



Processamento da Informação

Google Meet - 02

<http://professor.ufabc.edu.br/~jesus.mena/courses/pi-1q-2020/PI-google-meet/>



Laços aninhados

Laços

```
for i in range(0, 5):  
    print(i)
```

0
1
2
3
4

Laços aninhados

```
for i in range(0, 5):  
    for j in range(0, 3):  
        print(i, j)  
    print()
```

```
0 0  
0 1  
0 2
```

```
1 0  
1 1  
1 2
```

```
2 0  
2 1  
2 2
```

```
3 0  
3 1  
3 2
```

```
4 0  
4 1  
4 2
```

Laços aninhados

```
for i in range(0, 5):  
    for j in range(0, 3):  
        print("({},{}) ".format(i, j), end="")  
    print()
```

```
(0,0) (0,1) (0,2)  
(1,0) (1,1) (1,2)  
(2,0) (2,1) (2,2)  
(3,0) (3,1) (3,2)  
(4,0) (4,1) (4,2)
```

Laços aninhados

```
for i in range(0, 5):  
    for j in range(0, 3):  
        for k in range(0,7):  
            print(i, j, k)  
        print()
```

```
1 0 0  
1 0 1  
1 0 2  
1 0 3  
1 0 4  
1 0 5  
1 0 6  
1 1 0  
1 1 1  
1 1 2  
1 1 3  
1 1 4  
1 1 5  
1 1 6  
1 2 0  
1 2 1  
1 2 2  
1 2 3  
1 2 4  
1 2 5  
1 2 6
```

...

```
4 0 0  
4 0 1  
4 0 2  
4 0 3  
4 0 4  
4 0 5  
4 0 6  
4 1 0  
4 1 1  
4 1 2  
4 1 3  
4 1 4  
4 1 5  
4 1 6  
4 2 0  
4 2 1  
4 2 2  
4 2 3  
4 2 4  
4 2 5  
4 2 6
```



quantos segundos temos em um dia?



All

Images

Videos

News

Shopping

More

Settings

Tools

About 137,000,000 results (0.59 seconds)

Time

1

=

86400

Day

Second

Formula

multiply the time value by 86400

[More info](#)

[Feedback](#)

[brainly.com.br](#) › ... › Matemática ▾ [Translate this page](#)

[um dia tem quantos segundos - Brainly.com.br](#)

2 answers

Nov 26, 2012 - Como **temos** grandezas diretamente proporcionais, basta multiplicar esses valores para calcular **quantos segundos** existem em um **dia**.

```
soma = 0

for hora in range(0, 24):
    for minuto in range(0, 60):
        for segundo in range(0, 60):
            soma = soma+1

print(soma)
```

86400



Lembrando as matrizes

Matriz bidimensional

É uma estrutura de números (ou símbolos ou expressões) distribuídos em linhas e colunas.

Exemplo de uma matriz de 2 linhas e 3 colunas:

$$A = \begin{pmatrix} 1 & 9 & -13 \\ 20 & -5 & 60 \end{pmatrix}$$

Matriz bidimensional

Na grande maioria das linguagens de programação a numeração inicia em zero (e não em 1).

As posições de cada elemento Inicia em zero.

$$A = \begin{pmatrix} 7 & 9 & -10 \\ 20 & -5 & 60 \end{pmatrix}$$

$$A = \begin{pmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \end{pmatrix}$$

Matriz bidimensional em Python

$$A = \begin{pmatrix} 7 & 9 & -10 \\ 20 & -5 & 60 \end{pmatrix}$$

```
A = [ [7, 9, -10], [20, -5, 60] ]
```

Matriz bidimensional em Python

```
A = [ [7, 9, -10], [20, -5, 60] ]
```

```
print(A)
```

```
[[7, 9, -10], [20, -5, 60]]
```

```
print(A[0])
```

```
[7, 9, -10]
```

```
print(A[1])
```

```
[20, -5, 60]
```

```
print(len(A))
```

```
2
```

```
print(len(A[0]))
```

```
3
```

```
print(len(A[1]))
```

```
3
```

Matriz bidimensional em Python

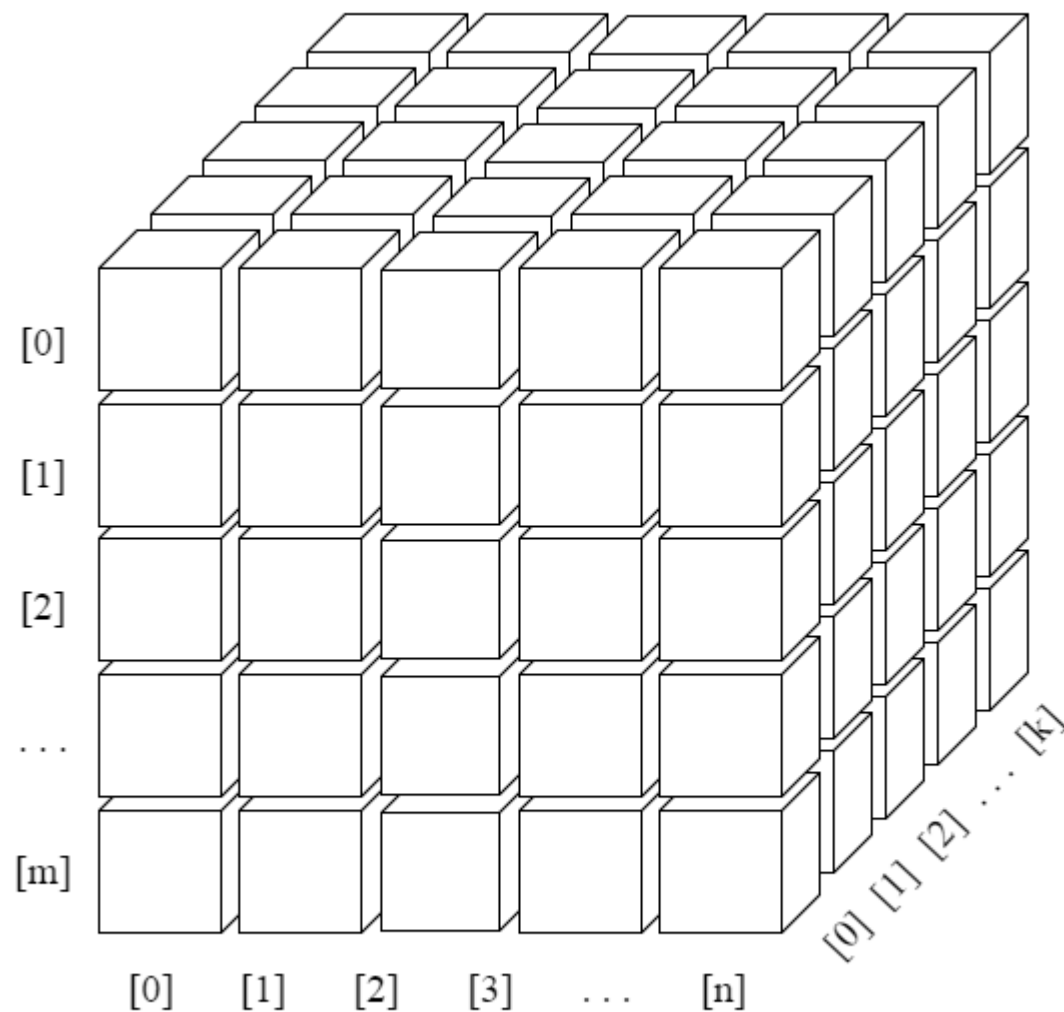
$$A = \begin{pmatrix} 7 & 9 & -10 \\ 20 & -5 & 60 \end{pmatrix}$$

```
for i in range(len(A)):
    for j in range(len(A[0])):
        print(i, j, A[i][j])
```

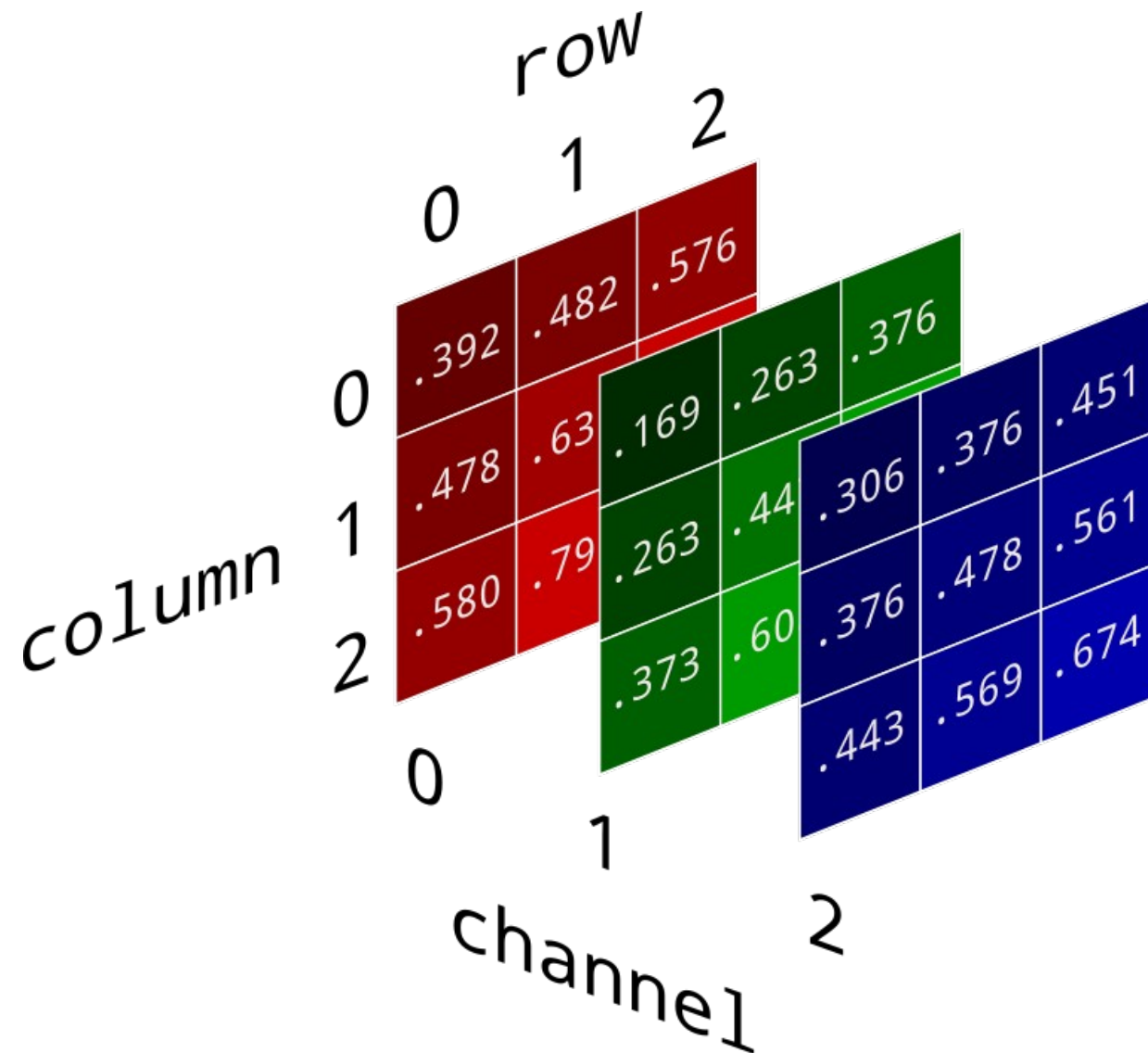
```
0 0 7
0 1 9
0 2 -10
1 0 20
1 1 -5
1 2 60
```

Matriz tridimensional

Tridimensional Matrix (or cube...)



Matriz tridimensional - Red-Green-Blue





Uma aplicação de matrizes:

Processamento de imagens



[Albrecht Dürer, 1525]

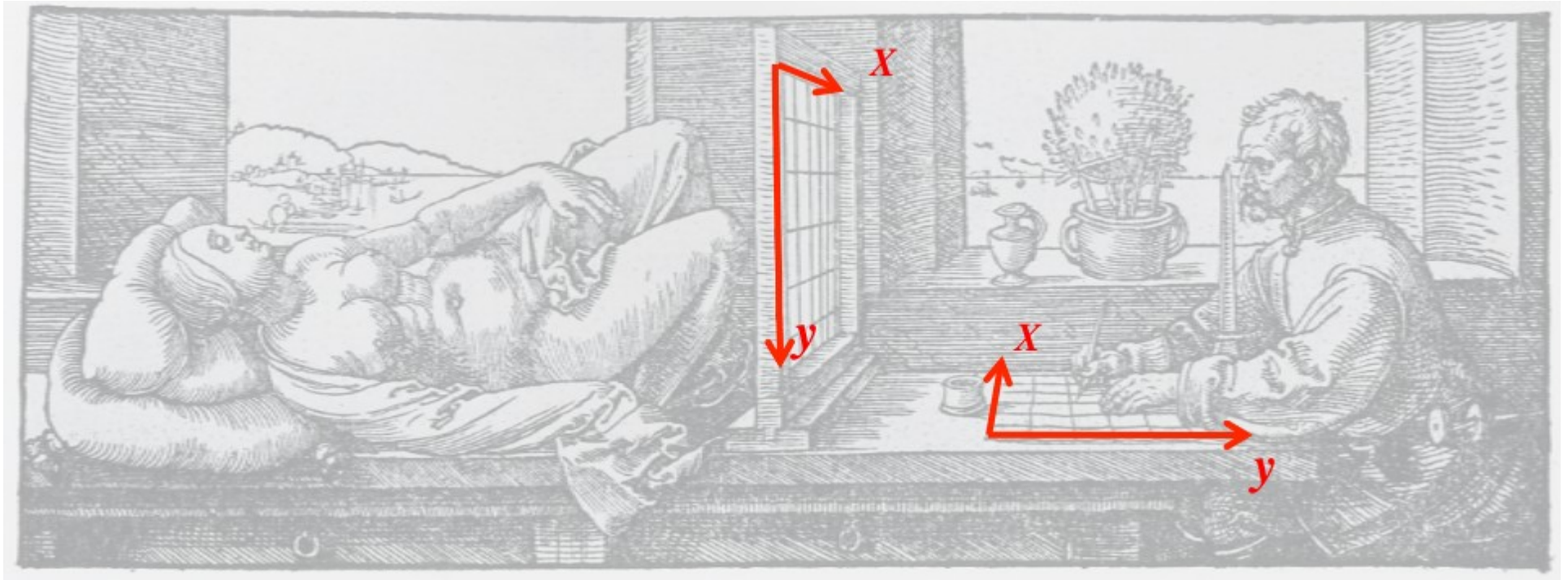


Imagem: Uma representação visual na forma de uma função $f(x,y)$ em que f está relacionado ao brilho (ou cor) no ponto (x,y) .

Imagem digital e pixels

- **Imagem digital:** amostras discretas $f[x,y]$ que representam imagens contínuas $f(x,y)$.
- Cada elemento da matriz 2-D é denominado um **pixel** (de “*picture element*”).



200x200



100x100

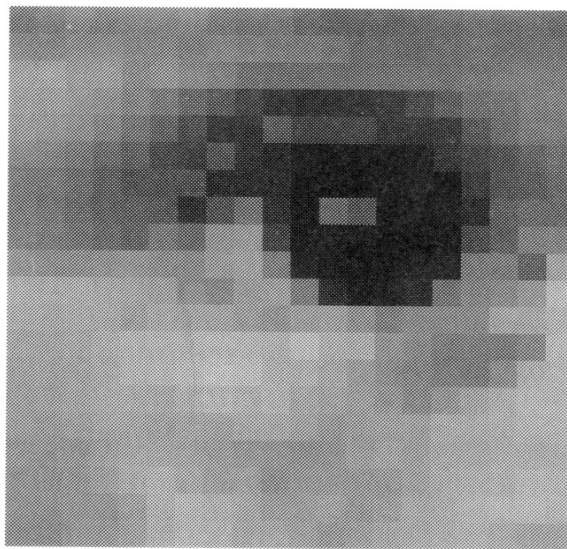


50x50



25x25

Imagem em tons de cinza



```
117 125 133 127 130 130 133 121 116 115 100 91 93 94 99 103 112 105 109 106
134 133 138 138 132 134 130 133 128 123 121 113 106 102 99 106 113 109 109 113
146 147 138 140 125 134 124 115 102 96 93 94 99 96 99 100 103 110 109 110
144 141 136 130 120 108 88 74 53 37 31 37 35 39 53 79 93 100 109 116
139 136 129 119 102 85 58 31 41 77 51 53 53 33 37 41 69 94 105 108
132 127 117 102 87 57 49 77 42 28 17 15 13 13 17 41 53 69 88 100
124 120 108 94 72 74 72 31 35 31 15 13 15 11 15 13 46 75 83 96
125 115 102 93 88 82 42 79 113 41 19 100 82 11 11 17 31 91 99 100
124 116 109 99 91 113 99 140 144 57 20 20 15 11 15 17 63 87 119 124
136 133 133 135 138 133 132 144 150 120 24 17 15 15 17 20 115 113 88 150
158 157 157 154 149 145 133 127 146 150 116 35 20 19 28 105 124 128 141 171
155 154 156 155 146 155 154 154 147 139 148 150 138 120 128 129 130 151 156 165
150 151 154 162 166 167 169 174 172 167 177 166 164 140 134 120 121 120 127 172
145 149 151 157 165 169 173 179 176 166 166 157 145 136 129 124 120 136 163 168
144 148 153 160 159 158 165 172 165 169 157 151 149 141 130 140 151 162 169 167
144 141 147 155 154 149 156 151 157 157 151 144 147 147 149 159 158 159 166 165
139 140 140 150 153 151 150 146 140 139 138 140 145 151 149 156 156 162 162 161
136 134 138 146 156 164 153 146 145 136 139 139 140 141 149 157 159 161 169 166
136 133 136 135 144 159 168 159 151 142 141 145 139 146 153 156 164 167 172 168
133 129 140 142 146 159 167 165 154 151 146 141 147 154 156 160 161 157 153 154
```

Imagem colorida



Red $R[x,y]$



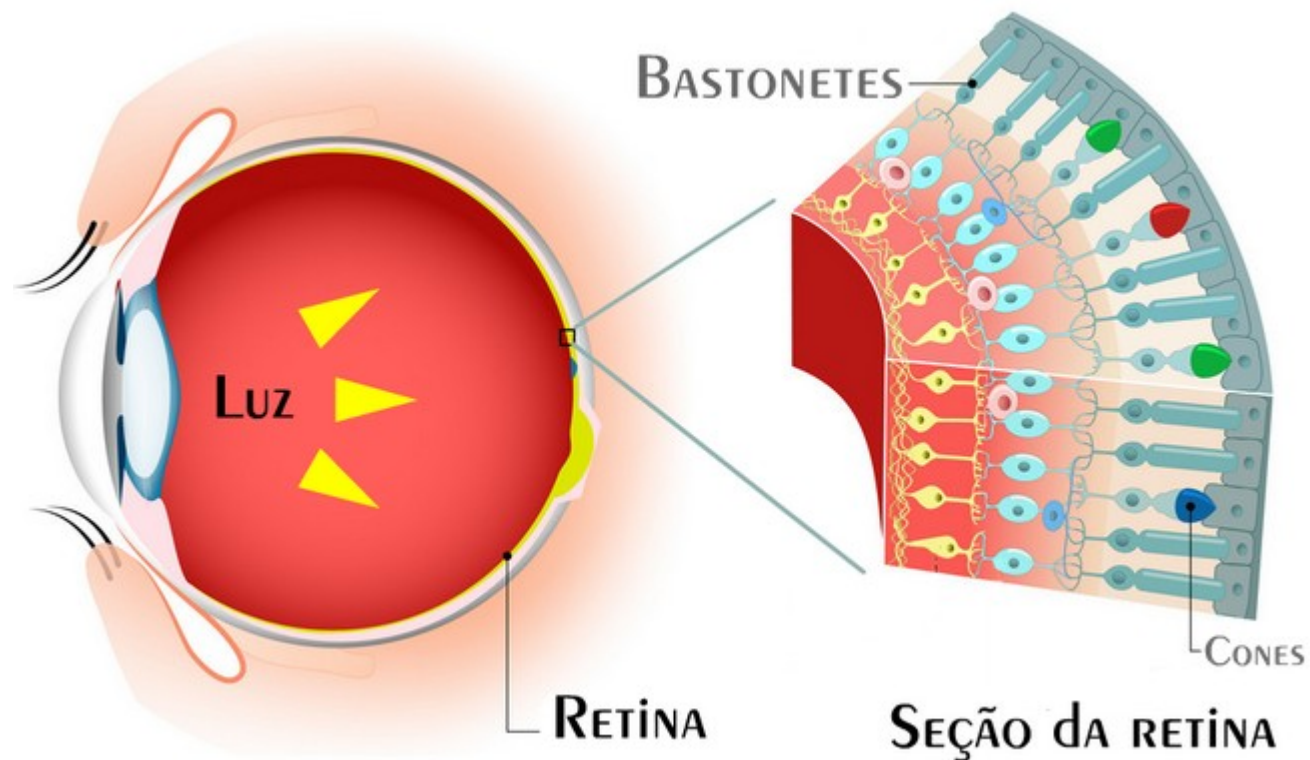
Green $G[x,y]$



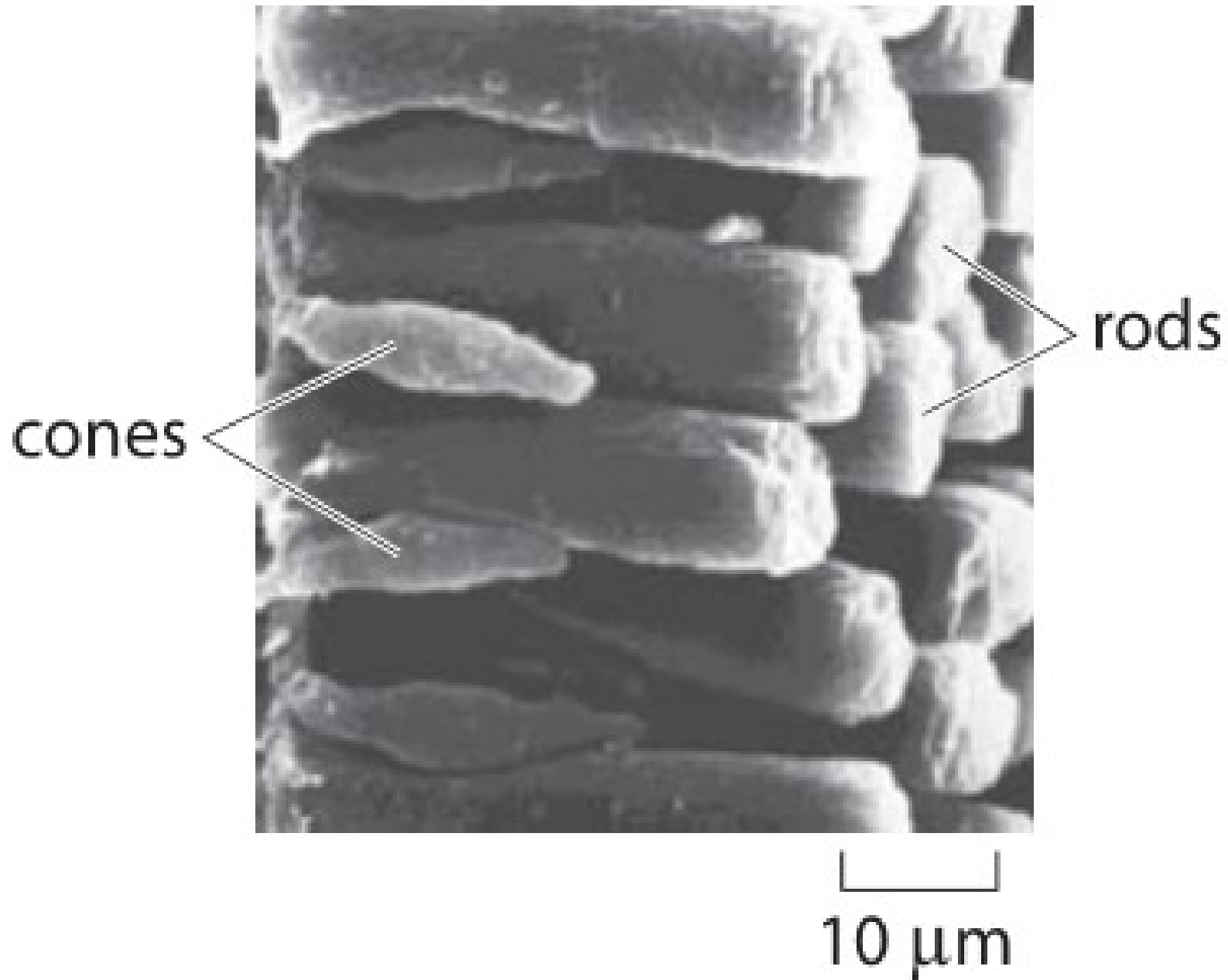
Blue $B[x,y]$

Cones e Bastonetes

RGB = Red, Green, Blue



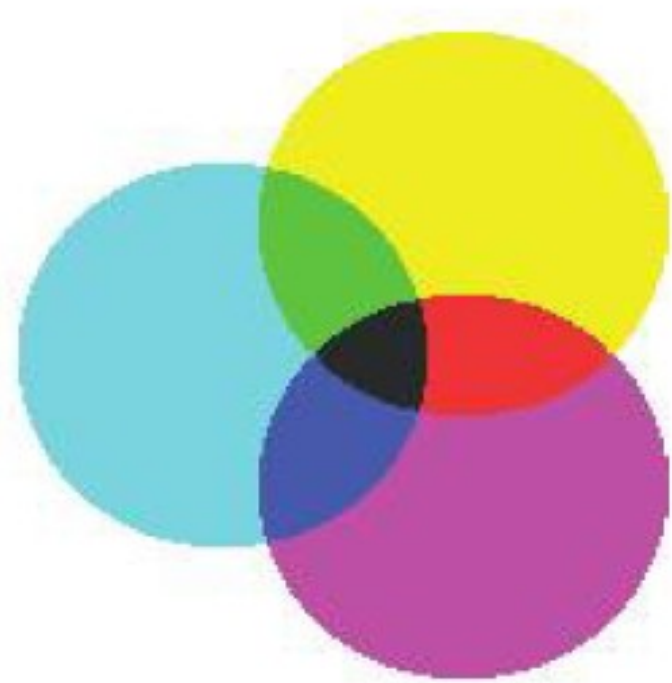
Cones e Bastonetes



Criando cores?



Sistema RGB (red, green, blue)



Sistema CMY (cyan, magenta, yellow)



Exemplos em Python

Operações básicas



teste0a.py

```
1  import matplotlib.pyplot as plt
2  import matplotlib.image as mpimg
3
4  # Leitura da fotografia
5  foto = mpimg.imread('foto0.png')
6  linhas = len(foto)
7  colunas = len(foto[0])
8
9  # Inserimos linhas pretas a cada 10 pixels
10 for i in range(linhas):
11     for j in range(colunas):
12         if i%10==0:
13             foto[i,j] = [0, 0, 0]
14
15 # Visualizar a fotografia
16 plt.imshow(foto)
17 plt.show()
```

teste0b.py

```
1  import matplotlib.pyplot as plt
2  import matplotlib.image as mpimg
3
4  # Leitura da fotografia
5  foto = mpimg.imread('foto0.png')
6  linhas = len(foto)
7  colunas = len(foto[0])
8
9  # Inserimos linhas pretas a cada 10 pixels
10 for i in range(linhas):
11     for j in range(colunas):
12         if i%10==0:
13             if sum(foto[i,j])<0.2:
14                 foto[i,j] = [1, 1, 1]
15             else:
16                 foto[i,j] = [0, 0, 0]
17
18 # Visualizar a fotografia
19 plt.imshow(foto)
20 plt.show()
```



teste1a.py

```
1  import matplotlib.pyplot as plt
2  import matplotlib.image as mpimg
3
4  # Leitura da fotografia
5  foto = mpimg.imread('foto1.png')
6  linhas = len(foto)
7  colunas = len(foto[0])
8
9  for i in range(linhas):
10     for j in range(colunas):
11         if i>200 and i<300:
12             foto[i,j] = foto[i,j][0]
13         if i>300 and i<400:
14             foto[i,j] = foto[i,j][1]
15         if i>400 and i<500:
16             foto[i,j] = foto[i,j][2]
17
18  # Visualizar a fotografia
19  plt.imshow(foto)
20  plt.show()
```



teste1b.py

```
1  import matplotlib.pyplot as plt
2  import matplotlib.image as mpimg
3
4  # Leitura da fotografia
5  foto = mpimg.imread('foto1.png')
6  linhas = len(foto)
7  colunas = len(foto[0])
8
9  for i in range(linhas):
10     |   for j in range(colunas):
11     |   |   foto[i,j] = sum(foto[i,j])/3
12
13  # Visualizar a fotografia
14  plt.imshow(foto)
15  plt.show()
```



teste1c.py

```
1  import matplotlib.pyplot as plt
2  import matplotlib.image as mpimg
3
4  # Leitura da fotografia
5  foto = mpimg.imread('foto1.png')
6  linhas = len(foto)
7  colunas = len(foto[0])
8
9  for i in range(linhas):
10     for j in range(colunas):
11         if i>100 and j>300:
12             foto[i,j] = 1-foto[i,j]
13
14  # Visualizar a fotografia
15  plt.imshow(foto)
16  plt.show()
```



Efeito "pôr do sol"

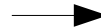
No atardecer existe uma redução do **alcance** das cores **azul** e **verde**, então esse efeito pode ser simulado através da redução dessas cores.



teste-pordosol.py

```
1  import matplotlib.pyplot as plt
2  import matplotlib.image as mpimg
3
4  # Leitura da fotografia
5  foto = mpimg.imread('foto2.png')
6  linhas = len(foto)
7  colunas = len(foto[0])
8
9  for i in range(linhas):
10     for j in range(colunas):
11         foto[i,j][1] = foto[i,j][1]*0.5
12         foto[i,j][2] = foto[i,j][2]*0.5
13
14  # Visualizar a fotografia
15  plt.imshow(foto)
16  plt.show()
```

Efeito sepia (envelhecimento)



Efeito sepia (envelhecimento)

Para escalas de cinzas:

$$\text{cinza} = (R+G+B)/3$$

O efeito sepia pode ser obtido da seguinte forma:

$$Y = 0.299*R + 0.587*G + 0.114*B$$

← luminância

Os novos valores para cada canal serão:

$$R = Y + 2*sepia$$

$$G = Y + 1*sepia$$

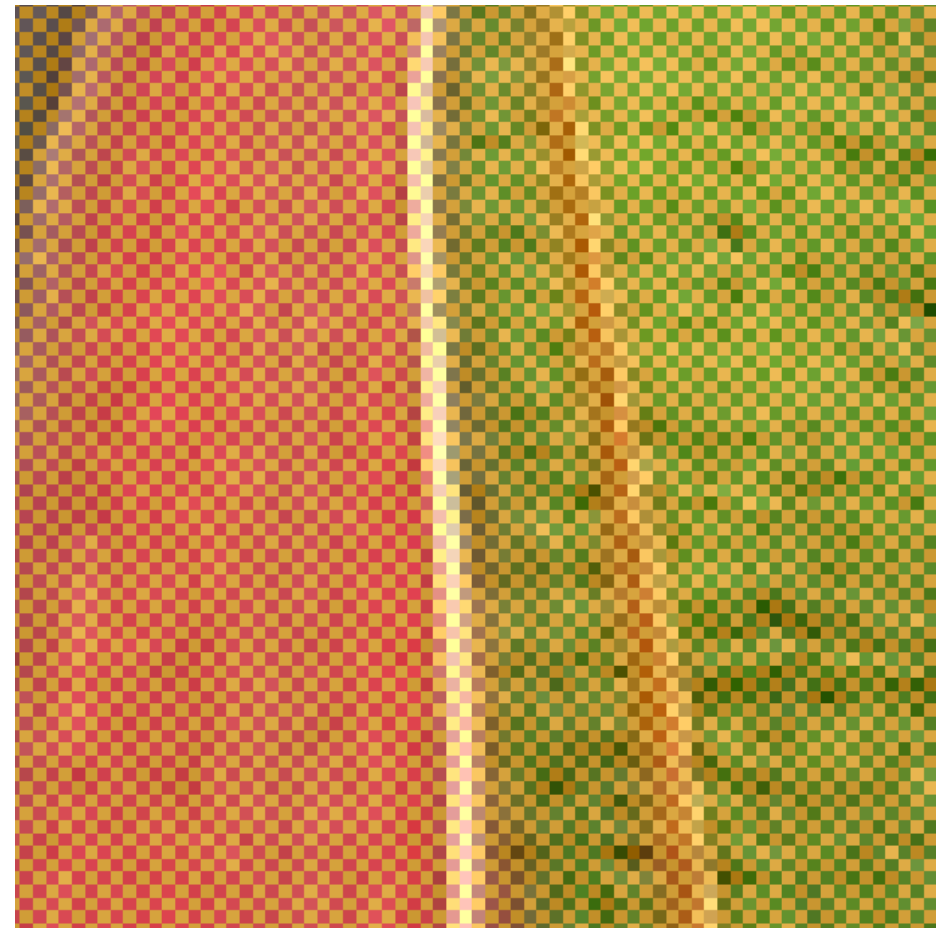
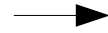
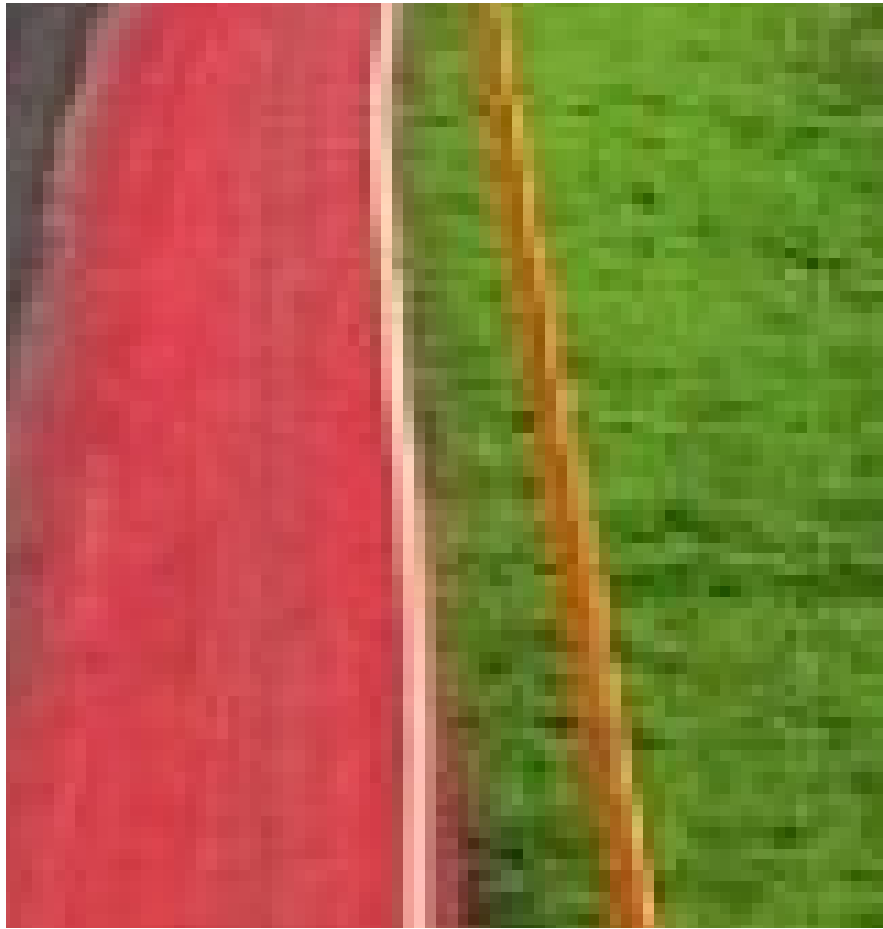
$$B = Y - 1*sepia$$

Considere valores se $sepia=0.20$ ou $sepia=0.40$

teste-sepia.py

```
1  import matplotlib.pyplot as plt
2  import matplotlib.image as mpimg
3
4  # Leitura da fotografia
5  foto = mpimg.imread('foto2.png')
6  linhas = len(foto)
7  colunas = len(foto[0])
8
9  for i in range(linhas):
10     for j in range(colunas):
11         Y = 0.299*foto[i,j][0] + 0.587*foto[i,j][1] + 0.114*foto[i,j][2]
12
13         sepia = 0.2
14         foto[i,j][0] = Y + 2*sepia
15         foto[i,j][1] = Y + 1*sepia
16         foto[i,j][2] = Y - 1*sepia
17
18  # Visualizar a fotografia
19  plt.imshow(foto)
20  plt.show()
```

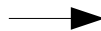
Efeito 'vintage'



teste-vintage.py

```
1  import matplotlib.pyplot as plt
2  import matplotlib.image as mpimg
3
4  # Leitura da fotografia
5  foto = mpimg.imread('foto2.png')
6  linhas = len(foto)
7  colunas = len(foto[0])
8
9  for i in range(linhas):
10     for j in range(colunas):
11         sepia = 0.2
12         Y = 0.299*foto[i,j][0] + 0.587*foto[i,j][1] + 0.114*foto[i,j][2]
13
14         if (i+j)%2==0:
15             foto[i,j][0] = Y + 2*sepia
16             foto[i,j][1] = Y + 1*sepia
17             foto[i,j][2] = Y - 1*sepia
18
19  # Visualizar a fotografia
20  plt.imshow(foto)
21  plt.show()
```

Efeito círculo



teste-circulo.py

```
# Desenho de um círculo
for i in range(linhas):
    for j in range(colunas):
        if (i-linhas//2)**2 + (j-colunas//2)**2 >= 250**2:
            foto[i,j] = sum(foto[i,j])/3 - 0.5
```

```
# Salvando a imagem
for i in range(linhas):
    for j in range(colunas):
        for k in range(3):
            if foto[i,j,k]<0:
                foto[i,j,k] = 0
            if foto[i,j,k]>1:
                foto[i,j,k] = 1
```

```
mpimg.imsave("foto2d.png", foto)
```



Para finalizar ...

Python como ferramenta!

FREE PYTHON TOOLS

Pandas - used for data analysis	NumPy - multidimensional arrays
SciPy - algorithms to use with numpy	Matplotlib - data visualization tool
HDFS - used to store and manipulate data	PyTables - used for managing HDF5 datasets
Jupyter - research collaboration tool	IPython - powerful shell
HDFS - C/C++ wrapper for Hadoop	Pymongo - MongoDB driver
SQLAlchemy - Python SQL Toolkit	Redis - Redis' access libraries
pyMySQL - MySQL connector	Scikit-learn - used for machine learning algorithms
Theano - deep learning with neural networks	Keras - high-level neural networks API
Lasagne - build and train neural networks in Theano	Bokeh - data visualization tool
Seaborn - data visualization tool	Dask - data engineering tool
Airflow - data engineering tool	Luigi - data engineering tool
Elasticsearch - data search engine	SymPy - symbolic math
PyBrain - algorithms for ML	Pattern - natural language processing