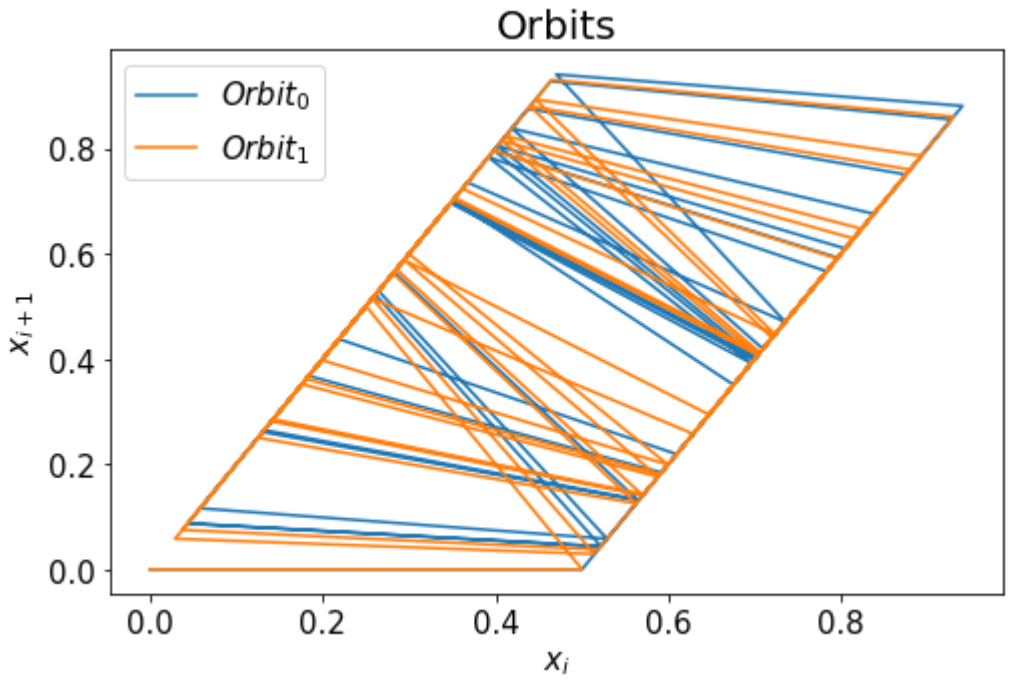
HW1

1. Plot the results of two cases in a plane x_i vs x_{i+1}.

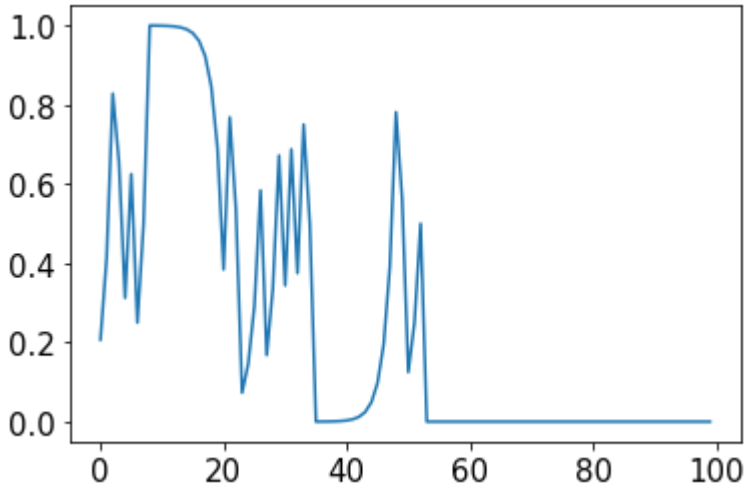
Let's take seed $\in [0, 1)$

In [6]:
```
import numpy as np
import matplotlib.pyplot as plt


def get_orbit(n, f):
    seed, X = np.random.rand(), np.zeros(n)
    X[0] = seed
    for i in range(1, 100):
        X[i] = f(X[i-1])
    return X


def plot_orbits(f, m):
    plt.figure(figsize=(8, 5))
    plt.rc('font', size=15)
    plt.title('Orbits', fontsize=20)
    plt.xlabel('$x_i$')
    plt.ylabel('$x_{i+1}$')

    legends = []

    for k in range(m):
        X = get_orbit(100, f)
        plt.plot([i for i in X][:-1], [i for i in X][1:])
        legends.append(f"$Orbit_{k}$")

    plt.legend(legends, loc=2)
    plt.show()


F = lambda x: 2.0 * x if 0.0 <= x < 0.5 else (2.0 * x) - 1.0
plot_orbits(F, 2)
```



In [7]:
```
plt.plot(get_orbit(100, F))
```

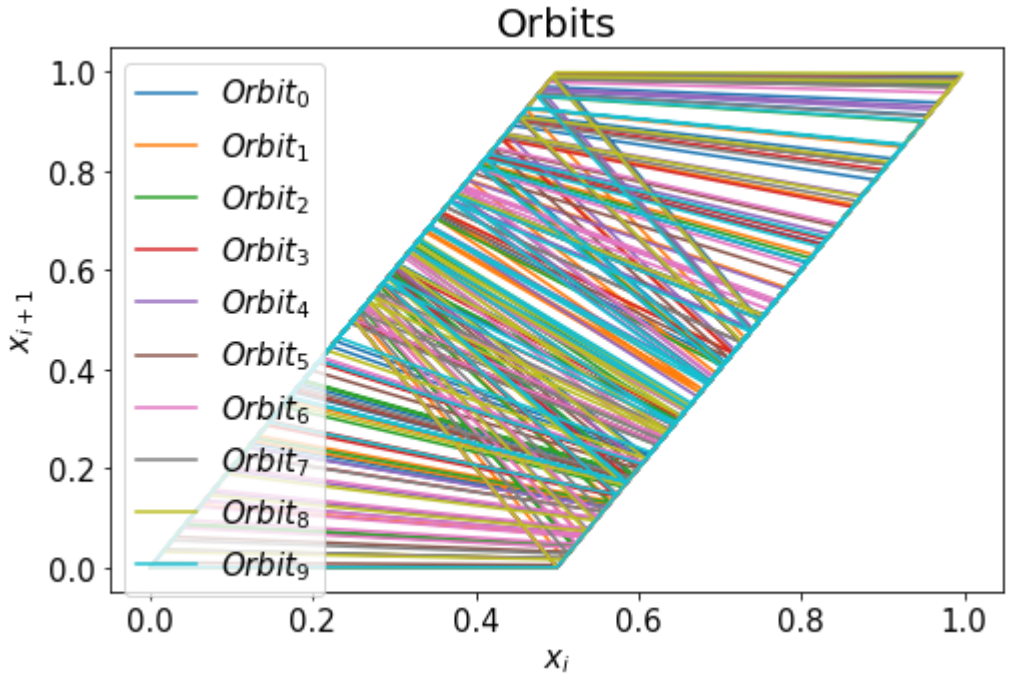
Out[7]:
```
[<matplotlib.lines.Line2D at 0x14fe7bcb648>]
```



2. Do they have a visible pattern? Do all (or almost all) orbits behave in the same way?

Answer:
Yes, it seem that each of the orbit have the same behavior. They all form a rhomboid with a spiderweb inside them. Below there's a plot with 10 orbits, so we can see more clearly the behavior of the orbits.

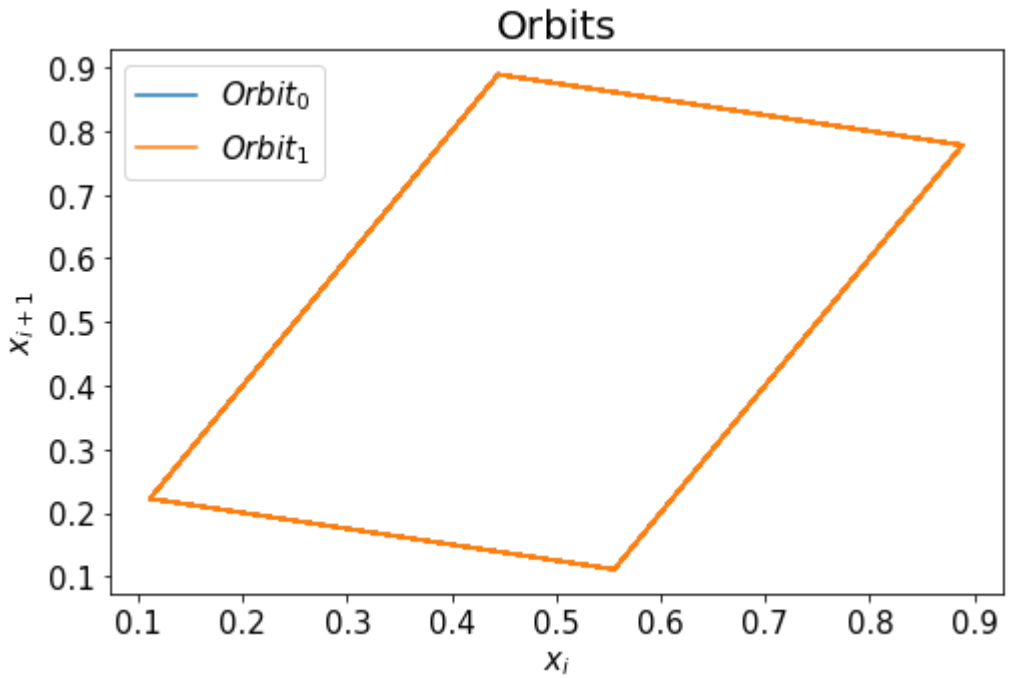
In [8]:
```
plot_orbits(F, 10)
```



3. For the same two discussed cases, instead of treating the seeds a floating point numbers, use

SymPy to do the exact rational arithmetic.
Do the same for the seed $x_0 = \frac{1}{9}$. What do you find for these three cases? Does the computer lie?

In [9]:
```
import sympy as sp


def get_rational_orbit(n, f):
    seed, Y = sp.Rational(1, 9), []
    Y.append(seed)

    for i in range(1, 100):
        Y.append(f(Y[i-1]))
    return Y


def plot_rational_orbits(f, m):
    plt.figure(figsize=(8, 5))
    plt.rc('font', size=15)
    plt.title('Orbits', fontsize=20)
    plt.xlabel('$x_i$')
    plt.ylabel('$x_{i+1}$')

    legends = []

    for k in range(m):
        X = get_rational_orbit(100, f)
        plt.plot([i for i in X][:-1], [i for i in X][1:])
        legends.append(f"$Orbit_{k}$")

    plt.legend(legends, loc=2)
    plt.show()


F_ = lambda x: sp.Rational(2 * x) if 0.0 <= x < 0.5 else sp.Rational(2 * x - 1)
n = 100
Y = get_rational_orbit(n, F_)
plot_rational_orbits(F_, 2)
```



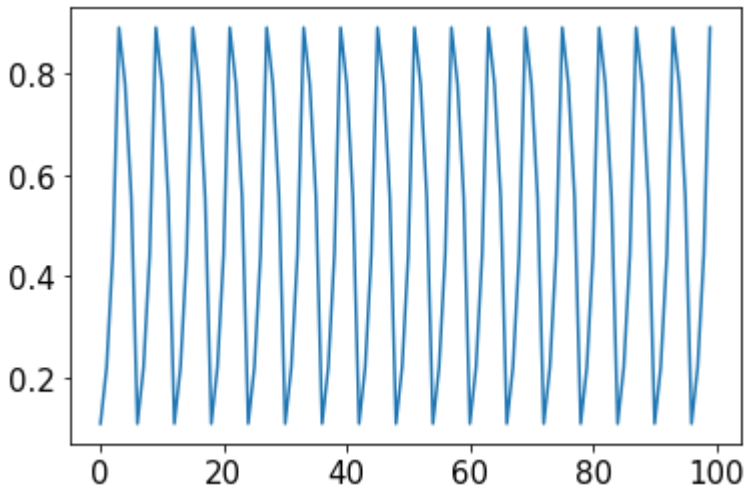
Answer:

I realized how floating points could lead to errors in the interpretations of data, since in the first two cases we end up with a line that around the fifty it decrease to 0. So, one way to deal with that problem is by doing symbolic operations with Sympy. It's not like the computer lie, it's just the fact that by doing floating operations the calculations should be rounded to finite representation, so they handle an error of rounding which could lead to errrs like the difference between the plots on the cases disscussed.

In [10]:
```
print(Y)
plt.plot(Y)
```

```
[1/9, 2/9, 4/9, 8/9, 7/9, 5/9, 1/9, 2/9, 4/9, 8/9, 7/9, 5/9, 1/9, 2/9, 4/9, 8/
9, 7/9, 5/9, 1/9, 2/9, 4/9, 8/9, 7/9, 5/9, 1/9, 2/9, 4/9, 8/9, 7/9, 5/9, 1/9,
2/9, 4/9, 8/9, 7/9, 5/9, 1/9, 2/9, 4/9, 8/9, 7/9, 5/9, 1/9, 2/9, 4/9, 8/9, 7/9,
5/9, 1/9, 2/9, 4/9, 8/9, 7/9, 5/9, 1/9, 2/9, 4/9, 8/9, 7/9, 5/9, 1/9, 2/9, 4/9,
8/9, 7/9, 5/9, 1/9, 2/9, 4/9, 8/9, 7/9, 5/9, 1/9, 2/9, 4/9, 8/9, 7/9, 5/9, 1/9,
2/9, 4/9, 8/9, 7/9, 5/9, 1/9, 2/9, 4/9, 8/9, 7/9, 5/9, 1/9, 2/9, 4/9, 8/9, 7/9,
5/9, 1/9, 2/9, 4/9, 8/9]
```

Out[10]:
```
[<matplotlib.lines.Line2D at 0x14fe7e1e8c8>]
```



In [ ]: