

# PROGRAMACIÓN DE INTELIGENCIA ARTIFICIAL

## PYTHON

Las Funciones Integradas están disponibles en el intérprete de Python.

**REUTILIZACIÓN DE CÓDIGO**  
*Biblioteca estándar Python =  
Funciones Integradas + Módulos*

Un **MÓDULO** es un fichero . py que incluye un conjunto de funciones, variables o clases y que puede ser usado por otros módulos.

```
>>> import math
>>> pi=3.14
>>> print(pi, math.pi)
3.14 3.141592653589793
```

- Se importan **todas** las entidades del módulo
- Se accede a las entidades **junto** con el nombre del módulo
- **Coexisten** con entidades del mismo nombre en el código

```
>>> from math import pi, sin
>>> print(pi)
3.141592653589793
>>> pi = 3.14
>>> print(pi)
3.14
```

- Se importan **determinadas** entidades del módulo (con \* se importan todas )
- Se accede a las entidades **unicamente** por su nombre
- Se **sobreescriben** entidades del mismo nombre, solo es accesible la última declarada

```
import random as r
r.seed(0)
pi="Hola"
e = "Soy una e"
from math import pi as PI, e as E
print (pi, PI, e, E, sep='-')
Hola-3.141592653589793-Soy una e-2.718281828
```

### **ALIAS (as)**

- Permiten **simplificar** el código
- Permiten **renombrar** entidades para evitar sobreescrituras

## PROGRAMACIÓN DE INTELIGENCIA ARTIFICIAL

## PYTHON

Módulo **math**

## Funciones matemáticas

```
>>> import math  
>>> dir(math)
```

- ⇒ **math.sin(x)**, **math.cos(x)**, **math.tan(x)**:  
seno, coseno o tangente de x
- ⇒ **math.asin(x)**, **math.acos(x)**, **math.atan(x)**:  
arcoseno, arcocoseno o arcotangente de x
- ⇒ **math.sinh(x)**, **math.cosh(x)**, **math.tanh(x)**:  
seno, coseno o tangente hiperbólicos de x
- ⇒ **math.asinh(x)**, **math.acosh(x)**, **math.atanh(x)**:  
arcoseno, arcocoseno o arcotangente  
hiperbólicos de x
- ⇒ **math.radians(x)**:  
convierte x grados a radianes
- ⇒ **math.degrees(x)**:  
convierte x radianes a grados
- ⇒ **math.pi**: constante numérica pi

- ⇒ **math.e**: constante numérica de Euler, e
- ⇒ **math.exp(x)**: devuelve  $e^x$
- ⇒ **math.log(x)**: logaritmo neperiano de x
- ⇒ **math.log(x, b)**: logaritmo en base b de x
- ⇒ **math.log10(x)**: logaritmo decimal de x
- ⇒ **math.log2(x)**: logaritmo binario de x
- ⇒ **pow(x,y)**: devuelve  $x^y$  (*no necesita importarse*)
  
- ⇒ **math.ceil(x)**: devuelve el entero más pequeño  $\geq x$
- ⇒ **math.floor(x)**: devuelve el mayor entero  $\leq x$
- ⇒ **math.trunc(x)**: devuelve el valor de x truncado a entero
  
- ⇒ **math.factorial(x)**: devuelve el valor del factorial de x
- ⇒ **math.hypot(x,y)**: devuelve el valor de la hipotenusa del triángulo de lados x e y -  $\sqrt{\text{pow}(x,2) + \text{pow}(y,2)}$

## PROGRAMACIÓN DE INTELIGENCIA ARTIFICIAL

# PYTHON

Módulo **random**  
Funciones pseudoaleatorias

```
>>> import random  
>>> dir(random)
```

- ⇒ **random.random():**  
devuelve un número flotante aleatorio en el intervalo [0 ,1]
- ⇒ **random.randrange(f):**  
devuelve un número entero aleatorio en el intervalo [0 ,f)
- ⇒ **random.randrange(i,f):**  
devuelve un número entero aleatorio en el intervalo [i ,f)
- ⇒ **random.randrange(i,f, inc):**  
devuelve un número entero aleatorio en el intervalo [i ,f) a incrementos inc
- ⇒ **random.randint(i, f):**  
devuelve un número entero aleatorio en el intervalo [i ,f]

- ⇒ **random.choice(secuencia):**  
devuelve un elemento de la secuencia
- ⇒ **random.sample(secuencia, num):**  
devuelve num elementos de la secuencia
- ⇒ **random.seed():**  
reinicializa la semilla del algoritmo con elementos de la hora actual
- ⇒ **random.seed(entero):**  
reinicializa la semilla del algoritmo con el número pasado como argumento

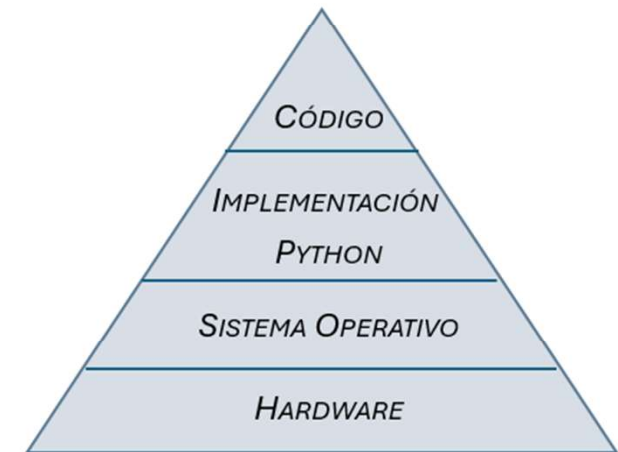
# PROGRAMACIÓN DE INTELIGENCIA ARTIFICIAL

## PYTHON

Módulo **platform**  
Funciones entorno ejecución

```
>>> import platform  
>>> dir(platform)
```

- ⇒ **platform.machine():** AMD64  
devuelve una cadena con el nombre genérico del procesador
- ⇒ **platform.processor():** Intel64 Family 6 Model 140 Stepping 1, GenuineIntel  
devuelve una cadena con el nombre real del procesador
- ⇒ **platform.system():** Windowsl  
devuelve una cadena con el nombre genérico del SO
- ⇒ **platform.version():** 10.0.22631  
devuelve una cadena con la versión del SO
- ⇒ **platform.python\_implementation():** CPython  
devuelve una cadena con la implementación Python
- ⇒ **platform.python\_version\_tuple():** ('3', '12', '6')  
devuelve una tupla con la versión, parte menor y parche de Python instalado

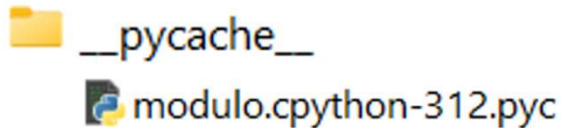


**platform.platform(aliased=True, terse=True):**  
aliased = true -> devuelve nombres alternativos más comprensibles  
terse = true -> devuelve el mínimo de información necesaria

## PROGRAMACIÓN DE INTELIGENCIA ARTIFICIAL

## PYTHON

Al importar un módulo no incluido en la biblioteca estándar, **se crea una carpeta** que contiene el código compilado del módulo.



**modulo:** nombre del módulo importado  
**cpython:** implementación que ha creado modulo.py  
**312:** números derivados de la versión utilizada  
**.pyc** extensión de Python Compilado

La variable **path** del módulo **sys** almacena una **lista** de ubicaciones en las que se buscará un módulo solicitado por una instrucción **import**. Si se necesita añadir una ruta nueva se agrega a la variable **path**.

```
from sys import path
```

```
path.append('../\\modules')
```

**ATENCIÓN.** Hay que duplicar el carácter \

```
#!/usr/bin/env python3
# Comentario obligatorio en Unix o SO similares
# Indica el programa que ejecutará el archivo

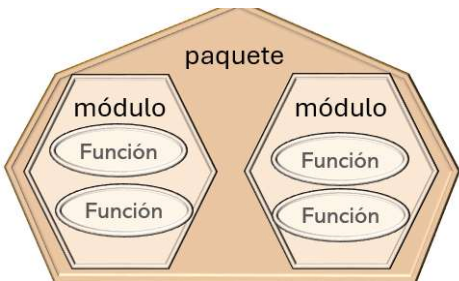
''' module.py - ejemplo de modulo en Python '''
# doc-string
# Explica proposito y contenido del módulo

__counter = 0
# acuerdo tácito
# informa a usuarios del módulo que no modifiquen la variable
```

*#! se conoce como shebang, shabang, ...*

- Cuando un módulo es importado por primera vez se ejecuta implícitamente. Así inicializa aspectos internos del módulo [Código fuera de cualquier función]
- No se vuelve a inicializar mientras el módulo no se modifique

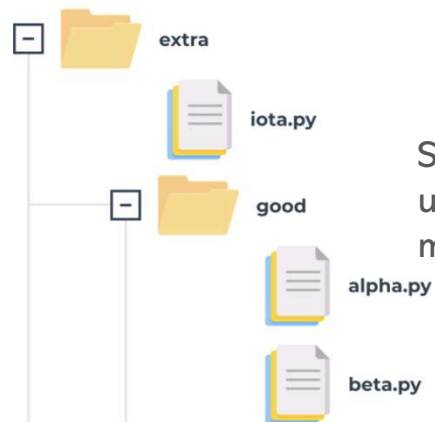
`__name__ = '__main__'` ⇒ El archivo se ejecuta directamente  
`__name__ = 'archivo'` ⇒ El archivo se ha importado como módulo desde archivo.py



Un **paquete** es una agrupación de módulos. Los módulos no necesariamente tienen que estar en el mismo nivel, sino que se puede crear una jerarquía de módulos en subniveles

Para señalar que *extra* es un paquete, en su carpeta debe estar presente un archivo `__init.py__` que contiene la lógica necesaria para la inicialización del paquete (si se necesita), y los **nombres de los módulos** del paquete.

Su contenido se ejecuta cuando se importa cualquiera de los módulos que componen el paquete



Si *extra* es el nombre del paquete, la ubicación de la función *a()* incluida en el módulo *alpha.py* se describe:

*extra.good.alpha.a()*

```
import extra.iota
import extra.good.beta
```

```
from extra.iota import i
from extra.good.beta import b
```

```
x = extra.iota.i()
y = extra.good.beta.b()
```

```
from sys import path

path.append('..\\packages\\extrapack.zip')

import extra.good.alpha as alp
import extra.good.beta as bet

print(alp.a())
print(bet.b())
```

La carpeta *extra* y todo su contenido puede comprimirse en un archivo **.zip**  
Se modifica *sys.path* para que contenga el archivo **.zip**





Un grupo de trabajo dentro de PSF (Python Software Foundation) se encarga de mantener el repositorio **PyPI** (Python Package Index). Su objetivo es apoyar a los desarrolladores de Python en la diseminación de código eficiente.

**pip** es la herramienta para poder descargar recursos de PyPI

El comando **pip --version** (**pip3** en algunos entornos) te permite comprobar si dispones de la herramienta. En entornos Windows la ausencia de este mensaje puede significar que la variable PATH apunta incorrectamente a la ubicación de los binarios de Python o no apunta a ellos en absoluto

```
C:\>pip --version
pip 24.2 from C:\Users\asun\AppData\Local\Programs\Python\Python312\Lib\site-packages\pip (python 3.12)
```

**C:\>pip help**

muestra todas los comandos disponibles

**C:\>pip help comando**

muestra una breve descripción comando

**C:\>pip list**

muestra paquetes actualmente instalados

**C:\>pip show nombre-paquete**

muestra información del paquete (dependencias)

**C:\>pip search cadena**

busca en los directorios de PyPI paquetes cuyo nombre contenga cadena

**C:\>pip install nombre**

instala el paquete nombre en todo el sistema

**C:\>pip install --user nombre**

instala el paquete nombre solo para ese usuario

**C:\>pip install -U nombre**

actualiza el paquete

**C:\>pip uninstall nombre**

desinstala el paquete nombre