

**Laboratorio 03**  
**Tema: JavaScript**

Estudiante	Escuela	Asignatura
Donny Moises Mara Mamani dmara@ unas.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Programación Web Semestre: III Código: 1702122

















Laboratorio	Tema	Duración
05	JavaScript	06 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2024 - A	31 Mayo 2024	5 junio 2024

## 1. Tarea

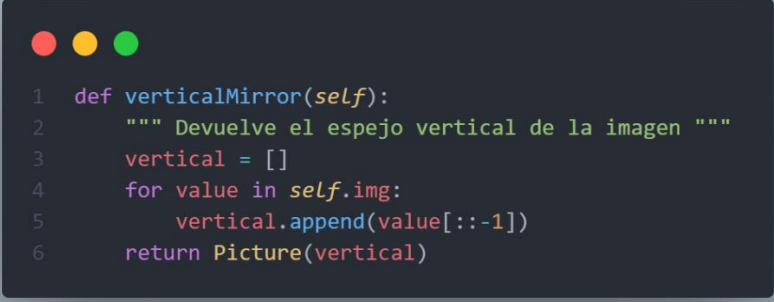
### 1.1. Descripción

#### ■ Commits:

<b>Resolucion de ejercicios</b> MMaraP committed 48 minutes ago	c7d4d7b	 
<b>metodos de rotacion</b> MMaraP committed 1 hour ago	876af74	 
<b>metodo de repeticiones verticales</b> MMaraP committed 2 hours ago	f51e90d	 
<b>metodo de repeticiones horizontales</b> MMaraP committed 2 hours ago	6e15090	 
<b>metodos up y under</b> MMaraP committed 2 hours ago	344cd07	 
<b>metodo join y prueba</b> MMaraP committed 2 hours ago	8c66a6d	 
<b>metodo negative</b> MMaraP committed 2 hours ago	bdf8ece	 
<b>metodo para invertir la pieza de manera horizontal</b> MMaraP committed 2 hours ago	501629c	 

Codigo:

#### METODOS DE LA CLASE PICTURE



```

1 def verticalMirror(self):
2     """ Devuelve el espejo vertical de la imagen """
3     vertical = []
4     for value in self.img:
5         vertical.append(value[::-1])
6     return Picture(vertical)

```

Se crea una lista vacía llamada "vertical" para almacenar las filas de la imagen espejada verticalmente.

Se itera sobre cada fila de la imagen original representada por self.img.

Para cada fila de la imagen original, se invierte el orden de los caracteres utilizando la sintaxis de rebanado [::-1], lo que equivale a espejarla verticalmente.

La fila espejada verticalmente se agrega a la lista "vertical".

Finalmente, se devuelve una nueva instancia de la clase `Picture`, utilizando la lista "vertical" generada como la nueva representación de la imagen espejada verticalmente.

```
1 def horizontalMirror(self):
2     """ Devuelve el espejo horizontal de la imagen """
3     horizontal = self.img[::-1]
4     return Picture(horizontal)
```

Se utiliza la técnica de rebanado de listas de Python para invertir el orden de las filas de la imagen original `self.img`, creando así una nueva lista llamada "horizontal" que representa la imagen espejada horizontalmente.

Se devuelve una nueva instancia de la clase `Picture`, utilizando la lista "horizontal" generada como la nueva representación de la imagen espejada horizontalmente.

```
1 def negative(self):
2     """ Devuelve un negativo de la imagen """
3     negativo = []
4     for fila in self.img:
5         nueva_fila = ""
6         for color in fila:
7             nueva_fila += self._invColor(color)
8         negativo.append(nueva_fila)
9     return Picture(negativo)
```

Se inicializa una lista vacía llamada "negativo" para almacenar las filas de la imagen negativa.

Se itera sobre cada fila de la imagen original representada por `self.img`.

Para cada fila de la imagen original, se inicializa una cadena vacía llamada "nueva\_fila".

Se itera sobre cada color en la fila actual.

Para cada color, se invoca el método `_invColor` para obtener su color inverso y se agrega a la cadena

"nueva\_fila".

Una vez que se completa la fila, se agrega la cadena "nueva\_fila" a la lista "negativo".

Finalmente, se devuelve una nueva instancia de la clase Picture, utilizando la lista "negativo" generada como la nueva representación de la imagen negativa.

```

1 def join(self, p):
2     """ Devuelve una nueva figura poniendo la figura del argumento
3         al lado derecho de la figura actual """
4     nueva_img = []
5     for fila1, fila2 in zip(self.img, p.img):
6         nueva_img.append(fila1 + fila2)
7     return Picture(nueva_img)

```

Se inicializa una lista vacía llamada nueva\_img para almacenar las filas de la nueva imagen combinada.

Se itera simultáneamente sobre las filas de la imagen actual representada por self.img y las filas de la imagen proporcionada como argumento, p.img, utilizando la función zip.

Para cada par de filas correspondientes, se concatenan horizontalmente utilizando el operador + y se agrega la fila resultante a la lista nueva\_img.

Una vez que se completan todas las filas, se devuelve una nueva instancia de la clase Picture, utilizando la lista nueva\_img generada como la nueva representación de la imagen combinada.

```

1 def up(self, p):
2     """ Devuelve una nueva figura con la pieza del argumento sobre la figura actual """
3     nueva_img = []
4     max_length = max(len(self.img), len(p.img))
5     for i in range(max_length):
6         fila_actual = self.img[i] if i < len(self.img) else ""
7         fila_nueva = p.img[i] if i < len(p.img) else ""
8         nueva_fila = ""
9         for j in range(max(len(fila_actual), len(fila_nueva))):
10             if j < len(fila_nueva) and fila_nueva[j] != " ":
11                 nueva_fila += fila_nueva[j]
12             else:
13                 nueva_fila += fila_actual[j] if j < len(fila_actual) else " "
14         nueva_img.append(nueva_fila)
15     return Picture(nueva_img)

```

Se inicializa una lista vacía llamada nueva\_img para almacenar las filas de la nueva imagen combinada.

Se determina la longitud máxima entre el número de filas de la imagen actual (self.img) y la imagen

proporcionada (p.img), ya que la nueva imagen será la combinación vertical de ambas.

Se itera sobre un rango que va desde 0 hasta la longitud máxima menos uno.

En cada iteración, se obtiene la fila correspondiente de la imagen actual (self.img) y de la imagen proporcionada (p.img). Si una de las imágenes tiene menos filas que la otra, se utiliza una fila vacía en su lugar.

Se crea una nueva fila combinando los caracteres de ambas filas, donde los caracteres de la imagen proporcionada (p.img) tienen prioridad si no son espacios en blanco.

La nueva fila se agrega a la lista nueva\_img.

Una vez que se han combinado todas las filas, se devuelve una nueva instancia de la clase Picture, utilizando la lista nueva\_img generada como la nueva representación de la imagen combinada verticalmente.

```
1 def under(self, p):
2     """ Devuelve una nueva figura poniendo la figura p sobre la
3         figura actual """
4     return Picture(self.img + p.img)
```

La función recibe como parámetro la imagen p que se colocará debajo de la imagen actual.

Se crea una nueva instancia de la clase Picture concatenando verticalmente la imagen actual (self.img) con la imagen p.img.

La nueva imagen resultante representa la combinación de ambas imágenes, con la imagen p ubicada debajo de la imagen actual.

```
1 def horizontalRepeat(self, n):
2     """ Devuelve una nueva figura repitiendo la figura actual al costado
3         la cantidad de veces que indique el valor de n """
4     nueva_img = []
5     for fila in self.img:
6         nueva_fila = fila * n
7         nueva_img.append(nueva_fila)
8     return Picture(nueva_img)
```

Se inicializa una lista vacía llamada `nueva_img` para almacenar las filas de la nueva imagen repetida.

Se itera sobre cada fila de la imagen original representada por `self.img`.

Para cada fila de la imagen original, se duplica horizontalmente utilizando el operador de multiplicación `*`, donde el número de repeticiones está determinado por el argumento `n`.

La fila duplicada se agrega a la lista `nueva_img`.

Una vez que se completan todas las filas, se devuelve una nueva instancia de la clase `Picture`, utilizando la lista `nueva_img` generada como la nueva representación de la imagen repetida horizontalmente.

```
1 def verticalRepeat(self, n):
2     """ Devuelve una nueva figura repitiendo la figura actual hacia abajo
3         la cantidad de veces que indique el valor de n """
4     return Picture(self.img * n)
```

La función recibe como parámetro `n`, que representa el número de repeticiones verticales de la imagen actual.

Se devuelve una nueva instancia de la clase `Picture`, donde la imagen original (`self.img`) se replica verticalmente `n` veces.

```
1 def rotate(self):
2     """ Devuelve una figura rotada en 90 grados, puede ser en sentido horario """
3     rotada = []
4     for i in range(len(self.img[0])):
5         nueva_fila = ""
6         for fila in reversed(self.img):
7             nueva_fila += fila[i]
8         rotada.append(nueva_fila)
9     return Picture(rotada)
```

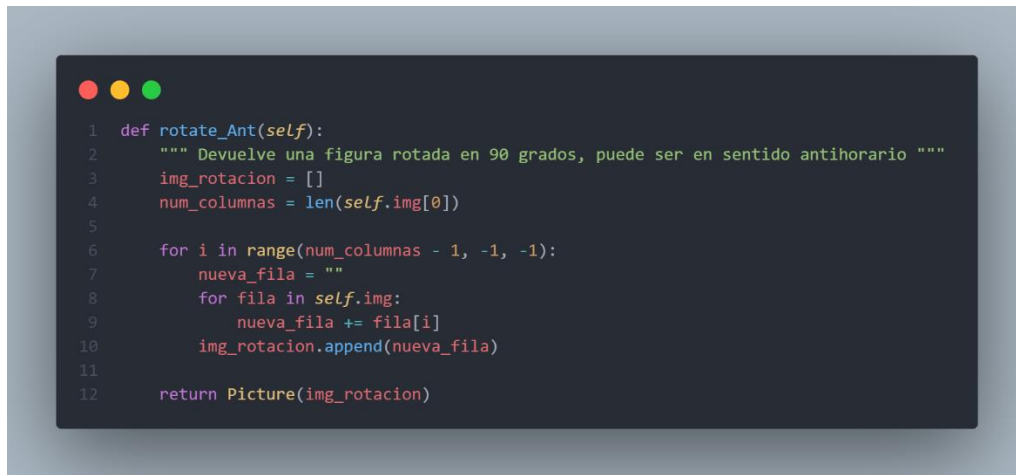
Se inicializa una lista vacía llamada `rotada` para almacenar las filas de la nueva imagen rotada.

Se itera sobre los índices de las columnas de la imagen original.

Para cada índice de columna, se recorre en orden inverso cada fila de la imagen original.

Se construye una nueva fila en la lista `rotada` concatenando los caracteres de la columna correspondiente de cada fila en orden inverso.

Una vez que se completan todas las filas, se devuelve una nueva instancia de la clase Picture, utilizando la lista rotada generada como la nueva representación de la imagen rotada.



```

1  def rotate_Ant(self):
2      """ Devuelve una figura rotada en 90 grados, puede ser en sentido antihorario """
3      img_rotacion = []
4      num_columnas = len(self.img[0])
5
6      for i in range(num_columnas - 1, -1, -1):
7          nueva_fila = ""
8          for fila in self.img:
9              nueva_fila += fila[i]
10             img_rotacion.append(nueva_fila)
11
12     return Picture(img_rotacion)

```

Se inicializa una lista vacía llamada img\_rotacion para almacenar las filas de la nueva imagen rotada en sentido antihorario.

Se determina el número de columnas en la imagen original.

Se itera sobre los índices de las columnas en orden inverso.

Para cada índice de columna, se recorre cada fila de la imagen original.

Se construye una nueva fila en la lista img\_rotacion concatenando los caracteres de la columna correspondiente de cada fila.

Una vez que se completan todas las filas, se devuelve una nueva instancia de la clase Picture, utilizando la lista img\_rotacion generada como la nueva representación de la imagen rotada en sentido antihorario. Se inicializa una lista vacía llamada img\_rotacion para almacenar las filas de la nueva imagen rotada en sentido antihorario.

Se determina el número de columnas en la imagen original.

Se itera sobre los índices de las columnas en orden inverso.

Para cada índice de columna, se recorre cada fila de la imagen original.

Se construye una nueva fila en la lista img\_rotacion concatenando los caracteres de la columna correspondiente de cada fila.

Una vez que se completan todas las filas, se devuelve una nueva instancia de la clase Picture, utilizando la lista img\_rotacion generada como la nueva representación de la imagen rotada en sentido antihorario.

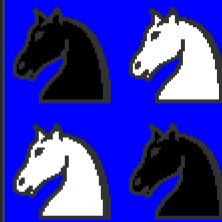
## 1.2. Ejercicios

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

pieza = knight
dupla = Picture.join(pieza, pieza.negative())
tablero = Picture.under(dupla.negative(), dupla)

draw(tablero)
```

 pygame window

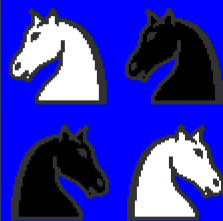


```
from interpreter import draw
from chessPictures import *
from picture import Picture
from colors import *

pieza = knight
dupla = Picture.join(pieza, pieza.negative())
tablero = Picture.under(dupla, dupla.verticalMirror())

draw(tablero)
```

 pygame window

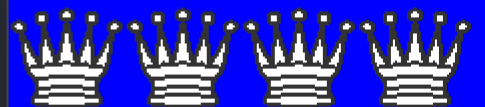


```
from interpreter import draw
from chessPictures import *
from picture import Picture
from colors import *

pieza = queen
tablero = Picture.horizontalRepeat(pieza, 4)

draw(tablero)
```

 pygame window





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

casilla = square
dupla = Picture.join(casilla, casilla.negative())
tablero = Picture.horizontalRepeat(dupla, 4)

draw(tablero)
```

pygame window



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

casilla = square
dupla = Picture.join(casilla.negative(), casilla)
tablero = Picture.horizontalRepeat(dupla, 4)

draw(tablero)
```

pygame window

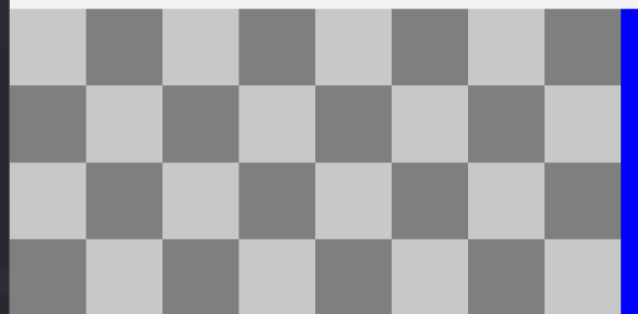


```
from interpreter import draw
from chessPictures import *
from picture import Picture
from colors import *

casilla = square
dupla = Picture.join(casilla, casilla.negative())
tablero = Picture.horizontalRepeat(dupla, 4)
tablero2 = Picture.negative(tablero)
tablero3 = Picture.under(tablero, tablero2)

draw(tablero3.verticalRepeat(2))
```

pygame window



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

casB = square
casN = casB.negative()

dupla1 = Picture.join(casN.up(rock), casB.up(knight))
dupla2 = Picture.join(casN.up(bishop), casB.up(queen))
dupla3 = Picture.join(casN.up(king), casB.up(bishop))
dupla4 = Picture.join(casN.up(knight), casB.up(rock))
fila1 = dupla1.join(dupla2).join(dupla3).join(dupla4)
dupla5 = Picture.join(casB.up(pawn), casN.up(pawn))
fila2 = dupla5.horizontalRepeat(4)
fila3 = Picture.join(casN, casB).horizontalRepeat(4)

tablero1 = fila1.negative().under(fila2.negative())
tablero2 = (fila3.negative().under(fila3)).verticalRepeat(2)
tablero3 = fila2.under(fila1)

draw(tablero1.under(tablero2).under(tablero3))
```



### 1.3. Pregunta

- Explique ¿Para qué sirve el directorio pycache?

El directorio `__pycache__` es un directorio especial utilizado por Python para almacenar archivos de caché que contienen el byte-code compilado de los archivos fuente de Python (.py). Estos archivos de caché se generan cuando se importan módulos en un programa Python.

La razón principal de la existencia del directorio `__pycache__` es mejorar el rendimiento de las aplicaciones Python. En lugar de compilar el código fuente en byte-code cada vez que se importa un módulo, Python almacena una versión compilada en este directorio. De esta manera, cuando se importa un módulo, Python primero verifica si hay una versión compilada en el directorio `__pycache__`. Si encuentra una coincidencia y el archivo fuente no ha cambiado desde la última compilación, Python utilizará el archivo de caché, lo que ahorra tiempo de compilación y mejora el rendimiento general del programa.

El directorio `__pycache__` se crea automáticamente en el mismo directorio que el archivo fuente de Python cuando se importa un módulo por primera vez. Si el módulo se importa desde varios lugares, Python creará archivos de caché separados para cada directorio de importación.

## 1.4. Entregables

- <https://github.com/MMaraP/pw2-24a>

## 2. Rúbricas

Tabla 3: Rúbrica para contenido del Informe y evidencias

Contenido y demostración		Puntos	Checklist	Estudiante	Profesor
<b>1. GitHub</b>	Repositorio se pudo clonar y se evidencia la estructura adecuada para revisar los entregables. (Se descontará puntos por error o omisión)	4	✓	3	
<b>2. Commits</b>	Hay porciones de código fuente asociado a los commits planificados con explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	✓	3	
<b>3. Ejecución</b>	Se incluyen comandos para ejecuciones y pruebas del código fuente explicadas gradualmente que permitirían replicar el proyecto. (Se descontará puntos por cada omisión)	4	✓	2	
<b>4. Pregunta</b>	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	✓	1	
<b>7. Ortografía</b>	El documento no muestra errores ortográficos. (Se descontará puntos por error encontrado)	2	✓	2	

<b>8. Madurez</b>	El Informe muestra de manera general una evolución de la madurez del código fuente con explicaciones puntuales pero precisas, agregando diagramas generados a partir del código fuente y refleja un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4	✓	3	
<b>Total</b>		20		14	

### 3. Referencias

- <https://github.com/>
- <https://docs.python.org/es/3/contents.html>
- <https://docs.python.org/es/3.12/faq/programming.html>