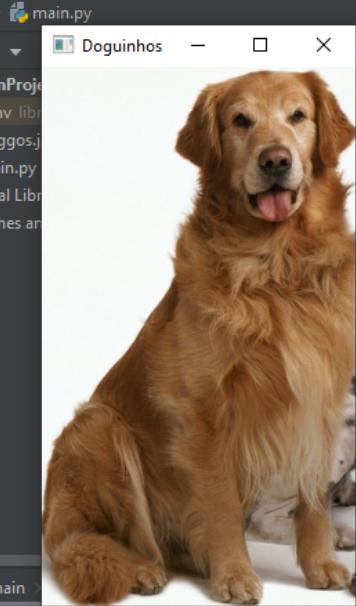

Roberta Yumi Romero Takahashi

1903220

Escolher uma imagem com vários cães, mais que 5.



1 - Localizar cada cão e gerar imagens menores, individuais de cada cão e apresentar cada uma delas.



```
main.py
Doguininhos  -  X  n.py x

import cv2

original_image = cv2.imread('doggos.jpg')

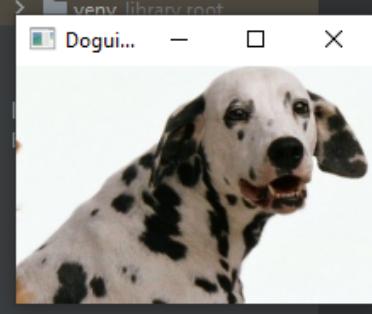
proportion = 1000.0 / original_image.shape[1]
new_size = (1000, int(original_image.shape[0] * proportion))
im = cv2.resize(original_image, new_size, interpolation=cv2.INTER_AREA)

azul = (255, 0, 0)

recorte = im[140:530, 93:320]
cv2.imshow("Doguininhos", recorte)
cv2.waitKey(0)

# cv2.rectangle(im, (90, 140), (320, 530), azul, 5)

C:\Users\Yumi Takahashi\PycharmProjects\pythonProject\venv\Scripts\python.exe" "C:\Users\Yumi Takahashi\Pycha
```



```
main.py x

# -----
# cv2.rectangle(im, (530,140), (320, 280), azul, 5)

# im[140:530, 320:550] = (0, 0, 255)

recorte = im[140:280, 320:530]

# ----

cv2.imshow("Doguininhos", recorte)
cv2.waitKey(0)
```



```
Doguininhos  -  X  Doguininhos x

# -----
# cv2.rectangle(im, (310, 550), (550, 280), azul, 5)

# im[280:550, 310:550] = (0, 0, 255)

recorte = im[285:535, 315:545]

# ----

cv2.imshow("Doguininhos", recorte)
cv2.waitKey(0)
```

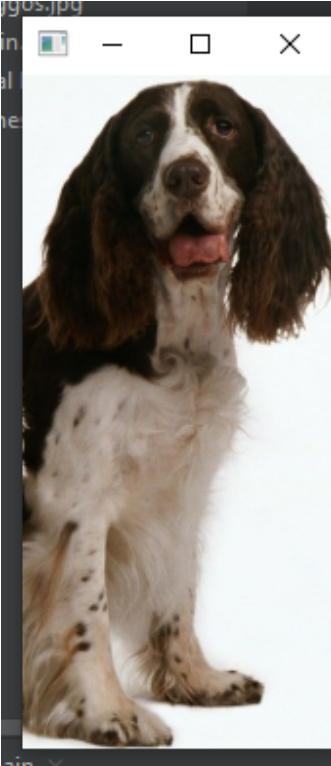


```
# (4, 2), (1, 3)
...
# -----
# cv2.rectangle(im, (530, 92), (700, 370), azul, 5)
recorte = im[92:370, 530:700]
#
cv2.imshow("Doguininhos", recorte)
cv2.waitKey(0)
```



```
# -----
# cv2.rectangle(im, (550, 280), (790, 527), azul, 5)
recorte = im[280:527, 550:790]
#
cv2.imshow("Doguininhos", recorte)
cv2.waitKey(0)

# start point (goes right and left, goes up and down)
# end point(going right and left, going up and down)
```



```
jgos.jpg
in  -  X
al
ne
im[280:527, 550:790] = (0, 0, 255)
cv2.rectangle(im, (550, 280), (790, 527), azul, 5)

# (2, 4), (1, 3)
...
# -----
# cv2.rectangle(im, (790, 155), (950, 500), azul, 5)
recorte = im[155:500, 790:950]

# ----

cv2.imshow("Doguinhos", recorte)
cv2.waitKey(0)
```

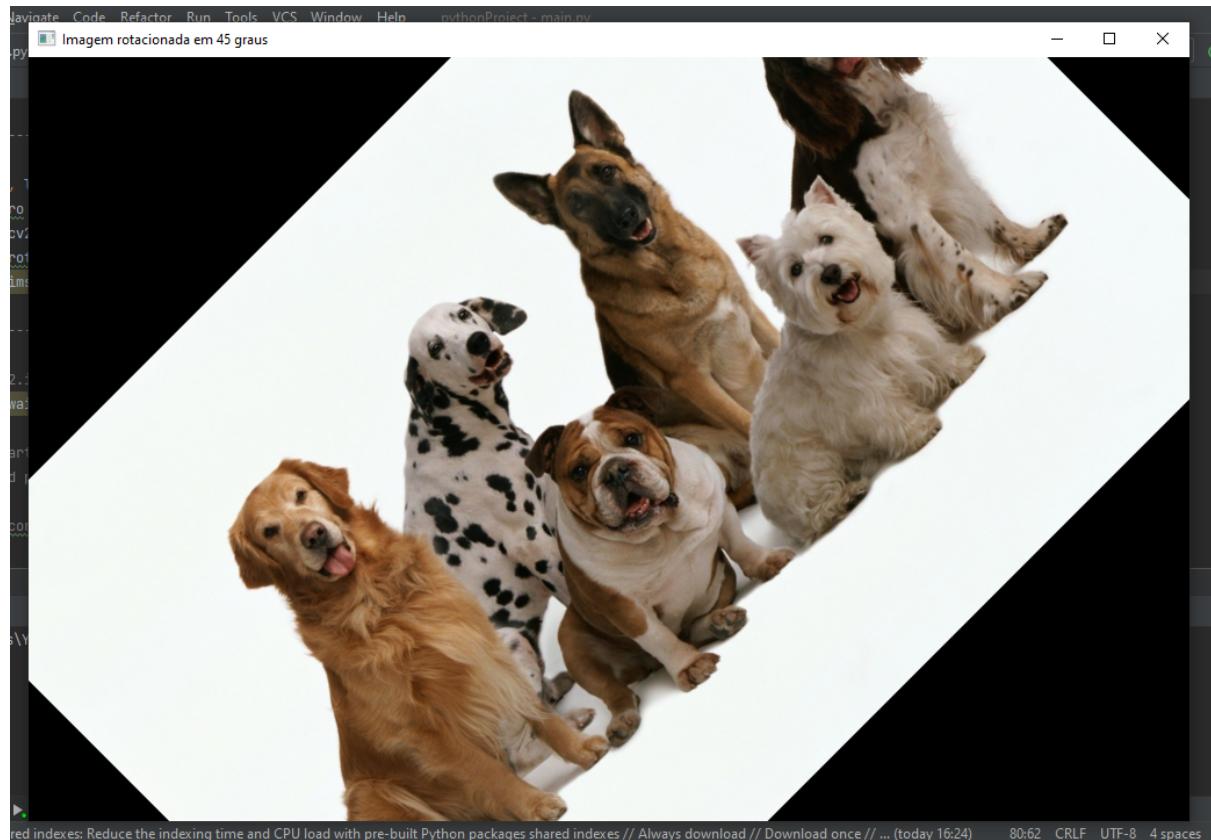
2 - Na mesma imagem (original), localizar todos os cães e remover o resto da imagem, colocando o fundo na cor preta, somente os cães devem ficar na imagem.



shared indexes: Reduce the indexing time and CPU load with pre-built Python packages shared indexes // Always download // Download once // ... (today 16:24) 84:44 CRLF UTF-8 4 space

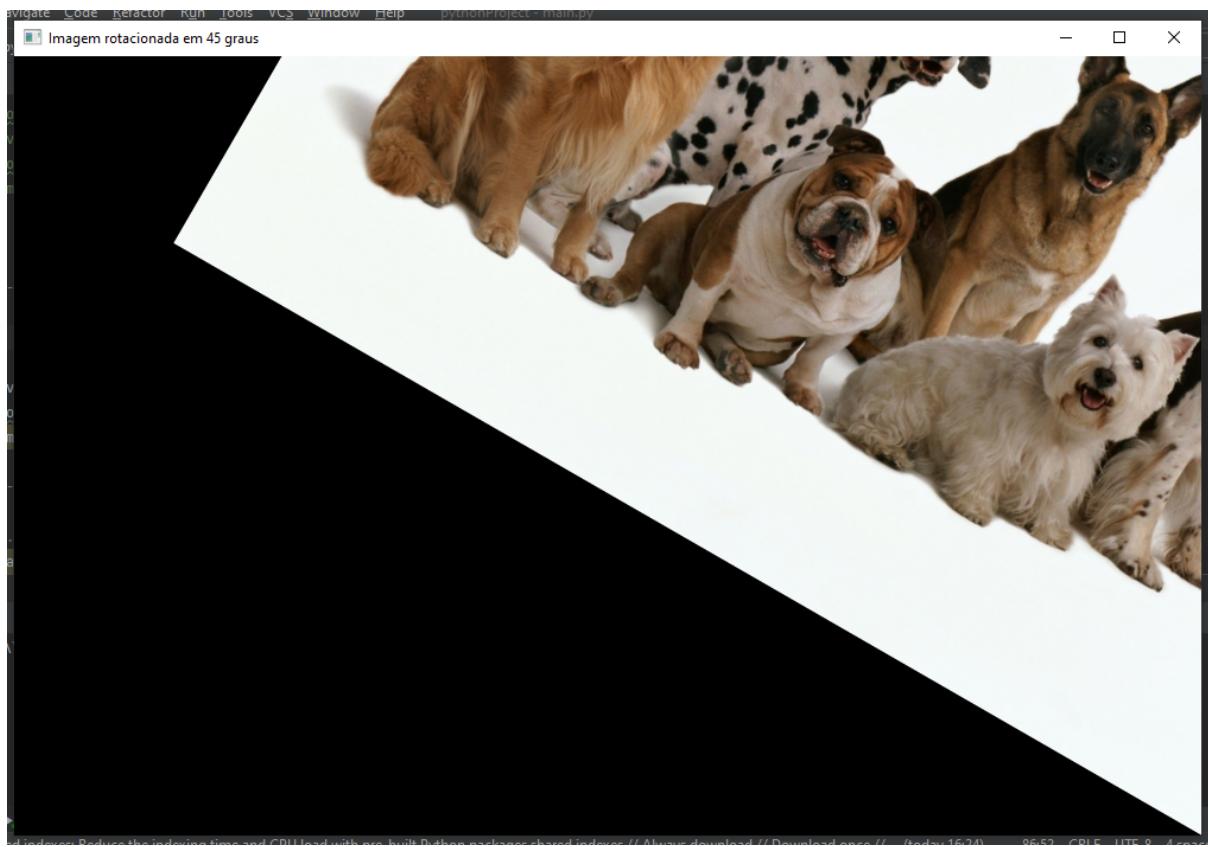
```
mask = np.zeros(im.shape[:2], dtype="uint8")
cv2.rectangle(mask, (90, 92), (950, 535), azul, -1)
masked = cv2.bitwise_and(im, im, mask=mask)
cv2.imshow("Mask Applied to Image", masked)
```

3 - Rotacionar a imagem com relação ao centro da imagem em 45 graus.



```
(alt, lar) = im.shape[:2] #captura altura e largura
centro = (lar // 2, alt // 2) #acha o centro
M = cv2.getRotationMatrix2D(centro, 45, 1.0) #30 graus
img_rotacionada = cv2.warpAffine(im, M, (lar, alt))
cv2.imshow("Imagen rotacionada em 45 graus", img_rotacionada)
```

4 - Rotacionar a imagem com relação ao centro do cão localizado mais a direita e abaixo, em -30 graus.



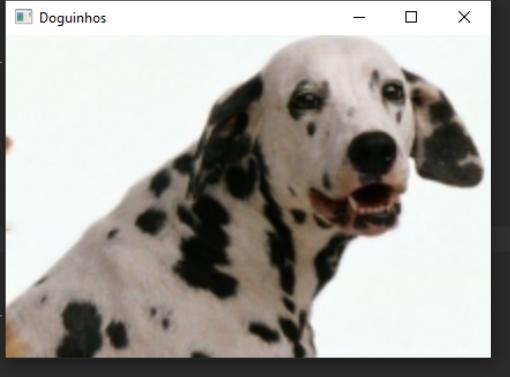
```
(alt, lar) = im.shape[:2] #captura altura e largura  
pin = (lar, alt) #acha o centro  
M = cv2.getRotationMatrix2D(pin, -30, 1.0) #30 graus  
img_rotacionada = cv2.warpAffine(im, M, (lar, alt))  
cv2.imshow("Imagen rotacionada em 45 graus", img_rotacionada)
```

5 - Reduzir a imagem do maior cão pela metade.



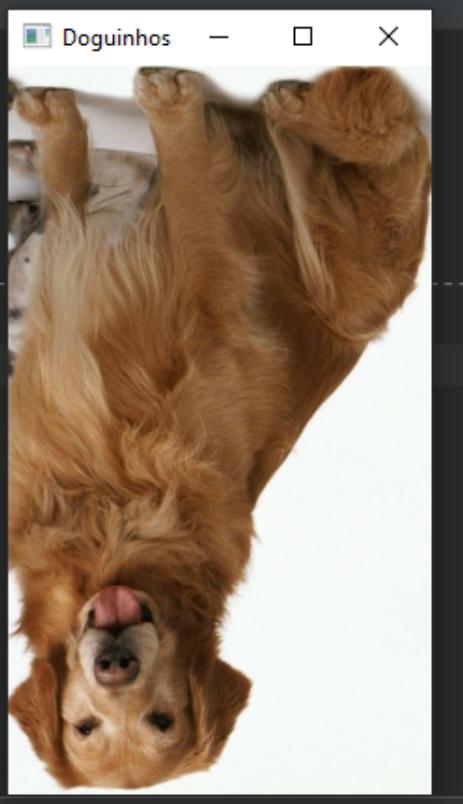
6 - Ampliar a imagem do menor cão em 2 vezes.

```
106     image = cv2.flip(im, 1)
107     '''
108
109     # -----
110
111     def zoom(img, zoom_factor):
112         return cv2.resize(img, None, fx=zoom_factor, fy=zoom_factor)
113
114
115     recorte_menor = im[140:280, 320:530]
116
117     zoomed = zoom(recorte_menor, 2)
118
119     # -----
120
121     cv2.imshow("Doguinhas", zoomed)
```



7 - Fazer um flip de -1, 0 e 1 com a imagem individual do maior e menor cão.

```
main.py ×
103 [EXERCI 7]
104     image = cv2.flip(im, -1)
105     image = cv2.flip(im, 0)
106     image = cv2.flip(im, 1)
107     '''
108
109     # -----
110
111     recorte_maior = im[140:530, 93:320]
112     recorte_menor = im[140:280, 320:530]
113
114     image = cv2.flip(recorte_maior, -1)
115
116     '''
117
118     image = cv2.flip(recorte_maior, 0)
119     image = cv2.flip(recorte_maior, 1)
120
121     image = cv2.flip(recorte_menor, -1)
```



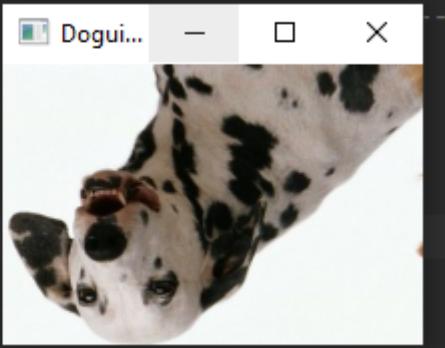
```
main.py x
103 [EXERCÍCIO 7]
104     image = cv2.flip(im, -1)
105     image = cv2.flip(im, 0)
106     image = cv2.flip(im, 1)
107     '''
108
109     # -----
110
111     recorte_maior = im[140:530, 93:320]
112     recorte_menor = im[140:280, 320:530]
113
114     image = cv2.flip(recorte_maior, 0)
115
116     '''
117
118
119     image = cv2.flip(recorte_maior, 1)
120
121     image = cv2.flip(recorte_menor, -1)
```



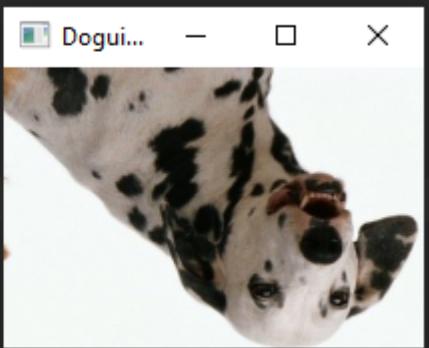
```
main.py x
103 [EXERCÍCIO 7]
104     image = cv2.flip(im, -1)
105     image = cv2.flip(im, 0)
106     image = cv2.flip(im, 1)
107     '''
108
109     # -----
110
111     recorte_maior = im[140:530, 93:320]
112     recorte_menor = im[140:280, 320:530]
113
114     image = cv2.flip(recorte_maior, 1)
115
116     '''
117
118
119
120
121     image = cv2.flip(recorte_menor, -1)
```



```
# -----  
  
recorte_maior = im[140:530, 93:320]  
recorte_menor = im[140:280, 320:530]  
  
image = cv2.flip(recorte_menor, -1)  
  
'''  
image = cv2.flip(recorte_menor, 0)
```



```
recorte_maior = im[140:530, 93:320]  
recorte_menor = im[140:280, 320:530]  
  
image = cv2.flip(recorte_menor, 0)  
  
'''
```



```
# -----  
  
recorte_maior = im[140:530, 93:320]  
recorte_menor = im[140:280, 320:530]  
  
image = cv2.flip(recorte_menor, 1)  
  
# -----
```

