

Informe de Laboratorio 5

Tema: PYTHON

Nota

Estudiante	Escuela	Asignatura
Sergio Hanco Mullisaca shanccom@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Programación Web 2 Semestre: II Código:

Laboratorio	Tema	Duración
5	PYTHON	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2024 - A	Del 20 Mayo 2024	Al 25 Mayo 2024

1. Tarea

- Informe de laboratorio
- Video en Flip
- Ejercicios Propuestos

2. Equipos, materiales y temas utilizados

- VS
- Git 2.39.2.
- Cuenta en GitHub con el correo institucional.

3. URL de Repositorio Github

- URL del video en yt.
- <https://youtu.be/9gy3cyhix1s>
- URL del video en flip.
- <https://flip.com/s/sdWGsD28pRbD>
- URL del GITHUB.
- https://github.com/shanccom/Programacion_Web_2.git

4. Actividades

4.1. EJERCICIO I

Listing 1: EJERCICIOS I

```
from colors import *
class Picture:
    def __init__(self, img):
        self.img = img

    def __eq__(self, other):
        return self.img == other.img

    def _invColor(self, color):
        if color not in inverter:
            return color
        return inverter[color]

    def verticalMirror(self):
        imgVertical = [fila for fila in reversed(self.img)]
        return Picture(imgVertical)

    def horizontalMirror(self):
        imgHorizontal = [fila[::-1] for fila in self.img]
        return Picture(imgHorizontal)

    def negative(self):
        imgInvertida = []
        for fila in self.img:
            filaInvertida = ''.join([self._invColor(char) for char in fila])
            imgInvertida.append(filaInvertida)
        return Picture(imgInvertida)

    def join(self, p):
        imgNueva = []
        for x in range(len(self.img)):
            imgNueva.append(self.img[x] + p.img[x])
        return Picture(imgNueva)

    def up(self, p):
        superposicion = []
```

```
for i in range(max(len(self.img), len(p.img))):
    if i < len(p.img):
        if i < len(self.img):
            superposicion.append(''.join([self.img[i][j] if self.img[i][j] != ' ' else
            p.img[i][j] for j in range(max(len(self.img[i]), len(p.img[i])))]))
        else:
            superposicion.append(p.img[i])
    else:
        superposicion.append(self.img[i])
return Picture(superposicion)

def under(self, p):
    imgNueva = []
    imgNueva = self.img + p.img
    return Picture(imgNueva)

def horizontalRepeat(self, n):
    if n <= 0:
        return Picture([])
    imgNueva = [row * n for row in self.img]
    return Picture(imgNueva)

def verticalRepeat(self, n):
    if n <= 0:
        return Picture([])
    imgNueva = self.img * n
    return Picture(imgNueva)
```

4.2. Ejercicio II

■ EJERCICIO 2 - A

Listing 2: EJERCICIO 2-A

```
from interpreter import draw
from chessPictures import *

filaPrimera = knight.join(knight.negative())
filaSegunda = knight.negative().join(knight)

final = filaPrimera.under(filaSegunda)

draw(final)
```



■ EJERCICIO 2 - B

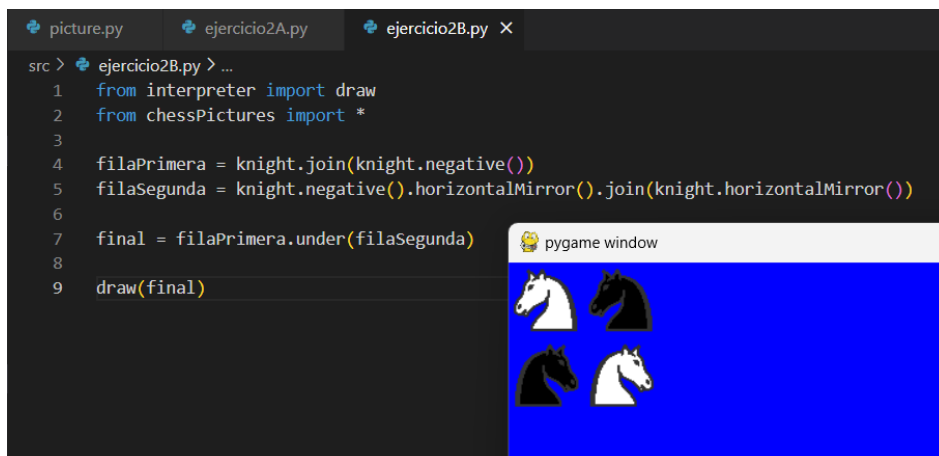
Listing 3: EJERCICIO 2-B

```
from interpreter import draw
from chessPictures import *

filaPrimera = knight.join(knight.negative())
filaSegunda = knight.negative().horizontalMirror().join(knight.horizontalMirror())

final = filaPrimera.under(filaSegunda)

draw(final)
```



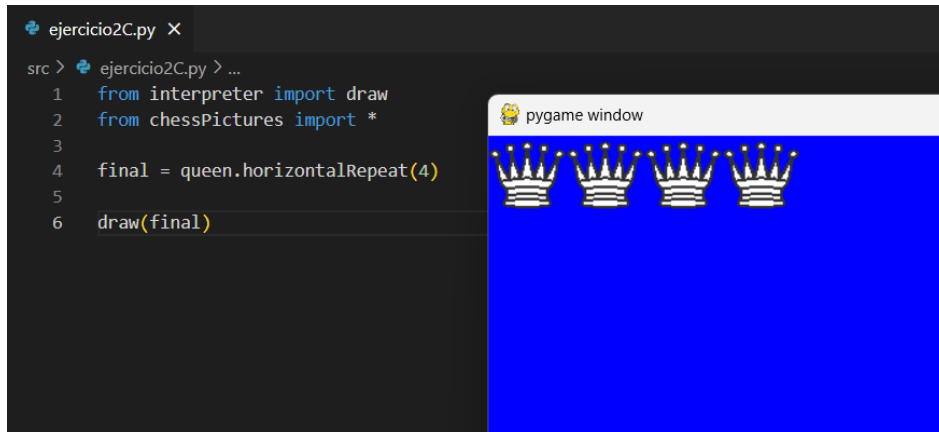
■ EJERCICIO 2 - C

Listing 4: EJERCICIO 2-C

```
from interpreter import draw
from chessPictures import *

final = queen.horizontalRepeat(4)

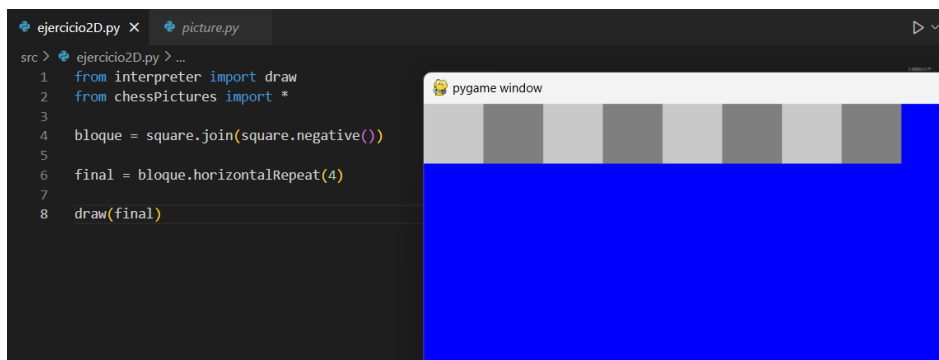
draw(final)
```



■ EJERCICIO 2 - D

Listing 5: EJERCICIO 2-D

```
from interpreter import draw  
from chessPictures import *  
  
bloque = square.join(square.negative())  
  
final = bloque.horizontalRepeat(4)  
  
draw(final)
```



■ EJERCICIO 2 - E

Listing 6: EJERCICIO 2-E

```
from interpreter import draw  
from chessPictures import *  
  
bloque = square.join(square.negative())  
  
final = bloque.horizontalRepeat(4)  
  
draw(final.negative())
```



```

ejercicio2D.py  ejercicio2E.py X
src > ejercicio2E.py > ...
1 from interpreter import draw
2 from chessPictures import *
3
4 bloque = square.join(square.negative())
5
6 final = bloque.horizontalRepeat(4)
7
8 draw(final.negative())
  
```

■ EJERCICIO 2 - F

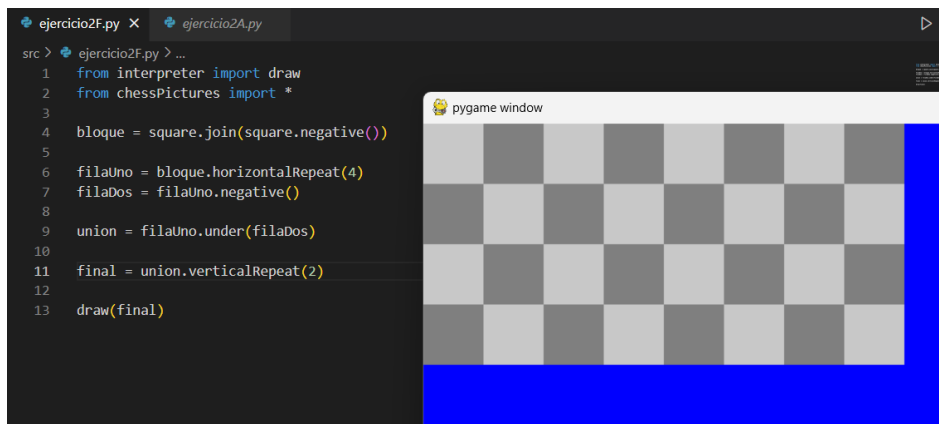
Listing 7: EJERCICIO 2-F

```

from interpreter import draw
from chessPictures import *

bloque = square.join(square.negative())
filaUno = bloque.horizontalRepeat(4)
filaDos = filaUno.negative()
union = filaUno.under(filaDos)
final = union.verticalRepeat(2)

draw(final)
  
```



```

ejercicio2F.py X  ejercicio2A.py
src > ejercicio2F.py > ...
1 from interpreter import draw
2 from chessPictures import *
3
4 bloque = square.join(square.negative())
5
6 filaUno = bloque.horizontalRepeat(4)
7 filaDos = filaUno.negative()
8
9 union = filaUno.under(filaDos)
10
11 final = union.verticalRepeat(2)
12
13 draw(final)
  
```

■ EJERCICIO 2 - G

Listing 8: EJERCICIO 2-G

```

from interpreter import draw
from chessPictures import *

#Bloques

bloqueB = square
bloqueN = square.negative()

# Fila de Peones

bloquePeones = pawn.up(square).join(pawn.up(square.negative()))
filaPeonesBlancos = bloquePeones.horizontalRepeat(4)
  
```

```

filaPeonesNegros = filaPeonesBlancos.negative()

#Parte del medio (bloques solos)

bloque = square.join(square.negative())
filaUno = bloque.horizontalRepeat(4)
filaDos = filaUno.negative()
union = filaUno.under(filaDos)
finalCuadrados = union.verticalRepeat(2)

#Parte de Piezas importantes

torre = rock.up(bloqueN)
caballo = knight.up(bloqueB)
alfil = bishop.up(bloqueN)
reyna = queen.up(bloqueB)
rey = king.up(bloqueN)
alfil2 = bishop.up(bloqueB)
caballo2 = knight.up(bloqueN)
torre2 = rock.up(bloqueB)

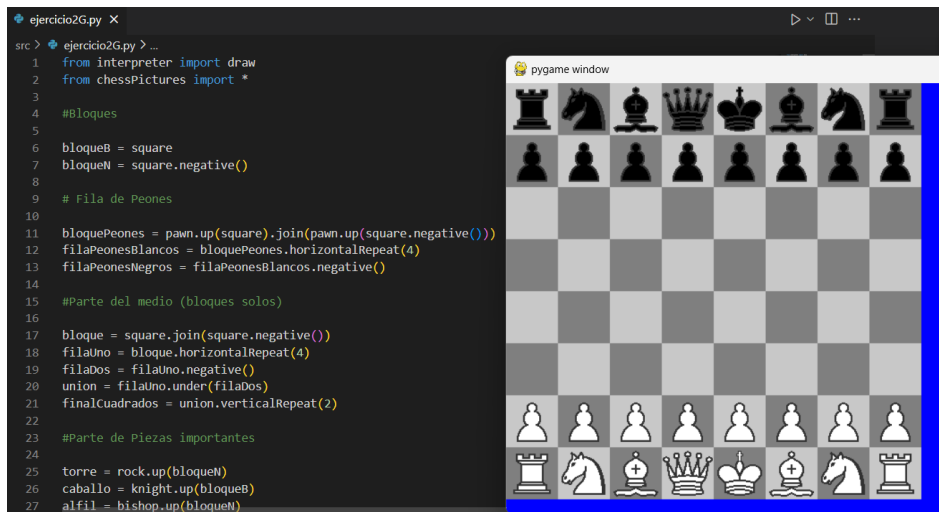
filaPiezas = torre.join(caballo).join(alfil).join(reyna).join(rey).join(alfil2)
              .join(caballo2).join(torre2)
filaPiezasNegras = filaPiezas.negative()

#Combinacion

final = filaPiezasNegras.under(filaPeonesNegros).under(finalCuadrados)
        .under(filaPeonesBlancos).under(filaPiezas)

draw(final)

```



The screenshot shows a PyGame window titled "pygame window" displaying a chessboard. The board has a blue border and a checkered pattern. The pieces are arranged as follows:

- Row 1 (top): Black King, Black Queen, Black Bishop, Black Knight, Black Pawn, Black Pawn, Black Pawn, Black Pawn.
- Row 2: Black Pawn, Black Pawn, Black Pawn, Black Pawn, Black Pawn, Black Pawn, Black Pawn, Black Pawn.
- Row 3: Empty.
- Row 4: Empty.
- Row 5: White Pawn, White Pawn, White Pawn, White Pawn, White Pawn, White Pawn, White Pawn, White Pawn.
- Row 6 (bottom): White King, White Queen, White Bishop, White Knight, White Pawn, White Pawn, White Pawn, White Pawn.

The code in the text editor on the left is as follows:

```

ejercicio2G.py X
src > ejercicio2G.py > ...
1  from interpreter import draw
2  from chessPictures import *
3
4  #Bloques
5
6  bloqueB = square
7  bloqueN = square.negative()
8
9  # Fila de Peones
10
11 bloquePeones = pawn.up(square).join(pawn.up(square.negative()))
12 filaPeonesBlancos = bloquePeones.horizontalRepeat(4)
13 filaPeonesNegros = filaPeonesBlancos.negative()
14
15 #Parte del medio (bloques solos)
16
17 bloque = square.join(square.negative())
18 filaUno = bloque.horizontalRepeat(4)
19 filaDos = filaUno.negative()
20 union = filaUno.under(filaDos)
21 finalCuadrados = union.verticalRepeat(2)
22
23 #Parte de Piezas importantes
24
25 torre = rock.up(bloqueN)
26 caballo = knight.up(bloqueB)
27 alfil = bishop.up(bloqueN)

```

5. Rúbricas

5.1. Entregable Informe

Tabla 1: Tipo de Informe

Informe	
Latex	El informe está en formato PDF desde Latex, con un formato limpio (buena presentación) y fácil de leer.

5.2. Rúbrica para el contenido del Informe y demostración

- El alumno debe marcar o dejar en blanco en celdas de la columna **Checklist** si cumple con el ítem correspondiente.
- Si un alumno supera la fecha de entrega, su calificación será sobre la nota mínima aprobada, siempre y cuando cumpla con todos los ítems.
- El alumno debe autocalificarse en la columna **Estudiante** de acuerdo a la siguiente tabla:

Tabla 2: Niveles de desempeño

Puntos	Nivel			
	Insatisfactorio 25 %	En Proceso 50 %	Satisfactorio 75 %	Sobresaliente 100 %
2.0	0.5	1.0	1.5	2.0
4.0	1.0	2.0	3.0	4.0

Tabla 3: Rúbrica para contenido del Informe y demostración

Contenido y demostración		Puntos	Checklist	Estudiante	Profesor
1. GitHub	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	X	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	X	2	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	2	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	1	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	2	
6. Fechas	Las fechas de modificación del código fuente están dentro de los plazos de fecha de entrega establecidos.	2	X	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	2	
8. Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4	X	4	
Total		20		17	