```
# 1. Pixel Based Visulization Techniques

import numpy as np
import matplotlib.pyplot as plt

# Generate some example data
data = np.random.rand(10, 10)  # Replace this with your own data


# Create a heatmap using matplotlib
plt.imshow(data, cmap='viridis', interpolation='nearest')

# Add colorbar for reference
plt.colorbar()

# Add labels and title
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Pixel-based Heatmap')

# Show the plot
plt.show()
```

2. Geometric Based Visualization Techniques:

```
import matplotlib.pyplot as plt
import numpy as np

# Generate sample data
x = np.random.rand(50)
y = np.random.rand(50)

# Create a scatter plot
plt.scatter(x, y)
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Scatter Plot')
plt.show()

import matplotlib.pyplot as plt
import numpy as np

# Generate sample data
x = np.random.rand(50)
y = np.random.rand(50)
sizes = np.random.rand(50) * 100  # Sizes of the bubbles
```

```
# Create a bubble chart
plt.scatter(x, y, s=sizes, alpha=0.5)
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Bubble Chart')
plt.show()

import matplotlib.pyplot as plt
import numpy as np

# Generate sample data
x = np.random.randn(1000)
y = np.random.randn(1000)

# Create a hexbin plot
plt.hexbin(x, y, gridsize=20, cmap='inferno')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Hexbin Plot')
plt.colorbar(label='Counts')
plt.show()
'''
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
#import plotly as plt
# Generate sample data
categories = ['A', 'B', 'C', 'D']
data = [np.random.randn(100) for _ in categories]

# Create a violin plot
sns.violinplot(data=data)
plt.xlabel('Categories')
plt.ylabel('Values')
plt.title('Violin Plot')
plt.show()
```

3. Icon-based visualization techniques

```
# Icon-based visualization techniques involve representing data using icons or
pictorial symbols. These icons can convey information more intuitively and
creatively compared to traditional graphs. Here are a few examples of how you
can implement icon-based visualization techniques using Python
#Icon Arrays:
#An icon array represents data using a grid of icons,
#where each icon represents a certain number of data points.
```

```
import matplotlib.pyplot as plt
import numpy as np
'''
# Generate sample data
data = np.random.randint(0, 3, size=(5, 5))  # Replace with your data

# Create an icon array
plt.imshow(data, cmap='Blues', vmin=0, vmax=3)

# Add colorbar for reference
plt.colorbar(ticks=[0, 1, 2, 3], label='Data Value')

# Remove x and y ticks
plt.xticks([])
plt.yticks([])

plt.title('Icon Array Visualization')
plt.show()
'''
'''
Emoji-based Visualization:
You can use emojis to represent different categories or states in your data.
python
Copy code
'''
'''
import matplotlib.pyplot as plt

# Example data
categories = ['☀', '🌧', '🌬', '❄']
counts = [25, 15, 10, 5]

# Create a bar chart using emojis
plt.bar(categories, counts)
plt.xlabel('Weather Category')
plt.ylabel('Counts')
plt.title('Emoji-based Visualization')
plt.show()
'''
'''
Icon Mapping with Geographical Data:
You can use icons to represent specific features on a map, such as landmarks,
cities, or points of interest.
python
Copy code
'''
'''
import geopandas as gpd
import matplotlib.pyplot as plt
```

```
# Load geospatial data
world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))

# Plot world map with city markers
ax = world.plot(figsize=(10, 6))
cities = gpd.read_file(gpd.datasets.get_path('naturalearth_cities'))
cities.plot(ax=ax, marker='o', color='red', markersize=50)
plt.title('Icon Mapping with Geographical Data')
plt.show()

'''
'''
Glyphs and Icon Fonts:
You can use icon fonts like Font Awesome or Material Icons to incorporate
icon-based visualization directly into text elements.
python
Copy code
'''
import matplotlib.pyplot as plt
from matplotlib.text import TextPath
from matplotlib.patches import PathPatch

# Create a glyph using a text path
#glyph = TextPath((0, 0), "🌟", size=100, prop='emoji', usetex=False)
glyph = TextPath((0, 0), "🌟", size=100, prop='emoji', usetex=False)
# Create a figure and axis
# Convert the glyph to a patch
patch = PathPatch(glyph, color='blue')

# fig, ax = plt.subplots(figsize=(4, 4))

# Create a figure and axis
fig, ax = plt.subplots(figsize=(4, 4))
ax.add_patch(patch)
# ax.add_patch(glyph,color= 'blue')

# Set axis limits
ax.set_xlim(-100, 100)
ax.set_ylim(-100, 100)

plt.axis('off')
plt.title('Glyph-based Visualization')
plt.show()
# These are just a few examples of how you can implement icon-based
visualization techniques using Python. The specific technique you choose will
depend on your data, context, and the type of insights you want to convey.

'''
```

```
'''
```

#4. Hierarchical-based visualization techniques

```
'''
Hierarchical-based visualization techniques
 involve representing data in a hierarchical structure,
 often in the form of trees or nested relationships.
 These techniques are useful for visualizing data with
 parent-child relationships or hierarchical structures.
   Here are a few examples of hierarchical-based visualization
     techniques you can implement using Python:

Tree Diagram with matplotlib and networkx:
'''
'''
import networkx as nx
import matplotlib.pyplot as plt

# Create a sample hierarchical data structure
G = nx.DiGraph()
G.add_edges_from([(1, 2), (1, 3), (2, 4), (2, 5), (3, 6)])

# Create a tree diagram
pos = nx.spring_layout(G)
nx.draw(G, pos, with_labels=True, node_size=2000, node_color="skyblue",
font_size=10, font_color="black")

plt.title('Tree Diagram')
plt.show()

'''
'''Sunburst Chart with plotly.express:'''
# Sunburst Chart with plotly.express:
import plotly.express as px

# Create a sample hierarchical data structure
data = {
    'id': ['root', 'A', 'B', 'C', 'D'],
    'parent': ['', 'root', 'root', 'root', 'B'],
    'value': [100, 40, 20, 30, 10]
}

# Create a sunburst chart
fig = px.sunburst(data, names='id', parents='parent', values='value')
```

```
fig.update_layout(title='Sunburst Chart')
fig.show()
```