Laboratorio 04
Tema: Python

Melsy Melany Huamaní Vargas

6 de junio de 2023

Profesor	Escuela	Asignatura
Carlo Corrales Anibal Sardon	Ingeniería de Sistemas	Programación Web 2

Laboratorio	Tema	Duración
04	Python	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - A	30/05/2023	06/06/2023

SOLUCIÓN Y RESULTADOS

ENLACE GITHUB Y GIT

Este trabajo se está presentando en el Github: `https://github.com/mhuamanivar/PW2-HuamaniV-Lab04` y el .git es: `https://github.com/mhuamanivar/PW2-HuamaniV-Lab04.git`.

INSTALACIÓN PARA EL USO DE PYTHON

Se utiliza el subsistema Ubuntu 22.04.2 LTS a través de WSL (Windows Subsystem for Linux) para usar Python y las herramientas necesarias para su uso en estos ejercicios.

- Primero se verifica la versión de Python.

```
melsy@bonne:~$ python3 --version
Python 3.10.6
```

- Luego actualizamos los paquetes del sistema de Linux.

```
melsy@bonne:~$ sudo apt-get update
[sudo] password for melsy:
Get:1 http://security.ubuntu.vom.ubuntu jammy-security InRelease [110 kB]
Hit:2 http://archive.ubuntu.vom.ubuntu jammy InRelease
Get:3 http://archive.ubuntu.vom.ubuntu jammy-updates InRelease [119 kB]
Get:4 http://security.ubuntu.vom.ubuntu jammy-security/main amd64 Packages [442 kB]
...
```

- Después se instala pip, una herramienta que (también) instala y administra los paquetes de programación que queramos usar en nuestros proyectos de desarrollo.

```
melsy@bonne:~$ sudo apt-get install -y python3-pip
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
...
```

- Se instalan paquetes y herramientas de desarrollo que garantizan una configuración sólida para nuestro entorno de programación.

```
melsy@bonne:~$ sudo apt-get install build-essential libssl-dev libffi-dev python3-dev
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
build-essential is already the newest version (12.9ubuntu3).
...
```

- Después, se configura un entorno virtual, los cuales permiten tener un espacio aislado en los proyectos Python y garantizan que puedan tener su propio conjunto de dependencias, como diferentes versiones de los paquetes o varios entornos de programación. En este caso, se usa el módulo venv, que es parte de la biblioteca estándar de Python y se instala de la siguiente forma.

```
melsy@bonne:~$ sudo apt install -y python3-venv
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
...
```

- Luego se crea el directorio en donde queremos crear el entorno virtual y se ingresa con “cd“.

```
melsy@bonne:~$ mkdir -p ~/univ/pw2/lab04/my_env
melsy@bonne:~$ cd ~/univ/pw2/lab04/my_env
```

- En el directorio se crea un entorno virtual ejecutando el siguiente comando.

```
melsy@bonne:~/univ/pw2/lab04/my_env$ virtualenv -p python3 .
created virtual environment CPython3.10.6.final.0-64 in 137ms
  creator CPython3Posix(dest=/home/melsy/univ/pw2/lab04/my_env, clear=False,
      no_vcs_ignore=False, global=False)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=
      copy, app_data_dir=/home/melsy/.local/share/virtualenv)
    added seed packages: pip==22.0.2, setuptools==59.6.0, wheel==0.37.1
  activators BashActivator,CShellActivator,FishActivator,NushellActivator,
      PowerShellActivator,PythonActivator
melsy@bonne:~/univ/pw2/lab04/my_env$ mkdir -p ~/univ/pw2/lab04/my_env/src
melsy@bonne:~/univ/pw2/lab04/my_env$ cd ~/univ/pw2/lab04/my_env/src
```

- Ahora se activa el entorno virtual.

```
melsy@bonne:~/univ/pw2/lab04/my_env/src$ source ../bin/activate
(my_env) melsy@bonne:~/univ/pw2/lab04/my_env/src$
```

- Se crea un archivo “hello.py“ para probar la ejecución de un archivo Python.

```
(my_env) melsy@bonne:~/univ/pw2/lab04/my_env/src$ vim hello.py
```

- Se edita el archivo ’hello.py’, colocando lo siguiente.

```
1  print("Hello, World!")
```

- Se sale de vim con “Esc“, luego se escribe “:wq“ para guardar y salir del archivo. Luego se ejecuta el archivo Python.

```
(my_env) melsy@bonne:~/univ/pw2/lab04/my_env/src$ python3 hello.py
Hello, World!
```

- Se ve que se ejecutó correctamente. Para desactivar el entorno virtual, se utiliza el siguiente comando.

```
(my_env) melsy@bonne:~/univ/pw2/lab04/my_env/src$ deactivate
melsy@bonne:~/univ/pw2/lab04/my_env/src$
```

I. EJERCICIOS RESUELTOS

Se activa el entorno virtual y se crean los directorios donde trabajaremos los ejercicios resueltos.

```
melsy@bonne:~/univ/pw2/lab04/my_env/src$ source ../bin/activate
(my_env) melsy@bonne:~/univ/pw2/lab04/my_env/src$ mkdir resueltos
(my_env) melsy@bonne:~/univ/pw2/lab04/my_env/src$ cd resueltos/
(my_env) melsy@bonne:~/univ/pw2/lab04/my_env/src/resueltos$
```

1. Determinar si una matriz de tamaño N x N es escalar.

- Se crea el archivo “esEscalar.py“.

```python
def esEscalar(m):
    d = m[0][0]
    for i in range(len(m)):
        for j in range(len(m)):
            if i != j:
                if m[i][j] != 0:
                    print(m[i][j])
                    return False
            elif m[i][j] != d:
                print(m[i][j])
                return False
    return True
```

- Se crea el archivo “test_esEscalar.py“, en el cual importamos el archivo anterior como función.

```python
import esEscalar as fu

def prueba(M):
    if fu.esEscalar(M):
        print("Si es escalar")
    else:
        print("No es escalar")

#Z = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
#Z = [[1, 2, 3], [4, 1, 6], [7, 8, 1]]
Z = [[1, 0, 0], [0, 1, 0], [0, 0, 1]]

prueba(Z)
```

- Se ejecuta el archivo “test_esEscalar.py“ con la matriz de prueba que no está comentada.

```
(my_env) melsy@bonne:~/univ/pw2/lab04/my_env/src/resueltos$ python3 test_esEscalar.py
Si es escalar
```

2. Determinar si una matriz de tamaño N x N es unitaria.

- Se crea el archivo “esUnitaria.py“.

```python
import esEscalar as fu

def esUnitaria(m):
    return m[0][0] == 1 and fu.esEscalar(m)
```

- Se crea el archivo “test_esUnitaria.py“, en el cual importamos el archivo anterior como función.

```python
import esUnitaria as fu

def prueba(M):
    if fu.esUnitaria(M):
        print("Si es unitaria")
    else:
        print("No es unitaria")

#Z = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
#Z = [[1, 2, 3], [4, 1, 6], [7, 8, 1]]
Z = [[2, 0, 0], [0, 2, 0], [0, 0, 2]]
#Z = [[1, 0, 0], [0, 1, 0], [0, 0, 1]]

prueba(Z)
```

- Se ejecuta el archivo “test_esUnitaria.py“ con la matriz de prueba que no está comentada.

```
(my_env) melsy@bonne:~/univ/pw2/lab04/my_env/src/resueltos$ python3 test_esUnitaria.py
No es unitaria
(my_env) melsy@bonne:~/univ/pw2/lab04/my_env/src/resueltos$ deactivate
```

II. EJERCICIOS PROPUESTOS

En este caso como se estaba trabajando en WSL, no está permitido el uso de gráficos como “pygame“, por lo que se trabajó en el sistema principal Windows. Por lo que se crea un nuevo ambiente y lo activamos.

```
C:\Users\melsy\Lab04>python -m venv my_env
C:\Users\melsy\Lab04>my_env\Scripts\activate
```

Ahora se actualiza el pip de python en Windows.

```
(my_env) C:\Users\melsy\Lab04>python.exe -m pip install --upgrade pip
Requirement already satisfied: pip in c:\users\melsy\lab04\my_env\lib\site-packages (22.3.1)
Collecting pip
  Downloading pip-23.1.2-py3-none-any.whl (2.1 MB)
     ---------------------------------- 2.1/2.1 MB 848.2 kB/s eta 0:00:00
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 22.3.1
    Uninstalling pip-22.3.1:
      Successfully uninstalled pip-22.3.1
Successfully installed pip-23.1.2
```

Posteriormente se instala pygame para mostrar los gráficos.

```
(my_env) C:\Users\melsy\Lab04>pip3 install pygame
Collecting pygame
  Downloading pygame-2.4.0-cp311-cp311-win_amd64.whl (10.6 MB)
     ---------------------------------- 10.6/10.6 MB 531.0 kB/s eta 0:00:00
Installing collected packages: pygame
Successfully installed pygame-2.4.0
```

Luego se ingresa a una carpeta ya creada llamada “propuestos“, donde (como dice el nombre) se guardarán todos los ejercicios propuestos.

```
(my_env) C:\Users\melsy\Lab04>cd my_env\Scripts\propuestos
(my_env) C:\Users\melsy\Lab04\my_env\Scripts\propuestos>
```

1. Implementar los métodos de la clase Picture.

Una vez guardados los archivos dados por el profesor se deben realizar pruebas para realizar los métodos que son dados en este primer ejercicio propuesto.

```
(my_env) C:\Users\melsy\Lab04\my_env\Scripts\propuestos>python
Python 3.11.1 (tags/v3.11.1:a7a450f, Dec  6 2022, 19:58:39) [MSC v.1934 64 bit (AMD64)] on
     win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from chessPictures import *
>>> from interpreter import draw
pygame 2.4.0 (SDL 2.26.4, Python 3.11.1)
Hello from the pygame community. https://www.pygame.org/contribute.html
```

Ahora se realizan los métodos a continuación:

- **verticalMirror:** Devuelve el espejo vertical de la imagen

  Se crea un arreglo vacío "vertical" en donde se coloca poco a poco cada valor del arreglo de la imagen dada, de manera que cada string parte de ese arreglo se invierta utilizando "[::-1]", de esta manera como cada elemento se está inviertando se obtiene un resultado en forma de reflejo vertical de la imagen.

```
    def verticalMirror(self):
        vertical = []
        for value in self.img:
            vertical.append(value[::-1])
        return Picture(vertical)
```

  - Para probar utilizamos el siguiente comando

```
>>> draw(knight)
```

  - Luego comparamos al utilizar el siguiente comando, para ver que funcionó correctamente

```
>>> draw(knight.verticalMirror())
```

- **horizontalMirror:** Devuelve el espejo horizontal de la imagen

  Se crea un arreglo vacío “horizontal“ en el cual se almacena el arreglo invertido del atributo “img“ del picture, se esta manera se obtiene un espejo horizontal de la figura, de la misma forma que el anterior para invertir el arreglo se utiliza “[::-1]“.

```
1       def horizontalMirror(self):
2             horizontal = self.img[::-1]
3             return Picture(horizontal)
```

  - Para probar utilizamos el siguiente comando

```
>>> draw(rock)
```

  - Luego comparamos al utilizar el siguiente comando, para ver que funcionó correctamente

```
>>> draw(rock.horizontalMirror())
```

- **negative:** Devuelve un negativo de la imagen

  Se crea un nuevo arreglo "negative", el cual por cada valor del arreglo del atributo "img" del picture, se invertira los colores al utilizar el método "_invColor()", luego se unen utilizando el método "join()" a un solo string y se devuelve como un nuevo elemento del nuevo arreglo creado.

```
    def negative(self):
        negative = []
        for value in self.img:
            invertedColor = "".join([self._invColor(c) for c in value
                ])
            negative.append(invertedColor)
        return Picture(negative)
```

  - Para probar utilizamos el siguiente comando

```
>>> draw(queen)
```

  - Luego comparamos al utilizar el siguiente comando, para ver que funcionó correctamente

```
>>> draw(queen.negative())
```

- **join:** Devuelve una nueva figura poniendo la figura del argumento al lado derecho de la figura actual

  Se crea un nuevo arreglo "joined" el cual for cada elemento del arreglo del atributo "img" del picture actual se agrega el elemento con el mismo índice del arreglo del atributo "img" del picture "p", de esta manera forman un nuevo arreglo que será almacenado en "joined".

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

  - Para probar utilizamos el siguiente comando

```
>>> draw(king)
```

  - Luego comparamos al utilizar el siguiente comando, para ver que funcionó correctamente

```
>>> draw(king.join(queen))
```

- **up:** Devuelve una nueva figura poniendo la figura recibida como argumento, encima de la figura actual

  Se crea un nuevo arreglo “newFigure“ que almacena el arreglo del atributo “img“ del picture que es recibido como argumento “p“, luego se le va añadiendo cada elemento del arreglo “img“ del picture actual, para que en “newFigure“, el picture “p“ se muestre encima del picture actual.

```
    def up(self, p):
        newFigure = p.img[:]
        for value in self.img:
            newFigure.append(value)
        return Picture(newFigure)
```

  - Para probar utilizamos el siguiente comando

```
>>> draw(bishop)
```

  - Luego comparamos al utilizar el siguiente comando, para ver que funcionó correctamente

```
>>> draw(bishop.up(pawn))
```

- **under:** Devuelve una nueva figura poniendo la figura recibida como argumento, sobre la figura actual

  Se crean las variables “filas“ y “columnas“ que almacena el valor adecuado al usar “range()“ en el arreglo. Luego se crea “newFigure“, el cual recorrerá por las filas y columnas, de esta forma cuando un caracter del picture actual sea diferente al del picture “p“, y este último no sea vacío, entonces se priorizará el caracter del picture “p“, de lo contrario, será del picture actual.

```
    def under(self, p):
        filas = range(len(self.img))
        columnas = range(len(self.img[0]))

        newFigure = []
        for i in filas:
            string = ""
            for j in columnas:
                if (self.img[i][j] != p.img[i][j] and p.img[i][j] != "
                    "):
                    string += p.img[i][j]
                else:
                    string += self.img[i][j]
            newFigure.append(string)
        return Picture(newFigure)
```

  - Para probar utilizamos el siguiente comando

```
>>> draw(square)
```

  - Luego comparamos al utilizar el siguiente comando, para ver que funcionó correctamente

```
>>> draw(square.under(knight))
```

- **horizontalRepeat::** Devuelve una nueva figura repitiendo la figura actual al costado la cantidad de veces que indique el valor de “n“

  Se crea un nuevo arreglo, el cual va almacenando los elementos del arreglo del atributo “img“ del pciture, y los va multiplicando la cantidad “n“ introducida, de tal manera que al final se vea de manera repetida horizontalmente la misma figura.

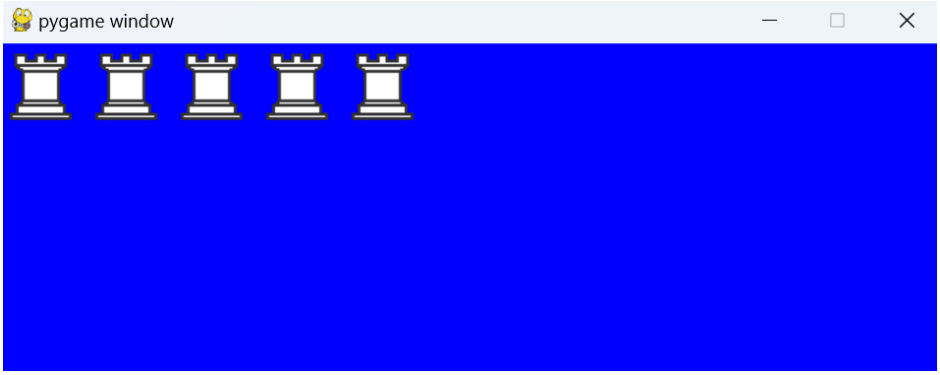
```
    def horizontalRepeat(self, n):
        newFigure = []
        for value in self.img:
            newFigure.append(value*n)
        return Picture(newFigure)
```

  - Para probar utilizamos el siguiente comando

```
>>> draw(rock)
```

  - Luego comparamos al utilizar el siguiente comando, para ver que funcionó correctamente

```
>>> draw(rock.horizontalRepeat(5))
```



- **verticalRepeat:** Devuelve una nueva figura repitiendo la figura actual debajo, la cantidad de veces que indique el valor de "n"

  Se crea un nuevo arreglo donde multiplica todo el arreglo del atributo "img" del picture, es por ello que en el resultado se ve de manera repetida verticalmente.

```
    def verticalRepeat(self, n):
        newFigure = self.img[:] * n
        return Picture(newFigure)
```

  - Para probar utilizamos el siguiente comando

```
>>> draw(king)
```

  - Luego comparamos al utilizar el siguiente comando, para ver que funcionó correctamente

```
>>> draw(king.verticalRepeat(2))
```

- **rotate (Extra):** DDevuelve una figura rotada en 90 grados

  De la misma forma que en la función “under()“ se obtienen las filas y columnas. Luego se crea un nuevo arreglo "newFigure", en el cual se irá almacenando el valor correspondiente de las columnas, pero para que vaya rotando entonces colocamos el índice de las filas de manera “-i“, para obtener el orden inverso solamente de la fila.

```
    def rotate(self):
        """Devuelve una figura rotada en 90 grados """
        filas = range(len(self.img))
        columnas = range(len(self.img[0]))

        newFigure = []
        for i in filas:
            string = ""
            for j in columnas:
                string += self.img[j][-i]
            newFigure.append(string)
        return Picture(newFigure)
```

  - Para probar utilizamos el siguiente comando

```
>>> draw(knight)
```

  - Luego comparamos al utilizar el siguiente comando, para ver que funcionó correctamente

```
>>> draw(knight.rotate())
```

2. Usando únicamente los métodos de los objetos de la clase Picture dibujar las siguientes figuras.

Se sale de "python" con "exit()", pero continuamos en el entorno virtual para probar los ejercicios y demostrar que cumplen de acuerdo a las imágenes pedidas.

- **Se crea el archivo "Ejercicio2a.py"**

  Utilizando las funciones creadas anteriormente, se contruye de acuerdo a la imagen dada en el laboratorio.

```
from interpreter import draw
from chessPictures import *

wKnight = knight
bKnight = knight.negative()

firstKnights = wKnight.join(bKnight)
secondKnights = bKnight.join(wKnight)

draw(secondKnights.up(firstKnights))
```

  - Se ejecuta de la siguiente manera.

```
(my_env) C:\Users\melsy\Lab04\my_env\Scripts\propuestos>python Ejercicio2a.py
pygame 2.4.0 (SDL 2.26.4, Python 3.11.1)
Hello from the pygame community. https://www.pygame.org/contribute.html
```

  - El resultado es:

- **Se crea el archivo "Ejercicio2b.py"**

  Utilizando las funciones creadas anteriormente, se contruye de acuerdo a la imagen dada en el laboratorio.

```
from interpreter import draw
from chessPictures import *

wKnight = knight
bKnight = knight.negative()

firstKnights = wKnight.join(bKnight)
secondKnights = firstKnights.verticalMirror()

draw(secondKnights.up(firstKnights))
```

  - Se ejecuta de la siguiente manera.

```
(my_env) C:\Users\melsy\Lab04\my_env\Scripts\propuestos>python Ejercicio2b.py
pygame 2.4.0 (SDL 2.26.4, Python 3.11.1)
Hello from the pygame community. https://www.pygame.org/contribute.html
```

  - El resultado es:

- **Se crea el archivo "Ejercicio2c.py"**

  Utilizando las funciones creadas anteriormente, se contruye de acuerdo a la imagen dada en el laboratorio.

```
from interpreter import draw
from chessPictures import *

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

  - Se ejecuta de la siguiente manera.

```
(my_env) C:\Users\melsy\Lab04\my_env\Scripts\propuestos>python Ejercicio2c.py
pygame 2.4.0 (SDL 2.26.4, Python 3.11.1)
Hello from the pygame community. https://www.pygame.org/contribute.html
```

  - El resultado es:

- **Se crea el archivo "Ejercicio2d.py"**

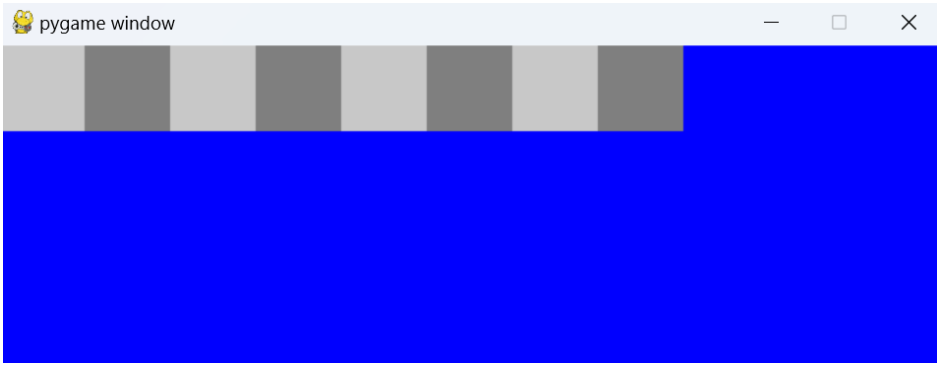
  Utilizando las funciones creadas anteriormente, se contruye de acuerdo a la imagen dada en el laboratorio.

```
    from interpreter import draw
    from chessPictures import *

    squares = square.join(square.negative())

    draw(squares.horizontalRepeat(4))
```

  - Se ejecuta de la siguiente manera.

```
(my_env) C:\Users\melsy\Lab04\my_env\Scripts\propuestos>python Ejercicio2d.py
pygame 2.4.0 (SDL 2.26.4, Python 3.11.1)
Hello from the pygame community. https://www.pygame.org/contribute.html
```

  - El resultado es:



- **Se crea el archivo "Ejercicio2e.py"**

  Utilizando las funciones creadas anteriormente, se contruye de acuerdo a la imagen dada en el laboratorio.

```
    from interpreter import draw
    from chessPictures import *

    squares = square.negative().join(square)

    draw(squares.horizontalRepeat(4))
```

  - Se ejecuta de la siguiente manera.

```
(my_env) C:\Users\melsy\Lab04\my_env\Scripts\propuestos>python Ejercicio2e.py
pygame 2.4.0 (SDL 2.26.4, Python 3.11.1)
Hello from the pygame community. https://www.pygame.org/contribute.html
```

  - El resultado es:

- **Se crea el archivo "Ejercicio2f.py"**

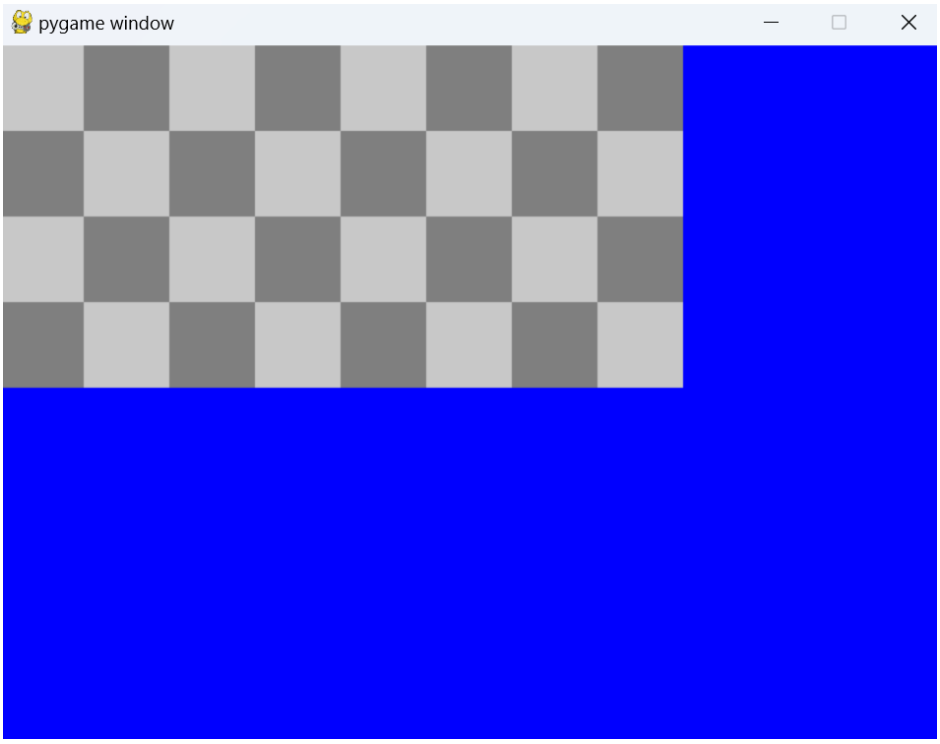
  Utilizando las funciones creadas anteriormente, se contruye de acuerdo a la imagen dada en el laboratorio.

```
from interpreter import draw
from chessPictures import *

squares1 = square.negative().up(square)
squares2 = square.up(square.negative())

finalSquares = squares1.join(squares2)

draw(finalSquares.verticalRepeat(2).horizontalRepeat(4))
```

  - Se ejecuta de la siguiente manera.

```
(my_env) C:\Users\melsy\Lab04\my_env\Scripts\propuestos>python Ejercicio2f.py
pygame 2.4.0 (SDL 2.26.4, Python 3.11.1)
Hello from the pygame community. https://www.pygame.org/contribute.html
```

  - El resultado es:



- **Se crea el archivo "Ejercicio2g.py"**

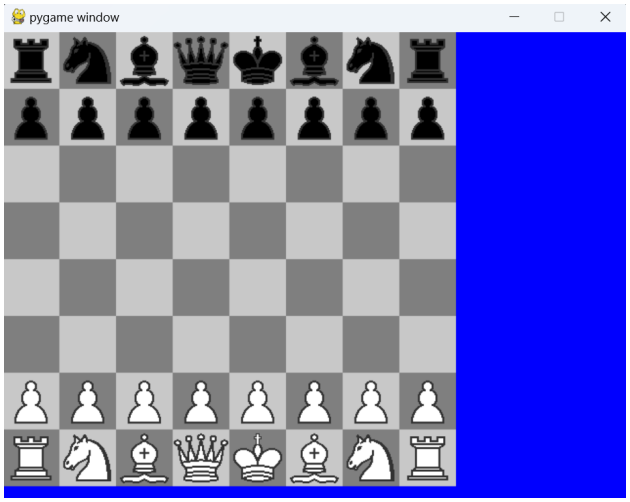
  Utilizando las funciones creadas anteriormente, se contruye de acuerdo a la imagen dada en el laboratorio.

```
from interpreter import draw
from chessPictures import *

wSquare = square
bSquare = square.negative()

wPawns = wSquare.under(pawn).join(bSquare.under(pawn)).
    horizontalRepeat(4)
bPawns = wPawns.negative()

wPieces = bSquare.under(rock).join(wSquare.under(knight)).join(
    bSquare.under(bishop)).join(wSquare.under(queen)).join(bSquare
    .under(king)).join(wSquare.under(bishop)).join(bSquare.under(
    knight)).join(wSquare.under(rock))
bPieces = wPieces.negative()

squares1 = bSquare.up(wSquare)
squares2 = wSquare.up(bSquare)

finalSquares = squares1.join(squares2).verticalRepeat(2).
    horizontalRepeat(4)

draw(wPieces.up(wPawns.up(finalSquares.up(bPawns.up(bPieces)))))
```

  - Se ejecuta de la siguiente manera.

```
(my_env) C:\Users\melsy\Lab04\my_env\Scripts\propuestos>python Ejercicio2g.py
pygame 2.4.0 (SDL 2.26.4, Python 3.11.1)
Hello from the pygame community. https://www.pygame.org/contribute.html
```

  - El resultado es:



- **Se crea el archivo "PruebaRotate.py"**

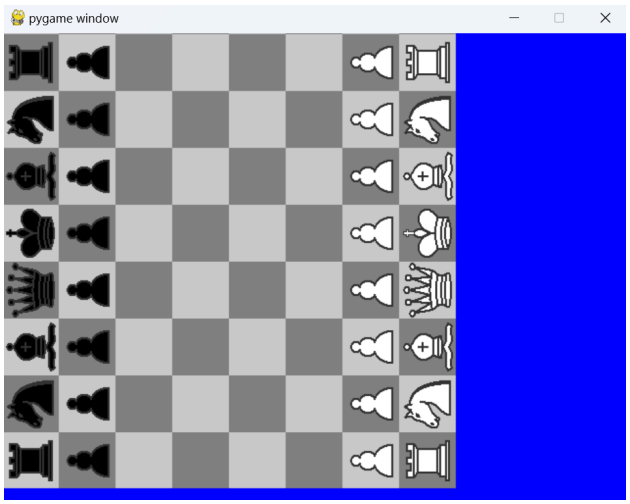
  Este es una figura extra que se creo en base a "Ejercicio2g.py", pero que se muestra el tablero de manera que ha rotado 90 grados.

```
from interpreter import draw
from chessPictures import *

wSquare = square
bSquare = square.negative()

wPawns = wSquare.under(pawn).join(bSquare.under(pawn)).
    horizontalRepeat(4)
bPawns = wPawns.negative()

wPieces = bSquare.under(rock).join(wSquare.under(knight)).join(
    bSquare.under(bishop)).join(wSquare.under(queen)).join(bSquare
    .under(king)).join(wSquare.under(bishop)).join(bSquare.under(
    knight)).join(wSquare.under(rock))
bPieces = wPieces.negative()

squares1 = bSquare.up(wSquare)
squares2 = wSquare.up(bSquare)

finalSquares = squares1.join(squares2).verticalRepeat(2).
    horizontalRepeat(4)

tabla = wPieces.up(wPawns.up(finalSquares.up(bPawns.up(bPieces))))

draw(tabla.rotate())
```

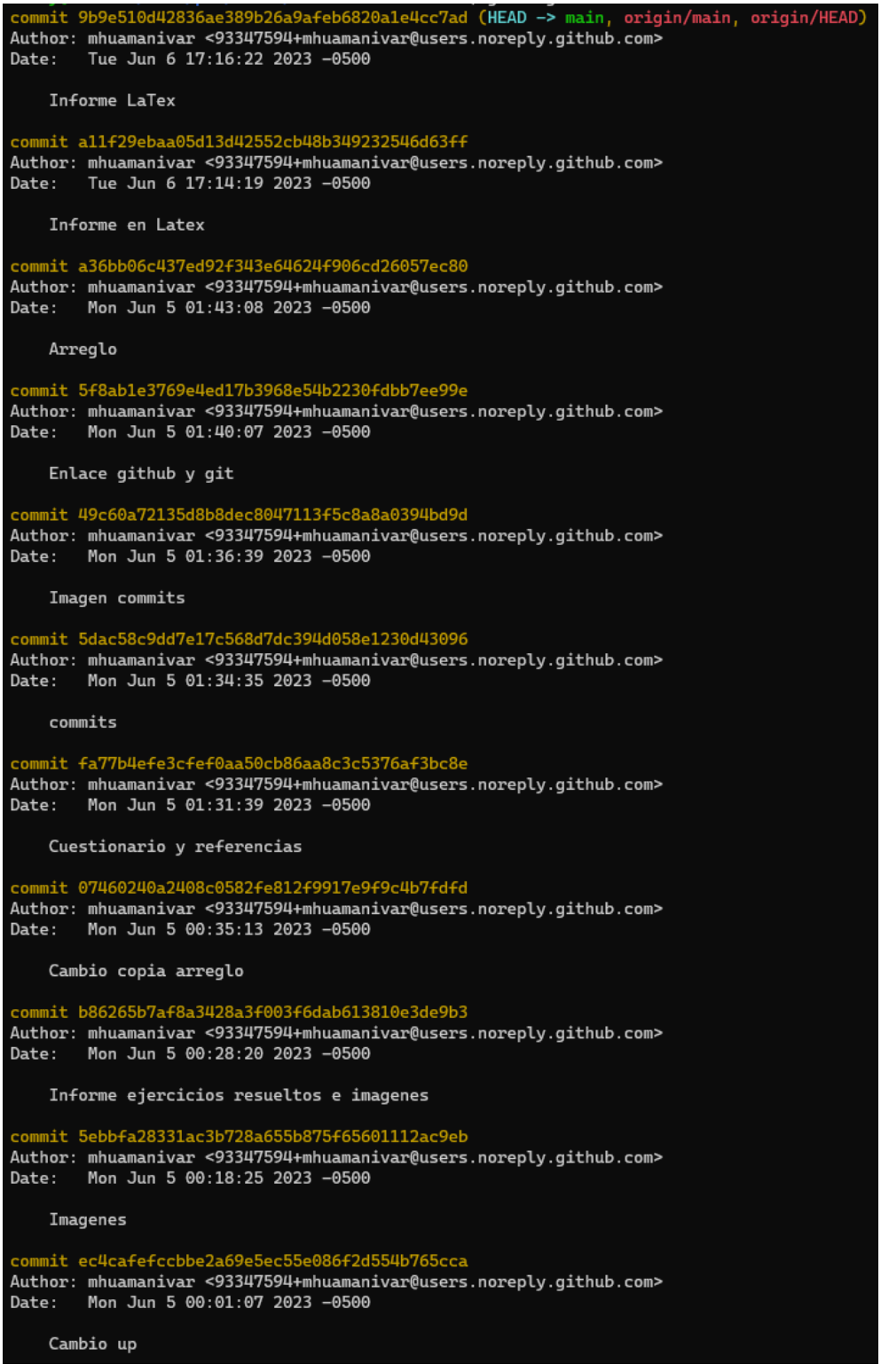
  - Se ejecuta de la siguiente manera.

```
(my_env) C:\Users\melsy\Lab04\my_env\Scripts\propuestos>python PruebaRotate.py
pygame 2.4.0 (SDL 2.26.4, Python 3.11.1)
Hello from the pygame community. https://www.pygame.org/contribute.html
```

  - El resultado es:



COMMITS MÁS IMPORTANTES



```
commit 9b9e510d42836ae389b26a9afeb6820a1e4cc7ad (HEAD -> main, origin/main, origin/HEAD)
Author: mhuamanivar <93347594+mhuamanivar@users.noreply.github.com>
Date:   Tue Jun 6 17:16:22 2023 -0500

    Informe LaTex

commit a11f29ebaa05d13d42552cb48b349232546d63ff
Author: mhuamanivar <93347594+mhuamanivar@users.noreply.github.com>
Date:   Tue Jun 6 17:14:19 2023 -0500

    Informe en Latex

commit a36bb06c437ed92f343e64624f906cd26057ec80
Author: mhuamanivar <93347594+mhuamanivar@users.noreply.github.com>
Date:   Mon Jun 5 01:43:08 2023 -0500

    Arreglo

commit 5f8ab1e3769e4ed17b3968e54b2230fdbb7ee99e
Author: mhuamanivar <93347594+mhuamanivar@users.noreply.github.com>
Date:   Mon Jun 5 01:40:07 2023 -0500

    Enlace github y git

commit 49c60a72135d8b8dec8047113f5c8a8a0394bd9d
Author: mhuamanivar <93347594+mhuamanivar@users.noreply.github.com>
Date:   Mon Jun 5 01:36:39 2023 -0500

    Imagen commits

commit 5dac58c9dd7e17c568d7dc394d058e1230d43096
Author: mhuamanivar <93347594+mhuamanivar@users.noreply.github.com>
Date:   Mon Jun 5 01:34:35 2023 -0500

    commits

commit fa77b4efe3cfef0aa50cb86aa8c3c5376af3bc8e
Author: mhuamanivar <93347594+mhuamanivar@users.noreply.github.com>
Date:   Mon Jun 5 01:31:39 2023 -0500

    Cuestionario y referencias

commit 07460240a2408c0582fe812f9917e9f9c4b7fdfd
Author: mhuamanivar <93347594+mhuamanivar@users.noreply.github.com>
Date:   Mon Jun 5 00:35:13 2023 -0500

    Cambio copia arreglo

commit b86265b7af8a3428a3f003f6dab613810e3de9b3
Author: mhuamanivar <93347594+mhuamanivar@users.noreply.github.com>
Date:   Mon Jun 5 00:28:20 2023 -0500

    Informe ejercicios resueltos e imagenes

commit 5ebbfa28331ac3b728a655b875f65601112ac9eb
Author: mhuamanivar <93347594+mhuamanivar@users.noreply.github.com>
Date:   Mon Jun 5 00:18:25 2023 -0500

    Imagenes

commit ec4cafefccbbe2a69e5ec55e086f2d554b765cca
Author: mhuamanivar <93347594+mhuamanivar@users.noreply.github.com>
Date:   Mon Jun 5 00:01:07 2023 -0500

    Cambio up
```

```
commit 53e5faf90cce10b8d19582522de68645e641e8ee
Author: mhuamanivar <93347594+mhuamanivar@users.noreply.github.com>
Date:   Sun Jun 4 15:25:40 2023 -0500

    Instalación y Ejercicios resueltos

commit ae519d325f5914a8c1b6a7306b86a50e5fe3cb9c
Author: mhuamanivar <93347594+mhuamanivar@users.noreply.github.com>
Date:   Sat Jun 3 23:58:05 2023 -0500

    Comentario

commit 1444453b9d468efb49fb03f047fe35eb7d11ae73
Author: mhuamanivar <93347594+mhuamanivar@users.noreply.github.com>
Date:   Sat Jun 3 19:12:21 2023 -0500

    PruebaRotate

commit e24caefd42f47a60cba76efefc413e45cb59b1e5
Author: mhuamanivar <93347594+mhuamanivar@users.noreply.github.com>
Date:   Sat Jun 3 19:11:55 2023 -0500

    Ejercicio2g

commit 615cdf5690907eb7be9c8725f1aec42b5fa43370
Author: mhuamanivar <93347594+mhuamanivar@users.noreply.github.com>
Date:   Sat Jun 3 19:11:24 2023 -0500

    Ejercicio2f

commit 17009c80c1f582ba1899700e96fbdb487cff0b66
Author: mhuamanivar <93347594+mhuamanivar@users.noreply.github.com>
Date:   Sat Jun 3 19:10:59 2023 -0500

    Ejercicio2e

commit 6a414102e08741fa0c85ae28a9d2f97380977c1a
Author: mhuamanivar <93347594+mhuamanivar@users.noreply.github.com>
Date:   Sat Jun 3 19:09:42 2023 -0500

    Ejercicio2d

commit 5759026299ac8ddc01566564a771a336c0aef06f
Author: mhuamanivar <93347594+mhuamanivar@users.noreply.github.com>
Date:   Sat Jun 3 19:09:11 2023 -0500

    Ejercicio2c

commit 7596897f95626397b61444840c7e387ac56c1925
Author: mhuamanivar <93347594+mhuamanivar@users.noreply.github.com>
Date:   Sat Jun 3 19:08:43 2023 -0500

    Ejercicio2b

commit 1c201565f021e4e3bb15f40873891a7a35003bcd
Author: mhuamanivar <93347594+mhuamanivar@users.noreply.github.com>
Date:   Sat Jun 3 19:07:56 2023 -0500

    Ejercicio2a

commit 4abed37385b93b0815b8e2f3d7df7d564f6f4538
Author: mhuamanivar <93347594+mhuamanivar@users.noreply.github.com>
Date:   Sat Jun 3 18:02:39 2023 -0500

    Función extra, rotate
```

```
commit ffe30b289134a7333ce922234912654b00e08149
Author: mhuamanivar <93347594+mhuamanivar@users.noreply.github.com>
Date:   Sat Jun 3 17:54:48 2023 -0500

    Funciones repeat

commit fa99d7e8ccfd4b6682502fd25386c5e63397fe46
Author: mhuamanivar <93347594+mhuamanivar@users.noreply.github.com>
Date:   Sat Jun 3 17:48:42 2023 -0500

    Función up y under

commit d56f298b0d3bd8ae73fa979e2e45484bff045ec2
Author: mhuamanivar <93347594+mhuamanivar@users.noreply.github.com>
Date:   Sat Jun 3 17:23:59 2023 -0500

    Función negative y join

commit 3afe6a8b9d43727de6dada9522215442220de622
Author: mhuamanivar <93347594+mhuamanivar@users.noreply.github.com>
Date:   Sat Jun 3 15:08:31 2023 -0500

    Clase chessPictures.py

commit 0408797f4b2d7025585258c1047f512d01df1a79
Author: mhuamanivar <93347594+mhuamanivar@users.noreply.github.com>
Date:   Sat Jun 3 15:08:07 2023 -0500

    Clase colors.py

commit 9ce5ed0c6d0dbaab5c28951734c393900163e994
Author: mhuamanivar <93347594+mhuamanivar@users.noreply.github.com>
Date:   Sat Jun 3 15:07:45 2023 -0500

    Clase interpreter.py

commit 1af7f0bf7d320e18c4cc21454751b2cbff4fa4d9
Author: mhuamanivar <93347594+mhuamanivar@users.noreply.github.com>
Date:   Sat Jun 3 15:07:12 2023 -0500

    Clase pieces.py

commit 1bfadc9cf739d5fe524c323de4e1fad5383a10a4
Author: mhuamanivar <93347594+mhuamanivar@users.noreply.github.com>
Date:   Sat Jun 3 15:06:00 2023 -0500

    Clase picture - mirrors

commit 031652f7d7e4f9d63c715d7bfb85aa1597c75849
Author: mhuamanivar <93347594+mhuamanivar@users.noreply.github.com>
Date:   Wed May 31 12:47:25 2023 -0500

    Eliminando carpeta

commit f66287679ccf93bddd283b6c4e3d366c5fb08290
Author: mhuamanivar <93347594+mhuamanivar@users.noreply.github.com>
Date:   Wed May 31 12:35:23 2023 -0500

    Ejercicios resueltos

commit 38ba07941bd3ae85ce2feb00849d1ea0a4536119
Author: mhuamanivar <93347594+mhuamanivar@users.noreply.github.com>
Date:   Wed May 31 12:33:13 2023 -0500

    Prueba Hola Mundo
```

CUESTIONARIO

- **¿Qué son los archivos ".pyc"?**

  Son los archivos compilados de Python, es una versión de un módulo que ya ha sido compilado en bytes con éxito. Sin embargo, si la compilación falla, entonces el archivo ".pyc" sera reconocido como inválido. En los archivos ".pyc" se deben tener en cuenta ciertas características:

  - Cuando el intérprete de Python es invocado con flag "-o", entonces los archivos que son compilados son optimizados y eliminan datos no importantes, por lo que ya no son ".pyc", sino que son ".pyo".
  - El programa no compila más rápido si es que ejecutar los ".pyc", lo único en lo que son rápidos es la velocidad con la que se cargan.
  - Es posible que existan archivos ".pyc", sin la necesidad de haber modulos con el nombre que tiene el ".pyc", esto puede usarse para distribuir una librería de Python.
  - El módulo "compile all" puede crear archivos ".pyc" para todos los módulos de un directorio.
  - Python verifica la fecha de modificación del código con la versión compilada para ver si está desactualizada y necesita ser recompilado, ya que en los archivos ".pyc" se puede visualizar la versión con la que ha sido compilado el módulo.

- **¿Para qué sirve el directorio pycache?**

  El directorio "__pycache__" sirve para acelerar la carga de módulos, puesto que Python almacena en caché la versión compilada de cada módulo bajo el nombre de "nombre_de_modulo.version.pyc", donde se muestra la versión del formato con el que se ha compilado el archivo, el cual generalmente es el número de versíon de Python. No se recomienda borrar a menudo estos archivos ni suprimir la creación de estos ya que como han sido compilados, se estaría restando a su función principal que es acelerar el proceso de la carga de módulos.

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

  El subguión tiene una gran variedad de usos en Python, además de una representación cuando se habla de atributos o métodos de una clase. Entre ellos tenemos:

  - Para omitir valores: Es decir, podemos usarlo como una variable "_" la cual solo la utilizaremos una vez, por ejemplo, en las iteraciones del "for". Por otro lado, cuando tenemos muchos valores y queremos seleccionar solo unos cuantos e ignorar un rango de ellos, entonces podemos utilizar "*_".
  - Como placeholder: Esto quiere decir que cuando nos encontremos en el intérprete de Python podemos guardar la anterior respuesta para utilizarla en una siguiente operación.
  - Como namespace: Cuando queremos utilizar un nombre para una variable, pero esta ya es una palabra reservada para Python, entonces podemos utilzar "_" al final del nombre de la palabra.
  - Para números: Podemos utilizar "_" para expresar números muy grandes, y queremos separar cada tres cifrar para un mejor entendimiento.
  - Representación en clases: Cuando utilizamos "_" dos veces antes y después del nombre de un atributo, de manera "__nombre_de_atributo__", entonces esto representa un atributo privado de la clase. Por otro lado, para los métodos, al utilizar de la siguiente manera "__nombre_del_metodo__" quiere decir que estos son métodos que se pueden sobreescribir facilmente en las clases que heredan, siendo esta usada en la clase padre.

REFERENCIAS

- https://www.w3schools.com/python/python_reference.asp
- https://docs.python.org/3/tutorial/
- https://docs.python.org/release/1.5.1p1/tut/node43.html
- https://docs.python.org/3/tutorial/modules.html
- https://pywombat.com/articles/guion-bajo-python