cellular_automata_ruido_invbits_1cell

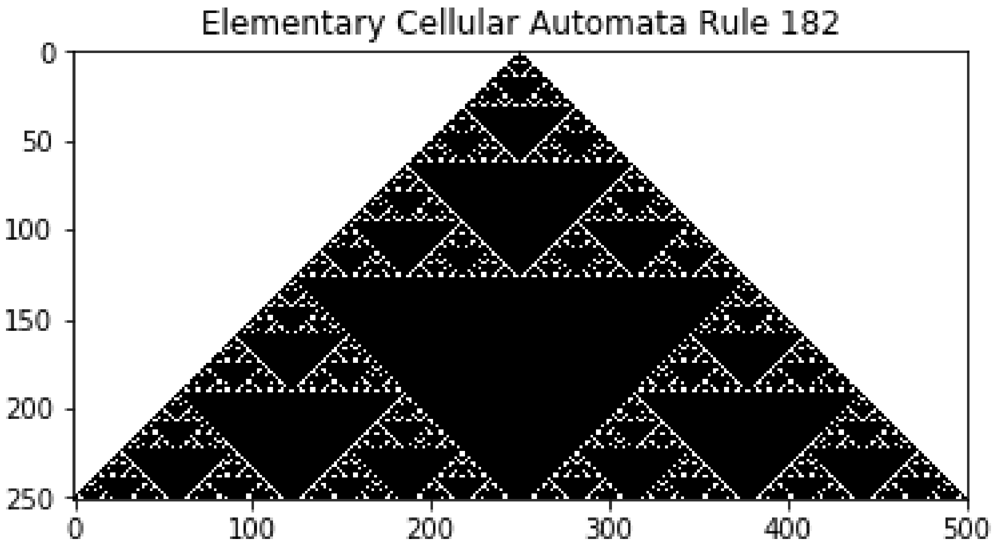
June 4, 2018

In [2]:

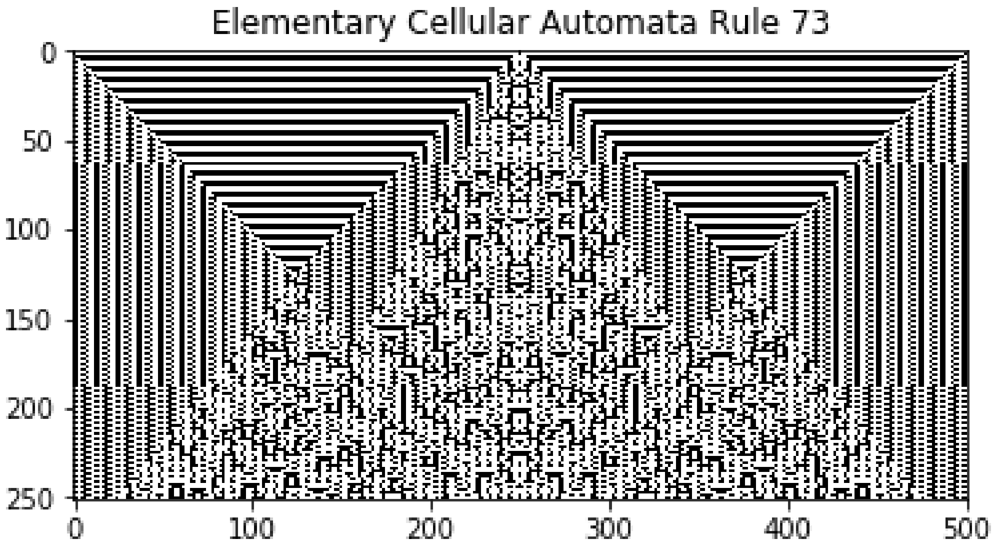
```
import numpy
import time
import matplotlib.pyplot as plt

# Pattern
#rulein = 30

#print 'Rule without noise:', rulein

rulein = input('Regra: ')

output_pattern = [int(x) for x in numpy.binary_repr(rulein, width=8)]
output_pattern

input_pattern = numpy.zeros([8, 3])
for i in range(8):
    input_pattern[i, :] = [int(x) for x in numpy.binary_repr(7-i, width=3)]

input_pattern

columns = 501
rows = int(columns/2)+1

canvas = numpy.zeros([rows, columns+2])
canvas[0, int(columns/2)+1] = 1

for i in numpy.arange(0, rows-1):
    for j in numpy.arange(0, columns):
        for k in range(8):
            if numpy.array_equal(input_pattern[k, :], canvas[i, j:j+3]):
                canvas[i+1, j+1] = output_pattern[k]

plt.imshow(canvas[:, 1:columns+1], cmap='Greys', interpolation='nearest')
plt.title("Elementary Cellular Automata Rule {}".format(rulein))
plt.show()
start_time1 = time.time()
print("Execution time 1 = %s seconds" % (time.time() - start_time1))
```

```
# Gerar ruído de inversão de todos os bits. Ex.: 10101010b => 01010101b

# Inverter os bits da "rulein"
bin(numpy.uint8(rulein))

comruido = bin(~numpy.uint8(rulein))

# Bits invertidos da "rulein"
print comruido

int(comruido,2)

ruleout = int(comruido,2)
print 'New Rule with Noise:', ruleout

output_pattern = [int(x) for x in numpy.binary_repr(ruleout, width=8)]

output_pattern

input_pattern = numpy.zeros([8, 3])
for i in range(8):
    input_pattern[i, :] = [int(x) for x in numpy.binary_repr(7-i, width=3)]

input_pattern

columns = 501
rows = int(columns/2)+1

canvas = numpy.zeros([rows, columns+2])
canvas[0, int(columns/2)+1] = 1

for i in numpy.arange(0, rows-1):
    for j in numpy.arange(0, columns):
        for k in range(8):
            if numpy.array_equal(input_pattern[k, :], canvas[i, j:j+3]):
                canvas[i+1, j+1] = output_pattern[k]

plt.imshow(canvas[:, 1:columns+1], cmap='Greys', interpolation='nearest')
plt.title("Elementary Cellular Automata Rule {}".format(ruleout))
plt.show()
start_time2 = time.time()
print("Execution time 2 = %s seconds" % (time.time() - start_time2))
print
print ("Execution time difference between them =", start_time2 - start_time1)
```

```
Regra: 182
```



```
Execution time 1 = 8.41617584229e-05 seconds
0b1001001
New Rule with Noise: 73
```



```
Execution time 2 = 5.3882598877e-05 seconds

('Execution time difference between them =', 5.026944160461426)
```