

Informe de Laboratorio 05

Tema: Python

Nota

Estudiante	Escuela	Asignatura
Fernando Miguel Garambel Marín fgarambel@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Laboratorio de Programación Web 2 Semestre: III Código: 1701212

Laboratorio	Tema	Duración
05	Python	04 horas


Semestre académico	Fecha de inicio	Fecha de entrega
2024 - A	Del 9 de abril 2024	Al 25 de mayo 2024

1. Actividades

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

2. 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.
- 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:
- verticalMirror: Devuelve el espejo vertical de la imagen
- horizontalMirror: Devuelve el espejo horizontal de la imagen
- negative: Devuelve un negativo de la imagen
- join: Devuelve una nueva figura poniendo la figura del argumento al lado derecho de la figura actual
- up: Devuelve una nueva figura poniendo la figura recibida como argumento, encima de la figura actual
- under: Devuelve una nueva figura poniendo la figura recibida como argumento, sobre la figura actual
- horizontalRepeat, Devuelve una nueva figura repitiendo la figura actual al costado la cantidad de veces que indique el valor de n
- 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>

3. Equipos, materiales y temas utilizados

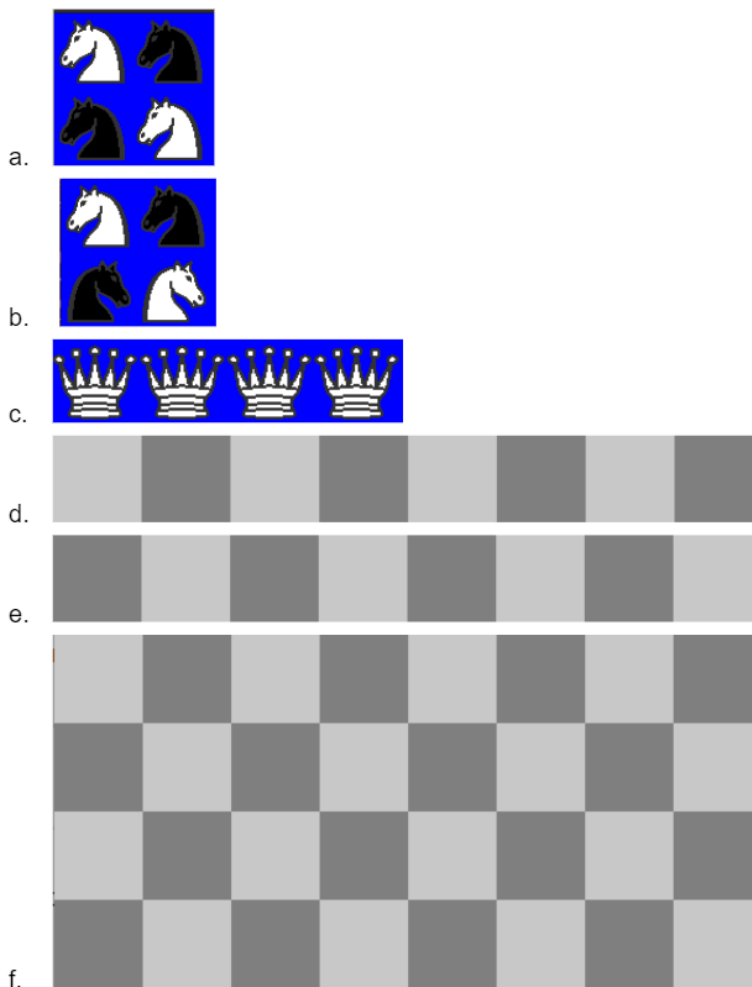
- Sistema operativo de 64 bits, procesador basado en x64.
- Latex.
- git version 2.41.0.windows.1
- Cuenta en GitHub con el correo institucional.

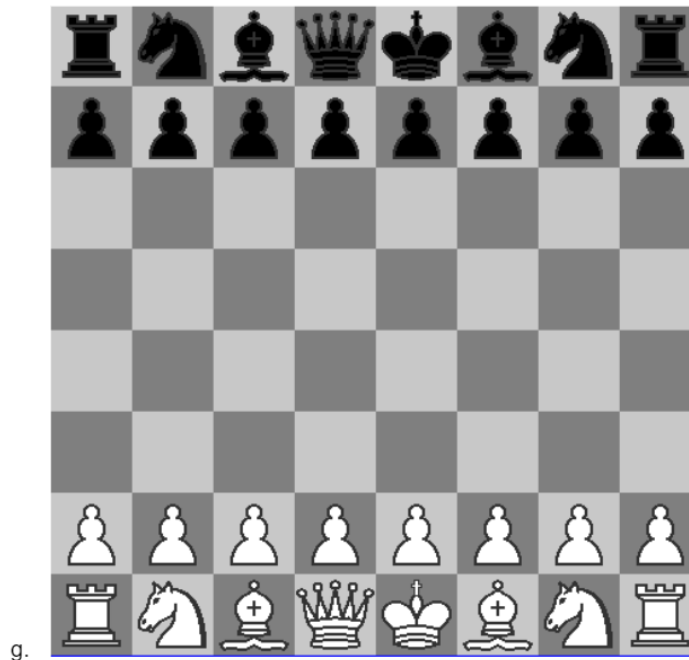
4. URL Github, Video

- URL del Repositorio GitHub para clonar o recuperar.
- <https://github.com/FernandoGarambelM/Python-Ajedrez.git>
- URL para el video flipgrid.
- <https://flip.com/s/DBLovQ25eDFg>

5. 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.
- 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):





6. Implementación de funciones de Picture

Listing 1: Función verticalMirror(self)

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

- Devuelve el espejo vertical de la imagen

Listing 2: Función horizontalMirror(self)

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

- Devuelve el espejo horizontal de la imagen

Listing 3: Función negative(self)

```
def negative(self):
    negative = []
    for value in self.img:
        negative.append([self._invColor(color) for color in value])
    return Picture(negative)
```

- Devuelve un negativo de la imagen

Listing 4: Función join(self, p)

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

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

Listing 5: Función up(self, p)

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

- Devuelve una nueva figura poniendo la figura actual por encima de la del argumento

Listing 6: Función under(self, p)

```
def under(self, p):  
    undering = [list(line) for line in self.img]  
    for i in range(len(self.img)):  
        for j in range(len(p.img[i])):  
            if p.img[i][j] != " "  
                undering[i][j] = p.img[i][j]  
    undering = ["".join(line) for line in undering]  
    return Picture(undering)
```

- Devuelve una nueva figura poniendo la figura p sobre la figura actual

Listing 7: Función horizontalRepeat(self, n)

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

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

Listing 8: verticalRepeat(self, n)

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

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

Listing 9: rotate(self)

```
def rotate(self):  
    b = []  
    selfLen = len(self.img)  
    for i in range(selfLen):  
        a = ""  
        for value in self.img:  
            a += value[selfLen - 1 - i]  
        b.append(a)  
    return Picture(b)
```

- Devuelve una figura rotada en 90 grados, en sentido antihorario

Listing 10: Código de Picture

```
1 from colors import *  
2 class Picture:  
3     def __init__(self, img):  
4         self.img = img  
5  
6     def __eq__(self, other):  
7         return self.img == other.img  
8  
9     def _invColor(self, color):  
10        if color not in inverter:  
11            return color  
12        return inverter[color]  
13  
14    def verticalMirror(self):  
15        """ Devuelve el espejo vertical de la imagen """  
16        vertical = []  
17        for value in self.img:  
18            vertical.append(value[::-1])  
19        return Picture(vertical)  
20  
21    def horizontalMirror(self):  
22        """ Devuelve el espejo horizontal de la imagen """  
23        horizontal = []  
24        for value in self.img[::-1]:  
25            horizontal.append(value)  
26        return Picture(horizontal)
```

```
27
28 def negative(self):
29     """ Devuelve un negativo de la imagen """
30     negative = []
31     for value in self.img:
32         negative.append([self._invColor(color) for color in value])
33     return Picture(negative)
34
35 def join(self, p):
36     """ Devuelve una nueva figura poniendo la figura del argumento
37         al lado derecho de la figura actual """
38     joinedImg = []
39     for i in range(len(self.img)):
40         joinedRow = "".join(self.img[i]) + "".join(p.img[i])
41         joinedImg.append(joinedRow)
42     return Picture(joinedImg)
43
44 def up(self, p):
45     uping = []
46     for value in self.img:
47         uping.append(value)
48     for value in p.img:
49         uping.append(value)
50     return Picture(uping)
51
52 def under(self, p):
53     """ Devuelve una nueva figura poniendo la figura p sobre la
54         figura actual """
55     undering = [list(line) for line in self.img]
56     for i in range(len(self.img)):
57         for j in range(len(p.img[i])):
58             if p.img[i][j] != " ":
59                 undering[i][j] = p.img[i][j]
60     undering = ["".join(line) for line in undering]
61     return Picture(undering)
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     horizontal = []
67     for value in self.img:
68         horizontal.append(value * n)
69     return Picture(horizontal)
70
71 def verticalRepeat(self, n):
72     vertical = []
73     for i in range(n):
74         for value in self.img:
75             vertical.append(value)
76     return Picture(vertical)
77
78 #Extra: Solo para realmente viciosos
79 def rotate(self):
80     """Devuelve una figura rotada en 90 grados, puede ser en sentido horario
81         o antihorario"""
82     b = []
```



```

83 selfLen = len(self.img)
84 for i in range(selfLen):
85     a = ""
86     for value in self.img:
87         a += value[selfLen - 1 - i]
88     b.append(a)
89 return Picture(b)

```

7. Ejercicios

■ Primer ejercicio

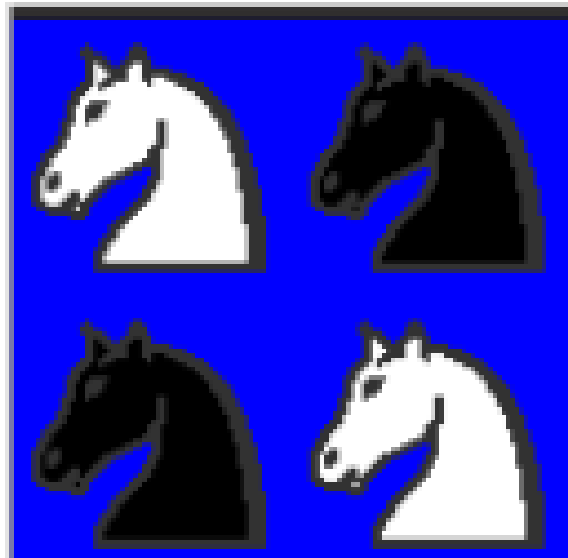
Listing 11: Código del Ejercicio2a

```

1 from interpreter import draw
2 from chessPictures import *
3 knight = Picture(KNIGHT)
4 knightN = knight.negative()
5 knightPar = knight.join(knightN)
6 knightImpar = knightPar.negative()
7 tablero = knightPar.up(knightImpar)
8 draw(tablero)

```

■ Resultado



■ Segundo ejercicio

Listing 12: Código del Ejercicio2b

```

1 from interpreter import draw
2 from chessPictures import *

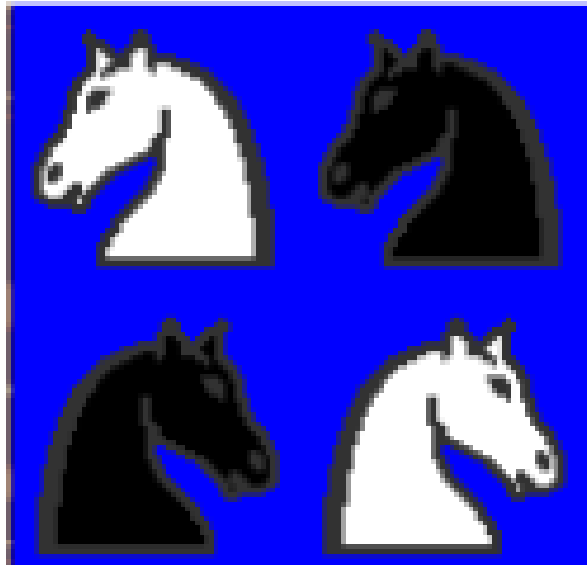
```

```

3 knight = Picture(KNIGHT)
4 knightN = knight.negative()
5 knightPar = knight.join(knightN)
6 knightImpar = knightPar.negative()
7 knightParV = knightPar.verticalMirror()
8 tablero = knightPar.up(knightParV)
9 draw(tablero)

```

■ Resultado



■ Tercer ejercicio

Listing 13: Código del Ejercicio2c

```

1 from interpreter import draw
2 from chessPictures import *
3 queen = Picture(QUEEN)
4 fourQueens = queen.horizontalRepeat(4)
5 draw(fourQueens)

```

■ Resultado



■ Cuarto ejercicio

Listing 14: Código del Ejercicio2d

```
1 from interpreter import draw
2 from chessPictures import *
3 square = Picture(SQUARE)
4 blackSquare = square.negative()
5 bloque = square.join(blackSquare)
6 fila = bloque.horizontalRepeat(4)
7 draw(fila)
```

■ Resultado



■ Quinto ejercicio

Listing 15: Código del Ejercicio2e

```
1 from interpreter import draw
2 from chessPictures import *
3 square = Picture(SQUARE)
4 blackSquare = square.negative()
5 bloque = square.join(blackSquare)
6 fila = bloque.horizontalRepeat(4)
7 draw(fila.negative())
```

■ Resultado

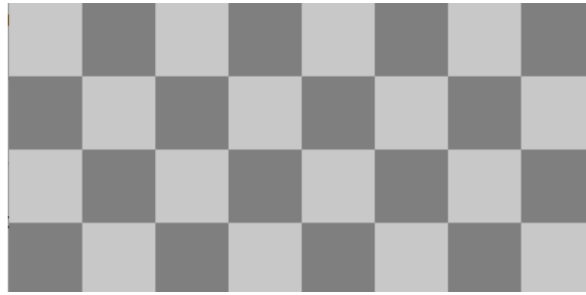


■ Sexto ejercicio

Listing 16: Código del Ejercicio2f

```
1 from interpreter import draw
2 from chessPictures import *
3 square = Picture(SQUARE)
4 blackSquare = square.negative()
5 bloque = square.join(blackSquare)
6 fila = bloque.horizontalRepeat(4)
7 filaNegativa = fila.negative()
8 tablero = fila.up(filaNegativa).verticalRepeat(2)
9 draw(tablero)
```

■ Resultado

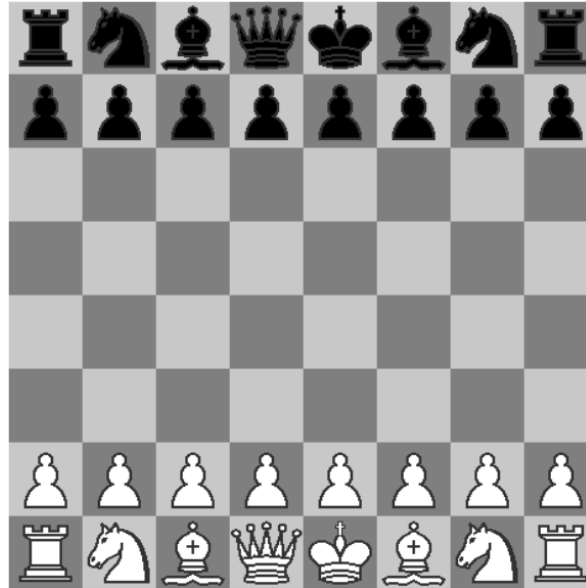


■ Séptimo ejercicio

Listing 17: Código del Ejercicio2g

```
1 from interpreter import draw
2 from chessPictures import *
3
4 bishop = Picture(BISHOP)
5 blackBishop = bishop.negative()
6 king = Picture(KING)
7 blackKing = king.negative()
8 knight = Picture(KNIGHT)
9 blackKnight = knight.negative()
10 pawn = Picture(PAWN)
11 blackPawn = pawn.negative()
12 queen = Picture(QUEEN)
13 blackQueen = queen.negative()
14 rock = Picture(ROCK)
15 blackRock = rock.negative()
16 square = Picture(SQUARE)
17 blackSquare = square.negative()
18
19 bloque = square.join(blackSquare)
20 fila = bloque.horizontalRepeat(4)
21 filaNegativa = fila.negative()
22
23 filaPiezasBlancas =
24     rock.join(knight).join(bishop).join(queen).join(king).join(bishop).join(knight).join(rock)
25 filaPeones = pawn.horizontalRepeat(8)
26 filaPeonesNegros = filaPeones.negative()
27 filaPiezasNegras = filaPiezasBlancas.negative()
28
29 primeraFila = fila.under(filaPiezasNegras)
30 segundaFila = filaNegativa.under(filaPeonesNegros)
31 tableroNegras = primeraFila.up(segundaFila)
32
33 tableroSinFichas = fila.up(filaNegativa).verticalRepeat(2)
34
35 penultimaFila = fila.under(filaPeones)
36 ultimaFila = filaNegativa.under(filaPiezasBlancas)
37 tableroBlancas = penultimaFila.up(ultimaFila)
38
39 tablero = tableroNegras.up(tableroSinFichas.up(tableroBlancas))
40
41 draw(tablero)
```

- Resultado



8. Referencias

- https://www.w3schools.com/python/python_reference.asp
- <https://docs.python.org/3/tutorial/complexandpowerful>