Universidad Politécnica de Cartagena

Grado en Arquitectura Naval e
Ingeniería de Sistemas Marinos

INFORMÁTICA

Apuntes de teoría, ejemplos prácticos y
ejercicios resueltos para ingeniería.

Autor: David Conesa Pagán

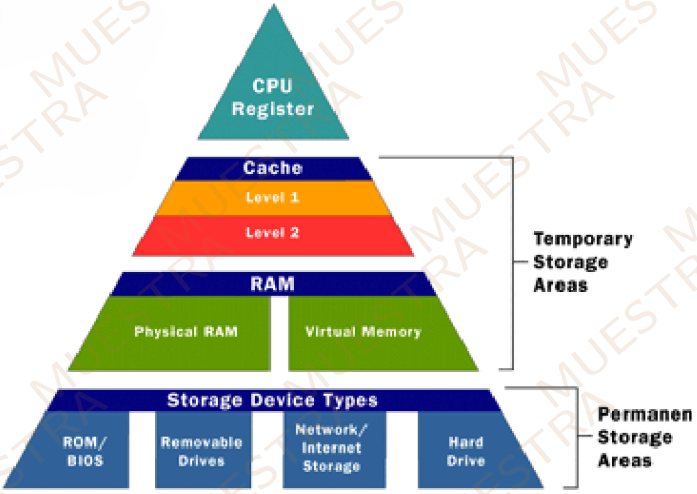
Curso Académico 2025 / 2026
1.ª Edición (Revisada)

ESTE EJEMPLAR PERTENECE AL ALUMNO/A:

.....

1.3.4. Distribución de la información en la jerarquía de memoria

- La CPU accede a la memoria siguiendo la **jerarquía de memoria**: registros → caché L1, L2, L3 → RAM → almacenamiento secundario.



- Si la información que busca la CPU estuviera siempre en caché, el sistema alcanzaría su máxima velocidad.
- El **Principio de Localidad** determina qué información debe ubicarse en cada nivel:
  - **Localidad espacial:** la información cercana a la que se acaba de utilizar tiene alta probabilidad de ser usada.
  - **Localidad temporal:** la información utilizada recientemente tiene alta probabilidad de volver a usarse pronto.

Analogía: La jerarquía de memoria funciona como un escritorio y un archivador: lo que estás usando ahora está en tu mano (caché), lo que usarás pronto está cerca del escritorio (RAM), y lo que no necesitas aún está en el archivo o almacenamiento secundario (disco o SSD).

1.4. Gestión de la entrada/salida

La **gestión de E/S** se encarga de coordinar y controlar todos los dispositivos periféricos del sistema, garantizando eficiencia y abstracción frente a los procesos.

Problemas de los dispositivos de E/S:

- Diferencia de velocidad entre CPU y periférico: la CPU es mucho más rápida que los dispositivos de E/S.
- Funcionamiento y características dispares entre periféricos: cada dispositivo tiene su propia forma de operar.
- Sincronización entre procesos concurrentes que acceden al mismo dispositivo.
- Compartición de recursos: varios procesos pueden necesitar acceso a un mismo dispositivo.

Objetivos de la gestión de E/S:

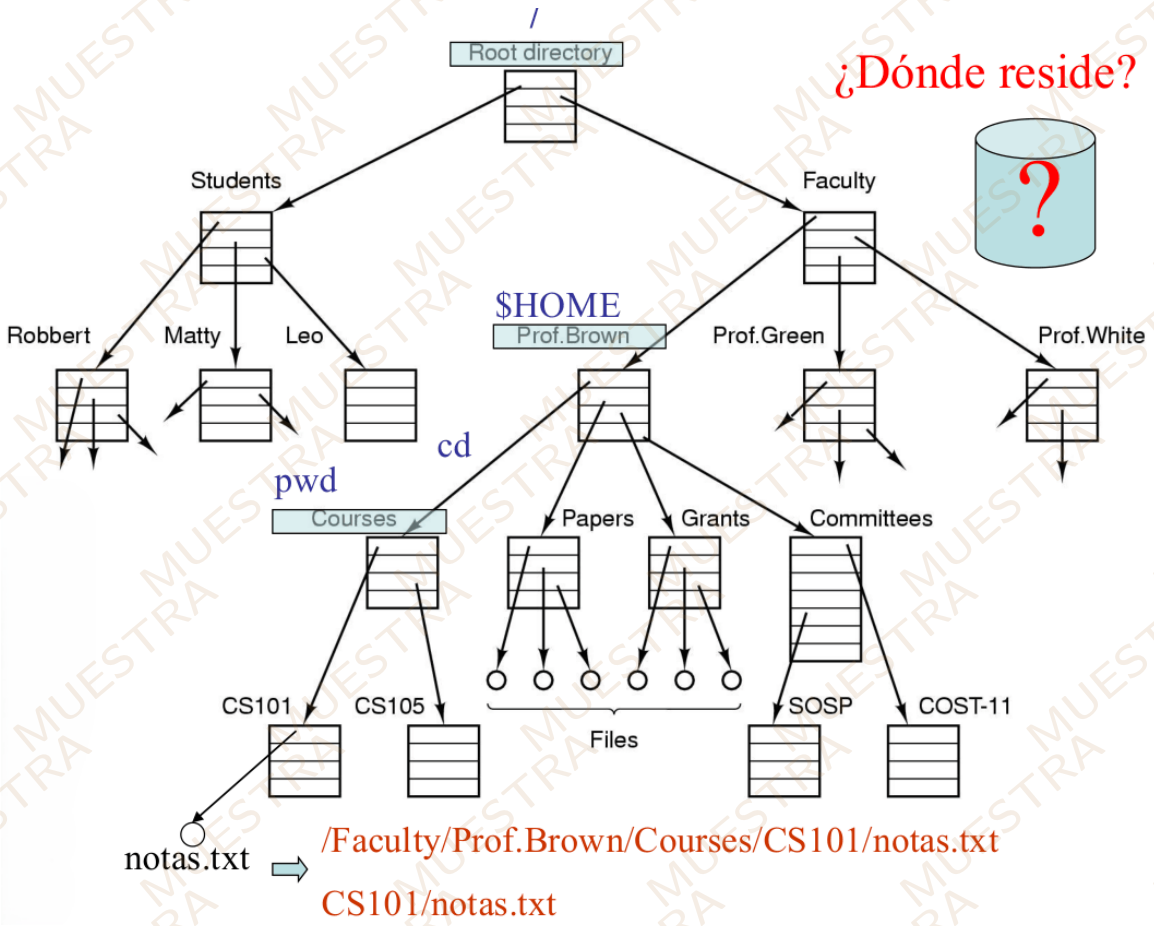
- Lograr que los periféricos se utilicen con eficiencia.
- Proporcionar una capa de abstracción que haga independiente a los procesos de los periféricos concretos.

Sistema de archivos:

- Define cómo se guarda la información en el dispositivo de almacenamiento:
  - Tamaño mínimo y máximo de archivos.
  - Control de acceso.
  - Integridad de los datos.
- Incluye mecanismos para:
  - Evitar fragmentación.
  - Asegurar la integridad de los datos.
  - Optimizar el acceso y la recuperación de información.
- Sistemas de archivos más habituales: FAT, FAT32, NTFS, EXT3, HFS, entre otros.

Analogía: Se puede comparar con una biblioteca:

- Los **archivos** son los libros, cada uno con título, autor, tamaño y lugar físico en la estantería.
- Los **directorios** son las estanterías que organizan los libros por categorías.
- El **sistema de archivos** es el bibliotecario que establece las reglas para almacenar los libros, evitar pérdidas y asegurar que los lectores encuentren rápidamente lo que buscan.



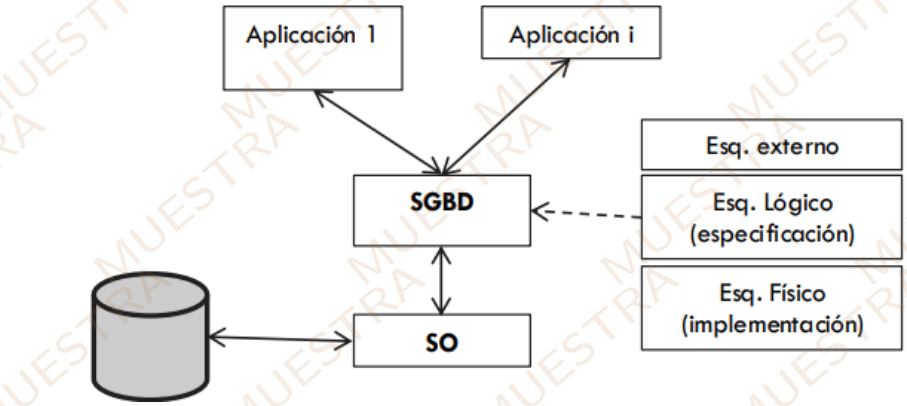
Los SGBD proporcionan:

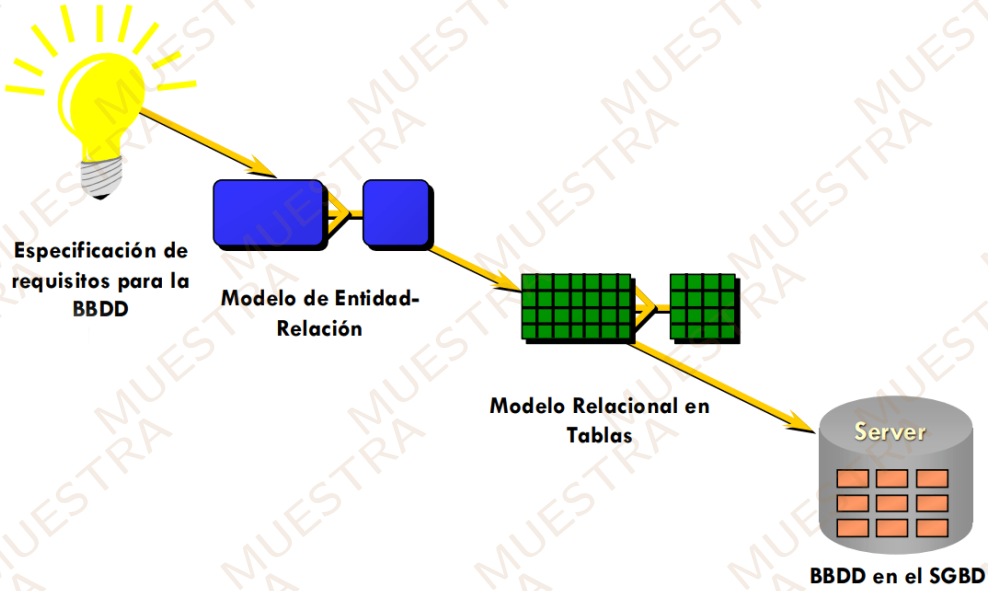
- Lenguajes de definición de esquemas.
- Lenguajes de manipulación de datos.

2.1.3. Arquitectura en un SGBD: Los Tres Niveles de Abstracción

La arquitectura de un SGBD se organiza en tres niveles de abstracción:

1. **Nivel Externo o Esquema externo:** Vistas de usuario derivadas del esquema lógico.
2. **Nivel Lógico o Esquema Lógico:** Estructuras conformes a un modelo de datos.
3. **Nivel Físico o Esquema Físico:** Implementación del nivel lógico.





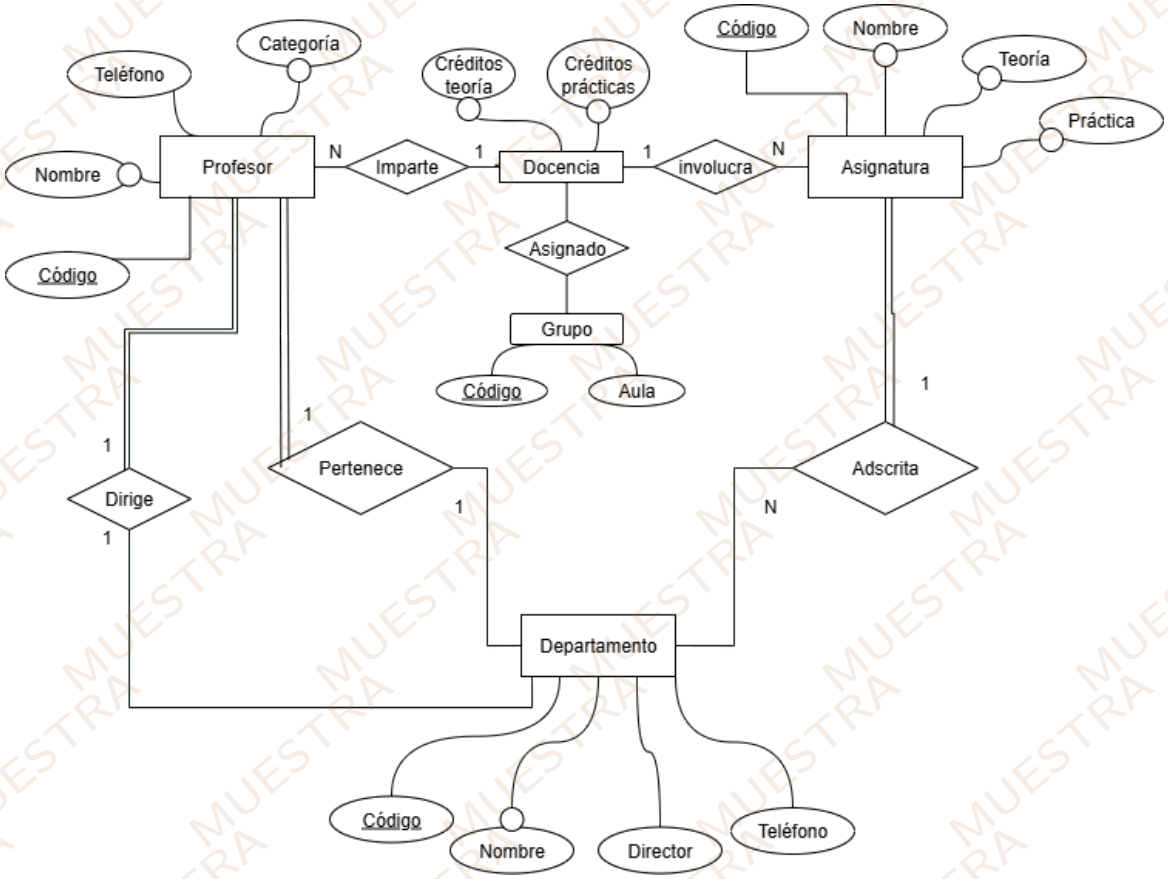
2.3. Ejercicio: Base de Datos Docente

Profesor:
- Código, nombre, teléfono, categoría,
- Departamento al que *pertenece*.
- Asignaturas que imparte, indicando en cada una de ellas el número total de *grupos* de práctica y teoría *asignados*.

Asignatura:
- Código, nombre, curso, semestre, créditos de teoría y práctica, grupos de teoría y práctica.
- Departamento al que está *adscrita*.
- Profesores que la imparten, indicando para cada uno de ellos el número total de grupos de práctica y teoría *asignados*.

Departamento:
- Código, nombre, director, teléfono.
- Profesores que *pertenecen* al mismo.
- Asignaturas *adscritas* al mismo.

Restricciones :
- Un profesor pertenece a un departamento y sólo a uno.
- Una asignatura está adscrita a un departamento y sólo a uno.
- No puede haber dos departamentos con el mismo código.
- No puede haber dos profesores con el mismo código.
- No puede haber dos asignaturas con el mismo código.
- El director del departamento debe ser un profesor.
- Un profesor sólo puede ser director de un departamento.
- Se debe conocer el nombre de los departamentos.
- Se deben conocer el nombre, los créditos, los grupos y el semestre de las asignaturas.
- Se deben conocer los grupos asignados a un profesor en una asignatura.
- Los grupos de clase deben ser enteros positivos.



- Python es un lenguaje de tipado fuerte.
  - Existen incompatibilidades entre tipos de datos. Por ejemplo en la realización de operaciones.

```
a=2
b="7"
c=a+b
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[20], line 3
      1 a=2
      2 b="7"
----> 3 c=a+b

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

3.4. Operadores Aritméticos

Los **operadores aritméticos** son símbolos que se usan para ejecutar operaciones sobre valores numéricos:

Operador	Función	Uso	Ejemplo
+	**Suma**	a + b	>>> 3 + 25
-	**Resta**	a - b	>>> 4 - 7-3
*	**Multiplicación**	a * b	>>> 2 * 612
/	**División**(puede devolver un número real)	a / b	>>> 3.5 / 21.75
//	**División entera**(solo devuelve la parte entera)	a // b	>>> 3.5 // 21.0
%	**Módulo**(devuelve el resto de la división)	a% b	>>> 7% 21
**	**Exponente**	b ** e	>>> 2 ** 664

Otros **operadores de asignación**:

Operador	Uso	Equivalencia
+=	a += n	a = a + n
-=	a -= n	a = a - n
*=	a *= n	a = a * n
/=	a /= n	a = a / n
//=	a //= n	a = a // n
%=	a%= n	a = a% n

Las **expresiones** son combinaciones de variables y operadores.

```
z = x**3 + x*y/2
```

Este código NO DEBEMOS USARLO:

```
edad=int(input('¿Cuántos años tiene?'))
if edad <18:
    print('Usted es menor de edad')
if edad >=18:
    print('Es usted mayor de edad')
print('Hasta la próxima')
```

En realidad, siempre que podamos debemos de usar `else`:

```
edad=int(input('¿Cuántos años tiene?'))
if edad <18:
    print('Usted es menor de edad')
else:
    print('Es usted mayor de edad')
print('Hasta la próxima')
```

- El bloque de sentencias del `if` o del `else` debe contener al menos una instrucción.
- En el caso de que no se quiera realizar ninguna sentencia se debe poner la instrucción `pass` (le indica al programa que ignore esa condición y continúe ejecutando el programa)

Los dos siguientes códigos pot tanto son equivalentes:

```
edad=int(input('¿Cuántos años tiene?'))
if edad < 120:
    pass
else:
    print('No me lo creo!')
```

```
edad=int(input('¿Cuántos años tiene?'))
if edad >= 120:
    print('No me lo creo!')
```

- Un bloque de instruciones puede contener varias instrucciones.
- Todas las instrucciones del bloque deben tener la misma identación (sangrado).

Este código por ejemplo está bien:

```
edad=int(input('¿Cuántos años tiene?'))
if edad <18:
    print('Usted es menor de edad')
    print('Recuerde que está en edad de aprender')
else:
    print('Es usted mayor de edad')
    print('Recuerde que debe seguir aprendiendo')
print('Hasta la próxima')
```

5.2.2. Ejercicios While:

1. Escriba un programa que cuente los números enteros positivos introducidos por el teclado. El programa termina si pulsa el 0.

```
numero = int(input("Introduzca un número"))

suma_positivos = 0

while numero != 0:
    if numero > 0:
        suma_positivos = suma_positivos + numero
    numero = int(input("Introduzca un número"))

print("La suma de los números es ", suma_positivos)
```

2. Escriba un programa que pida de uno en uno números enteros positivos al usuario y muestre el mayor de ellos. El programa termina cuando el numero leído es 0.

```
numero = int(input("Dime un número: "))

maximo = 0

while numero != 0:
    if numero > maximo:
        maximo = numero
        print("El nuevo maximo es",maximo)
    numero = int(input("Dime un número: "))
```

3. Escriba un programa que pida de uno en uno números enteros positivos al usuario y muestre aquellos que sean pares. El programa termina cuando el número leído es 0.

```
numero=int(input("Introduce un número: "))

while numero != 0:
    if numero%2 == 0:
        print("El número ",numero, "que introduciste es par")
        numero=int(input("Introduce un número: "))
    else:
        print("El número introducido no es par, intenteló ")
        numero=int(input("Introduce un número: "))
```

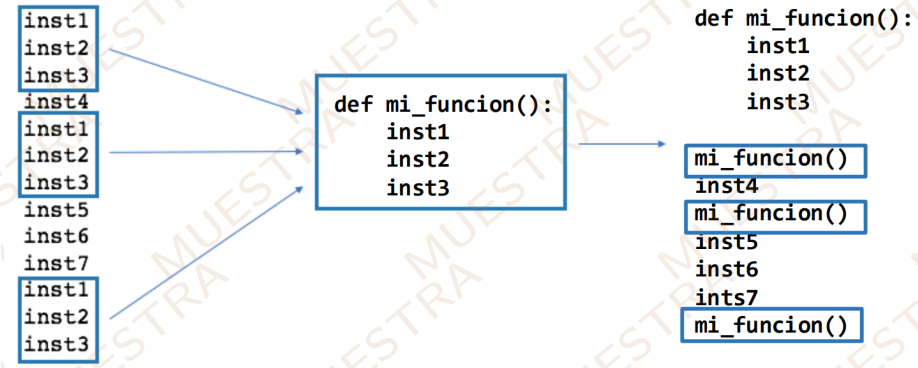
```python
#Ejercicio de clase
def suma (x,y):
    resultado = x + y
    return resultado
n1=int(input("Dame un número"))
n2=int(input("Dame otro número"))

r=suma(n1,n2)

print(r)
```

6.5. Uso habitual

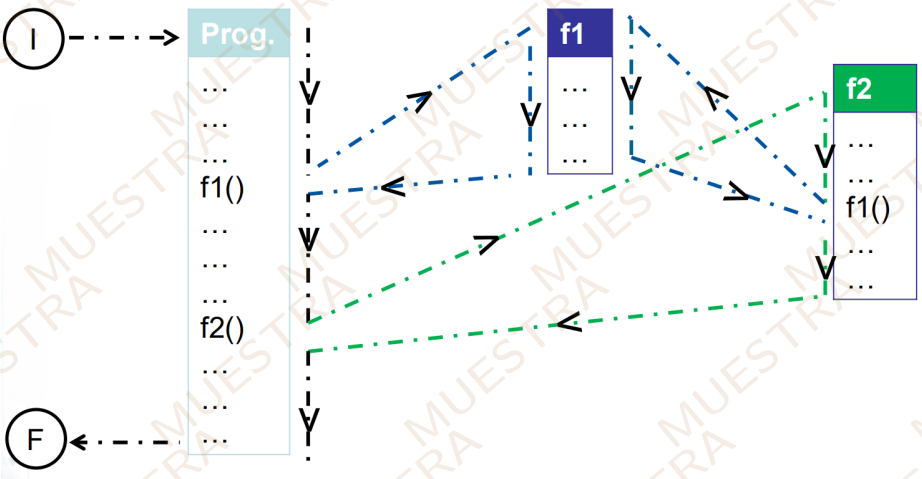
- Reutilización código. Se escribe una vez y se reutiliza muchas veces y se reutiliza muchas veces (por ejemplo `fprint` y `input`).
- Organización del código o para evitar tener un programa principal muy grande.



Hay que elegir bien qué código se convierte en función, cuáles son los parámetros que debe tener y sus tipos.

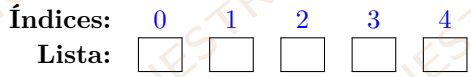
6.6. Flujo de ejecución de un programa

- Cada vez que se invoca una función, el flujo de ejecución abandona la función que está ejecutando en ese momento y salta a la función invocada.
- En cuanto la función invocada termina, el flujo de ejecución retoma el punto en que se produjo la llamada y continua la ejecución por donde se quedó



```
elementos = [3, 'a', 7.2, 'hola']
lista=[1, ['a', 'e', 'i', 'o', 'u'], 8.9, 'hola']
```

- Los elementos de una lista se almacenan en una posición concreta en la lista. Esto permite utilizar un **índice** para acceder a cualquier elemento.
- Un **índice** es un número entero que indica la posición de un elemento en una lista. El primer elemento de una lista siempre comienza en el índice 0.



7.2.1. Acceso a elementos

- Para acceder a un elemento se debe indicar el nombre de la lista seguido de la posición a la que se quiere acceder entre corchetes cuadrados.

```
lista = ['a', 'b', 'd', 'i', 'j']
print(lista[0]) # Imprime 'a'
print(lista[3]) # Imprime 'i'
```

```
a
i
```

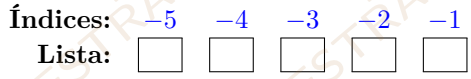
- Si se intenta acceder a un índice que está fuera del rango de la lista, el intérprete lanzará la excepción IndexError. De igual modo, si se utiliza un índice que no es un número entero, se lanzará la excepción TypeError:

```
lista = [1, 2, 3]
lista[8] # Esto lanzará IndexError
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[2], line 2
      1 lista = [1, 2, 3]
----> 2 lista[8] # Esto lanzará IndexError

IndexError: list index out of range
```

- En Python está permitido usar índices negativos para acceder a los elementos de una secuencia desde el final hacia el principio (sentido inverso).



- En este caso, el índice -1 hace referencia al último elemento de la secuencia, el -2 al penúltimo y así, sucesivamente:

```
vocales = ['a', 'e', 'i', 'o', 'u']
print(vocales[-1])  # Imprime 'u'
print(vocales[-4])  # Imprime 'e'
```

```
u
e
```