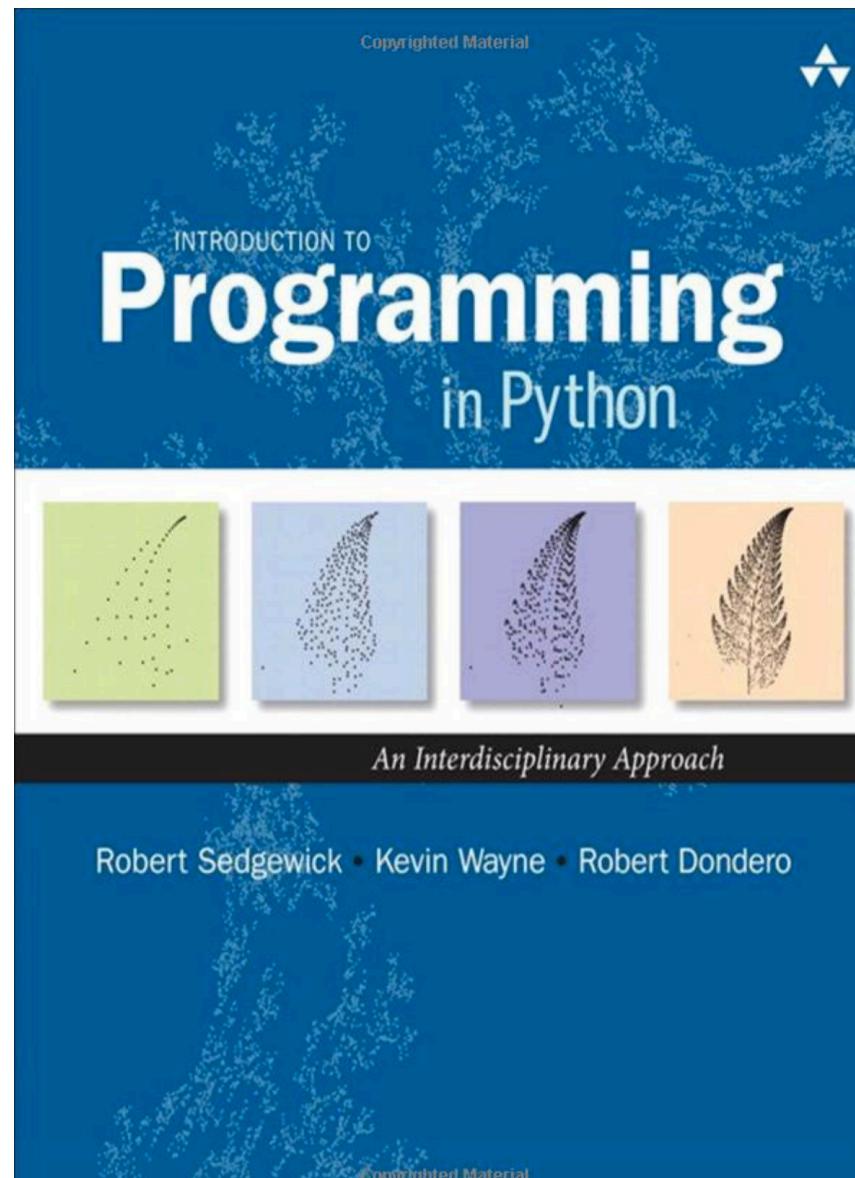


Parte II: Computación científica

Clase 14: Usando tipos de datos

Diego Caro
dcaro@udd.cl



Basada en presentaciones oficiales de libro *Introduction to Programming in Python* (Sedgewick, Wayne, Dondero).

Disponible en <https://introcs.cs.princeton.edu/python>

Outline

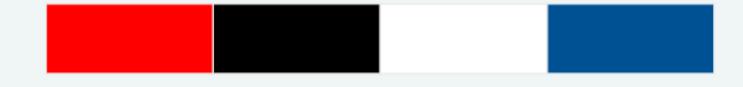
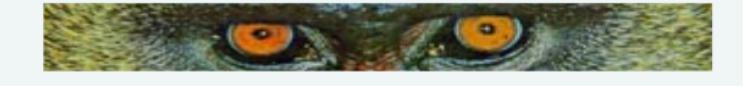
- ¿Por qué usar clases?
 - Tipo abstracto de datos: cliente no debe conocer implementación
- ¿Qué es un cliente?

¿Por qué usar orientación a objetos?

Nuestro tipo de dato
(clase + atributos +
métodos)

- Programación Orientada a Objetos (OOP)
 - Crear nuestro propio tipo de datos
 - Usarlos en nuestros programas (manipular los objetos)
- Buenas prácticas:
 - Usar un tipo de dato abstracto (**API**)
 - **Impacto:** Clientes usan tipo de dato sin tener que conocer los detalles de la implementación!

<i>sample call</i>	<i>method</i>	<i>function</i>
<i>typically invoked with</i>	x.bit_length()	stdio.writeln(bits)
<i>parameters</i>	variable name	module name
<i>primary purpose</i>	object reference and argument(s)	argument(s)
	manipulate object value	compute return value
	<i>Methods versus functions</i>	

<i>data type</i>	<i>set of values</i>	<i>examples of operations</i>	
Color	three 8-bit integers	get red component, brighten	
Picture	2D array of colors	get/set color of pixel (i, j)	
String	sequence of characters	length, substring, compare	C A T A G C G C

¿Por qué usar orientación a objetos?

Implementación funciones+módulos

```
1 def creatematrix(n, m):
2     """Return a matrix of n rows and m columns."""
3     return [ [0]*m ]*n
4
5 def strmatrix(m):
6     s = '\n'
7     for i in range(len(m)):
8         s += '['
9         for j in range(len(m[i])):
10            s += str(m[i][j]) + ' '
11        s += ']\n'
12    return s
13
14 m = creatematrix(4, 5)
15 print('type(m):', type(m))
16 print('m:', strmatrix(m))
17 m[1] = [99]
18 print('m:', strmatrix(m))
```

```
$ python3 matriz.py
type(m): <class 'list'>
m:
[ 0 0 0 0 0 ]
[ 0 0 0 0 0 ]
[ 0 0 0 0 0 ]
[ 0 0 0 0 0 ]
m:
[ 0 0 0 0 0 ]
[ 99 ]
[ 0 0 0 0 0 ]
[ 0 0 0 0 0 ]
```

Nuestro tipo de dato
(clase + atributos +
métodos)

¿Por qué usar orientación a objetos?

Nuestro tipo de dato
(clase + atributos +
métodos)

Implementación Orientación a Objetos

```
1 class Matrix:  
2     """Create a matrix of n rows and m columns."""  
3     def __init__(self, n, m):  
4         self.cols = m  
5         self.rows = n  
6         self.m = [ [0]*m ]*n  
7  
8     def __str__(self):  
9         s = '\n'  
10        for i in range(len(self.m)):  
11            s += '['  
12            for j in range(len(self.m[i])):  
13                s += str(self.m[i][j]) + ','  
14            s += ']\n'  
15        return s  
16  
17 m = Matrix(4, 5)  
18 print('type(m):', type(m))  
19 print('m:', m)  
20 m[1] = [99]  
21 print('m:', m)
```

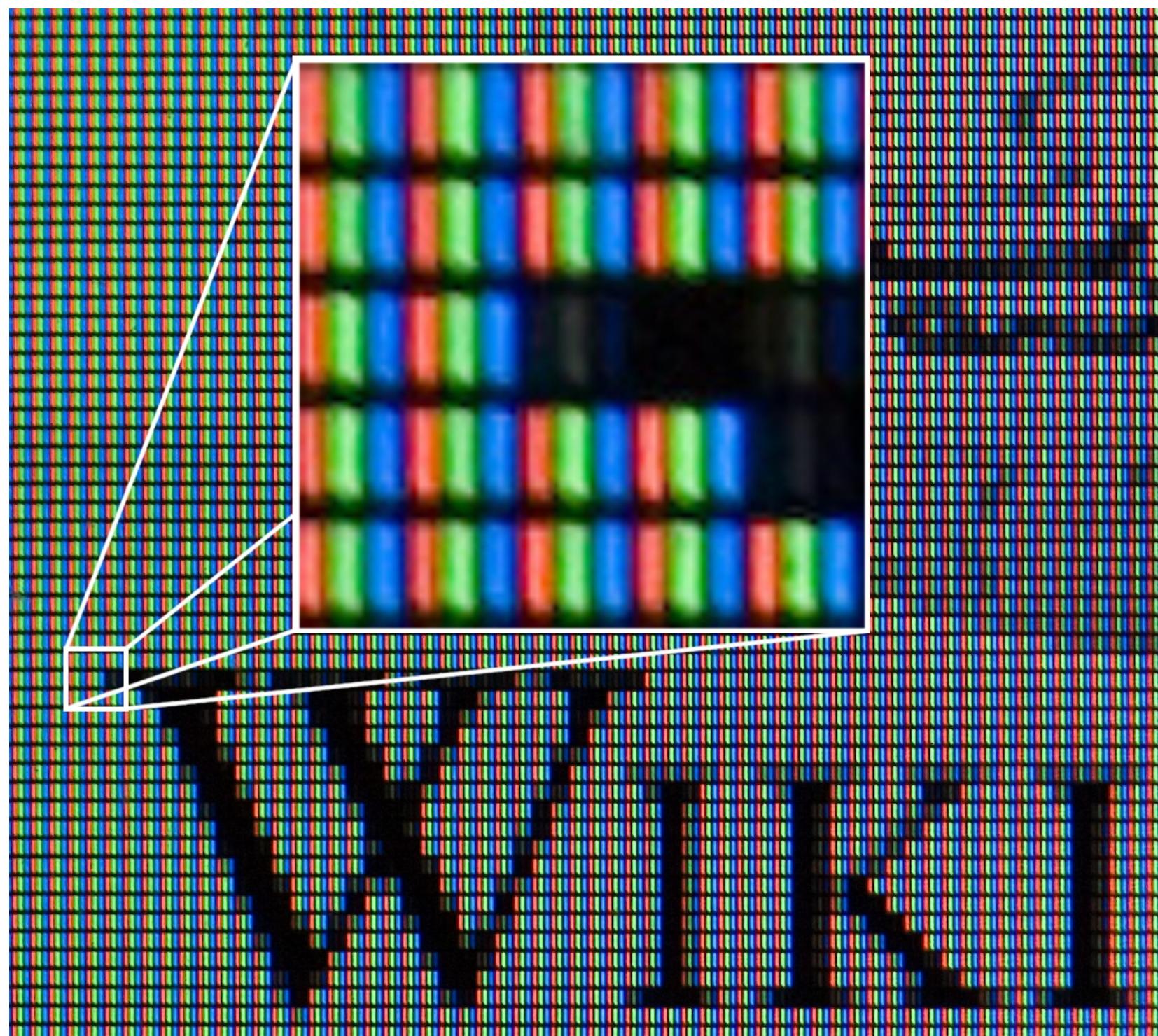
```
$ python3 matrizclass.py  
type(m): <class '__main__.Matrix'>  
m:  
[ 0 0 0 0 0 ]  
[ 0 0 0 0 0 ]  
[ 0 0 0 0 0 ]  
[ 0 0 0 0 0 ]  
  
Traceback (most recent call last):  
  File "matrizclass.py", line 20, in <module>  
    m[1] = [99]  
TypeError: 'Matrix' object does not support item assignment
```

Preguntas

1. ¿Qué es un tipo de dato?
 - Un conjunto de valores y operaciones sobre esos valores. Una class.
2. ¿Qué es un tipo de datos abstracto?
 - Un tipo de dato cuya representación está oculta al cliente. La/el usuaria programadora no tiene que entender como funciona, simplemente usa la API (y confía que hace lo que debe hacer).

Ejemplo: API Color

- **Color:** sensación en el ojo producto de radiación electromagnética.



Valores

examples										
R (8 bits)	red intensity	255	0	0	0	255	0	119	105	
G (8 bits)	green intensity	0	255	0	0	255	64	33	105	
B (8 bits)	blue intensity	0	0	255	0	255	128	27	105	
color										

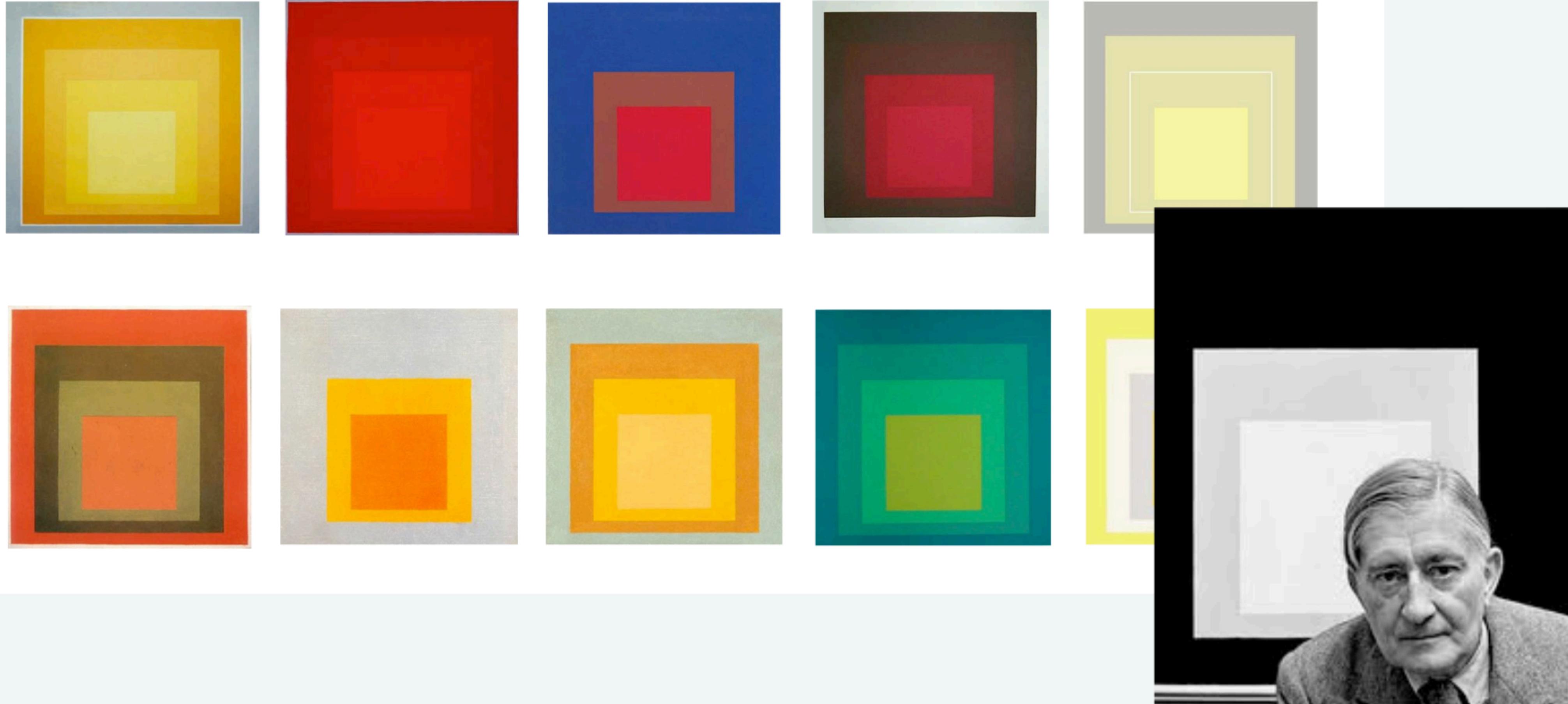
class Color

	Color(r, g, b)	
int	red()	Retorna intensidad roja.
int	green()	Retorna intensidad verde.
int	blue()	Retorna intensidad azul.
Color	brighter()	Retorna una versión brillante.
Color	darker()	Retorna una versión opaca.
bool	equal(c)	Este color es el mismo que c?
str	__str__()	Representación en string del color.

Operaciones (API)

Albers squares

Josef Albers. A 20th century artist who revolutionized the way people think about color.



Josef Albers 1888–1976

Ejemplo: cuadrados de Albert

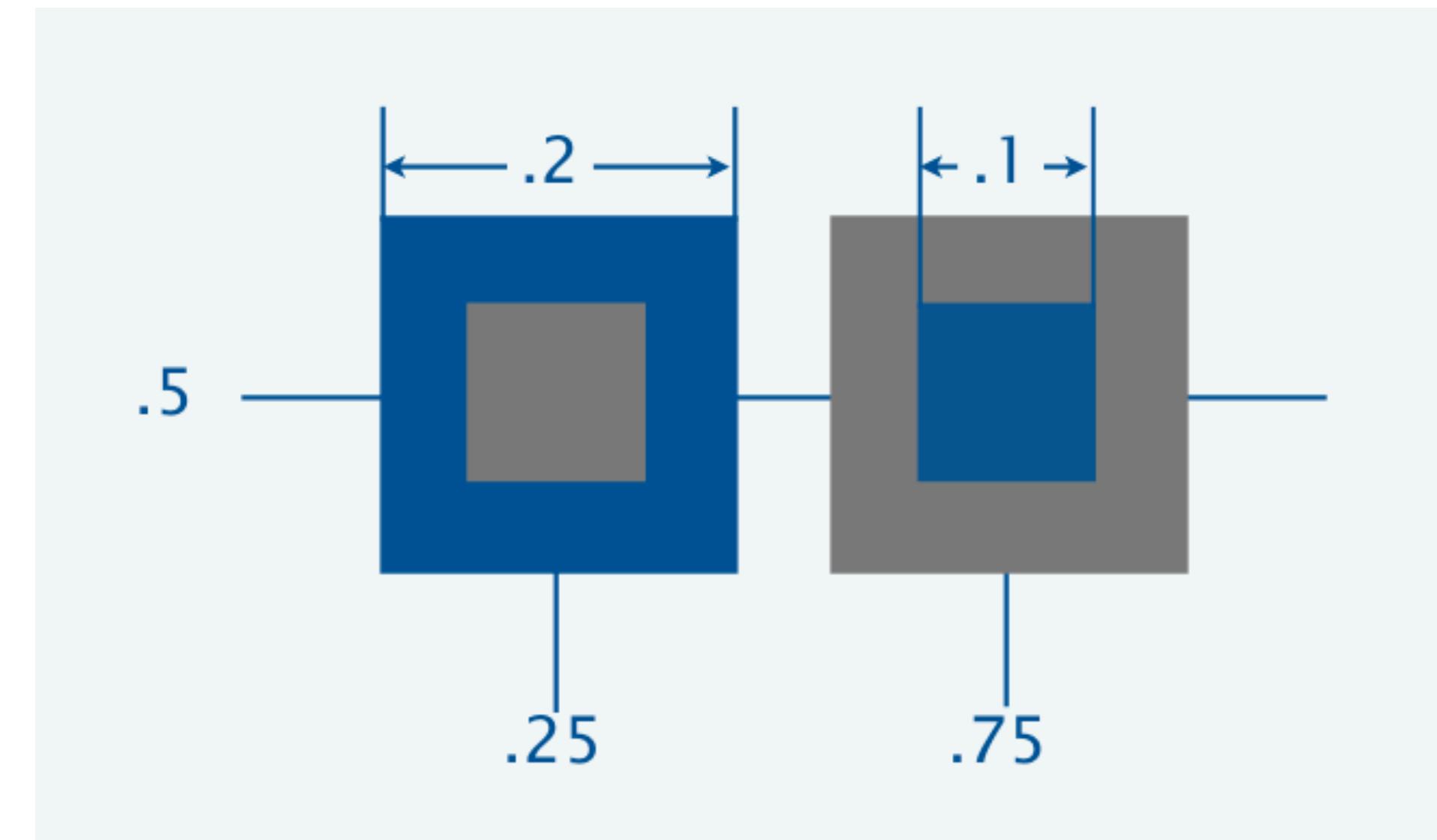
```
1 import stddraw → ¡Módulo para dibujar!
2 from color import Color
3
4 def readint(): return int(input())
5
6 r1 = readint()
7 g1 = readint()
8 b1 = readint()
9 c1 = Color(r1, g1, b1)
10
11 r2 = readint()
12 g2 = readint()
13 b2 = readint()
14 c2 = Color(r2, g2, b2)
15
16 stddraw.setCanvasSize(512, 256)
17 stddraw.setScale(.25, .75)
18
19 stddraw.setPenColor(c1)
20 stddraw.filledSquare(.25, .5, .2)
21
22 stddraw.setPenColor(c2)
23 stddraw.filledSquare(.25, .5, .1)
24
25 stddraw.setPenColor(c2)
26 stddraw.filledSquare(.75, .5, .2)
27
28 stddraw.setPenColor(c1)
29 stddraw.filledSquare(.75, .5, .1)
30
31 stddraw.show()
```

Crea primer color

Crea segundo color

Primer cuadrado

Segundo cuadrado





DEMO TIME

Computación con color: luminiscencia

- Def: la luminiscencia monocromática de un color cuantifica su **brillo efectivo**.
- Fórmula para luminiscencia de NTSC:
 - $0.299r + 0.587g + 0.114b$

```
1 from mycolor import Color
2 def readint(): return int(input())
3
4 def luminance(c):
5     red    = c.red()
6     green  = c.green()
7     blue   = c.blue()
8     return (.299 * red) + (.587 * green) + (.114 * blue)
9
10 if __name__ == '__main__':
11     r = readint()
12     g = readint()
13     b = readint()
14     c = Color(r, g, b)
15     print(round(luminance(c)))
```

Esta función usa tipo de dato Color

	examples								
red intensity	255	0	0	0	255	0	119	105	
green intensity	0	255	0	0	255	64	33	105	
blue intensity	0	0	255	0	255	128	27	105	
color									
luminance	76	150	29	0	255	52	58	105	

Aplicaciones:

- Convertir a escala de grises
- Escoger colores sobre fondo



Aplicación: Compatibilidad de color

- **Pregunta:** ¿Cuál color escojo para mi tipografía dado un color de fondo?
- Buena práctica: que la diferencia de luminiscencia sea >128

```
def compatible(a, b):  
    return abs(lum(a) - lum(b)) > 128
```

¿Qué tipo de dato retorna la función?



	76	0	255	52
76	255	76	179	24
0	76		255	52
255	179	255		203
52	24	52	203	

Aplicación: escala de grises

- Cuando los valores de rojo, verde y azul son iguales, el color resultante es una escala de grises, desde 0 (negro) a 255 (blanco).
- Pregunta: ¿Qué valor asigno para un color? (alguna combinación de rgb)
 - Luminiscencia!

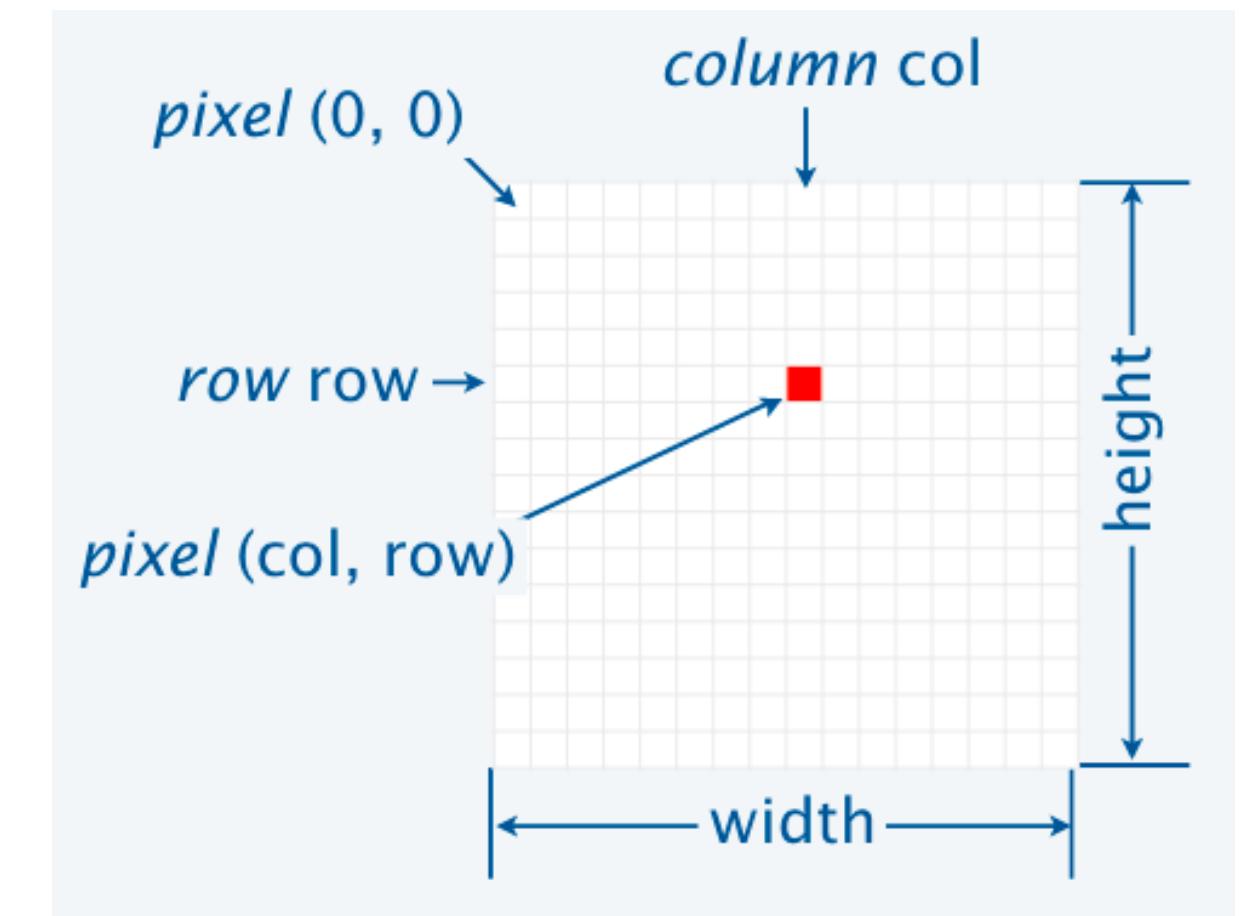
```
def grayscale(c):  
    y = round(lum(c))  
    return Color(y, y, y)
```



	<i>examples</i>								
red intensity	255	0	0	0	255	0	119	105	
green intensity	0	255	0	0	255	64	33	105	
blue intensity	0	0	255	0	255	128	27	105	
color									
luminance	76	150	29	0	255	52	58	105	
grayscale									

API para imágenes

- Una imagen es un arreglo 2D de pixeles.
 - **Valores:** color de cada pixel.



class Picture

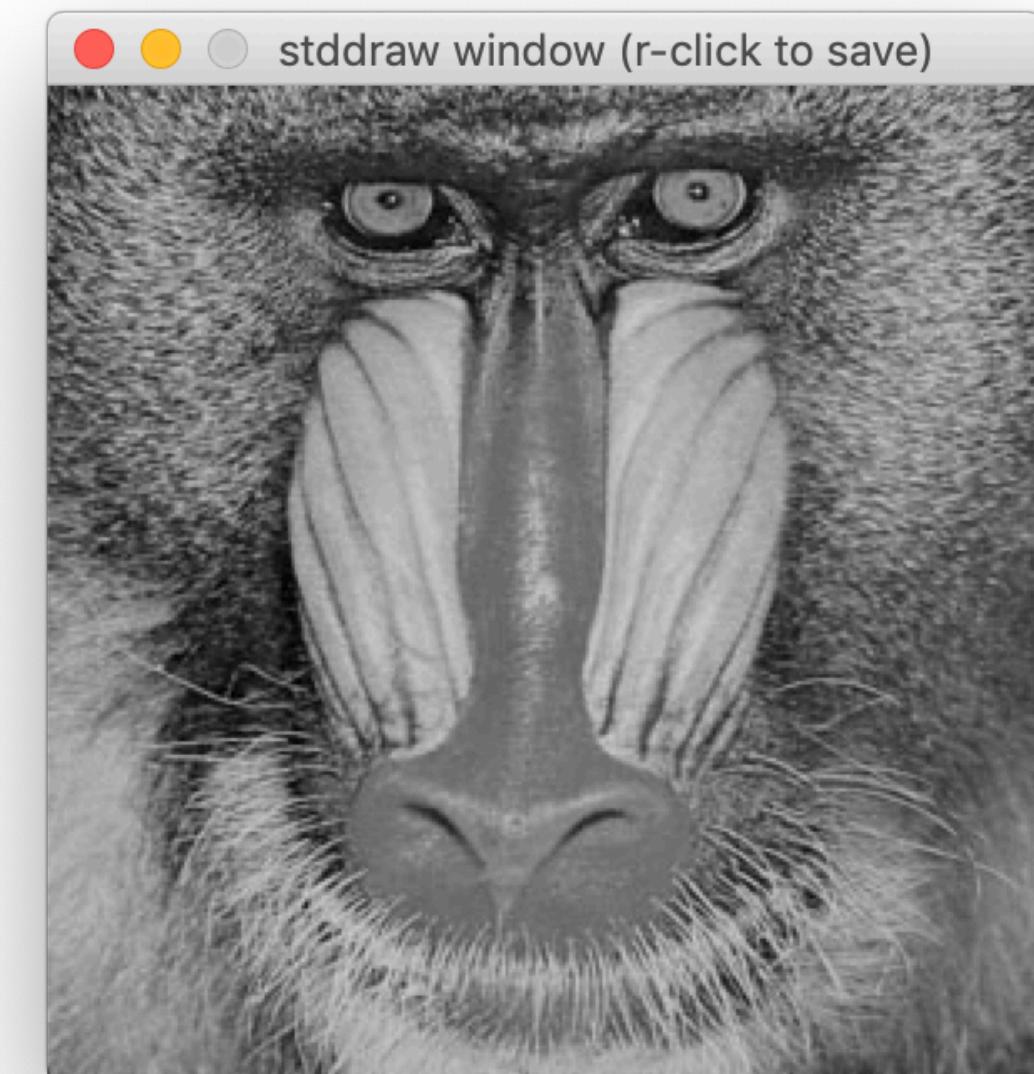
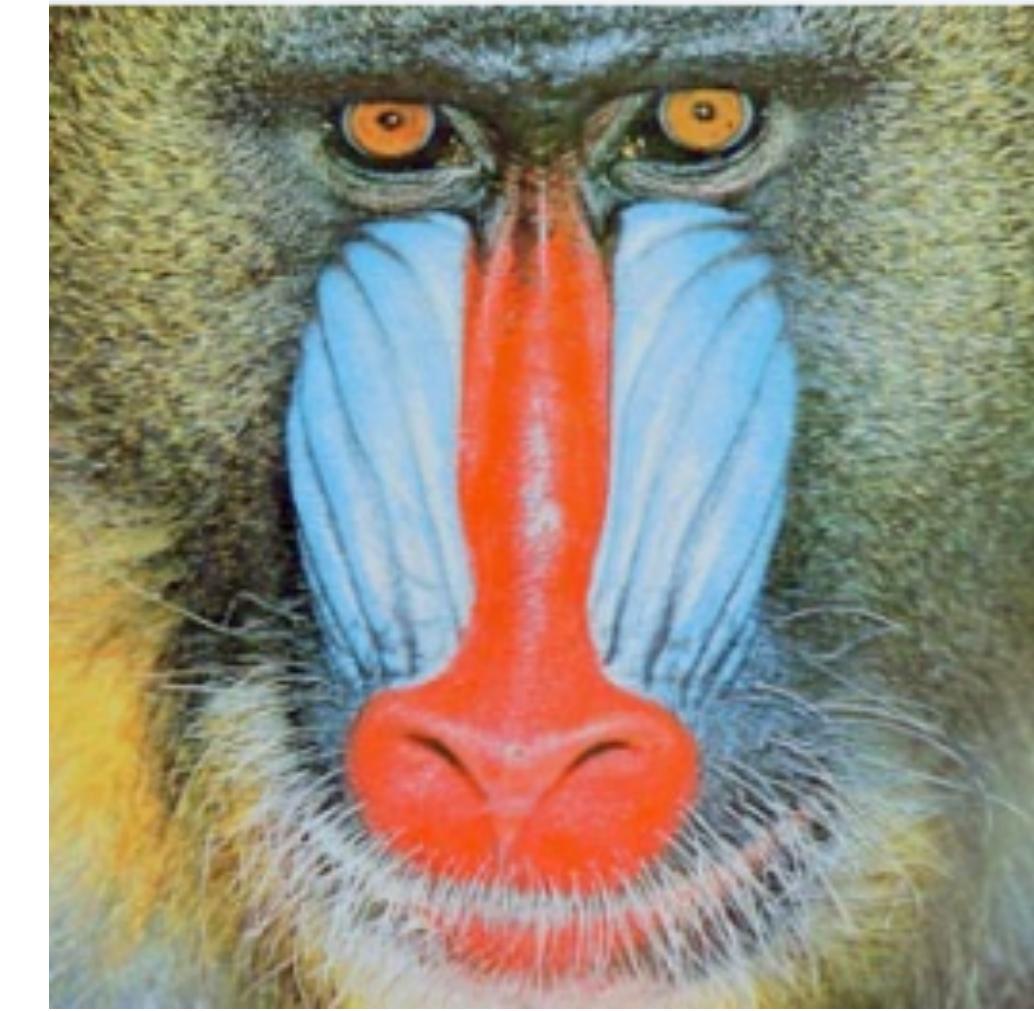
	Picture(filename)	Crea una imagen desde archivo
	Picture(w, h)	Crea una imagen vacía de w por h
int	width()	Ancho de la imagen
int	height()	Altura de imagen
Color	get(col, row)	Color del pixel (col, row)
	set(col, row, c)	Asigna color c a pixel en (col, row)
	show()	Muestra la imagen en una ventana
	save(filename)	Guarda imagen en archivo

Aplicación: convertir a blanco y negro

```
1 import sys  
2 import stddraw  
3 import luminance  
4 from picture import Picture  
5  
6 pic = Picture(sys.argv[1])          Crea imagen desde archivo  
7  
8 for col in range(pic.width()):  
9     for row in range(pic.height()):  
10        pixel = pic.get(col, row)  
11        gray = luminance.toGray(pixel)  
12        pic.set(col, row, gray)  
13  
14 stddraw.setCanvasSize(pic.width(), pic.height())  
15 stddraw.picture(pic)  
16 stddraw.show()
```

Transforma color en
blanco y negro

Crea ventana
y muestra imagen

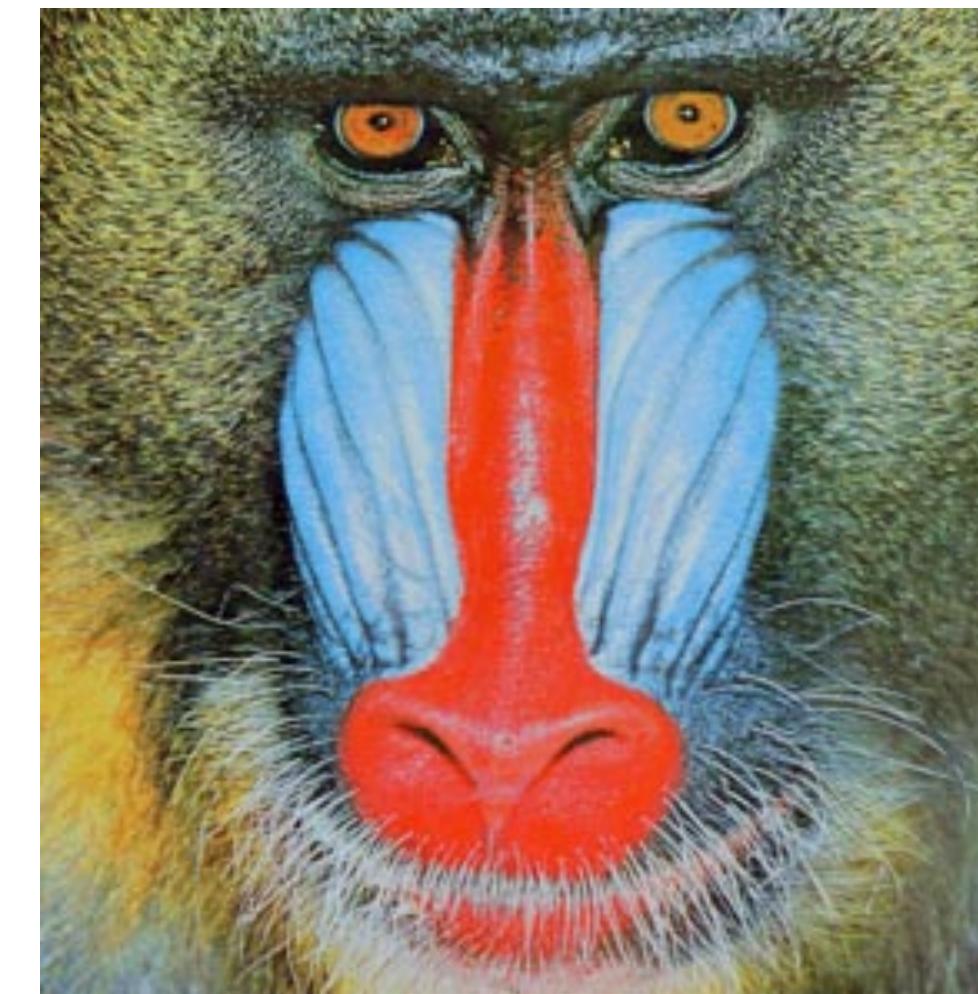
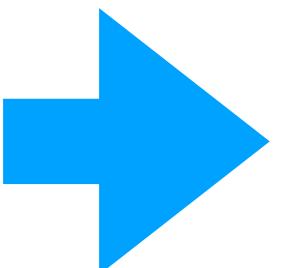


\$ python3 grayscale.py mandrill.jpg

Human-based python interpreter™

- ¿Qué hace el siguiente código?

```
for col in range(pic.width()):  
    for row in range(pic.height()):  
        pic.set(col, row, pic.get(col, row))  
  
stddraw.show()
```

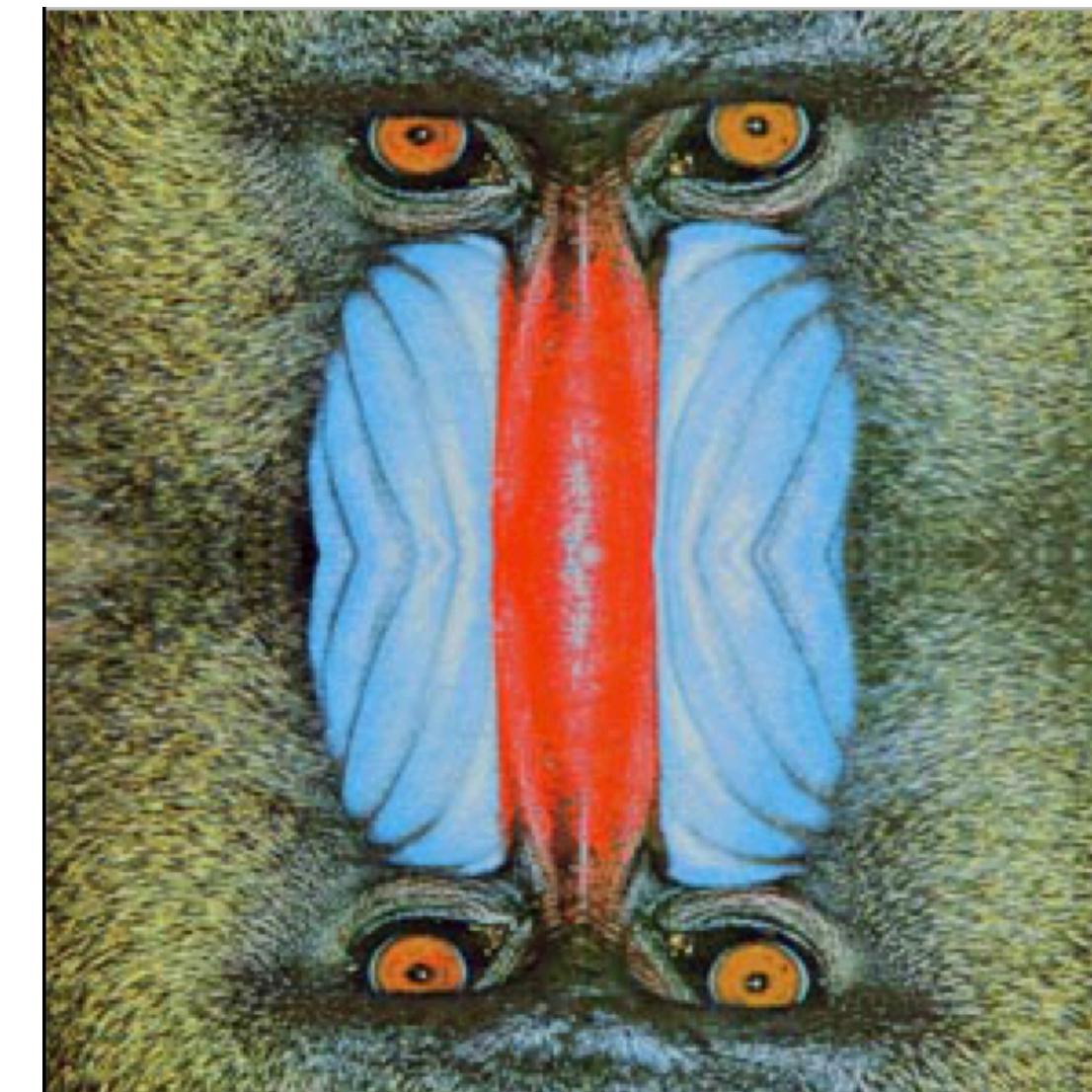
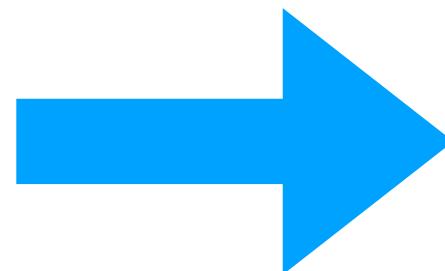
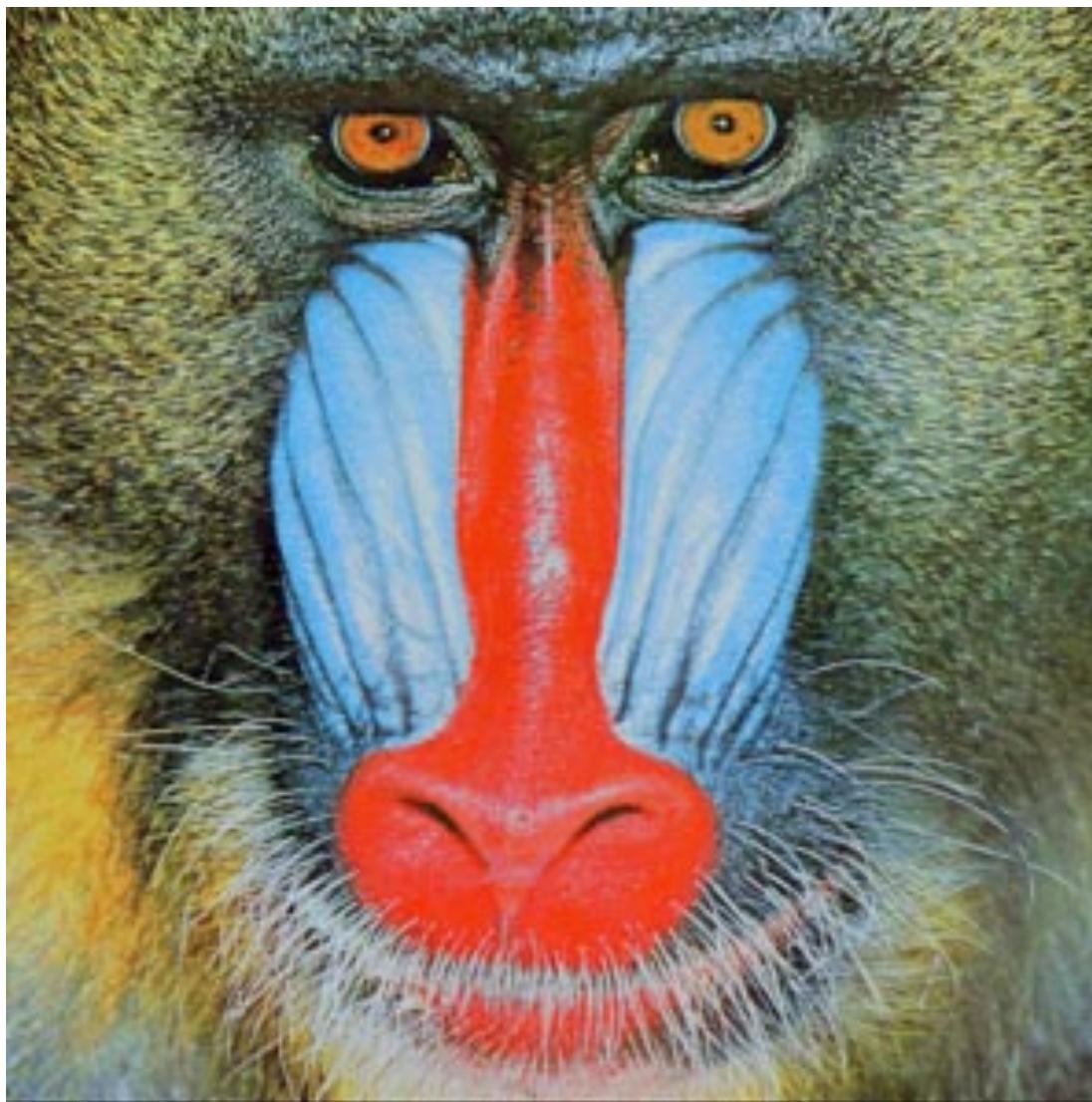
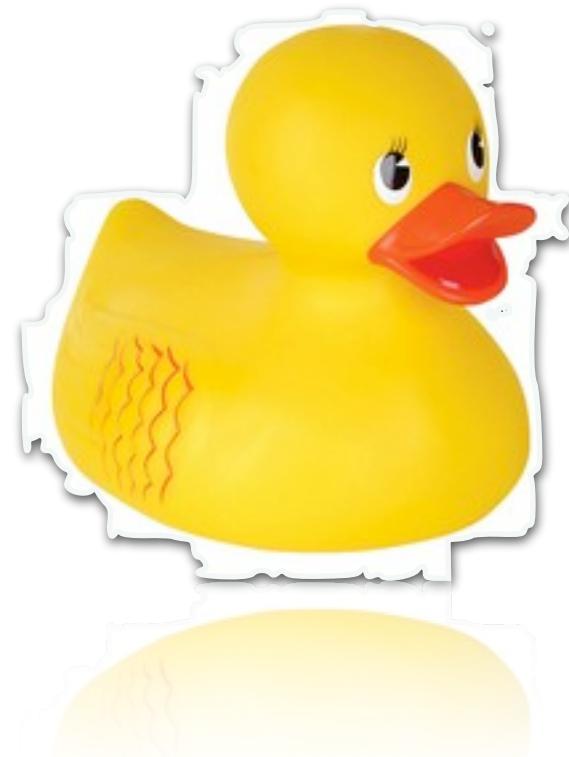


Human-based python interpreter™

- ¿Qué hace el siguiente código? (no tan fácil como el anterior)

```
for col in range(pic.width()):  
    for row in range(pic.height()):  
        pic.set(col, pic.height()-row-1, pic.get(col, row))  
  
stddraw.show()
```

Intenta reflejar verticalmente la imagen...
Pero falla.

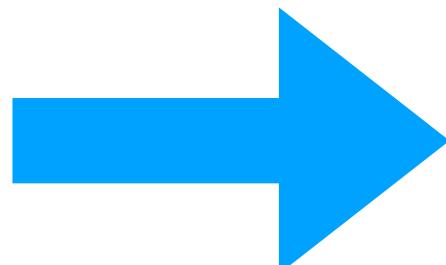


Human-based python interpreter™

- **Solución:** completar una imagen vacía.

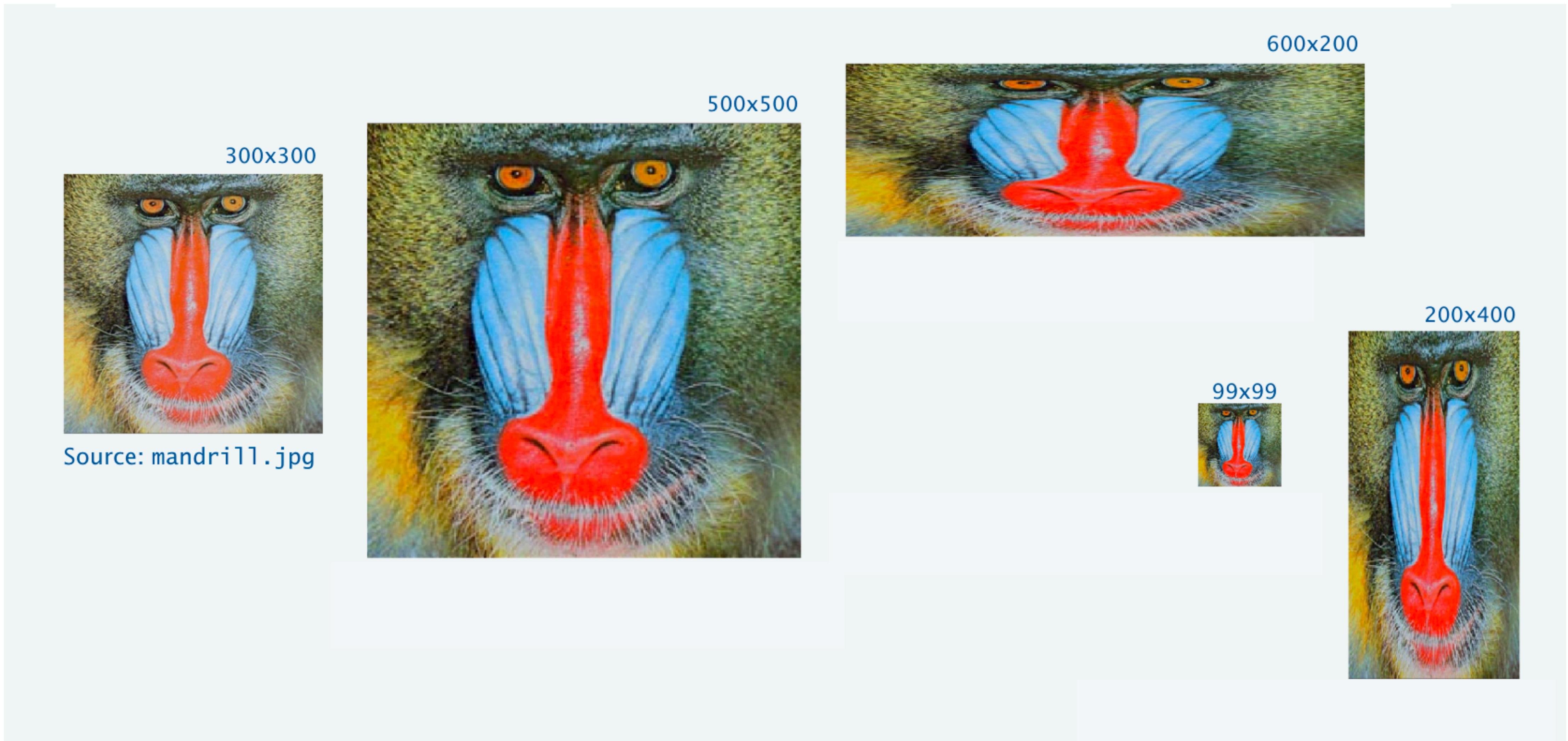
```
source = Picture(sys.argv[1])
target = Picture(source.width(), source.height())
for col in range(source.width()):
    for row in range(source.height()):
        target.set(col, source.height()-row-1, source.get(col, row))

stddraw.setCanvasSize(target.width(), target.height())
stddraw.picture(target)
stddraw.show()
```



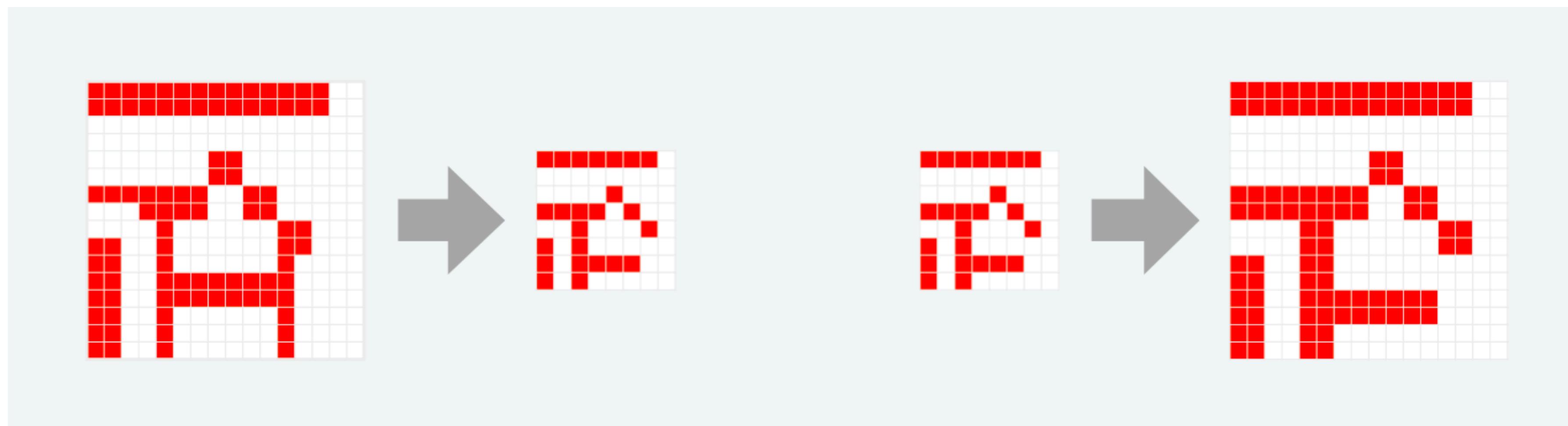
Ejemplo: escalar imagen

- Objetivo: escalar la imagen independiente del ancho y alto destino.



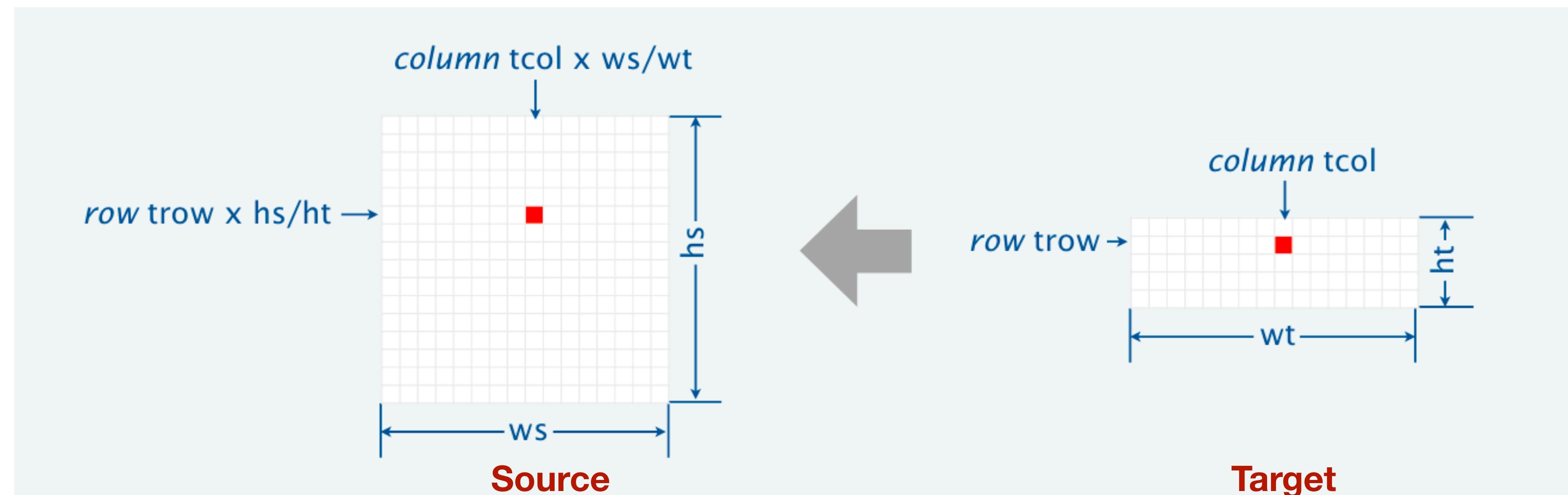
Ejemplo: escalar imagen

- Objetivo: escalar la imagen independiente del ancho y alto destino.
- Ejemplo de estrategia:
 - Achicar imagen eliminando filas y columnas alternadamente.
 - Agrandar imagen cuadruplicando cada pixel.
 - ¿Problema?



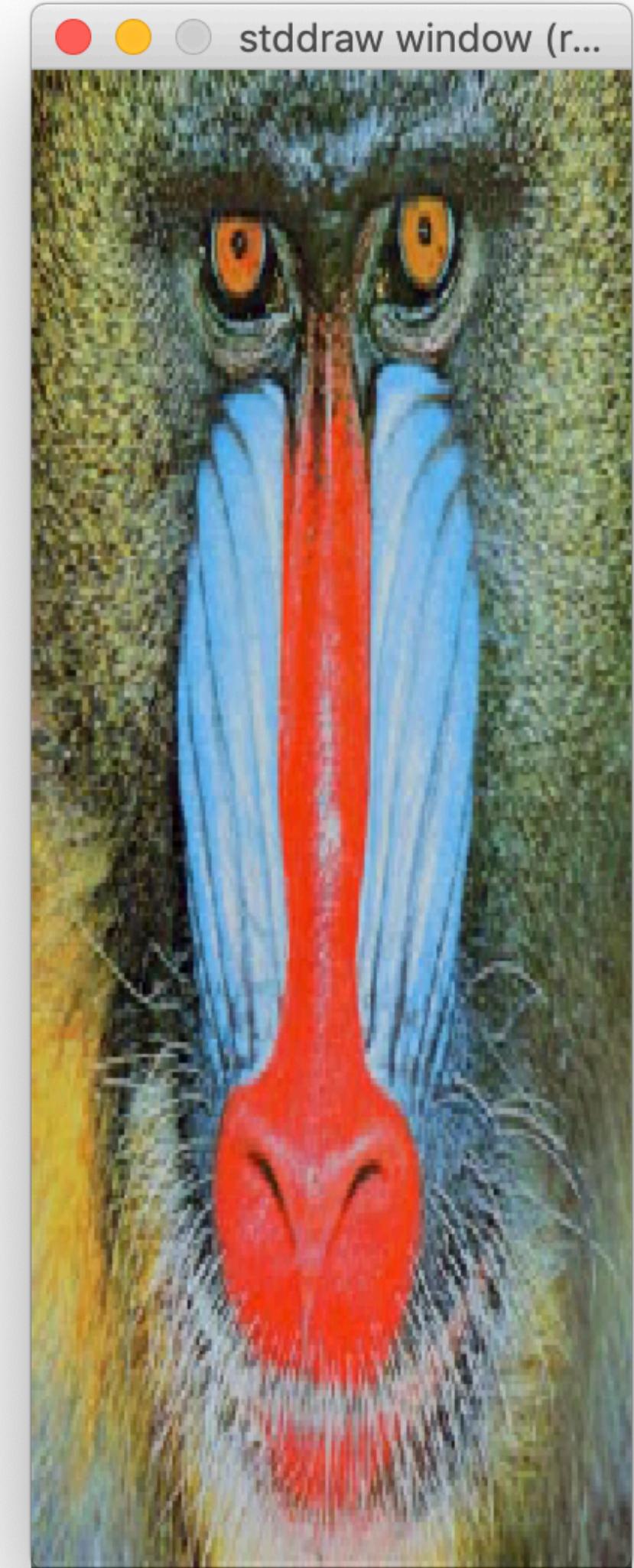
Ejemplo: escalar imagen

- Objetivo: escalar la imagen independiente del ancho y alto destino.
- **Estrategia** uniforme:
 - escalar columnas ubicando el índice en posición $ws//wt$
 - escalar filas ubicando el índice en posición $hs//ht$
- **Enfoque:** computar cada pixel en la imagen destino (del tamaño a escalar).

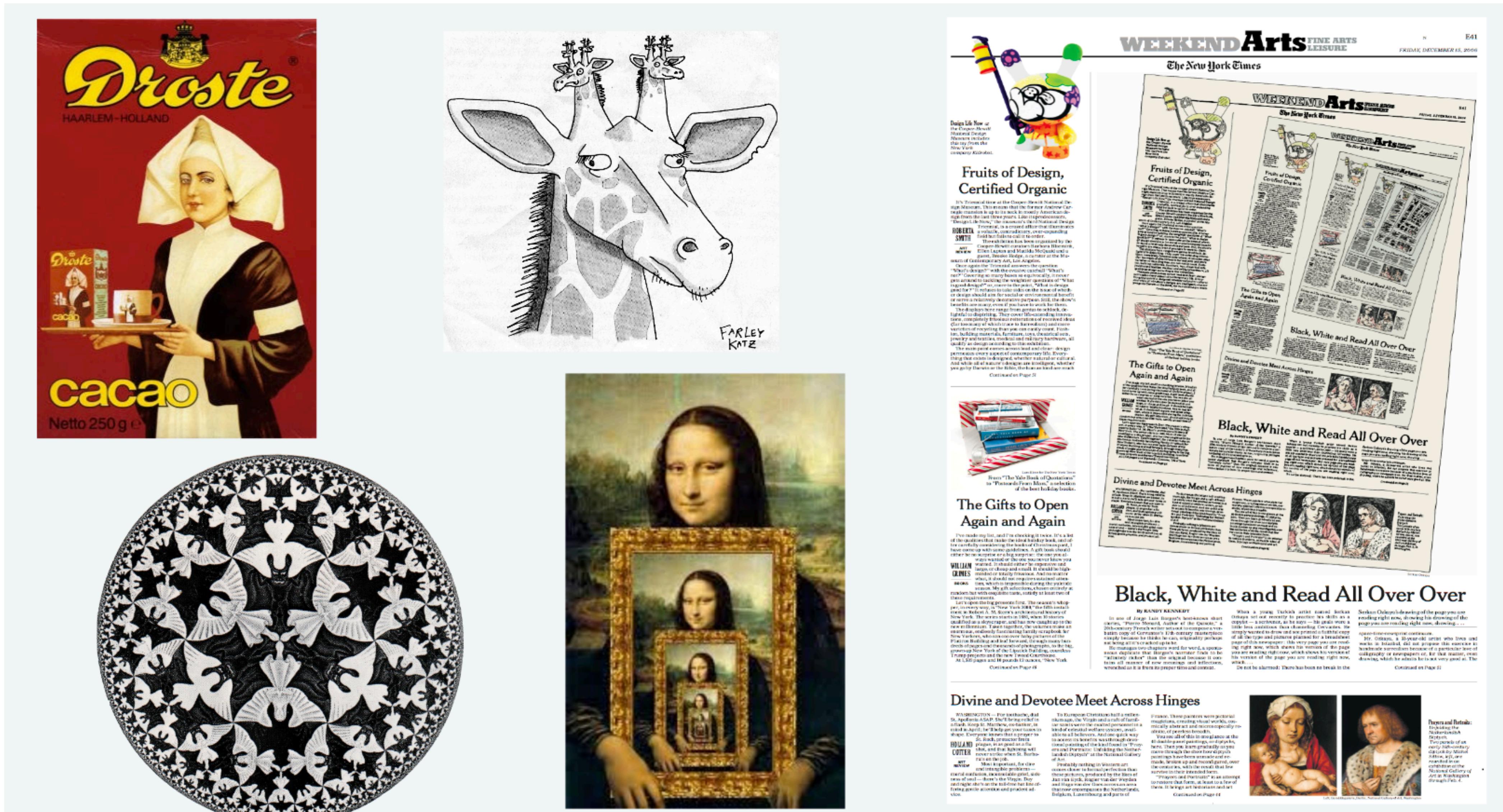


Ejemplo: escalar imagen

```
1 import stddraw
2 import sys
3 from picture import Picture
4
5 fileName = sys.argv[1]
6 w = int(sys.argv[2])
7 h = int(sys.argv[3])
8
9 source = Picture(fileName)
10 target = Picture(w, h)
11
12 for tCol in range(w):
13     for tRow in range(h):
14         sCol = tCol * source.width() // w | Posición en imagen origen
15         sRow = tRow * source.height() // h | Posición en imagen origen
16         target.set(tCol, tRow, source.get(sCol, sRow))
17
18 stddraw.setCanvasSize(w, h)
19 stddraw.picture(target)
20 stddraw.show()
```

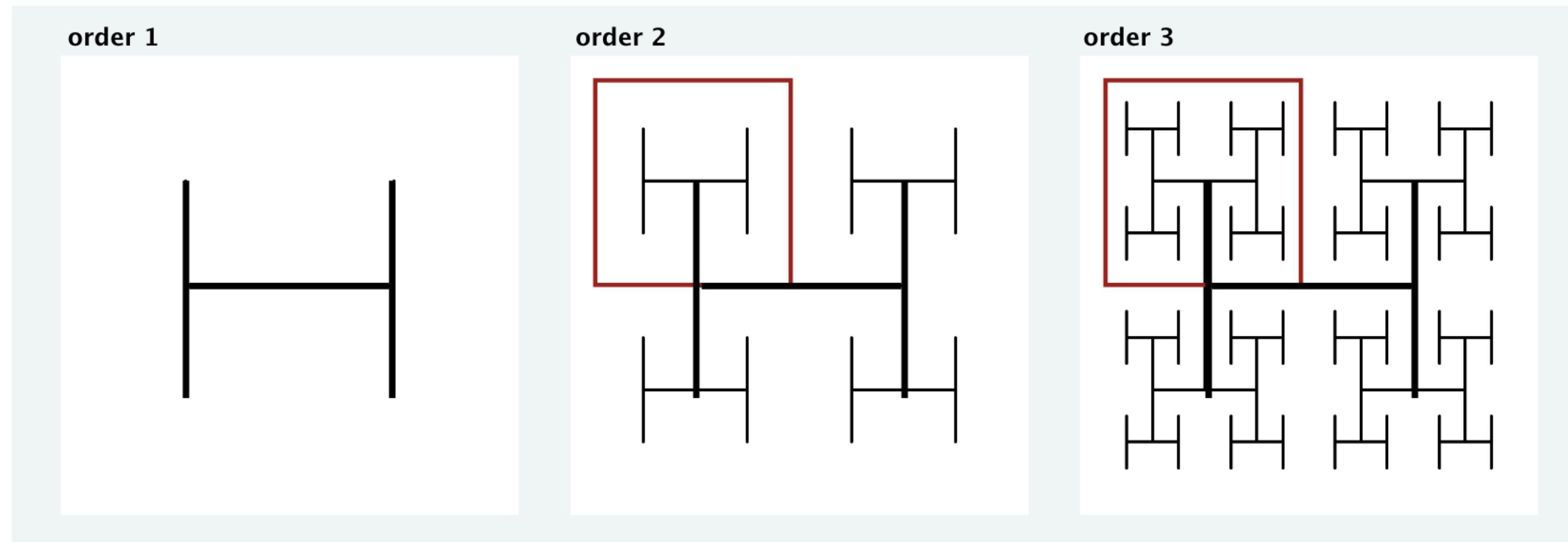


Recursividad, imágenes y arte



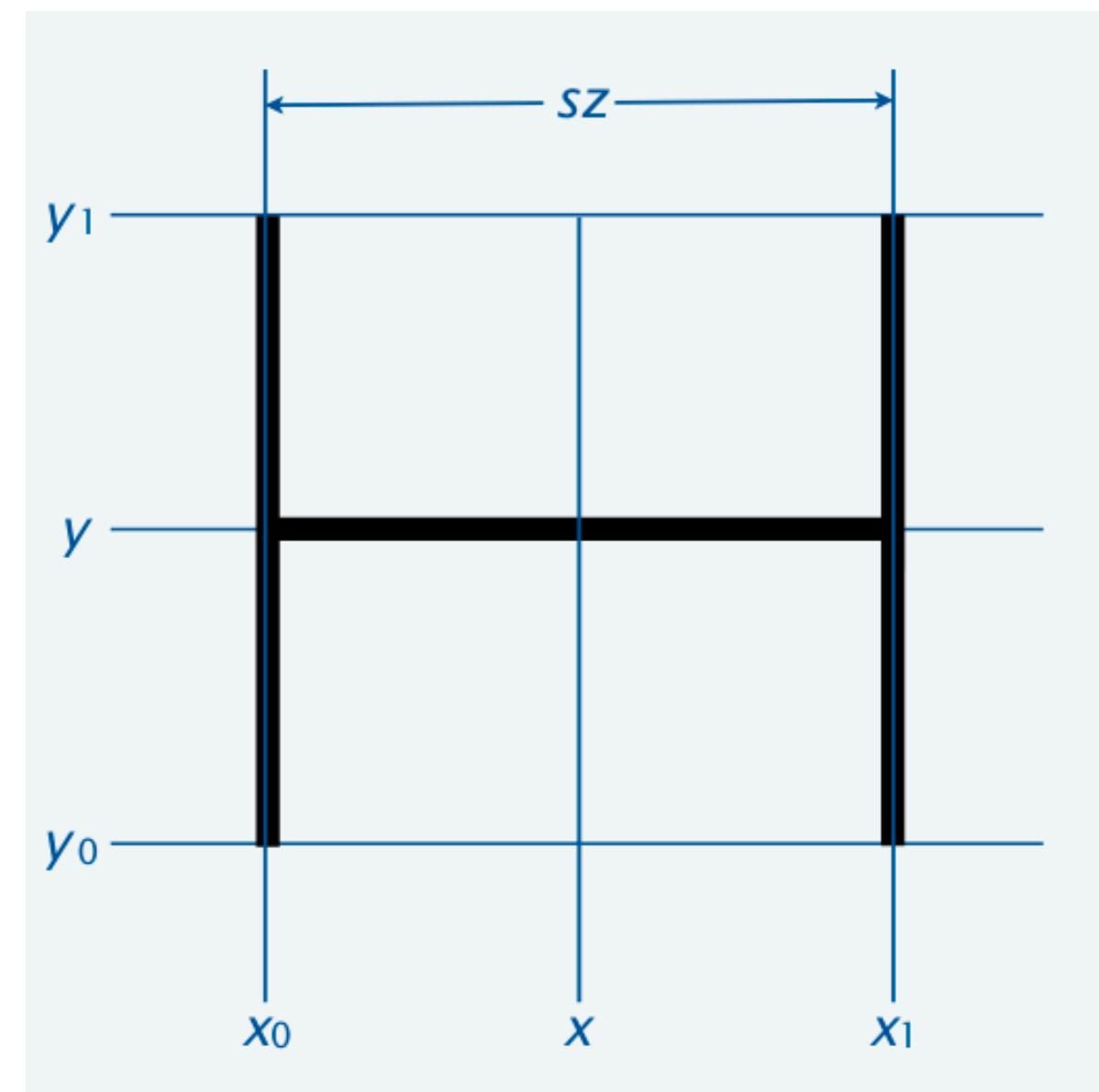
H-Trees

- Un H-Tree de orden n:
 - Si n es 0, hacer nada.
 - Dibujar una H en el centro
 - Dibujar cuatro H-trees de orden n-1, de la mitad del tamaño y centradas en los bordes de la H



Implementación H-Tree

```
1 import stddraw
2 import sys
3
4 def draw(n, lineLength, x, y):
5     if n == 0:
6         return
7     x0 = x - lineLength/2
8     x1 = x + lineLength/2
9     y0 = y - lineLength/2
10    y1 = y + lineLength/2
11
12    stddraw.line(x0, y, x1, y)
13    stddraw.line(x0, y0, x0, y1)
14    stddraw.line(x1, y0, x1, y1)
15
16    draw(n-1, lineLength/2, x0, y0)
17    draw(n-1, lineLength/2, x0, y1)
18    draw(n-1, lineLength/2, x1, y0)
19    draw(n-1, lineLength/2, x1, y1)
20
21 def main():
22     n = int(sys.argv[1])
23     stddraw.setPenRadius(0.0)
24     draw(n, .5, .5, .5)
25     stddraw.show()
26
27 if __name__ == '__main__':
28     main()
```



Credits

- This presentation is heavily based on class number 8 of the COS 126 course of the Princeton University.
- Thanks Prof. Robert Sedgewick and the whole team for making it available for the world.



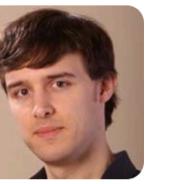
✉ Robert
Sedgewick
Faculty
Instructor



✉ Alan
Kaplan
Faculty
Co-Lead
Preceptor



✉ Dan
Leyzberg
Faculty
Co-Lead
Preceptor



✉ Jérémie
Lumbruso
Faculty
Co-Lead Preceptor



✉ Jenny Louthan
Graduate Student
Preceptor



✉ Rachel
Protacio
Graduate
Student
Preceptor



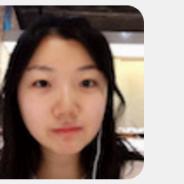
✉ Anastasiya
Kravchuk-Kirilyuk
Graduate Student
Preceptor



✉ Linda Cai
Graduate
Student
Preceptor



✉ Xin Sun
Graduate
Student
Preceptor



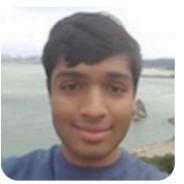
✉ Zhongqiao
'Olivia' Gao
Graduate Student
Preceptor



✉ Naorin Hossain
Graduate Student
Preceptor



✉ Priscilla Lee
Graduate
Student
Preceptor



✉ Suriya
Kodeswaran
Graduate Student
Preceptor



✉ Jennifer
Lam
Graduate
Student
Preceptor



✉ Melanie
Weber
Graduate
Student
Preceptor



✉ Mihir Kulkarni
Graduate Student
ISC-Only
Preceptor



✉ Xi Chen
Graduate
Student
ISC-Only
Preceptor

<http://www.cs.princeton.edu/courses/archive/fall18/cos126/>