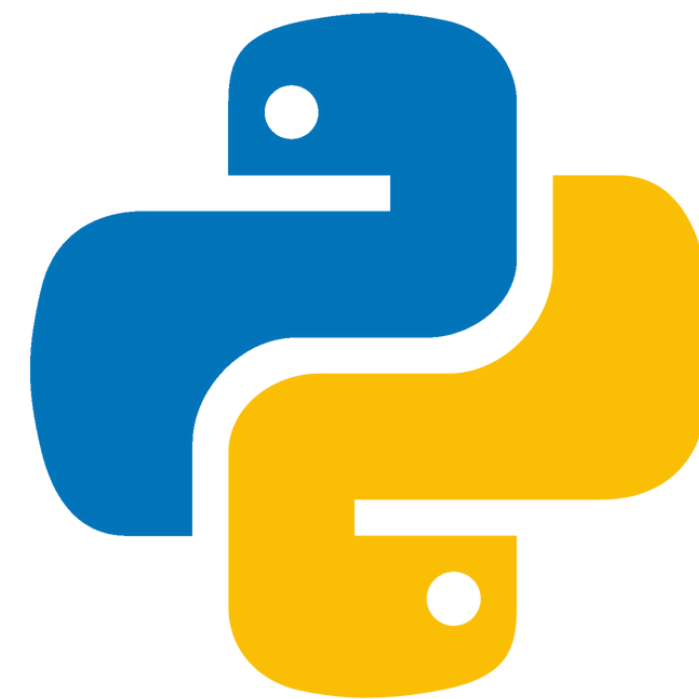


TEMA 0 - INTRODUCCIÓN A PYTHON

Algoritmos













¿Por qué Python?

- **Simplicidad:** código compacto y limpio
- **IDE:** Pycharm es un IDE completo para el desarrollo y depuración de código Python
- **Multiparadigma:** puede utilizarse tanto de manera imperativa como orientada a objetos
- **Estructuras de datos:** tiene implementada una colección de las estructuras más utilizadas



¿Por qué Python?

- **Prototipado rápido:** similar a pseudocódigo
- **Extensiones:** módulos que amplían la funcionalidad del lenguaje
- **Multiplataforma:** Windows, Mac OS, Linux, ...
- **Muy extendido:** muy solicitado en entornos profesionales

Aug 2025	Aug 2024	Change	Programming Language		Ratings	Change
1	1			Python	26.14%	+8.10%
2	2			C++	9.18%	-0.86%
3	3			C	9.03%	-0.15%
4	4			Java	8.59%	-0.58%
5	5			C#	5.52%	-0.87%
6	6			JavaScript	3.15%	-0.76%
7	8	^		Visual Basic	2.33%	+0.15%
8	9	^		Go	2.11%	+0.08%
9	25	^^		Perl	2.08%	+1.17%
10	12	^		Delphi/Object Pascal	1.82%	+0.19%

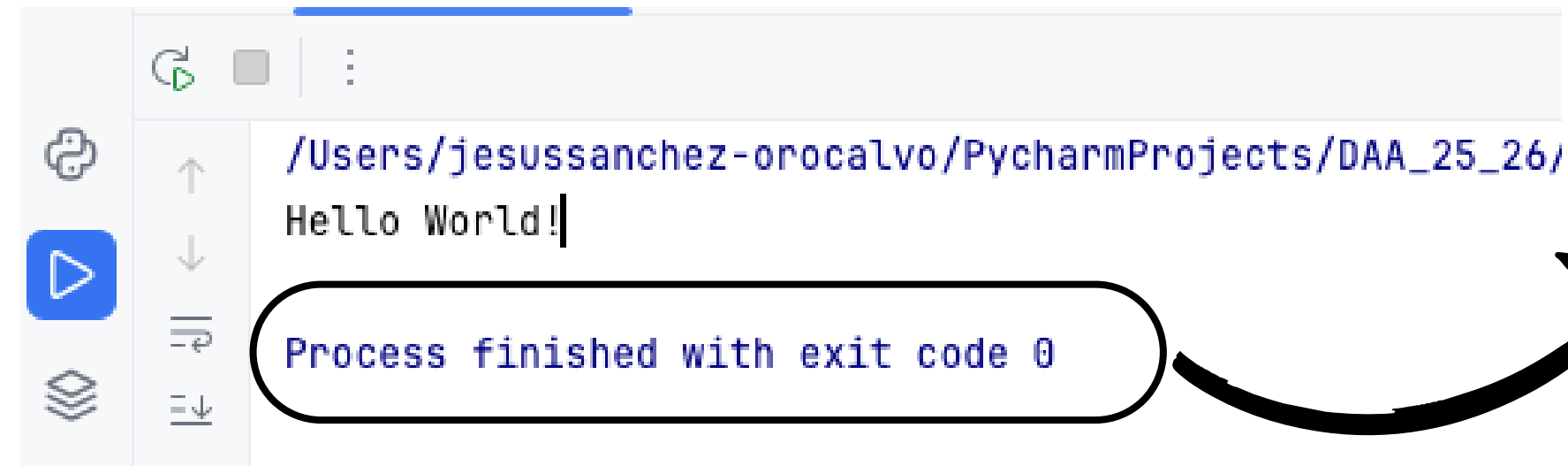
<https://www.tiobe.com/tiobe-index/>

Hello World!

- ¿Cómo implemento el clásico Hello World en Python?

```
1 print("Hello World!")
```

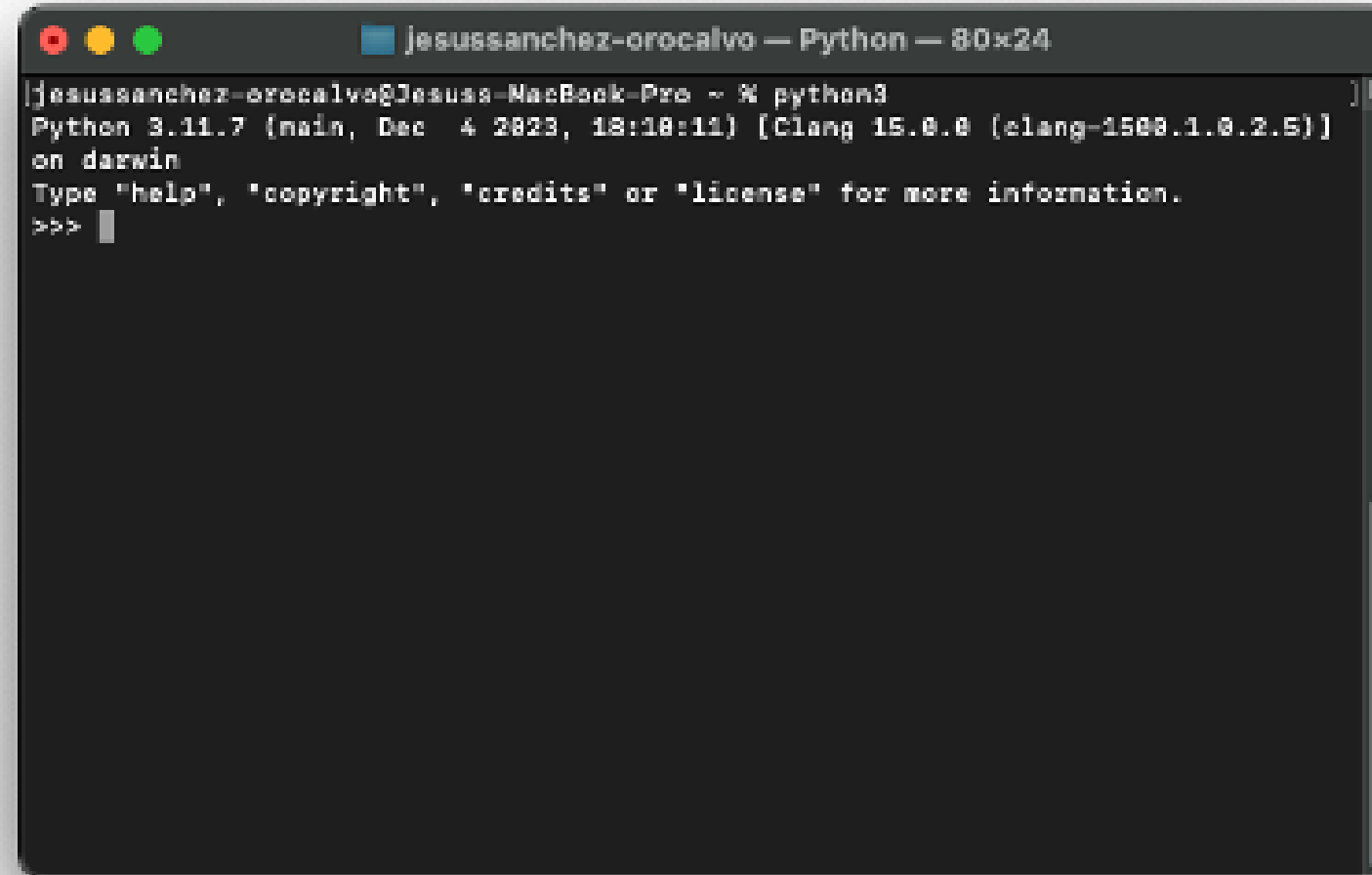
- Nada más, ni imports, ni creación de un main, ni una clase, ni nada similar, solo imprimir la cadena por pantalla



Todo ha ido bien

¿Dónde Programo?

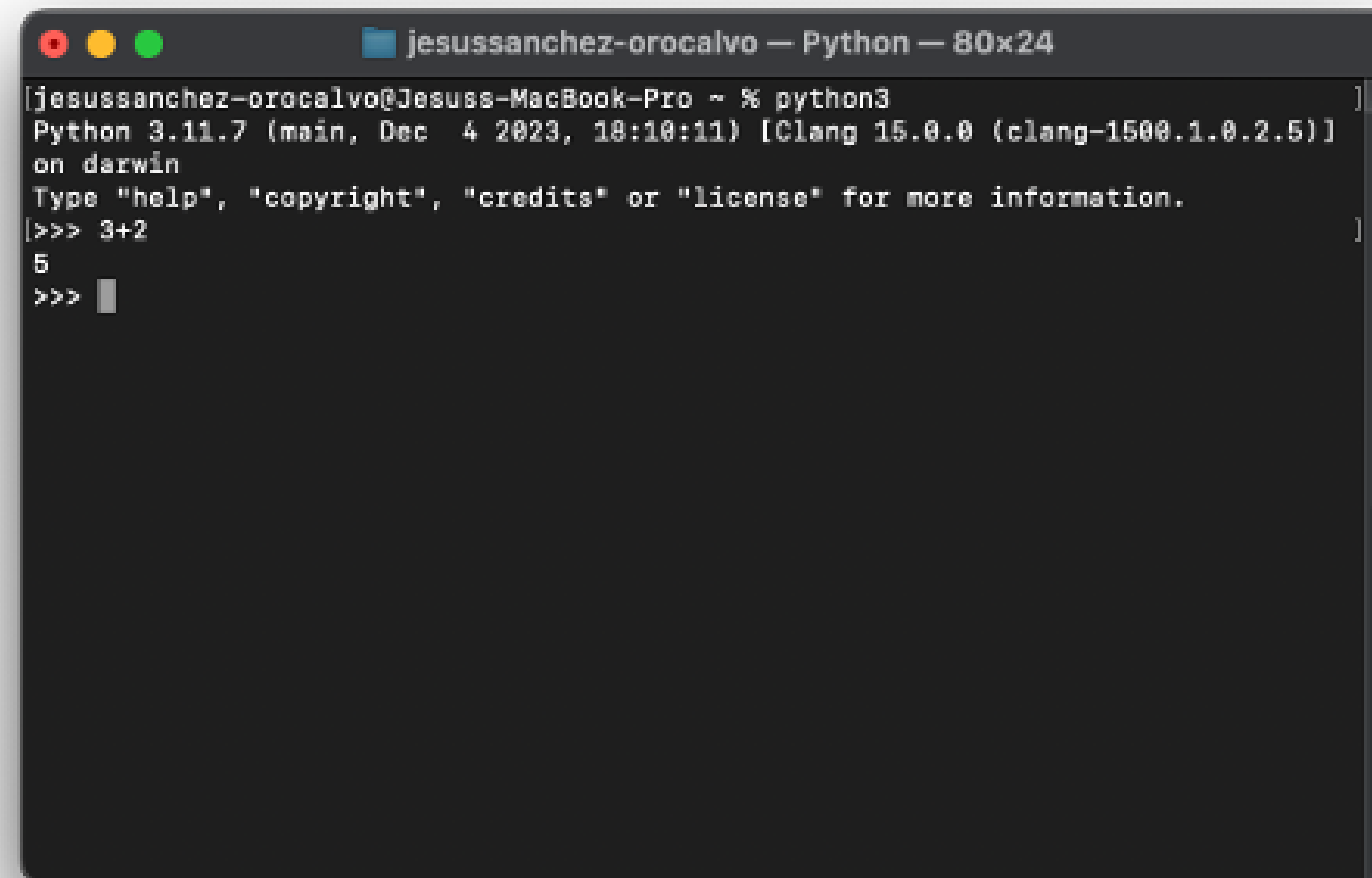
- Directamente en el terminal, escribimos python3 para arrancar el *prompt*



```
jesussanchez-orocalvo — Python — 80x24
jesussanchez-orocalvo@Jesuss-MacBook-Pro ~ % python3
Python 3.11.7 (main, Dec  4 2023, 18:18:11) [Clang 15.0.0 (clang-1500.1.0.2.5)]
on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>> 
```

¿Dónde Programo?

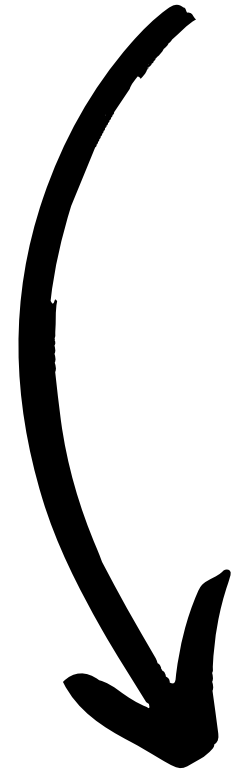
- Ahí ya podemos empezar a escribir instrucciones, operaciones matemáticas, etc.



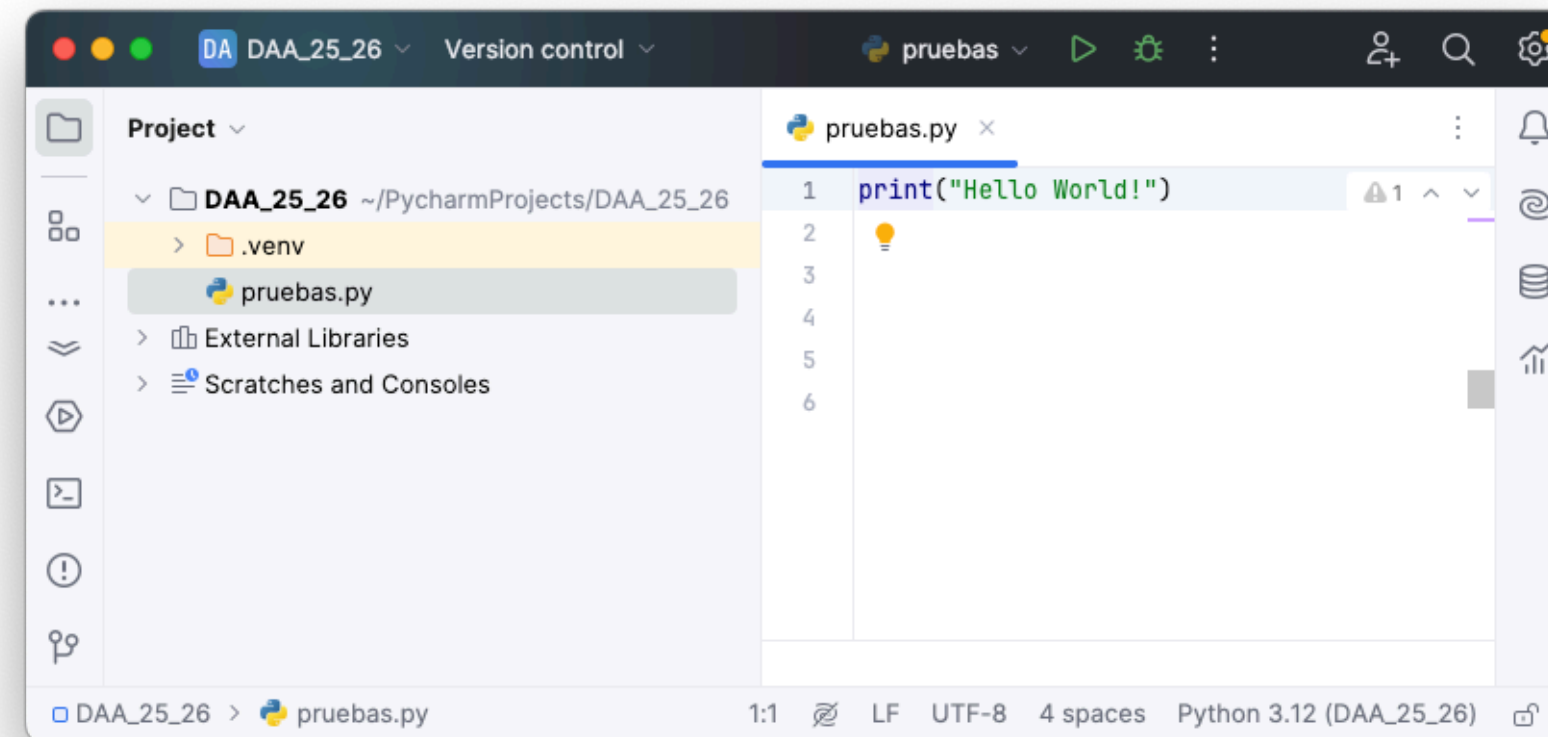
```
jesussanchez-orocalvo@Jesuss-MacBook-Pro ~ % python3
Python 3.11.7 (main, Dec  4 2023, 18:10:11) [Clang 15.0.0 (clang-1500.1.0.2.5)]
on darwin
Type "help", "copyright", "credits" or "license" for more information.
[>>> 3+2
5
>>> ]
```

Pycharm

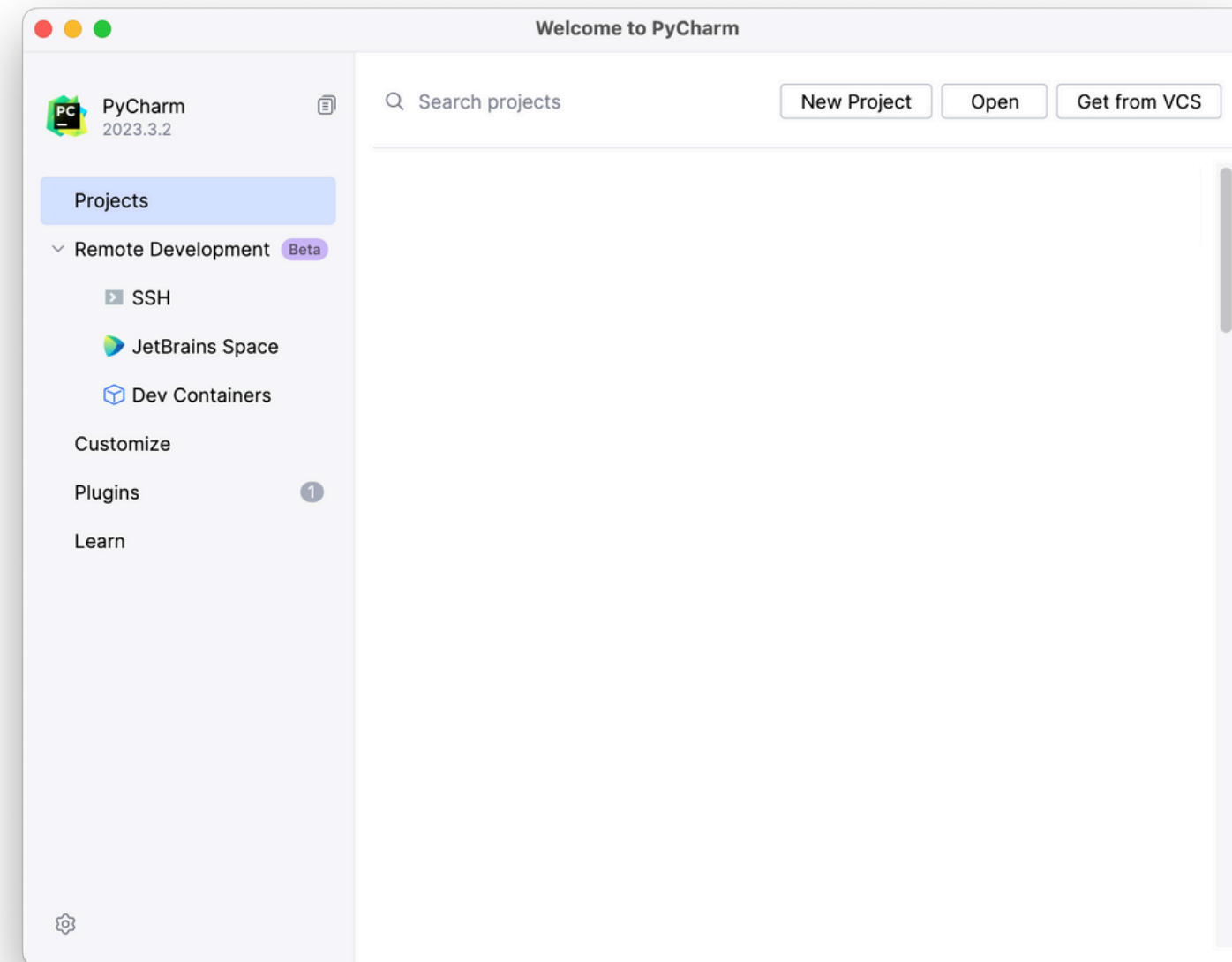
- IDE desarrollado por JetBrains, disponible de forma gratuita con la cuenta de alumno URJC.



Integrated Development Environment

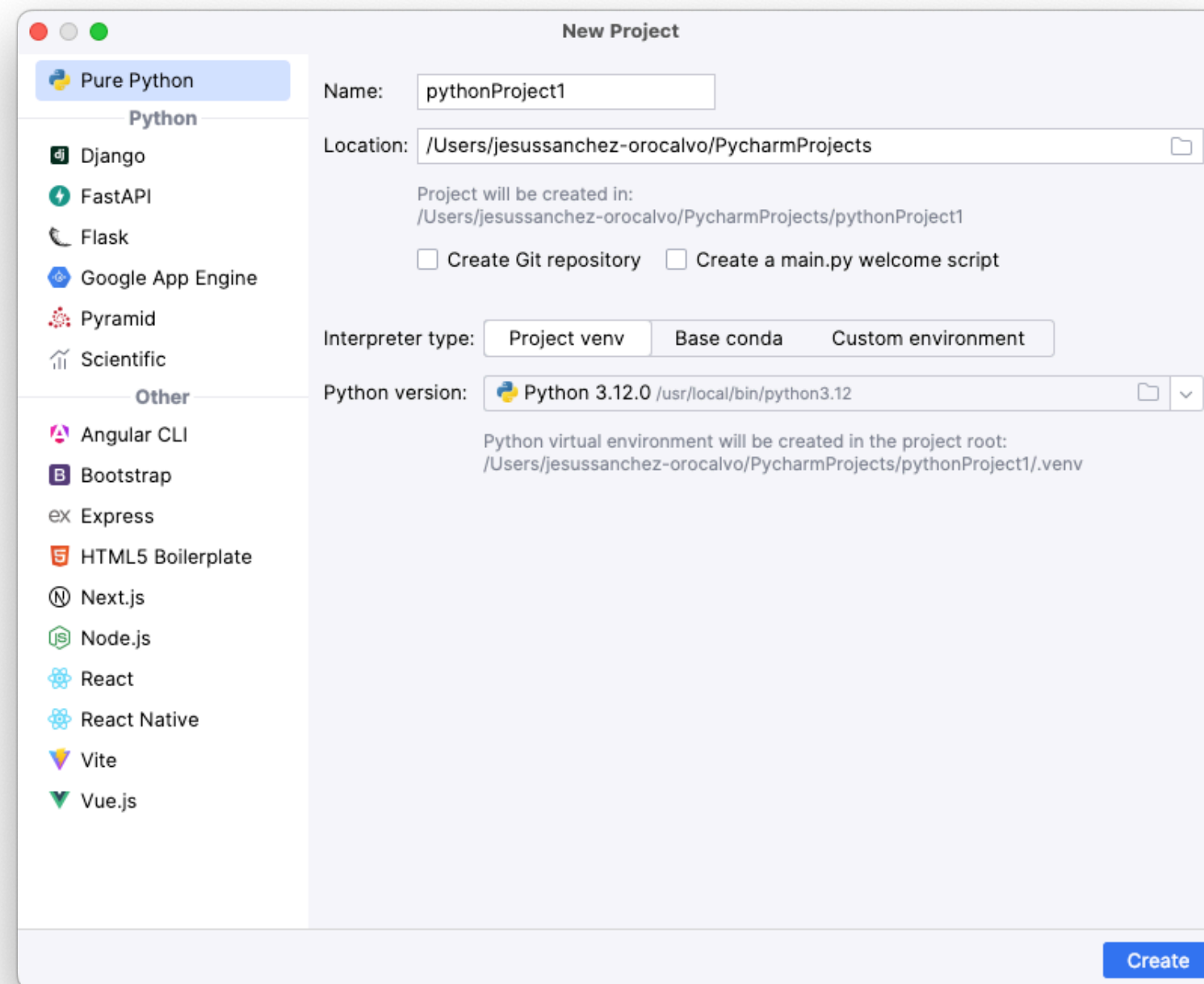


Pycharm - Nuevo proyecto



Pycharm - Nuevo proyecto

- Última versión de Python y el resto de opciones por defecto



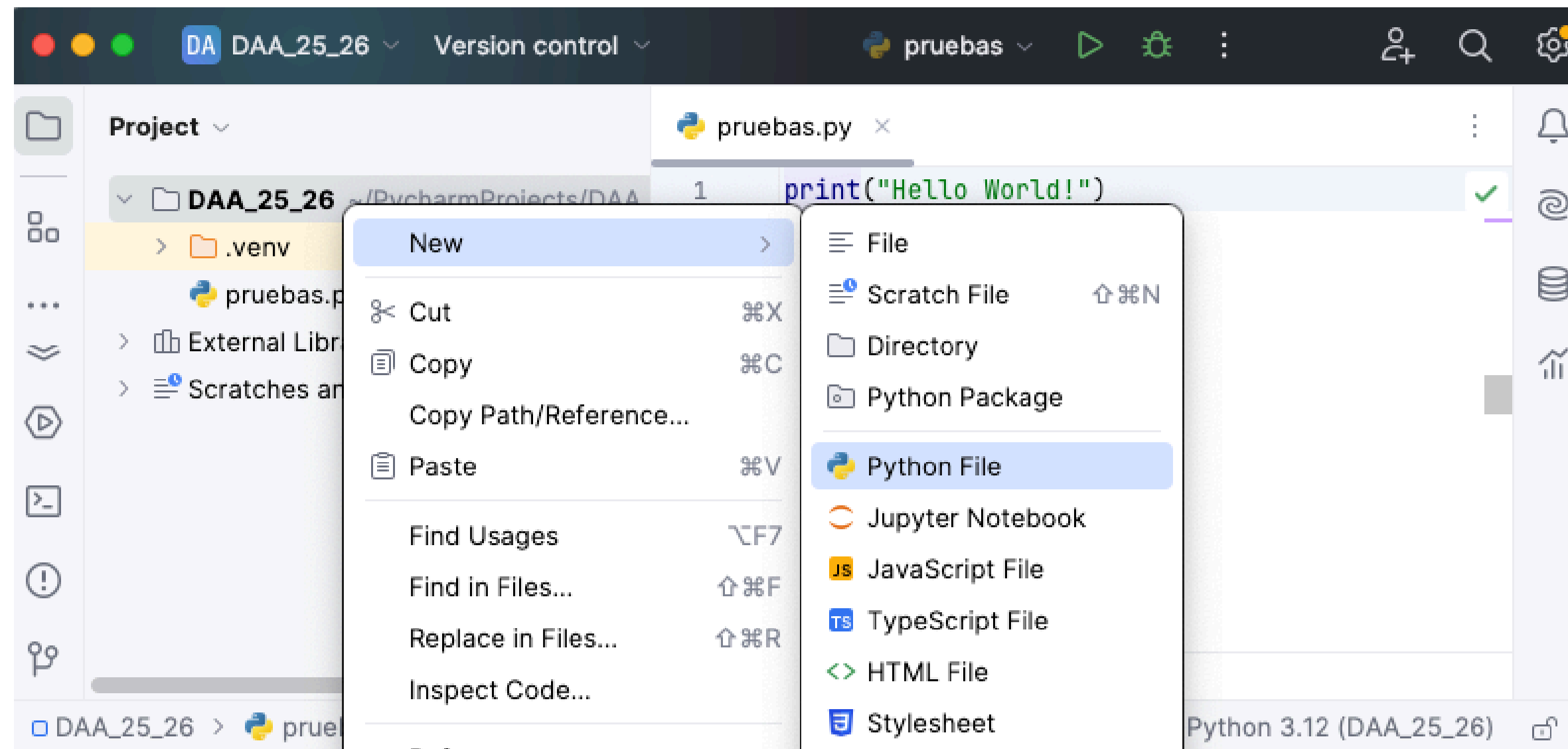
—————→ Nombre del proyecto que sea descriptivo

—————→ Directorio donde se creará el proyecto

—————→ Entorno virtual

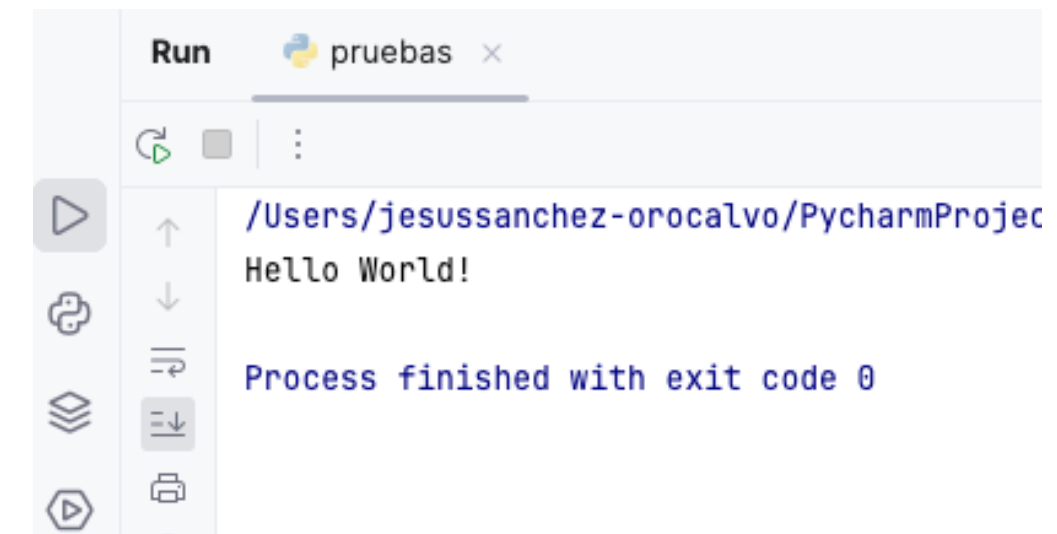
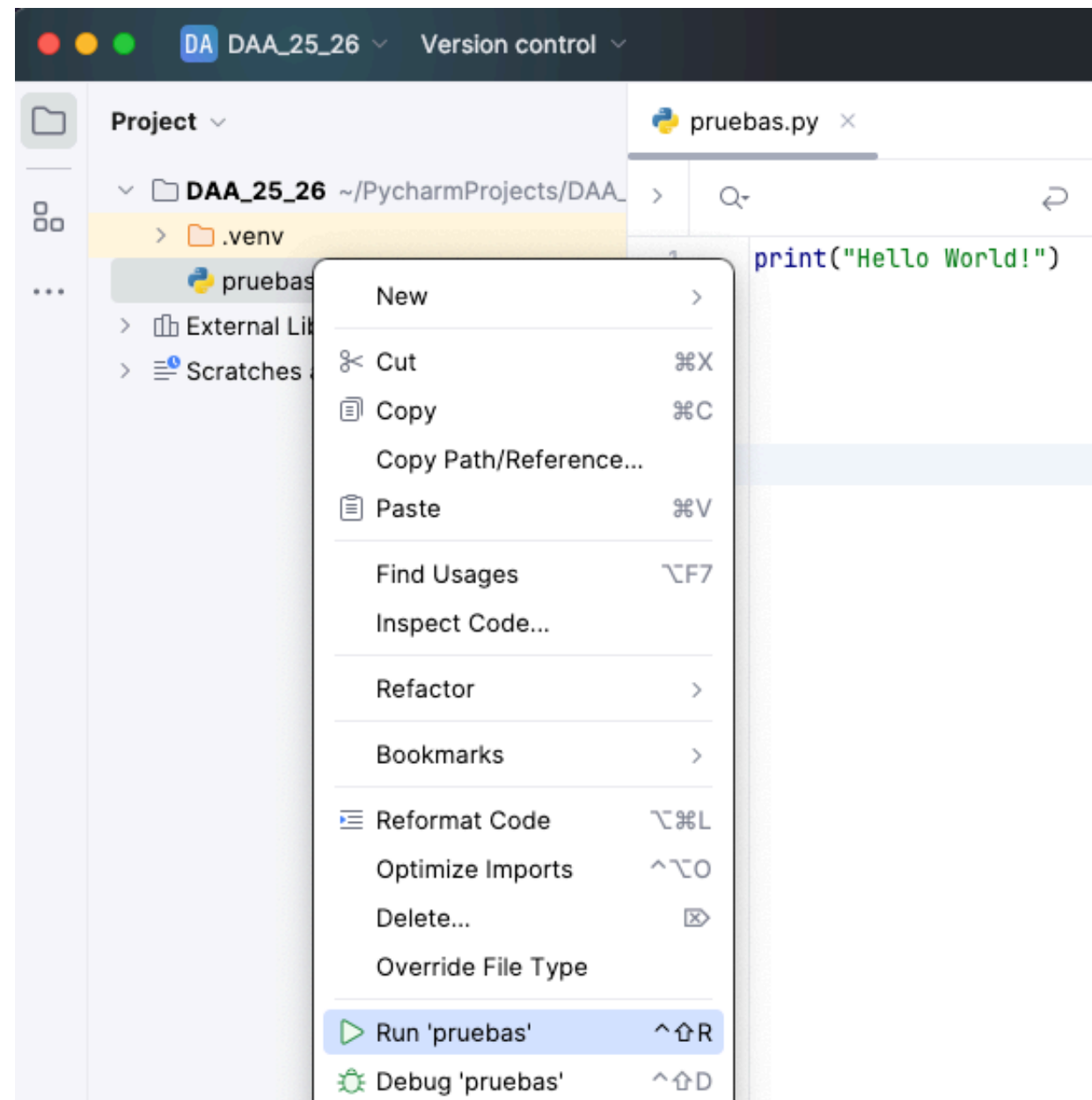
Pycharm - Nuevo fichero

- Click derecho en el proyecto, y elegimos nuevo fichero Python



Pycharm - Ejecutar script

- Click derecho en el fichero → *Run nombre_fichero*



Operadores y expresiones

Operación	Operador	Aridad	Asociatividad	Preced.
Exponenciación	**	Binario	Por la dcha	1
Identidad, cambio de signo	+, −	Unario	−	2
Multip, div	*, /	Binario	Por la izq	3
Div ent, mód	//, %	Binario	Por la izq	3
Suma, resta	+, −	Binario	Por la izq	4
Distinto, igual que	!=, ==	Binario	−	5
Menor, menor o igual que	<, <=	Binario	−	5
Mayor, mayor o igual que	>, >=	Binario	−	5
Negación	not	Unario	−	6
Conjunción	and	Binario	Por la izq	7
Disyunción	or	Binario	Por la izq	8

Tipos de datos

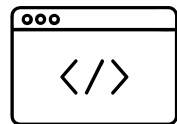
- Enteros
 - Ocupan menos memoria
 - Más rápidos para operar que los reales
- Reales
 - Diferentes notaciones: 3.2E-3, .01, 2.
- Booleano
 - Operadores and, or y not

Operadores de relación

operador	comparación
!=	distinto que
==	igual que
<	es menor que
<=	es menor o igual que
>	es mayor que
>=	es mayor o igual que

Cadenas de caracteres

- Se escriben entre comillas simples o dobles
- Operadores
 - Concatenación: +
 - Repetición: *

```
 1 name = "Jesus"  
2 print("Hello World " + name + "!!")  
3 print(("Hello World " + name + "!!")*4)
```

Identificadores

- Los nombres de variables, funciones, etc. deben cumplir ciertas normas:
 - Combinación de letras, dígitos y/o el carácter subrayado '_'
 - El primer carácter no puede ser un dígito
 - No puede coincidir con una palabra reservada del lenguaje
 - Distingue mayúsculas y minúsculas

 area

 altura_persona

 3pekeño


 for

Funciones predefinidas

- **abs()**: valor absoluto
- **float()**: valor real de un entero o cadena de caracteres
- **int()**: valor entero de un real o cadena de caracteres
- **str()**: cadena de caracteres a partir de un entero o real
- **round()**: uno o dos argumentos
 - Con 1 redondea al entero más cercano
 - Con 2 indicamos cuántos decimales queremos conservar

Módulos

- Python dispone de módulos que amplían su utilidad
- Para poder utilizarlos, debemos importarlos:



```
1 # Opción 1: importar el módulo completo
2 import math
3
4 # Opción 2: importar las funciones que necesitamos
5 from math import sin
6 from math import *
```

Módulos

- No se recomienda importar el módulo completo
- Si importamos elemento a elemento, mantenemos su procedencia, mejorando la legibilidad de nuestro código
- Si no, podríamos definir una variable con el nombre de una función y crear conflictos

```
1 from math import *  
2  
3 sin(90)
```

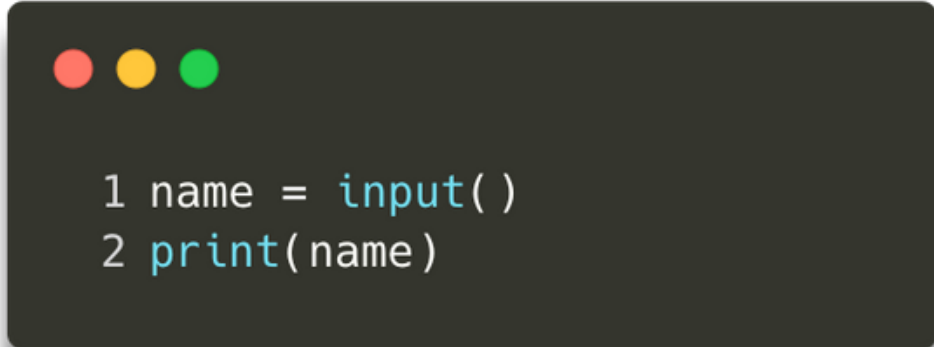


```
1 import math  
2  
3 math.sin(90)
```



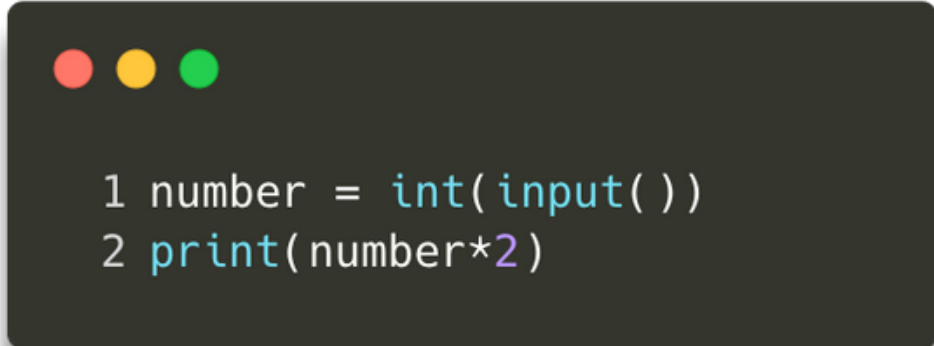
Entrada / Salida

- La sentencia `input()` recibe datos por teclado como una cadena de caracteres
- La sentencia `print()` imprime caracteres por consola



```
1 name = input()  
2 print(name)
```

- Si necesitamos que sea otro tipo de datos, podemos convertirlo



```
1 number = int(input())  
2 print(number*2)
```

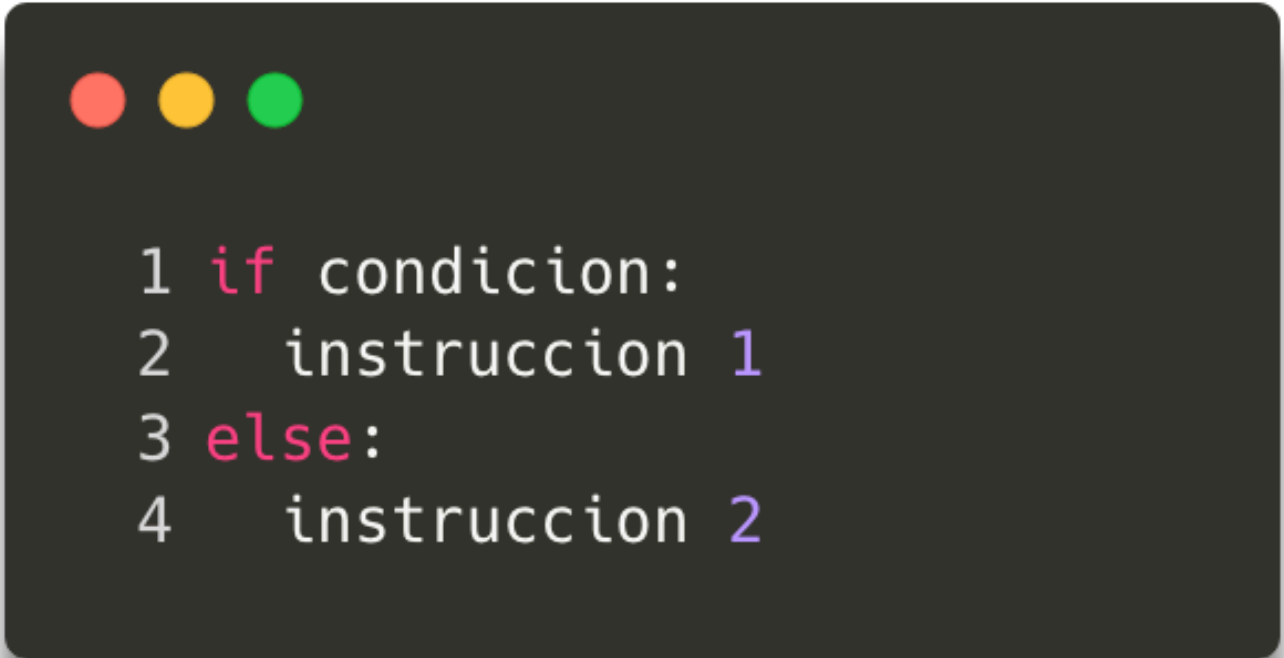
Sentencia condicional - if

```
1 if condicion:  
2     instruccion 1  
3     instruccion 2  
4     ...  
5     instruccion N
```



Importante: ¡¡¡hay que respetar el indentado!!!

Sentencia condicional - if-else



```
1 if condicion:  
2     instruccion 1  
3 else:  
4     instruccion 2
```

- Evaluación **perezosa**: si con la primera parte de la expresión sabemos el resultado, no evaluamos el resto

Sentencia condicional - if-else-if

```
1 if condicion:  
2     instruccion 1  
3 elif condicion 2:  
4     instruccion 2  
5 else:  
6     instruccion 3
```

- Si hay más de una condición, podemos utilizar más de un elif

Sentencia iterativa - for

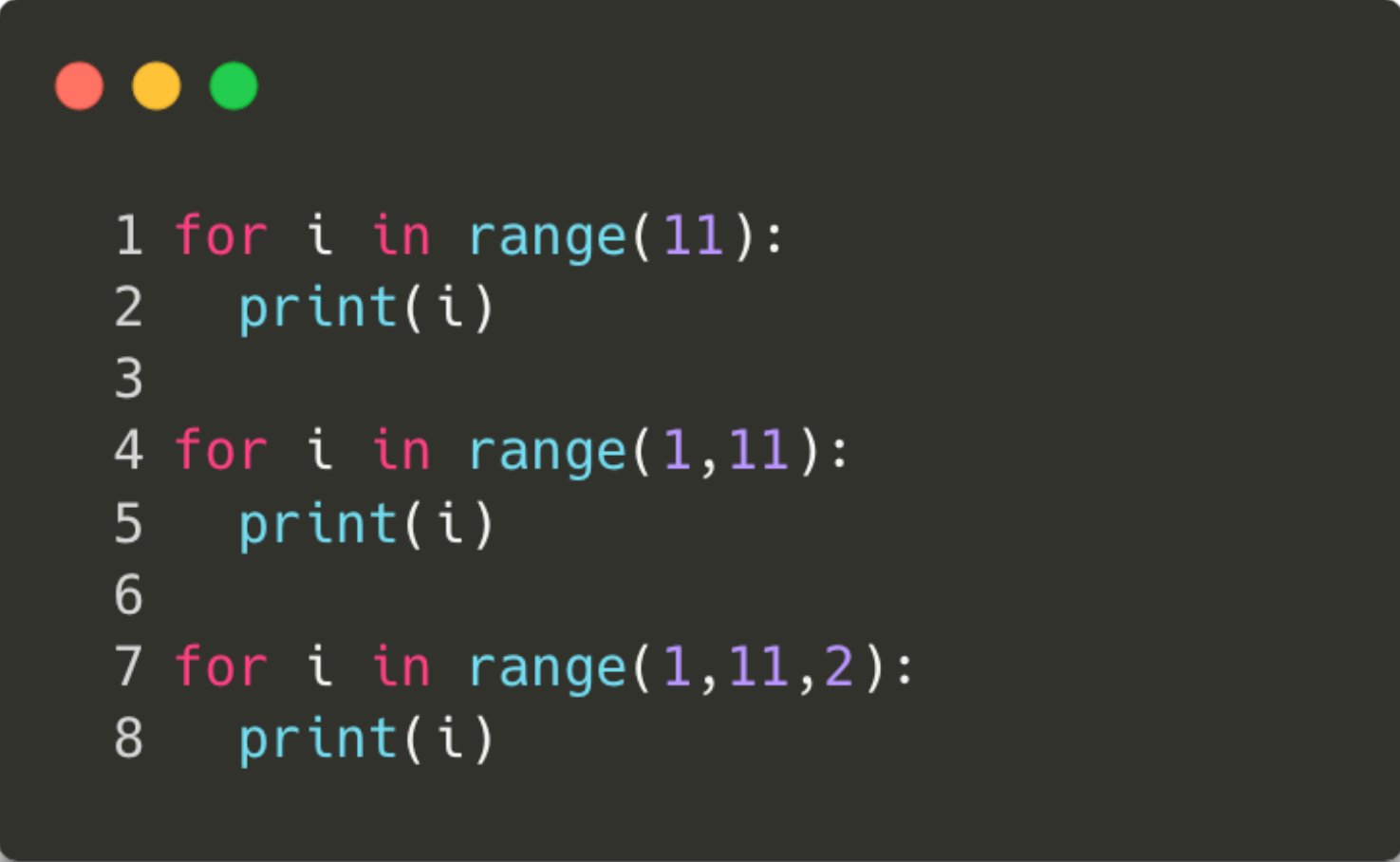
```
1 for variable in serie_de_valores:  
2     instruccion_1  
3     instruccion_2  
4     ...  
5     instruccion_N
```

- Para cada elemento de la serie de valores, se ejecuta el código

Función range

- A veces no podemos escribir el rango de valores
- range es una función que, dado un inicio y un final, genera el rango de valores intermedio
 - El inicio es inclusivo y el final exclusivo
- Si la utilizamos con un argumento, solo especificamos el final
 - El inicio será 0
- Si la utilizamos con 3 argumentos , el tercero indica el incremento

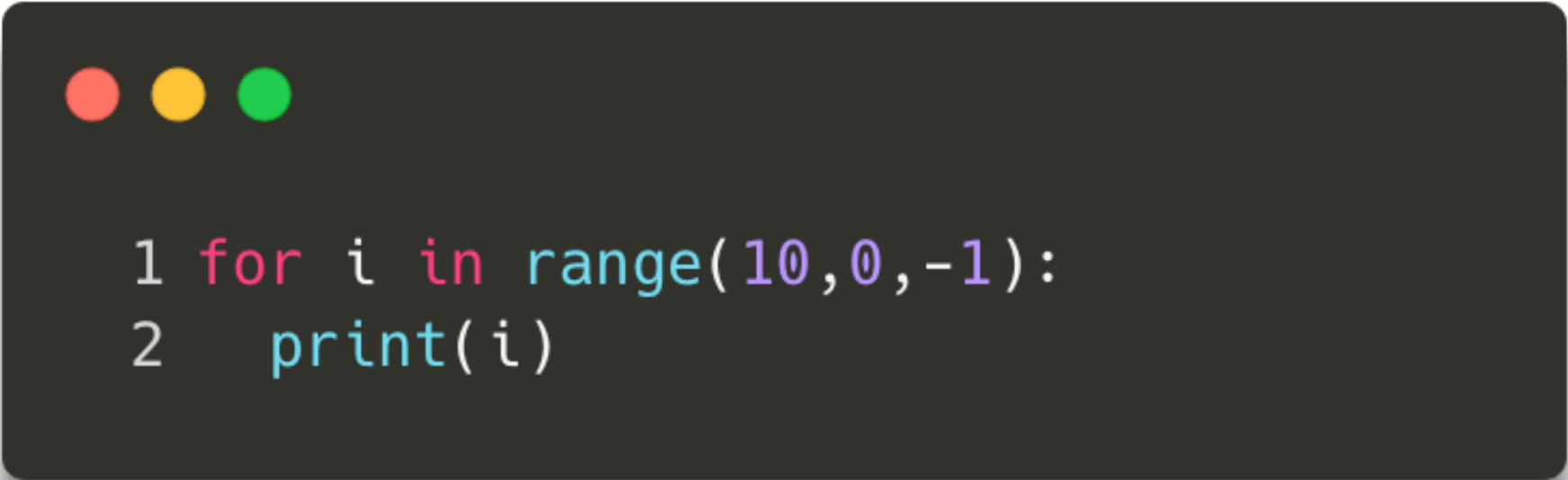
Función range



```
1 for i in range(11):  
2     print(i)  
3  
4 for i in range(1,11):  
5     print(i)  
6  
7 for i in range(1,11,2):  
8     print(i)
```

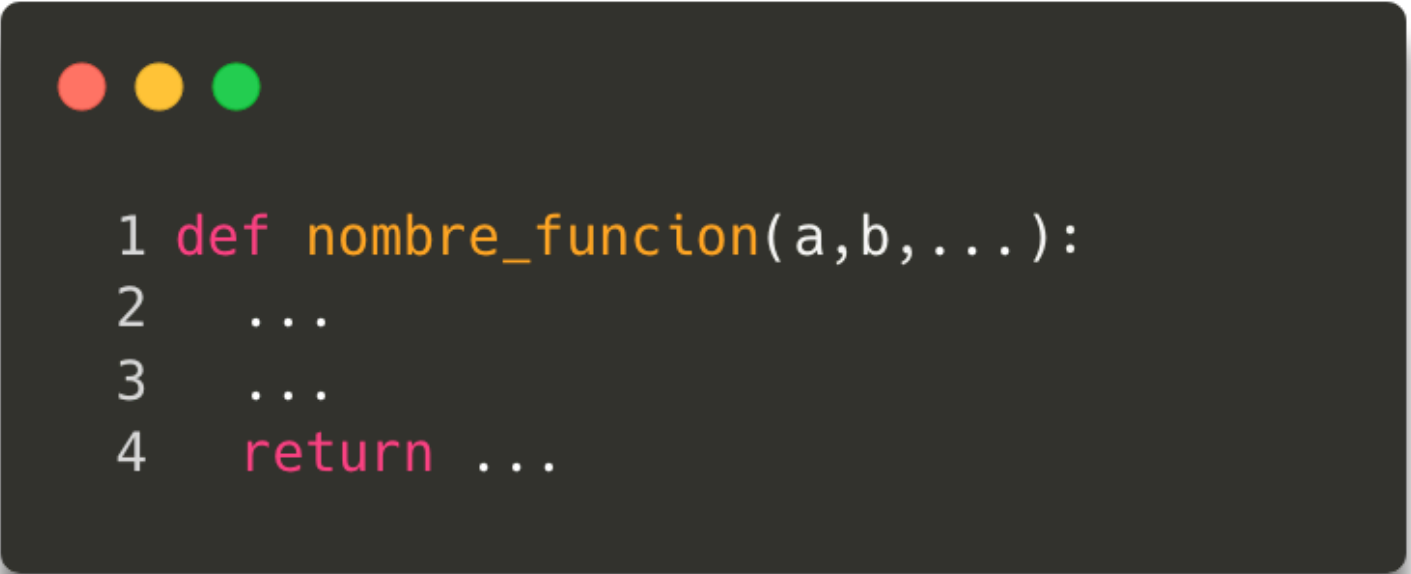
Función range

- ¿Y si queremos una serie decreciente?



```
1 for i in range(10,0,-1):  
2     print(i)
```

Definición de funciones



```
1 def nombre_funcion(a,b,...):  
2     ...  
3     ...  
4     return ...
```

- def indica que vamos a definir una función
- return indica lo que devuelve la función
- Si no ponemos return, tendremos un procedimiento

Listas

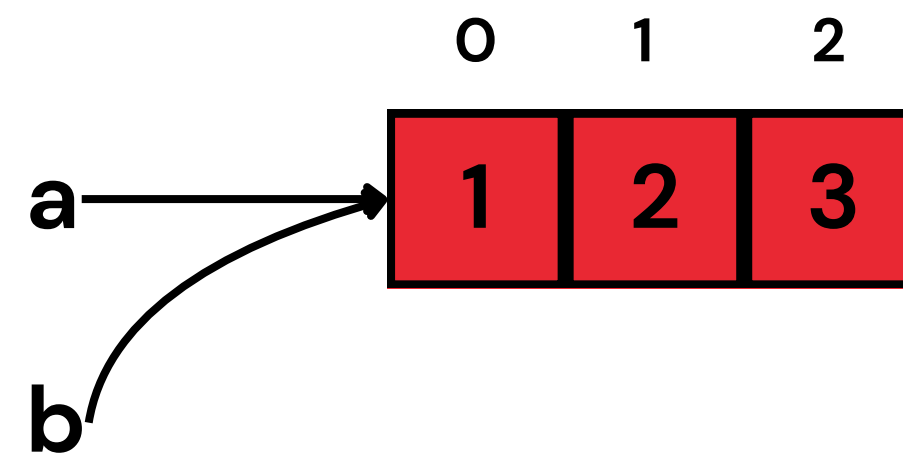


```
l = [1,2,3]
```

- Podemos definir listas de cualquier tipo de datos
- Los valores se escriben entre corchetes y separados por comas

Listas

```
1 a = [1,2,3]
2 b = a
```



Las listas son punteros

Listas

- Longitud de una lista → `len()`
- Concatenar listas → operador `+`
- Repetir una lista → operador `*`
- Obtener un fragmento → operador corte `[:]`
- Añadir elementos → concatenar o método `append`
- Eliminar elementos → operador `del`
- Saber si una lista contiene un elemento → operador `in`

Matrices

- Las matrices se almacenan como listas de listas

```
1 m = [[1,2,3], [4,5,6], [7,8,9]]  
2 m[0][1]
```

- El acceso se hace de forma análoga

Diccionarios

- Ofrecen una correspondencia entre pares clave-valor
- Se crean con el operador { }
- Se añaden de forma similar a las listas, pero indexando a través de la clave

```
1 d = {}  
2 d['uno'] = 1  
3 d['dos'] = 2
```

TEMA 0 - INTRODUCCIÓN A PYTHON

Algoritmos

