Informe de Laboratorio 04

Tema: Python

Nota

Estudiante	Escuela	Asignatura
Jorge Luis Escobedo Ocaña jescobedooc@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Programación Web 2 Semestre: III Código: 20231001

Laboratorio	Tema	Duración
04	Python	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - A	Del 25 Mayo 2023	Al 08 Junio 2023

1. Tarea

Implementar la clase Picture contará además con varios métodos:

- 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

2. Equipos, materiales y temas utilizados

- Sistema Operativo Ubuntu GNU Linux 23 lunar 64 bits Kernell 6.2.
- VIM 9.0.
- Python 3.
- Git 2.39.2.
- Cuenta en GitHub con el correo institucional.
- Programación Orientada a Objetos.

3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- `https://github.com/Juerges28/Lab04-Pweb2.git`
- URL para el laboratorio 04 en el Repositorio GitHub.
- `https://github.com/rescobedoq/programacion/tree/main/lab04`

4. Actividades con el repositorio GitHub

4.1. Creando e inicializando repositorio GitHub

- Como es el primer laboratorio se creo el repositorio GitHub.
- Se realizaron los siguientes comandos en la computadora:

Listing 1: Creando directorio de trabajo

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

Listing 2: Dirijíéndonos al directorio de trabajo

```
$ cd Lab04-Pweb2/
```

Listing 3: Creando directorio para repositorio GitHub

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

Listing 4: Inicializando directorio para repositorio GitHub

```
$ cd Lab04-Pweb2/
$ echo "# programacion" >> README.md
$ git init
$ git config --global user.name "Jorge Luis Escobedo Ocaa"
$ git config --global user.email jescobedooc@unsa.edu.pe
$ git add README.md
$ git commit -m "first commit"
$ git branch -M main
$ git remote add origin https://github.com/Juerges28/Lab04-Pweb2.git
$ git push -u origin main
```

4.2. Commits

Listing 5: Primer Commit Creando carpeta/archivo para laboratorio 01

```
$ mkdir Tarea-del-Ajedrez/
$ touch Tarea-del-Ajedrez/pieces.py
$ touch Tarea-del-Ajedrez/picture.py
$ touch Tarea-del-Ajedrez/colors.py
$ touch Tarea-del-Ajedrez/chessPictures.py
$ git add .
$ git commit -m "Creando Tarea-del-Ajedrez/archivos para laboratorio 04"
$ git push -u origin main
```

- Para el siguiente commit se implementaron los metodos en la clase picture.py necesarios para realizar los ejercicios.

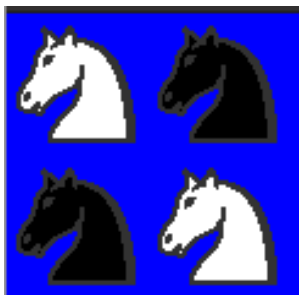

Figura 1: Ejercicio 2a

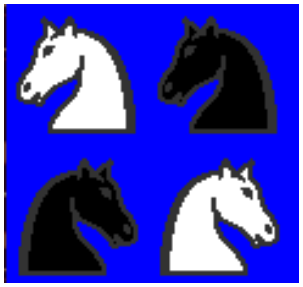

Figura 2: Ejercicio 2b

Figura 3: Ejercicio 2c

Figura 4: Ejercicio 2d

Figura 5: Ejercicio 2e

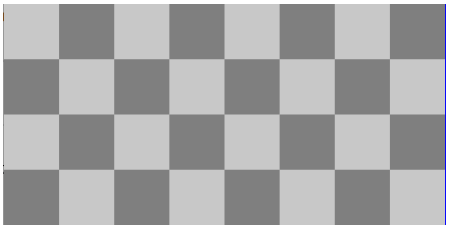

Figura 6: Ejercicio 2f

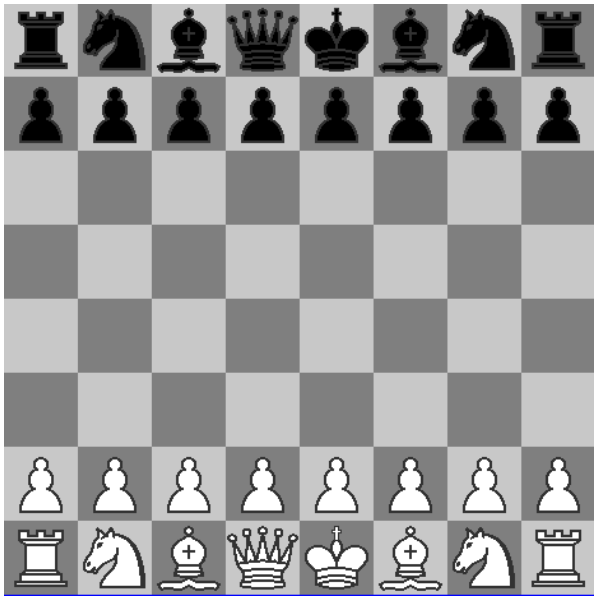

Figura 7: Ejercicio 2g

- Implementamos los metodos necesarios en la clase picture.py para el desarrollo en los ejercicios.

Listing 6: Modificando picture.py

```
$ vim Tarea-del-Ajedrez/picture.py
```

Listing 7: 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 """
    return Picture(self.img[::-1])

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

```

```
25  def negative(self):
26    """ Devuelve un negativo de la imagen """
27    negative = []
28    for row in self.img:
29      negative_row = [self._invColor(pixel) for pixel in row]
30      negative.append(negative_row)
31    return Picture(negative)
32
33  def join(self, p):
34    """ Devuelve una nueva figura poniendo la figura del argumento
35        al lado derecho de la figura actual """
36    joined_img = []
37    for i in range(len(self.img)):
38        joined_row = ''.join(self.img[i]) + ''.join(p.img[i])
39        joined_img.append(joined_row)
40    return Picture(joined_img)
41
42  def up(self, p):
43    """ Devuelve una nueva figura poniendo la figura p bajo la figura actual """
44    new_img = p.img + self.img
45    return Picture(new_img)
46
47  def under(self, p):
48    """ Devuelve una nueva figura poniendo la figura p sobre la figura actual """
49    new_img = self.img + p.img
50    return Picture(new_img)
51
52  def insert(self, p):
53    """ Devuelve una nueva figura poniendo la figura p sobre la figura actual """
54    return Picture(None)
55
56  def horizontalRepeat(self, n):
57    """ Devuelve una nueva figura repitiendo la figura actual al costado
58        la cantidad de veces que indique el valor de n """
59    repeated_img = [row * n for row in self.img]
60    return Picture(repeated_img)
61
62  def verticalRepeat(self, n):
63    """ Devuelve una nueva figura repitiendo la figura actual hacia abajo
64        la cantidad de veces que indique el valor de n """
65    repeated_img = []
66    for _ in range(n):
67        repeated_img.extend(self.img)
68    return Picture(repeated_img)
69
70  def insert(self, p):
71    """ Devuelve una nueva figura sobreponiendo la figura actual sobre la figura p """
72    insert_img = []
73    for row_self, row_p in zip(self.img, p.img):
74        insert_row = ''.join([c_self if c_self != ' ' else c_p for c_self, c_p in
              zip(row_self, row_p)])
75        insert_img.append(insert_row)
76    return Picture(insert_img)
```

Listing 8: Commit: Guardando cambios en picture.py

```
$ git add .
$ git commit -m "Guardando cambios en picture.py"
$ git push -u origin main
```

Listing 9: Creando Ejercicio2a.py

```
$ vim Tarea-del-Ajedrez/Ejercicio2a.py
```

Listing 10: Ejercicio2a.py

```
from interpreter import draw
from chessPictures import *

horse = knight
horseN = horse.negative()
horses = horse.join(horseN)

horsesN = horses.negative()
horses4 = horses.under(horsesN)

draw(horses4)
```

- Se comprueba la igualdad a la imagen planteada como ejercicio.

Listing 11: Probando código

```
$ cd Tarea-del-Ajedrez
$ python Ejercicio2a.py
```

Listing 12: Commit: Comprobacion de exito del Ejercicio 2a

```
$ git add .
$ git commit -m "Comprobacion de exito del Ejercicio 2a"
$ git push -u origin main
```

Listing 13: Creando Ejercicio2b.py

```
$ vim Tarea-del-Ajedrez/Ejercicio2b.py
```

Listing 14: Ejercicio2b.py

```
from interpreter import draw
from chessPictures import *

horse = knight
horseN = horse.negative()

horses = horse.join(horseN)
horsesInv = horses.horizontalMirror()

horses4 = horses.under(horsesInv)
```

```
11
12 draw(horses4)
```

- Se comprueba la igualdad a la imagen planteada como ejercicio.

Listing 15: Probando código

```
$ cd Tarea-del-Ajedrez
$ python Ejercicio2b.py
```

Listing 16: Commit: Comprobacion de exito del Ejercicio 2b

```
$ git add .
$ git commit -m "Comprobacion de exito del Ejercicio 2b"
$ git push -u origin main
```

Listing 17: Creando Ejercicio2c.py

```
$ vim Tarea-del-Ajedrez/Ejercicio2c.py
```

Listing 18: Ejercicio2c.py

```
1 from interpreter import draw
2 from chessPictures import *
3
4 queen4 = queen.horizontalRepeat(4)
5
6 draw(queen4)
```

- Se comprueba la igualdad a la imagen planteada como ejercicio.

Listing 19: Probando código

```
$ cd Tarea-del-Ajedrez
$ python Ejercicio2c.py
```

Listing 20: Commit: Comprobacion de exito del Ejercicio 2c

```
$ git add .
$ git commit -m "Comprobacion de exito del Ejercicio 2c"
$ git push -u origin main
```

Listing 21: Creando Ejercicio2d.py

```
$ vim Tarea-del-Ajedrez/Ejercicio2d.py
```

Listing 22: Ejercicio2d.py

```
1 from interpreter import draw
2 from chessPictures import *
```

```
3
4  square2 = square.join(square.negative())
5  squareF = square2.horizontalRepeat(4)
6
7  draw(squareF)
```

- Se comprueba la igualdad a la imagen planteada como ejercicio.

Listing 23: Probando código

```
$ cd Tarea-del-Ajedrez
$ python Ejercicio2d.py
```

Listing 24: Commit: Comprobacion de exito del Ejercicio 2d

```
$ git add .
$ git commit -m "Comprobacion de exito del Ejercicio 2d"
$ git push -u origin main
```

Listing 25: Creando Ejercicio2e.py

```
$ vim Tarea-del-Ajedrez/Ejercicio2e.py
```

Listing 26: Ejercicio2e.py

```
1  from interpreter import draw
2  from chessPictures import *
3
4  square2 = square.negative().join(square)
5  squareF = square2.horizontalRepeat(4)
6
7  draw(squareF)
```

- Se comprueba la igualdad a la imagen planteada como ejercicio.

Listing 27: Probando código

```
$ cd Tarea-del-Ajedrez
$ python Ejercicio2e.py
```

Listing 28: Commit: Comprobacion de exito del Ejercicio 2e

```
$ git add .
$ git commit -m "Comprobacion de exito del Ejercicio 2e"
$ git push -u origin main
```

Listing 29: Creando Ejercicio2f.py

```
$ vim Tarea-del-Ajedrez/Ejercicio2f.py
```

Listing 30: Ejercicio2f.py

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

square2 = square.join(square.negative())
squareF = square2.horizontalRepeat(4)
squareFinv = squareF.horizontalMirror()

square4 = squareF.under(squareFinv).verticalRepeat(2)

draw(square4)
```

- Se comprueba la igualdad a la imagen planteada como ejercicio.

Listing 31: Probando código

```
$ cd Tarea-del-Ajedrez
$ python Ejercicio2f.py
```

Listing 32: Commit: Comprobacion de exito del Ejercicio 2f

```
$ git add .
$ git commit -m "Comprobacion de exito del Ejercicio 2f"
$ git push -u origin main
```

Listing 33: Creando Ejercicio2g.py

```
$ vim Tarea-del-Ajedrez/Ejercicio2g.py
```

Listing 34: Ejercicio2g.py

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

square2 = square.join(square.negative())
squareF = square2.horizontalRepeat(4)
squareFinv = squareF.horizontalMirror()
tablero1 = squareF.under(squareFinv)
tablero2 = tablero1.verticalRepeat(2)

b1 =
    rock.join(knight.join(bishop).join(queen.join(king.join(bishop.join(knight.join(rock))))))
b2 = pawn.horizontalRepeat(8)

b3 = b1.negative().under(b2.negative())
b3 = b3.insert(tablero1)
b4 = b2.under(b1)
b4 = b4.insert(tablero1.horizontalMirror())

b5 = b3.under(tablero2)
b5 = b5.under(b4.horizontalMirror())
draw(b5)
```

- Se comprueba la igualdad a la imagen planteada como ejercicio.

Listing 35: Probando código

```
$ cd Tarea-del-Ajedrez
$ python Ejercicio2g.py
```

Listing 36: Commit: Comprobacion de exito del Ejercicio 2g

```
$ git add .
$ git commit -m "Comprobacion de exito del Ejercicio 2g"
$ git push -u origin main
```

4.3. Estructura de laboratorio 04

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

```
Tarea-del-Ajedrez/
|--- Ejercicio2a.py
|--- Ejercicio2b.py
|--- Ejercicio2c.py
|--- Ejercicio2d.py
|--- Ejercicio2e.py
|--- Ejercicio2f.py
|--- Ejercicio2g.py
|--- chessPictures.py
|--- colors.py
|--- interpreter.py
|--- picture.py
|--- pieces.py
|--- latex
   |--- img
   |   |--- logo_abet.png
   |   |--- logo_episunsa.png
   |   |--- logo_unsa.jpg
   |   |--- pseudocodigo_insercion.png
   |   |--- 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
   |--- pweb2_lab04_jescobedooc_v1.0.pdf
   |--- pweb2_lab04_jescobedooc_v1.0.tex
   |--- src
       |--- Ejercicio2a.py
       |--- Ejercicio2b.py
       |--- Ejercicio2c.py
       |--- Ejercicio2d.py
       |--- Ejercicio2e.py
       |--- Ejercicio2f.py
       |--- Ejercicio2g.py
       |--- picture.py
```

5. CUESTIONARIO

- ¿Qué son los archivos *.pyc?
  Los archivos .pyc son archivos de código compilado generados por Python. Contienen el código fuente de un archivo .py que ha sido compilado en bytecode de Python. Los archivos .pyc permiten una ejecución más rápida del código, ya que evitan la necesidad de recompilar el archivo .py cada vez que se ejecuta.

- ¿Para qué sirve el directorio pycache?
  El directorio "pycache"se utiliza para almacenar los archivos .pyc generados por Python. Estos archivos se crean para mejorar el rendimiento al ejecutar el código Python, ya que evitan la necesidad de recompilar el código fuente cada vez que se ejecuta.

- ¿Cuáles son los usos y lo que representa el subguión en Python?
  El subguión (_) en Python tiene varios usos:
  1. Como nombre de variable temporal o descartada cuando no es relevante.
  2. En interpretadores interactivos, se utiliza para almacenar el resultado de la expresión anterior.
  3. En convenciones de nomenclatura, puede indicar que una variable o método es de uso interno y no debe accederse directamente desde fuera de la clase o módulo.
  En resumen, el subguión se utiliza para representar variables o elementos que no son relevantes o que tienen un uso específico en Python.

6. Referencias

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