


	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 1</p>

INFORME DE LABORATORIO

N° 05

INFORMACIÓN BÁSICA					
ASIGNATURA:	Programación Web 2.				
TÍTULO DE LA PRÁCTICA:	Python				
NÚMERO DE PRÁCTICA:	05	AÑO LECTIVO:	2025-A	NRO. SEMESTRE:	III
FECHA DE PRESENTACIÓN	14/05/2025	HORA DE PRESENTACIÓN	18:30:00		
INTEGRANTE (s) Subia Huaicane Edson Fabricio				NOTA (0-20)	Nota colocada por el docente
DOCENTE(s): Ing. Carlo Corrales					

RESULTADOS Y PRUEBAS
<ul style="list-style-type: none"> EJERCICIOS RESUELTOS: Cree un Proyecto ... En este enlace se encuentra en el repositorio y los commits que se realizaron para la creación y/o mejora de este programa: https://github.com/Q3son/Pweb2---Edson-Subia.git https://github.com/Q3son/Pweb2---Edson-Subia/tree/main/Lab_05 VIDEO SOBRE LA PÁGINA WEB CREADA: https://youtu.be/SzRJI2OoxPk

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 2</p>

1. Descripción General del Sistema

Desarrollamos un sistema de visualización de ajedrez que:

- Representa piezas y tableros mediante arte ASCII.
- Implementa operaciones gráficas avanzadas (superposición, espejos, repeticiones).
- Incluye 7 ejercicios progresivos (A-G) que demuestran capacidades del sistema.

Tecnologías clave:

- Python + Pygame (para interpretación visual).
- Programación orientada a objetos (clase Picture).
- Git para control de versiones.

2. Métodos implementados en Picture.py

2.1. Métodos Básicos

- `__init__(img)`: Constructor: Crea una imagen a partir de una lista de strings.
- `join(p)`: Une **horizontalmente** dos imágenes (self + p).
- `up(p)`: Apila **verticalmente** dos imágenes (self arriba de p).
- `negative()`: Invierte colores (blanco ↔ negro).

2.2. Métodos de Transformación

- ❖ `verticalMirror()`: Reflejo vertical (espejo horizontal).
- ❖ `horizontalMirror()`: Reflejo horizontal (espejo vertical).
- ❖ `horizontalRepeat(n)`: Repite la imagen **n veces** en horizontal.
- ❖ `verticalRepeat(n)`: Repite la imagen **n veces** en vertical.

2.3. Método Innovador

- `overlay(background)`: Superpone la imagen actual sobre un fondo, conservando transparencias.
- **Uso definitivo**: Para colocar piezas *dentro* de los cuadros del tablero y usarlo para programación directa del juego de ajedrez (captura de movimiento)

3. Ejercicios implementados

3.1. Ejercicio A (4 caballos)

```
caballo_blanco.join(caballo_negro).up(caballo_negro.join(caballo_blanco))
```

3.2. Ejercicio B (caballos inferiores invertidos)

```
caballo.verticalMirror() # Voltea caballos inferiores
```

3.3. Ejercicio C: 4 Reinas Blancas



```
reina.horizontalRepeat(4)
```

3.4. Ejercicio D: Línea de tablero

```
square.join(square.negative()).horizontalRepeat(4)
```

3.5. Ejercicio E: Línea de tablero invertida

```
square.negative().join(square).horizontalRepeat(4)
```

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 3</p>

3.6. Ejercicio F: Mitad de Tablero

```
fila_base.up(fila_base.negative()).verticalRepeat(2)
```

3.7. Ejercicio G: El tablero completo (Eso lo mostraremos como parte del código principal junto a picture.py)

4. Innovación vs Enfoque Tradicional

4.1. Ventajas del Sistemas actual

4.1.1. Precisión

- Piezas centradas automáticamente en cuadros (overlay()), tomadas como objetos aparte.

```
# Sin overlay (problema):
draw("X ") # Pieza desalineada
# Con overlay (solución):
draw(place_piece(torre, cuadro_negro)) # Perfectamente centrada
```

4.1.2. Extensibilidad

- Métodos como overlay() permiten:
 - ✓ Animación de movimientos.
 - ✓ Efectos de captura (superponer piezas).
 - ✓ Ocupar el máximo espacio con el tablero.

5. En la siguiente sección mostraré picture.py (todos los métodos usados) y el código del tablero donde aplicamos todo (El código fuente versionado por commits y su ejecución se visualizan mucho mejor en el repositorio y el video al inicio de este informe).

```
You, 2 hours ago | 1 author (You)
1 from colors import *
2
3 You, 2 hours ago | 1 author (You)
4 class Picture:
5     def __init__(self, img):
6         self.img = img
7
8     def __eq__(self, other):
9         return self.img == other.img
10
11     def _invColor(self, color):
12         return inverter.get(color, color) # Usando .get() como condicional implícito
13
14     def verticalMirror(self):
15         """Espejo vertical: invierte cada fila"""
16         return Picture([row[::-1] for row in self.img]) # Lista por comprensión
17
18     def horizontalMirror(self):
19         """Espejo horizontal: invierte el orden de las filas"""
20         return Picture(self.img[::-1]) # Slicing permitido
21
22     def negative(self):
23         """Negativo: aplica _invColor a cada carácter"""
24         return Picture([
25             ''.join([self._invColor(c) for c in row]) # join + lista por comprensión
26             for row in self.img
27         ])
28
29     def join(self, p):
30         """Une horizontalmente: self + p"""
31         return Picture([
32             self.img[i] + p.img[i] # Concatenación (+) con range
33             for i in range(len(self.img))
34         ])
```

```

34
35     def up(self, p):
36         """Une verticalmente: self arriba de p"""
37         return Picture(self.img + p.img) # Concatenación de listas (+)
38
39     def under(self, p):
40         """Une verticalmente: p arriba de self"""
41         return Picture(p.img + self.img) # Concatenación de listas (+)
42
43     def horizontalRepeat(self, n):
44         """Repite la imagen horizontalmente n veces"""
45         return Picture([
46             row * n # Uso de * para strings (equivalente a join)
47             for row in self.img
48         ])
49
50     def verticalRepeat(self, n):
51         """Repite la imagen verticalmente n veces"""
52         return Picture(self.img * n) # Uso de * para listas
53
54     def overlay(self, background):
55         """Superpone esta imagen (pieza) sobre un fondo (cuadro).
56         La pieza reemplazará los espacios vacíos ( ' ') del fondo."""
57         # Verificar que las imágenes tienen el mismo tamaño
58         if len(self.img) != len(background.img) or len(self.img[0]) != len(background.img[0]):
59             raise ValueError("Las imágenes deben tener las mismas dimensiones")
60
61         combined = []
62         for piece_row, bg_row in zip(self.img, background.img):
63             combined_row = ""
64             for piece_char, bg_char in zip(piece_row, bg_row):
65                 # Mantener el carácter de la pieza si no es espacio vacío, sino usar el fondo
66                 combined_row += piece_char if piece_char.strip() else bg_char
67             combined.append(combined_row)
68         return Picture(combined)

```

ExerciseG



```

1  from chessPictures import *
2  from interpreter import draw
3
4  def place_piece(piece, square_img):
5      """Coloca una pieza centrada
6      piece_img = piece.img
7      background = [list(row) for row in square_img]
8
9      # Calcular posición de inicio
10     start_y = (len(background) - len(piece_img)) // 2
11     start_x = (len(background[0]) - len(piece_img[0])) // 2
12
13     # Superponer la pieza
14     for y in range(len(piece_img)):
15         for x in range(len(piece_img[y])):
16             if piece_img[y][x].strip(): # Si no es espacio vacío
17                 background[y + start_y][x + start_x] = piece_img[y][x]
18
19     # Convertir de vuelta a strings
20     combined = [''.join(row) for row in background]
21     return Picture(combined)

```

```
23 def create_piece_row(pieces, is_white_row):
24     """Crea una fila de piezas sobre cuadros alternados"""
25     row_pictures = []
26     for i, piece in enumerate(pieces):
27         square_color = square if (i + is_white_row) % 2 == 0 else square.negative()
28         placed_piece = place_piece(piece, square_color)
29         row_pictures.append(placed_piece)
30
31     # Unir progresivamente las imágenes
32     result = row_pictures[0]
33     for pic in row_pictures[1:]:
34         result = result.join(pic)
35     return result
```

```
36
37 # Configuración de piezas
38 black_pieces = [rock, knight, bishop, queen, king, bishop, knight, rock]
39 white_pieces = [rock, knight, bishop, queen, king, bishop, knight, rock]
40
41 # Construir filas
42 first_row = create_piece_row([p.negative() for p in black_pieces], False)
43 second_row = create_piece_row([pawn.negative()*8, True)
44 seventh_row = create_piece_row([pawn]*8, False)
45 eighth_row = create_piece_row(white_pieces, True)
46
47 # Tablero central vacío
48 empty_pair = square.join(square.negative()).horizontalRepeat(4)
49 center_board = empty_pair.up(empty_pair.negative()).verticalRepeat(2)
50
51 # Ensamblar tablero completo
52 chessboard = (
53     first_row
54     .up(second_row)
55     .up(center_board)
56     .up(seventh_row)
57     .up(eighth_row)
58 )
59
60 draw(chessboard)
```

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 6</p>

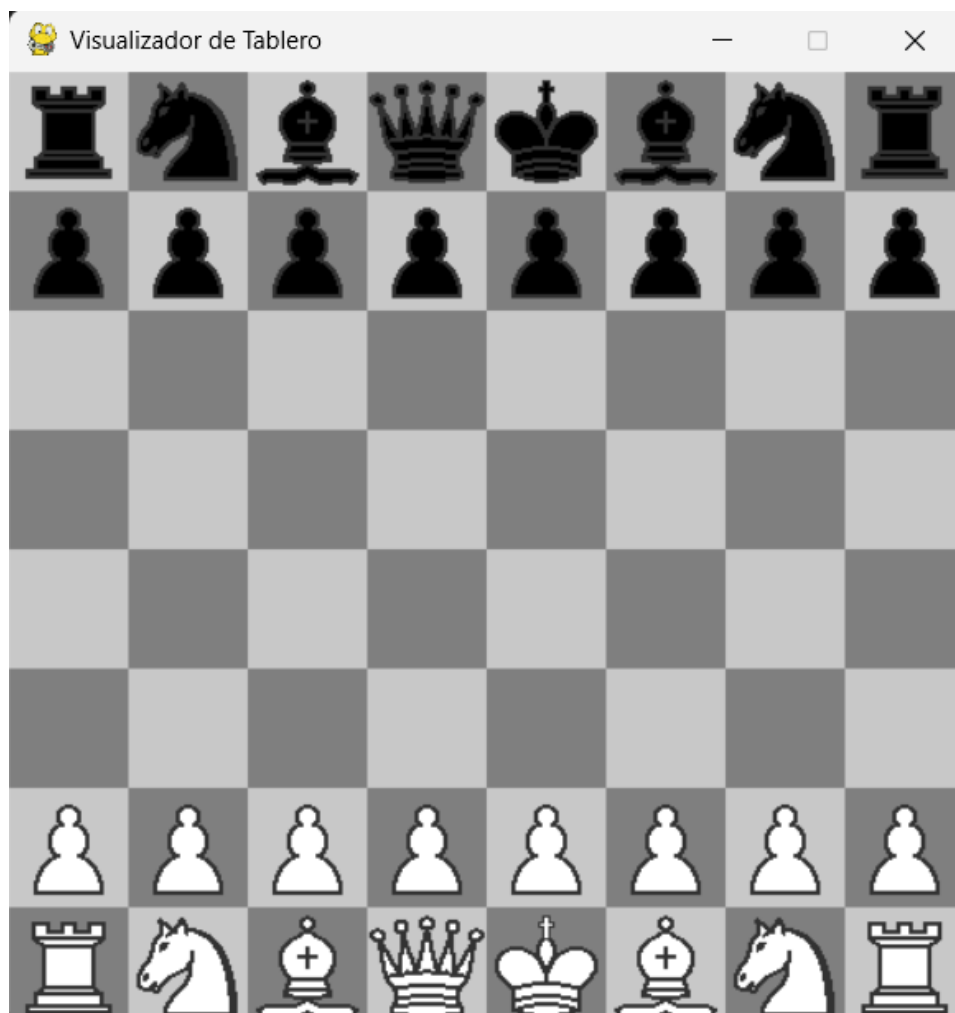
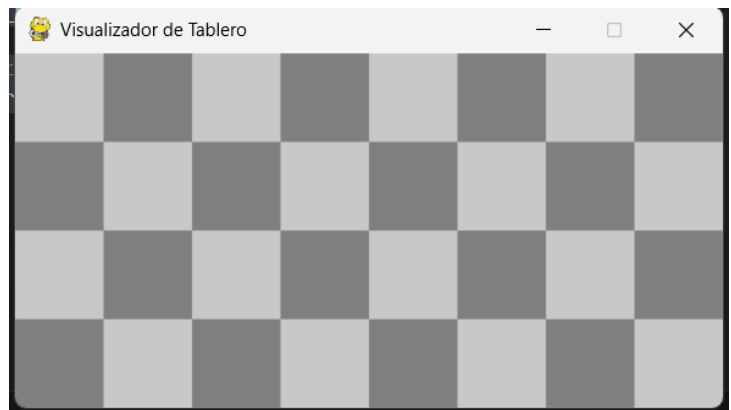
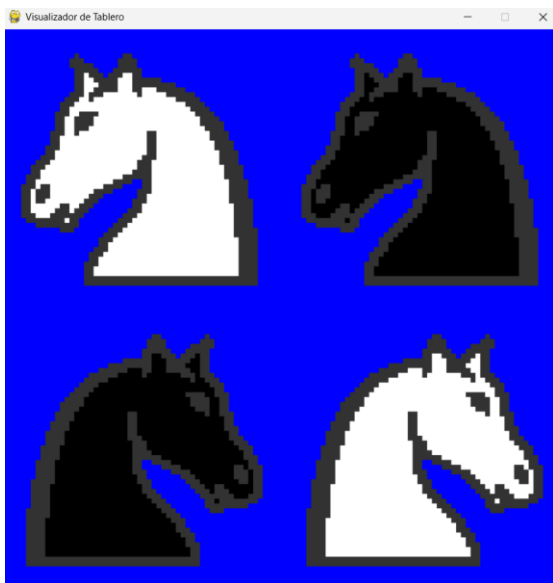
Cuestionario



1. ¿Qué método de la clase **Picture** permite superponer piezas sobre los cuadros del tablero y cómo funciona?
 - Se implementó el método **overlay()** que combina una pieza con un cuadro
 - Conserva los caracteres visibles de la pieza y rellena los espacios vacíos con el fondo
 - Permite centrado automático para posicionamiento preciso

2. ¿Cómo se resolvió el posicionamiento de las piezas negras en el Ejercicio G?
 - Uso de **negative()** para convertir piezas blancas a negras
 - Creación de filas con alternancia de cuadros (blanco/negro) mediante **join()**
 - Aplicación de **overlay()** para integrar cada pieza en su cuadro correspondiente

3. ¿Qué ventaja ofrece el método **horizontalRepeat()** en la construcción del tablero?
 - Simplifica la creación de patrones repetitivos como filas de peones
 - Evita duplicación manual de código al generar 8 cuadros consecutivos
 - Mantiene consistencia visual al garantizar repeticiones exactas

(Algunas capturas sobre el buen funcionamiento de nuestra página web)



	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 8</p>

Lecciones Clave

- **Optimización gráfica: El método overlay() demostró ser eficiente para superposiciones, pero requiere que las imágenes tengan dimensiones compatibles.**
- **Métodos reutilizables: Implementar operaciones como join() y up() facilitó la composición modular del tablero.**
- **Precisión visual: Centrar piezas con overlay() fue crucial para un diseño profesional, aunque exigió ajustes manuales en algunos casos.**

Recomendaciones para Futuros Laboratorios

- **Extender la clase Picture** para soportar animaciones básicas (ej: mover piezas).
- **Implementar validación de movimientos** usando las coordenadas de los cuadros.
- **Añadir interacción** con Pygame para selección de piezas.
- **Optimizar renderizado** para tableros más grandes.

CONCLUSIONES



*El laboratorio demostró que un enfoque **orientado a objetos** para gráficos basados en texto permite construir sistemas complejos (como un tablero de ajedrez) mediante operaciones composicionales. La implementación de overlay() destaca como base para futuras extensiones interactivas.*

METODOLOGÍA DE TRABAJO

Colocar la metodología de trabajo que ha utilizado el estudiante o el grupo para resolver la práctica, es decir el procedimiento/secuencia de pasos en forma general.

1. **Desarrollo incremental:** Cada ejercicio (A-G) validó métodos específicos de Picture.
2. **Refactorización:** El método overlay() surgió de iteraciones en el Ejercicio G.
3. **Pruebas visuales:** Uso constante de draw() para verificar resultados.

REFERENCIAS Y BIBLIOGRAFÍA

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 9</p>

Colocar las referencias utilizadas para el desarrollo de la práctica en formato IEEE

- Javascript tutorial. \url <https://www.w3schools.com/js/>, 2024. Accessed: 02-05-2024.
- Loiane Groner. *Learning JavaScript Data Structures and Algorithms: Write complex and powerful*
- <https://github.com/Q3son/Pweb2---Edson-Subia.git>
- <https://youtu.be/SzRJI2OoxPk>