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.

```python
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):
    """ Devuelve el espejo vertical de la imagen """
    vertical = []
    for value in self.img:
      vertical.append(value[::-1])
    return Picture(vertical)

  def horizontalMirror(self):
    """ Devuelve el espejo horizontal de la imagen """
    horizontal = self.img[::-1]
    return Picture(horizontal)

  def negative(self):
    """Devuelve un negativo de la imagen"""
    neg_img = []
    for row in self.img:
      rowBlack = ""
      for pixel in row:
        rowBlack += self._invColor(pixel)
      neg_img.append(rowBlack)
    return Picture(neg_img)

  def join(self, p):
    """ Devuelve una nueva figura poniendo la figura del argumento
        al lado derecho de la figura actual """
    joinRslt = []
    i = 0
    while i<len(self.img):
      joinRslt.append(self.img[i]+p.img[i])
      i+=1
    return Picture(joinRslt)
```

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 viciosos
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

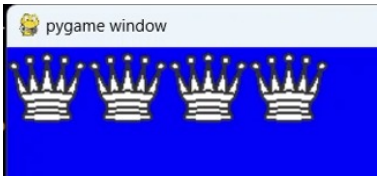
```
from interpreter import draw
from chessPictures import *

horse=knight
horseblack=horse.negative()
horse4mirror=horse.join(horseblack).verticalMirror().up(horse.join(horseblack))
draw(horse4mirror)
```

Ejercicio2b código

Ejecución de Ejercicio2b

```
from interpreter import draw
from chessPictures import *

queenx4= queen.horizontalRepeat(4)
draw(queenx4)
```

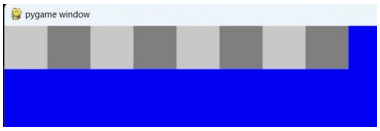
Ejercicio2c código



Ejecución de Ejercicio2c

```
from interpreter import draw
from chessPictures import *

squarenegative = square.negative()
fila = square.join(squarenegative).horizontalRepeat(4)
draw(fila)
```

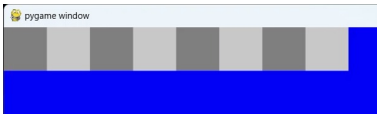
Ejercicio2d código



Ejecución de Ejercicio2d

```
from interpreter import draw
from chessPictures import *

squarenegative = square.negative()
fila = squarenegative.join(square).horizontalRepeat(4)
draw(fila)
```

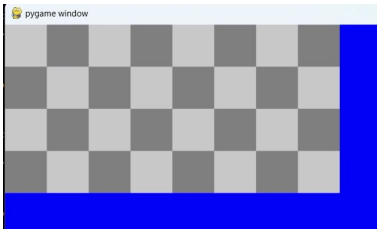
Ejercicio2e código



Ejecución de Ejercicio2e

```python
from interpreter import draw
from chessPictures import *

squarenegative = square.negative()
fila = square.join(squarenegative).horizontalRepeat(4)
filax4=fila.negative().up(fila).verticalRepeat(2)
draw(filax4)
```

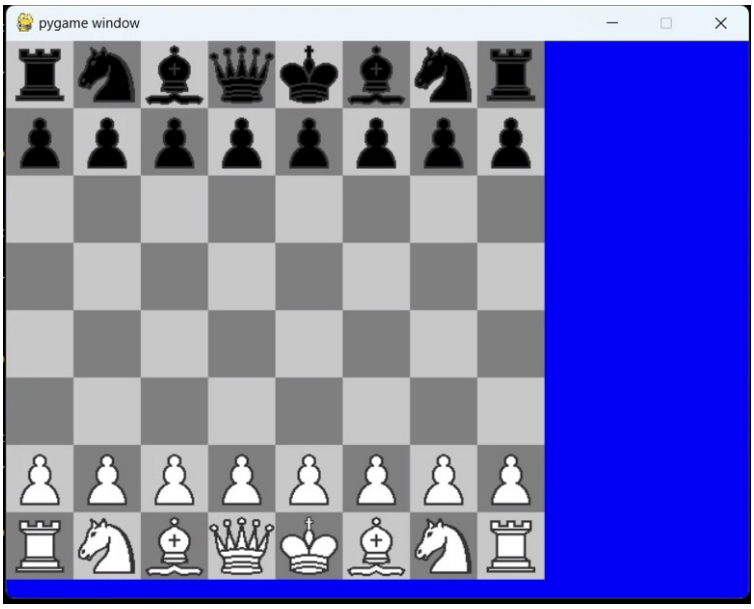
Ejercicio2f código



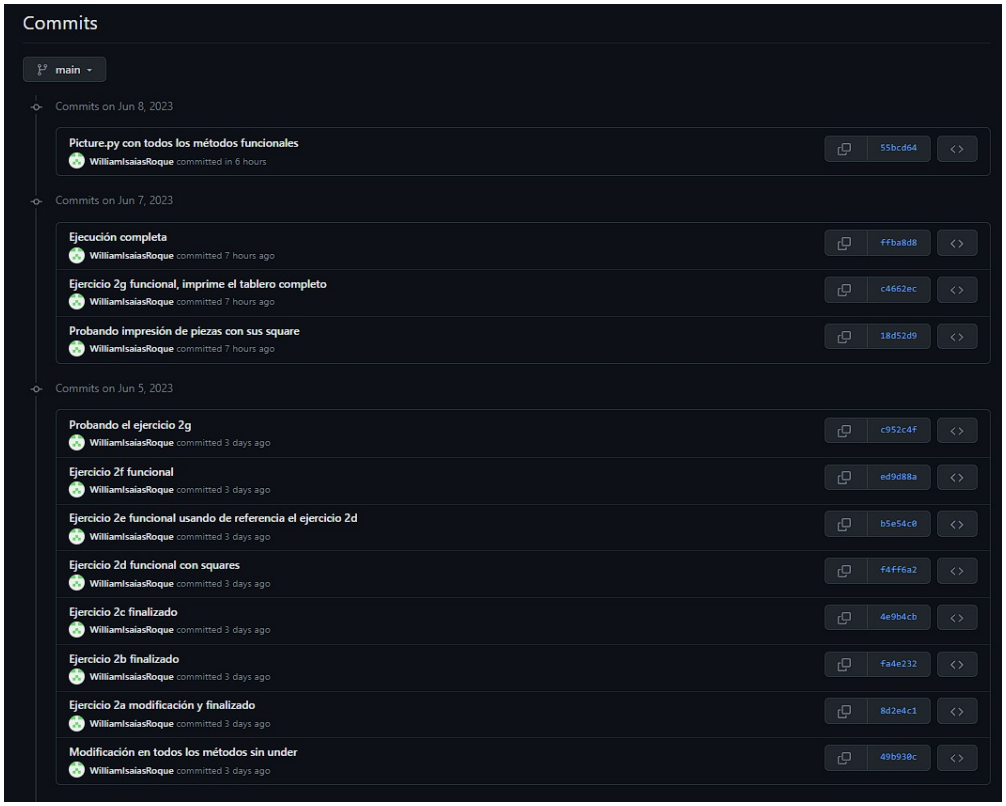
Ejecución de Ejercicio2f

```python
from interpreter import draw
from chessPictures import *

squareN=square.negative()
#Reutilizando la filax4 para imprimir squares vacios para el tablero
fila = square.join(squareN).horizontalRepeat(4)
filax4=fila.negative().up(fila).verticalRepeat(2)
#Creando "attack" este contiene el torre,caballo,alfil,rey,reina con su square
attack = squareN.under(rock)
attack = attack.join(square.under(knight))
attack = attack.join(squareN.under(bishop))
attack = attack.join(square.under(queen))
attack = attack.join(squareN.under(king))
attack = attack.join(square.under(bishop))
attack = attack.join(squareN.under(knight))
attack = attack.join(square.under(rock))
#creando "pawns" este contiene los 8 peones con su respectivo square usando while
pawns = square.under(pawn)
i=0
while(i<7):
    if i%2==0:
        pawns = pawns.join(squareN.under(pawn))
    else:
        pawns = pawns.join(square.under(pawn))
    i+=1

#creando los equipos
whitePawns = pawns
whiteAttack = attack
whiteTeam = whiteAttack.up(whitePawns)
blackTeam= whitePawns.negative().up(whiteAttack.negative())
#creando chesstable y agregando whiteTeam, filax4 y blackTeam
chessTable = whiteTeam.up(filax4).up(blackTeam)
draw(chessTable)
```

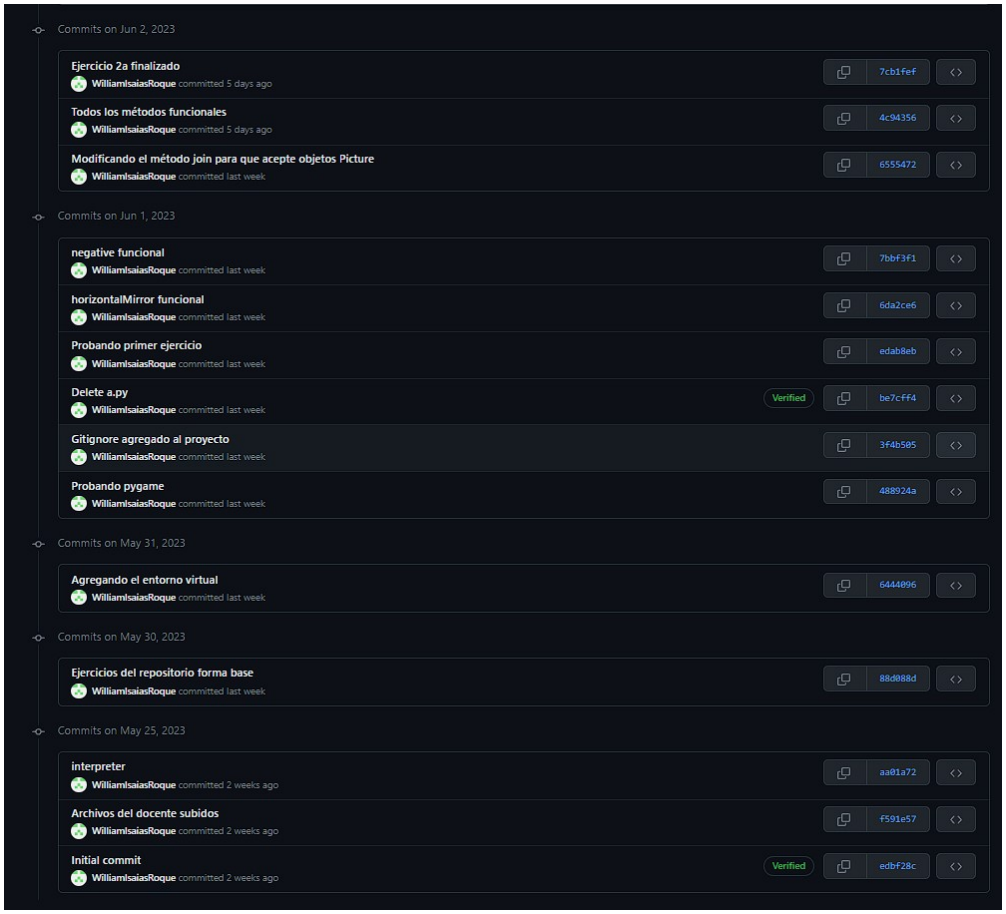
Ejercicio2g código



Ejecución de Ejercicio2g



Historial de commits 1



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