

Laboratorio 5

Python

Apellidos y Nombres: Vilca Huarca, Laura Lucía

Grupo de Laboratorio: A

I. Objetivos

- Practicar los principios de programación usando Python
- Mostrar un ejemplo de separación de intereses en clases: el modelo (lista de strings) de su vista (dibujo de gráficos).

II. Temas a tratar

- Listas
- Ciclos
- Programación orientada a objetos
- ¿Programación funcional?

III. Marco teorico

https://www.w3schools.com/python/python_reference.asp

<https://docs.python.org/3/tutorial/>

Virtual Environment

Muchos lenguajes/entornos de programación como Java o Node JS (el lenguaje es ECMAScript) usan paquetes para reusar código de otros y esto nos permite construir software más complejo. En el caso de Node JS los paquetes son instalados en el directorio de trabajo y no de manera global; el registro de estos paquetes y sus versiones se almacenan en formato json en el archivo **package.json**. Este modo de trabajo es sumamente útil porque nos permite tener distintos proyectos, que trabajan con distintas bibliotecas, de distintas versiones, en la misma máquina, sin que existan conflictos. Si desea compartir el proyecto con alguien y que pueda seguir trabajando con las mismas bibliotecas y versiones de estas que se usaron en el proyecto original, se podrá compartir el archivo package.json y simplemente llamar a “npm install”, instrucción que usará lo indicado en package.json para instalar las bibliotecas adecuadas para el proyecto.

El lenguaje Java usa ant y maven, junto con archivos xml para realizar estas tareas, pero quizá lo mejor de conseguir este efecto de entorno virtual, sería utilizar contenedores como docker.

En python se puede usar el programa virtualenv, para crear este espacio de trabajo. En él podremos instalar paquetes de manera local al mismo estilo del manejador de paquetes de Node JS, npm. En el caso de python el manejador de paquetes será pip.

IV. Actividades

1. Instale Python.
2. Cree su entorno de trabajo:

```
mkdir lab04 && cd lab04
virtualenv -p python3 pyl (python3 -m venv pyl)
mkdir src
cd src
git init .
```
3. Active el entorno virtual

```
source ../bin/activate (.\pyl\Scripts\activate)
(pyl) C:\>
```
4. Ahora podrá instalar las bibliotecas que necesite usando pip

```
pip install pygame
python -m pip show pygame
```
5. Cuando quiera terminar de usar el entorno virtual ejecute

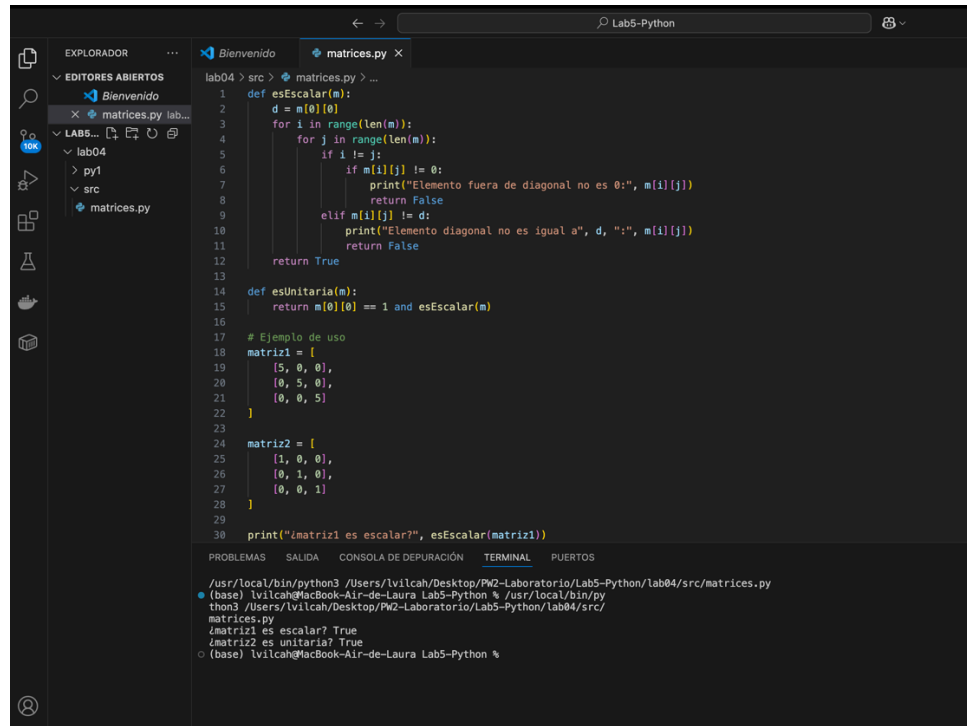
```
deactivate
C:\>
```

V. Ejercicios Resueltos

Resuelva de manera individual los siguientes ejercicios sobre matrices de tamaño NxN:

1. Determine si una matriz es escalar

```
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
```



```

lab04 > src > matrices.py > ...
1 def esEscalar(m):
2     d = m[0][0]
3     for i in range(len(m)):
4         for j in range(len(m)):
5             if i != j:
6                 if m[i][j] != 0:
7                     print("Elemento fuera de diagonal no es 0:", m[i][j])
8                     return False
9             elif m[i][j] != d:
10                print("Elemento diagonal no es igual a", d, ":", m[i][j])
11                return False
12    return True
13
14 def esUnitaria(m):
15     return m[0][0] == 1 and esEscalar(m)
16
17 # Ejemplo de uso
18 matriz1 = [
19     [5, 0, 0],
20     [0, 5, 0],
21     [0, 0, 5]
22 ]
23
24 matriz2 = [
25     [1, 0, 0],
26     [0, 1, 0],
27     [0, 0, 1]
28 ]
29
30 print("¿matriz1 es escalar?", esEscalar(matriz1))

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS
(base) /usr/local/bin/python3 /Users/lvilcah/Desktop/PW2-Laboratorio/Lab5-Python/lab04/src/matrices.py
thon3 /Users/lvilcah/Desktop/PW2-Laboratorio/Lab5-Python/lab04/src/
matrices.py
¿matriz1 es escalar? True
¿matriz2 es unitaria? True
(base) /usr/local/bin/python3 /Users/lvilcah/Desktop/PW2-Laboratorio/Lab5-Python/lab04/src/
matrices.py

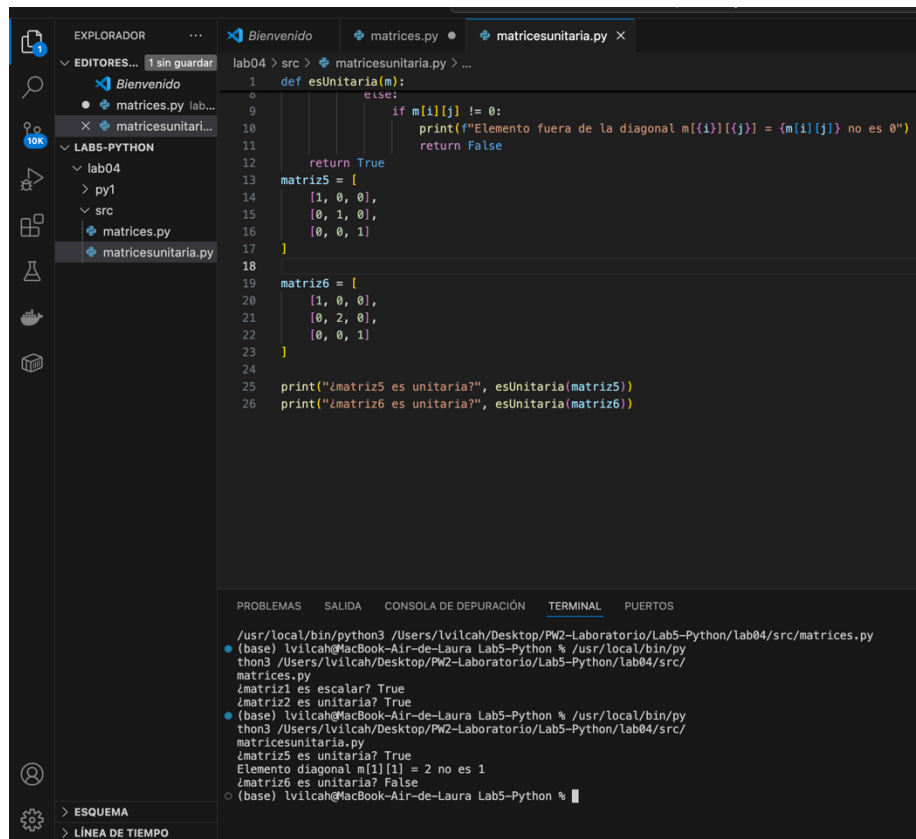
```

2. Determine si una matriz es unitaria

```

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

```



```

lab04 > src > matricesunitaria.py > ...
1 def esUnitaria(m):
2     if not esEscalar(m):
3         return False
4     else:
5         if m[0][0] != 1:
6             print(f"Elemento fuera de la diagonal m[{i}][{j}] = {m[i][j]} no es 0")
7             return False
8         return True
9
10 matriz5 = [
11     [1, 0, 0],
12     [0, 1, 0],
13     [0, 0, 1]
14 ]
15
16 matriz6 = [
17     [1, 0, 0],
18     [0, 2, 0],
19     [0, 0, 1]
20 ]
21
22 print("¿matriz5 es unitaria?", esUnitaria(matriz5))
23 print("¿matriz6 es unitaria?", esUnitaria(matriz6))

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS
/usr/local/bin/python3 /Users/lvilcah/Desktop/PW2-Laboratorio/Lab5-Python/lab04/src/matrices.py
(base) /usr/local/bin/python3 /Users/lvilcah/Desktop/PW2-Laboratorio/Lab5-Python/lab04/src/matrices.py
¿matriz1 es escalar? True
¿matriz2 es unitaria? True
(base) /usr/local/bin/python3 /Users/lvilcah/Desktop/PW2-Laboratorio/Lab5-Python/lab04/src/matricesunitaria.py
¿matriz5 es unitaria? True
Elemento diagonal m[1][1] = 2 no es 1
¿matriz6 es unitaria? False
(base) /usr/local/bin/python3 /Users/lvilcah/Desktop/PW2-Laboratorio/Lab5-Python/lab04/src/matricesunitaria.py







```

VI. Ejercicios Propuestos

En esta tarea, individualmente 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.

Agregar en el informe final el código fuente y una imagen de los Commits en github.

Con el código proporcionado usted dispondrá de varios objetos de tipo Picture para poder realizar su tarea:

| | |
|---|--------|
|  | rock |
|  | knight |
|  | bishop |
|  | queen |
|  | king |
|  | square |

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:

`_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:

1. verticalMirror: Devuelve el espejo vertical de la imagen
2. horizontalMirror: Devuelve el espejo horizontal de la imagen
3. negative: Devuelve un negativo de la imagen
4. join: Devuelve una nueva figura poniendo la figura del argumento al lado derecho de la figura actual
5. up: Devuelve una nueva figura poniendo la figura recibida como argumento, encima de la figura actual
6. under: Devuelve una nueva figura poniendo la figura recibida como argumento, sobre la figura actual
7. horizontalRepeat, Devuelve una nueva figura repitiendo la figura actual al costado la cantidad de veces que indique el valor de n
8. verticalRepeat Devuelve una nueva figura repitiendo la figura actual debajo, la cantidad de veces que indique el valor de n

Tenga en cuenta que para implementar todos estos métodos, sólo deberá trabajar sobre la representación interna de un Picture, es decir su atributo img.

Para dibujar una objeto Picture bastará importar el método draw de la biblioteca interpreter y usarlo de la siguiente manera:

```
>>> from chessPictures import *
>>> from interpreter import draw
>>> draw(rock)
```



Considerar el repositorio:

<https://github.com/rescobedoq/pw2/tree/main/labs/lab04/Tarea-del-Ajedrez>

Ejercicios

Para resolver los siguientes ejercicios sólo está permitido usar **ciclos, condicionales, definición de listas por comprensión, sublistas, map, join, (+), lambda, zip, append, pop, range.**

1. 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.

Codigo de Picture.py

```
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:
        fila_invertida = value[::-1]
        vertical.append(fila_invertida)
    return Picture(vertical)

def horizontalMirror(self):
    """ Devuelve el espejo horizontal de la imagen """
    resultado = []
    for i in range(len(self.img)-1, -1, -1):
        resultado.append(self.img[i])
    return Picture(resultado)

def negative(self):
    """ Devuelve un negativo de la imagen """
    resultado = []
    for fila in self.img:
        nueva_fila = ""
        for caracter in fila:
            nuevo_color = self._invColor(caracter)
            nueva_fila += nuevo_color
        resultado.append(nueva_fila)
    return Picture(resultado)

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

def up(self, p):
    resultado = []
    for fila in p.img:

```

```

        resultado.append(fila)
    for fila in self.img:
        resultado.append(fila)
    return Picture(resultado)

def under(self, p):
    """ Devuelve una nueva figura poniendo la figura p sobre la
        figura actual """
    resultado = []
    for i in range(len(self.img)):
        nueva_fila = ""
        for j in range(len(self.img[0])):
            if p.img[i][j] != " ":
                nueva_fila += p.img[i][j]
            else:
                nueva_fila += self.img[i][j]
        resultado.append(nueva_fila)
    return Picture(resultado)

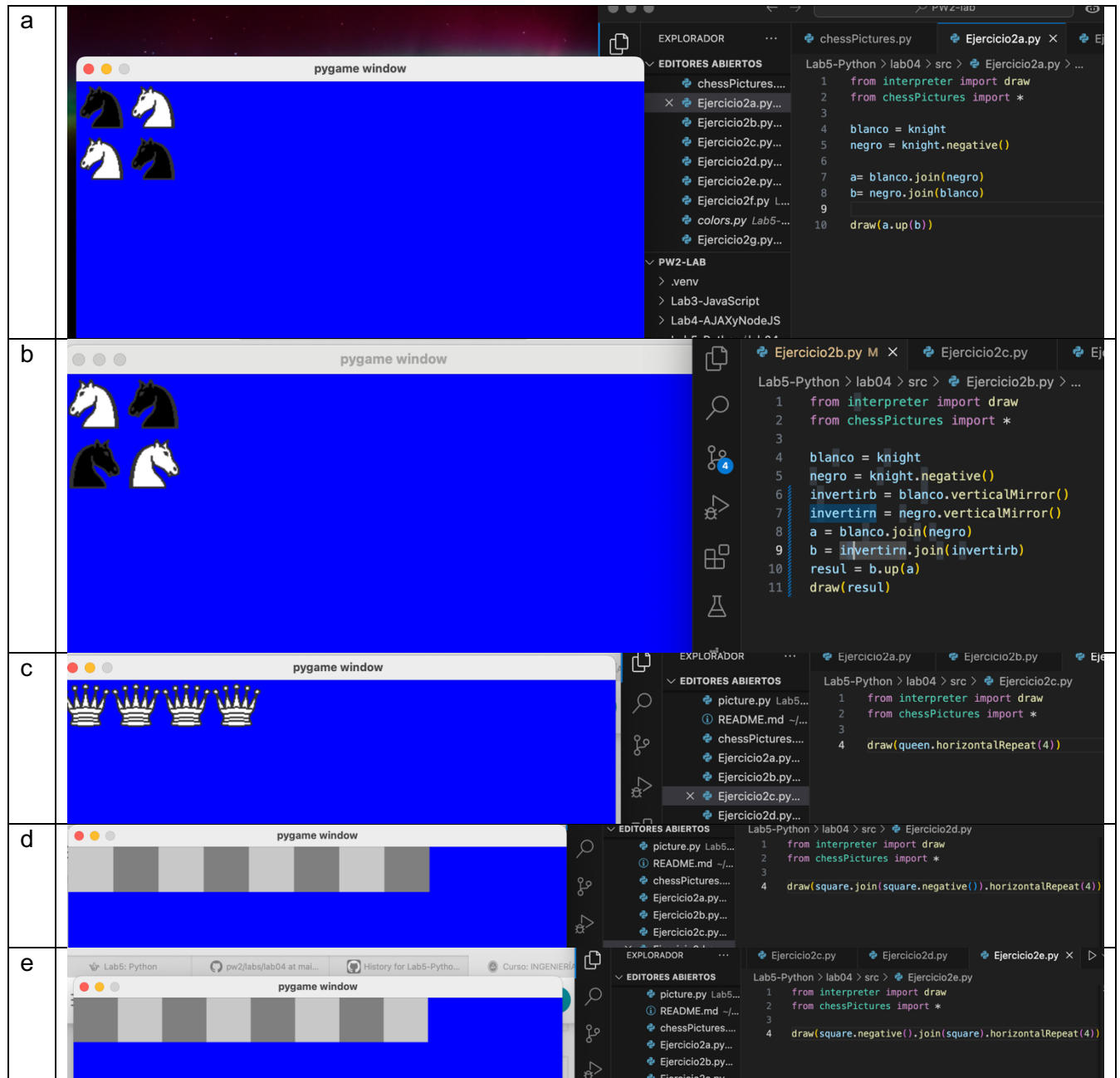
def horizontalRepeat(self, n):
    """ Devuelve una nueva figura repitiendo la figura actual al costado
        la cantidad de veces que indique el valor de n """
    resultado = []
    for fila in self.img:
        nueva_fila = ""
        for i in range(n):
            nueva_fila += fila
        resultado.append(nueva_fila)
    return Picture(resultado)

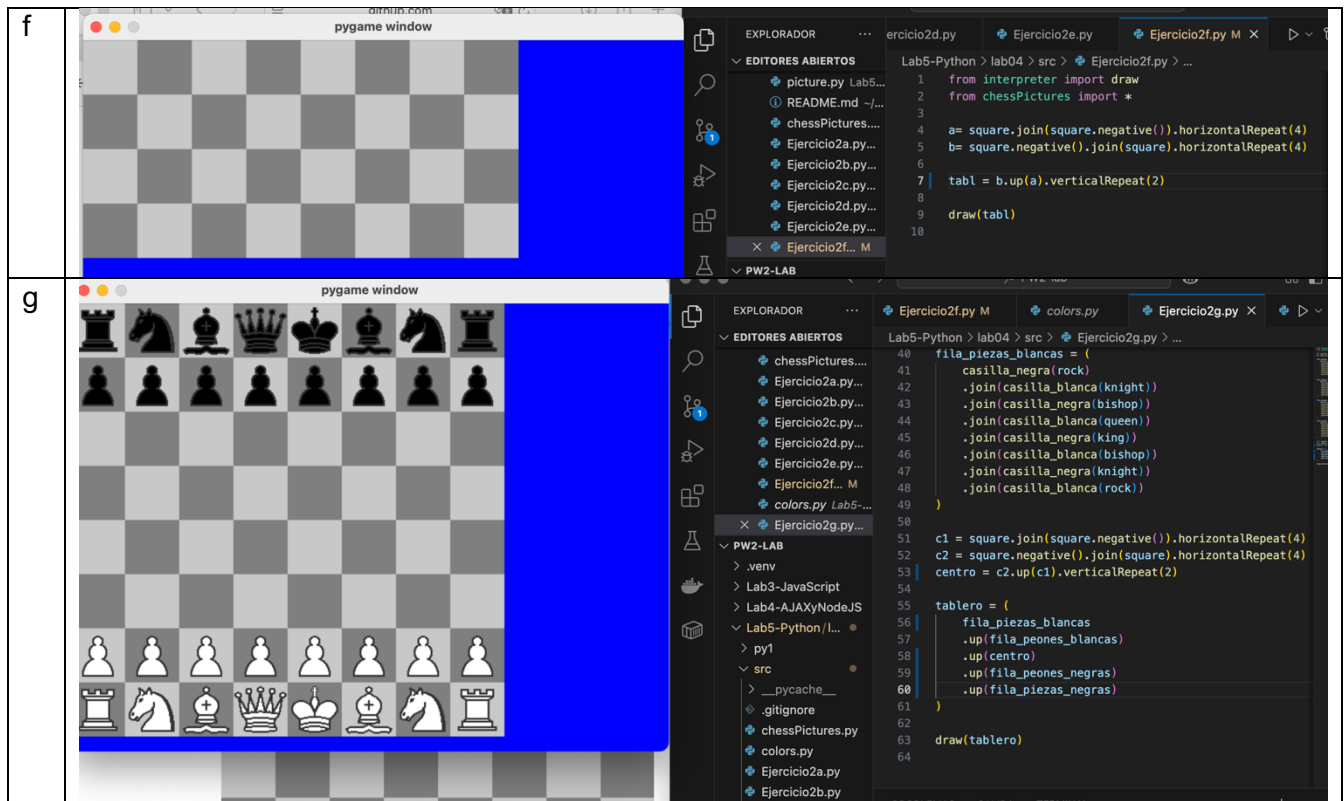
def verticalRepeat(self, n):
    resultado = []
    for i in range(n):
        for fila in self.img:
            resultado.append(fila)
    return Picture(resultado)

def rotate(self):
    """Devuelve una figura rotada en 90 grados, puede ser en sentido horario
    o antihorario"""
    resultado = []
    for columna in range(len(self.img[0])):
        nueva_fila = ""
        for fila in range(len(self.img) - 1, -1, -1):
            nueva_fila += self.img[fila][columna]
        resultado.append(nueva_fila)
    return Picture(resultado)

```

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





3. Sacar un pantallazo de los commits en github (<https://github.com/Lau230595/PW2-lab.git>)

