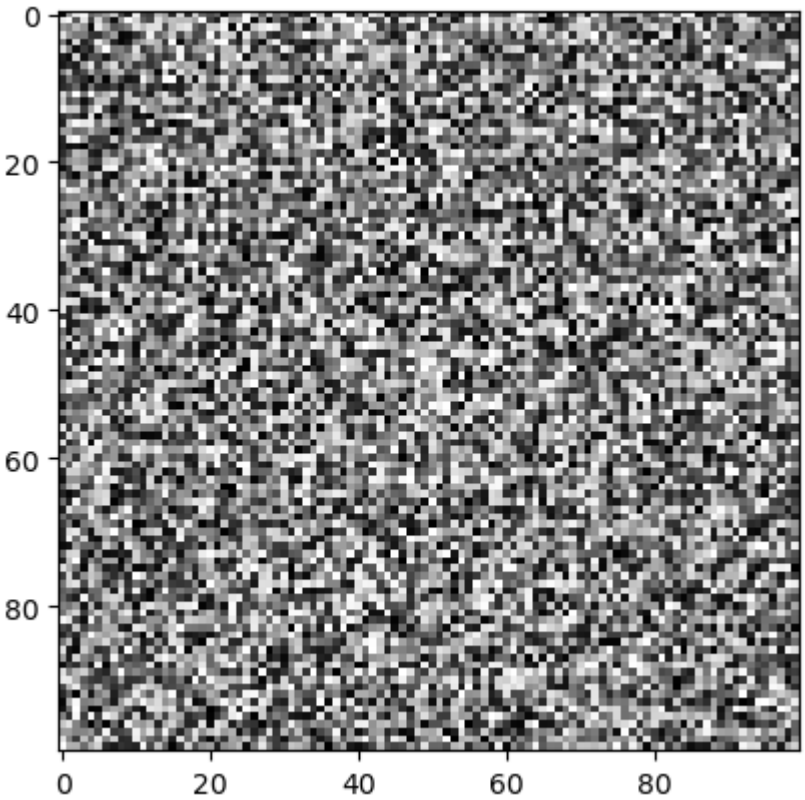
In [1]:
```
pip install matplotlib rasterio
```

```
Requirement already satisfied: six>=1.5 in c:\users\91947\anaconda3\lib
\site-packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
Downloading rasterio-1.3.9-cp311-cp311-win_amd64.whl (23.4 MB)
   ---------------------------------------- 0.0/23.4 MB ? eta -:--:--
   ---------------------------------------- 0.0/23.4 MB 660.6 kB/s eta
0:00:36
   ---------------------------------------- 0.1/23.4 MB 656.4 kB/s eta
0:00:36
   ---------------------------------------- 0.1/23.4 MB 491.5 kB/s eta
0:00:48
   ---------------------------------------- 0.1/23.4 MB 853.3 kB/s eta
0:00:28
   ---------------------------------------- 0.2/23.4 MB 787.7 kB/s eta
0:00:30
   ---------------------------------------- 0.2/23.4 MB 692.4 kB/s eta
0:00:34
   ---------------------------------------- 0.3/23.4 MB 811.7 kB/s eta
0:00:29
    ---------------------------------------- 0.3/23.4 MB 819.2 kB/s eta
0:00:29
```

In [3]:
```
import matplotlib.pyplot as plt
from matplotlib.widgets import RectangleSelector
from matplotlib.patches import Rectangle
import numpy as np

class RasterViewer:
    def __init__(self, raster_data):
        self.raster_data = raster_data
        self.fig, self.ax = plt.subplots()
        self.img = self.ax.imshow(self.raster_data, cmap='gray')
        self.selector = RectangleSelector(self.ax, self.on_select,
                                          drawtype='box', useblit=True,
                                          button=[1], minspanx=5, minspany=
                                          spancoords='pixels', interactive=
        plt.connect('scroll_event', self.on_scroll)
        plt.show()

    def on_select(self, eclick, erelease):
        x1, y1 = int(eclick.xdata), int(eclick.ydata)
        x2, y2 = int(erelease.xdata), int(erelease.ydata)
        selected_region = self.raster_data[y1:y2, x1:x2]
        plt.figure()
        plt.imshow(selected_region, cmap='gray')
        plt.show()

    def on_scroll(self, event):
        if event.button == 'up':
            self.zoom(1.1)
        elif event.button == 'down':
            self.zoom(0.9)

    def zoom(self, scale):
        xlim = self.ax.get_xlim()
        ylim = self.ax.get_ylim()
        x_center = np.mean(xlim)
        y_center = np.mean(ylim)
        new_width = (xlim[1] - xlim[0]) * scale
        new_height = (ylim[1] - ylim[0]) * scale
        self.ax.set_xlim(x_center - new_width / 2, x_center + new_width / 2)
        self.ax.set_ylim(y_center - new_height / 2, y_center + new_height /
        plt.draw()

# Example usage
if __name__ == "__main__":
    # Load raster data (replace this with your own raster data loading mecha
    raster_data = np.random.rand(100, 100)

    # Create and display the raster viewer
    viewer = RasterViewer(raster_data)
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
Cell In[3], line 49
     46 raster_data = np.random.rand(100, 100)
     48 # Create and display the raster viewer
---> 49 viewer = RasterViewer(raster_data)

Cell In[3], line 11, in RasterViewer.__init__(self, raster_data)
      9 self.fig, self.ax = plt.subplots()
     10 self.img = self.ax.imshow(self.raster_data, cmap='gray')
---> 11 self.selector = RectangleSelector(self.ax, self.on_select,
     12                                     drawtype='box', useblit=True,
     13                                     button=[1], minspanx=5, minspan
y=5,
     14                                     spancoords='pixels', interactiv
e=True)
     15 plt.connect('scroll_event', self.on_scroll)
     16 plt.show()

TypeError: RectangleSelector.__init__() got an unexpected keyword argument
'drawtype'
```



In [4]:
```
#q2
```

In [16]:
```
import matplotlib.pyplot as plt
from matplotlib.widgets import RectangleSelector, Dropdown
import numpy as np

class RasterViewer:
    def __init__(self, raster_data):
        self.raster_data = raster_data
        self.num_bands = self.raster_data.shape[2] if len(self.raster_data.s
        self.current_band_indices = list(range(self.num_bands))
        self.fig, self.ax = plt.subplots()
        self.img = self.ax.imshow(self.get_display_data(), cmap='gray')
        self.selector = RectangleSelector(self.ax, self.on_select,
                                          drawtype='box', useblit=True,
                                          button=[1], minspanx=5, minspany=
                                          spancoords='pixels', interactive=
        plt.connect('scroll_event', self.on_scroll)

        self.band_dropdown = Dropdown(plt.axes([0.1, 0.92, 0.1, 0.05]), 'Ban
        self.band_dropdown.on_changed(self.on_band_change)

        plt.show()

    def get_display_data(self):
        return self.raster_data[:, :, self.current_band_indices]

    def update_display(self):
        self.img.set_data(self.get_display_data())
        plt.draw()

    def on_select(self, eclick, erelease):
        x1, y1 = int(eclick.xdata), int(eclick.ydata)
        x2, y2 = int(erelease.xdata), int(erelease.ydata)
        selected_region = self.get_display_data()[y1:y2, x1:x2]
        plt.figure()
        plt.imshow(selected_region, cmap='gray')
        plt.show()

    def on_scroll(self, event):
        if event.button == 'up':
            self.zoom(1.1)
        elif event.button == 'down':
            self.zoom(0.9)

    def zoom(self, scale):
        xlim = self.ax.get_xlim()
        ylim = self.ax.get_ylim()
        x_center = np.mean(xlim)
        y_center = np.mean(ylim)
        new_width = (xlim[1] - xlim[0]) * scale
        new_height = (ylim[1] - ylim[0]) * scale
        self.ax.set_xlim(x_center - new_width / 2, x_center + new_width / 2)
        self.ax.set_ylim(y_center - new_height / 2, y_center + new_height /
        plt.draw()

    def on_band_change(self, index):
        self.current_band_indices = [index]
        self.update_display()

# Example usage
if __name__ == "__main__":
    # Load raster data (replace this with your own raster data loading mecha
```

```
    # Assuming raster_data is a 3D numpy array (height x width x bands)
    raster_data = np.random.rand(100, 100, 3)  # Example random 3-band raste

    # Create and display the raster viewer
    viewer = RasterViewer(raster_data)
```

```
---------------------------------------------------------------------------
-
ImportError                               Traceback (most recent call las
t)
Cell In[16], line 2
      1 import matplotlib.pyplot as plt
----> 2 from matplotlib.widgets import RectangleSelector, Dropdown
      3 import numpy as np
      5 class RasterViewer:

ImportError: cannot import name 'Dropdown' from 'matplotlib.widgets' (C:\U
sers\91947\anaconda3\Lib\site-packages\matplotlib\widgets.py)
```

In [ ]:
```
#q3
```

In [15]:
```
import geopandas as gpd
import matplotlib.pyplot as plt

# Load shapefile
def load_shapefile(file_path):
    return gpd.read_file(file_path)

# Generate charts for attribute data
def generate_charts(data):
    # Example: Create a bar chart for a categorical attribute
    attribute_counts = data['attribute_column'].value_counts()
    attribute_counts.plot(kind='bar')
    plt.title('Attribute Data Distribution')
    plt.xlabel('Attribute Values')
    plt.ylabel('Frequency')
    plt.show()

# Interactive exploration (optional)
def interactive_exploration(data):
    # Example: Allow users to select attribute column interactively
    attribute_column = input("Enter attribute column name: ")
    if attribute_column in data.columns:
        generate_charts(data[[attribute_column]])
    else:
        print("Invalid attribute column name!")
```

In [11]:
```
pip install geopandas
```

```
     ---------------------- ---------------- 30.7/51.1 kB ? eta -:--:--
     ---------------------------------------- 51.1/51.1 kB 375.0 kB/s eta
0:00:00
Requirement already satisfied: packaging in c:\users\91947\anaconda3\lib
\site-packages (from geopandas) (23.0)
Requirement already satisfied: pandas>=1.4.0 in c:\users\91947\anaconda3
\lib\site-packages (from geopandas) (1.5.3)
Collecting pyproj>=3.3.0 (from geopandas)
  Obtaining dependency information for pyproj>=3.3.0 from https://files.
pythonhosted.org/packages/79/95/eb68113c5b5737c342bde1bab92705dabe69c162
99c5a122616e50f1fbd6/pyproj-3.6.1-cp311-cp311-win_amd64.whl.metadata (ht
tps://files.pythonhosted.org/packages/79/95/eb68113c5b5737c342bde1bab927
05dabe69c16299c5a122616e50f1fbd6/pyproj-3.6.1-cp311-cp311-win_amd64.whl.
metadata)
  Downloading pyproj-3.6.1-cp311-cp311-win_amd64.whl.metadata (31 kB)
Collecting shapely>=1.8.0 (from geopandas)
  Obtaining dependency information for shapely>=1.8.0 from https://file
s.pythonhosted.org/packages/9e/39/029c441d8af32ab423b229c4525ce5ce670731
8155b59634811a4c56f5c4/shapely-2.0.2-cp311-cp311-win_amd64.whl.metadata
(https://files.pythonhosted.org/packages/9e/39/029c441d8af32ab423b229c45
```

In [ ]:
```
#q4
```

In [14]:
```
import rasterio
import numpy as np

# Load raster data into NumPy array
def load_raster(file_path):
    with rasterio.open(file_path) as src:
        raster_array = src.read()  # Read raster data into NumPy array
    return raster_array

# Perform mathematical transformation on raster data
def transform_raster(raster_array):
    # Example: Normalize raster data
    min_val = np.min(raster_array)
    max_val = np.max(raster_array)
    normalized_array = (raster_array - min_val) / (max_val - min_val)
    return normalized_array

# Filter raster data using convolution
def filter_raster(raster_array):
    # Example: Apply a simple averaging filter
    kernel = np.ones((3, 3)) / 9  # 3x3 averaging kernel
    filtered_array = np.zeros_like(raster_array)
    for i in range(raster_array.shape[0]):
        filtered_array[i] = np.convolve(raster_array[i], kernel, mode='same'
    return filtered_array

# Perform statistical analysis on raster data
def analyze_raster(raster_array):
    # Example: Calculate mean and standard deviation
    mean_val = np.mean(raster_array)
    std_dev = np.std(raster_array)
    return mean_val, std_dev
```

In [ ]:
```
```