

INFORME DE LABORATORIO

Información básica

Asignatura:	Programación Web 2
Título de práctica:	Python
Número de práctica:	4
Año lectivo:	2023
Integrante:	Roque Quispe William Isaias
Docente:	Carlo Jose Luis Corrales Delgado

Solución y Resultados

Enlace GitHub: <https://github.com/WilliamIsaiasRoque/Lab04-Pweb2.git>

I. Solución de ejercicios/problemas

- Implemente los métodos de la clase Picture. Se recomienda que implemente la clase picture por etapas, probando realizar los dibujos que se muestran en la siguiente preguntas.

```
1  from colors import *
2  class Picture:
3      def __init__(self, img):
4          self.img = img
5
6      def __eq__(self, other):
7          return self.img == other.img
8
9      def _invColor(self, color):
10         if color not in inverter:
11             return color
12         return inverter[color]
13
14     def verticalMirror(self):
15         """ Devuelve el espejo vertical de la imagen """
16         vertical = []
17         for value in self.img:
18             vertical.append(value[::-1])
19         return Picture(vertical)
20
21     def horizontalMirror(self):
22         """ Devuelve el espejo horizontal de la imagen """
23         horizontal = self.img[::-1]
24         return Picture(horizontal)
25
26     def negative(self):
27         """Devuelve un negativo de la imagen"""
28         neg_img = []
29         for row in self.img:
30             rowBlack = ""
31             for pixel in row:
32                 rowBlack += self._invColor(pixel)
33             neg_img.append(rowBlack)
34         return Picture(neg_img)
35
36     def join(self, p):
37         """ Devuelve una nueva figura poniendo la figura del argumento
38             al lado derecho de la figura actual """
39         joinRsult = []
40         i = 0
41         while i < len(self.img):
42             joinRsult.append(self.img[i]+p.img[i])
43             i+=1
44         return Picture(joinRsult)
```

Primera parte de la clase picture.py

```

45
46 def up(self, p):
47     new_img = p.img + self.img
48     return Picture(new_img)
49
50 def under(self, p):
51     """ Devuelve una nueva figura poniendo la figura p sobre la
52     | figura actual """
53     underRslt = []
54     for i in range(len(self.img)):
55         merged_row = ""
56         for j in range(len(self.img)):
57             if p.img[i][j] != self.img[i][j] and p.img[i][j] != " ":
58                 merged_row += p.img[i][j]
59             else:
60                 merged_row += self.img[i][j]
61         underRslt.append(merged_row)
62
63     return Picture(underRslt)
64
65 def horizontalRepeat(self, n):
66     """ Devuelve una nueva figura repitiendo la figura actual al costado
67     | la cantidad de veces que indique el valor de n """
68     repeated_img = [row * n for row in self.img]
69     return Picture(repeated_img)
70
71 def verticalRepeat(self, n):
72     """ Devuelve una nueva figura repitiendo la figura actual hacia abajo
73     | la cantidad de veces que indique el valor de n """
74     repeated_img = self.img * n
75     return Picture(repeated_img)
76
77 #Extra: Sólo para realmente viciados
78 def rotate(self):
79     """Devuelve una figura rotada en 90 grados, puede ser en sentido horario
80     o antihorario"""
81     return Picture(None)
82
83

```

Segunda parte de la clase picture.py

- Usando únicamente los métodos de los objetos de la clase Picture dibuje las siguientes figuras (invoque a draw):

```

1  from interpreter import draw
2  from chessPictures import *
3
4  horse=knight
5  horseblack=horse.negative()
6  horse4=horseblack.join(horse).up((horse).join(horseblack))
7  draw(horse4)

```

Ejercicio2a código



Ejecución de Ejercicio2a

```
1 from interpreter import draw
2 from chessPictures import *
3
4 horse=knight
5 horseblack=horse.negative()
6 horse4mirror=horse.join(horseblack).verticalMirror().up(horse.join(horseblack))
7 draw(horse4mirror)
```

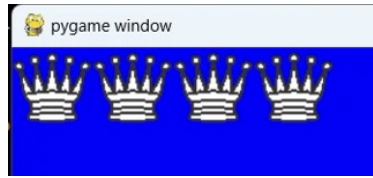
Ejercicio2b código



Ejecución de Ejercicio2b

```
1 from interpreter import draw
2 from chessPictures import *
3
4 queenx4= queen.horizontalRepeat(4)
5 draw(queenx4)
```

Ejercicio2c código



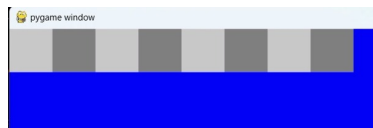
Ejecución de Ejercicio2c

```

1  from interpreter import draw
2  from chessPictures import *
3
4  squarenegative = square.negative()
5  fila = square.join(squarenegative).horizontalRepeat(4)
6  draw(fila)

```

Ejercicio2d código



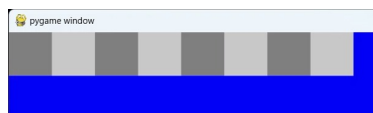
Ejecución de Ejercicio2d

```

1  from interpreter import draw
2  from chessPictures import *
3
4  squarenegative = square.negative()
5  fila = squarenegative.join(square).horizontalRepeat(4)
6  draw(fila)

```

Ejercicio2e código



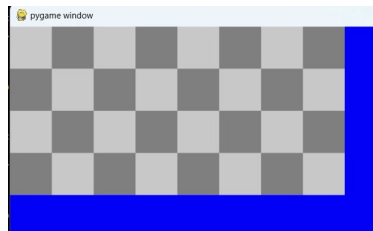
Ejecución de Ejercicio2e

```

1  from interpreter import draw
2  from chessPictures import *
3
4  squarenegative = square.negative()
5  fila = square.join(squarenegative).horizontalRepeat(4)
6  filax4=fila.negative().up(fila).verticalRepeat(2)
7  draw(filax4)

```

Ejercicio2f código



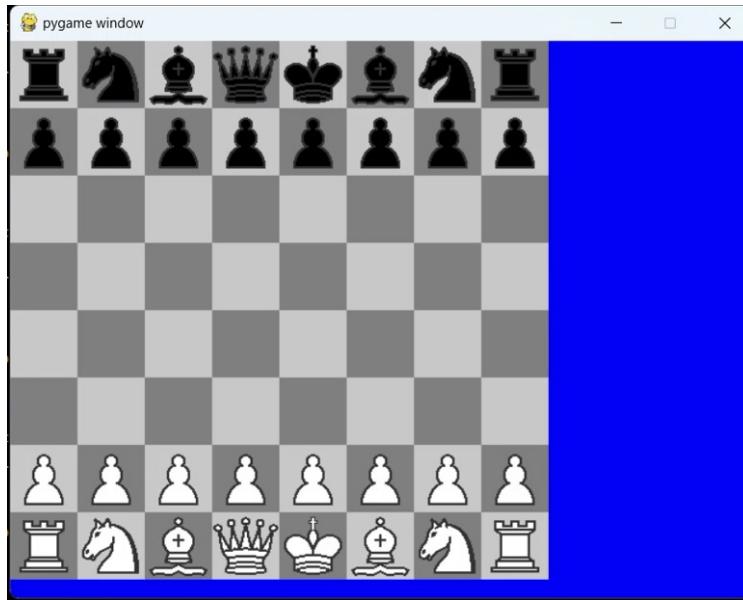
Ejecución de Ejercicio2f

```

1  from interpreter import draw
2  from chessPictures import *
3
4  squareN=square.negative()
5  #Reutilizando la filax4 para imprimir squares vacios para el tablero
6  fila = square.join(squareN).horizontalRepeat(4)
7  filax4=fila.negative().up(fila).verticalRepeat(2)
8  #Creando "attack" este contiene el torre,caballo,alfil,rey,reina con su square
9  attack = squareN.under(rock)
10 attack = attack.join(square.under(knight))
11 attack = attack.join(squareN.under(bishop))
12 attack = attack.join(square.under(queen))
13 attack = attack.join(squareN.under(king))
14 attack = attack.join(square.under(bishop))
15 attack = attack.join(squareN.under(knight))
16 attack = attack.join(square.under(rock))
17 #creando "pawns" este contiene los 8 peones con su respectivo square usando while
18 pawns = square.under(pawn)
19 i=0
20 while(i<7):
21     if i%2==0:
22         pawns = pawns.join(squareN.under(pawn))
23     else:
24         pawns = pawns.join(square.under(pawn))
25     i+=1
26
27 #creando los equipos
28 whitePawns = pawns
29 whiteAttack = attack
30 whiteTeam = whiteAttack.up(whitePawns)
31 blackTeam= whitePawns.negative().up(whiteAttack.negative())
32 #creando chesstable y agregando whiteTeam, filax4 y blackTeam
33 chessTable = whiteTeam.up(filax4).up(blackTeam)
34 draw(chessTable)

```

Ejercicio2g código



Ejecución de Ejercicio2g

Commits		
main		
Commits on Jun 8, 2023		
Picture.py con todos los métodos funcionales	WilliamIsaíasRoque committed 6 hours	55bc6d4
Commits on Jun 7, 2023		
Ejecución completa	WilliamIsaíasRoque committed 7 hours ago	fFba8d8
Ejercicio 2g funcional, imprime el tablero completo	WilliamIsaíasRoque committed 7 hours ago	c4662ec
Probando impresión de piezas con sus square	WilliamIsaíasRoque committed 7 hours ago	18d52d9
Commits on Jun 5, 2023		
Probando el ejercicio 2g	WilliamIsaíasRoque committed 3 days ago	c952c4f
Ejercicio 2f funcional	WilliamIsaíasRoque committed 3 days ago	ed9d88a
Ejercicio 2e funcional usando de referencia el ejercicio 2d	WilliamIsaíasRoque committed 3 days ago	b5e54c8
Ejercicio 2d funcional con squares	WilliamIsaíasRoque committed 3 days ago	f4ff6a2
Ejercicio 2c finalizado	WilliamIsaíasRoque committed 3 days ago	4e9b4cb
Ejercicio 2b finalizado	WilliamIsaíasRoque committed 3 days ago	fa4e232
Ejercicio 2a modificación y finalizado	WilliamIsaíasRoque committed 3 days ago	8d2e4c1
Modificación en todos los métodos sin under	WilliamIsaíasRoque committed 3 days ago	49b938c

Historial de commits 1

Commits on Jun 2, 2023		
Ejercicio 2a finalizado	WilliamIsaiaRoque committed 5 days ago	7cb1fef
Todos los métodos funcionales	WilliamIsaiaRoque committed 5 days ago	4c94356
Modificando el método join para que acepte objetos Picture	WilliamIsaiaRoque committed last week	6555472
Commits on Jun 1, 2023		
negative funcional	WilliamIsaiaRoque committed last week	7bbf3f1
horizontalMirror funcional	WilliamIsaiaRoque committed last week	6da2ce6
Probando primer ejercicio	WilliamIsaiaRoque committed last week	edab8eb
Delete a.py	WilliamIsaiaRoque committed last week	Verified be7cfff4
Citignore agregado al proyecto	WilliamIsaiaRoque committed last week	3f4b505
Probando pygame	WilliamIsaiaRoque committed last week	488924a
Commits on May 31, 2023		
Agregando el entorno virtual	WilliamIsaiaRoque committed last week	6444096
Commits on May 30, 2023		
Ejercicios del repositorio forma base	WilliamIsaiaRoque committed last week	88d888d
Commits on May 25, 2023		
interpreter	WilliamIsaiaRoque committed 2 weeks ago	aa01a72
Archivos del docente subidos	WilliamIsaiaRoque committed 2 weeks ago	f591e57
Initial commit	WilliamIsaiaRoque committed 2 weeks ago	Verified edbf28c

Historial de commits 2

II. Solución del cuestionario

- ¿Qué son los archivos *.pyc?
 - Son archivos de código bytecode compilado en Python, se generan automáticamente por el interpreter cuando se ejecuta un código “.py”. Esto permite una ejecución más veloz del código, ya que se traduce a un formato más eficiente para el interpreter y se puede reutilizar para futuras ejecuciones.
- ¿Para qué sirve el directorio pycache?
 - Es un directorio creado automáticamente como caché para mejorar el rendimiento al ejecutar un programa o un conjunto de módulos de determinado proyecto.
- ¿Cuáles son los usos y lo que representa el subguión en Python?
 - Se puede representar como una variable temporal en un ciclo. También es usado como referencia para indicar que una variable no se está utilizando o no se le da importancia en una línea de código. Es usado además para nombrar variables que lleven más de dos palabras.

Conclusiones

- Pygame posee una gran biblioteca para el desarrollo de videojuegos 2D que es utilizado un Python, un lenguaje de programación con una sintaxis legible y fácil de entender.
- Python tiene una comunidad que frecuentemente aporta con nuevo conocimiento, recursos y documentación para Pygame.

Referencias y bibliografía

- https://www.overleaf.com/learn/latex/Choosing_a_LaTeX_Compiler
- <https://www.w3schools.com/python/>
- <https://www.pygame.org/news>