SISTEMAS INTELIGENTES
Departamento de Ciencia de la Computación e Inteligencia Artificial
Universidad de Alicante **Curso 2025/2026**

Práctica 1. Satisfacción de restricciones

Objetivos:

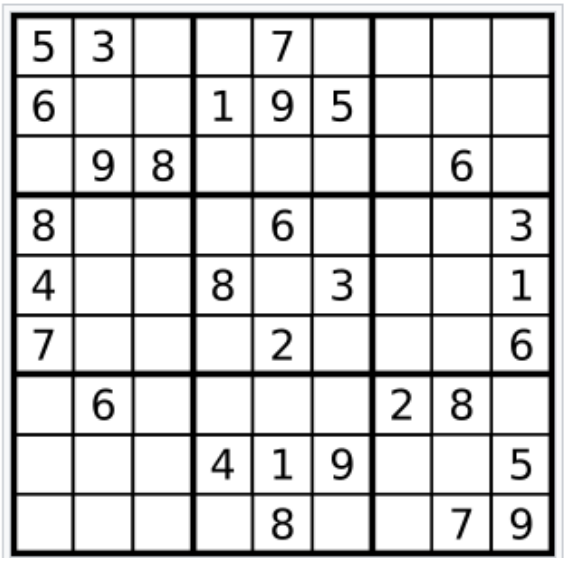
- Implementar tres algoritmos básicos en satisfacción de restricciones.
- Aplicar estos algoritmos a un problema concreto, la resolución de sudokus.

Sesiones 1 y 2: Introducción, entorno de trabajo y backtracking

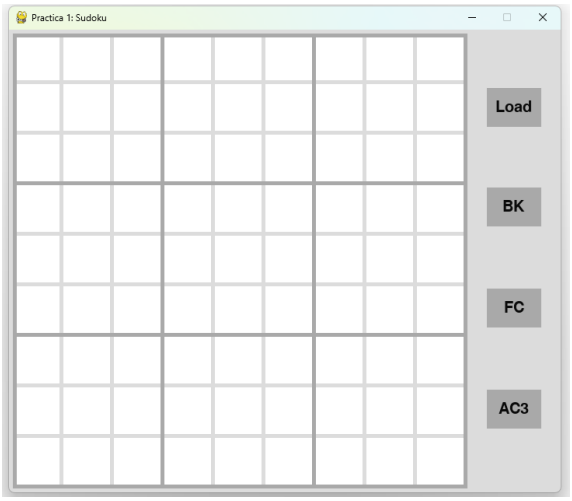
En esta primera práctica de la asignatura se deben desarrollar algoritmos de búsqueda en problemas de satisfacción de restricciones para resolver un sudoku.

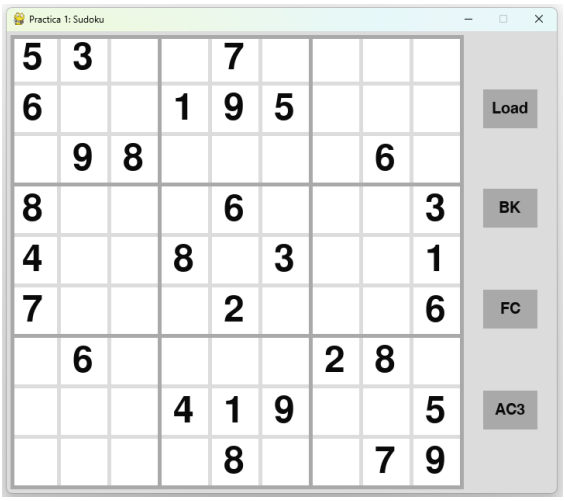
El sudoku es un juego en el que hay que rellenar una cuadrícula con números del 1 al 9. Se utiliza un tablero de 9x9, subdividido en 9 submatrices de 3x3. En cada casilla del tablero se pueden colocar números del 1 al 9, con la restricción que ese número no puede repetirse en la misma fila, en la misma columna y en la submatriz donde se encuentre. Visto de otra manera, en cada fila deben aparecer los números del 1 al 9, en cada columna deben aparecer los números del 1 al 9 y cada submatriz debe tener los números del 1 al 9.

Para empezar el sudoku, se dan una serie de números iniciales y el resto de las casillas se dejan vacías. El sudoku está completo cuando se rellenan todas las casillas cumpliendo las reglas establecidas. Debemos ir colocando números del 1 al 9 en cada casilla vacía, de tal forma que no se repitan en su fila, columna o submatriz.



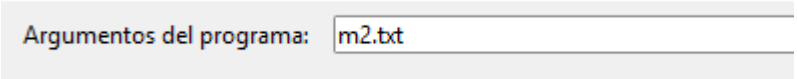
Se proporciona un entorno gráfico desarrollado en Python en el cual es posible cargar un fichero de texto con la configuración inicial del sudoku (botón Load).





Funcionamiento del juego

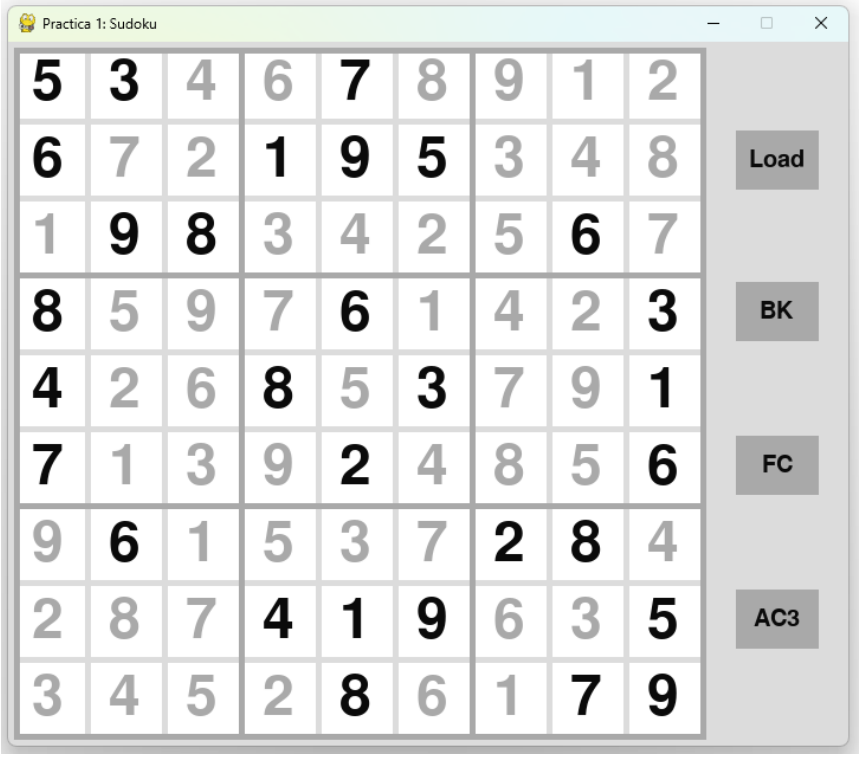
Al ejecutar el juego se cargará por defecto m1.txt. Si se quiere elegir otra plantilla distinta, hay que pasarla como argumento. Para ello, desde Thonny acceder a menú Visualización- Parámetros de ejecución del programa, en el cuadro de texto que aparece escribir el nombre de la plantilla que se quiere usar.



Botones:

- **Load**: carga una configuración inicial de Sudoku.
- **BK**: aplicar el algoritmo backtracking. Este algoritmo puede aplicarse tanto para una cuadrícula con su configuración inicial como tras la ejecución de AC3. En este último caso los dominios de las variables al comenzar a ejecutar FC serán los devueltos por AC3.
- **FC**: aplicar el algoritmo Forward Checking. Este algoritmo puede aplicarse tanto para una cuadrícula con su configuración inicial como tras la ejecución de AC3. En este último caso los dominios de las variables al comenzar a ejecutar FC serán los devueltos por AC3.
- **AC3**: aplicar el algoritmo AC3. Este algoritmo permite reducir los dominios de las variables y elimina inconsistencias de arista.

Al pinchar en los botones BK o FC se resolverá el sudoku empleando el algoritmo correspondiente y se mostrará en el tablero la solución del sudoku.



EL color gris o negro del número indica si la celda la ha rellenado el algoritmo (gris) o estaba en la plantilla (negro).

Desarrollo

El entorno se ha desarrollado en Python usando el paquete `pygame`. `Pygame` facilita la creación de videojuegos en dos dimensiones y permite programar la parte multimedia (gráficos, sonido y manejo de eventos) de forma sencilla.

El entorno proporcionado está compuesto por 2 ficheros:

- main: es el fichero que hay que ejecutar para lanzar el entorno. Tiene el bucle principal de manejo del juego, así como el desarrollo del interfaz gráfico
- tablero: permite construir un objeto tablero que contiene el tamaño del mismo, así como la matriz que representa las celdas de la cuadrícula.

Como herramienta de desarrollo se utilizará `Thonny`. `Thonny` es un IDE con los elementos básicos para crear aplicaciones en el lenguaje Python, es posible depurar y observar las variables en ejecución de forma sencilla.

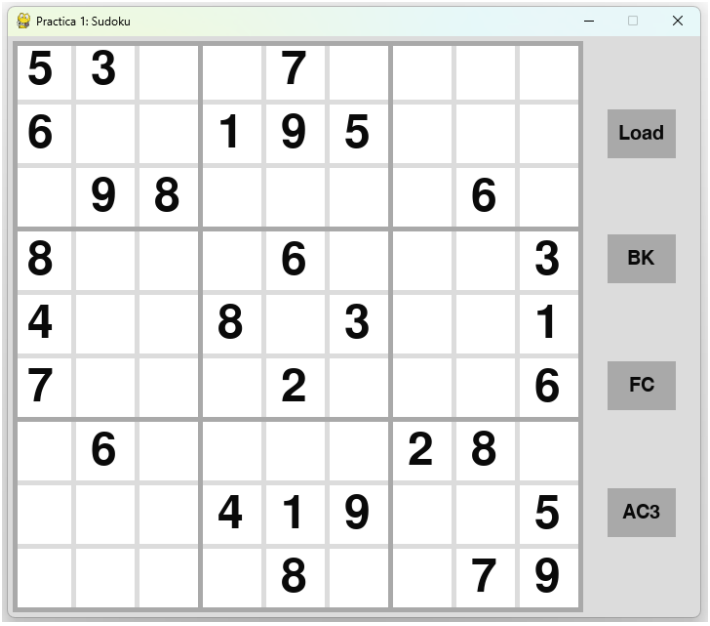
El entorno muestra en todo momento el contenido de la variable tablero. Por eso, cuando una celda del tablero tiene valor muestra su contenido en pantalla (función **pintarTablero**)

Ejecución del proyecto Ejecutar el fichero main.py

Definición de la plantilla del sudoku

El fichero de texto que contiene la plantilla tiene 9 filas donde en cada fila se indica el número correspondiente a la celda o un 0 si la casilla está vacía:

```
5 3 0 0 7 0 0 0 0
6 0 0 1 9 5 0 0 0
0 9 8 0 0 0 0 6 0
8 0 0 0 6 0 0 0 3
4 0 0 8 0 3 0 0 1
7 0 0 0 2 0 0 0 6
0 6 0 0 0 0 2 8 0
0 0 0 4 1 9 0 0 5
0 0 0 0 8 0 0 7 9
```



Tareas a realizar en estas sesiones:

- Prueba el entorno. Averigua cómo acceder a una celda del sudoku.
- Implementa la clase Variable
- Implementa el algoritmo backtraking para resolver el problema

Sesiones 3, 4 y 5: Implementación del algoritmo Forward Checking. Hito intermedio de entrega

El algoritmo Forward Checking asigna valores a las variables que componen el problema. Cada vez que asigna un valor a una variable (variable actual), comprueba esa asignación con las variables que todavía están sin instanciar (variables futuras) que están restringidas con la variable actual. Los valores de las variables futuras que son inconsistentes con esa asignación son eliminados temporalmente de los dominios de las variables. Si el dominio de una variable futura se queda vacío, la instanciación de la variable actual se deshace y se prueba con otro valor.

1. Seleccionar x_i.
2. Instanciar $x_i \leftarrow a_i : a_i \in D_i$.
3. Razonar hacia adelante (forward-check):
   - Eliminar de los dominios de las variables $(x_{i+1}, \ldots, x_n)$ aún no instanciadas, aquellos valores inconsistentes con respecto a la instanciación (x_i, a_i), de acuerdo al conjunto de restricciones.
4. Si quedan valores posibles en los dominios de todas las variables por instanciar, entonces:
   - Si $i < n$, incrementar i, e ir al paso (1).
   - Si $i = n$, salir con la solución.
5. Si existe una variable por instanciar, sin valores posibles en su dominio, entonces retractar los efectos de la asignación $x_i \leftarrow a_i$. Hacer:
   - Si quedan valores por intentar en D_i, ir al paso (2).
   - Si no quedan valores:
     - Si $i > 1$, decrementar i y volver al paso (2).
     - Si $i = 1$, salir sin solución.

```
funcion FC(i variable): booleano
    para cada a∈ factibles[i] hacer
        Xi ← a
        si i=N solución retorna CIERTO
        sino
                si forward (i,a)
                        si FC(i+1) retorna CIERTO
                restaura (i)
    retorna FALSO
funcion forward(i variable, a valor): booleano
    para toda j=i+1 hasta N hacer
        Vacio ← CIERTO
        para cada b ∈ factibles[j] hacer
                si (a,b) ∈ Rij vacio ←FALSO
                sino eliminar b de factible[j]
                        Añadir b a podado[j]
        si vacio retorna FALSO
    retorna CIERTO
procedimiento restaura(i variable)
    para toda j=i+1 hasta N hacer
        para todo b ∈ podado[j] hacer
                si Xi responsable filtrado b
                        Eliminar b de podado[j]
                        Añadir b a factible[j]
```

Detalles de implementación

Para que se considere válido el algoritmo forward checking se debe implementar de manera adecuada. Debe haber una clase **Variable** y los objetos de dicha clase deben tener un dominio, el algoritmo va eliminando y restaurando palabras de ese dominio conforme va avanzando en la ejecución. El objetivo final del algoritmo es que cada variable tenga un valor asignado. Cuando se asigna un valor a una variable, esa asignación debe causar podas en los dominios de las variables futuras (variables que todavía no tienen un valor asignado) y cuando se deshace una asignación de un valor a una variable, el procedimiento restaura debe revisar las variables futuras que están restringidas con la variable actual y debe deshacer las eliminaciones causadas por dicha variable.

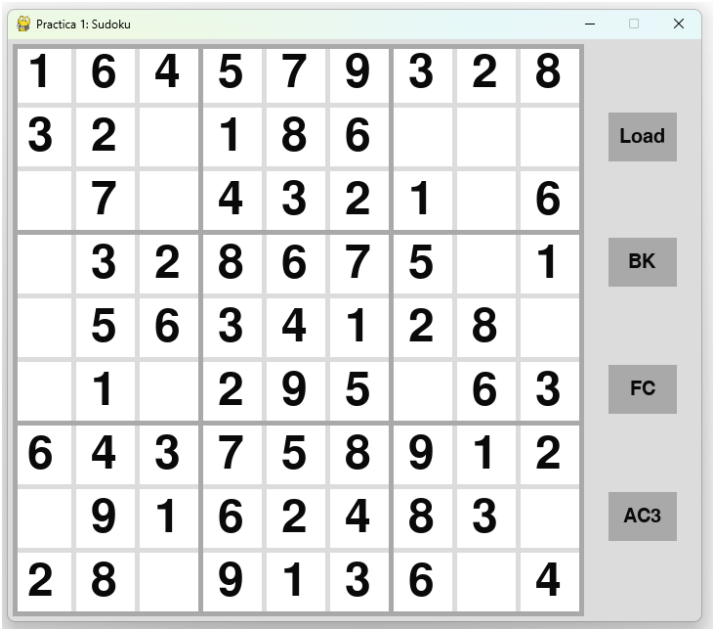
Tareas a realizar en estas sesiones:

- Implementar el algoritmo Forward checking.
- **Antes de la sesión 6 se debe entregar el proyecto**

Sesiones 6 y 7: Implementación el algoritmo AC3

En estas sesiones se va a implementar el algoritmo AC3. Este algoritmo transforma un problema en otro sin inconsistencias de arco. Cuando se pulse en el botón del AC3 se debe mostrar por pantalla los dominios de cada variable antes y después de ejecutar el algoritmo.

Ejemplo:



```
DOMINIOS ANTES DEL AC3
00   Dominio: 1,
01   Dominio: 6,
02   Dominio: 4,
03   Dominio: 5,
04   Dominio: 7,
05   Dominio: 9,
06   Dominio: 3,
07   Dominio: 2,
08   Dominio: 8,
10   Dominio: 3,
11   Dominio: 2,
12   Dominio: 1, 2, 3, 4, 5, 6, 7, 8, 9,
13   Dominio: 1,
14   Dominio: 8,
15   Dominio: 6,
16   Dominio: 1, 2, 3, 4, 5, 6, 7, 8, 9,
17   Dominio: 1, 2, 3, 4, 5, 6, 7, 8, 9,
18   Dominio: 1, 2, 3, 4, 5, 6, 7, 8, 9,
20   Dominio: 1, 2, 3, 4, 5, 6, 7, 8, 9,
21   Dominio: 7,
22   Dominio: 1, 2, 3, 4, 5, 6, 7, 8, 9,
23   Dominio: 4,
24   Dominio: 3,
25   Dominio: 2,
26   Dominio: 1,
27   Dominio: 1, 2, 3, 4, 5, 6, 7, 8, 9,
28   Dominio: 6,
30   Dominio: 1, 2, 3, 4, 5, 6, 7, 8, 9,
31   Dominio: 3,
32   Dominio: 2,
```

```
DOMINIOS DESPUES DEL AC3
00   Dominio: 1,
01   Dominio: 6,
02   Dominio: 4,
03   Dominio: 5,
04   Dominio: 7,
05   Dominio: 9,
06   Dominio: 3,
07   Dominio: 2,
08   Dominio: 8,
10   Dominio: 3,
11   Dominio: 2,
12   Dominio: 5, 9,
13   Dominio: 1,
14   Dominio: 8,
15   Dominio: 6,
16   Dominio: 4, 7,
17   Dominio: 4, 5, 7, 9,
18   Dominio: 5, 7, 9,
20   Dominio: 5, 8, 9,
21   Dominio: 7,
22   Dominio: 5, 8, 9,
23   Dominio: 4,
24   Dominio: 3,
25   Dominio: 2,
26   Dominio: 1,
27   Dominio: 5, 9,
28   Dominio: 6,
30   Dominio: 4, 9,
31   Dominio: 3,
32   Dominio: 2,
```

$Q = \{c(e_p) = \langle V_i, V_j \rangle | e_p \in E, i \neq j\}$
Mientras $Q \neq \emptyset$ hacer
    $\langle V_k, V_m \rangle$ = seleccionar_y_borrar(Q)
    cambio = falso
    Para todo $v_k \in D_k$ hacer
        Si no_consistente (v_k, D_m) entonces
            borrar (v_k, D_k)
            cambio = cierto
        FinSi
    FinPara
    Si Dk = Ø entonces salir_sin_solución FinSi
    Si cambio = cierto entonces
        $Q = Q \cup \{c(e_r) = \langle V_i, V_k \rangle \mid e_r \in E, i \neq k, i \neq m\}$
    FinSi
FinMientras

Si se pulsa primero en el botón del AC3 y a continuación en el botón del forward checking o en el de backtracking, estos algoritmos deben trabajar con los dominios ya reducidos por el AC3.

Tarea a realizar en estas sesiones:

- Implementar el algoritmo AC3

Sesión 8: Documentación y pruebas. Hito final

Esta última sesión está dedicada a terminar las pruebas y finalizar la documentación que se habrá ido elaborando de manera continua durante todo el desarrollo de la práctica.

La documentación es la parte más importante de la práctica (60%). **Como mínimo** debe contener:

- Explicación de la clase Variable
- Explicación de cómo se han tratado las casillas con números fijos.
- Especificación **formal** del problema. La especificación formal debe detallar la tupla **<V, E, c, l, a>**
- Apartado de experimentación en la que se describan las diferentes pruebas realizadas a los algoritmos, dejando bien claro el objetivo de las pruebas: plantillas con muchas celdas fijas, con pocas, etc.
- Estudio de tiempos. Determinar si la aplicación de AC3 reduce los tiempos de ejecución de los otros algoritmos con distintos sudokus.
- Utilización de gráficas para el análisis de tiempos.
- Dificultades encontradas.
- Uso de IAs generativas: explicación del uso que se ha hecho de ellas.
- Apartado de referencias bibliográficas.

Entrega de la práctica

La fecha límite de entrega de la práctica es el 2 **de noviembre de 2025** a las 23:55h. La entrega se realizará a través de Moodle.

Formato de entrega del proyecto para el hito 2 (entrega final)

La entrega debe consistir en un **fichero comprimido zip** con tres carpetas:

- /Cod: todos los ficheros .py, elementos gráficos y plantillas de sudokus que constituyen la práctica
- /Doc: fichero pdf con la documentación

¡¡¡AVISO IMPORTANTE!!!

No cumplir cualquiera de las normas de formato/entrega puede suponer un suspenso en la práctica.

Recordad que **las prácticas son INDIVIDUALES** y NO se pueden hacer en parejas o grupos.

Cualquier código copiado supondrá un suspenso de la práctica para todas las personas implicadas en la copia y, como indica el Reglamento para la Evaluación de Aprendizajes de la Universidad de Alicante (BOUA 9/12/2015) y el documento de Actuación ante copia en pruebas de evaluación de la EPS, **se informará a la dirección de la Escuela Politécnica Superior para la toma de medidas oportunas**.

Plan de entrega por hitos

Durante el periodo de ejecución de la práctica se realizarán dos hitos de entrega. Es obligatorio cumplir las fechas de las entregas correspondientes:

Hito	Entrega	Fecha tope
1 (intermedio)	Todos los ficheros .py y elementos gráficos que constituyen la práctica. No es necesario entregar documentación	12 de octubre
2 (final)	Práctica completa siguiendo la estructura del formato de entrega	2 de noviembre

- La no entrega del hito 1 en la fecha prevista supone una penalización del 20%.
- Para la entrega final se dejará el programa empleando la heurística más adecuada. Las otras heurísticas analizadas aparecerán en el código, aunque no se utilicen.

La nota de la práctica sufrirá una penalización de dos puntos si no se cumple rigurosamente con los requisitos de la entrega (tanto en la estructura de los ficheros entregados como en la salida que debe generar la práctica)