The provided code analyses and plots a number of signals using Python's NumPy and Matplotlib modules. The analysis tries to investigate the characteristics of each signal, which each reflects a different mathematical function.

(a) Signal: y[n] = cos[n/6]

The signal is assumed to be discrete and represented by the integer values 'n'.

Conclusion: The discrete cosine signal in the provided example has a range of 'n' between -50 and 50. The cosine function is presented throughout a number of intervals in this range, allowing us to see its periodic behaviour. The discontinuous nature of the signal is effectively shown by the stem plot by choosing this range.It does not have a fundamental frequency and so it is printed on the screen

(b) Signal: y[n] = cos[8πn/31]

Assumption: The signal is discrete and represented in terms of integer values 'n'.

Inference: The discrete cosine signal is plotted over the range of 'n' from -50 to 50. This range is selected to capture enough periods of the signal and showcase its periodicity. The stem plot helps visualize the discrete nature of the cosine function. The fundamental frequency is accurately calculated as 1/31 cycles per unit 'n' and is printed.

(c) Signal: y(t) = cos(t/6)

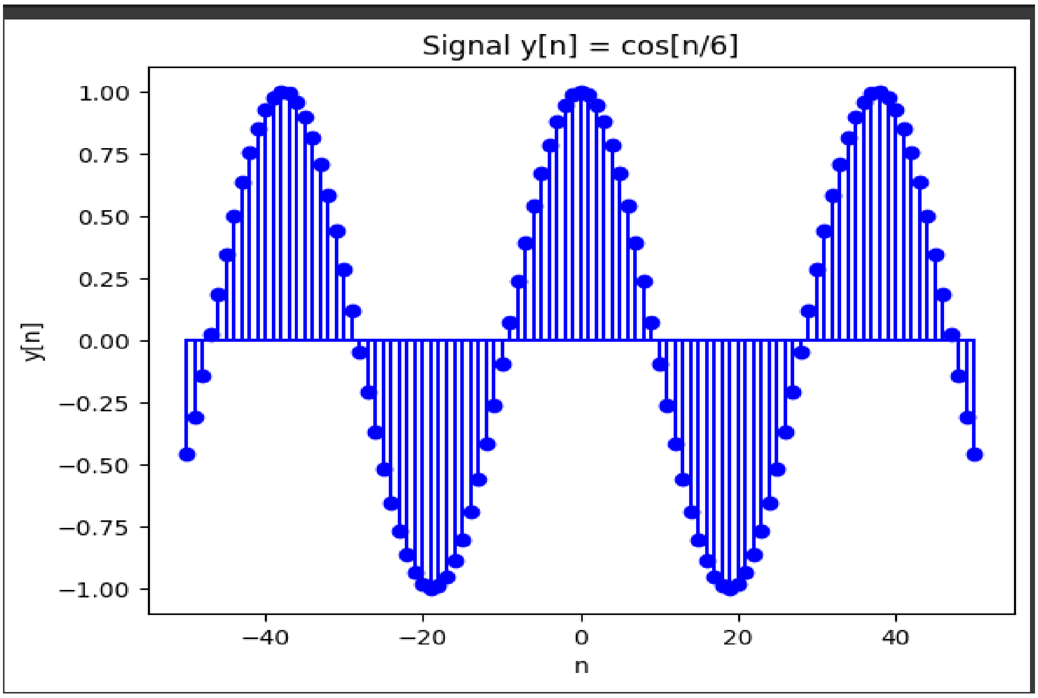
The signal is assumed to be discrete and represented by the integer values 'n'.

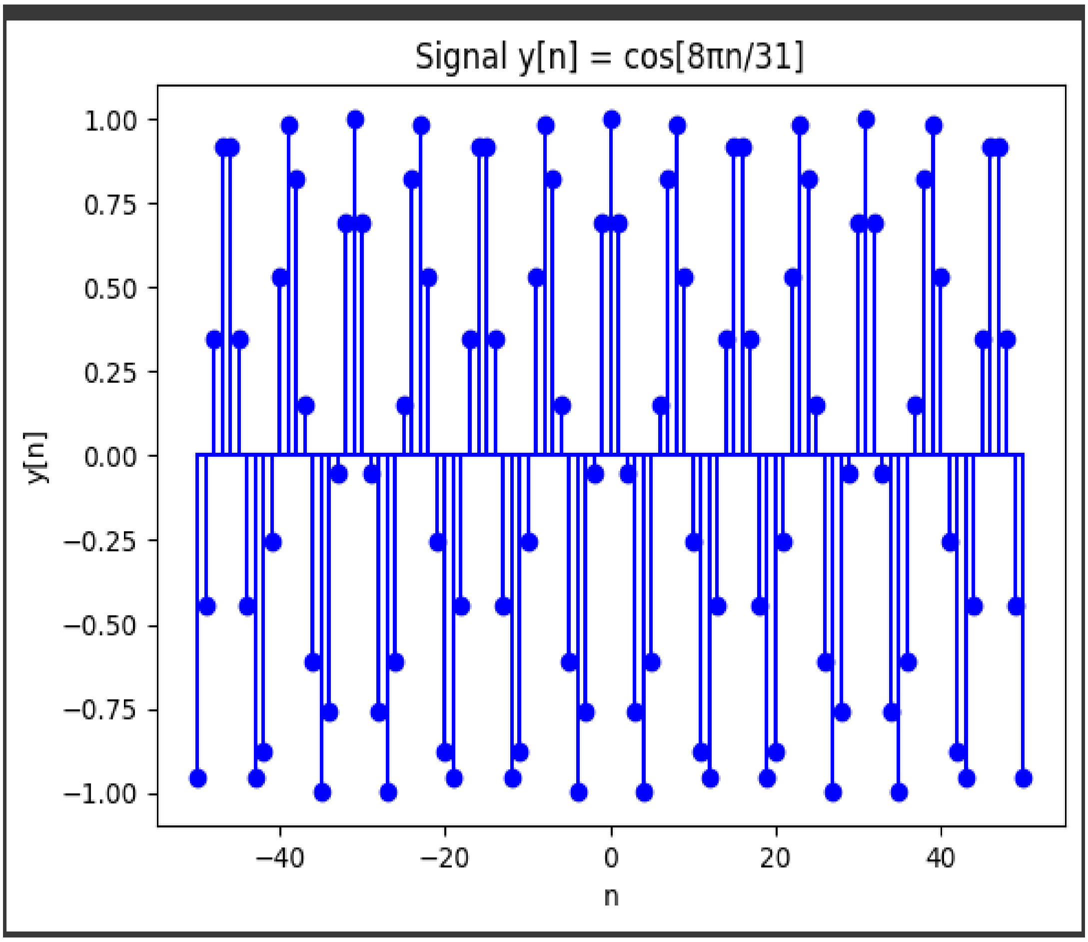
Conclusion: A plot of the discrete cosine signal is shown over the range of 'n' from -50 to 50. This range was chosen to highlight the signal's periodicity and capture sufficient periods of the signal. The stem plot makes the discontinuous aspect of the cosine function easier to understand in the chosen range. It is determined with accuracy that the fundamental frequency is 1/31 cycles per unit 'n', and this value is written.The step size of 0.01 between consecutive values ensures a fine-grained representation of the continuous parameter 't'.It does have a fundamental frequency that is even printed on screen

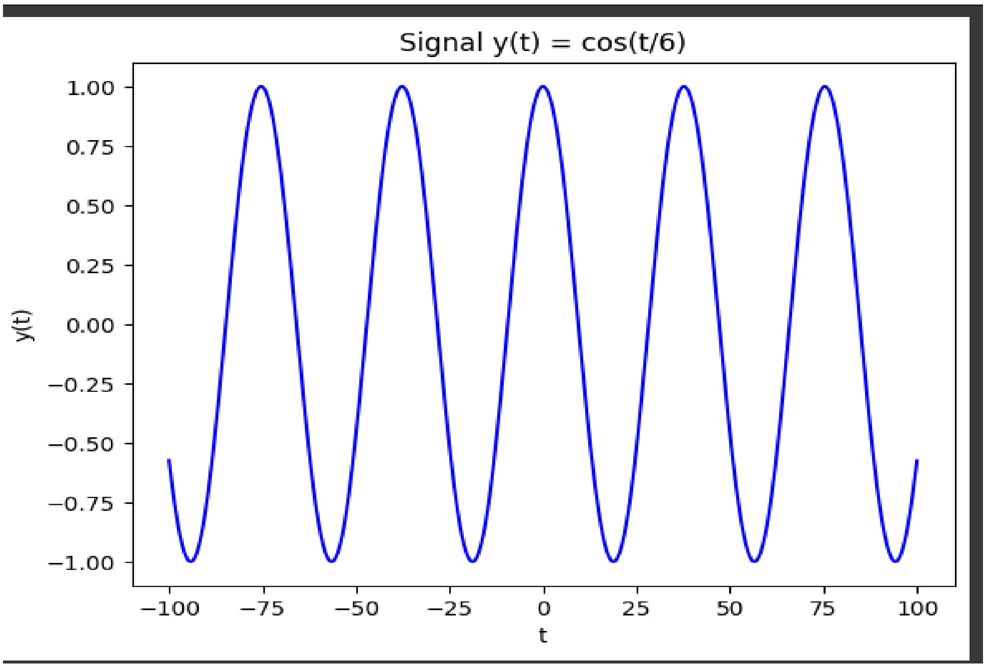
(d) Signal: y(t) = cos(t/6) + sin(2π/3t)

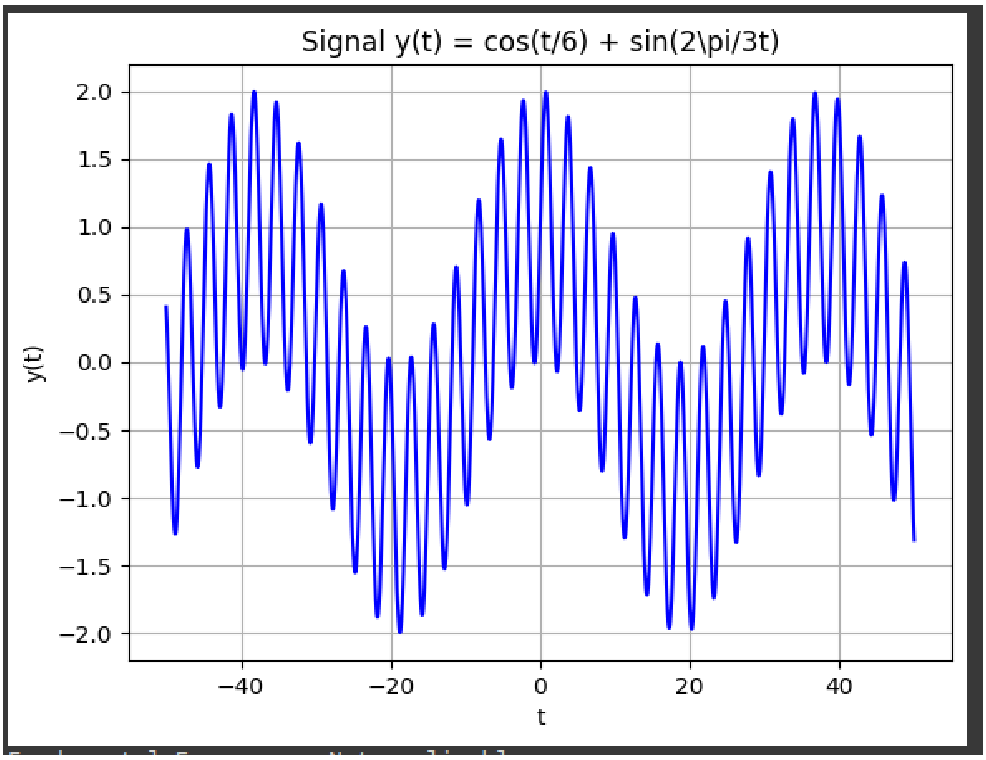
The signal is assumed to be continuous and is represented by the continuous parameter 't'.

Conclusion: The combined signal, which consists of a cosine function and a sine function, is shown over the range of 't' from -50 to 50. This range is chosen so that it is possible to see how both signal components behave and spot any peaks or valleys. The concept of a single fundamental frequency might not be appropriate given the complexity of the combined signal because it doesn't exhibit straightforward periodic behaviour.The step size of 0.01 between consecutive values ensures a fine-grained representation of the continuous parameter 't'.It does not have a fundamental frequency and so it is printed on the screen


Signal y[n] = cos[n/6]
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
y[n]
−40
−20
0
20
40
n


Signal y[n] = cos[8πn/31]
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
y[n]
−40
−20
0
20
40
n


Signal y(t) = cos(t/6)
y(t)
t

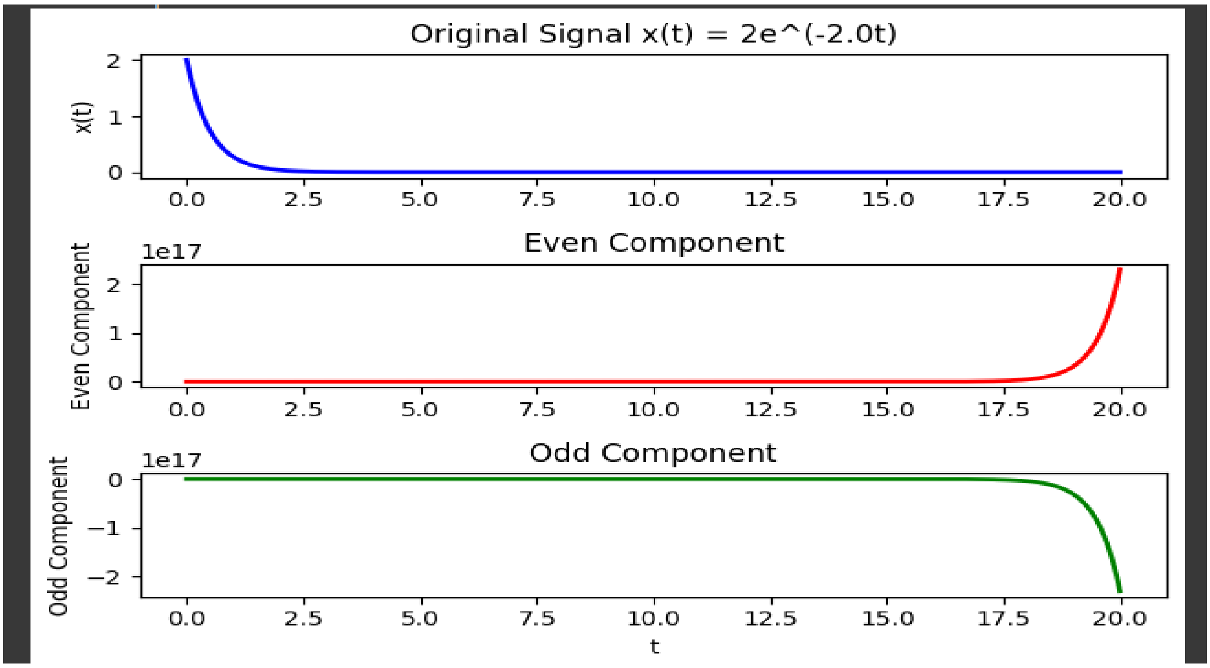

Signal y(t) = cos(t/6) + sin(2\pi/3t)
y(t)
t

Signal decomposition is carried out by the given code utilising Python's NumPy and Matplotlib modules. The goal is to use exponential decay functions to split an original signal into even and odd halves.

Procedure:
User input: To calculate the decay rate, the algorithm asks the user for a "alpha" value.
I have given user a choice to choose alpha but if i had to choose
Alpha it would have been 0.5 as it would have been beat for visualising
the odd and even part of the graph for a users view
Signal construction: A step of 0.01 is used to generate positive time values A range of (0,20) is chosen in it . A thorough representation is ensured by this range.
Original Signal: The exponential decay function is used to generate the initial signal, x_positive, as follows: x_positive = 2 * exp(-alpha * t_positive).
Odd Component: The odd component, x_odd, is calculated as (x_positive - 2 * exp(alpha * t_positive))/2. Even Component: The even component, x_even, is computed as (x_positive + 2 * exp(alpha * t_positive))/2.

Graphical Results
The programme creates a single figure with three subplots that are vertically aligned:
Original Signal: Displayed as a blue line with the title "Original Signal x(t) = 2e(-t)" and the original signal's shape.
Displayed in red, the even component highlights the symmetry. The 'Even Component' is what the title refers to.
Green coloration draws attention to the strange symmetry of the odd component. It is referred to as the "Odd Component" in the title.

Conclusion:
With the use of this analysis, we can show how an original signal can be split into even and odd parts, each of which has a different behaviour. This decomposition process can be clearly understood by the selection of the time range and "alpha" parameter. We can see and comprehend how the components contribute to the original signal thanks to the subplots that result.



Original Signal x(t) = 2e^(-2.0t)
x(t)
Even Component
1e17
Even Component
Odd Component
1e17
Odd Component
t