Informe de Laboratorio 05

Tema: Python

Nota

Estudiante	Escuela	Asignatura
Arce Mayhua Leonardo Velasque Arcos Mikhail Quispe Saavedra Dennis Choquehuanca Bedoya Brayan	Escuela Profesional de Ingeniería de Sistemas	Programación web 2 Semestre: I Código:

Laboratorio	Tema	Duración
05	Python	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2024 - A	Del 26 Mayo 2024	Al 2 Junio 2024

1 INTRODUCCION

- En el siguiente informe expondre las actividades del laboratorio 05 del curso de programacion web 02 ,
- La exposcion de los ejercicios seran de la siguiente manera
- Primero:Expondre las partes mas importantes del codigo, en este caso los metodos ,centrandonos en la logica de estos
- Segundo:Expondremos los ejercicios explicando el uso de los metodos para la resolucion de los ejercicios

2 Clase Picture

- Los metodos desarrollados por el equipo estaran en la clase picture
- verticalMirror()
- Devuelve el espejo vertical de la imagen

Listing 1: VerticalMirror

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

return Picture(vertical)
```

- En este metodo iteraremos sobre la imagen dada como atributo
- Crearemos una imagen vacia(lista) y le daremos los valores de la imagen entregada,pero de manera inversa

  para esto utilizaremos la expresion slice en cada string de la imagen , -1 indicara que comenzara desde el ultimo elemento y el primer: indicara al siguiente

  de esta manera se invertiran todos los caracteres del string para su posterior agregacion a nueva imagen.
- Este metodo lo usaremos en el ejercicio 2b
- horizontalMirror()
- Devuelve el espejo horizontal de la imagen

Listing 2: horizontalMirror

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

- En este metodo iteraremos sobre la imagen dada como atributo
- Crearemos una imagen vacia(lista) y le daremos los valores de la imagen entregada,pero de manera invertida

  para iteraremos con el metodo reversed lo que nos dara el ultimo elemento al primero , lo que permitira que la imagen se muestre de cabeza

  devido a que la nueva imagen tendra como primer elemento el ultimo string de la imagen original y asi hasta el primer elemento de la imagen original
- Este metodo quedara obsoleto
- negative()
- Devuelve el negativo de la imagen

Listing 3: negative

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

- En este metodo cambiaremos el color de la imagen dada con su inverso
- Crearemos una imagen vacia(lista) , luego iteraremos sobre la imagen y agregaremos a la nueva imagen los strings invertidos de la

  imagen original , esto sera posible gracias a el motodo $_{i}nvColorqueusaeldiccionarioinverterdelacalseColor, estediccionari$

  llave un elemento que puede cambiar de color , solo seran 4 los elementos que tendran inversa $_{i}nvColorretornaraelinversodelcaracteryloagregaraalanuevaimagen$

  de esta manera se obtendra el inverso de cada pieza del ajedrez
- Este metodo se utilizara mucho en varios ejercicio destacando su uso en el ejericicio 2g.
- join()
- pone la imagen al lado derecho usando el argumento p

Listing 4: Join

```
    def join(self, p):
nuevaImagen = []

for index, value in enumerate(self.img):
  nuevaImagen.append(list(value) + list(p.img[index]))

return Picture(nuevaImagen)
```

- En este metodo crearemos una imagen que tendra a su costado otra imagen ,algo parecido a una concatenacion
- Crearemos una imagen vacia, iteraremos sobre la imagen dada ,en este caso nos pasaran dos imagenes pero como el taño es el mismo solo usaremos index como indice de la segunda imagen

  luego simplemente concatenaremos los strings de las dos imagenes brindadas osea , el primer string de la primera imagen se concatenara con el primer string de la segunda imagen ,su concatenacion sera

  agregada al primer elemento de la nueva imagen y asi con todos los elementos, deesta manera se podra colocar una imagen al costado de otra.
- Este metodo se utilizara en algunos otros metodos como horizontalRepeat.
- up()
- Devuelve a la figura encima de la actual

Listing 5: up

```
      def up(self, p):
      nuevaImagen = []
      #copiar todo p a nueva imange
      for value in p.img:
        nuevaImagen.append(value)

      for value in self.img:
        nuevaImagen.append(value)
      return Picture(nuevaImagen)
```

- En este metodo crearemos una imagen que tendra encima a otra imagen
- Crearemos una imagen vacia, iteraremos respectivamente con las imagenes dadas, en la iteracion de la primera imagen

  agregaremos los elemento a la imagen vacia creada ,luego iteraremos sobre la segunda imagen y seguiremos añadiendo estos elementos a la imagen anteriormente vacia

  lo que tendremos sera una imagen con el doble de longitud de una imagen normal lo que dara la impresion de que una pieza esta encima de la otra
- Este metodo se utilizara en algunos otros metodos como verticalRepeat.
- under()
- Devuelve una nueva imagen y la sobrepone encima de la actual

Listing 6: under

```
     def under(self, p):
  nuevaImagen = []
  for value in self.img:
      nuevaImagen.append(list(value))

  for i, value in enumerate(p.img):
    for j, caracter in enumerate(value):
      if(nuevaImagen[i][j] == ' '):
        nuevaImagen[i][j] = caracter

  return Picture(nuevaImagen)
```

- En este metodo crearemos una imagen que sobrepondra otra imagen dando una simulacion como un cudrado del ajedres con su pieza encima
- Crearemos una imagen vacia , e iteraremos sobre la imagen que tiene la figura de una pieza, luego haremos una iteracion anidad para consultar todos los caracteres de la nueva imagen

  dentro del bucle anidado utilizaremos una condicional para saber que caracteres de la nueva iamgen se encuentran vacios (" ") ,y caso que este vacio colocaremos el caracter de la otra imagen que sera

  un square de esta manera se colocara la pieza encima de su cuadrado de ajedres
- Este metodo se utilizara en el ultimo ejercicio 2g.
- horizontalRepeat()

- Devuelve una nueva figura repitiendo la figura actual al costado

  la cantidad de veces que indique el valor de n

Listing 7: horizontalRepeat

```
def horizontalRepeat(self, n):
aux = self
for _ in range(n-1):
  aux = aux.join(self)
return aux
```

- En este metodo crearemos una imagen que tendra n veces a la misma imagen a su lado
- Crearemos una imagen qeu tendra lo mismo que la imagen principal sobre todo para que nuestra imagen original no quede modificada , luego iteraremos n-1 veces debido

  a que utilizaremos el metodo join dentro de este bucle , por lo que el trabajo de duplicar n veces la imagen lo hara join ,sin embargo como en la primera iteracion genera2 imagenes se tendra qeu iterar n-1 veces

  , de esta manera colocaremos n veces la imagen al cosatado de la misma imagen
- Este metodo se utilizara en el ejercicio 2c.
- verticalRepeat()
- Devuelve una nueva figura repitiendo la figura actual debajo, la cantidad

  de veces que indique el valor de n

Listing 8: verticalRepeat

```
def verticalRepeat(self, n):
aux = self
for _ in range(n-1):
  aux = aux.up(self)
return aux
```

- En este metodo crearemos una imagen que tendra n veces a la misma imagen encima
- Este metodo se comportara de la misma manera que en anterior metodo , con la diferencia que dentro de las iteraciones se llamara al metodo up envez del join

  lo que permitira sobre poner n veces la imagen .
- Este metodo se utilizara en varios ejercicios.

3 Código del ejercicio 2a

Listing 9: Código del ejercicio A

```
from chessPictures import *
from interpreter import draw

tab = knight
```

```
# Une el caballo con su inverso
tab = Picture.join(tab, Picture.negative(knight))
```

– Tablero resultante del ejercicio A

Listing 10: Tablero resultante del ejercicio A

```
# Muestra el tablero resultante
draw(Picture.up(Picture.negative(tab), tab))
```

4 Código del ejercicio B

Listing 11: Código del ejercicio B

```
from chessPictures import *
from interpreter import draw

tab = knight
# Une el caballo con su inverso
tab = Picture.join(tab, Picture.negative(knight))
# Refleja verticalmente y muestra el tablero resultante
tab = Picture.up(Picture.verticalMirror(tab), tab)
```

– Tablero resultante del ejercicio B

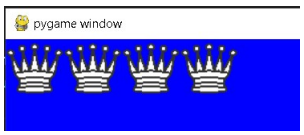
Listing 12: Tablero resultante del ejercicio B

```
# Muestra el tablero resultante
draw(tab)
```

5 Código del ejercicio C

Listing 13: Código del ejercicio C

```
from interpreter import draw
from chessPictures import *

# Repite horizontalmente la imagen del rey cuatro veces
draw(king.horizontalRepeat(4))
```



6 Ejercicio2d

– Creación del cuadro inicial del tablero

Listing 14: Creacion de un primer cuadro

```
eb = Picture(SQUARE)
```

– Creación del cuadro de color negativo(color negro)

Listing 15: Creacion de un primer cuadro negativo

```
en = eb.negative()
```

– Se concluye el ejercicio imprimiendo la imagen fusionada de el cuadro normal y su negativo al costado, a esta imagen se le repite con ”horizontalRepeat” 4 veces.

Listing 16: Creacion e impresion de una fila

```
draw(eb.join(en).horizontalRepeat(4))
```

– Ver impresión:

7 Ejercicio2e

∗ Se realiza lo mismo que el ejercicio anterior pero el orden de invocación de los cuadros se invierte en este caso primero se pone el negativo ”en”.

Listing 17: Creacion e impresion de una fila con primer cuadro negativo

```
draw(en.join(eb).horizontalRepeat(4))
```

* Ver impresión:

8 Ejercicio2f

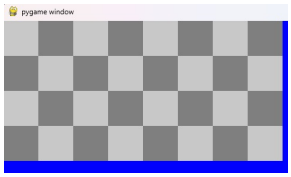
· En este caso reutilizaremos las iniciaciones de los cuadros de los ejercicios anteriores.

Listing 18: Creacion e impresion de medio tablero

```
eb = Picture(SQUARE)
en = eb.negative()
fila1 = eb.join(en).horizontalRepeat(4)
fila2 = en.join(eb).horizontalRepeat(4)
imagen = fila2.up(fila1).verticalRepeat(2)
draw(imagen)
```

en "imagen" se junta de manera horizontal la fila2 a la fila1 y se hace vertical repeat 2 veces lo que nos retorna un tablero de 4X8.

· Ver impresión:



9 Clase o tablero del Ajedrez(Ejercicio 2g)

· En esta sección se utilizarán métodos avanzados de manipulación de imágenes desarrollados hasta ahora, como el método de inversión, que toma la imagen y la invierte, y el método de repetición, que toma la imagen y la repite de forma horizontal o vertical.

· Creación de la fila inicial para las piezas blancas:

Listing 19: Fila inicial para piezas blancas

```
from chessPictures import *
from interpreter import draw
from picture import Picture
from colors import *

filaInicial = Picture.under(rock, Picture.negative(square))
filaInicial = Picture.join(filaInicial, Picture.under(knight, square))
filaInicial = Picture.join(filaInicial, Picture.under(bishop, Picture.negative(square)))
filaInicial = Picture.join(filaInicial, Picture.under(queen, square))
filaInicial = Picture.join(filaInicial, Picture.under(king, Picture.negative(square)))
filaInicial = Picture.join(filaInicial, Picture.under(bishop, square))
filaInicial = Picture.join(filaInicial, Picture.under(knight, Picture.negative(square)))
filaInicial = Picture.join(filaInicial, Picture.under(rock, square))
```

- Se crea una fila con las piezas blancas en su posición inicial alternando los colores de los cuadrados.
- Creación de la fila inicial para las piezas negras:

Listing 20: Fila inicial para piezas negras

```
filaNegro = Picture.negative(filaInicial)
```

- Se invierten los colores de la fila inicial blanca para obtener la fila inicial negra.
- Creación de la fila de peones:

Listing 21: Fila de peones

```
filaPeon = Picture.under(pawn, square)
filaPeon = Picture.join(filaPeon, Picture.under(pawn, Picture.negative(square)))
filaPeon = Picture.horizontalRepeat(filaPeon, 4)
```

- Se crea una fila de peones alternando colores y repitiendo el patrón para obtener 8 peones.
- Creación de un patrón de tablero de ajedrez vacío:

Listing 22: Patrón de tablero vacío

```
tabla1 = square
tabla1 = Picture.join(tabla1, Picture.negative(square))
tabla1 = (Picture.horizontalRepeat(tabla1, 4))
tabla2 = Picture.negative(tabla1)
tabla3 = Picture.verticalRepeat(Picture.up(tabla2, tabla1), 2)
```

- Se crea un patrón de tablero vacío de 8x8 cuadrados alternando los colores.
- Construcción del tablero completo con piezas:

Listing 23: Tablero completo con piezas

```
tablero = Picture.up(Picture.negative(filaPeon), filaNegro)
tablero = Picture.up(tabla3, tablero)
tablero = Picture.up(filaPeon, tablero)
tablero = Picture.up(filaInicial, tablero)

draw(tablero)
```

- Se colocan las filas de piezas y peones en el tablero vacío para formar el tablero completo.
- Ver la creación de la imagen:

- Código completo:

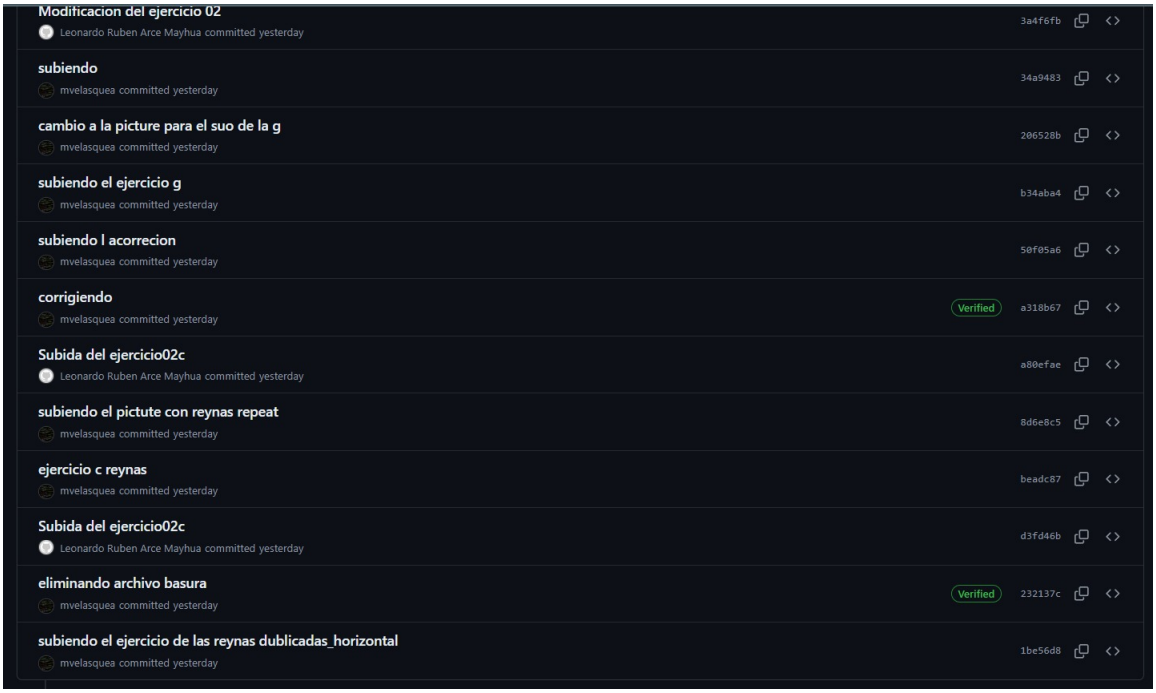
Listing 24: Código completo

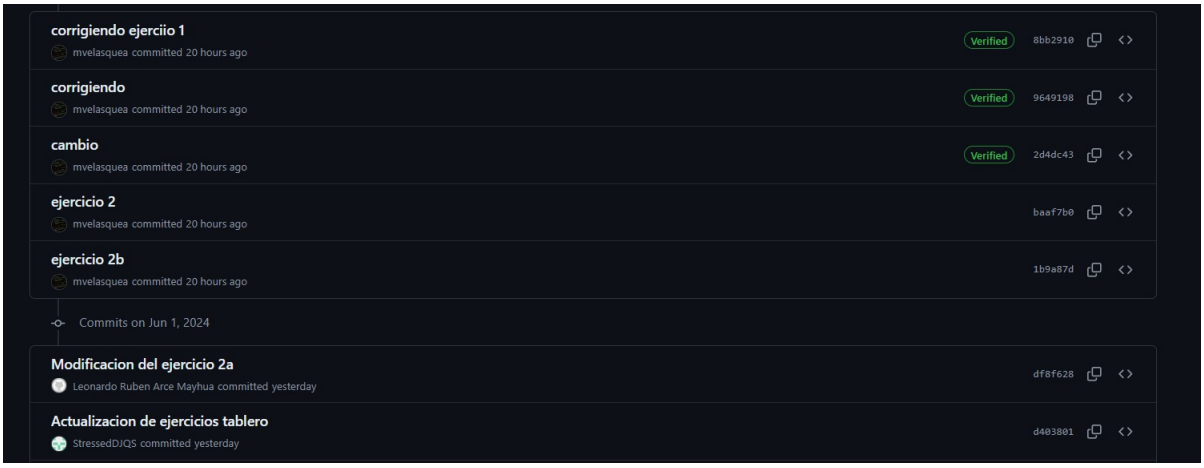
```
from chessPictures import *
from interpreter import draw
from picture import Picture
from colors import *

#Ejercicio g tablero completo

filaInicial = Picture.under(rock, Picture.negative(square))
filaInicial = Picture.join(filaInicial, Picture.under(knight, square))
filaInicial = Picture.join(filaInicial, Picture.under(bishop, Picture.negative(square)))
filaInicial = Picture.join(filaInicial, Picture.under(queen, square))
filaInicial = Picture.join(filaInicial, Picture.under(king, Picture.negative(square)))
filaInicial = Picture.join(filaInicial, Picture.under(bishop, square))
filaInicial = Picture.join(filaInicial, Picture.under(knight, Picture.negative(square)))
filaInicial = Picture.join(filaInicial, Picture.under(rock, square))

filaNegro = Picture.negative(filaInicial)

filaPeon = Picture.under(pawn, square)
filaPeon = Picture.join(filaPeon, Picture.under(pawn, Picture.negative(square)))
filaPeon = Picture.horizontalRepeat(filaPeon, 4)

tabla1 = square
tabla1 = Picture.join(tabla1, Picture.negative(square))
tabla1 = (Picture.horizontalRepeat(tabla1, 4))
tabla2 = Picture.negative(tabla1)
tabla3 = Picture.verticalRepeat(Picture.up(tabla2, tabla1), 2)

tablero = Picture.up(Picture.negative(filaPeon), filaNegro)
tablero = Picture.up(tabla3, tablero)
tablero = Picture.up(filaPeon, tablero)
tablero = Picture.up(filaInicial, tablero)

draw(tablero)
```

10 URL GitHub

- https://github.com/mvelasquea/Pweb2_E_04.git

11 Commits


Modificacion del ejercicio 02
Leonardo Ruben Arce Mayhua committed yesterday
3a4f6fb
subiendo
mvelasquea committed yesterday
34a9483
cambio a la picture para el suo de la g
mvelasquea committed yesterday
206528b
subiendo el ejercicio g
mvelasquea committed yesterday
b34aba4
subiendo l acorrecion
mvelasquea committed yesterday
50f05a6
corrigiendo
mvelasquea committed yesterday
Verified
a318b67
Subida del ejercicio02c
Leonardo Ruben Arce Mayhua committed yesterday
a80efae
subiendo el pictute con reynas repeat
mvelasquea committed yesterday
8d6e8c5
ejercicio c reynas
mvelasquea committed yesterday
beadc87
Subida del ejercicio02c
Leonardo Ruben Arce Mayhua committed yesterday
d3fd46b
eliminando archivo basura
mvelasquea committed yesterday
Verified
232137c
subiendo el ejercicio de las reynas dublicadas_horizontal
mvelasquea committed yesterday
1be56d8


corrigiendo ejerciio 1
mvelasquea committed 20 hours ago
Verified
8bb2910
corrigiendo
mvelasquea committed 20 hours ago
Verified
9649198
cambio
mvelasquea committed 20 hours ago
Verified
2d4dc43
ejercicio 2
mvelasquea committed 20 hours ago
baaf7b0
ejercicio 2b
mvelasquea committed 20 hours ago
1b9a87d
Commits on Jun 1, 2024
Modificacion del ejercicio 2a
Leonardo Ruben Arce Mayhua committed yesterday
df8f628
Actualizacion de ejercicios tablero
StressedDJQS committed yesterday
d403801

12 Rúbricas

12.1 Rúbrica para entregable Informe

Tabla 1: Rúbrica para tipo de Informe

	Informe	Cumple	No cumple
Latex	El informe está en formato PDF desde Latex, con un formato limpio (buena presentación) y fácil de leer.	20	17
MarkDown	El informe está en formato PDF desde MarkDown README.md, con un formato limpio (buena presentación) y fácil de leer.	17	0
MS Word	El informe está en formato PDF desde plantilla MS Word, con un formato limpio (buena presentación) y fácil de leer.	15	0
Observaciones	Por cada observación se le descontará puntos.	-	-

12.2 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 2: Niveles de desempeño

height

	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

height

Tabla 3: 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	1	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	X	2	
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	1	
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	3	
	Total	20		16	