Informe de Laboratorio 04

Tema: Python

Nota

Estudiante	Escuela	Asignatura
Joshep Antony Ccahuana Larota jccahuanala@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Programación Web 2 Semestre: III Código: 20223010

Laboratorio	Tema	Duración
04	Python	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - A	Del 30 mayo 2023	Al 09 junio 2023

1. Tarea

- 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.
- Usando únicamente los métodos de los objetos de la clase Picture, dibuje las figuras (invocando el método "draw()").

2. Equipos, materiales y temas utilizados

- Sistema operativo Windows 11.
- Librería PyGame.
- Python 3.10.11.
- Git 2.39.2.
- Cuenta en GitHub con el correo institucional (Joshep-c).
- TextMaker / MikTex
- Programación Orientada a Objetos.
- Manejo de matrices unidimensionales.

3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- `https://github.com/Joshep-c/PW2LAB.git`
- URL para el laboratorio 04 en el Repositorio GitHub.
- `https://github.com/Joshep-c/PW2LAB/tree/68f2e2f2197a1bb6bfe93bfcbef903c246a52603/Python`

4. Implementación de los métodos

4.1. Método *verticalMirror()*

Se modifica la variable de retorno del método, por un Picture, y ya no un List.

Listing 1: verticalMirror.py

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

4.2. Método *horizontalMirror()*

El siguiente método toma como parámetro un Picture, en la cual recorre de manera inversa la lista "[::-1]", para después almacenarla en otra. Y retornar otro Picture.

Listing 2: horizontalMirror.py

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

4.3. Método *negative()*

En este método el Picture recibido por el parámetro, se recorre cada elemento, y con un sub for cada caracter es cambiado de color mediante el metodo "_invColor()", despues cada el valor del color nuevo es almacenado en conjunto en una nueva lista, que posteriormente es transformada en Picture y se retorna.

Listing 3: negative.py

```
def negative(self):
    """Devuelve un negativo de la imagen"""
    negative_self = []
    for row in self.img:
        value_color = ""
        for value in row:
```

```
            value_color += self._invColor(value)
        negative_self.append(value_color)

    return Picture(negative_self)
```

4.4. Método *join()*

La función de este método era poder fusionar dos imágenes en una, colocando en la derecha la imagen recibida (p). Para lograr esto, se convirtió las imágenes en arrays con el .img para luego agregar ambos en un array new_img.

Listing 4: join.py

```
def join(self, p):
    """Devuelve una nueva figura poniendo la figura del argumento
    al lado derecho de la figura actual"""
    union_self = []
    for iterator in range(len(self.img)):
        union_self.append(self.img[iterator] + (p.img[iterator]))

    return Picture(union_self)
```

4.5. Método *up()*

El funcionamiento del método se basa en la agregación de un nuevo Picture por encima del inicial, como los Picture.img, son listas se puede lograr con una simple agregación con "+".

Listing 5: up.py

```
def up(self, p):
    up_picture = p.img + self.img
    return Picture(up_picture)
```

4.6. Método *under()*

De igual manera que el anterior método, pero invirtiendo los dos picture, para que el picture "p" quede por debajo del instanciado.

Listing 6: under.py

```
def under(self, p):
    """Devuelve una nueva figura poniendo la figura p sobre la
    figura actual"""
    under_picture = self.img + p.img
    return Picture(under_picture)
```

4.7. Método *horizontalRepeat()*

Haciendo uso de un bucle for, modificado para recorrer mediante una numeración ya que el parámetro ingresado es un entero, se va repitiendo la misma funcionalidad del método join(), agregando como argumento el mismo Picture instanciado al momento de su llamado.

Listing 7: horizontalRepeat.py

```
def horizontalRepeat(self, n):
    """Devuelve una nueva figura repitiendo la figura actual al costado
    la cantidad de veces que indique el valor de n"""
    hori_rep = self

    for iterator in range(n-1):
        hori_rep = hori_rep.join(self)

    return hori_rep
```

4.8. Método *verticalRepeat()*

El método tiene la misma base de funcionamiento que el anterior, pero como se busca la repetición de un Picture de manera vertical, se hace uso del método "up()" implementado anteriormente.

Listing 8: verticalRepeat.py

```
def verticalRepeat(self, n):

    vert_rep = self

    for iterator in range(n-1):
        vert_rep = vert_rep.up(self)

    return vert_rep
```

4.9. Método *rotate()*

Este método cambia las posiciones de la lista .img del Picture, ósea obteniendo cada carácter en el eje "y", se coloca de manera que esté en el eje "x" en una nueva lista, que posteriormente se retorna como Picture.

Listing 9: rotate.py

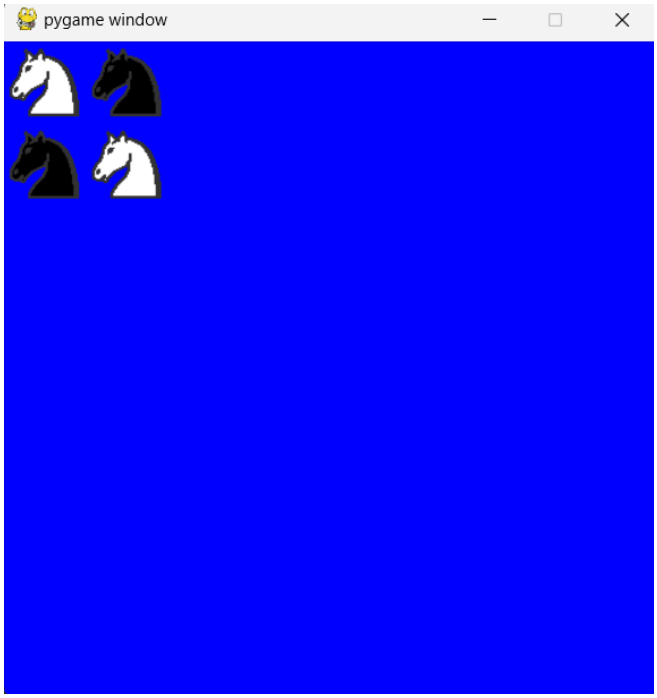
```
def rotate(self):
    """Devuelve una figura rotada en 90 grados, puede ser en sentido horario
    o antihorario"""
    self_rotate = []
    for i in range(len(self.img)):
        row = ""
        for j in range(len(self.img[i])):
            row += self.img[j][i]
        self_rotate.append(row)

    return Picture(self_rotate)
```

5. Ejecuciones y código

5.1. Ejercicio A:

En el siguiente código se crea las dos primeras líneas por separado, la primera generando un Picture con un caballo negro que se coloco al costado e un caballo blanco, de manera contaaria la segunda fila, y finalmente se coloca uno debajo del otro con el método under.

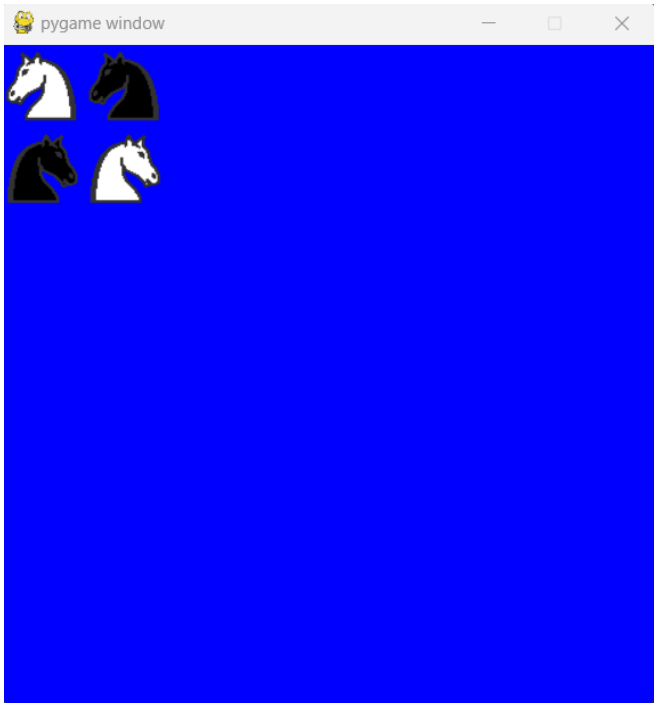
```
1  from interpreter import draw
2  from chessPictures import *
3
4  fil = knight.join(knight.negative())
5  fil2 = knight.negative().join(knight)
6
7  draw(fil.under(fil2))
```



5.2. Ejercicio B:

En este ejercicio, se inicia con la primera linea de caballos, se inserta en un caballo negro al lado derecho de un caballo negro, despues se obtiene un Picture con el método verticalMirror. Después el primer picture es colocado por encima del segundo Picture, con el método under.

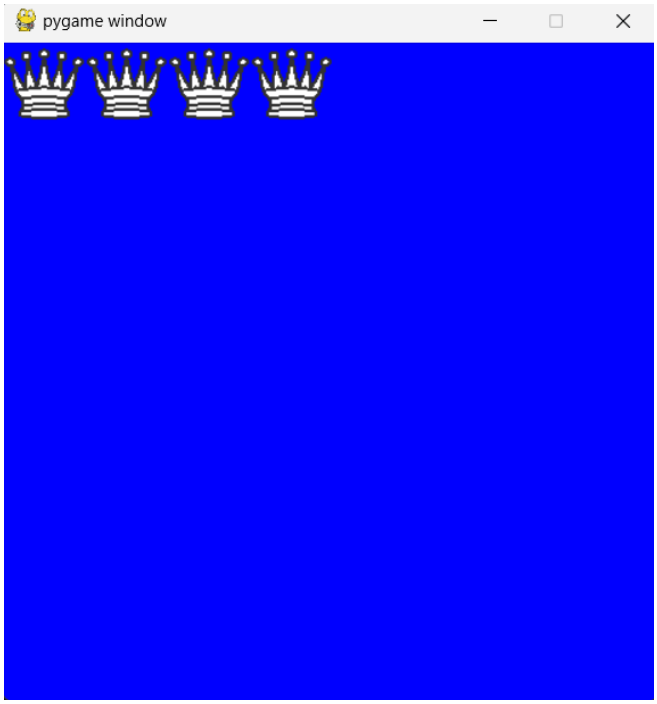
```
1  from interpreter import draw
2  from chessPictures import *
3
4  fil = knight.join(knight.negative())
5  fil2 = fil.verticalMirror()
6
7  draw(fil.under(fil2))
```



5.3. Ejercicio C:

Para este ejercicio se usa una pieza queen se repite mediante el método horizontalRepeat con un argumento 4, para que se repita cuatro veces y que se dibuje en la ventana.

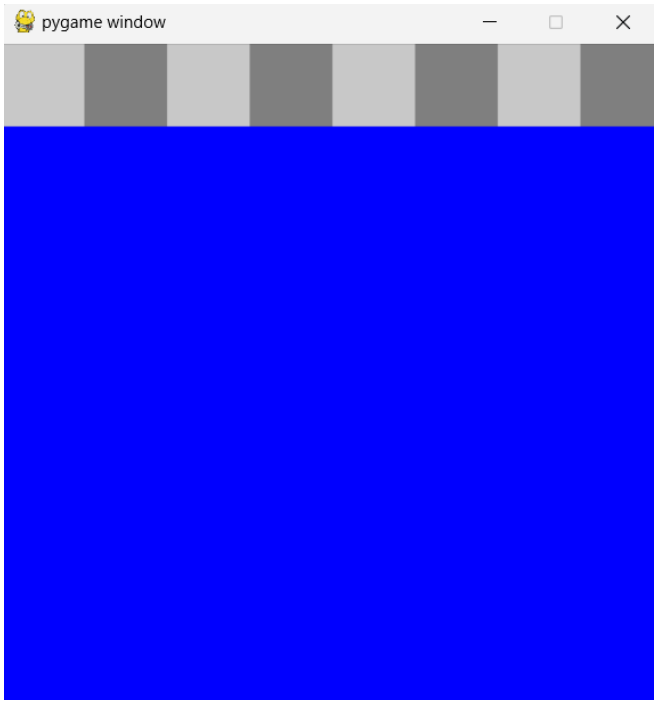
```
1  from interpreter import draw
2  from chessPictures import *
3
4  fil = queen.horizontalRepeat(4)
5
6  draw(fil)
```



5.4. Ejercicio D:

En el ejercicio D, primero se coloca un Picture square.negative al lado derecho de un square, para después repetir este Picture 4 veces con el método horizontalRepeat.

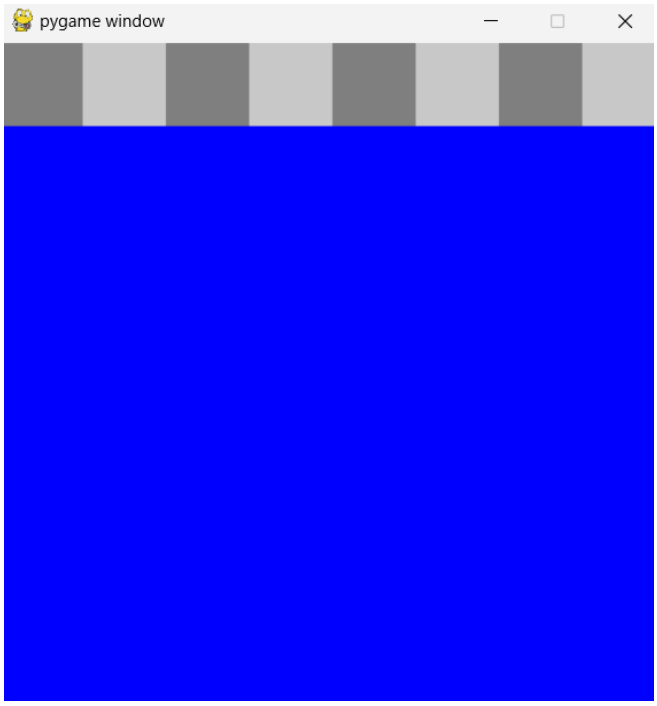
```
1  from interpreter import draw
2  from chessPictures import *
3
4  fil = square.join(square.negative()).horizontalRepeat(4)
5
6  draw(fil)
```



5.5. Ejercicio E:

De igual manera con el anterior ejercicio, pero invirtiendo las posiciones de las dos primeras square.

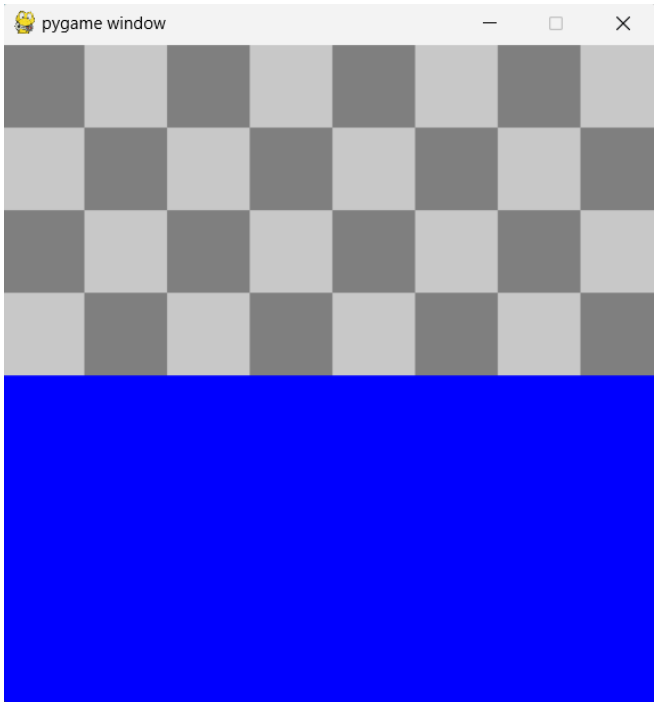
```
1  from interpreter import draw
2  from chessPictures import *
3
4  fil = square.negative().join(square).horizontalRepeat(4)
5
6  draw(fil)
```



5.6. Ejercicio F:

Utilizando los dos ejercicios anteriores, pero colocando uno debajo de otro con el método under, y duplicando de manera vertical este último Picture generado, con el método under con el argumento de si mismo.

```
1  from interpreter import draw
2  from chessPictures import *
3
4  fil1 = square.negative().join(square).horizontalRepeat(4)
5  fil2 = square.join(square.negative()).horizontalRepeat(4)
6
7  fil3 = fil1.under(fil2)
8
9  draw(fil3.under(fil3))
```



5.7. Ejercicio G:

Aquí podemos ver las tres funciones que ayudan a la principal en la generación de las piezas y el espacio entre las piezas negras y blancas.

add_pawn: genera una imagen con los 8 peones (pawn) de manera horizontal y de color blanco (por defecto).

negative_pieces: recibe un list por argumento y recorre cada uno de los Picture para volver para modificar cada uno y volverlo negativo, para después almacenarlo en una nueva lista y retorna esta (arr_negative).

add_main_list: esta función tiene como propósito generar una lista con los Pieces principales (con excepción de los peones), para después retornar esta lista (main_list).

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

def add_pawn():
    pawns = pawn.horizontalRepeat(8)
    return pawns

def negative_pieces(arr):
    arr_negative = []
    for i in arr:
        arr_negative.append(i.negative())

    return arr_negative

def add_main_list():
    list_prim = []
    list_prim.append(rock)
    list_prim.append(knight)
    list_prim.append(bishop)

    main_list = copy.deepcopy(list_prim)
    main_list.append(queen)
    main_list.append(king)
    main_list.extend(list(reversed(list_prim)))

    return main_list
```

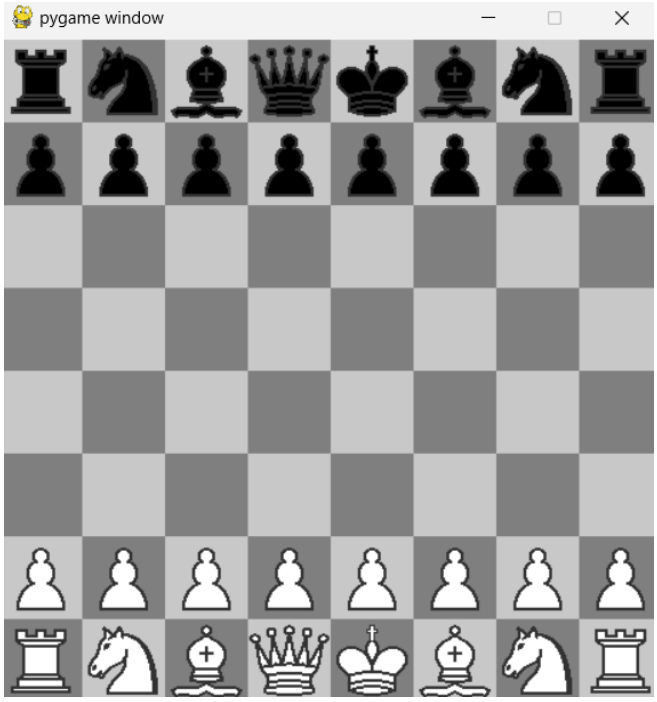
Esta es la función principal “add_pieces” donde se crean un Picture con los peones, una lista con las piezas principales en blanco (método add_main_list), después se crea un Picture y lista nuevos negros usando lo anteriormente creado. Para después mediante un for ingresar cada elemento de la lista de piezas negras a un Picture, los peones, y posteriormente el espacio entre las piezas blancas y negras.

A continuación se crea el Picture general con los peones blancos, se desarrolla un Picture con las piezas blancas, y se acopla al general (pieces), para finalizar el Picture pieces_black se acopla con “pieces”, y se retorna este último.

```
def add_pieces(assignment):

    #Crear un Picture con los peones blancos
    picture_pawns = pawn.horizontalRepeat(8)

    #Crear un lista con las piezas principales en blanco
    main_list = add_main_list()

    #Creacion de una lista con piezas principales y peones negros
    main_list_black = negative_pieces(main_list)
    picture_pawns_black = picture_pawns.negative()


    pieces_black = main_list_black[0]

    for i in range(len(main_list)-1):
        pieces_black = pieces_black.join(main_list_black[i+1])

        if(i == 6):
            pieces_black = pieces_black.under(picture_pawns_black)
            space = square.negative().join(square).horizontalRepeat(4)
            space2 = square.join(square.negative()).horizontalRepeat(4)
            space_vacio = space.under(space2).verticalRepeat(2)
            pieces_black = pieces_black.under(space_vacio)

    pieces = picture_pawns

    pieces_white = main_list[0]
    for i in range(len(main_list)-1):
        pieces_white = pieces_white.join(main_list[i+1])

    pieces = pieces.under(pieces_white)

    pieces = pieces.up(pieces_black)

    return pieces
```

Para finalizar se crea el tablero completo con la misma lógica del ejercicio anterior, y posteriormente se hace un llamado a la función “add_pieces” creándose un Picture pieces, y finalmente se dibujan ambos, en el orden de creación.

```
picture1 = square.negative().join(square).horizontalRepeat(4)
picture2 = square.join(square.negative()).horizontalRepeat(4)

picture_complete = picture1.under(picture2).verticalRepeat(4)

pieces = add_pieces("negative")

draw(picture_complete, pieces)
```



5.8. Estructura de laboratorio 04

- El contenido que se entrega en este laboratorio es el siguiente:

```
Pweb2-Lab03/
|-- Laboratorio 04
|   |-- chessPictures.py
|   |-- colors.py
|   |-- Ejercicio2a.py
|   |-- Ejercicio2b.py
|   |-- Ejercicio2c.py
|   |-- Ejercicio2d.py
|   |-- Ejercicio2e.py
|   |-- Ejercicio2f.py
|   |-- Ejercicio2g.py
|   |-- interpreter.py
|   |-- picture.py
|   |-- pieces.py
|   |-- prac.py
|   |
|   |-- __pycache__
|       |-- chessPictures.cpython-310.pyc
|       |-- colors.cpython-310.pyc
|       |-- interpreter.cpython-310.pyc
|       |-- picture.cpython-310.pyc
|       |-- pieces.cpython-310.pyc
|
|-- Latex
|   |-- Laboratorio_04.aux
|   |-- Laboratorio_04.log
|   |-- Laboratorio_04.out
```

```
|   |-- Laboratorio_04.pdf
|   |-- Laboratorio_04.synctex.gz
|   |-- Laboratorio_04.tex
|   |
|   |-- img
|   |   |-- logo_abet.png
|   |   |-- logo_episunsa.png
|   |   |-- logo_unsa.jpg
|   |   |-- Picture A.png
|   |   |-- Picture B.png
|   |   |-- Picture C.png
|   |   |-- Picture D.png
|   |   |-- Picture E.png
|   |   |-- Picture F.png
|   |   |-- Picture G.png
|   |   |-- Picture H.png
|   |
|   |-- src
|       |-- horizontalMirror.py
|       |-- horizontalRepeat.py
|       |-- join.py
|       |-- negative.py
|       |-- rotate.py
|       |-- under.py
|       |-- up.py
|       |-- verticalRepeat.py
```

5.9. Commits de laboratorio 04

1er commit:

En este commit se copian todos los archivos necesarios para el funcionamiento de la actividad: colors.py, interpreter, etc.

```
commit a1d13914649d08a55b35eaddab22e3bcf85573e5
Author: Joshep-c <jccahuanala@unsa.edu.pe>
Date:   Tue Jun 6 09:27:28 2023 -0500

    Python: reinicio de actividad
```

2do commit:

Implementación del método horizontalMirror en picture.py

```
commit 700eba8dbfc6fa04b9afced61107a6b76bf4ad90
Author: Joshep-c <jccahuanala@unsa.edu.pe>
Date:   Tue Jun 6 10:14:19 2023 -0500

    Funcionalidad de horizontalMirror en picture.py
```

3er commit:

Implementación de los métodos negative, join, up y under, de manera seguida y directa porque necesitaban mucho razonamiento, principalmente de los dos últimos ya que fue simplemente una agregación de listas.

```
commit 639420e07e68f5df04187013496a1708ffd9d50e
Author: Joshep-c <jccahuanala@unsa.edu.pe>
Date:   Tue Jun 6 14:04:50 2023 -0500

    Se agrega funcionalidad a las funciones de negative, join, up y under
```

4to commit:

En este commit se terminó con la implementación de todos los métodos, ósea se completaron los siguientes métodos: horizontalRepeat, verticalRepeat y rotate. De esta manera se completaron los 9 métodos del módulo picture, donde se ubica la clase Picture.

```
commit 13e3c2591dd255f07d2cee815983a433c4eaa3e1
Author: Joshep-c <jccahuanala@unsa.edu.pe>
Date:   Wed Jun 7 16:50:38 2023 -0500

    Completamos todos los metodos de picture.py
```

5to y 6to commit:

Commit de la resolución del ejercicio a, dibujando las 4 piezas de caballo en la ventana del juego. Usando el método draw del módulo interpreter.py.

```
commit 11ede9936cd7d4b1dd280f684efe39e0499da335
Author: Joshep-c <jccahuanala@unsa.edu.pe>
Date:   Wed Jun 7 17:15:24 2023 -0500

    Correccion de ejercicio

commit 81f4dee7d82ae1ce2190b9a11908d6f665a59fe2
Author: Joshep-c <jccahuanala@unsa.edu.pe>
Date:   Wed Jun 7 17:09:00 2023 -0500

    Ejercicio 1 resuelto
```

7mo commit:

Commit de la resolución del ejercicio a, dibujando las 4 piezas de caballo en la ventana del juego.

```
commit bde73452e017eee1a95b6d8fe6f1f72922b4a063
Author: Joshep-c <jccahuanala@unsa.edu.pe>
Date:   Wed Jun 7 17:16:47 2023 -0500

    Resolucion ejercicio b
```

8vo commit:

Se guardó la resolución del ejercicio 3, del dibujo de las piezas “queen” cuatro veces en horizontal.

```
commit 17beba03e4525bb0333cc956b51fc7d240dadeac
Author: Joshep-c <jccahuanala@unsa.edu.pe>
Date:   Wed Jun 7 17:18:56 2023 -0500

    Ejercicio c resuelto
```

9no commit:

Culminación del ejercicio d, en la cual se dibuja la primera línea del tablero, intercalando “square” de diferentes colores.

```
commit 349f6ce1ccf99dec962831bb8f3d06de46b99956
Author: Joshep-c <jccahuanala@unsa.edu.pe>
Date:   Wed Jun 7 17:20:51 2023 -0500

    Ejercicio 4 resuelto
```

10mo commit:

Se guarda el código para el ejercicio e, lo mismo que el anterior pero invertido.

```
commit 92989983b2d4849a80344a6a218895dec021a741
Author: Joshep-c <jccahuanala@unsa.edu.pe>
Date:   Wed Jun 7 17:22:29 2023 -0500

    Ejercicio 5 resuelto
```

11vo commit:

Para este commit se guardó el ejercicio F, el cual consiste en mezclar las dos filas creadas en los anteriores ejercicios y duplicarlas hacia abajo de manera intercalada.

```
commit 2660cd4ec7b0309cb45bb1fafe9e70e622944faa
Author: Joshep-c <jccahuanala@unsa.edu.pe>
Date:   Wed Jun 7 17:26:58 2023 -0500

    Ejercicio 6 resuelto
```

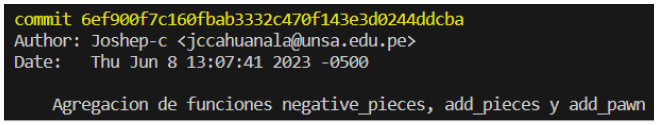
12vo commit:

Modificación de los módulos colors.py e interpreter.py, el primero para crear una transparencia donde anteriormente se tornaba de azul en “ “ (los espacios), y el segundo para poder ingresar más de un argumento al método draw del módulo interpreter.py.

```
commit a296181823a2cc01a70e983488ae388150734a33
Author: Joshep-c <jccahuanala@unsa.edu.pe>
Date:   Thu Jun 8 12:24:00 2023 -0500

    Modificaciion de los archivos colors.py y interpreterpy!
```

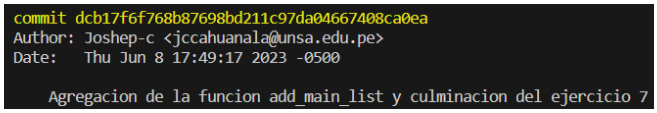
13vo commit:

Se agregan las funciones negative_pieces, add_pieces y add_pawn. La primera para recibir una lista de Picture y mediante un for, cambiando el color por su perspectivismo negativo de cada elemento, por el método negative; el segundo es una función general para crear las piezas del tablero además de crear los espacios "square" entre las fichas blancas y negras; y el tercero para crear y retornar los peones o pawns. Además de modificar el tipo de retorno del método verticalMirror del módulo picture.py.



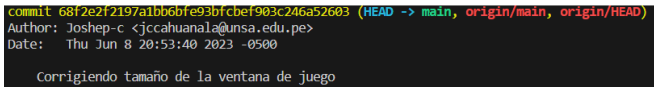
14vo commit:

En este penúltimo commit se guarda la agregación de la función add_main_list, la cual retorna un arreglo con las piezas principales del ajedrez, excepto los peones, y de color blanco, osea por defecto.



15vo commit:

En este último commit se modificó el tamaño de la ventana para que se ajuste a las dimensiones del tablero en sí, ya que aparecía espacio sobrante de color azul, osea con espacios. Además se estaban modificando algunos detalles y nombres de variables en los diferentes módulos y funciones.



6. Cuestionario:

¿Qué son los archivos *.pyc?

Los archivos ".pyc"son archivos de código compilado en Python que se generan automáticamente al ejecutar o importar un archivo ".py". Contienen el código fuente compilado a un formato más eficiente para mejorar el rendimiento en ejecuciones futuras del programa. Estos archivos son específicos de la versión y configuración del intérprete de Python utilizado y pueden volverse obsoletos si cambias la versión o ciertas configuraciones.

¿Para qué sirve el directorio pycache?

El directorio "pycache.[es] una carpeta especial que se crea automáticamente cuando ejecutas o importas archivos de Python. En esta carpeta se guardan los archivos que contienen el código compilado del programa para que se ejecuten más rápidamente en el futuro. El directorio "pycache.[a]yuda a organizar y separar estos archivos compilados de los archivos fuente, facilitando el mantenimiento del proyecto y mejorando el rendimiento de tu programa.

¿Cuáles son los usos y lo que representa el subguión en Python?

El subguión (_) en Python tiene varios usos y significados adicionales además de ser utilizado en constructores y para indicar métodos privados de una clase (como en la clase Picture). Algunos de estos usos incluyen:

- **Traducción de cadenas:** En la función str.maketrans() o en el método str.translate(), puedes utilizar el subguión para indicar que un carácter debe ser eliminado o dejado sin cambios durante una operación de traducción de cadenas.

Listing 10: Ejemplo

```
translation_table = str.maketrans("aeiou","_____")
resultado = "Hola, mundo!".translate(translation_table)
print(resultado)

Ouput >> H_l_, m_nd_!
```

- **Ignorar valores en desempaquetado:** Al desempaquetar una secuencia o iterador, puedes utilizar el subguión para ignorar los valores que no necesitas.

Listing 11: Ejemplo

```
a, _, c = (1, 2, 3)
```

7. Rúbricas

7.1. Entregable Informe

Tabla 1: Tipo de Informe

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

7.2. Rúbrica para el contenido del Informe y demostración

- El alumno debe marcar o dejar en blanco en celdas de la columna **Checklist** si cumplio 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 lo items.
- El alumno debe autocalificarse en la columna **Estudiante** de acuerdo a la siguiente tabla:

Tabla 2: Niveles de desempeño

	Nivel			
Puntos	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	4	
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	2	
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 estan 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		20	

8. Referencias

- https://www.w3schools.com/java/default.asp
- https://www.geeksforgeeks.org/insertion-sort/
- https://github.com/Joshep-c/PW2LAB.git