# Informe de Laboratorio 04
## Tema: Python

| Nota |
|---|
| |

| Estudiante | Escuela | Asignatura |
|---|---|---|
| Klismann Chancuaña Alvis<br>kchancuana@unsa.edu.pe | Escuela Profesional de Ingeniería de Sistemas | PWeb2<br>Semestre: I<br>Código: 20224231 |

| Laboratorio | Tema | Duración |
|---|---|---|
| 04 | Python | 04 horas |

| Semestre académico | Fecha de inicio | Fecha de entrega |
|---|---|---|
| 2023 - A | Del 31 mayo 2023 | Al 8 junio 2023 |

## 1. Tarea

- En esta tarea usted pondrá en práctica sus conocimientos de programación en Python para dibujar un tablero de Ajedrez.
- La parte gráfica ya está programada, usted sólo tendrá que concentrarse en las estructuras de datos subyacentes.
- Con el código proporcionado usted dispondrá de varios objetos de tipo Picture para poder realizar su tarea:
- Estos objetos estarán disponibles importando la biblioteca: chessPictures y estarán internamente representados con arreglos de strings que podrá revisar en el archivo pieces.py
- La clase Picture tiene un sólo atributo: el arreglo de strings img, el cual contendrá la representación en caracteres de la figura que se desea dibujar.
- La clase Picture ya cuenta con una función implementada, no debe modificarla, pero si puede usarla para implementar sus otras funciones:

  o invColor: recibe un color como un carácter de texto y devuelve su color negativo, también como texto, deberá revisar el archivo colors.py para conocer los valores negativos de cada carácter.
- La clase Picture contará además con varios métodos que usted deberá implementar:

  SOLUCION 2:

  Primeramente, se pasar a explicar la clase “picture.py”

  En esta clase Picture se representa una imagen y permite comparar dos instancias de Picture por igualdad, así como invertir el color de un color dado utilizando el método invColor.

- El método init es el constructor de la clase Picture. Recibe un parámetro img que representa la imagen y lo asigna al atributo self.img de la instancia.
- El método .^eq"sobrecarga el operador de igualdad (===) para comparar dos instancias.
- El método invColor es un método privado que toma un color como argumento y devuelve el color invertido.

```python
from colors import *
Generate tests for the below class
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]
```

- El método verticalMirror recorre cada fila de la imagen (self.img) y agrega una versión invertida de la fila al resultado (vertical). Luego, devuelve una nueva instancia de Picture con la imagen verticalmente reflejada.
- El método horizontalMirror crea una lista vacía llamada horizontal. Luego, recorre cada fila de la imagen (self.img) y la inserta al principio de la lista horizontal.

```python
  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 = []
    for row in self.img: """Permite obtener una versión reflejada horizontalmente de una imagen."""
    horizontal.insert(0, row)
    return Picture(horizontal)

```

- El método negative recorre cada fila de la imagen (self.img) y crea una nueva fila en la lista negative.
- El método join recorre las filas de la imagen actual (self.img) y la imagen pasada como argumento (p.img).
- El método up toma otra imagen (p) como argumento y combina verticalmente ambas imágenes.

```python
  def negative(self):
    """ Devuelve un negativo de la imagen """
    negative = [
      ''.join(self._invColor(char) for char in value)
    for value in self.img
    ]
    return Picture(negative)

  def join(self, p):
    """ Devuelve una nueva figura poniendo la figura del argumento
        al lado derecho de la figura actual """
    join = []
    """Almacenará las filas combinadas de las dos imágenes."""
    for i in range(len(self.img)):
      join.append(self.img[i] + "" + p.img[i])
    return Picture(join)
"""Permite combinar verticalmente dos imágenes, apilando una encima de la otra."""
Generate tests for the below function
def up(self, p):
  image = self.img + p.img
  return Picture(image)
```

- El método under toma otra imagen p como argumento y devuelve una nueva imagen que combina la imagen actual (self.img) con la imagen p colocada sobre ella.
- El método horizontalRepeat toma un valor entero n como argumento y devuelve una nueva imagen que repite la imagen actual (self.img) a su lado n veces.

```
45    def up(self, p):
46      image = self.img + p.img
47      return Picture(image)
48
49      def under(self, p):
50        """ Devuelve una nueva figura poniendo la figura p sobre la
51            figura actual """
52        image = []
53        for i in range(len(self.img)):
54          line = ""    """almacenará la fila combinada."""
55          for j in range(self.img[i]):
56            if p.img[i][j] != " ":
57              line += p.img[i][j]
58            else:
59              line += self.img[i][j]
60          image.append(line)
61        return Picture(image)
62
63      def horizontalRepeat(self, n):
64        """ Devuelve una nueva figura repitiendo la figura actual al costado
65            la cantidad de veces que indique el valor de n """
66        image = []
67        for i in range(0,len(self.img)):
68            image.append(self.img[i] * n)
69        return Picture(image)
```

- Este método permite repetir una imagen verticalmente, creando una nueva imagen que contiene múltiples copias de la imagen original una encima de la otra.

```
71      def verticalRepeat(self, n):
72          image = self.img * n
73          """permite repetir verticalmente la imagen actual"""
74          return Picture(image)
```

RESOLVIENDO:

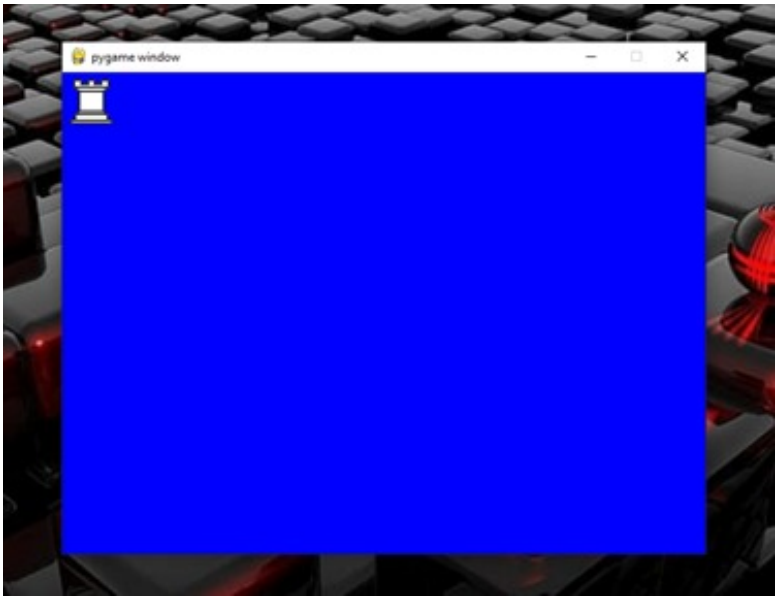
- Para resolver los siguientes ejercicios sólo está permitido usar ciclos, condicionales, definición de listas por comprensión, sublistas, map, join, (+), lambda, zip, append, pop, range.
- Implemente los métodos de la clase Picture. Se recomienda que implemente la clase picture por etapas, probando realizar los dibujos que se muestran en la siguiente preguntas.
- Usando únicamente los métodos de los objetos de la clase Picture dibuje las siguientes figuras (invoque a draw):

  Al clonar todos los documentos del GitHub en nuestro repositorio local obtenemos el siguiente resultado.

  Mediante los comandos:

```
Klismann@DESKTOP-MG9N9AV MINGW64 /c/tee/ProyectosPweb2/Lab04 (main)
$ python
ense" for more information.
>>> from chessPictures import *
>>> from interpreter import draw
pygame 2.4.0 (SDL 2.26.4, Python 3.11.2)
Hello from the pygame community. https://www.pygame.org/contribute.html
>>> draw(rock)
```

Obtenemos:




Resolviendo:

Ejercicio2a.

- Se crea una figura compuesta por dos filas. En la primera fila, se muestra la imagen knight seguida por su negativo (knight negative).

```
1   from interpreter import draw
2   from chessPictures import *
3
4   knight_negative = knight.negative()
5   line1 = "".join([knight, knight_negative])
6
7   line2 = "".join([knight_negative, knight])
8   final = line1.up(line2)
9
10  draw(final)
```

Obtenemos:


Ejercicio2b.

- El código crea una figura compuesta por dos filas. En la primera fila, se muestra la imagen knight seguida por su negativo (knightA). En la segunda fila, se muestra la imagen knight espejada verticalmente seguida por la versión espejada verticalmente de knightA.

```python
from interpreter import draw    Missing module docstring
from chessPictures import *    Wildcard import chessPictures

# Obtener la representación negativa del knight
knightA = knight.negative()
line1 = knight.join(knightA)
knightA2 = knightA.verticalMirror() # Obtener la versión espejada verticalmente de knightA
line2 = knight.verticalMirror().join(knightA2)
final = line1.up(line2)

draw(final)

```

Obtenemos:


Ejercicio2c.

- Se crea una figura compuesta por cuatro copias de la imagen de la reina, donde cada copia se obtiene duplicando la imagen anterior y agregando una nueva copia de la imagen original de la reina.

```
Generate tests for the below function
from interpreter import draw    Missing module docstring
from chessPictures import *    Wildcard import chessPictur

#Creamos varias variables para almacenar el resultado
# de concatenar la cadena "queen" en diferentes momentos.
queen_times_1 = queen
queen_times_2 = queen_times_1 + queen_times_1
queen_times_3 = queen_times_2 + queen_times_1
queen_times_4 = queen_times_3 + queen_times_1

final = queen_times_4
draw(final)

```

Obtenemos:


Ejercicio2d.

- El código crea una figura compuesta por la concatenación de la imagen square con su negativo.

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

square_negative = square.negative()
square_concatenated = square.join(square_negative)
final = square_concatenated.horizontalRepeat(4)
draw(final)
```

Obtenemos:


Ejercicio2e.

- Se crea una figura compuesta por la concatenación del negativo de la imagen square con la imagen square. Luego, esta figura se repite horizontalmente cuatro veces y se muestra mediante la función draw.

```
1   from interpreter import draw
2   from chessPictures import *
3
4   # Obtener la representación negativa del cuadrado
5   square_negative = square.negative()
6   concatenated = square_negative.join(square)
7   repeated = concatenated.horizontalRepeat(4)  # Rep
8
9   draw(repeated)
10
```

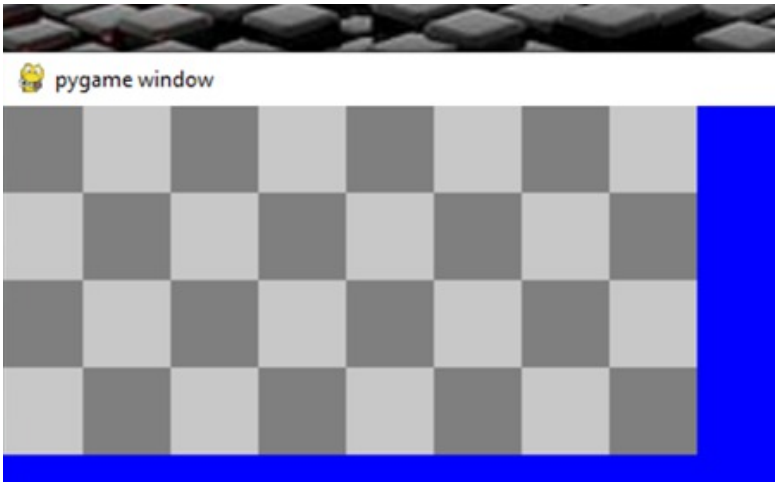
Obtenemos:


Ejercicio2f.

- Se crea una figura compuesta por la concatenación de la imagen square y su negativo, repetida verticalmente dos veces. Luego, esta figura repetida se concatena con su negativo, y finalmente se repite horizontalmente cuatro veces.

```
1   from interpreter import draw
2   from chessPictures import *
3
4   # Obtener la representación negativa del cuadrado y el cuadrado original
5   square_negative = square.negative()
6   square_image = square.img
7
8   # Concatenar la representación negativa y el cuadrado original para formar una imagen
9   image = Picture(square_negative.img + square_image)
10  image_repeated = image.verticalRepeat(2) # Repetir verticalmente la imagen dos veces
11  image_repeated_negative = image_repeated.negative() # Obtener la representación negativa de la imagen repetida
12  concatenated_image = image_repeated.join(image_repeated_negative)
13  final_image = concatenated_image.horizontalRepeat(4)
14
15  draw(final_image)
```

Obtenemos:




Ejercicio2g.

- Se obtienen los negativos de las figuras.
- Se crean las representaciones de las figuras negras en cuadrados claros utilizando el método under y se asignan a las variables correspondientes.
- Se crean las representaciones de las figuras negras en cuadrados oscuros utilizando el método under y se asignan a las variables correspondientes.
- Se crean las representaciones de las figuras blancas en cuadrados claros.
- Se crean las representaciones de las figuras blancas en cuadrados oscuros.
- Se crean las representaciones de los reyes y reinas en sus respectivos cuadrados utilizando el método under.
- Se crean las filas del tablero uniendo las representaciones de las figuras y cuadrados según la disposición del tablero de ajedrez.
- Se llama a la función draw con el argumento tablero para mostrar el tablero de ajedrez resultante.

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

# Obtener las representaciones negativas de las figuras
rockN = rock.negative()
knightA = knight.negative()
bishopN = bishop.negative()
queenN = queen.negative()
kingN = king.negative()
pawnN = pawn.negative()
squareN = square.negative()

# Figuras negras en cuadrados claros
rockNsquare = square.under(rockN)
knightAsquare = square.under(knightA)
bishopNsquare = square.under(bishopN)
queenNsquare = square.under(queenN)
kingNsquare = square.under(kingN)

# Figuras negras en cuadrados oscuros
rockNsquareN = squareN.under(rockN)
knightAsquareN = squareN.under(knightA)
bishopNsquareN = squareN.under(bishopN)
pawnNsquareN = squareN.under(pawnN)

```

```python
# Figuras blancas en cuadrados claros
rockSquare = square.under(rock)
knightSquare = square.under(knight)
bishopSquare = square.under(bishop)
queenSquare = square.under(queen)
kingSquare = square.under(king)
pawnSquare = square.under(pawn)

# Figuras blancas en cuadrados oscuros
rockSquareN = squareN.under(rock)
knightSquareN = squareN.under(knight)
bishopSquareN = squareN.under(bishop)
pawnSquareN = squareN.under(pawn)

# Reyes y reinas en sus respectivos cuadrados
kingNSquare = square.under(kingN)
kingSquareN = squareN.under(king)
queenNSquare = squareN.under(queenN)
queenSquare = square.under(queen)

# Crear las filas del tablero
fila1 = rockNsquare.join(knightAsquareN).join(bishopNsquare).join(queenNSquare).join(kingNSquare).join(bishopNsquar
fila2 = pawnNsquareN.join(pawnNsquare).horizontalRepeat(4)

centro1 = square.join(squareN).horizontalRepeat(4)
centro2 = centro1.verticalMirror()
centro = centro1.up(centro2).verticalRepeat(2)
```

```python
fila3 = pawnSquare.join(pawnSquareN).horizontalRepeat(4)
fila4 = rockSquareN.join(knightSquare).join(bishopSquareN).join(queenSquare).join(kingSquare).join(bishopSquare).joi

# Dibujar el tablero
tablero = fila1.up(fila2).up(centro).up(fila3).up(fila4)
draw(tablero)
```

Obtenemos:


COMMITS

```
commit 7417e7954e09019214e4ccd71efaf2a27a06ac65
Author: klismannSis <kchancuana@unsa.edu.pe>
Date:   Sun Jun 4 18:40:02 2023 -0500

    ----------

commit e7a284e05f4125bdb39e076ec35af61d0fa7d7ad
Author: klismannSis <kchancuana@unsa.edu.pe>
Date:   Sun Jun 4 16:41:52 2023 -0500

    ........

commit af108711a32807a6a7b249961871637156c1dfa5
Author: klismannSis <kchancuana@unsa.edu.pe>
Date:   Sun Jun 4 16:30:27 2023 -0500

    ....

commit 6a954c7d6bcb834311952ad3f31395cee1a3b089
Author: klismannSis <kchancuana@unsa.edu.pe>
Date:   Sun Jun 4 16:25:13 2023 -0500

    ----

commit 7f5b10c125c81748db8f0476b67b08bad35d2f67
Author: Klismann30 <118670708+klismannSis@users.noreply.github.com>
Date:   Sun May 28 15:02:15 2023 -0500

    Update README.md

commit 6800ee3483b3d068fd0d81f24afdea17877d1cb2
Author: Klismann30 <118670708+klismannSis@users.noreply.github.com>
Date:   Sun May 28 15:00:59 2023 -0500
```

```
commit 808b328ffcaa44aeef3b7111a3da674d61d75427
Author: klismannSis <kchancuana@unsa.edu.pe>
Date:   Sun Jun 4 22:24:48 2023 -0500

    /Continuando con la implementacion.

commit e6ae09549531260677cf8cb226c1e37f212b324c
Author: klismannSis <kchancuana@unsa.edu.pe>
Date:   Sun Jun 4 21:01:38 2023 -0500

    /Completando la codificacion faltante.

commit 1447afa81d8034c8a29e848d9ea772929bcdfe7f
Author: klismannSis <kchancuana@unsa.edu.pe>
Date:   Sun Jun 4 20:16:53 2023 -0500

    /Completo.

commit 6ae15c359a5d22b0a56aca4d30ece0eac5beb38c
Author: klismannSis <kchancuana@unsa.edu.pe>
Date:   Sun Jun 4 20:14:08 2023 -0500

    /Completo.

commit 136e751f33bd83755314e5b78df1d0eab018eb3c
Author: klismannSis <kchancuana@unsa.edu.pe>
Date:   Sun Jun 4 20:08:39 2023 -0500

    /Completo.

commit 7417e7954e09019214e4ccd71efaf2a27a06ac65
Author: klismannSis <kchancuana@unsa.edu.pe>
Date:   Sun Jun 4 18:40:02 2023 -0500
```

```
commit 497e39fee4782204c34a4a912684e9ef659340ef
Author: klismannSis <kchancuana@unsa.edu.pe>
Date:   Mon Jun 5 19:57:28 2023 -0500

    /Corrigiendo errores.

commit abf25ef00fbfb6f574158eff5a29ba13509323d6
Author: klismannSis <kchancuana@unsa.edu.pe>
Date:   Sun Jun 4 23:52:22 2023 -0500

    /Rellenando codigo.

commit 4b80dbc64ca3db9373a88f19d088aa1d91df1120
Author: klismannSis <kchancuana@unsa.edu.pe>
Date:   Sun Jun 4 23:32:19 2023 -0500

    /Iplementando Codigo.

commit b639dd53f6e57d823daa35c5cbd850c37d797364
Author: klismannSis <kchancuana@unsa.edu.pe>
Date:   Sun Jun 4 23:01:42 2023 -0500

    /Codigo Completado.

commit ff8c7117eaf7130be27c99b5659833be225abc3e
Author: klismannSis <kchancuana@unsa.edu.pe>
Date:   Sun Jun 4 22:28:51 2023 -0500

    /Casi completa...
```

```
commit 7f5a799de28b47d45389acf7527bb8a7fb967ddf
Author: klismannSis <kchancuana@unsa.edu.pe>
Date:   Mon Jun 5 21:00:30 2023 -0500

    .........

commit c0bd5e5288a7f3815022d8176cbe8b31bdfa2db9
Author: klismannSis <kchancuana@unsa.edu.pe>
Date:   Mon Jun 5 20:52:50 2023 -0500

    COMPLETADO.

commit 0d26758e609a465241a08fd5642b6af008553abf
Author: klismannSis <kchancuana@unsa.edu.pe>
Date:   Mon Jun 5 20:43:48 2023 -0500

    COMPLETADO.

commit 611dc945b240cb568341099d6552f80a04d0065d
Author: klismannSis <kchancuana@unsa.edu.pe>
Date:   Mon Jun 5 20:36:11 2023 -0500

    Corregido y completo.

commit 696618dc585f77dada072d24c8ba1e2ffe00cb8f
Author: klismannSis <kchancuana@unsa.edu.pe>
Date:   Mon Jun 5 20:00:55 2023 -0500

    /Retocado.
```

```
commit 8fb6251f8782fc99968fbb11cc665e572dffffec
Author: Klismann30 <118670708+klismannSis@users.noreply.github.com>
Date:   Tue Jun 6 02:15:46 2023 -0500

    Rename Ejercicio2g.py to README.md

commit 2389932db0c54379daaae21e7885735f1f052595
Author: klismannSis <kchancuana@unsa.edu.pe>
Date:   Tue Jun 6 02:12:30 2023 -0500

    Ready

commit 4c40bcb9bdd81eb47c7688d21850c08711b097a9
Author: klismannSis <kchancuana@unsa.edu.pe>
Date:   Tue Jun 6 01:55:23 2023 -0500

    /Ready.

commit 80eaad4fc86363abc23f64989363f27e742949fa
Author: klismannSis <kchancuana@unsa.edu.pe>
Date:   Tue Jun 6 01:42:55 2023 -0500

    C:/Program Files/Git/Comletando...

commit f3e6d28d1a1ecdef82f72770eb762573fa0d53e3
Author: klismannSis <kchancuana@unsa.edu.pe>
Date:   Mon Jun 5 21:45:20 2023 -0500

    /Implementadas.
```

```
commit 251f1921c103870bb93accfaa079343bdf59fc0b
Author: klismannSis <kchancuana@unsa.edu.pe>
Date:   Tue Jun 6 07:46:50 2023 -0500

    Ready

commit f547ac37cda01e10bef149074a9b57b233d27cd3
Author: Klismann30 <118670708+klismannSis@users.noreply.github.com>
:...skipping...
commit 390cc38f2d880016640d484d0c9ea698ea9c83a7 (HEAD -> main, origin/main, origin/HEAD)
Author: klismannSis <kchancuana@unsa.edu.pe>
Date:   Thu Jun 8 11:26:02 2023 -0500

    img.Codigos

commit 82bd3175bbbac893aad2939d5c66205919e581e6
Author: klismannSis <kchancuana@unsa.edu.pe>
Date:   Tue Jun 6 07:59:14 2023 -0500

    Implementando codigo

commit 132937a6dbd14845a33e775031dd5f85777d8300
Merge: 251f192 f547ac3
Author: klismannSis <kchancuana@unsa.edu.pe>
Date:   Tue Jun 6 07:47:17 2023 -0500

    Merge branch 'main' of https://github.com/klismannSis/ProyectosPweb2

commit 251f1921c103870bb93accfaa079343bdf59fc0b
Author: klismannSis <kchancuana@unsa.edu.pe>
Date:   Tue Jun 6 07:46:50 2023 -0500

    Ready
```

```
Klismann@DESKTOP-MG9N9AV MINGW64 /c/tee/ProyectosPweb2/Lab04/Labt04 (main)
$ ls
__pycache__/      colors.py       Ejercicio2b.py  Ejercicio2d.py  Ejercicio2f.py  Imagenes/       picture.py
chessPictures.py  Ejercicio2a.py  Ejercicio2c.py  Ejercicio2e.py  Ejercicio2g.py  interpreter.py  pieces.py

Klismann@DESKTOP-MG9N9AV MINGW64 /c/tee/ProyectosPweb2/Lab04/Labt04 (main)
$ git log
commit 390cc38f2d880016640d484d0c9ea698ea9c83a7 (HEAD -> main, origin/main, origin/HEAD)
Author: klismannSis <kchancuana@unsa.edu.pe>
Date:   Thu Jun 8 11:26:02 2023 -0500

    img.Codigos

commit 82bd3175bbbac893aad2939d5c66205919e581e6
:...skipping...
commit 390cc38f2d880016640d484d0c9ea698ea9c83a7 (HEAD -> main, origin/main, origin/HEAD)
Author: klismannSis <kchancuana@unsa.edu.pe>
Date:   Thu Jun 8 11:26:02 2023 -0500

    img.Codigos

commit 82bd3175bbbac893aad2939d5c66205919e581e6
Author: klismannSis <kchancuana@unsa.edu.pe>
Date:   Tue Jun 6 07:59:14 2023 -0500

    Implementando codigo

commit 132937a6dbd14845a33e775031dd5f85777d8300
Merge: 251f192 f547ac3
Author: klismannSis <kchancuana@unsa.edu.pe>
Date:   Tue Jun 6 07:47:17 2023 -0500
```

## 2. Equipos, materiales y temas utilizados

- Sistema Operativo Ubuntu GNU Linux 23 lunar 64 bits Kernell 6.2.
- VIM 9.0.
- Python 3.11.2.
- Git 2.39.2.
- Cuenta en GitHub con el correo institucional.
- Programación Orientada a Objetos.
- Algoritmo de ordenamiento por inserción

## 3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- `https://github.com/rescobedoq/pw2.git`
- URL para el laboratorio 01 en el Repositorio GitHub.
- `https://github.com/rescobedoq/pw2/tree/main/labs/lab04`
- Link del repositorio github
- `https://github.com/klismannSis/ProyectosPweb2/tree/main/Lab04`

# 4. Cuestionario

¿Qué son los archivos *.pyc?

los archivos ".pyc"son archivos de código byte compilado en Python que se utilizan para acelerar la carga y ejecución de programas, así como para proteger el código fuente original.

¿Para qué sirve el directorio pycache?

Los archivos ".pyc.[en] Python son archivos que contienen el código compilado de un programa. Cuando un programa en Python se ejecuta, el intérprete traduce el código fuente a un formato intermedio llamado bytecode. Este bytecode se guarda en archivos ".pyc"para permitir su reutilización en ejecuciones futuras del programa.

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

- Un guión bajo () (class) Se utiliza para evitar conflictos con palabras clave o elementos relacionados.
- Un guión bajo (variable) En este caso indica que el nombre que sigue al guion es una clase, función, método o variable. - Un doble guión bajo () (Klismann)

Cualquier mombre de la forma anonimo se sustituye por NombreClase anonimo. - Un doble guion bajo () ( casa ). Se utiliza para indicar métodos específicos conocidos como métodos mágicos, init, file. Su objetivo es evitar conflictos entre los métodos mágicos y algún método definido por nosotros.

# 5. Conclusion

El código de tablero de ajedrez demuestra el uso de diferentes conceptos y técnicas de programación. Las clases y los métodos se utilizan para organizar y organizar el código. Además, se controlan imágenes y se colocan aplicaciones para combinar y repetir los procesos para construir un tablero completo. Sin embargo, el código demuestra el uso de programación orientada a objetos, operaciones de visualización de datos y el uso de lógica de bucle para lograr el resultado deseado.

# 6. Rúbricas

## 6.1. Entregable Informe

Tabla 1: Tipo de Informe

| **Informe** | |
|---|---|
| **Latex** | En conclusion este informe esta editado en latex y en formato PDF. |

# 7. Referencias


- `https://www.w3schools.com/python/python_reference.asp`
- `https://docs.python.org/3/tutorial/`