Informe de Laboratorio 04

Tema: Python

Nota

Estudiante	Escuela	Asignatura
Piero Douglas Mejia Ramos pmejia@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Programación Web 2

Laboratorio	Tema	Duración
04	Python	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - A	Del 29 Mayo 2023	Al 05 Junio 2023

1. Tarea

- URL GitHub de Tarea del Ajedrez https://github.com/rescobedoq/pw2/tree/main/labs/ lab04/Tarea-del-Ajedrez
- Para resolver los siguientes ejercicios sòlo esta permitido usar ciclos, condicionales, definicion de listas por comprensi on, 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 unicamente los métodos de los objetos de la clase Picture dibuje las siguientes figuras (invoque a draw):

2. Equipos, materiales y temas utilizados

- Listas
- Python
- Ciclos
- Programación orientada a objetos
- Programación funcional
- Librerias de pip como pygame

3. URL de Repositorio Github

- URL para el laboratorio 04 en el Repositorio GitHub.
- `https://github.com/pieroMejiaR/PWeb2-lab/tree/main/lab04`

4. Actividades realizadas

4.1. Commits

Listing 1: Primer Commit Creando carpeta para laboratorio 04

```
$ mkdir lab04
$ git add .
$ git commit -m "Creando carpeta para laboratorio 04"
$ git push -u origin main
```

Listing 2: Agregando metodo join de la clase Picture

```
41    def join(self, p):
42        joined = []
43        for i in range(len(self.img)):
44            joined.append(self.img[i] + p.img[i])
45        return Picture(joined)
46
```

Figura 1: Agregando metodo join

- El metodo join recibe 2 parametros: self y p
- El metodo se encarga de juntar las lineas de cadena de String de cada uno y devolver un nuevo objeto Picture

Listing 3: Agregando metodo negative de la clase Picture

```
27    def negative(self):
28        # Crear una lista vacía para almacenar las líneas de la imagen negativa
29        negative_img = []
30
31        # Iterar sobre las líneas de la imagen actual
32        for line in self.img:
33            # Reemplazar el color de cada carácter en la línea
34            inverted_line = [self._invColor(c) for c in line]
35            # Agregar la línea modificada a la lista de la imagen negativa
36            negative_img.append("".join(inverted_line))
37
38        # Crear y retornar una nueva instancia de Picture con la imagen negativa
39        return Picture(negative_img)
```

Figura 2: Captura del método negative

- El metodo negative reemplaza el color de cada linea de la imagen mandada

Listing 4: Completando Ejercicio2a

```
    $ git add Ejercicio2a
    $ git commit -m "Completando Ejercicio2a"
```

Listing 5: Ejercicio2a.py

```
from chessPictures import *
from interpreter import draw
from pieces import *
from picture import *


white_knight = Picture(KNIGHT)
black_knight = white_knight.negative()

board_knights = white_knight.join(black_knight).up(black_knight.join(white_knight))

draw(board_knights)
```

- Se soluciona usando los metodos join y up de la clase Picture

Figura 3: Compilando y probando ejercicicio2a

Listing 6: Completando Ejercicio2b

```
    $ git add Ejercicio2b
    $ git commit -m "Completando Ejercicio2b"
```

Listing 7: Ejercicio2b.py

```
from interpreter import draw
from chessPictures import *

white_knight = Picture(KNIGHT)
black_knight = white_knight.negative()

```

```
board_knights =
    white_knight.join(black_knight).up(black_knight.verticalMirror().join(white_knight.verticalMirror()))

draw(board_knights)
```

- Es similar al ejercicio2a pero en este caso támbien se usa el metodo verticalMirror()

Figura 4: Compilando y probando ejercicicio2b

Listing 8: Completando Ejercicio2c

```
$ git add Ejercicio2c
$ git commit -m "Completando Ejercicio2c"
```

Listing 9: Ejercicio2c.py

```
from interpreter import draw
from chessPictures import *

white_queen = Picture(QUEEN)

board_queens = white_queen.horizontalRepeat(4)

draw(board_queens)
```

- En este ejercicio se hace uso del método horizontalRepeat()

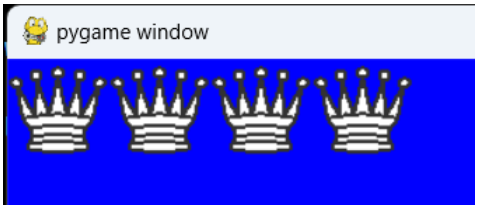


Figura 5: Compilando y probando ejercicicio2c

Listing 10: Completando Ejercicio2d

```
    $ git add Ejercicio2d
    $ git commit -m "Completando Ejercicio2d"
```

Listing 11: Ejercicio2d.py

```
from interpreter import draw
from chessPictures import *

white_space = Picture(SQUARE)
black_space = white_space.negative()

board = white_space.join(black_space).horizontalRepeat(4)

draw(board)
```

- Se usa Picture(SQUARE) para crear el tablero de ajedrez
- Se usa el método horizontal repeat para facilitar el código

Figura 6: Compilando y probando ejercicicio2d

Listing 12: Completando Ejercicio2f

```
    $ git add Ejercicio2f
    $ git commit -m "Completando Ejercicio2f"
```

Listing 13: Ejercicio2f.py

```
from interpreter import draw
from chessPictures import *

white_space = Picture(SQUARE)
black_space = white_space.negative()

board =
    white_space.join(black_space).horizontalRepeat(4).under(black_space.join(white_space).horizontalRepeat(4)).v
```

```
8
9 draw(board)
```

- Se usa lo hecho en los ejercicios 2d y 2e
- Se añade el metodo verticalRepeat()

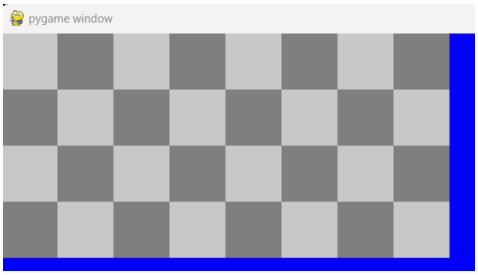
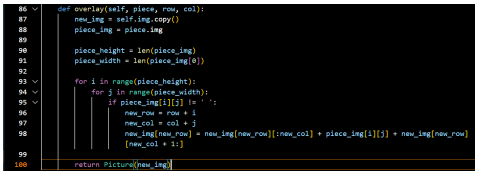


Figura 7: Compilando y probando ejercicicio2f

Listing 14: Agregando metodo overlay de la clase Picture

- El metodo overlay recibe 4 parametros un objeto Picture, una pieza y dos posiciones
- Se usa para colocar las piezas en los espacios del tablero



```
86  def overlay(self, piece, row, col):
87      new_img = self.img.copy()
88      piece_img = piece.img
89
90      piece_height = len(piece_img)
91      piece_width = len(piece_img[0])
92
93      for i in range(piece_height):
94          for j in range(piece_width):
95              if piece_img[i][j] != ' ':
96                  new_row = row + i
97                  new_col = col + j
98                  new_img[new_row] = new_img[new_row][:new_col] + piece_img[i][j] + new_img[new_row]
                    [new_col + 1:]
99
100     return Picture(new_img)
```

Figura 8: Agregando metodo overlay

Listing 15: Completando Ejercicio2g

```
$ git add Ejercicio2g
$ git commit -m "Completando Ejercicio2g"
```

Listing 16: Ejercicio2g.py

```
from interpreter import draw
from chessPictures import *

white_space = Picture(SQUARE)
black_space = white_space.negative()

black_rock = white_space.join(black_space.join(rock)).join(white_space)
board = white_space.join(black_space.up(knight.negative())).up(black_space.join(white_space))

draw(knight.merge(queen))
```

- Se usa el metodo overlay para colocar las piezas en sus espacios correspondientes
- Se usa lo hecho en los ejercicios anteriores

Figura 9: Compilando y probando ejercicicio2g

5. Cuestionario

5.1. ¿Qué son los archivos *.pyc?

Los archivos con extensión ".pyc"son archivos de código de byte generados por el intérprete de Python. Cuando se ejecuta un programa de Python, el código fuente se compila en un formato de código de byte, que es más eficiente para ser interpretado por la máquina virtual de Python. Los archivos .pyc contienen esta versión compilada del código fuente y se utilizan para acelerar la carga y ejecución de programas de Python en futuras ocasiones.

5.2. ¿Para qué sirve el directorio pycache?

El directorio "pycache.[es] utilizado por el intérprete de Python para almacenar los archivos .pyc generados durante la compilación de código fuente en módulos. Este directorio se crea automáticamente en el mismo directorio que el archivo .py cuando se importa un módulo.

La finalidad del directorio "pycache.[es] proporcionar un lugar específico para almacenar los archivos .pyc y evitar el desorden en el directorio principal donde se encuentran los archivos .py. Al separar los archivos .pyc en un directorio aparte, se mejora la organización del proyecto y se mantiene más limpio el espacio de trabajo.

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

El subrayado o guion bajo tiene diversos usos y significados en Python. En primer lugar, se utiliza como un nombre de variable descartable, cuando se quiere ignorar un valor en un bucle o en una asignación. Además, el subrayado al comienzo de un nombre de variable se emplea como convención para indicar que esa variable es de uso interno y no debe ser accedida directamente desde fuera de la clase o módulo.

6. Rúbricas

6.1. Entregable Informe

Tabla 1: Tipo de Informe

Informe	
Latex	El informe está en formato PDF desde Latex, con un formato limpio (buena presentación) y facil de leer.

6.2. 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

	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 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			
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4			
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2			
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2			
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2			
6. Fechas	Las fechas de modificación del código fuente estan dentro de los plazos de fecha de entrega establecidos.	2			
7. Ortografía	El documento no muestra errores ortográficos.	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			
Total		20			

7. Referencias

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