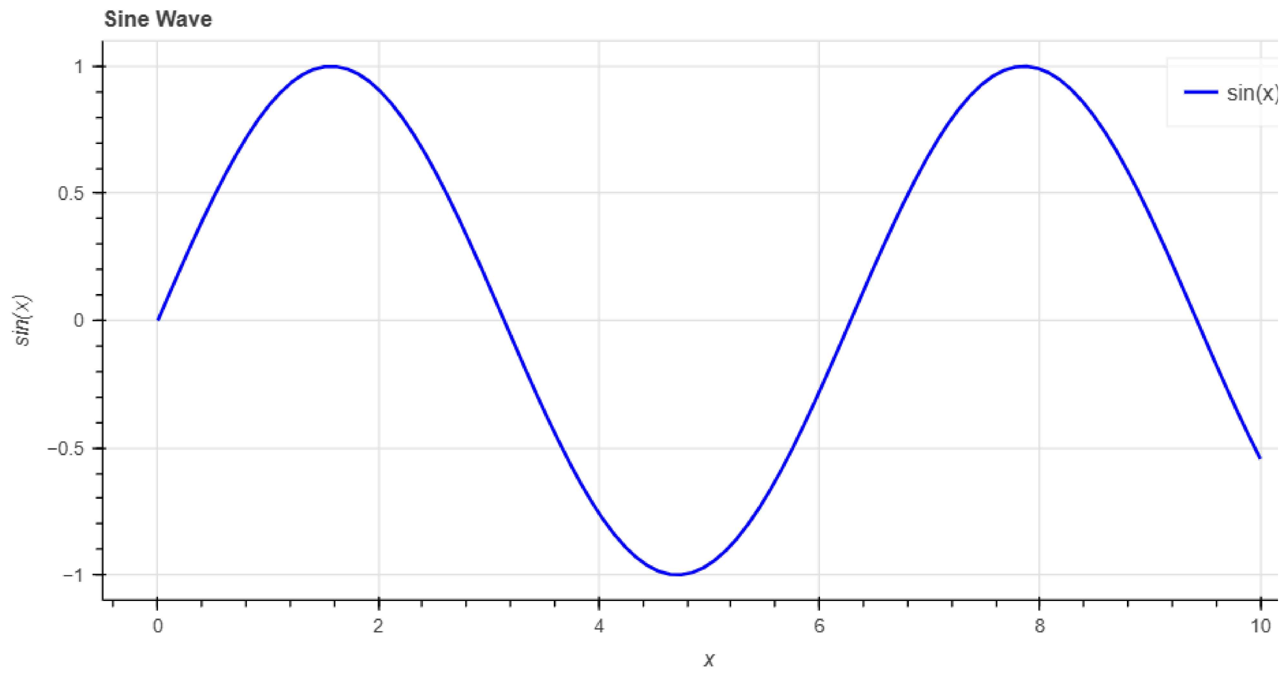
1.Create a Bokeh plot displaying a sine wave. Set x-values from 0 to 10 and y-values as the sine of x

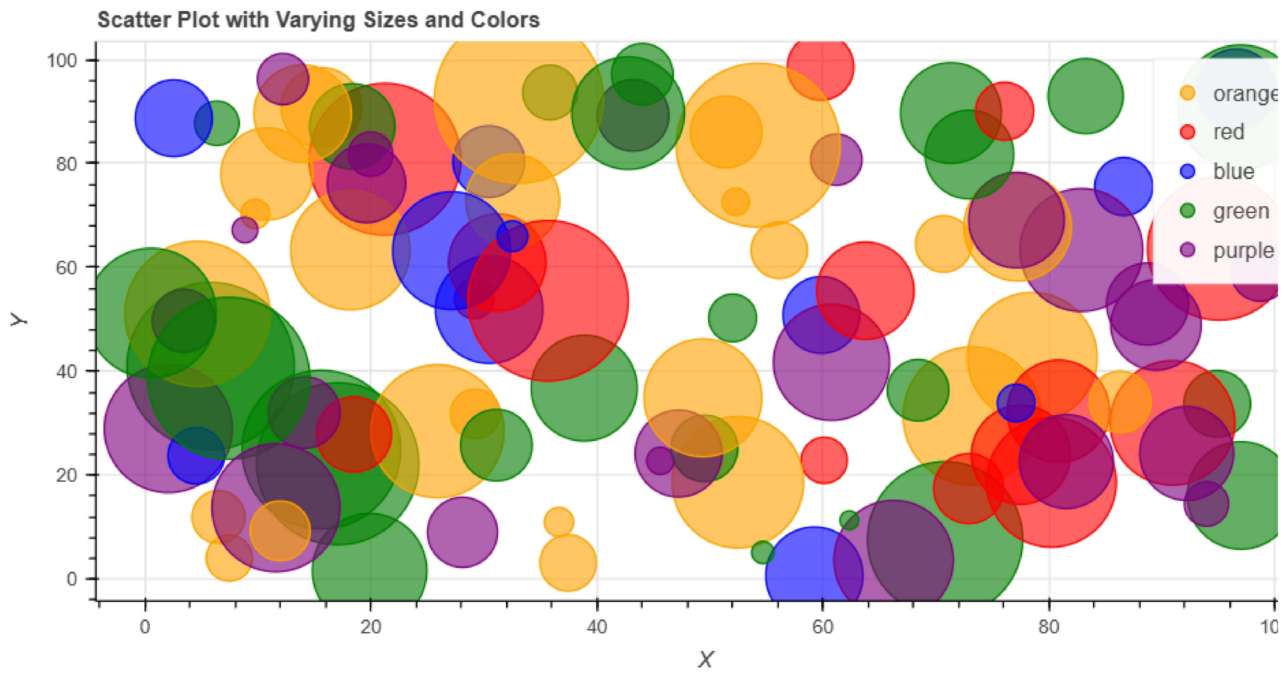
In [3]:
```
from bokeh.plotting import figure, show, output_notebook
import numpy as np

# Output to notebook (if you're using a Jupyter notebook)
output_notebook()

# Generate x values and calculate sine values
x = np.linspace(0, 10, 100)
y = np.sin(x)

# Create a Bokeh figure
p = figure(title="Sine Wave", x_axis_label='x', y_axis_label='sin(x)',width=800,height=400)

# Add a line to the plot
p.line(x, y, line_width=2, color='blue', legend_label='sin(x)')

# Show the plot
show(p)
```



2.Create a Bokeh scatter plot using randomly generated x and y values. Use different sizes and colors for the markers based on the 'sizes' and 'colors' columns

In [5]:
```
from bokeh.plotting import figure, show, output_notebook
from bokeh.io import curdoc
import numpy as np
import pandas as pd

# Output to notebook (if you're using a Jupyter notebook)
output_notebook()

# Generate random data
np.random.seed(42)
n_points = 100
x = np.random.rand(n_points) * 100  # x values between 0 and 100
y = np.random.rand(n_points) * 100  # y values between 0 and 100
sizes = np.random.randint(10, 100, size=n_points)  # Sizes for markers
colors = np.random.choice(['red', 'green', 'blue', 'orange', 'purple'], size=n_points)  # Random colors
```

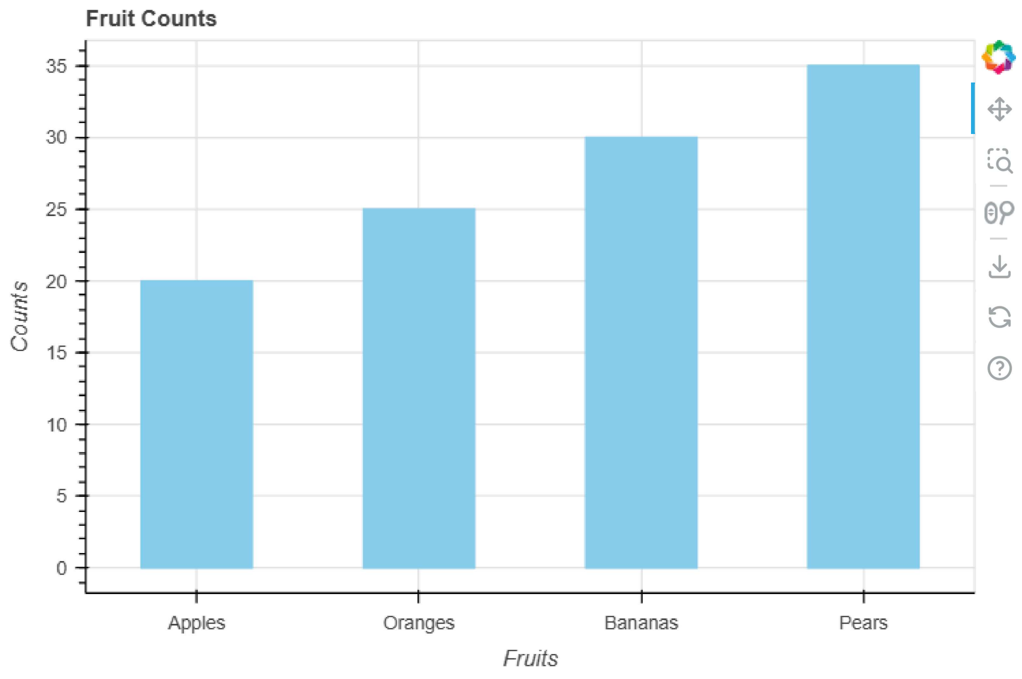
```
# Create a DataFrame
data = pd.DataFrame({'x': x, 'y': y, 'sizes': sizes, 'colors': colors})

# Create a Bokeh figure
p = figure(title="Scatter Plot with Varying Sizes and Colors",
           x_axis_label='X', y_axis_label='Y', width=800, height=400)

# Add a scatter renderer
p.scatter('x', 'y', size='sizes', color='colors', source=data, legend_field='colors', fill_alpha=0.6)

# Show the plot
show(p)
```



1. Generate a Bokeh bar chart representing the counts of different fruits using the following dataset. fruits = ['Apples', 'Oranges', 'Bananas', 'Pears'] counts = [20, 25, 30, 35]

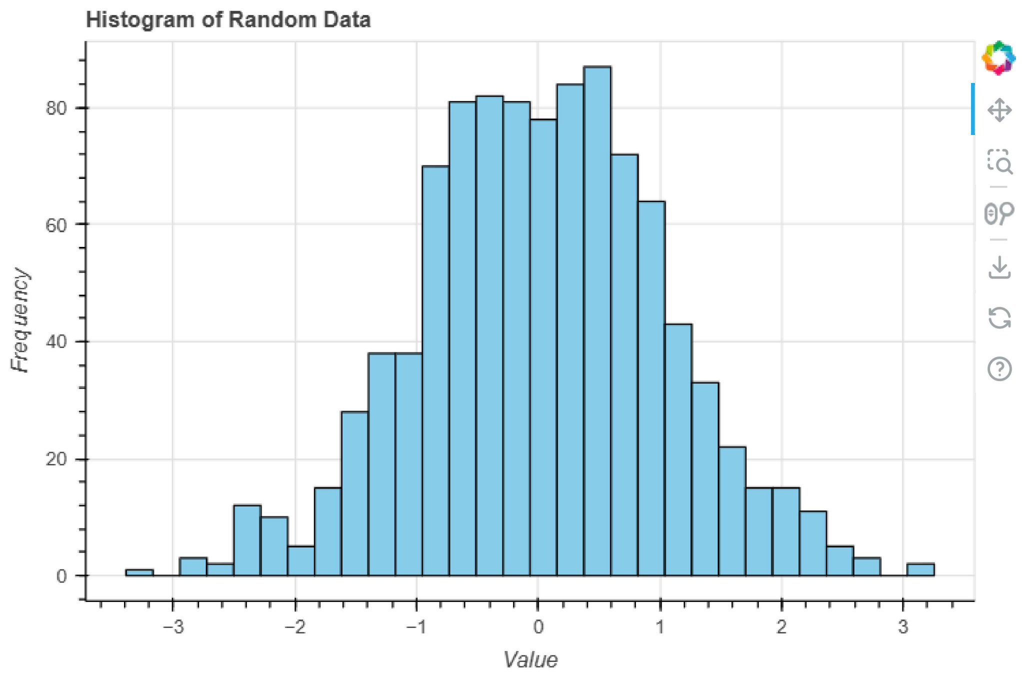
In [9]:
```
from bokeh.plotting import figure, show, output_notebook

# Output to notebook (if you're using a Jupyter notebook)
output_notebook()

# Data for the bar chart
fruits = ['Apples', 'Oranges', 'Bananas', 'Pears']
counts = [20, 25, 30, 35]

# Create a Bokeh figure
p = figure(x_range=fruits, title="Fruit Counts",
           x_axis_label='Fruits', y_axis_label='Counts',
           height=400,width=600)

# Add a vertical bar renderer
p.vbar(x=fruits, top=counts, width=0.5, color='skyblue')

# Adding labels on top of bars
for fruit, count in zip(fruits, counts):
    p.text(fruit, count + 1, text=str(count), text_align='center')

# Show the plot
show(p)
```

```
ERROR:bokeh.core.validation.check:E-1001 (BAD_COLUMN_NAME): Glyph refers to nonexistent column name. Thi
s could either be due to a misspelling or typo, or due to an expected column being missing. : text='30'
[no close matches], x='Bananas' [no close matches] {renderer: GlyphRenderer(id='p1259', ...)}
ERROR:bokeh.core.validation.check:E-1001 (BAD_COLUMN_NAME): Glyph refers to nonexistent column name. Thi
s could either be due to a misspelling or typo, or due to an expected column being missing. : text='25'
[no close matches], x='Oranges' [no close matches] {renderer: GlyphRenderer(id='p1250', ...)}
ERROR:bokeh.core.validation.check:E-1001 (BAD_COLUMN_NAME): Glyph refers to nonexistent column name. Thi
s could either be due to a misspelling or typo, or due to an expected column being missing. : text='20'
[no close matches], x='Apples' [no close matches] {renderer: GlyphRenderer(id='p1241', ...)}
ERROR:bokeh.core.validation.check:E-1001 (BAD_COLUMN_NAME): Glyph refers to nonexistent column name. Thi
s could either be due to a misspelling or typo, or due to an expected column being missing. : text='35'
[no close matches], x='Pears' [no close matches] {renderer: GlyphRenderer(id='p1268', ...)}
```



1. Create a Bokeh histogram to visualize the distribution of the given data. data_hist = np.random.randn(1000) hist, edges = np.histogram(data_hist, bins=30)

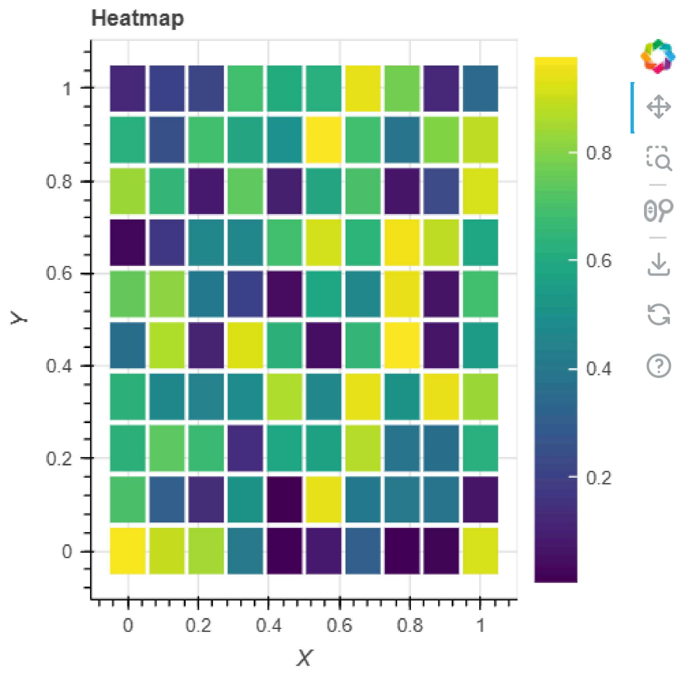
In [11]:
```python
from bokeh.plotting import figure, show, output_notebook
import numpy as np

# Output to notebook (if you're using a Jupyter notebook)
output_notebook()

# Generate random data
data_hist = np.random.randn(1000)

# Create histogram data
hist, edges = np.histogram(data_hist, bins=30)

# Create a Bokeh figure
p = figure(title="Histogram of Random Data",
           x_axis_label='Value', y_axis_label='Frequency',
           height=400, width=600)

# Add a quad renderer for the histogram
p.quad(top=hist, bottom=0, left=edges[:-1], right=edges[1:],
       fill_color='skyblue', line_color='black')

# Show the plot
show(p)
```



1. Create a Bokeh heatmap using the provided dataset. data_heatmap = np.random.rand(10, 10) x = np.linspace(0, 1, 10) y = np.linspace(0, 1, 10) xx, yy = np.meshgrid(x, y)

In [17]:
```
from bokeh.plotting import figure, show, output_notebook
from bokeh.transform import linear_cmap
from bokeh.models import ColorBar, LinearColorMapper
import numpy as np

# Output to notebook (if you're using a Jupyter notebook)
output_notebook()

# Generate random data for the heatmap
data_heatmap = np.random.rand(10, 10)

# Create x and y coordinates
x = np.linspace(0, 1, 10)
y = np.linspace(0, 1, 10)
xx, yy = np.meshgrid(x, y)

# Flatten the data for Bokeh
data_flat = data_heatmap.flatten()
x_flat = xx.flatten()
y_flat = yy.flatten()

# Create a Bokeh figure
p = figure(title="Heatmap",
           x_axis_label='X', y_axis_label='Y',
           height=400, width=400)

# Define the color mapper
mapper = LinearColorMapper(palette="Viridis256", low=data_flat.min(), high=data_flat.max())

# Add rectangles to create the heatmap
p.rect(x='x', y='y', width=0.1, height=0.1, source={'x': x_flat, 'y': y_flat, 'color': data_flat},
        line_color=None, fill_color=linear_cmap('color', mapper.palette,low=data_flat.min(), high=data_f

# Add a color bar
color_bar = ColorBar(color_mapper=mapper, location=(0, 0))
p.add_layout(color_bar, 'right')
```

```
# Show the plot
show(p)
```



```
In [ ]:
```