Informe de Laboratorio 04
Tema: Python

Nota

Estudiante	Escuela	Asignatura
Paulo Andre Hidalgo Chinchay phidalgo@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Programación web 2 Semestre: III Código: 20223011

Laboratorio	Tema	Duración
04	Python	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - A	Del 2 Junio 2023	Al 9 Junio 2023

1. Tarea

- 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):

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

2. Equipos, materiales y temas utilizados

- Sistema Operativo Ubuntu GNU Linux 23 lunar 64 bits Kernell 6.2.
- Sistema Operativo Windows 11 pro versión 22H2 de 64 bits.
- VIM 9.0.
- Git 2.39.2.
- Visual Studio Code 1.78.2.
- Pycharm 2023.1.2
- Cuenta en GitHub con el correo institucional.
- Clases de 1 a 31 del curso de Udemy "Python Practicando. Desde 0 hasta Desarrollador en Python"

- https://www.udemy.com/share/101sJC3@xod_U1pjR_efxIhsxPzAUuEnE3ok_9rlPTHMowoqaob1g4YfN7M-j7alvT0_tbf4Vw==/
- Uso de append por w3schools
- https://www.w3schools.com/python/ref_list_append.asp

3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- https://github.com/PauloUNSA/pw2-lab-c-23a.git
- URL para el laboratorio 03 en el Repositorio GitHub.
- https://github.com/PauloUNSA/pw2-lab-c-23a/tree/main/lab4

4. Configurar espacio de trabajo

- El archivo picture.py es la clase picture que contiene algunos metodos ya desarrollados los otros se desarrollaran en y explicaran aqui.
- Para poder ejecutar todos los ejercicios se tuvo que instalar la liberia Pygame con el siguiente comando.

Listing 1: Instalar libreria Pygame

```
$ pip install pygame
```

5. Ejercicios

5.1. Primer ejercicio

- Se tuvieron que implementar las funciones negative, join y under. En el codigo se detalla lo que hacen.

Listing 2: Picture version 1

```
import colors
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 negative(self):
        """ Devuelve un negativo de la imagen """
        neg = []
        for fila in self.img:
```

```
            cadena = ""
            for color in fila:
                cadena += self._invColor(color)
            neg.append(cadena)
        return Picture(neg)
    def join(self, p):
        """ Devuelve una nueva figura poniendo la figura del argumento
            al lado derecho de la figura actual """
        juntos = []
        for i in range(len(self.img)):
            juntos.append(self.img[i] + p.img[i])
        return Picture(juntos)
    def under(self, p):
        """ Devuelve una nueva figura poniendo la figura p sobre la
                figura actual """
        return Picture(self.img[::] + p.img[::])
```

Para implementar el negativo se cambia el color de cada carácter, a excepción del espacio ya que este no tiene inverso con doble for(1 anidado). Juntando estos caracteres por medio en una cadena y despues juntando esta cadena al nuevo arreglo con el metodo append.

Listing 3: Ejercicio 2 a

```
from interpreter import draw
from chessPictures import *

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

Para poder hacer que se dibuje como se mostrara el la figura de abajo se tuvo de juntar un caballo blanco con uno negro en la primera fila; para ello se utilizo join y negative. Para la segunda fila se necesito saltar a la siguiente fila por lo que se utilizo el metodo under.

Figura 1: Ejecución exitosa ejercicio 2 a

5.2. Segundo ejercicio

- Se tuvieron que implementar la función horizontalMirror Y verticalMirror. Se detalla lo que hace a continuación.

Listing 4: Picture version 2

```
from colors import *
class Picture:
  def __init__(self, img):
    self.img = img;
def horizontalMirror(self):
    """ Devuelve el espejo horizontal de la imagen """
    horizontal = self.img[::-1]
    return Picture(horizontal)
def verticalMirror(self):
    """ Devuelve el espejo vertical de la imagen """
    vertical = []
    for value in self.img:
      vertical.append(value[::-1])
    return Picture(vertical)
```

Para no gastar espacio de más a partir de aqui solo se coloco las funciones las funciones de clase y el métodos horizontalMirror y verticalMirror. El método horizontalMirror solo devuelve el arreglo al revés, el 1er elemento al ultimo y el ultimo al 1ro. verticalMirror tiene que entrar en cadena y ordenarla al revés, juntándola en el arreglo vertical y retornándola como Picture.

Listing 5: Ejercicio 2 b

```
from interpreter import draw
from chessPictures import *

draw(knight.join(knight.negative()).under(knight.negative().verticalMirror().join(knight.verticalMirror())))
```

La 1ra fila se hace de igual forma que en el 1er ejercicio sin embargo para la fila de abajo se agregar .verticalMirror a ambos caballos para invertir verticalmente las figuras.

Figura 2: Ejecución exitosa ejercicio 2 b

5.3. Tercer ejercicio

- No se tuvo que implementar método adicionar

Listing 6: Ejercicio 2 c

```
from interpreter import draw
from chessPictures import *

draw(queen.join(queen).join(queen).join(queen))
```

Una sola final de 4 reinas del mismo color con el método join.

Figura 3: Ejecución exitosa ejercicio 2 c

5.4. Cuarto ejercicio

- Se tuvieron que implementar la función horizontalRepeat Y verticalRepeat. Se detalla lo que hace a continuación.

Listing 7: Picture version 3

```
from colors import *
class Picture:
  def __init__(self, img):
    self.img = img;
  def horizontalRepeat(self, n):
    """ Devuelve una nueva figura repitiendo la figura actual al costado
        la cantidad de veces que indique el valor de n """
    return Picture(self*n)

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

El método horizontalRepeat multiplica n veces abajo el arreglo y verticalMirror tiene la misma lógica pero en este caso hacia la derecha.

Listing 8: Ejercicio 2 d

```
from interpreter import draw
from chessPictures import *
draw((square.join(square.negative())).verticalRepeat(4))
```

1 sola fila de 8 casillas intercaladas, primero clara y después oscura. Para ello se referecian 2 y despues con el método verticalRepeat se repite 4 veces.

Figura 4: Ejecución exitosa ejercicio 2 d

5.5. Quinto ejercicio

- No se implemento método adicional.

Listing 9: Ejercicio 2 e

```
from interpreter import draw
from chessPictures import *
draw((square.negative().join(square)).verticalRepeat(4))
```

Igual que el 4to sino que las casillas van al revés, oscuras después claras

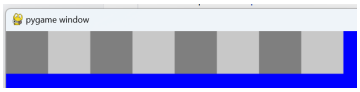

Figura 5: Ejecución exitosa ejercicio 2 e

5.6. Sexto ejercicio

- No se implemento método adicional.

Listing 10: Ejercicio 2 f

```
from interpreter import draw
from chessPictures import *
draw((((square.join(square.negative())).verticalRepeat(4)).
     under((square.negative().join(square)).
          verticalRepeat(4))).horizontalRepeat(2))
```

Se baso en el 4to y 5to ejercicio donde se juntaron ambos con el método under y se duplico con el método horizontalRepeat.

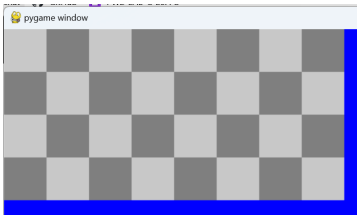

Figura 6: Ejecución exitosa ejercicio 2 f

5.7. Ultimo ejercicio

- Se implemento el método up.

Listing 11: Picture version 4

```
from colors import *
class Picture:
  def __init__(self, img):
    self.img = img;
  def up(self, p):
    sobre = []
    for i in range(len(self.img)):
      linea = []
      for j in range(len(self.img[i])):
        if p.img[i][j] != ' ':
          linea.append(p.img[i][j])
        else:
          linea.append(self.img[i][j])
      sobre.append(linea)
    return Picture(sobre)
```

El método up hace que se sobreponga la figura recibida como argumento. Donde primero se valida si el carácter de la figura que ira encima no sea un espacio, en caso se cumpla ira el carácter de la figura de encima sino ira el de la figura de abajo.

Listing 12: Ejercicio 2 d

```
from interpreter import draw
from chessPictures import *
derecha =
    square.negative().up(rock).join(square.up(knight)).join(square.negative().up(bishop))
centro = (square.up(queen)).join(square.negative().up(king))
izquierda = square.up(bishop).join(square.negative().up(knight)).join(square.up(rock))
filaAbajo = derecha.join(centro).join(izquierda)
filaArriba = filaAbajo.negative()
peonesBlancos = (square.up(pawn).join(square.negative().up(pawn))).verticalRepeat(4)
pNegros = peonesBlancos.negative()
superio = filaArriba.under(pNegros)
inferior = peonesBlancos.under(filaAbajo)
medio = (((square.join(square.negative())).verticalRepeat(4)).
       under((square.negative().join(square)).verticalRepeat(4))).horizontalRepeat(2)
draw(superio.under(medio).under(inferior))
```

Se reutilizo el codigo del 6to ejercicio para medio. y de subdividio en derecha, izquiera y centro para la fila principal, donde van las torres alfiles, caballos, reina y rey. Se junto todo esto en filaAbajo ya que era donde iban los blancos. Y para filaArriba se añadio negative. Los peones solo eran una fila de casilla clara y oscura con peones encima. Quedando las partes superior que era igual a filaArriba concatenada con pNegros (los peones neegros) y para la parte inferior, filaAbajo concatenada con peonesBlancos. finalmente al draw se le paso el argumento superio.under(medio).under(inferior) quedando asi.

Figura 7: Ejecución exitosa ejercicio 2 g

5.8. Estructura de laboratorio 04

- Gracias al .gitignore no se suben la carpeta pycache ni .idea, del editor pycahrm
- El contenido que se entrega en este laboratorio es el siguiente:

```
C:\USERS\PAULO\PW2-LAB-C-23A\LAB4
|   .gitignore
|   chessPictures.py
|   colors.py
|   Ejercicio2a.py
|   Ejercicio2b.py
|   Ejercicio2c.py
|   Ejercicio2d.py
|   Ejercicio2e.py
|   Ejercicio2f.py
|   Ejercicio2g.py
|   interpreter.py
|   picture.py
|   pieces.py
|   prueba.py
|
|------.idea
|   |   lab4.iml
|   |   misc.xml
|   |   vcs.xml
|   |   workspace.xml
|   |
|   +------inspectionProfiles
|          profiles_settings.xml
|
```

```
|------latex
|   |   lab4_paulo-hidalgo.tex
|  |  lab4_paulo-hidalgo.pdf
|   |
|   |------build
|   |       lab4_paulo-hidalgo.aux
|   |       lab4_paulo-hidalgo.fdb_latexmk
|   |       lab4_paulo-hidalgo.fls
|   |       lab4_paulo-hidalgo.log
|   |       lab4_paulo-hidalgo.out
|   |       lab4_paulo-hidalgo.pdf
|   |       lab4_paulo-hidalgo.synctex.gz
|   |
|   |------img
|   |       crea-evento.png
|   |       e2a.png
|   |       e2b.png
|   |       e2c.png
|   |       e2d.png
|   |       e2e.png
|   |       e2f.png
|   |       e2g.png
|   |       eliminar.png
|   |       l-eventos.png
|   |       localhost01.png
|   |       localhost02.png
|   |       localhost03.png
|   |       localhost_agenda.png
|   |       logo_abet.png
|   |       logo_episunsa.png
|   |       logo_unsa.jpg
|   |       ver-eventos01.png
|   |       ver-eventos02.png
|   |
|   +------src
|           e2a.py
|           e2b.py
|           e2c.py
|           e2d.py
|           e2e.py
|           e2f.py
|           e2g.py
|           pic01.py
|           pic02.py
|           pic03.py
|           pic04.py
|
+------__pycache__
       *
```

5.9. Pregunta: Explique: ¿Para qué sirve el directorio pycache?

- Sirve para guardar los compilados de python, asi como en java al ejecutarlos se crean los .class en python se crean los .pyc. Esto se hace automaticamente ya que se importan modulos de otras clases como se da en este caso.

6. Rúbricas

6.1. Rúbrica para el contenido del Informe y demostración

- El alumno debe marcar o dejar en blanco en celdas de la columna **Checklist** si cumplio con el ítem correspondiente.
- Si un alumno supera la fecha de entrega, su calificación será sobre la nota mínima aprobada, siempre y cuando cumpla con todos lo items.
- El alumno debe autocalificarse en la columna **Estudiante** de acuerdo a la siguiente tabla:

Tabla 1: Niveles de desempeño

	Nivel			
Puntos	Insatisfactorio 25 %	En Proceso 50 %	Satisfactorio 75 %	Sobresaliente 100 %
2.0	0.5	1.0	1.5	2.0
4.0	1.0	2.0	3.0	4.0

Tabla 2: Rúbrica para contenido del Informe y demostración

Contenido y demostración		Puntos	Checklist	Estudiante	Profesor
1. GitHub	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	X	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	X	4	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	2	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	2	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	2	
6. Fechas	Las fechas de modificación del código fuente estan dentro de los plazos de fecha de entrega establecidos.	2	X	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	2	
8. Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4	X	4	
Total		20		20	

7. Referencias

- https://www.udemy.com/share/101sJC3@xod_U1pjR_efxIhsxPzAUuEnE3ok_9rlPTHMowoqaob1g4YfN7M-j7alvT0_tbf4Vw==/
- https://www.w3schools.com/python/ref_list_append.asp