Laboratorio 04

Tema: Python

June 6, 2023

Profesor	Escuela	Asignatura
Prof. Annibal Sardon	EPIS	Programacion Web 2

Laboratorio	Tema	Duracion
04	Python	04 Horas

Semestre academico	Fecha de inicio	Fecha de entrega
2023 - A	29 Mayo 2023	06 Junio 2023

Estudiante	Grupo
Carrasco Choque Arles Melvin	B

1 Resolucion de ejercicios

URL de GitHub : https://github.com/carrascoArles/lab04-pweb2.git.

1.1 Clase Pictures

```
29      def negative(self):
30        """ Devuelve un negativo de la imagen """
31        negativo = [
32            ''.join(self._invColor(char) for char in value)#sirve para concatenar los caracteres invertidos en cada
      fila
33            for value in self.img
34        ]
35        return Picture(negativo)
36
37      def join(self, p):
38        """ Devuelve una nueva figura poniendo la figura del argumento
39            al lado derecho de la figura actual """
40        join = []
41        for i in range(len(self.img)):
42            join.append(self.img[i] + "" + p.img[i])
43        return Picture(join)
44
45      def up(self, p):
46        image = self.img + p.img
47        return Picture(image)
```

El metodo negative lo quehace es convertir los caracteres de la imagen que le damos y lo da de otro color y lo va hacer con un metodo base que viene a ser el invcolor pero este solo convierte caracteres por lo que se usa for para cada fila y devolver el arreglo ya convertido.
Luego tenemos el metodo join lo que hace este solo es juntar dos imagenes en la misma fila, para esto vamos a crear el array y con el for vamos a recorrer todas las filas para asi poder usar el append() que

su funcion es agregar al final del arreglo un valor, se itera y lo retornamos.
El metodo up es basico solo usa el + para juntar imagenes pero cada una por separado en cada fila.

```
def under(self, p):
  """ Devuelve una nueva figura poniendo la figura p sobre la
      figura actual """
  image = []
  for i in range(len(self.img)):
      line = ""
      for j in range(len(self.img[i])):
          if p.img[i][j] == " ":
              line += self.img[i][j]

          else:line += p.img[i][j]

      image.append(line)
  return Picture(image)

def horizontalRepeat(self, n):
  """ Devuelve una nueva figura repitiendo la figura actual al costado
      la cantidad de veces que indique el valor de n """
  repetido = []
  for i in range(len(self.img)):
      repetido.append(self.img[i] * n)
  return Picture(repetido)

def verticalRepeat(self, n):
  repetido=self.img * n
  return Picture(repetido)
```

luego hacemos uso del under() este va a recorrer por filas y columnas para verificar los valores de la imagen que se va a poner, entonces tenemos dos casos posibles, si es que el caracter[i][j] de p esta vacio en ese caracter pues solo se pone la imagen del cuadrado, en cambio si p tiene un valor existente se pone ese.

Para el horizontal repeat,muy similar al join, se va recorrer cada fila, agregando con el append una multiplicacion de cada figura siendo estas por filas.

En cambio en el vertical repeat,muy similar al up ,se extrae la imagen propio y completa y solo se multiplica por el tamaño ya que esto lo hace en una fila aparte, mas abajo.

1.2 Ejercicio a

```
1  from interpreter import draw
2  from chessPictures import *
3  knightN = knight.negative()
4  conjunto1 = knight.join(knightN)
5  conjunto2 = knightN.join(knight)
6  conjunto = conjunto1.up(conjunto2)
7
8  draw(conjunto)
```

Se usaron los metodos negative,join y up, se usa las variables predeterminada de las figuras, como se hace por lineas, solo se juntan con el join los primeros caballos, luego solo se invierten y al final se usa un up para concatenar imagenes de manera vertical.

1.3 Ejercicio b

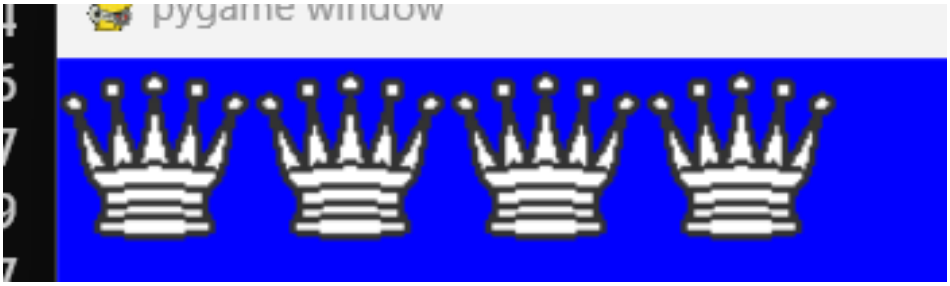
```
1  from interpreter import draw
2  from chessPictures import *
3  knightN = knight.negative()
4  conjunto1 = knight.join(knightN)
5  knightN2 = knightN.verticalMirror()
6  conjunto2 = knightN2.join(knight.verticalMirror())
7  conjunto = conjunto1.up(conjunto2)
8
9  draw(conjunto)
```

Se hizo practicamente lo mismo que el primero, pero en este caso se agrego el vertical mirror para que los caballos miren a otro lado en la segunda fila, y se junta con up.

1.4 Ejercicio c

```
1    from interpreter import draw
2    from chessPictures import *
3
4    conjunto = queen.horizontalRepeat(4)
5    draw(conjunto)
```

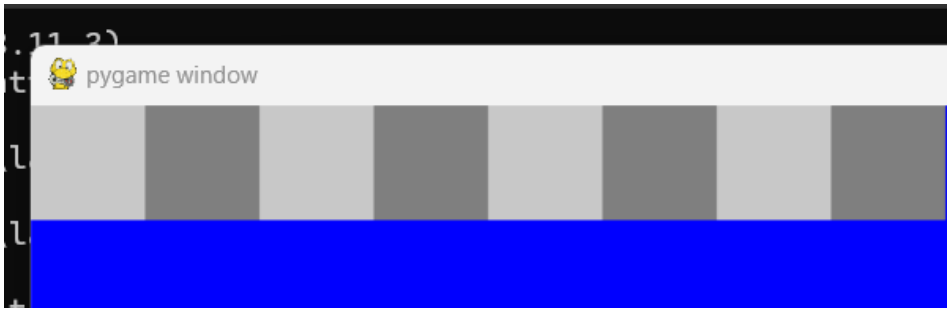
Se uso el horizontal repeat ya que si se hubiera usado join nos hubiera salido una cadena muy grande de joins



1.5 Ejercicio d

```
1  from interpreter import draw
2  from chessPictures import *
3
4  concatenado = square.join(square.negative()).horizontalRepeat(4)
5
6  draw(concatenado)
```

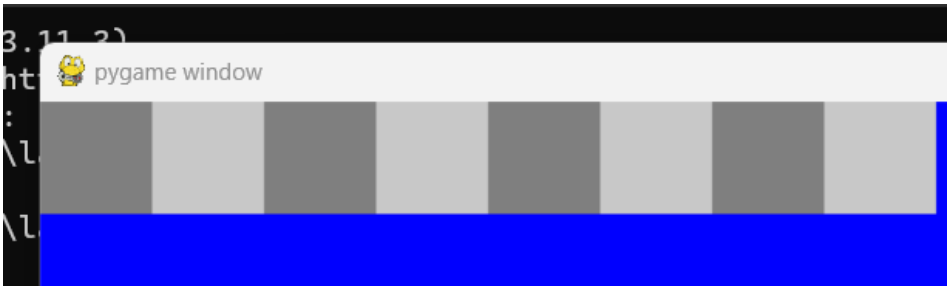
Como en el anterior se uso el horizontal repeat pero ahora con los cuadrados concatenando de 2 y empezando por el blanco.



1.6 Ejercicio e

```
1  from interpreter import draw
2  from chessPictures import *
3
4  conjunto = square.negative().join(square).horizontalRepeat(4)
5  draw(conjunto)
```

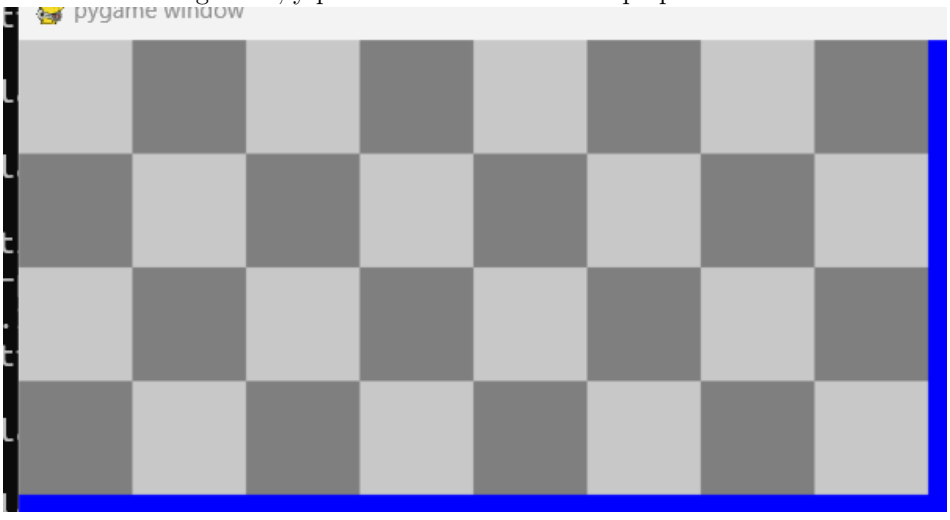
Se concatena dos cuadrados ya que estos se repiten y solo se repite 4 veces emezando ahora con el negro.



1.7 Ejercicio f

```
1   from interpreter import draw
2   from chessPictures import *
3
4   conjunto = square.up(square.negative())
5   conjunto2 = conjunto.verticalRepeat(2)
6   conjunto3 = conjunto2.join(conjunto2.negative())
7   conjunto4 = conjunto3.horizontalRepeat(4)
8   draw(conjunto4)
9
10
```

En este ejercicio primero lo generalize para luego repetirlo, primero agregue dos cuadrados uno enciama del otro luego lo duplique verticalmente, seguidamente se dublico horizontalmente pero con los cuadrados negativos, y para finalizar solo lo multiplique horizontalmente 4 veces.



1.8 Ejercicio g

```
from interpreter import draw
from chessPictures import *
#primero crear todas las variables a usar

bishopN = bishop.negative()
knightN = knight.negative()
rockN = rock.negative()
kingN = king.negative()
queenN = queen.negative()
pawnN = pawn.negative()
squareN = square.negative()

#fichas blancas en sus cuadrados
rockSquare = square.under(rock)
knightSquare = square.under(knight)
bishopSquare = square.under(bishop)
pawnSquare = square.under(pawn)
queenSquare = square.under(queen)

rockSquareN = squareN.under(rock)
knightSquareN = squareN.under(knight)
bishopSquareN = squareN.under(bishop)
pawnSquareN = squareN.under(pawn)
kingSquare = squareN.under(king)

#fichas negras en sus cuadrados
rockNsquare = square.under(rockN)
knightNsquare = square.under(knightN)
bishopNsquare = square.under(bishopN)
pawnNsquare = square.under(pawnN)
kingNSquare = square.under(kingN)

rockNsquareN = squareN.under(rockN)
knightNsquareN = squareN.under(knightN)
bishopNsquareN = squareN.under(bishopN)
pawnNsquareN = squareN.under(pawnN)
queenNSquare = squareN.under(queenN)



base1 = rockNsquare.join(knightNsquareN).join(bishopNsquare).join(queenNSquare).join(kingNSquare).join(bishopNsquareN).join(knightNsquare).join(rockNsquareN)
peones12 = (pawnNsquareN.join(pawnNsquare)).horizontalRepeat(4)
jug1=base1.up(peones12)

conjunto = square.up(square.negative())
conjunto2 = conjunto.verticalRepeat(2)
conjunto3 = conjunto2.join(conjunto2.negative())
conjunto4 = conjunto3.horizontalRepeat(4)

peones2 = (pawnSquare.join(pawnSquareN)).horizontalRepeat(4)
base21 = rockSquareN.join(knightSquare).join(bishopSquareN).join(queenSquare).join(kingSquare).join(bishopSquare).join(knightSquareN).join(rockSquare)
jug2=peones2.up(base21)
draw(jug1.up(conjunto4).up(jug2))
```

Primero inicialize las variables para reciclarlas despues. Haciendo uso del under ponemos cada ficha en su respectivo cuadrado. ya sean las blancas como las negras.

Luego vamos con las bases que que su funcion es poner las fichas para jugar, la base1 va a ser uso

de puro join para juntar los cuadrados con sus fichas, para los peones solo se repite en conjunto de dos en cuadrados blancos y negros, para el jugador 2 seria lo mismo.

Para la separacion del medio reutilize el codigo del ejercicio anterior y solo lo agregue, haciendo uso del up para juntar las filas y salgan juntas.

2 Cuestionario

¿Qué son los archivos *.pyc?
Son archivos creados creados por el sistema interprete de phyton, se crean cuendo compilas un archivo por primera vez y se importan los modulos, guarda lo neceseraio, para que posteriormente cuando queramos abrir el mismo archivo este lo abra mas rapido sin necesidad de crear otro archivo

¿Para qué sirve el directorio pycache?
Es donde se guardan los pyc que vienen hacer lo que se importa de los modulos puestos en el archivo.

¿Cuáles son los usos y lo que representa el subguión en Python?
El subguion se usa como cualquier otra variables, pero en terminos de comprension suele ser usada como una variable irrelevante o que no se va a usar en el momento que fue creado, como una variable temporal y una variable de descarte

3 Conclusiones

El lenguaje python es comodo, pero un poco confuso en cuanto a su sencilles, las variables de entorno son de mucha utilidad ya que para este laboratorio se importaron muchos modulos los cuales se importaron y servian para imprimir en pantalla los valores de la imagen, como toda funciona con objetos se trabajo con una clase la clase Pictures y en base a esa solo se llamaban los metodos, algo que me parecio interesante es el uso del draw() algo que no habia usado, claro que el ajedres no es funcional pero es como una imagen construida por filas como una imprenta.