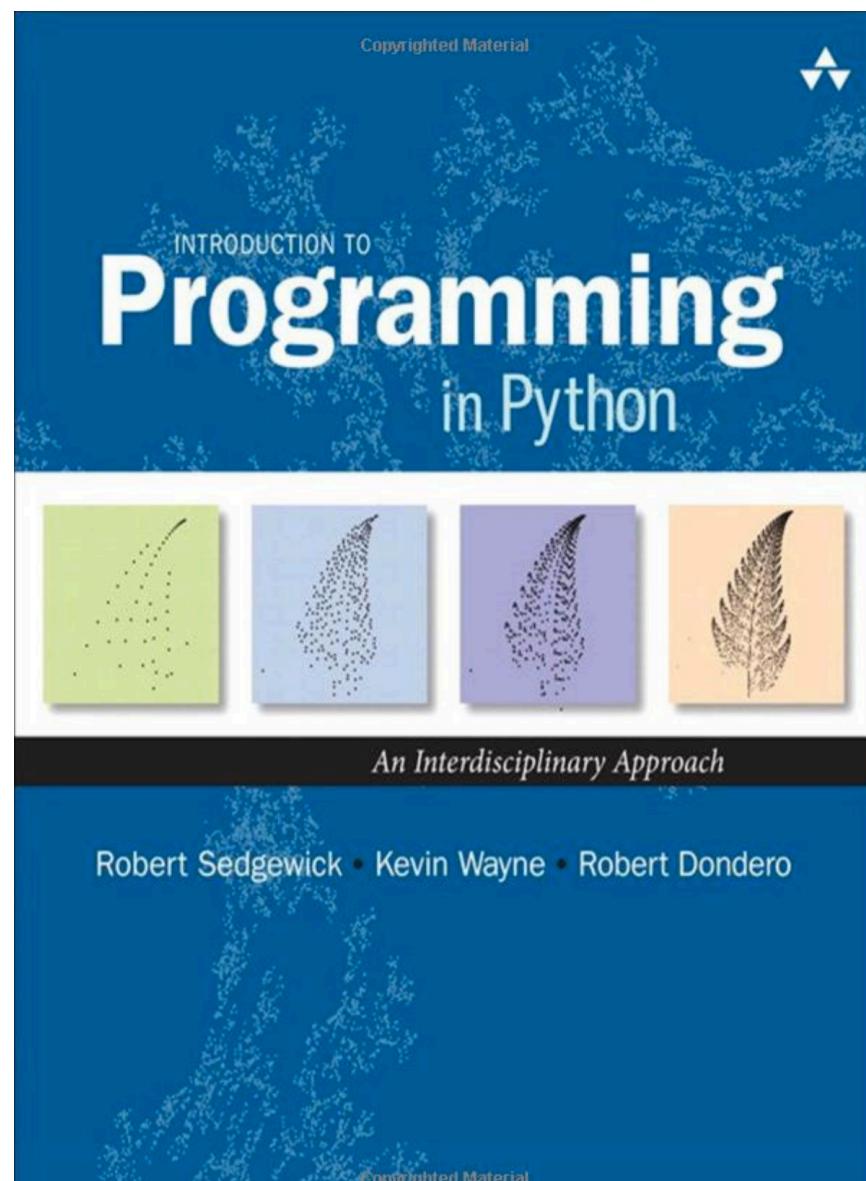


# Parte I: Intro pensamiento computacional

## Clase 01: Variables y tipos de datos

Diego Caro, Daniela Opitz  
[dcaro@udd.cl](mailto:dcaro@udd.cl) [dopitz@udd.cl](mailto: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>

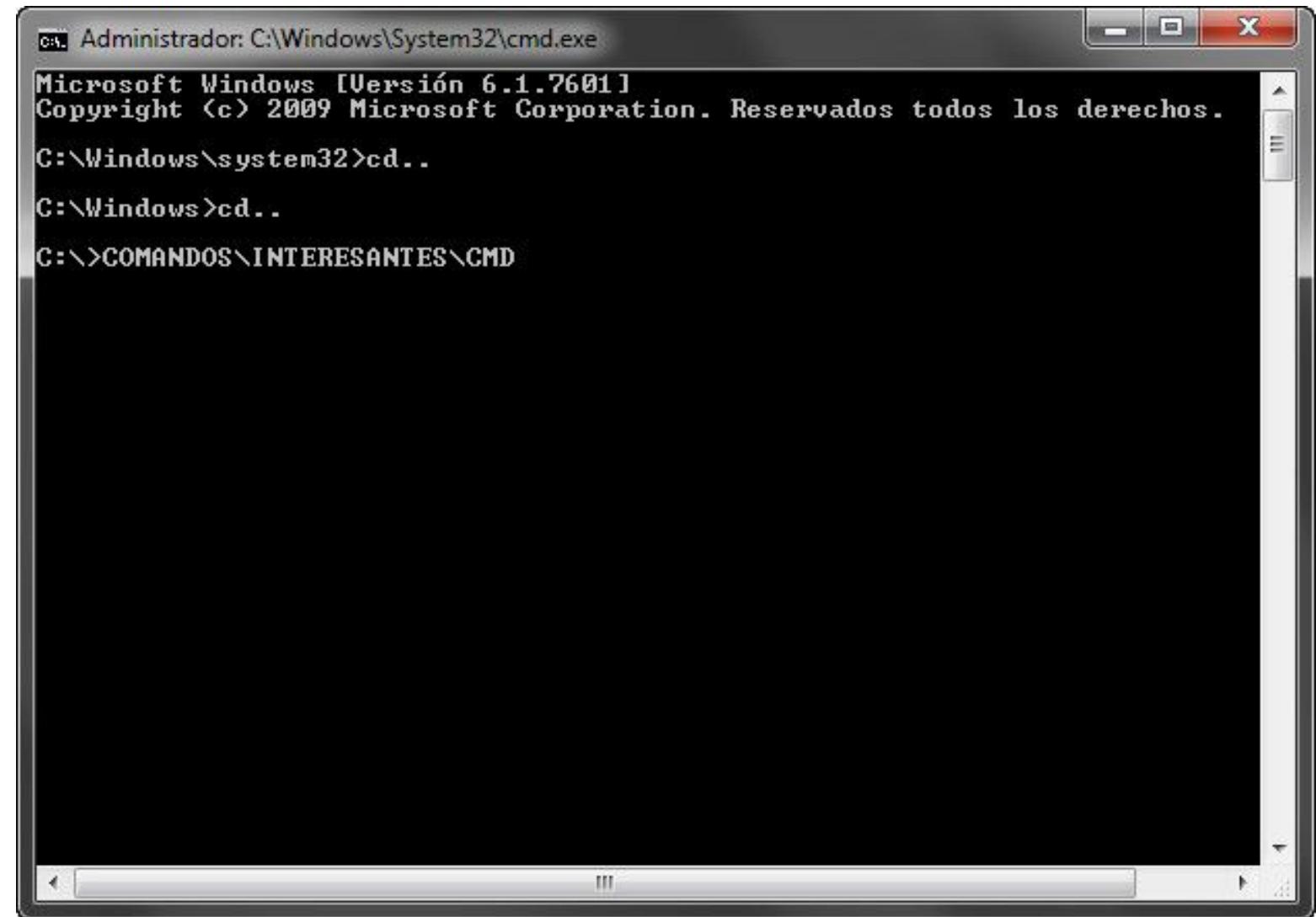
# Objetivos de la clase

- Identificar diferentes tipos de interfaces de usuario
- Reconocer el concepto de variable y tipo de datos
- Identificar tipos de datos numéricos y de texto
- Utilizar operadores de aritmética básicos
- Reconocer uso de mecanismo para entrada/salida de datos

# Interfaces de usuario

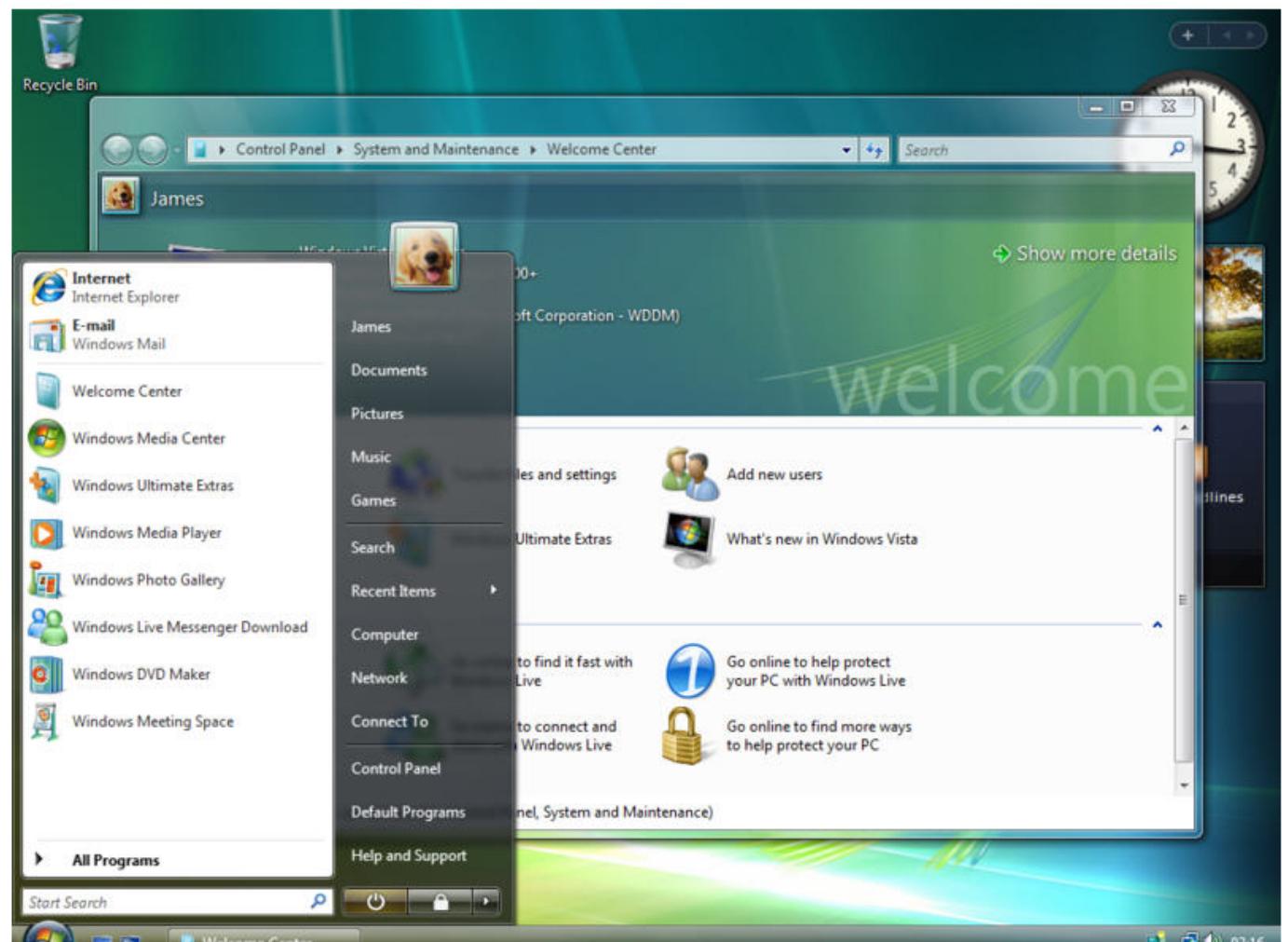
- **Shell:** intérprete de comandos. Es una interfaz de usuarios para acceder a funciones del sistema operativo. Ejemplo:
  - Leer archivos desde un pendrive
  - Dibujar imágenes en el monitor
- Tipos de Shell:
  - Interfaz de línea de comando (CLI)
  - Interfaz gráfica de usuario (GUI)

CLI →



```
c:\ Administrador: C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.
C:\Windows\system32>cd..
C:\Windows>cd..
C:\>COMANDOS\INTERESANTES\CMD
```

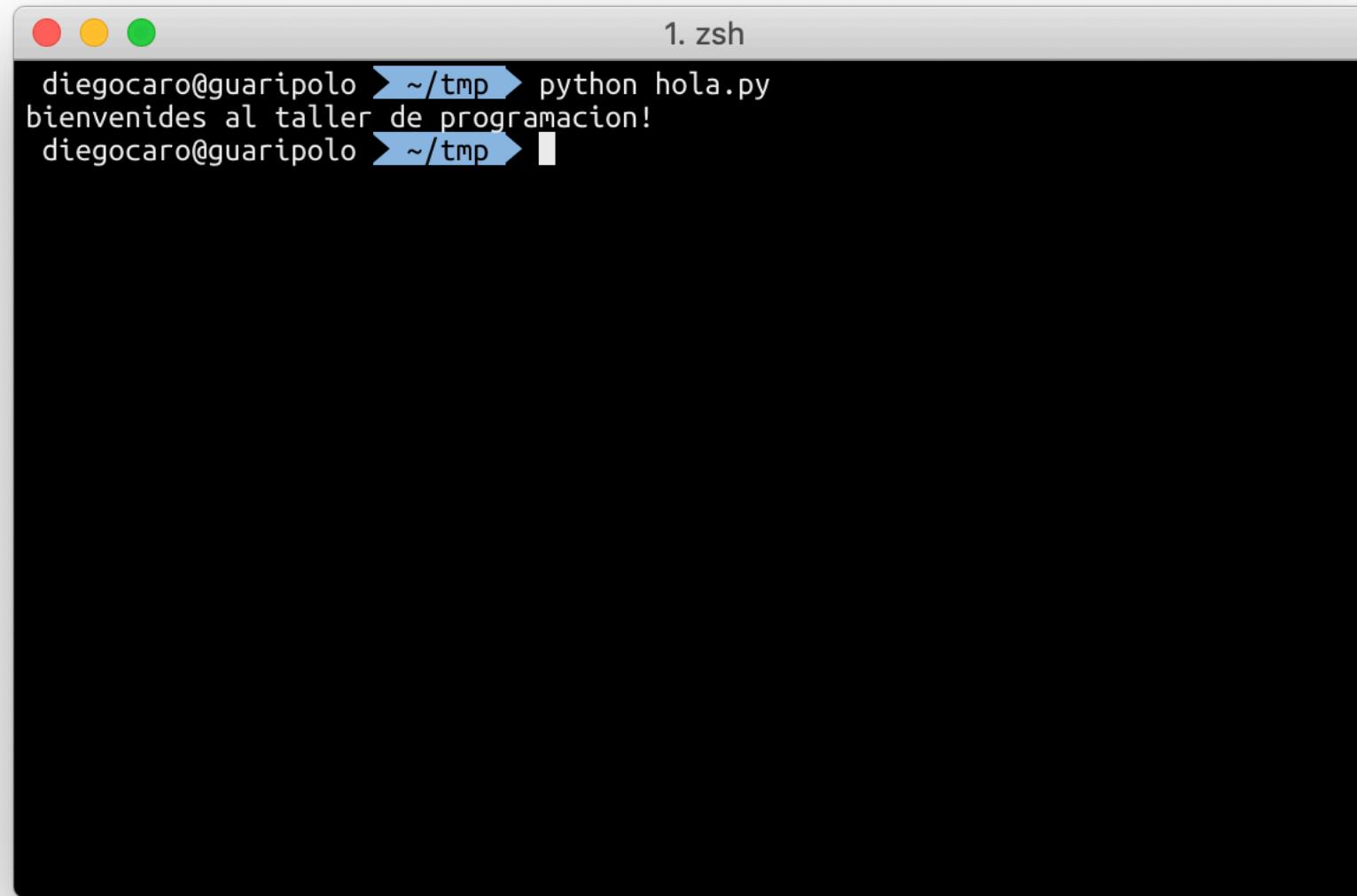
GUI →



# Interfaces para programar en Python

- Para editar código utilizaremos el programa Spyder
- Y también ejecutaremos código desde la línea de comandos.

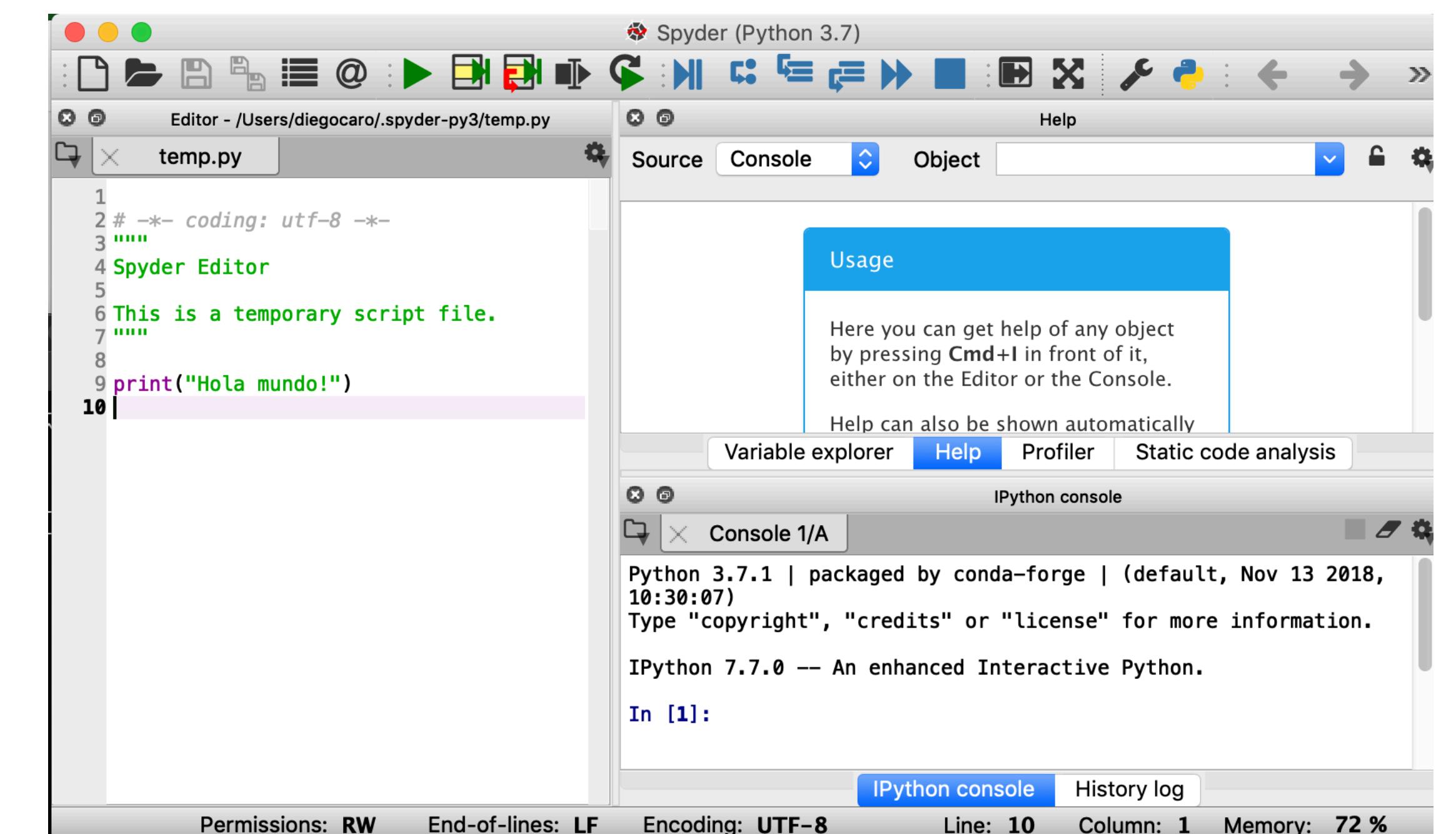
**CLI**



```
1. zsh
diegocaro@guaripolo ~tmp python hola.py
bienvenides al taller de programacion!
diegocaro@guaripolo ~tmp
```

Ejecutando código en línea de comandos

**GUI**



Spyder (Python 3.7)

Editor - /Users/diegocaro/spyder-py3/temp.py

temp.py

```
1
2 # -*- coding: utf-8 -*-
3 """
4 Spyder Editor
5
6 This is a temporary script file.
7
8
9 print("Hola mundo!")
10
```

Source Console Object

Usage

Here you can get help of any object by pressing Cmd+I in front of it, either on the Editor or the Console.

Help can also be shown automatically

Variable explorer Help Profiler Static code analysis

IPython console

Console 1/A

```
Python 3.7.1 | packaged by conda-forge | (default, Nov 13 2018, 10:30:07)
Type "copyright", "credits" or "license" for more information.

IPython 7.7.0 -- An enhanced Interactive Python.

In [1]:
```

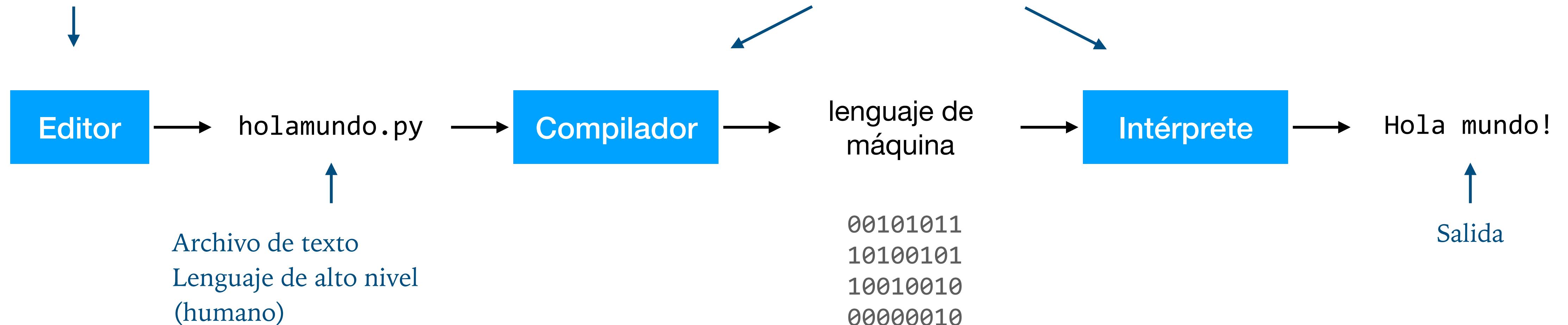
IPython console History log

Permissions: RW End-of-lines: LF Encoding: UTF-8 Line: 10 Column: 1 Memory: 72 %

Spyder en MacOS

# Escribiendo un programa

Puedes utilizar cualquier editor de texto!



```
2. bash
$ python3 holamundo.py
Hola mundo!
$
```

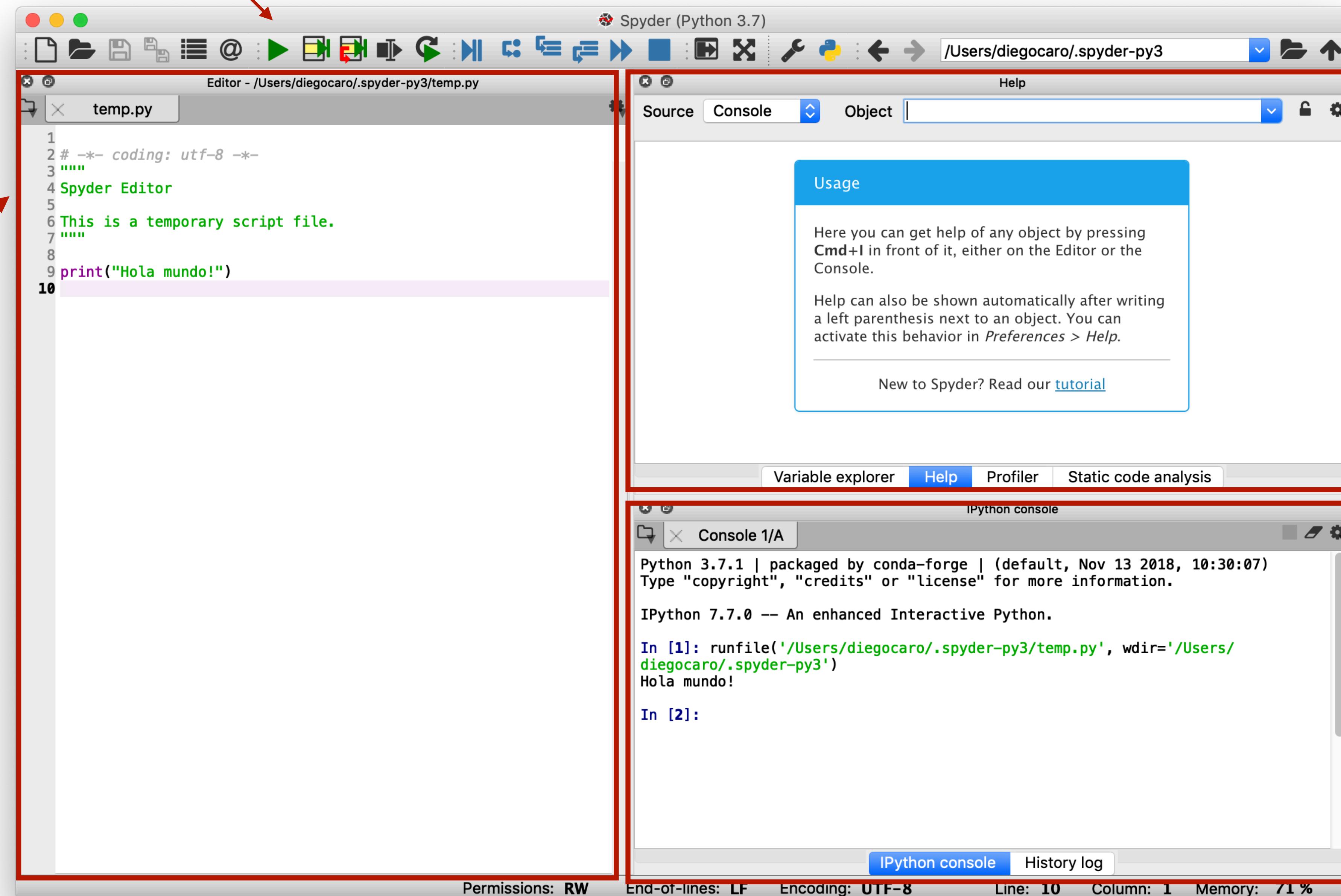
```
IPython console
Console 1/A
Python 3.7.1 | packaged by conda-forge | (default, Nov 13 2018, 10:30:07)
Type "copyright", "credits" or "license" for more information.

IPython 7.7.0 -- An enhanced Interactive Python.

In [1]: runfile('/Users/diegocaro/.spyder-py3/temp.py', wdir='/Users/diegocaro/.spyder-py3')
Hola mundo!

In [2]:
```

## Ejecutar código



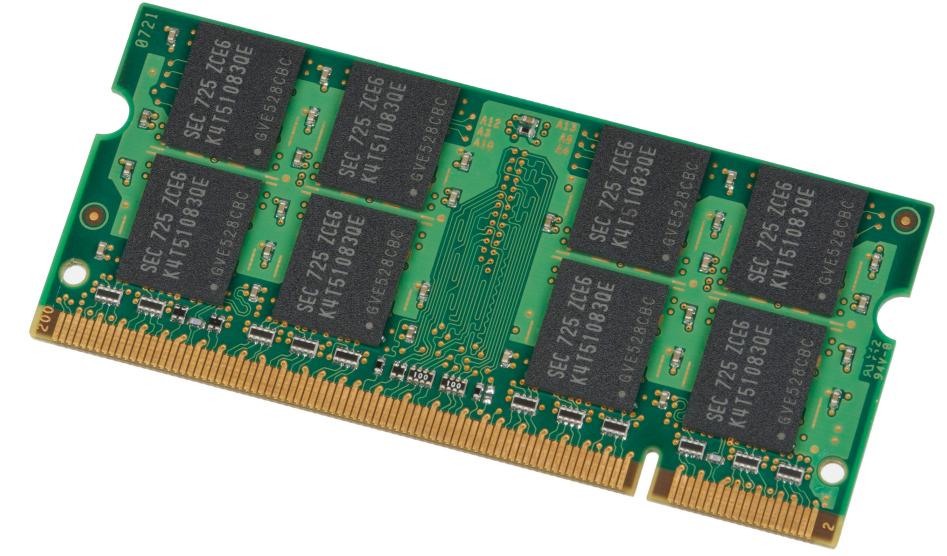
## Editor de código

## Explorador de Variables y Documentación

## Intérprete

# Memoria

- Dispositivo electrónico que permite almacenar datos durante un intervalo de tiempo. Ejemplos:
  - memoria RAM (memoria de acceso directo, toma el mismo tiempo acceder a cada lugar de la memoria)
- Los números, letras y todo lo que vemos se almacenan en memoria!
  - Ejemplo de memoria muy pequeña



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

# Sintaxis: primer programa en Python

- Sintaxis: reglas de cómo escribir en algún lenguaje/idioma.
  - El español tiene reglas... Por ejemplo:
  - Python tiene su propia sintaxis!

La casa verde  
\_\_\_\_\_  
artículo sustantivo adjetivo



Comentario → 1 # Hacker en Halt and Catch Fire  
2 nombre = 'Cameron Howe'  
3 print(nombre)

Identificador      Operador (asignación)      Literal  
Identificador      Argumento

P: ¿Qué es una **variable**?  
R: Memoria que permite almacenar y consultar datos!

nombre: 'Cameron Howe'

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

# Entrada y salida de datos

## Entrada de texto

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

## 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')
```

- Funciones de entrada/salida:
  - `input(...)` recibe datos por teclado
  - `print(...)` imprime datos en la pantalla
- Funciones de cambios de tipo:
  - `int(...)` transforma texto en número
  - `str(...)` transforma número a texto

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

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

# Operaciones sobre tipos de datos

- Operaciones en tipos de datos string (texto):
  - Concatenar (unir): +
  - Agregar al final: +=
- Operaciones en tipos de datos numéricos:
  - Aritmética: +, -, /, \*, %, //
  - Y también se puede usar +=, -=, \*=, etc...

values	integers						
typical literals	1234	99	0	1000000			
operations	sign	add	subtract	multiply	floored divide	remainder	power
operators	+	-	*	//	%	**	
Python's int data type							
values	real numbers						
typical literals	3.14159	6.022e23	2.0	1.4142135623730951			
operations	addition	subtraction	multiplication	division	exponentiation		
operators	+	-	*	/	**		
Python's float data type							

## Manejo de texto

```
1 primero = input('Indica tu nombre: ')
2 segundo = input('Indica tu apellido: ')
3 nombre = primero + ' ' + segundo
4 print('Tu nombre es', nombre)
```

## Operaciones aritméticas

```
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))
```

**Nota:** la operación/funcionalidad se realiza dependiendo del tipo de dato asociado (ej: entero o string).

# Conversión de tipos de datos

- La función `input` por defecto representa variables de tipo `str` (texto).
- Podemos convertir `str` en números usando:
  - `int(...)` para números enteros
  - `float(...)` para números decimales
- Y también podemos convertir números en texto, usando `str(...)`
- **Nota:** la función `type(variable)` indica el tipo de dato de una variable

```
1 amigos = input('Cuantos amigues tienes en facebook? ')
2 comun = input('Cuantos amigues tienes en comun con tu mejor amigue? ')
3
4 num_amigos = int(amigos)
5 num_comun = int(comun)
6 porcentaje = num_comun/num_amigos*100
7 print('Wow, tienes', porcentaje, '% de amigues en comun con tu mejor amigue!')
8
9 print('tipo variable amigos:', type(amigos))
10 print('tipo variable num_amigos:', type(num_amigos))
```

**P:** Qué sucede si no usamos la conversión `int(...)` ?

**R:**

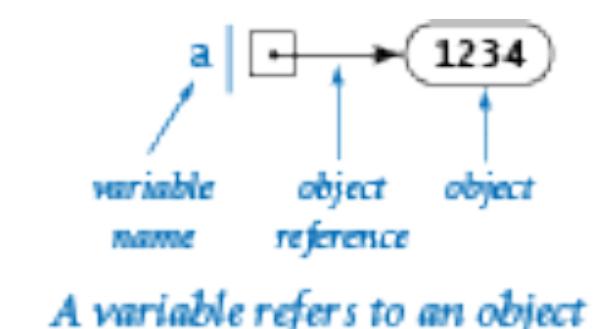
```
Cuantos amigues tienes en facebook? 12
Cuantos amigues tienes en comun con tu mejor amigue? 1
Traceback (most recent call last):
  File "conversion-bad.py", line 4, in <module>
    porcentaje = num_comun/num_amigos*100
TypeError: unsupported operand type(s) for /: 'str' and 'str'
```

# Un poco más de sintaxis

- **Literal:** código que representa un tipo de dato.
  - "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,
  - Hay algunas reglas que debemos cumplir...
- **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.

type	set of values	common operators	sample literals
int	integers	+ - * // % **	99 12 2147483647
float	floating-point numbers	+ - * / **	3.14 2.5 6.022e23
bool	true-false values	and or not	True False
str	sequences of characters	+	'AB' 'Hello' '2.5'

Basic built-in data types

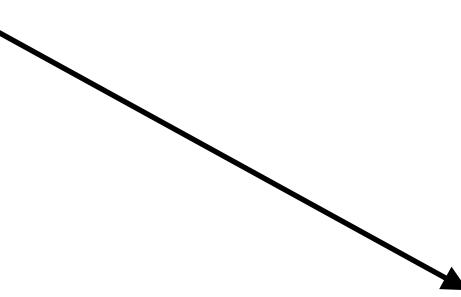


a	b	c
a = 1234	1234	
b = 99	1234	99
c = a + b	1234	99 1333

Your first informal trace

# Sintaxis: 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...
- Ejemplos:
  - edad; suma\_parcial;
  - nombre; temperatura;



- Deben ser informativos!
- Siglas son confusas: LAX, CCP, myvar
- No use nombres largos: mi\_variable\_de\_tipo\_string
- Buenos ejemplos:
  - process\_type, is\_moving, current\_time
- **Nota:** Recuerde que su código será leído por otra persona... Incluído su “futuro yo”

## Palabras reservadas en Python

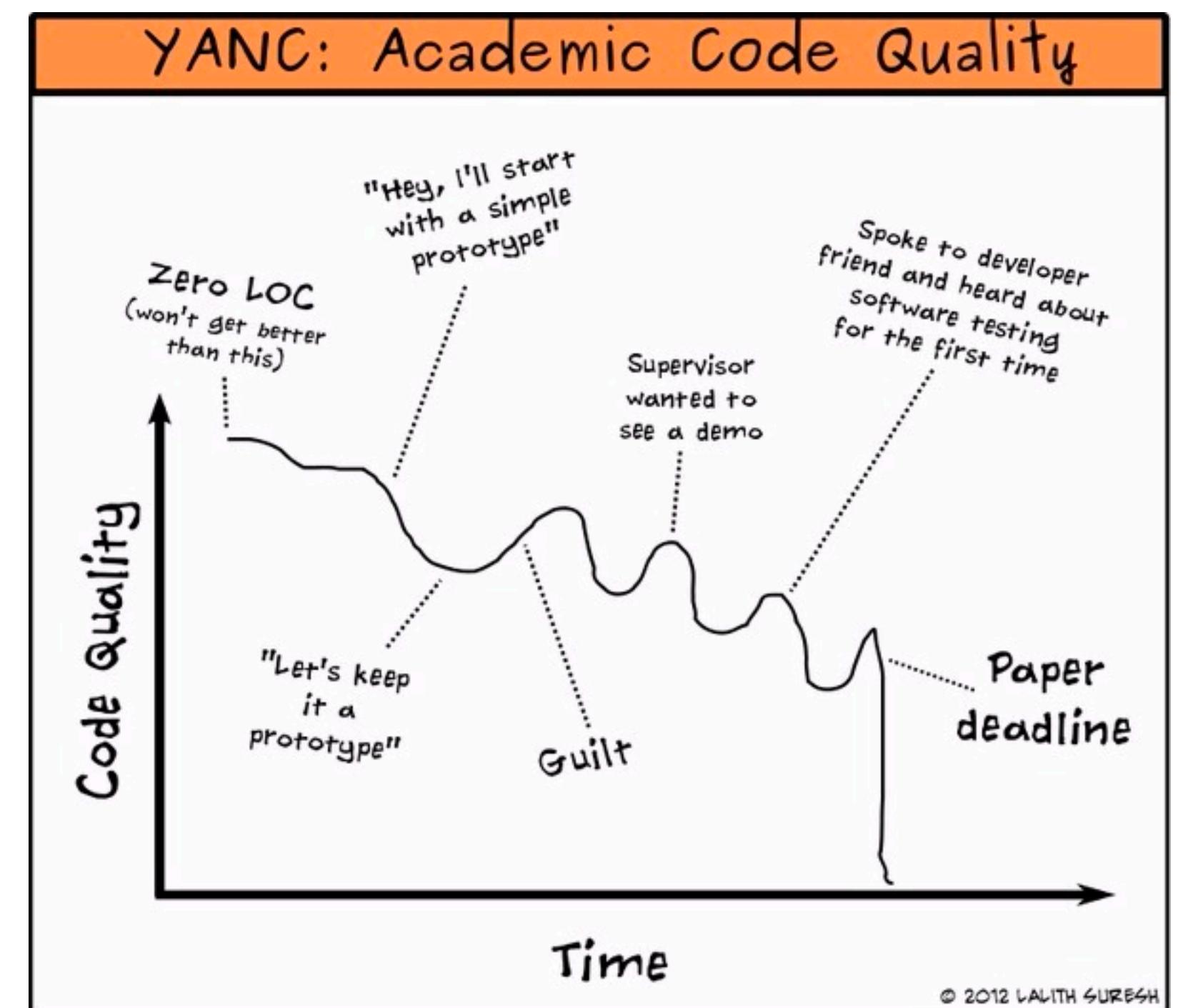
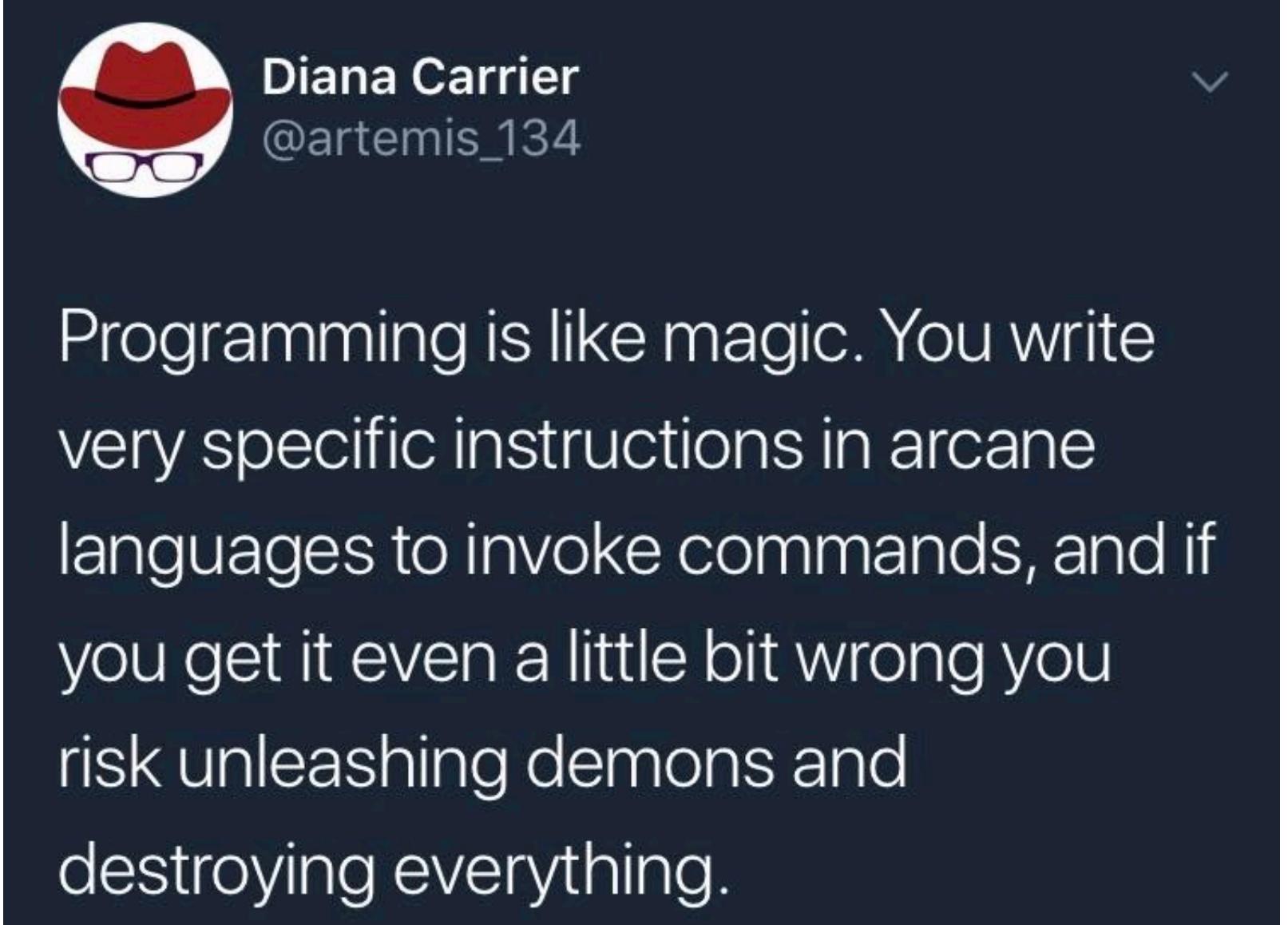
False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

# Actividad

- Escribe un programa que imprima:

```
**      ***      *****      **      *      **
**      ***      **      **      ***      **
**      ***      **      **      ***      **
**      ***      **      **      ***      **
*****      **      **      **      **      **
**      ***      **      **      **      **
**      ***      **      **      **      **
**      ***      **      **      ***      ***
**      ***      *****      *      *
```

- 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  $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$
  - Mira el archivo aritmetica.py
- Modifica la segunda línea del archivo entero.py
  - ¿Qué sucede? ¿Qué significa el error?



# 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']
```

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

## Conceptos

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

## Sintáxis

```
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
```