Informe de Laboratorio 04

Tema: Python

Estudiante	Escuela	Asignatura
Sebastian Diaz Ticona sdiazt@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Programación Web 2 Semestre: III

Laboratorio	Tema	Duración
04	Python	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - A	Del 22 Abril 2023	Al 07 Junio 2023

1. Tarea

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

  a) 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 pregunta.

  b) Usando únicamente los métodos de los objetos de la clase `Picture`, dibuje las siguientes figuras (invoque a `draw`):

  (a)

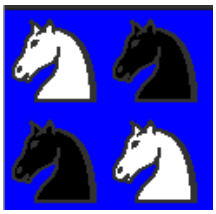

Figura 1

  (b)

Figura 2

(c)

Figura 3

(d)

Figura 4

(e)

Figura 5

(f)

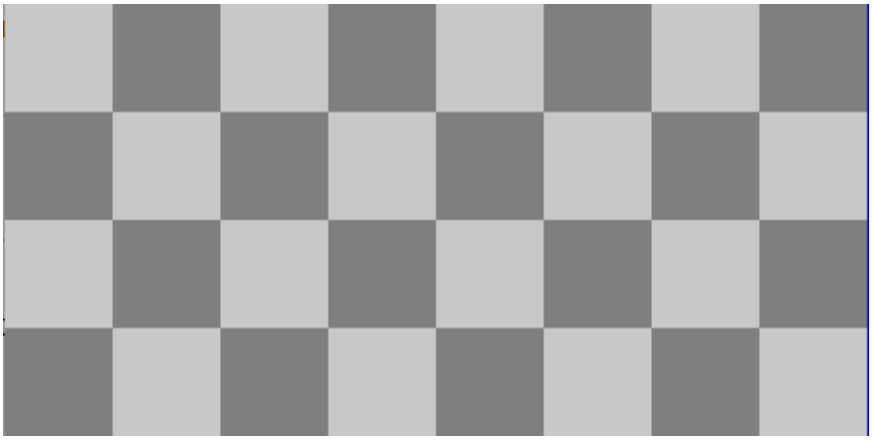

Figura 6

(g)

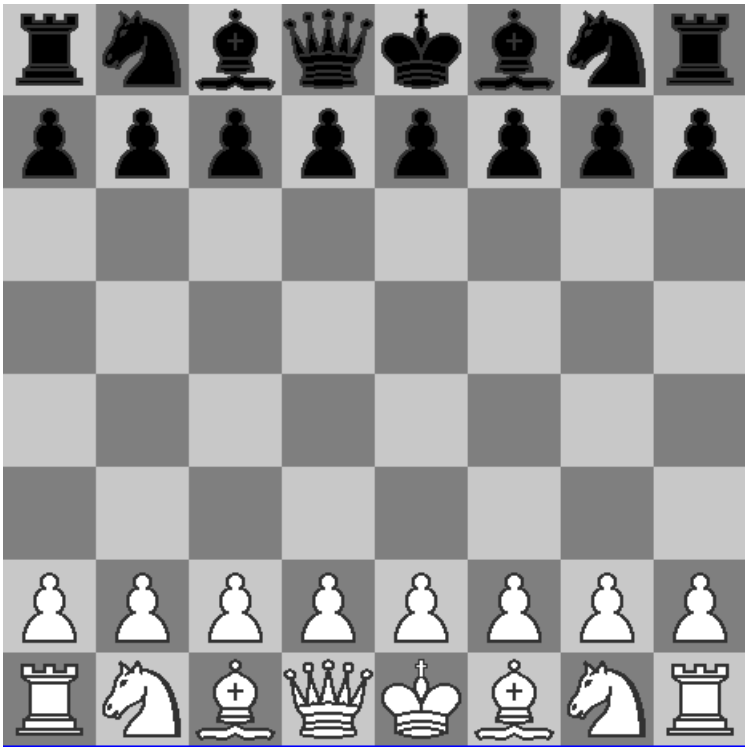

Figura 7

2. Equipos, materiales y temas utilizados

- Sistema Operativo Windows 10.
- VIM 9.0.
- Python 3.11.3.
- Pip 23.1.2
- Pygame 2.4.0
- Git 2.41.0.
- Cuenta en GitHub con el correo institucional.
- Programación Orientada a Objetos.

3. URL de Repositorio Github

- URL del Repositorio GitHub.
- `https://github.com/Sebastian04081/Lab04-Pweb2.git`
- URL para el laboratorio 04 en el Repositorio GitHub.
- `https://github.com/rescobedoq/pw2/tree/main/labs/lab04`

4. Actividades con el repositorio GitHub

4.1. Creando e inicializando repositorio GitHub

- Se creo el repositorio GitHub.
- Se realizaron los siguientes comandos en la computadora:

Listing 1: Creando directorio de trabajo

```
$ mkdir ~/workspace/Pweb2/
```

Listing 2: Dirijiéndonos al directorio de trabajo

```
$ cd ~/workspace/Pweb2/
```

Listing 3: Creando directorio para repositorio GitHub

```
$ mkdir Lab04-Pweb2/
```

Listing 4: Inicializando directorio para repositorio GitHub

```
$ git init
$ git remote add origin https://github.com/Sebastian04081/Lab04-Pweb2.git
```

Listing 5: Inicializando directorio para crear el entorno virtual

```
$ pip install virtualenv
$ mkdir ./my_env
$ source my_env/Scripts/activate
```

Listing 6: Instalando pygame para poder trabajar los ejercicios propuestos

```
$ pip install pygame
$ pip list
Package    Version
---------- -------
pip        23.1.2
pygame     2.4.0
setuptools 67.7.2
wheel      0.40.0
```

Listing 7: Inicializando directorio para cargar los archivos .py

```
$ mkdir ./src
$ cd ./src
$ ls -l
total 52
drwxr-xr-x 1 __pycache__/
-rw-r--r-- 1 chessPictures.py
-rw-r--r-- 1 colors.py
-rw-r--r-- 1 Ejercicio2a.py
-rw-r--r-- 1 Ejercicio2b.py
```

```
-rw-r--r-- 1 Ejercicio2c.py
-rw-r--r-- 1 Ejercicio2d.py
-rw-r--r-- 1 Ejercicio2e.py
-rw-r--r-- 1 Ejercicio2f.py
-rw-r--r-- 1 Ejercicio2g.py
-rw-r--r-- 1 interpreter.py
-rw-r--r-- 1 picture.py
-rw-r--r-- 1 pieces.py
```

4.2. Commits

Listing 8: Primer Commit Creando carpeta/src para los archivos .py

```
$ git add .
$ git commit -m "Agregando los archivos .py"
$ git push -u origin main
```

- Se creo el archivo **.gitignore** para no considerar los archivos ***.class** que son innecesarios hacer seguimiento.

Listing 9: Creando requirements.txt

```
$ vim ../requirements.txt
```

Listing 10: ../requirements.txt

```
pygame==2.4.0
```

Listing 11: Commit: Creando requirements.txt para especificar las verciones de las librerias

```
$ git add .
$ git commit -m "Agregando el archivo requirements.txt"
$ git push -u origin main
```

- Para el siguiente commit se implemento los métodos de la clase picture.py.
- Los métodos implementados son los siguientes:

Listing 12: picture.py

```
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
24    horizontal = list(reversed(self.img))
25    return Picture(horizontal)
26
27  def negative(self):
28    """ Devuelve un negativo de la imagen """
29    negative = []
30
31    for row in self.img:
32        neg_row = ""
33        for pixel in row:
34            neg_row += self._invColor(pixel)
35        negative.append(neg_row)
36
37    return Picture(negative)
38
39  def join(self, p):
40    """ Devuelve una nueva figura poniendo la figura del argumento
41        al lado derecho de la figura actual """
42    join = []
43
44    for i in range(len(self.img)):
45        join.append(self.img[i] + p.img[i])
46
47    return Picture(join)
48
49  def up(self, p):
50    """ Devuelve una nueva figura poniendo la figura p sobre la
51        figura actual """
52    result = []
53
54    for row in p.img:
55        result.append(row)
56
57    for row in self.img:
58        result.append(row)
59
60    return Picture(result)
61
62  def under(self, p):
63    """ Devuelve una nueva figura poniendo la figura p sobre la
64        figura actual """
65
66    merged = []
67    upper = p.img
```

```
68    under = self.img
69
70    for i in range(len(under)):
71        merged_row = ""
72
73        for j in range(len(under)):
74            if upper[i][j] != under[i][j] and upper[i][j] != " ":
75                merged_row += upper[i][j]
76            else:
77                merged_row += under[i][j]
78
79        merged.append(merged_row)
80
81    return Picture(merged)
82
83  def horizontalRepeat(self, n):
84    """ Devuelve una nueva figura repitiendo la figura actual al costado
85        la cantidad de veces que indique el valor de n """
86    repeated = []
87
88    for row in self.img:
89        repeated_row = []
90        for _ in range(n):
91            repeated_row.extend(row)
92        repeated.append(repeated_row)
93
94    return Picture(repeated)
95
96  def verticalRepeat(self, n):
97    """ Devuelve una nueva figura que repite verticalmente la imagen actual
98        del objeto n veces """
99    repeated = []
100
101   for _ in range(n):
102       repeated.extend(self.img)
103
104   return Picture(repeated)
105
106 # Extra: Solo para realmente viciosos
107 def rotate(self):
108   """ Devuelve una figura rotada en 90 grados, puede ser en sentido horario
109   o antihorario """
110
111   rows = len(self.img)
112   cols = len(self.img[0])
113
114   rotated = [[self.img[rows-1-j][i] for j in range(rows)] for i in range(cols)]
115
116   return Picture(rotated)
```

Listing 13: Para el siguiente commit se implementaron los Ejercicios 2a hasta el 2g.

```
$ ls -l
total 56
drwxr-xr-x 1 __pycache__/
-rw-r--r-- 1 chessPictures.py
```

```
-rw-r--r-- 1 colors.py
-rw-r--r-- 1 Ejercicio2a.py
-rw-r--r-- 1 Ejercicio2b.py
-rw-r--r-- 1 Ejercicio2c.py
-rw-r--r-- 1 Ejercicio2d.py
-rw-r--r-- 1 Ejercicio2e.py
-rw-r--r-- 1 Ejercicio2f.py
-rw-r--r-- 1 Ejercicio2g.py
-rw-r--r-- 1 interpreter.py
-rw-r--r-- 1 picture.log
-rw-r--r-- 1 picture.py
-rw-r--r-- 1 pieces.py
```

Listing 14: Ejercicio2a.py

```
from interpreter import draw
from chessPictures import *

draw(knight.negative().join(knight).up(knight.join(knight.negative())))
```

Listing 15: Ejercicio2b.py

```
from interpreter import draw
from chessPictures import *

negative = knight.negative()
draw(negative.verticalMirror().join(knight.verticalMirror()).up(knight.join(negative))
```

Listing 16: Ejercicio2c.py

```
from interpreter import draw
from chessPictures import *

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

Listing 17: Ejercicio2d.py

```
from interpreter import draw
from chessPictures import *

draw(square.join(square.negative()).horizontalRepeat(4))
```

Listing 18: Ejercicio2e.py

```
from interpreter import draw
from chessPictures import *

draw(square.negative().join(square).horizontalRepeat(4))
```

Listing 19: Ejercicio2f.py

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

```

figure1 = square.join(square.negative()).horizontalRepeat(4)
figure2 = square.negative().join(square).horizontalRepeat(4)

final_figure = figure2.up(figure1).verticalRepeat(2)

draw(final_figure)
```

Listing 20: Ejercicio2g.py

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

def buildSquare(piece, x):
    if x == 0:
        return square.negative().under(piece)
    if x == 1:
        return square.under(piece)
    return None

def buildPawns():
    figure = buildSquare(pawn, 1)
    for i in range(7):
        if i % 2 == 0:
            figure = figure.join(buildSquare(pawn, 0))
        else:
            figure = figure.join(buildSquare(pawn, 1))
    return figure

def buildMain():
    figure = buildSquare(rock, 0)
    figure = figure.join(buildSquare(knight, 1))
    figure = figure.join(buildSquare(bishop, 0))
    figure = figure.join(buildSquare(queen, 1))
    figure = figure.join(buildSquare(king, 0))
    figure = figure.join(buildSquare(bishop, 1))
    figure = figure.join(buildSquare(knight, 0))
    figure = figure.join(buildSquare(rock, 1))

    return figure

# Building Rows
whitePawns = buildPawns()
mainWhite = buildMain()

# Building empty squares
squaresRow = square.join(square.negative()).horizontalRepeat(4)
invertedSquaresRow = squaresRow.verticalMirror()
emptySquares = invertedSquaresRow.up(squaresRow).verticalRepeat(2)

# Building formations
whiteTroop = mainWhite.up(whitePawns)
blackTroop = whitePawns.negative().up(mainWhite.negative())

# Building the chessboard
chessboard = whiteTroop.up(emptySquares).up(blackTroop)
```

```
47
48 draw(chessboard)
```

4.3. Estructura de laboratorio 04

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

```
Lab04-Pweb2/
|--- requirements.txt
|--- Informe
|    |--- Informe_Lab04(Python).pdf
|    |--- img
|        |--- logo_abet.png
|        |--- logo_episunsa.png
|        |--- logo_unsa.jpg
|        |--- ejercicio_02_a.png
|        |--- ejercicio_02_b.png
|        |--- ejercicio_02_c.png
|        |--- ejercicio_02_d.png
|        |--- ejercicio_02_e.png
|        |--- ejercicio_02_f.png
|        |--- ejercicio_02_g.png
|
|--- src
|    |--- chessPictures.py
|    |--- colors.py
|    |--- Ejercicio2a.py
|    |--- Ejercicio2b.py
|    |--- Ejercicio2c.py
|    |--- Ejercicio2d.py
|    |--- Ejercicio2e.py
|    |--- Ejercicio2f.py
|    |--- Ejercicio2g.py
|    |--- interpreter.py
|    |--- picture.log
|    |--- picture.py
|    |--- pieces.py
```

5. Cuestionario

- ¿Qué son los archivos *.pyc?

  Los archivos '.pyc' son archivos de código de byte compilado en Python. Cuando se ejecuta un archivo de Python, el intérprete de Python compila el código fuente en un formato de código de byte que es más eficiente para la ejecución. Estos archivos '.pyc' contienen el código de byte compilado y se utilizan para acelerar la carga y ejecución de un módulo o programa en Python.

- ¿Para qué sirve el directorio pycache?

  El directorio '__pycache__', a menudo denominado 'pycache', es utilizado por el intérprete de Python para almacenar los archivos .pyc. Es un directorio especial creado automáticamente cuando se compila un archivo .py y se almacenan allí los archivos .pyc correspondientes. El

propósito principal de este directorio es mantener los archivos compilados en un lugar separado para evitar la necesidad de recompilarlos cada vez que se ejecute el programa.

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

  El subguión (_) en Python se utiliza para indicar que un elemento no es relevante en un contexto específico. También se usa en importaciones especiales y para marcar cadenas de texto que deben ser traducidas en aplicaciones internacionales.

6. Referencias

- `https://www.w3schools.com/python/python_reference.asp`
- `https://docs.python.org/`
- `https://www.geeksforgeeks.org/python-programming-language/`