Data Visualization in Python (Part II)

Before we get Started:

let's first install the necessary libraries (if required)

In [1]:
```
#!pip install matplotlib
#!pip install seaborn
#!pip install pandas
#!pip install squarify
#!pip install plotly
```

Step 1: Import the necessary libraries

First, we need to import the necessary libraries into our Python script.

In [2]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
import matplotlib.cm as cm
import plotly.express as px
import squarify
```

Step 2: Load the data

Next, we need to load the data that we want to visualize. For this tutorial, we will be using the Iris dataset, which is a popular dataset for data visualization and machine learning.

In [3]:
```
iris = sns.load_dataset('iris') # Loading the dataset
df = pd.DataFrame(iris) # Storing the dataframe into Pandas
```

In [4]:
```
df.head()
```

Out[4]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

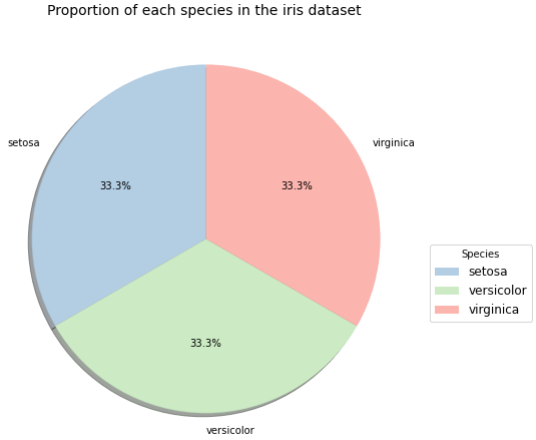
In [5]:
```
df['species'] = df['species'].astype('category')
```

Step 3: Create a pie chart with customization

A pie chart is a circular chart that shows the proportion of each category in a dataset. The whole circle represents the total amount and each slice represents a category. The size of each slice corresponds to the proportion of that category in the dataset.

Let's use the iris dataset to create a pie chart that shows the proportion of each species:

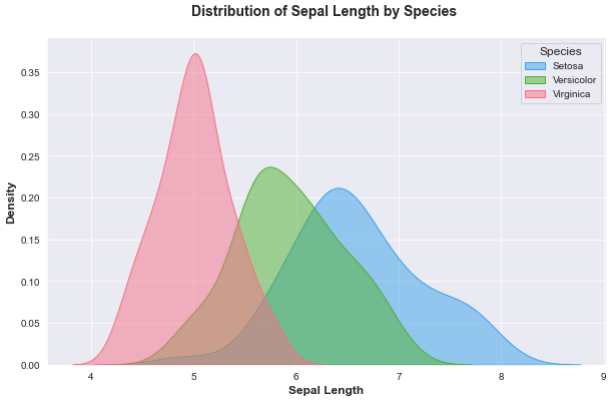
In [6]:
```
# Get the counts for each species
sizes = df['species'].value_counts()
labels = sizes.index

# Choose colors for the chart
colors = cm.Pastel1([0.2, 0.3, 0.1])

# Set the figure size
plt.figure(figsize=(10, 8))

# Create the pie chart with customizations
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90, colors=colors, shadow=True)

# Add a title
plt.title('Proportion of each species in the iris dataset ', fontsize=14, loc='center')

# Add a legend
plt.legend(title='Species', loc='best', bbox_to_anchor=(1, 0.5), fontsize=12)

# Increase the tick label size
plt.tick_params(labelsize=12)

# Display the chart
plt.show()
```



Step 4: Create a area plot with customization

An area plot can show how different variables change over time or across categories. In this case, we can use it to show the distribution of sepal length for each species of Iris. We'll use seaborn for this:

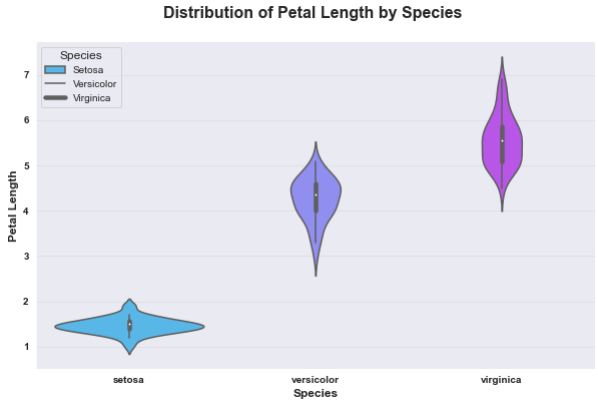
In [7]:
```
# Set style and color palette
sns.set_style("darkgrid")
sns.set_palette("husl", 3)

# Create the area plot
ax = sns.kdeplot(data=df, x="sepal_length", hue="species", fill=True, alpha=.5)

# Add title and axis labels
ax.set_title('Distribution of Sepal Length by Species \n', fontsize=14, fontweight='bold')
ax.set_xlabel('Sepal Length', fontsize=12, fontweight='bold')
ax.set_ylabel('Density', fontsize=12, fontweight='bold')

# Adjust legend
ax.legend(title='Species', labels=['Setosa', 'Versicolor', 'Virginica'],
          fontsize=10, title_fontsize=12, loc='upper right', frameon=True)

# Add horizontal grid lines
ax.yaxis.grid(True)

# Remove top and right spines
sns.despine(top=True, right=True)

# Set figure size and save plot
plt.gcf().set_size_inches(10, 6)
# plt.savefig('area_plot_customized.png', dpi=300)

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



Step 5: Create a violin plot

A violin plot is similar to a box plot, but also shows the density of the data at different values. We can use it to show the distribution of petal length for each species of Iris. We'll use seaborn again:

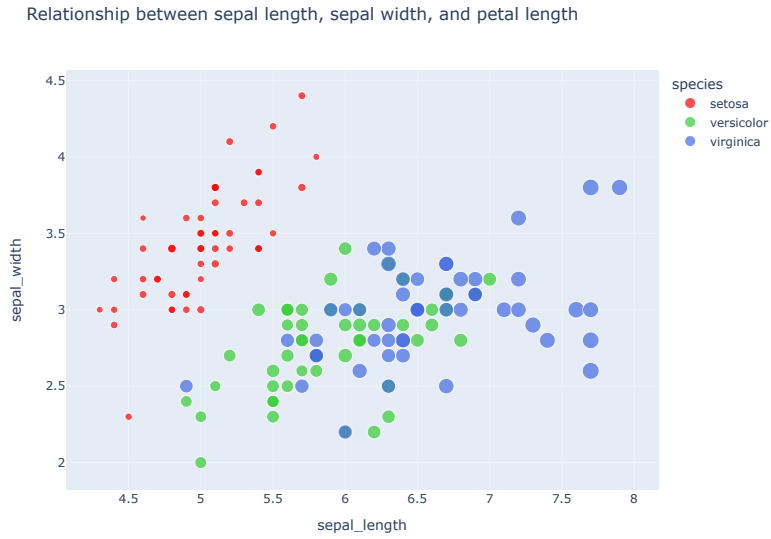
In [8]:
```
# Create violin plot
fig, ax = plt.subplots(figsize=(10, 6))
sns.violinplot(data=iris, x='species', y='petal_length', palette='cool', width=0.8, ax=ax)

# Set title and axis labels
plt.title('Distribution of Petal Length by Species \n', fontsize=16, fontweight='bold')
plt.xlabel('Species', fontsize=12, fontweight='bold')
plt.ylabel('Petal Length', fontsize=12, fontweight='bold')

# Customize tick labels and font sizes
plt.xticks(fontsize=10, fontweight='bold')
plt.yticks(fontsize=10, fontweight='bold')

# Remove spines and set grid
sns.despine()
ax.yaxis.grid(True, linestyle='-', which='major', color='lightgrey', alpha=0.5)

# Adjust violin plot elements
for i, artist in enumerate(ax.artists):
    # Set the linecolor on the violin's edges
    artist.set_edgecolor('black')

    # Set the fill color for the violin's body
    artist.set_facecolor('lightblue')

    # Iterate over the child objects (i.e., the individual points inside each violin)
    for j, child in enumerate(artist.get_children()):
        # Set the marker style and size for each point
        child.set_markeredgecolor('black')
        child.set_markerfacecolor('white')
        child.set_markersize(7)

        # Set the alpha value for each point
        child.set_alpha(0.7)

# Adjust legend
ax.legend(title='Species', labels=['Setosa', 'Versicolor', 'Virginica'],
          fontsize=10, title_fontsize=12, loc='upper left', frameon=True)

# Show plot
plt.show()
```



Step 6: Create a bubble chart

A bubble chart can show the relationship between three variables. In this case, we can use it to show the relationship between sepal length, sepal width, and petal length for each species of Iris. We'll use plotly for this:

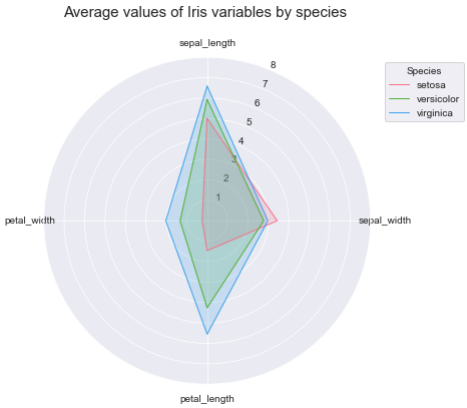
In [9]:
```
fig = px.scatter(df, x='sepal_length', y='sepal_width', size='petal_length', color='species',
                 hover_name='species', title='Relationship between sepal length, sepal width, and petal length',
                 size_max=12, color_discrete_sequence=['#FF0000', '#32CD32', '#4169E1'],width=800, height=600)
fig.show()
```



Step 8: radar chart

A radar chart, also known as a spider chart, can show how different variables compare across categories. In this case, we can use it to show the average values of the four Iris variables for each species. We'll use matplotlib for this:

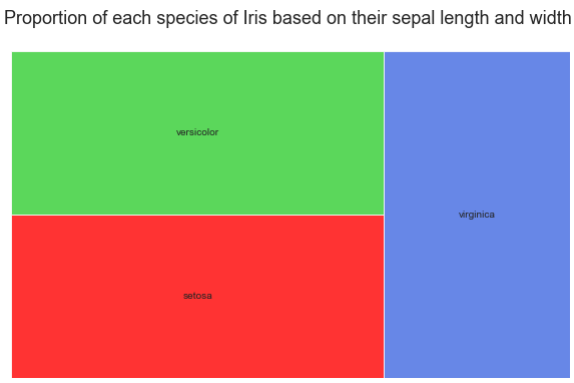
In [10]:
```
# Get the variables and angles
variables = df.columns[:-1]
angles = np.linspace(0, 2*np.pi, len(variables), endpoint=False)
angles = np.concatenate((angles, [angles[0]]))

# Set up the polar plot
fig, ax = plt.subplots(subplot_kw={'projection': 'polar'},figsize=(10, 6))
ax.set_theta_offset(np.pi/2)
ax.set_theta_direction(-1)
ax.set_ylim(0, 8)

# Plot the data for each species
for i, species in enumerate(df['species'].unique()):
    data = df[df['species'] == species][variables].mean().values
    values = np.concatenate((data, [data[0]]))
    ax.plot(angles, values, linewidth=1, linestyle='solid', label=species)
    ax.fill(angles, values, alpha=0.2)

# Set the labels for the axes
ax.set_xticks(angles[0:4])
ax.set_xticklabels(variables)

# Add a title and legend
plt.title('Average values of Iris variables by species \n', fontsize = 15)
plt.legend(title='Species', loc='upper right', bbox_to_anchor=(1.3, 1))

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



Step 8: Create a treemap

A treemap can show hierarchical data using nested rectangles. In this case, we can use it to show the proportion of each species of Iris based on their sepal length and width. We'll use squarify and matplotlib for this:

In [11]:
```
# Group data by species and calculate mean values for sepal length and width
sizes = iris.groupby('species')[['sepal_length', 'sepal_width']].mean()

# Add a column for the count of each species
sizes['count'] = iris['species'].value_counts()

# Reset index to use columns for plotting
sizes.reset_index(inplace=True)

# Create a figure and plot the treemap
plt.figure(figsize=(10, 6))
squarify.plot(sizes=sizes['count'], label=sizes['species'],
              color=['#FF0000', '#32CD32', '#4169E1'], alpha=.8)
plt.title('Proportion of each species of Iris based on their sepal length and width \n', fontsize=18)
plt.axis('off')
plt.show()
```



Conclusion

- In this tutorial, we covered the basics of data visualization in Python using the Matplotlib and Seaborn libraries.
- We showed how to create pie chart, area plot, violin plot, bubble chart, radar chart, treemap using Python code.
- By following these steps, you should now have a good understanding of how to create various types of plots in Python for data analysis and visualization.
- **Follow Muhammad Bilal Alam for more tutorials like this**