This script compares the results of the previous two examples. We obtained the inverse Z-transform of the function

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

in two different ways. The results are

- In partial fraction method:

$$x_{pf}[n] = -(\frac{1}{4})^n u[n] + 2(\frac{1}{2})^n u[n]$$

- In power series expansion method (in the form of impulses):

$$x_{pse}[n] = \delta[n] + 0.75\delta[n-1] + 0.4375\delta[n-2] + \ldots$$

So, to compare these two sequences, we will define and plot them:

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 [ ]:
```
n = np.arange(0,10,1)               # define the index vector for 10 points
xpf  = -(1.0/4)**n + 2*(1.0/2)**n   # result of Ornek-7

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

n1, xpse = sgnl.dimpulse((num, denum, 1),x0=0, n=10)
xpse = np.squeeze(xpse)
```

We have defined the outputs of the examples, now by plotting on the same figure, we will confirm that these two sequences are the same, as expected.

In [ ]:
```
plt.subplot(2,1,1), plt.stem(n, xpf), plt.ylabel('$x_{pf}[n]$')
plt.subplot(2,1,2), plt.stem(n, xpse), plt.ylabel('$x_{pse}[n]$')
plt.xlabel('index vector (sample)')
```

as can be seen in the figure, the samples are the same for the two signals, therefore, we have confirmed that both of the methods result in the same time domain signal, but in different forms.