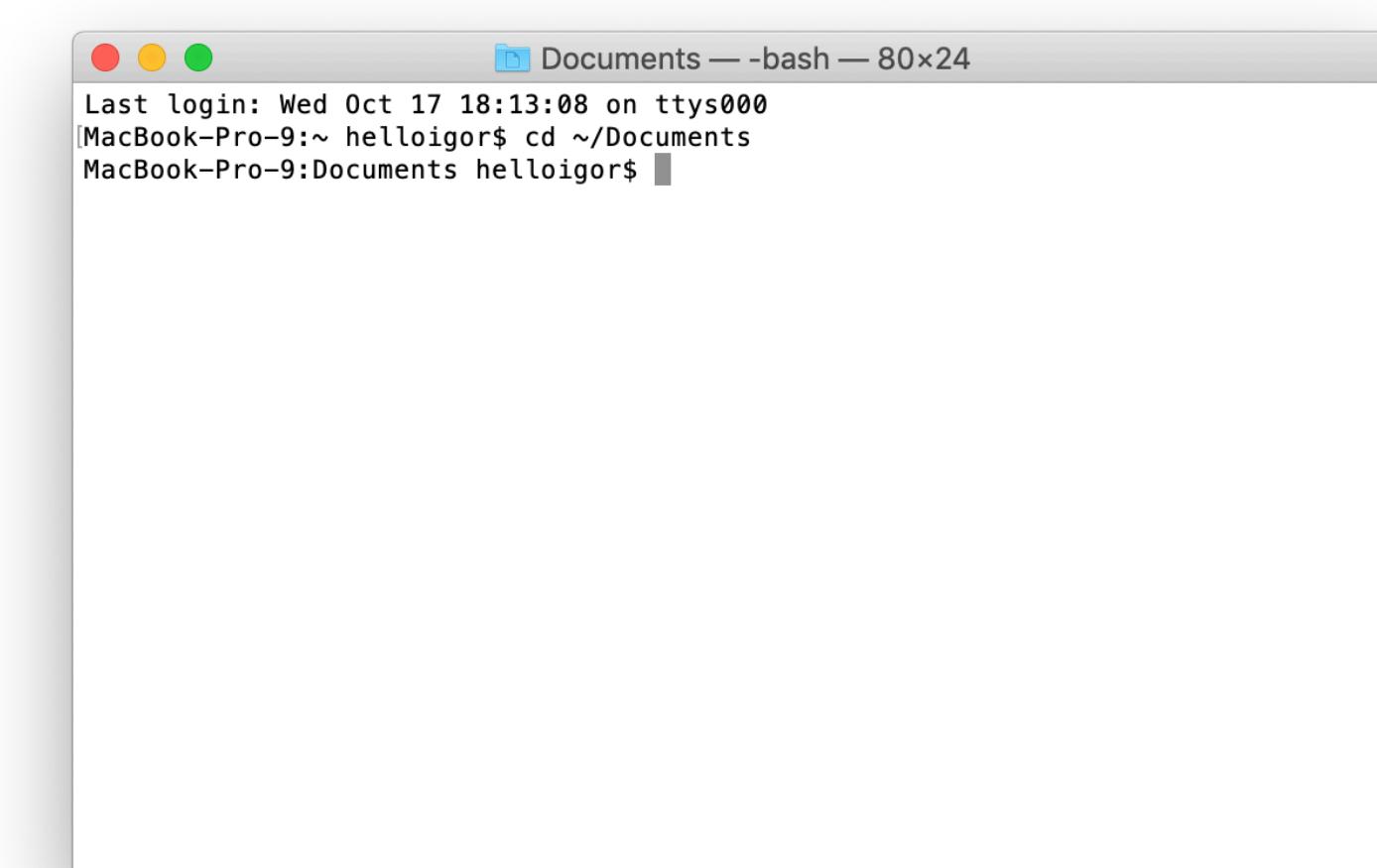
- **Mac**

>>**cd**: Cambia a otro directorio

>>**ls**: Muestra una lista de archivos y subdirectorios en un directorio



**cmd (Windows)**



**Terminal (Mac)**

# Actividades

1. Crea un programa que calcule la solución de una ecuación de segundo grado de la forma:

$$ax^2 + bx + c = 0$$

- Asume que el usuario entrega el valor de a, b y c.
- Recuerda que la raíz se puede calcular con:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

2. Utilizando el archivo 10\_argumento.py

- Explóralo y ejecútalo desde la consola
- Modifícalo de modo que imprima tres nombres

# Resumen

## Tipos de datos

- **int**: números enteros. Ej: -1, 10, 12121
- **float**: números decimales. Ej: 1.1, 23.99, 2.0
- **str**: texto. Ej: 'Ingeniería Civil', '😊'
- **bool**: binario: Ej: True, False ← **La mayúscula es obligatoria!**

## Funciones

- **input(...)**: captura de datos desde el teclado
- **print(...)**: imprimir en pantalla
- **type(...)**: verificar el tipo de dato de una variable o literal

## Funciones matemáticas

```
>>> import math
>>> dir(math)
['__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__', 'acos',
 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos',
 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial',
 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite',
 'isinf', 'isnan', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf',
 'nan', 'pi', 'pow', 'radians', 'remainder', 'sin', 'sinh', 'sqrt', 'tan',
 'tanh', 'tau', 'trunc']
```

## Conceptos

- **concatenar**: unir dos variables de tipo str
- **entrada/salida**: ingresar y extraer datos
- **traza**: seguimiento de variables según se ejecuta el programa

## Sintaxis

```
var_texto = 'literal con text'
var_entero = -112
var_float = 3.1416

#concatenar
t1 = 'mi casa'
t2 = 'es bonita'
texto = t1 + ' ' + t2 # 'mi casa es bonita'

#operaciones aritméticas
suma = 1+2 # -, *, /
div = 1 // 2 # division entera
mod = 1 % 2 # resto
```

Más info en <https://docs.python.org/3/library/math.html>