Task 1- Dataframe(CSV File)

In [1]:
```
import pandas as pd
import seaborn
```

```
c:\Users\HP\anaconda3\Lib\site-packages\pandas\core\arrays\masked.py:61: UserWarni
ng: Pandas requires version '1.3.6' or newer of 'bottleneck' (version '1.3.5' curr
ently installed).
  from pandas.core import (
c:\Users\HP\anaconda3\Lib\site-packages\seaborn\_statistics.py:32: UserWarning: A
NumPy version >=1.25.2 and <2.6.0 is required for this version of SciPy (detected
version 1.24.4)
  from scipy.stats import gaussian_kde
```

In [2]:
```
data = pd.read_csv("supermarket_sales.csv")
data.head(10)
```

Out[2]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80.2200
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880	489.0480
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085	634.3785
5	699-14-3026	C	Naypyitaw	Normal	Male	Electronic accessories	85.39	7	29.8865	627.6165
6	355-53-5943	A	Yangon	Member	Female	Electronic accessories	68.84	6	20.6520	433.6920
7	315-22-5665	C	Naypyitaw	Normal	Female	Home and lifestyle	73.56	10	36.7800	772.3800
8	665-32-9167	A	Yangon	Member	Female	Health and beauty	36.26	2	3.6260	76.1460
9	692-92-5582	B	Mandalay	Member	Female	Food and beverages	54.84	3	8.2260	172.7460

In [3]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 17 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Invoice ID               1000 non-null   object 
 1   Branch                   1000 non-null   object 
 2   City                     1000 non-null   object 
 3   Customer type            1000 non-null   object 
 4   Gender                   1000 non-null   object 
 5   Product line             1000 non-null   object 
 6   Unit price               1000 non-null   float64
 7   Quantity                 1000 non-null   int64  
 8   Tax 5%                   1000 non-null   float64
 9   Total                    1000 non-null   float64
 10  Date                     1000 non-null   object 
 11  Time                     1000 non-null   object 
 12  Payment                  1000 non-null   object 
 13  cogs                     1000 non-null   float64
 14  gross margin percentage  1000 non-null   float64
 15  gross income             1000 non-null   float64
 16  Rating                   1000 non-null   float64
dtypes: float64(7), int64(1), object(9)
memory usage: 132.9+ KB
```

In [4]:
```
selected_columns = data.iloc[:, :3]
selected_columns.head()
```

Out[4]:

	Invoice ID	Branch	City
0	750-67-8428	A	Yangon
1	226-31-3081	C	Naypyitaw
2	631-41-3108	A	Yangon
3	123-19-1176	A	Yangon
4	373-73-7910	A	Yangon

In [5]:
```
selected_columns = data[['Branch', 'Tax 5%']]
selected_columns.head()
```

Out[5]:

	Branch	Tax 5%
0	A	26.1415
1	C	3.8200
2	A	16.2155
3	A	23.2880
4	A	30.2085

In [6]:
```
data['Discounted_value'] = data['Total'] - (data['Total'] * 0.1)
data['Discounted_value'].head()
```

Out[6]:
```
0    494.07435
1     72.19800
2    306.47295
3    440.14320
4    570.94065
Name: Discounted_value, dtype: float64
```

In [7]:
```
data['Net_Payment'] = data['Total'] - data['Tax 5%']
data.head()
```

Out[7]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80.2200
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880	489.0480
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085	634.3785

In [8]:
```
# Apply function to multiple columns
numeric_cols = data.select_dtypes(include=['number']).columns[:3]

# Apply square root to multiple columns
data_transformed = data[numeric_cols].apply(lambda x: x ** 0.5)
data_transformed.columns = [f'{col}_sqrt' for col in numeric_cols]

print("Applied square root to numeric columns:")
print(data_transformed.head())

# Alternative: using applymap (element-wise operation)
print("\n" + "="*50 + "\n")
print("Original values (first 3 numeric columns):")
print(data[numeric_cols].head())
```

```
Applied square root to numeric columns:
   Unit price_sqrt  Quantity_sqrt  Tax 5%_sqrt
0         8.642338       2.645751     5.112876
1         3.908964       2.236068     1.954482
2         6.806614       2.645751     4.026847
3         7.630203       2.828427     4.825764
4         9.290318       2.645751     5.496226

==================================================

Original values (first 3 numeric columns):
   Unit price  Quantity   Tax 5%
0       74.69         7  26.1415
1       15.28         5   3.8200
2       46.33         7  16.2155
3       58.22         8  23.2880
4       86.31         7  30.2085
```

In [13]:

```python
# Delete columns
print(f"Columns before deletion: {data.shape[1]}")
print(f"Column names: {data.columns.tolist()}")


print(f"\nColumns after deletion: {data.shape[1]}")
print(f"Column names: {data.columns.tolist()}")
print("\nDataframe after deletion:")
print(data.head())
```

```
Columns before deletion: 19
Column names: ['Invoice ID', 'Branch', 'City', 'Customer type', 'Gender', 'Product
line', 'Unit price', 'Quantity', 'Tax 5%', 'Total', 'Date', 'Time', 'Payment', 'co
gs', 'gross margin percentage', 'gross income', 'Rating', 'Discounted_value', 'Net
_Payment']

Columns after deletion: 19
Column names: ['Invoice ID', 'Branch', 'City', 'Customer type', 'Gender', 'Product
line', 'Unit price', 'Quantity', 'Tax 5%', 'Total', 'Date', 'Time', 'Payment', 'co
gs', 'gross margin percentage', 'gross income', 'Rating', 'Discounted_value', 'Net
_Payment']

Dataframe after deletion:
    Invoice ID Branch       City Customer type  Gender  \
0  750-67-8428      A     Yangon        Member  Female
1  226-31-3081      C  Naypyitaw        Normal  Female
2  631-41-3108      A     Yangon        Normal    Male
3  123-19-1176      A     Yangon        Member    Male
4  373-73-7910      A     Yangon        Normal    Male

             Product line  Unit price  Quantity   Tax 5%     Total       Date  \
0       Health and beauty       74.69         7  26.1415  548.9715   1/5/2019
1  Electronic accessories       15.28         5   3.8200   80.2200   3/8/2019
2      Home and lifestyle       46.33         7  16.2155  340.5255   3/3/2019
3       Health and beauty       58.22         8  23.2880  489.0480  1/27/2019
4       Sports and travel       86.31         7  30.2085  634.3785   2/8/2019

    Time      Payment    cogs  gross margin percentage  gross income  Rating  \
0  13:08      Ewallet  522.83                 4.761905       26.1415     9.1
1  10:29         Cash   76.40                 4.761905        3.8200     9.6
2  13:23  Credit card  324.31                 4.761905       16.2155     7.4
3  20:33      Ewallet  465.76                 4.761905       23.2880     8.4
4  10:37      Ewallet  604.17                 4.761905       30.2085     5.3

   Discounted_value  Net_Payment
0         494.07435       522.83
1          72.19800        76.40
2         306.47295       324.31
3         440.14320       465.76
4         570.94065       604.17
```

In [14]:

```python
# Write dataframe to different file formats

# 1. Write to CSV
data.to_csv('output_data.csv', index=False)
print("Data written to 'output_data.csv'")

# 2. Write to Excel
try:
    data.to_excel('output_data.xlsx', index=False)
    print("Data written to 'output_data.xlsx'")
except ImportError:
    print("Excel export error")

# 3. Write to JSON
data.to_json('output_data.json', orient='records', indent=2)
print("Data written to 'output_data.json'")

print(f"\nTotal rows written: {len(data)}")
```

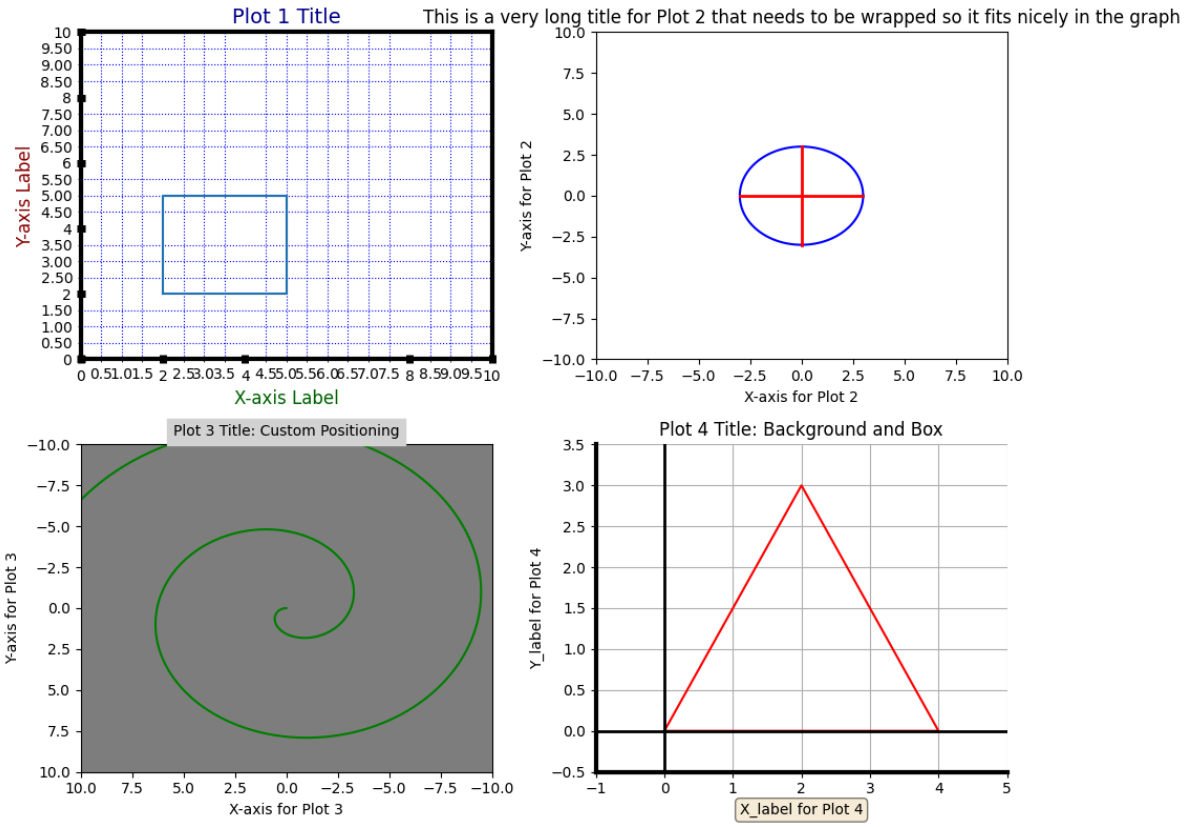
```
Data written to 'output_data.csv'
Data written to 'output_data.xlsx'
Data written to 'output_data.json'

Total rows written: 1000
```

Task 2: Drawing shapes (shapes + axes + grid)

In [18]:

```python
import numpy as np
import matplotlib.pyplot as plt
import textwrap
from matplotlib.ticker import AutoMinorLocator, FormatStrFormatter
fig, ax = plt.subplots(2,2, figsize=(10,8))
ax[0,0].set_title("Plot 1 Title", fontsize=14, color='darkblue')
ax[0,0].set_xlabel("X-axis Label", fontsize=12, color='darkgreen')
ax[0,0].set_ylabel("Y-axis Label", fontsize=12, color='darkred')
## Square
ax[0,0].plot([2,2,5,5,2],[2,5,5,2,2])
ax[0,0].set_xlim(0,10)
ax[0,0].set_ylim(0,10)
## Set ticks
ax[0,0].set_xticks([0,2,4,8,10])
ax[0,0].tick_params(axis='both', which='major', direction='inout', length=6, width=
ax[0,0].minorticks_on()
ax[0,0].grid(True, which='both',color='blue', linestyle='dotted')
ax[0,0].set_xticks([1,3,5,7,9], minor=True)
ax[0,0].xaxis.set_minor_locator(AutoMinorLocator())
ax[0,0].yaxis.set_minor_locator(AutoMinorLocator())
ax[0,0].xaxis.set_minor_formatter(FormatStrFormatter('%.1f'))
ax[0,0].yaxis.set_minor_formatter(FormatStrFormatter('%.2f'))

ax[0,1].set_title("This is a very long title for Plot 2 that needs to be wrapped sc
ax[0,1].set_xlabel("X-axis for Plot 2")
ax[0,1].set_ylabel("Y-axis for Plot 2")
r = 3
theta = np.linspace(0, 2*np.pi, 400)
# Circle
x = r * np.cos(theta)
y = r * np.sin(theta)
ax[0,1].plot(x, y, color='blue')
# Plus sign
ax[0,1].plot([-3, 3], [0, 0], color='red', linewidth=2)  # Horizontal line
ax[0,1].plot([0, 0], [-3, 3], color='red', linewidth=2)  # Vertical line
ax[0,1].set_xlim(-10,10)
ax[0,1].set_ylim(-10,10)
ax[1,0].set_title("Plot 3 Title: Custom Positioning", fontsize=10, backgroundcolor=
ax[1,0].set_xlabel("X-axis for Plot 3")
ax[1,0].set_ylabel("Y-axis for Plot 3")
## Spiral
r = np.linspace(0, 4*np.pi, 400)
x = r * np.cos(r)
y = r * np.sin(r)
ax[1,0].plot(x,y, color='green')
ax[1,0].set_xlim(-10,10)
ax[1,0].set_ylim(-10,10)
ax[1,0].invert_xaxis()
ax[1,0].invert_yaxis()
ax[1,1].set_title("Plot 4 Title: Background and Box")
ax[1,1].set_xlabel("X_label for Plot 4", backgroundcolor='yellow', color='black', b
ax[1,1].set_ylabel("Y_label for Plot 4")
```

```
## Triangle
ax[1,1].plot([0,4,2,0],[0,0,3,0], color='red')
##Clears the plot
##ax[1,1].clear()
ax[1,1].axhline(0, color='black', linewidth=2)
ax[1,1].axvline(0, color='black', linewidth=2)
ax[1,1].set_xlim(-1,5)
ax[1,1].set_ylim(-0.5, 3.5)
ax[1,1].spines['top'].set_visible(False)
ax[1,1].spines['right'].set_visible(False)
ax[1,1].spines['left'].set_linewidth(3)
ax[1,1].spines['bottom'].set_linewidth(3)
ax[1,1].grid()
ax[0,0].spines['top'].set_linewidth(3)
ax[0,0].spines['right'].set_linewidth(3)
ax[0,0].spines['left'].set_linewidth(3)
ax[0,0].spines['bottom'].set_linewidth(3)
ax[0,0].grid()
ax[1,0].set_facecolor('gray')
plt.tight_layout()
plt.show()
```

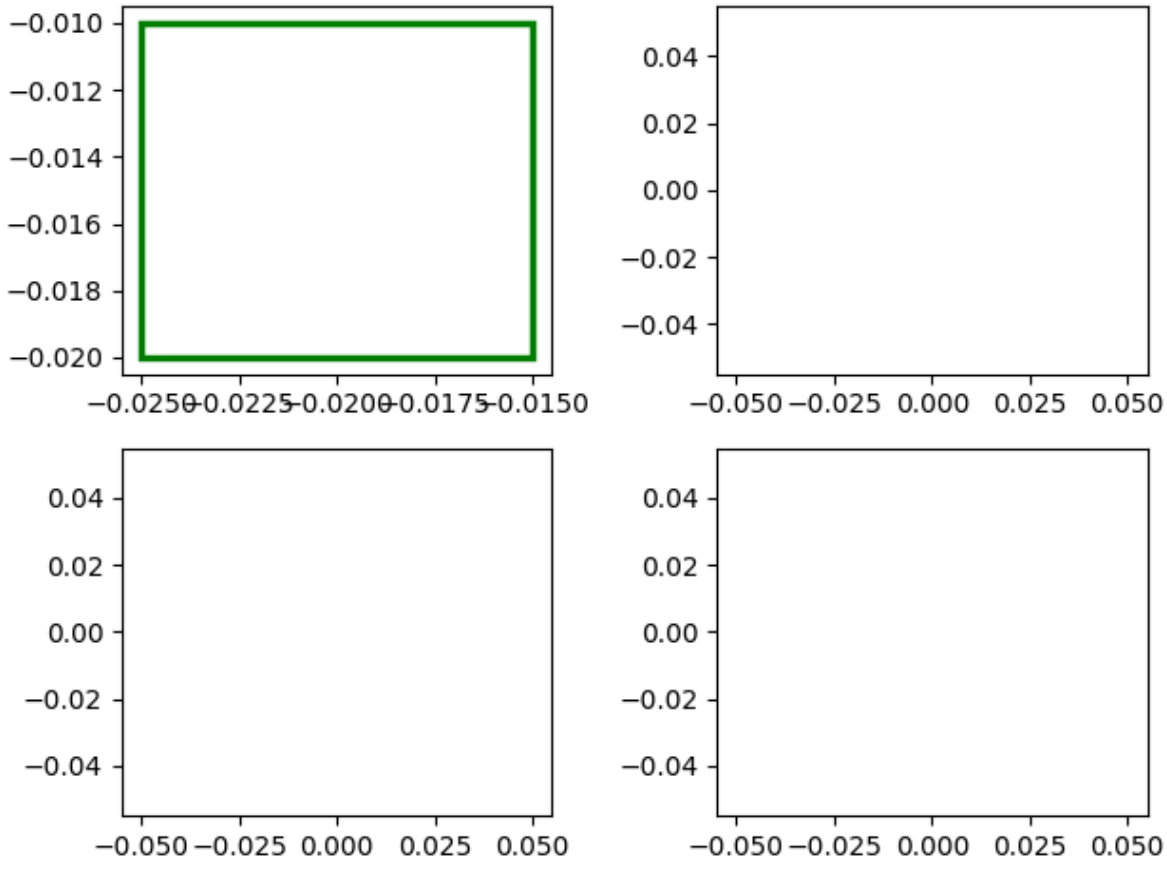


Task-3: Shapes using patches(ticks and tick labels(minor and major))

In [3]:

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

# Creating subplots
fig, ax = plt.subplots(2, 2)

# Plot random data in each subplot
for row in ax:
    for col in row:
```

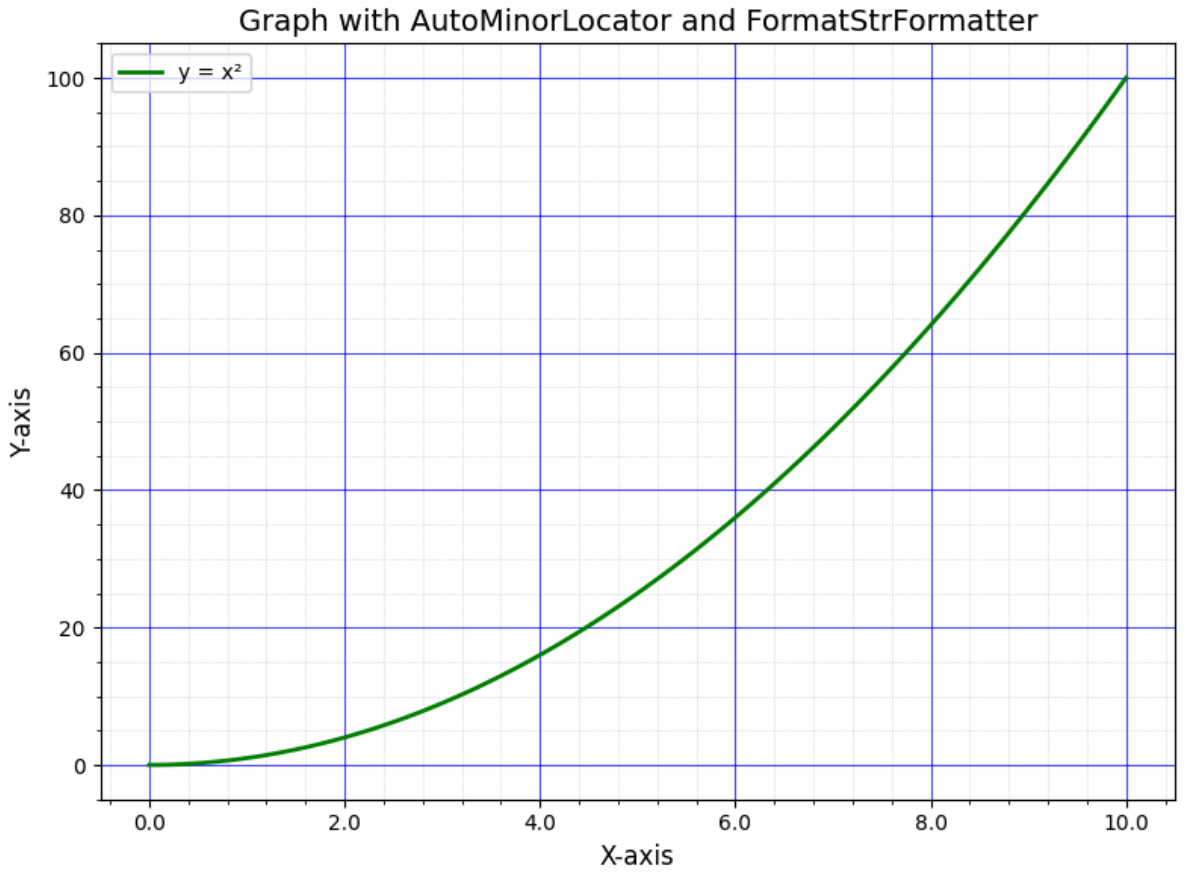
```
        col.plot()

# Add a rectangle patch to subplot [0][0]
rect = patches.Rectangle(
    (-0.025, -0.02),
    0.01,
    0.01,
    # (x, y) lower-left corner
    linewidth=2.5,
    edgecolor='green',
    fill = False
)
ax[0][0].add_patch(rect)

plt.tight_layout()
plt.show()
```



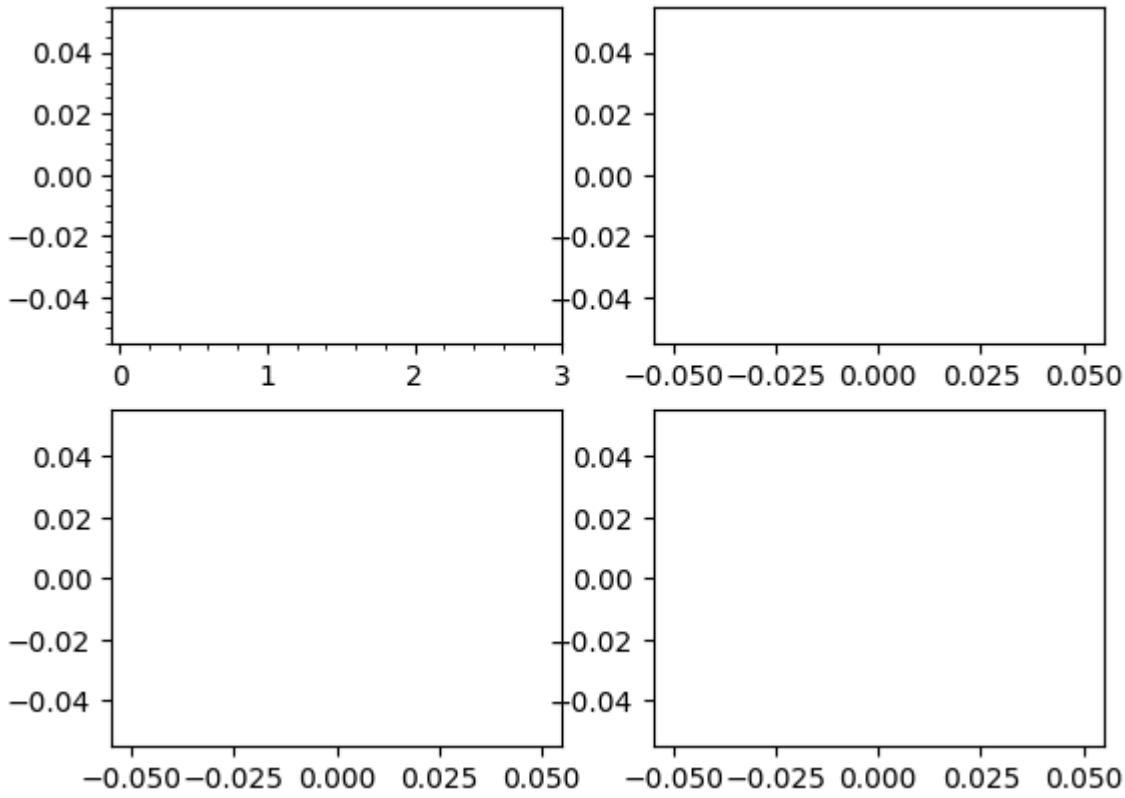
In [20]:

```
# Graph with AutoMinorLocator and FormatStrFormatter
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.ticker import AutoMinorLocator, FormatStrFormatter

# Sample data
x = np.linspace(0, 10, 100)
y = x**2

# Create plot
fig, ax = plt.subplots(figsize=(8, 6))
ax.plot(x, y, color='green', linewidth=2, label='y = x²')

# Set title and labels
ax.set_title('Graph with AutoMinorLocator and FormatStrFormatter', fontsize=14)
ax.set_xlabel('X-axis', fontsize=12)
ax.set_ylabel('Y-axis', fontsize=12)

# Use AutoMinorLocator for automatic minor ticks
```

```
ax.xaxis.set_minor_locator(AutoMinorLocator(5))  # 5 minor ticks between major tick
ax.yaxis.set_minor_locator(AutoMinorLocator(4))  # 4 minor ticks between major tick

# Use FormatStrFormatter for tick label formatting
ax.xaxis.set_major_formatter(FormatStrFormatter('%.1f'))  # One decimal place
ax.yaxis.set_major_formatter(FormatStrFormatter('%.0f'))  # No decimal places

# Customize grid
ax.grid(True, which='major', linestyle='-', linewidth=0.8, alpha=0.7, color='blue')
ax.grid(True, which='minor', linestyle=':', linewidth=0.5, alpha=0.4, color='gray')

ax.legend()
plt.tight_layout()
plt.show()
```



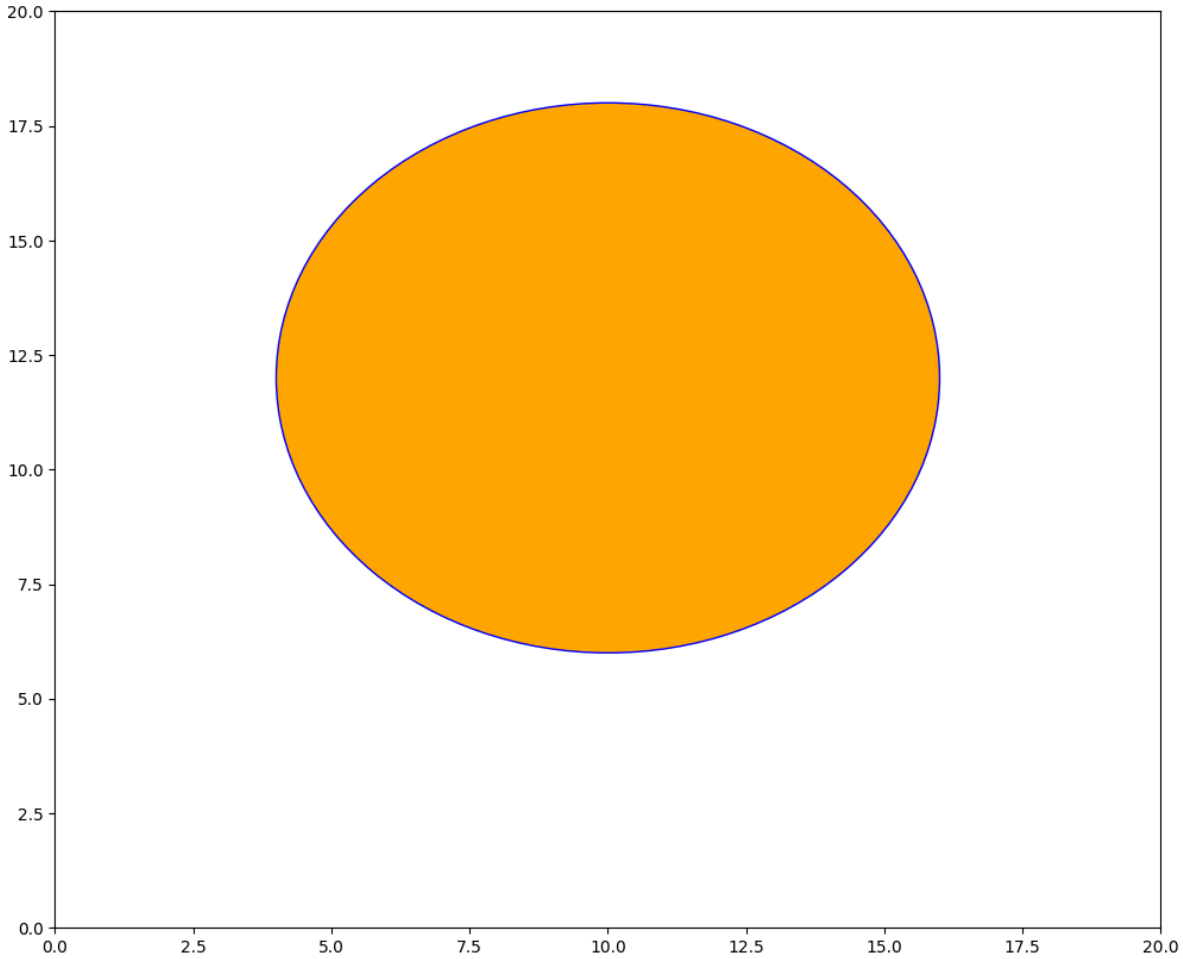
In [12]:
```
fig, ax = plt.subplots(2, 2)

# Plot random data in each subplot
for row in ax:
    for col in row:
        col.plot()
ax[0][0].set_xticks([1,3], minor=True)
ax[0][0].minorticks_on()
plt.show()
```



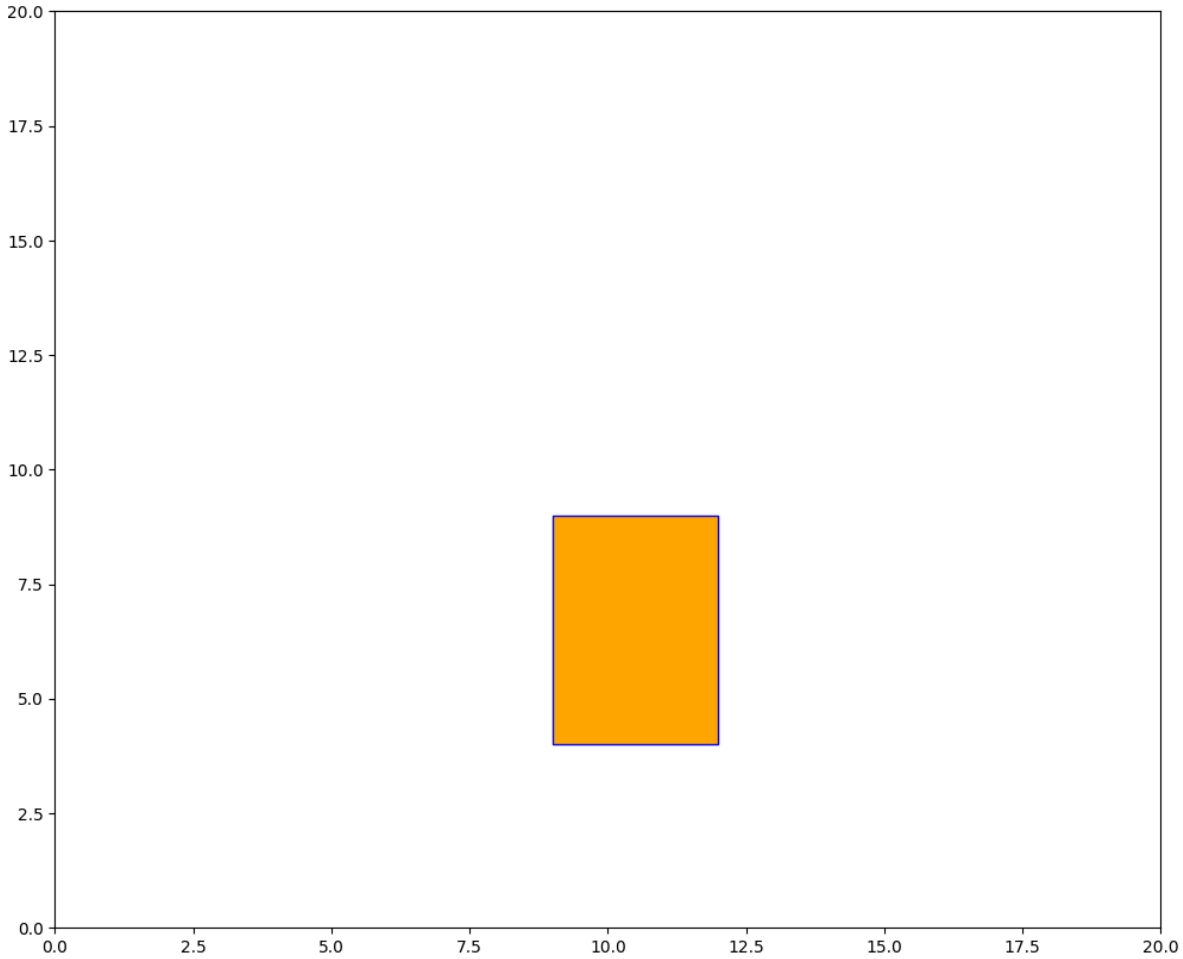
Plotting with matplotlib patches

In [20]:

```
from matplotlib.patches import Rectangle, Circle, Polygon
circle = Circle(
    (10,12),
    6,
    fill=True,
    facecolor='orange',
    edgecolor='blue',
    linewidth=1,
    alpha=1.0
)

fig, ax = plt.subplots(figsize=(12, 10))
ax.add_patch(circle)
ax.set_xlim(0, 20)
ax.set_ylim(0, 20)
plt.show()
```

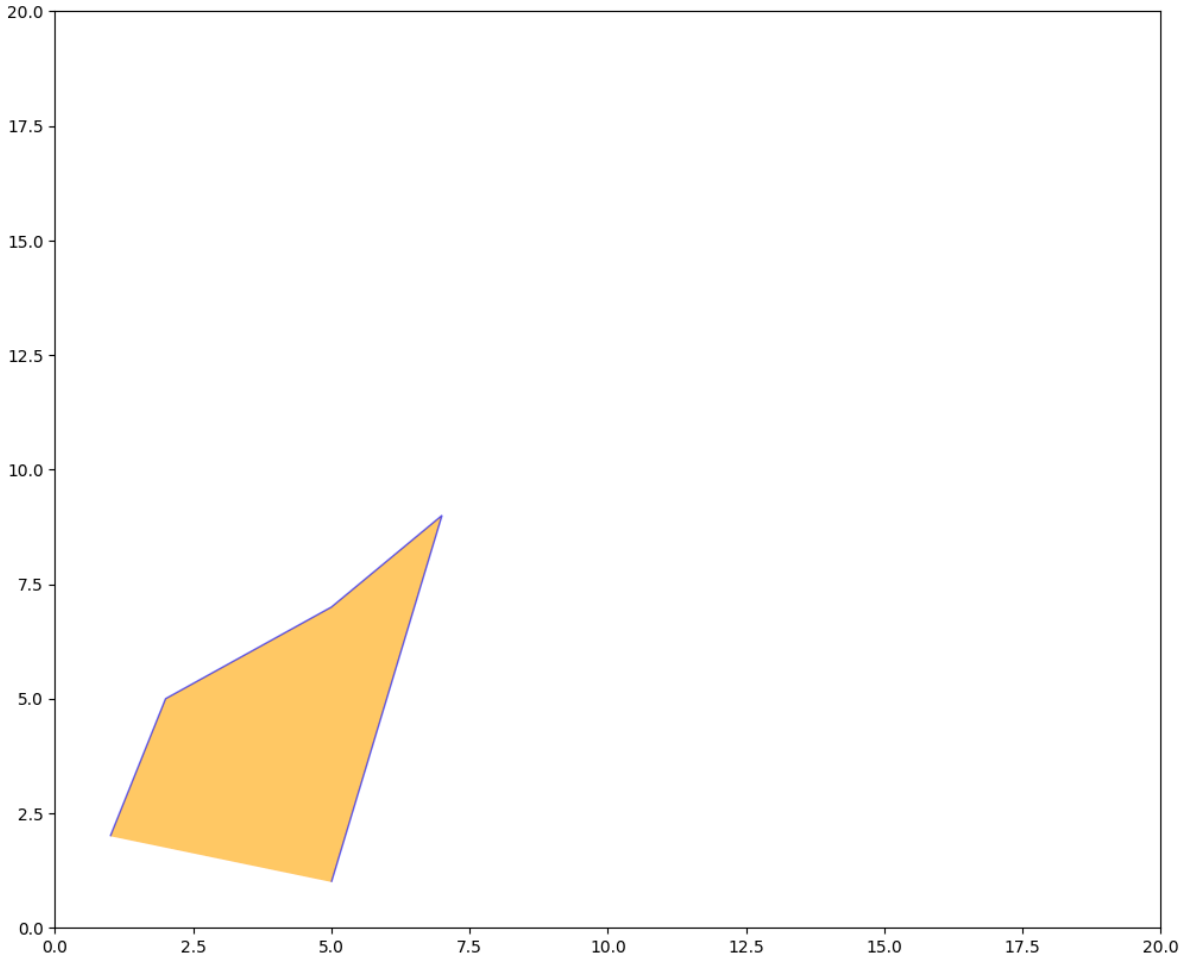


In [19]:
```
from matplotlib.patches import Rectangle, Circle, Polygon
rect = Rectangle(
    (12,4),
    5,
    3,
    angle=90,
    fill=True,
    facecolor='orange',
    edgecolor='blue',
    linewidth=1,
    alpha=1.0
)

fig, ax = plt.subplots(figsize=(12, 10))
ax.add_patch(rect)
ax.set_xlim(0, 20)
ax.set_ylim(0, 20)
plt.show()
```



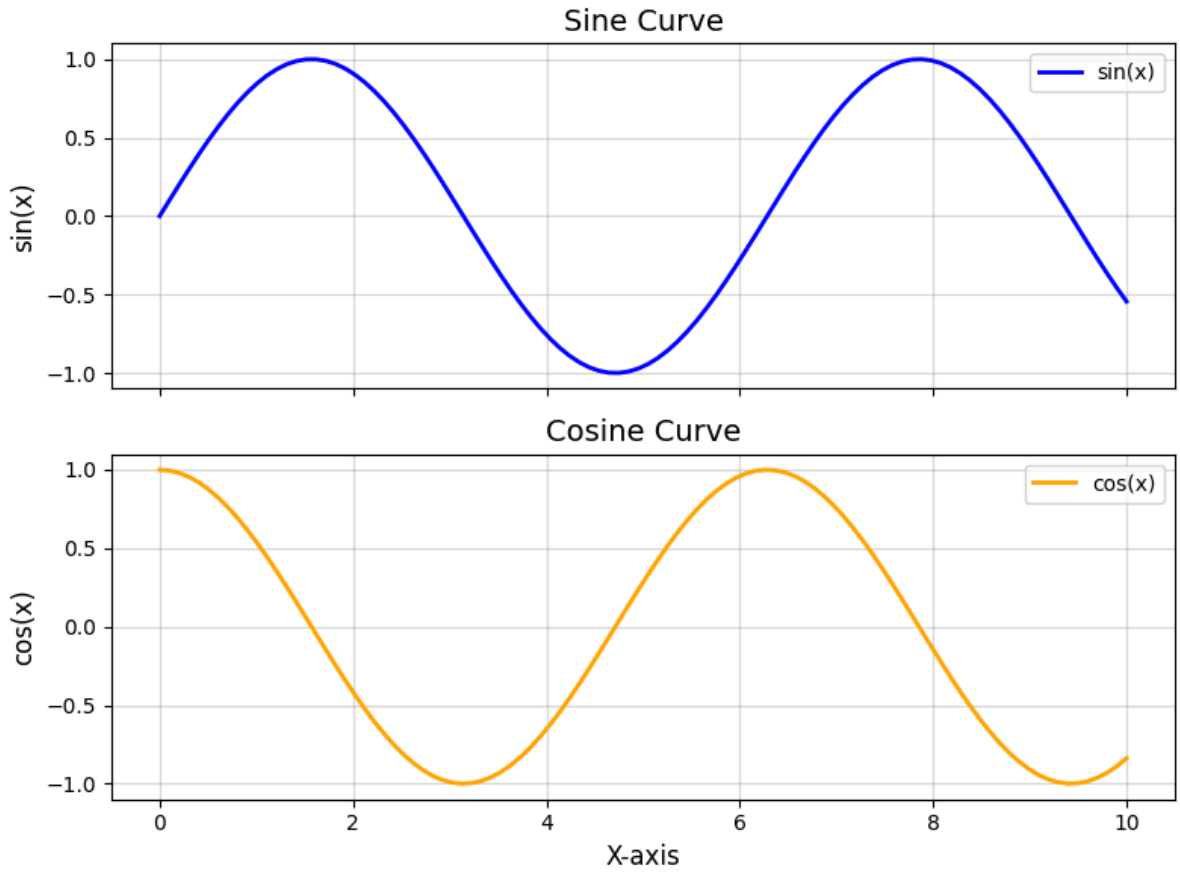
In [25]:

```
from matplotlib.patches import Rectangle, Circle, Polygon

vertices = [(1, 2),(2, 5),(5, 7),(7, 9),(5, 1)]

poly_bhai = Polygon(
    vertices,
    closed=False,
    fill=True,
    facecolor='orange',
    edgecolor='blue',
    linewidth=1,
    alpha=0.6
)

fig, ax = plt.subplots(figsize=(12, 10))
ax.add_patch(poly_bhai)
ax.set_xlim(0, 20)
ax.set_ylim(0, 20)
plt.show()
```



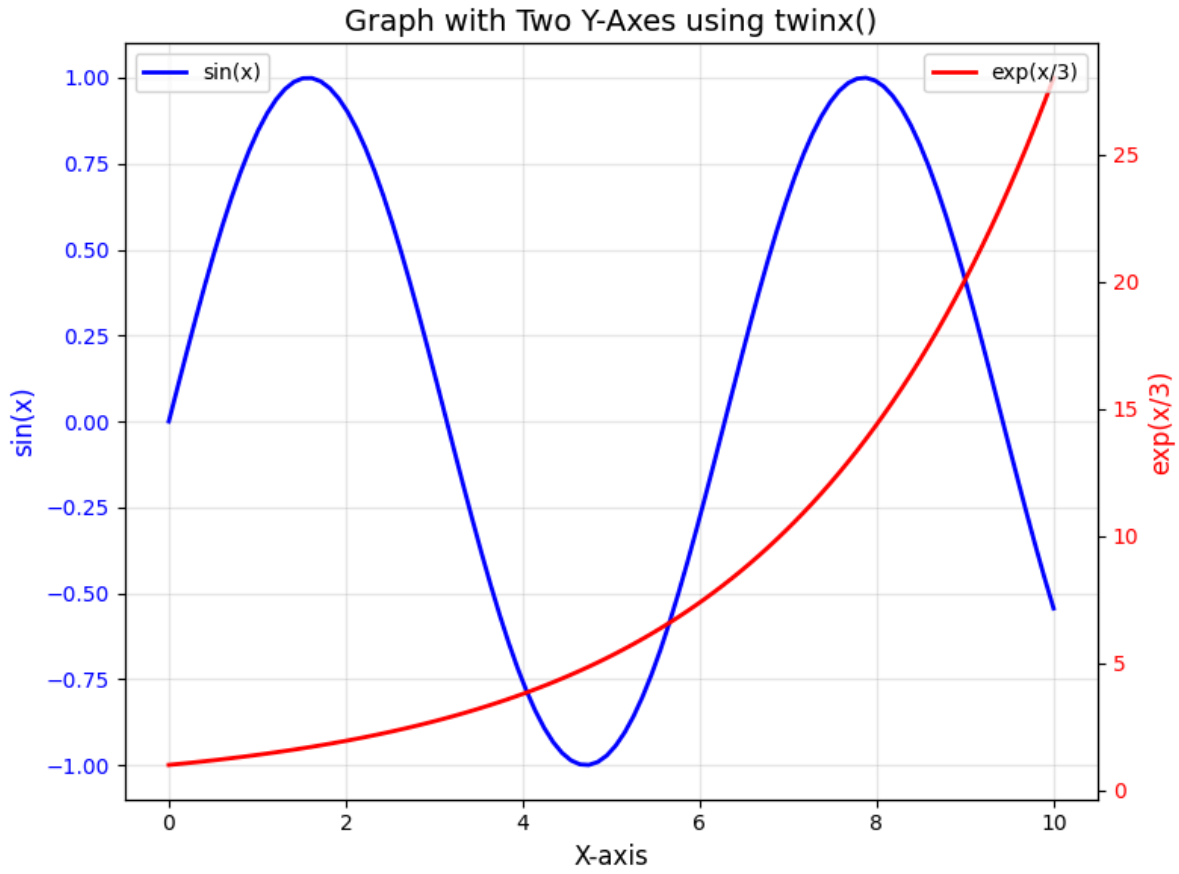
In [19]:

```
# Create a sin curve and a cos curve which share x axis
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(0, 10, 100)
y1 = np.sin(x)
y2 = np.cos(x)

# Create subplots with shared x-axis
fig, (ax1, ax2) = plt.subplots(nrows=2, ncols=1, sharex=True, figsize=(8, 6),sharey

# Plot sine curve
ax1.plot(x, y1, color='blue', linewidth=2, label='sin(x)')
ax1.set_ylabel('sin(x)', fontsize=12)
ax1.set_title('Sine Curve', fontsize=14)
ax1.legend()
ax1.grid(True, alpha=0.5)

# Plot cosine curve
ax2.plot(x, y2, color='orange', linewidth=2, label='cos(x)')
ax2.set_xlabel('X-axis', fontsize=12)
ax2.set_ylabel('cos(x)', fontsize=12)
ax2.set_title('Cosine Curve', fontsize=14)
ax2.legend()
ax2.grid(True, alpha=0.5)

plt.tight_layout()
plt.show()
```



In [21]:

```python
# Graph using twinx - two y-axes sharing the same x-axis
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(0, 10, 100)
y1 = np.sin(x)
y2 = np.exp(x/3)

# Create figure and first axis
fig, ax1 = plt.subplots(figsize=(8, 6))

# Plot first data on left y-axis
ax1.plot(x, y1, color='blue', linewidth=2, label='sin(x)')
ax1.set_xlabel('X-axis', fontsize=12)
ax1.set_ylabel('sin(x)', fontsize=12, color='blue')
ax1.tick_params(axis='y', labelcolor='blue')
ax1.grid(True, alpha=0.3)

# Create second y-axis sharing the same x-axis
ax2 = ax1.twinx()
ax2.plot(x, y2, color='red', linewidth=2, label='exp(x/3)')
ax2.set_ylabel('exp(x/3)', fontsize=12, color='red')
ax2.tick_params(axis='y', labelcolor='red')

# Add title and legends
ax1.set_title('Graph with Two Y-Axes using twinx()', fontsize=14)
ax1.legend(loc='upper left')
ax2.legend(loc='upper right')

plt.tight_layout()
plt.show()
```



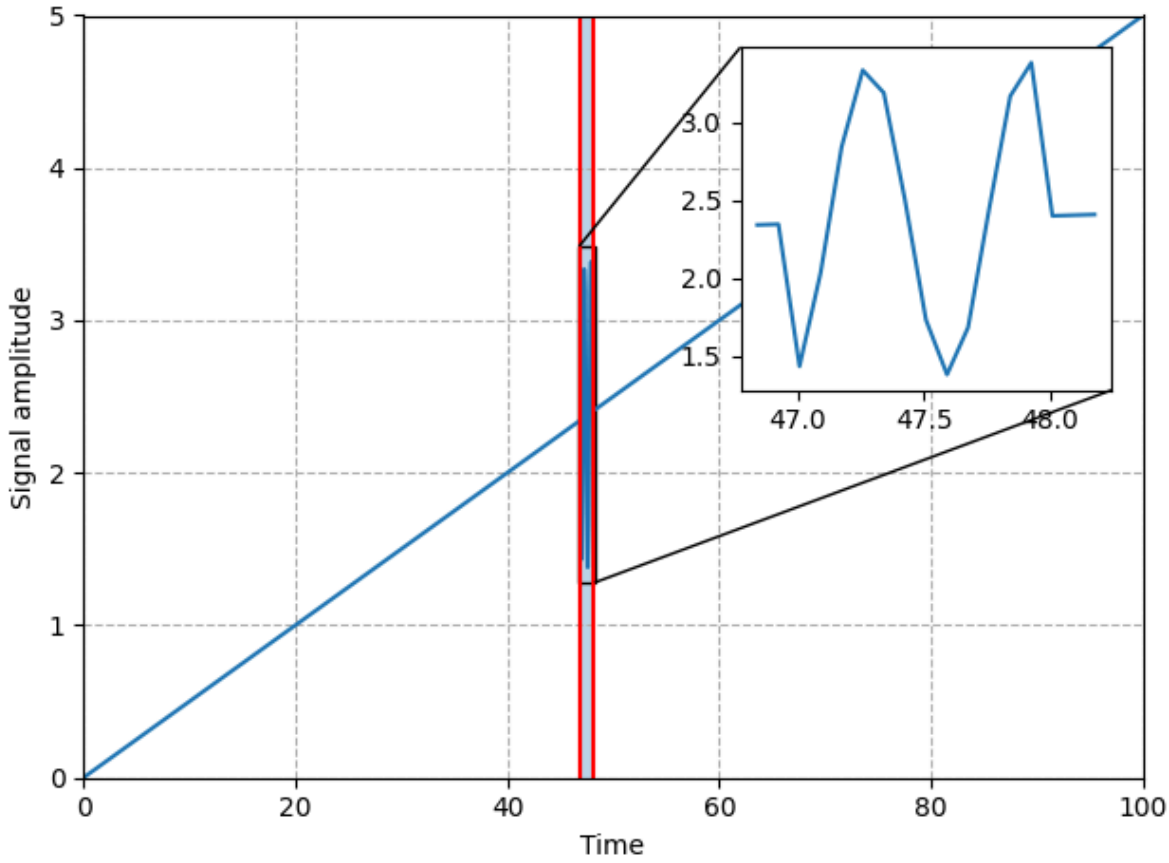
Task-4: Inset plotting (plot + legend + title)

In [41]:
```
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.axes_grid1.inset_locator import inset_axes, mark_inset
x = np.linspace(0, 500, 6000)
y = 0.05*x
# fig, ax = plt.subplots()
# ax.plot(x, y)
# plt.show()
```

In [42]:
```
for i in range(len(x)):
    if 47 < x[i] < 48:
        y[i] += np.sin(10*x[i])

x1, x2 = 46.8, 48.2
xy_mask = (x > x1)&(x < x2)
x_cord=x[xy_mask]
y_cord=y[xy_mask]

fig, ax = plt.subplots()
ax.plot(x, y)

plt.axvline(x =46.8, color ='red')
plt.axvline(x =48.2, color='red')
ax.axvspan(xmin=46.8,xmax=48.2, alpha=0.3)

ax.set_xlim(0, 100)
ax.set_ylim(0, 5)
ax.grid(True, linestyle='--')
ax.set_xlabel("Time")
ax.set_ylabel("Signal amplitude")
```

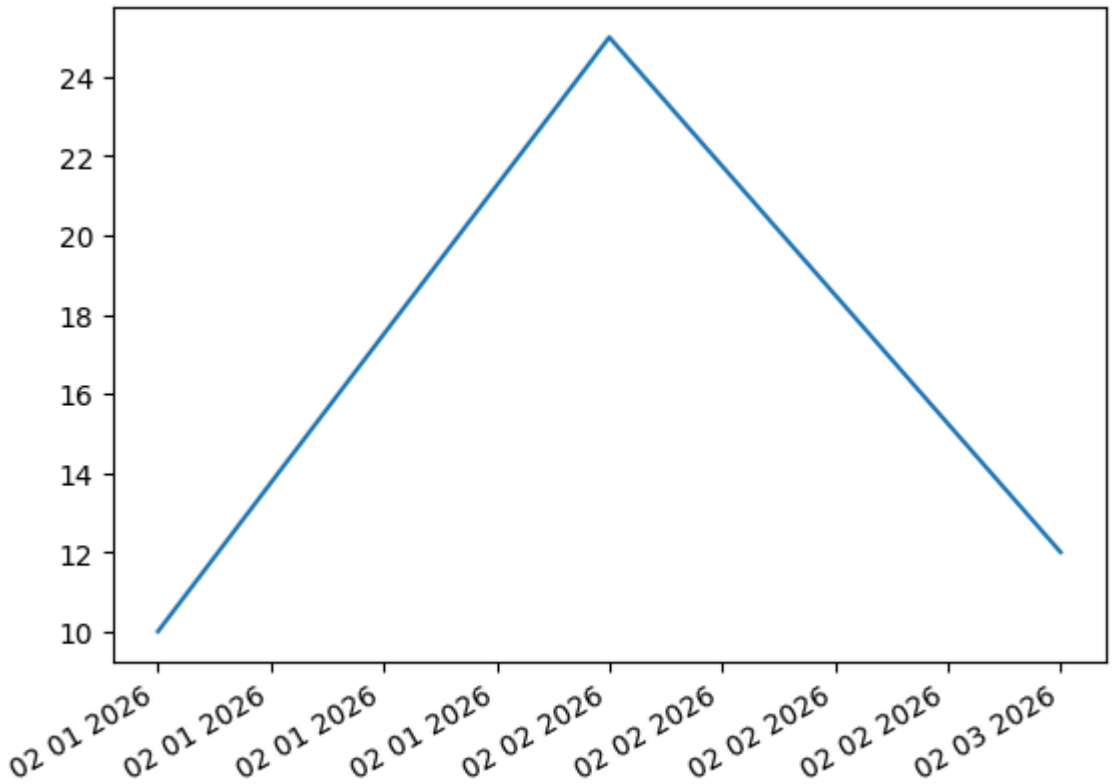
```
axins= inset_axes(
ax,  #parent axes
width ="35%",  #35%of the parent axes
height = "45%",  #45% of the parent axes
loc="upper right",
borderpad = 1.2)

axins.plot(x_cord, y_cord)
mark_inset(ax, axins, loc1=2, loc2=4, fc="none", ec="black", linewidth=1)

plt.tight_layout()
plt.show()
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_24012\4166149564.py:33: UserWarning: This
figure includes Axes that are not compatible with tight_layout, so results might b
e incorrect.
  plt.tight_layout()
```



Task-5 working with date and time data

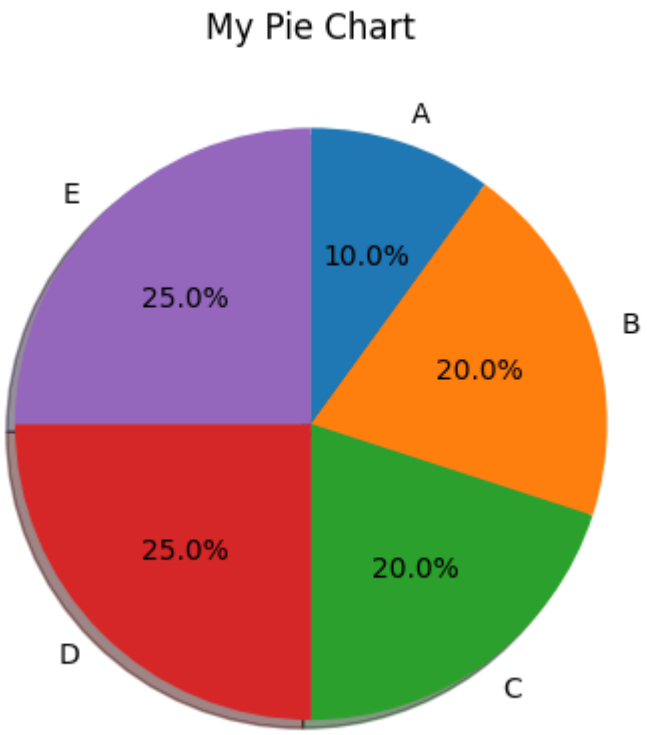
In [12]:
```
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from datetime import datetime

x = [datetime.strptime("2026-2-1", "%Y-%m-%d"),
     datetime.strptime("2026-2-2", "%Y-%m-%d"),
     datetime.strptime("2026-2-3", "%Y-%m-%d")]
y = [10, 25, 12]
fig, ax = plt.subplots()
ax.plot(x, y)
fig.autofmt_xdate()
ax.xaxis.set_major_formatter(mdates.DateFormatter('%m %d %Y'))
```



In [ ]:
```
now = datetime.now()
timestamp = now.strftime("%Y=%m-%d %H:%M:%S")
ax.text(0.02, 0.5)
```

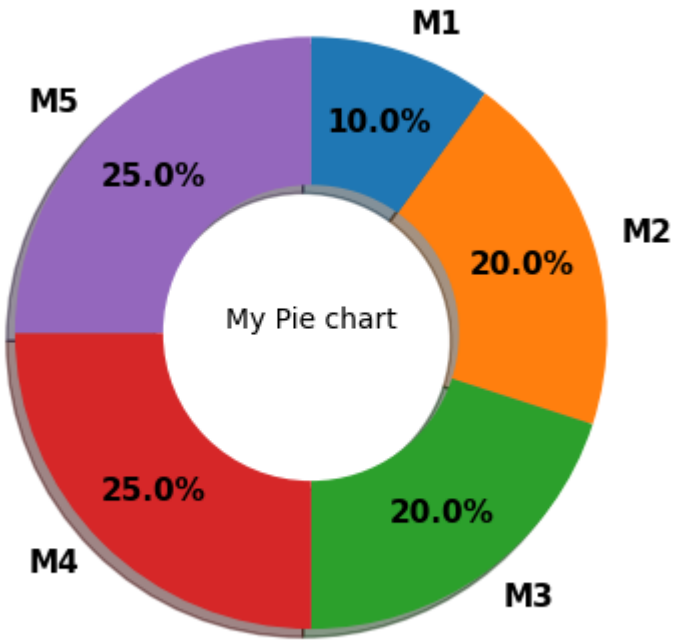
Task-6: Pie and Donut

In [23]:
```
import matplotlib.pyplot as plt

x = [10, 20, 20, 25, 25]

fig, ax = plt.subplots()   # ax is an instance of matplotlib.axes.Axes
ax.pie(
    x,
    labels=["M1", "M2", "M3", "M4", "M5"],   # optional labels
    autopct="%1.1f%%",                  # show percentages
    startangle=90,                      # rotate start angle
    shadow=True,                         # add shadow
    counterclock = False
)

ax.set_title("My Pie Chart")
plt.show()
```

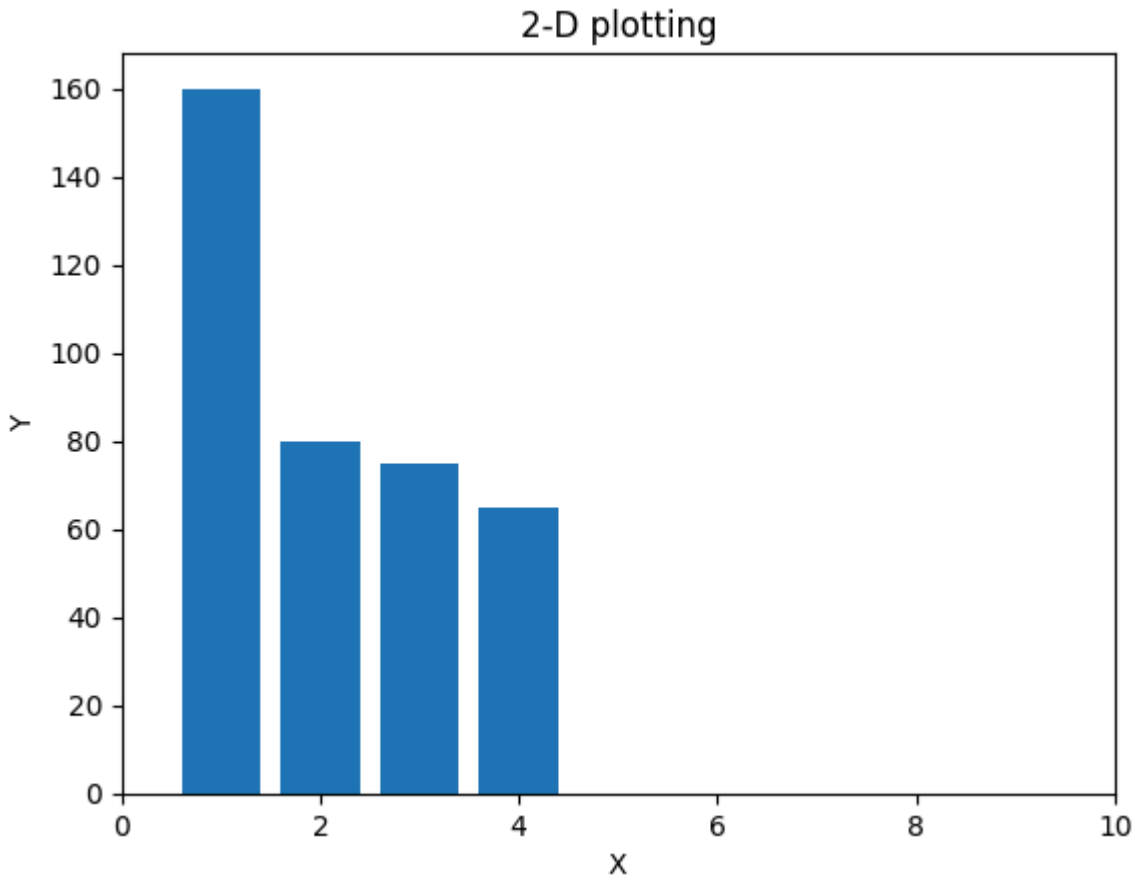


In [28]:

```
import matplotlib.pyplot as plt

x = [10, 20, 20, 25, 25]

fig, ax = plt.subplots()   # ax is an instance of matplotlib.axes.Axes
ax.pie(
    x,
    labels=["M1", "M2", "M3", "M4", "M5"],   # optional labels
    autopct="%1.1f%%",                       # show percentages
    pctdistance=0.75,
    startangle=90,                        # rotate start angle
    shadow=True,                           # add shadow
    counterclock = False,
    wedgeprops={'width':0.5},
    textprops={'fontsize' :11, 'fontweight':'bold'},
    normalize = True,
    rotatelabels = False
)

ax.text(0, 0, "My Pie chart", ha='center', va='bottom')
plt.show()
```

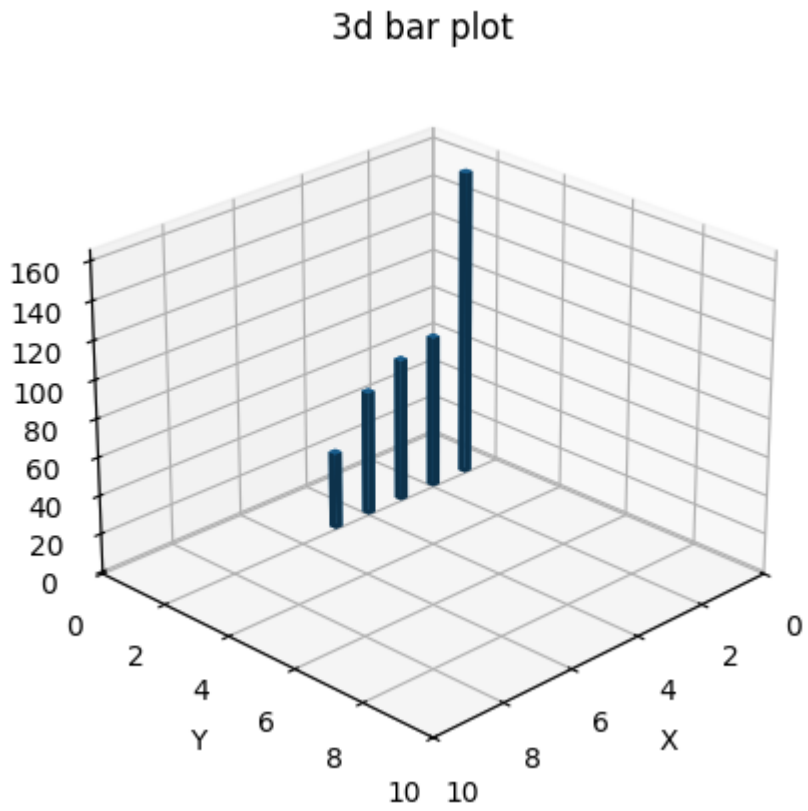


In [32]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

x = [1,2, 3, 4]
y = [160, 80, 75, 65]

fig,ax = plt.subplots()

ax.bar(x, y)
ax.set_title("2-D plotting")
ax.set_xlabel("X")
ax.set_ylabel("Y")
ax.set_xlim(0, 10)

plt.show()
```



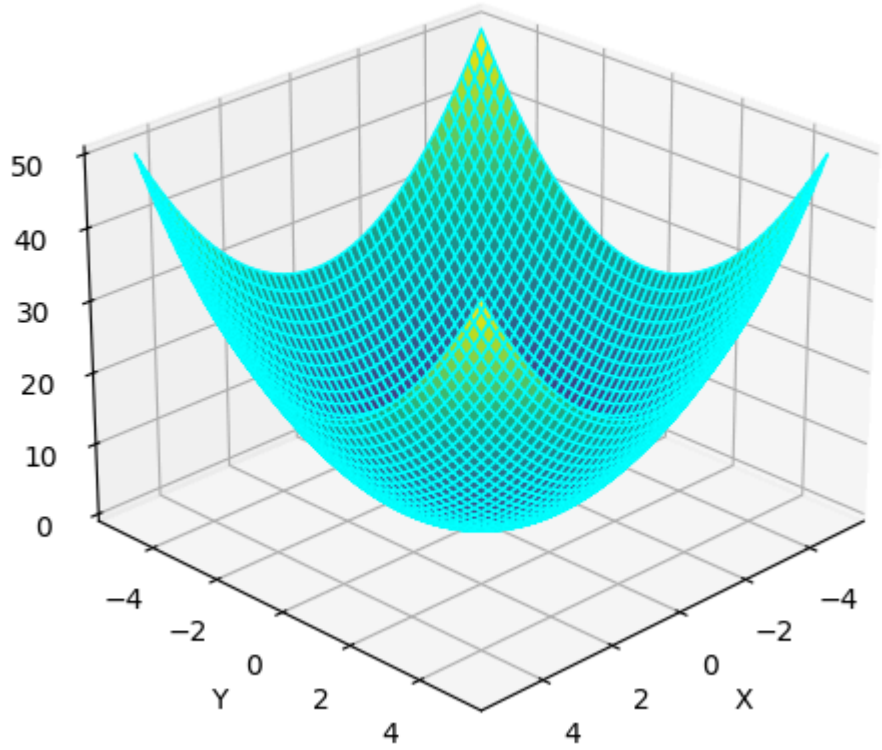
In [35]:

```
from mpl_toolkits.mplot3d import Axes3D

x = [1, 2, 3, 4, 5]
z = [160, 80, 75, 65, 40]

y = 2

x = np.array(x)
z_base = 2

dx = 0.2
dy = 0.2
dz = z

fig = plt.figure()

ax = fig.add_subplot(111, projection='3d')
ax.bar3d(x, y, z_base, dx, dy, dz)
ax.set_xlabel("X")
ax.set_ylabel("Y")
ax.set_zlabel("Z")
ax.set_xlim(0, 10)
ax.set_ylim(0, 10)
ax.set_title("3d bar plot")
ax.view_init(elev=25, azim=45)
plt.show()
```



Task-7: 3D plotting (Bar)

In [13]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

x = np.linspace(-5, 5, 100)
y = np.linspace(-5, 5, 100)
X, Y = np.meshgrid(x, y)
Z = X ** 2 + Y **2
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.set_xlabel("X")
ax.set_ylabel("Y")
ax.set_title("Plotting equations")
ax.plot_surface(X,Y, Z, cmap='viridis', color='cyan')
ax.view_init(elev=25, azim=45)
plt.tight_layout()
plt.show()
```

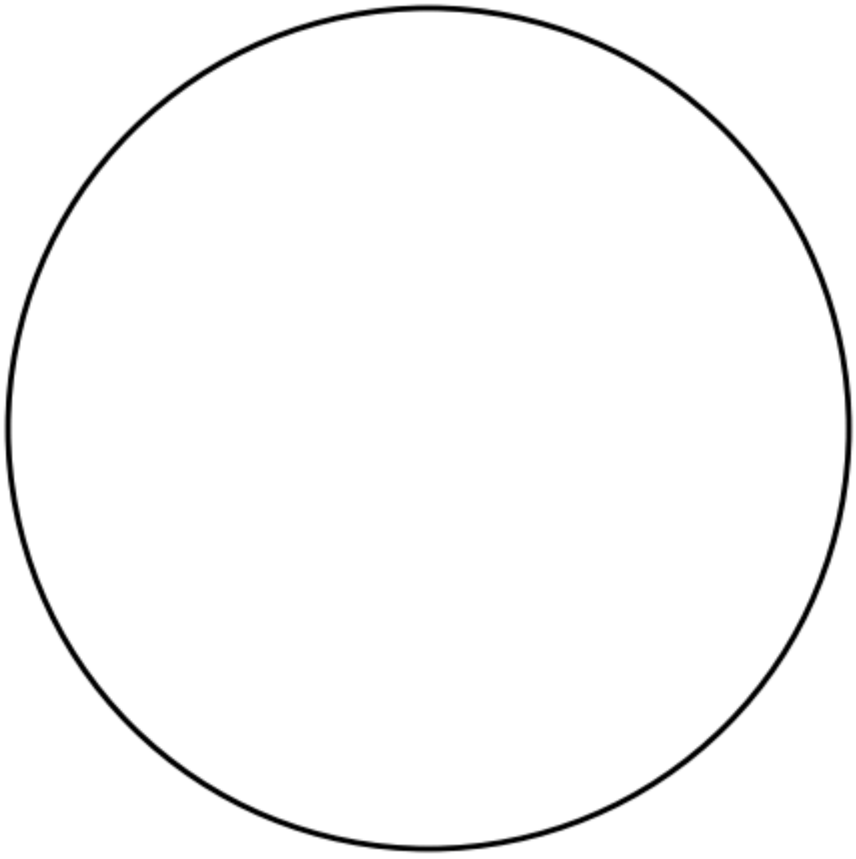


In [1]:

```
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
from datetime import datetime
from zoneinfo import ZoneInfo
import numpy as np

tz = ZoneInfo("Asia/Kolkata")

def angle_rad(angle, length):
    theta = np.radians(90 - angle)
    return length * np.cos(theta), length * np.sin(theta)

fig, ax = plt.subplots(figsize=(6, 6))

circle = plt.Circle((0, 0), 1, fill=False, linewidth=2)
ax.add_artist(circle)

ax.set_xlim(-1.1, 1.1)
ax.set_ylim(-1.1, 1.1)
ax.set_aspect('equal')
ax.axis('off')

hour_hand, = ax.plot([], [], linewidth=6, color='black')
minute_hand, = ax.plot([], [], linewidth=3, color='blue')
second_hand, = ax.plot([], [], linewidth=1, color='red')

title = ax.set_title("")

def update(frame):
    now = datetime.now(tz)

    hour = now.hour % 12
    minute = now.minute
    second = now.second
```

```
    hour_angle = (hour + minute / 60) * 30
    minute_angle = (minute + second / 60) * 6
    second_angle = second * 6

    hx, hy = angle_rad(hour_angle, 0.5)
    mx, my = angle_rad(minute_angle, 0.7)
    sx, sy = angle_rad(second_angle, 0.9)

    hour_hand.set_data([0, hx], [0, hy])
    minute_hand.set_data([0, mx], [0, my])
    second_hand.set_data([0, sx], [0, sy])

    title.set_text(f"Analog Clock - IST ({now.strftime('%H:%M:%S')})")

ani = FuncAnimation(fig, update, interval=1000)

plt.show()
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_17168\4144378672.py:50: UserWarning: fram
es=None which we can infer the length of, did not pass an explicit *save_count* an
d passed cache_frame_data=True.  To avoid a possibly unbounded cache, frame data c
aching has been disabled. To suppress this warning either pass `cache_frame_data=F
alse` or `save_count=MAX_FRAMES`.
  ani = FuncAnimation(fig, update, interval=1000)
```



In [ ]: