ECA Demo

The purpose of this notebook is to demonstrate a simple python implementation of elementary cellular automata. Work on this class is ongoing, and features like custom plot colors, randomized initial states and rule classification have yet to be implemented. As with any python module, it is necessary to first declare the import. This module will eventually be extended to other kinds of cellular automata, but for now it is necessary to import ECA from ECA.

```
from ECA import ECA
```

Using the ECA class is as simple as declaring an ECA object. The class constructor takes an optional list of arguments, described below.

- N - Integer: Number of nodes per row
- it - Integer: Number of iterations to execute
- ru - Interger : ECAin range [0,255]: Rule to apply
- rand - Boolean: Indication for random start state. False initiates ECA with a single black cell (pending)
- col - List: Two-element list of colors for the ECA plot

If these arguments are not passed when the ECA object is created, they will be initialized to None and must be passed as arguments to the generate function. Below is an example of the output from trying to generate an ECA with undefined parameters. Note that the rand and col values are intialized with values False and ['white', 'black'], respectively.

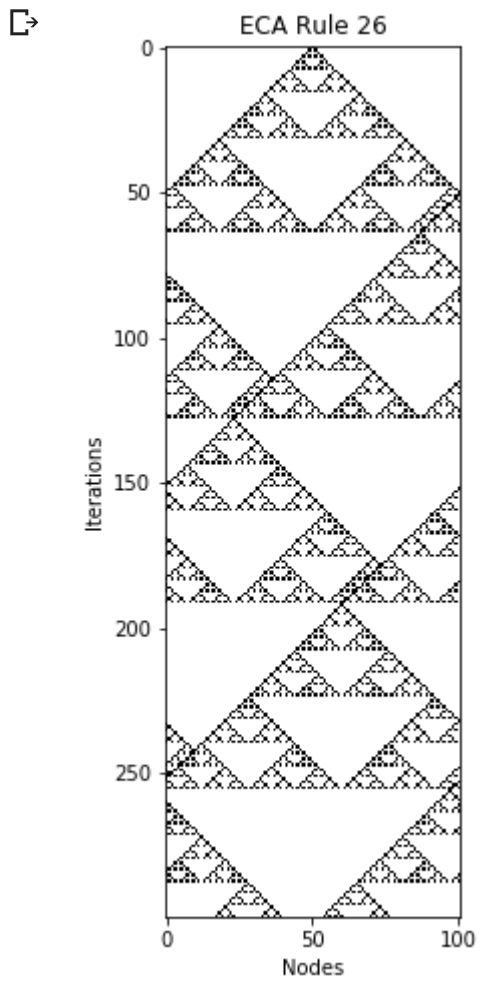
```
e = ECA()
e.generate()
```

```
---------------------------------------------------------------------------
Exception                                 Traceback (most recent call last)
<ipython-input-2-2e592007e5d1> in <module>()
      1 e = ECA()
----> 2 e.generate()

                                  1 frames
/content/ECA.py in __err_check(self)
     79                 err_details.append("ru")
     80         if err_details:
---> 81             raise Exception(err_msg.format(', '.join(err_details)))
     82
     83         if type(self.N) is not int or self.N <= 0:

Exception: ECA generation requires properties N, it, ru to be defined with valid value
```

SEARCH STACK OVERFLOW

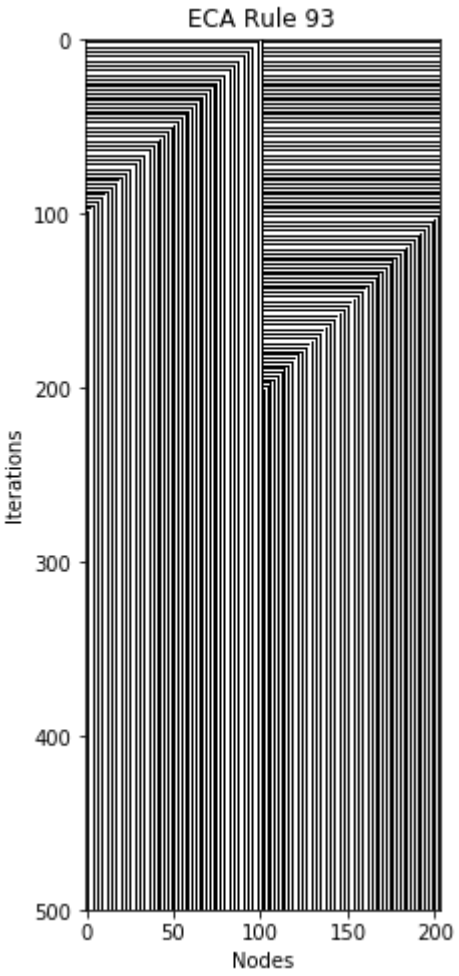
There are two ways to avoid this error. The first is to pass the parameters as arguments to the ECA object instantiation, like so.

```
e = ECA(101, 300, 26, False, ['white', 'black'])
e.generate()
```



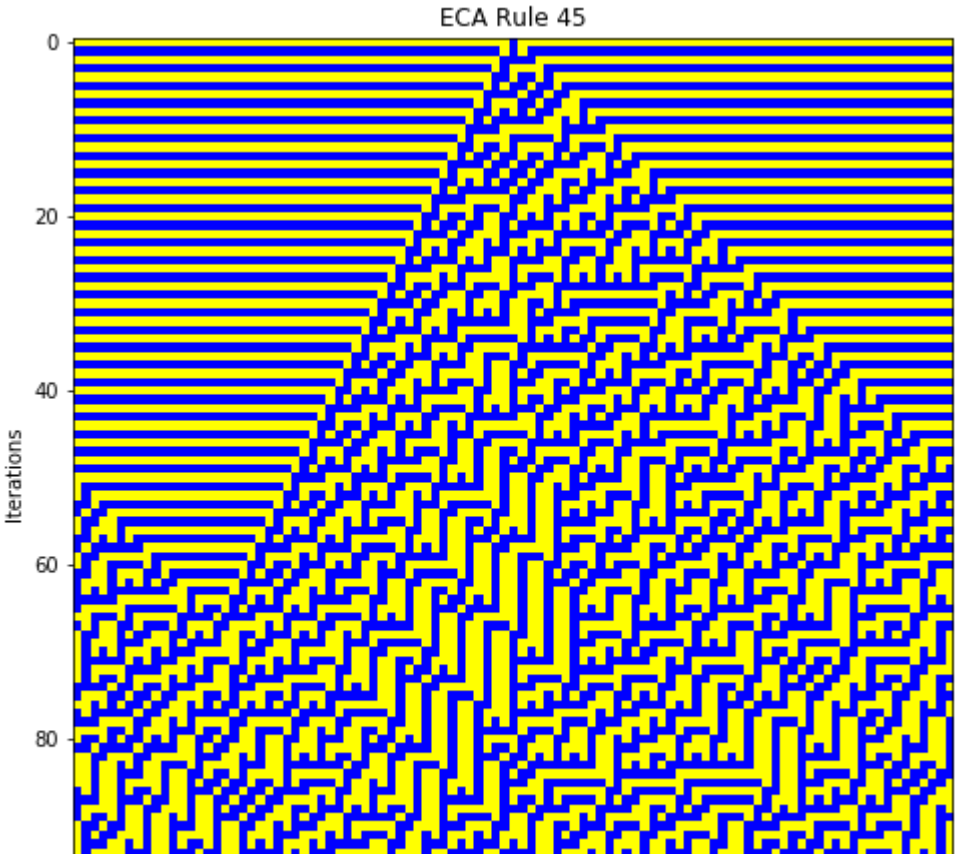
The other option is to pass the parameters as keyword arguments to the generate function, as shown below.

```
e = ECA()
e.generate(N=203, it=500, ru=93, rand=False, col=['white', 'black'])
```



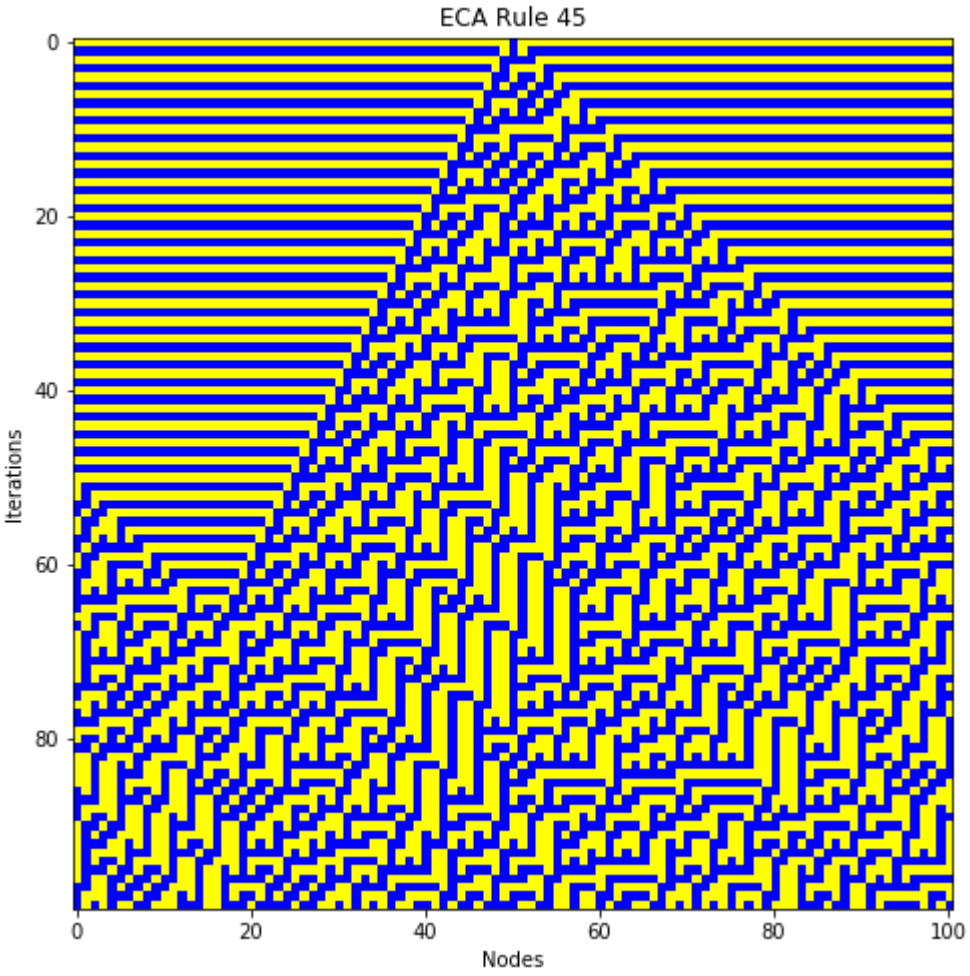
Additionally, it is not necessary to create adiitional ECA objects to generate new automata. Parameters of an ECA object can be changed at any time by passing them as arguments to the generate function. In the example below, e is the same ECA object that was created above.

```
e.generate(ru=45, it=100, N=101, col=['yellow','blue'])
```



Finally, the ECA object with current params can be plotted by using the plot() function. Note that this will only work if the automaton has already been generated using the generate() function.

```
e.plot()
```


ECA Rule 45
Iterations
Nodes
0
20
40
60
80
100