The chart tool used to draw graphs and charts is part of Forms programming and appears in the Toolbox. It is not yet in WPF but you can add it with an additional Toolkit from Microsoft.

If you are using the professional version of Visual Studio, verify that NuGet has been installed. (Click on Help → About Microsoft Visual Studio and verify that the NuGet Package Manager is in the list of installed programs.) If NuGet is not installed you can download it here: www.nuget.org. If you have Visual Studio updates turned on, NuGet was probably added as an update.

After NuGet is installed click on Tools → Library Package manager → Package Manager Console. The package manager console will appear at the bottom of your screen with a prompt that looks like this: pm>

At the prompt enter the following line and push enter:

`Install-Package WPFToolkit.DataVisualization`

After this is successfully installed you can click on References in the Solution Explorer to see a window that looks something like that shown in Figure 1.

![Figure 1](image.png)

References list after installing Data Visualization Toolkit.

The Data Visualization Toolkit has its own namespace so you must add namespace declarations to the XML code in your project before it becomes available.

```xml
```
**DVC** will be the shorthand name for the charting tools. Build your empty project and the charting tools will become visible in the *intellisense* in the XML code.

As a simple example of creating a line graph enter the following code in XML:

```xml
<Window x:Class="FFTPlotToolKit.MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="MainWindow" Height="350" Width="525">
    <Grid>
        <DVC:Chart Canvas.Top="80" Canvas.Left="10" Name="chtChart1"
            Width="400" Height="250" Background="Beige">
            <DVC:Chart.Series>
                <DVC:LineSeries Title="Sinusoid"
                    IndependentValueBinding="{Binding Path=Key}"
                    DependentValueBinding="{Binding Path=Value}"
                />
            </DVC:Chart.Series>
        </DVC:Chart>
    </Grid>
</Window>
```

We will need a button to start the plot so add a button to your project at the bottom center. You can do this by dragging and dropping a button from the toolbox or you can enter it in XML code. Here is the code from my button which I named btnPlot:

```xml
<Button Content="Plot Data" Height="47"
    HorizontalAlignment="Left" Margin="506,524,0,0"
    Name="btnPlot" VerticalAlignment="Top"
    Width="145" FontSize="24" Click="btnPlot_Click"/>
```

This Button code goes between the `<DVC:Chart>` line and the `</Grid>` line. Note that I have added the Click event for the button also.

For the code behind we need the following *Using directive*:

```csharp
```

In the button click event we want to calculate the values for a sine wave for value of x going from 0 to 4π in 200 increments. We will save the results of the calculations in a List and make that list the source of the chart we are plotting.

Create a new list in the button click event like this:

```csharp
List<KeyValuePair<double, double>> valueList = new List<KeyValuePair<double, double>>(());
```
Next write a loop which allows x to go from 0 to $4\pi$ in steps of $4\pi/200$ and for each value of x calculate a value for $y = \sin(x)$. Put both of these items in the list as doubles.

Finally, assign the list as the source of the chart data. The complete code is shown on the following page. Run the program to get the results shown in Figure 2.

Note that in the project layout, you can click on the chart and get a list of properties which you can change to effect the appearance. These can also be changed in the XAML code.
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Shapes;

namespace FFTPlotToolKit
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
        }

        private void btnPlot_Click(object sender, RoutedEventArgs e)
        {
            List<KeyValuePair<double, double>> valueList = new List<KeyValuePair<double, double>>();
            double x, y, xIncr;
            xIncr = 4*Math.PI/200;
            x = 0; y = Math.Sin(x);
            while (x <= 4*Math.PI)
            {
                valueList.Add(new KeyValuePair<double, double>(x, y));
                x += xIncr;
                y = Math.Sin(x);
            }
            ((LineSeries)chtChart1.Series[0]).ItemsSource = valueList;
        }
    }
}