This script presents an example of how to use python signal processing toolbox to calculate the inverse Z-transform of a discrete time signal by using power series expansion method. The code executes a long division (or polynomial division) algorithm to calculate the power series.

In this example, the same expression as the previous example is given:

$$X(z) = \frac{1}{(1-\frac{1}{4}z^{-1})(1-\frac{1}{2}z^{-1})}, |z| > 0.5$$

and we will calculate its inverse Z-transform by power series expansion

In [ ]:

```
# import the necessary libraries
import numpy as np               # for using basic array functions
import matplotlib.pyplot as plt # for this example, it may not be necessary

# the main package for signal processing is called "scipy" and we will use "signal"
import scipy.signal as sgnl
# alternative syntax: from scipy import signal as sgnl
%matplotlib notebook
```

In [ ]:

```
num = np.array([1,0,0])                 # we add zeros to match the size of num and
denum = np.array([1, -3.0/4, 1.0/8])      # coeffs of denum

n, x = sgnl.dimpulse((num, denum, 1),x0=0, n=10)
x
```

This result is interpreted as follows: the output of the $dimpulse$ function is an array of coefficients, starting with the constant term:

$$X(z) = x[0] + x[1]z^{-1} + x[2]z^{-2} + \dots$$

then the inverse transform of a polynomial series is

$$x[n] = [x[0], x[1], x[2], \dots]$$

Observe that;

- The first argument of the dimpulse function is the system coeffs (with a constant 1) as a $tuple$ which means it must be passed in parantheses ().
- The second argument $x0$ is initial state (usually zero)
- The third argument k is the number of points we want to calculate the inverse transform. The outputs;
- n is the sample index vector from 0 to 10 for this example
- x is the inverse transform as an n-dimensional array, so we use the $np.squeeze()$ function to reduce the redundant dimensions.

Note: if we do not reduce the dimensions of the output, the plotting function throws error.

In [ ]:

```
plt.stem(n, np.squeeze(x))
```