Informe de Laboratorio 05

Tema: Python

Nota

Estudiante	Escuela	Asignatura
David Alfredo Huamani Ollachica dhuamanio@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Programación Web 2 Semestre: III Código: 20230485

Laboratorio	Tema	Duración
05	Python	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2024 - A	Del 27 Mayo 2024	Al 31 Mayo 2024

1. Tarea

- URL GitHub de Tarea del Ajedrez `https://github.com/dev1d123/pw2_lab05`
- Imagen de los commits realizados

Listing 1: imagen de los commits realizados

```
git log --graph --pretty=oneline --abbrev-commit --all
* a2fef64 (HEAD -> main, origin/main, origin/HEAD) Todos los ejercicios implementados
* 0d095f3 Corregir los return en las funciones
* bf6a347 Correccion del argumento devuelto en negative
* 7f9fe68 Correcion del metodo under
* 592a40a Logica e implementacion para el metodo rotate
* 42e4b7d Implementacion del metodo verticalRepeat, se reutiliza codigo de up
* e2c04bb Implementacion del metodo HorizontalRepeat, se reutiliza codigo de join
* 1869c7f Implementacion del metodo under, solo invertir lo de up
* e37e608 Implementacion del metodo up terminada
* a096cfe Correcion general del metodo join, devuelve un objeto del tipo Picture!
* 016b012 Corregir el metodo join, si el parametro era una lista o una sola pieza
* 113f841 Metodo join
* e45fd28 Metodo para devolver el negativo de una imagen completado!
* 2b96dc1 Implementacion del metodo horizontalMirror
* 03b2479 Agregando los archivos iniciales del repositorio
* 60569f8 Initial commit
```

2. Metodos de la clase Picture

2.1. verticalMirror y horizontalMirror

- Para el método `verticalMirror`, se genera un arreglo `vertical[]`, el cual tiene todos los strings de `self.img`, y los agrega al arreglo de manera invertida, debido al `[::-1]`.
- Para el método `horizontalMirror`, se declara un arreglo `horizontal[]` y se le pasan los strings de manera revertida de `self.img` usando `reversed(self.img)`.

Listing 2: metodos verticalMirror y horizontalMirror

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

def horizontalMirror(self):
  horizontal = []
  for value in reversed(self.img):
    horizontal.append(value[::1])
  return Picture(horizontal)
```

2.2. negative y _invColor

- Para este método, se itera en cada elemento del parámetro enviado `self.img`. Luego, para cada string de este, se itera carácter por carácter. A cada carácter se le aplica el método `_invColor`, el cual recibe un carácter y, dependiendo del carácter, en el archivo `colors.py` se crea un mapa llamado `inverter`, el cual retorna el carácter con el color opuesto.

Listing 3: metodo 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)
```

Listing 4: metodo _invColor

```
def _invColor(self, color):
  if color not in inverter:
    return color
  return inverter[color]
```

Listing 5: caracteres opuestos en colors.py

```
inverter = {
  '_': '=',
  '=': '_',
  '.': '@',
```

```
  '@': '.',
}
```

2.3. join

- Para este método, se genera un nuevo arreglo llamado `nuevaImagen[]`. A este arreglo se le agregan todos los strings de la lista `self.img`, cada uno convertido en una lista, debido a que la clase `string` es inmutable. Luego, se concatenan con cada string correspondiente del parámetro `p.img`.
- Esto se logra mediante el método `enumerate`, el cual permite devolver el índice y el valor correspondiente. El índice se utiliza para acceder al arreglo de `p.img`.

Listing 6: metodo join

```
def join(self, p):
  nuevaImagen = []
  for index, value in enumerate(self.img):
    nuevaImagen.append(list(value) + list(p.img[index]))
  return Picture(nuevaImagen)
```

2.4. up

- Este método es muy simple: solo coloca la figura `p` encima de `self`. Para lograr esto, se crea un nuevo arreglo llamado `nuevaImagen[]`. Primero, se agregan los strings de `p.img` y luego se añaden los strings de la figura que debe ir debajo, es decir, de `self.img`.

Listing 7: metodo up

```
def up(self, p):
  nuevaImagen = []
  for value in p.img:
    nuevaImagen.append(value[::1])
  for value in self.img:
    nuevaImagen.append(value[::1])
  return Picture(nuevaImagen)
```

2.5. under

- Este método devuelve la figura `p` sobre la figura `self`. Primero, se llena el arreglo `nuevaImagen[]` con los caracteres de `self`, a modo de fondo de la imagen, sin embargo se agregan como una lista para agregar los caracteres facilmente. Luego, para cada carácter de `p.img`, se agrega a `nuevaImagen`, si y solo si esa posición está vacía.

Listing 8: metodo 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)
```

2.6. horizontalRepeat

- Se utiliza el metodo join n veces

Listing 9: metodo horizontalRepeat

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

2.7. verticalRepeat

- Se utiliza el metodo up n veces

Listing 10: metodo verticalRepeat

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

2.8. rotate

- Este método devuelve la figura rotada 90 grados. Para la implementación de este método, se definen los índices normales y luego los índices rotados. Tras un análisis, se llega a la conclusión de que: el índice de la matriz rotada $[n-j][i]$ es igual al índice $[i][j]$.

$$\text{rotated\_matrix}[n-j][i] = \text{original\_matrix}[i][j]$$

  Esto rota una matriz 90 grados en sentido horario. Para la implementacion se accede a cada elemento de la matriz manualmente y se modifica manualmente.

Listing 11: rotate

```
def rotate(self):


  """
  00 01 02 03 0n
  10 11 12 13 1n
  20 21 22 23 2n
  30 31 32 33 3n
  n0 n1 n2 n3 nn

  n0 30 20 10 00
  n1 31 21 11 01
  n2 32 22 12 02
  n3 33 23 13 03
  nn 3n 2n 1n 0n

  -> The index ij -> n - j, i
  """

  matrizOriginal = []

  for value in self.img:
    matrizOriginal.append(value)

  size = len(matrizOriginal)

  matrizRotate = [['' for _ in range(size)] for _ in range(size)] #crea una matriz vacia!

  for i in range(len(matrizOriginal)):
    for j in range(len(matrizOriginal[i])):
      matrizRotate[size - j - 1][i] = matrizOriginal[i][j]

  return Picture(matrizRotate)
```

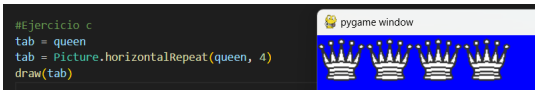
3. Ejecución del código

3.1. Implementación de todos los ejercicios

- Codigo y ejecucion de los metodos e imagenes de la ejecucion

(a) Código y ejecución del ejercicio A

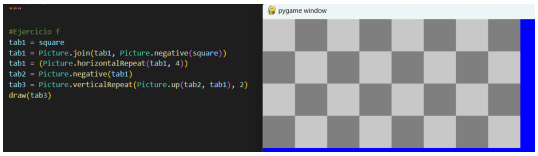
(b) Código y ejecución del ejercicio B



(c) Código y ejecución del ejercicio C

(d) Código y ejecución del ejercicio D

(e) Código y ejecución del ejercicio E



(f) Código y ejecución del ejercicio F

(g) Código y ejecución del ejercicio G

Figura 1: Implementación y ejecución de los ejercicios

4. Pregunta

4.1. ¿Para qué sirve el directorio pycache?

- Cuando ejecutas un programa en Python, el intérprete primero lo compila a bytecode (esto es una simplificación excesiva) y lo almacena en la carpeta `__pycache__`.

  Si miras dentro de esa carpeta, encontrarás un montón de archivos que comparten los nombres de los archivos .py en la carpeta de tu proyecto, solo que sus extensiones serán .pyc o .pyo. Estos son versiones compiladas a bytecode y versiones compiladas y optimizadas a bytecode de los archivos de tu programa, respectivamente.

  Todo lo que hace es hacer que tu programa arranque un poco más rápido. Cuando tus scripts cambian, serán recompilados, y si eliminas los archivos o la carpeta completa y ejecutas tu programa nuevamente, volverán a aparecer (a menos que suprimas específicamente ese comportamiento).

5. Rúbricas

5.1. Sobre el 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.

5.2. 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	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	3	
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	3	
Total		20		18	

6. Referencias

- https://www.pygame.org/news
- https://micro.recursospython.com/recursos/la-funcion-enumerate.html
- https://docs.python.org/es/3/tutorial/venv.html
- https://www.w3schools.com/python/
- https://stackoverflow.com/questions/16869024/what-is-pycache