# 复习题

## 1、数据的全生命周期管理包括哪些阶段?

数据的全生命周期管理包括几个阶段：数据的产生、存储、处理、使用和销毁。首先，数据会被收集和存储，然后经过处理，变得可用。之后，数据用于支持决策或业务，最后，当数据不再需要时，会安全地销毁。

## 2、数据采集的概念是什么？都有哪些方法?

数据采集就是收集信息的过程，目的是为了分析和应用。常见的方法有问卷调查，比如设计问题让人填写；传感器可以自动获取数据，比如温度、湿度等；网络爬虫则是用程序从网上抓取信息；某些企业内部经常使用的就是从数据库中直接查询所需的信息；最后，还有通过实验观察来记录数据，这在科学研究中特别常见。

## 3、什么是数据管理？比较传统的数据管理和大数据管理技术有什么异同?

数据管理是指对数据的收集、存储和使用进行管理，确保数据能够被有效利用和安全保存。传统的数据管理主要是针对结构化数据，像关系数据库就比较常用，适合处理一些小规模的数据。这种管理方法相对简单，使用的技术也比较成熟。

相比之下，大数据管理处理的是更复杂和更大规模的数据，包括结构化、半结构化和非结构化的数据。大数据的特点是量大、类型多、变化快，所以需要更高级的工具和技术，比如Hadoop这样的分布式存储，或者一些流处理和数据分析平台。大数据管理更注重快速处理和实时性，适合应对海量数据和快速变化的环境。

## 4、大数据的计算模式可以分为哪几类?

大数据的计算模式有几个主要类型。一个是**批处理**，这个模式就是把大量的数据集中在一起，慢慢处理，像Hadoop这种工具比较常用，适合一些不需要实时反馈的场景。还有**流处理**，这个适合实时数据，像社交媒体更新这种，需要快速反应，常见的工具有Kafka和Flink。

接着是**交互式处理**，这就允许用户即时查询数据，适合想要快速分析的情况，像Drill和Presto这样的工具就会用到。最后是**混合处理**，有些场景可能需要同时处理实时和历史数据，这种情况下就会结合这两种方式。

## 5、什么是数据分析？有哪些数据分析的方法或者模型?

数据分析就是通过对数据的整理和分析，找出有用的信息，帮助做出更好的决策。它可以用在商业、科学研究和市场营销等多个领域。

常见的数据分析方法有几种。描述性分析主要是对数据进行总结，比如计算平均值和标准差，帮助我们了解数据的基本情况。诊断性分析则关注数据变化的原因，像用相关性分析来看两个变量之间的关系。

还有预测性分析，它通过历史数据来预测未来的趋势，常用的方法有回归分析和时间序列分析。最后是规范性分析，提供决策建议，通常结合多种因素来评估不同的选择，比如使用优化模型和决策树。

## 6、数据可视化的原因有哪些?

数据可视化的原因主要有几个方面。它可以让复杂的信息更容易理解，通过图表和图形，我们能快速抓住数据的趋势和模式，而不用逐个数字分析。

另外，数据可视化也有助于发现一些潜在的关系和异常值，这些在表格里可能不太明显。通过可视化，我们可以更容易看到不同变量之间的关系。

还有就是，在团队讨论或者汇报时，用图表传达信息更高效，听众能更快理解关键点。

最后，数据可视化能帮助做出更好的决策，让决策者能清晰看到各种选项的影响。

# 实践题

## 7、熟悉可视化包Matplotlib,绘制任一数据集的三种常见图形。

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

# 创建示例数据
x = np.linspace(0, 10, 100)  # x 值从 0 到 10，分成 100 个点
y1 = np.sin(x)               # y1 为 sin(x)
y2 = np.cos(x)               # y2 为 cos(x)
y3 = np.random.rand(100)     # y3 为随机数据

plt.figure(figsize=(10, 6))  # 设置图形大小
plt.plot(x, y1, label='sin(x)', color='b', linestyle='-')  # 绘制 sin(x) 折线
plt.plot(x, y2, label='cos(x)', color='r', linestyle='--')  # 绘制 cos(x) 折线
plt.title('Line Plot of Sin and Cos Functions')  # 图形标题
plt.xlabel('X-axis')                             # X 轴标签
plt.ylabel('Y-axis')                             # Y 轴标签
plt.legend()                                     # 显示图例
plt.grid()                                       # 显示网格
plt.show()                                       # 显示图形

plt.figure(figsize=(10, 6))  # 设置图形大小
plt.scatter(x, y3, color='g', marker='o', label='Random Data')  # 绘制随机数据的散点图
plt.title('Scatter Plot of Random Data')  # 图形标题
plt.xlabel('X-axis')                      # X 轴标签
plt.ylabel('Y-axis')                      # Y 轴标签
plt.legend()                              # 显示图例
plt.grid()                                # 显示网格
plt.show()                                # 显示图形

# 创建柱状图数据
categories = ['Category 1', 'Category 2', 'Category 3', 'Category 4']
values = [10, 20, 15, 25]

plt.figure(figsize=(10, 6))  # 设置图形大小
plt.bar(categories, values, color='purple')  # 绘制柱状图
plt.title('Bar Plot of Categories')  # 图形标题
plt.xlabel('Categories')             # X 轴标签
plt.ylabel('Values')                 # Y 轴标签
plt.grid(axis='y')                   # 显示 Y 轴网格
plt.show()                           # 显示图形
```

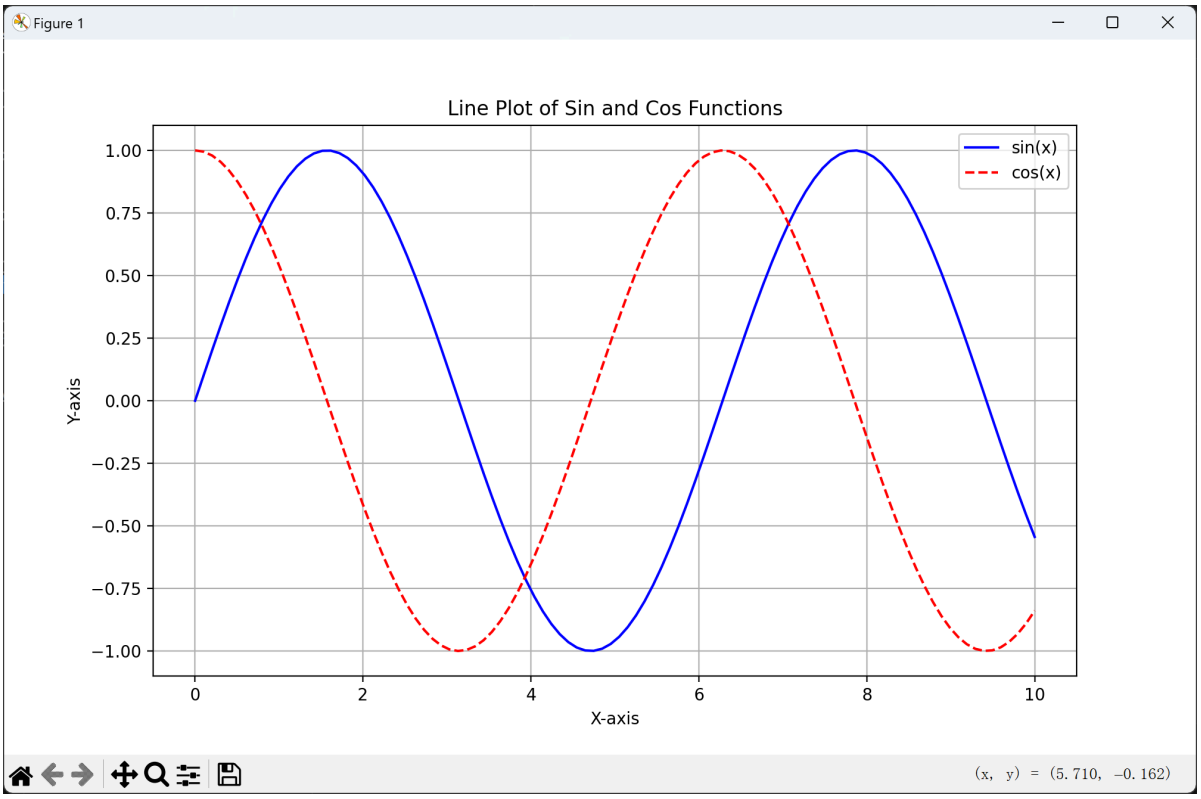


Figure 1
Line Plot of Sin and Cos Functions
sin(x)
cos(x)
Y-axis
X-axis
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
2
4
6
8
10
(x, y) = (5.710, −0.162)

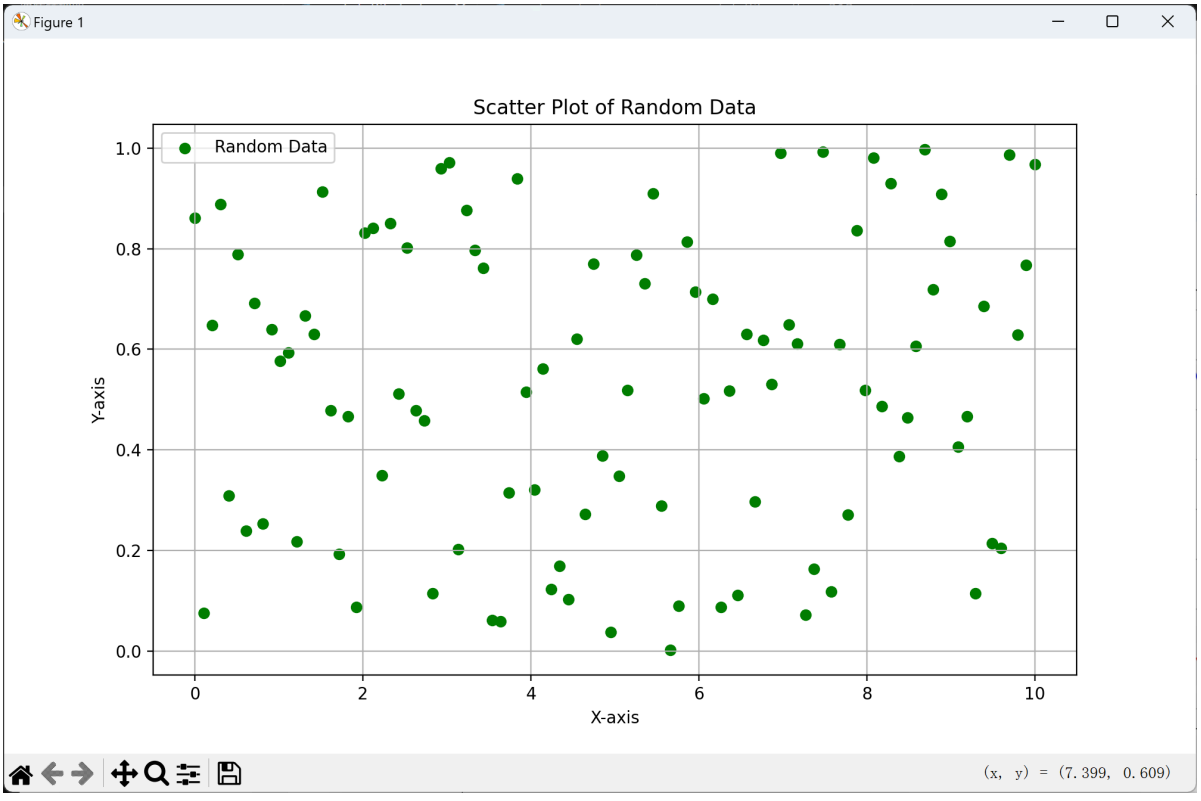
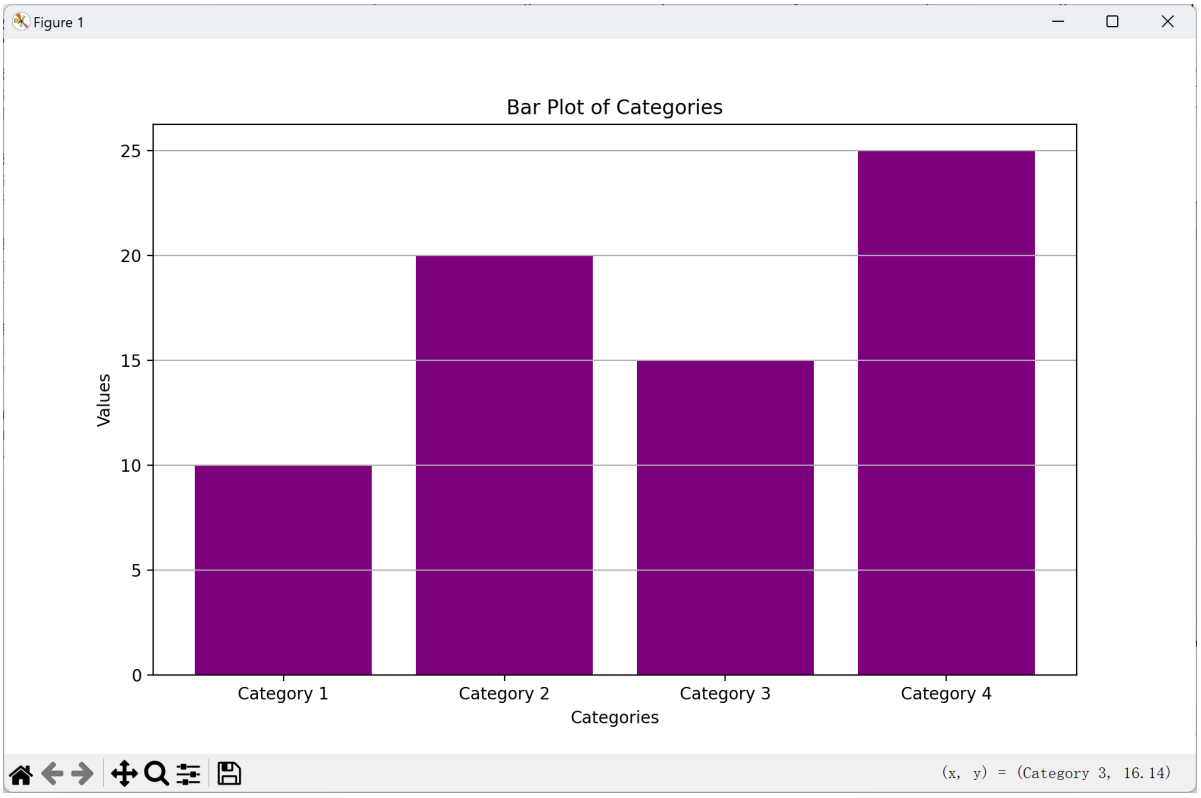


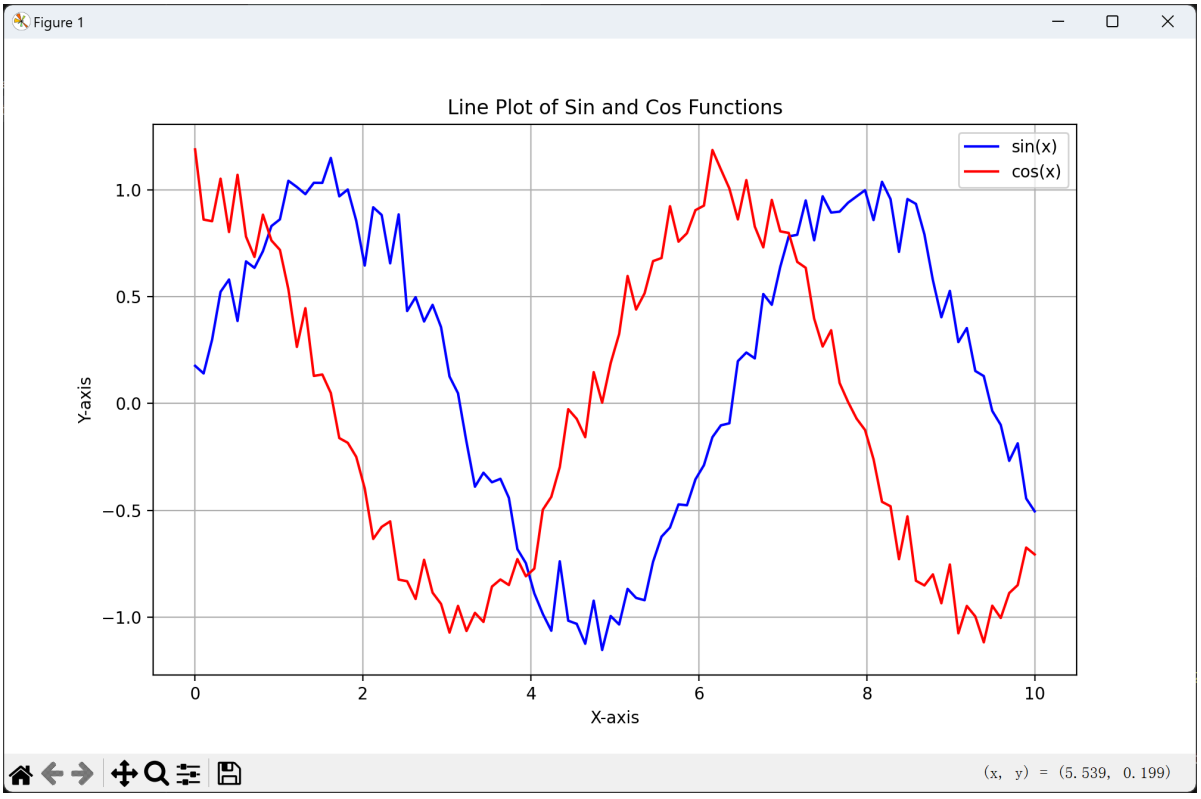
Figure 1
Scatter Plot of Random Data
Random Data
Y-axis
X-axis
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10
(x, y) = (7.399, 0.609)




# 8、熟悉可视化包Seaborn,绘制任一数据集的三种常见图形。

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

# 创建示例数据
np.random.seed(0)  # 设置随机种子以便结果可重现
x = np.linspace(0, 10, 100)  # x 值从 0 到 10，分成 100 个点
y1 = np.sin(x) + np.random.normal(0, 0.1, size=x.shape)  # y1 为 sin(x) 加上随机噪声
y2 = np.cos(x) + np.random.normal(0, 0.1, size=x.shape)  # y2 为 cos(x) 加上随机噪声

plt.figure(figsize=(10, 6))  # 设置图形大小
sns.lineplot(x=x, y=y1, label='sin(x)', color='b')  # 绘制 sin(x) 折线
sns.lineplot(x=x, y=y2, label='cos(x)', color='r')  # 绘制 cos(x) 折线
plt.title('Line Plot of Sin and Cos Functions')  # 图形标题
plt.xlabel('X-axis')                                # X 轴标签
plt.ylabel('Y-axis')                                # Y 轴标签
plt.legend()                                        # 显示图例
plt.grid()                                          # 显示网格
plt.show()                                          # 显示图形

# 创建散点图数据
x_scatter = np.random.rand(100) * 10  # 生成 100 个随机 x 值
y_scatter = np.random.rand(100) * 10  # 生成 100 个随机 y 值

plt.figure(figsize=(10, 6))  # 设置图形大小
sns.scatterplot(x=x_scatter, y=y_scatter, color='g', label='Random Data')  # 绘制散点图
plt.title('Scatter Plot of Random Data')  # 图形标题
```

```
plt.xlabel('X-axis')                          # X 轴标签
plt.ylabel('Y-axis')                          # Y 轴标签
plt.legend()                                  # 显示图例
plt.grid()                                    # 显示网格
plt.show()                                    # 显示图形

# 创建柱状图数据
categories = ['Category 1', 'Category 2', 'Category 3', 'Category 4']
values = [10, 20, 15, 25]

plt.figure(figsize=(10, 6))  # 设置图形大小
sns.barplot(x=categories, y=values, palette='viridis')  # 绘制柱状图
plt.title('Bar Plot of Categories')  # 图形标题
plt.xlabel('Categories')                      # X 轴标签
plt.ylabel('Values')                          # Y 轴标签
plt.grid(axis='y')                            # 显示 Y 轴网格
plt.show()                                    # 显示图形
```



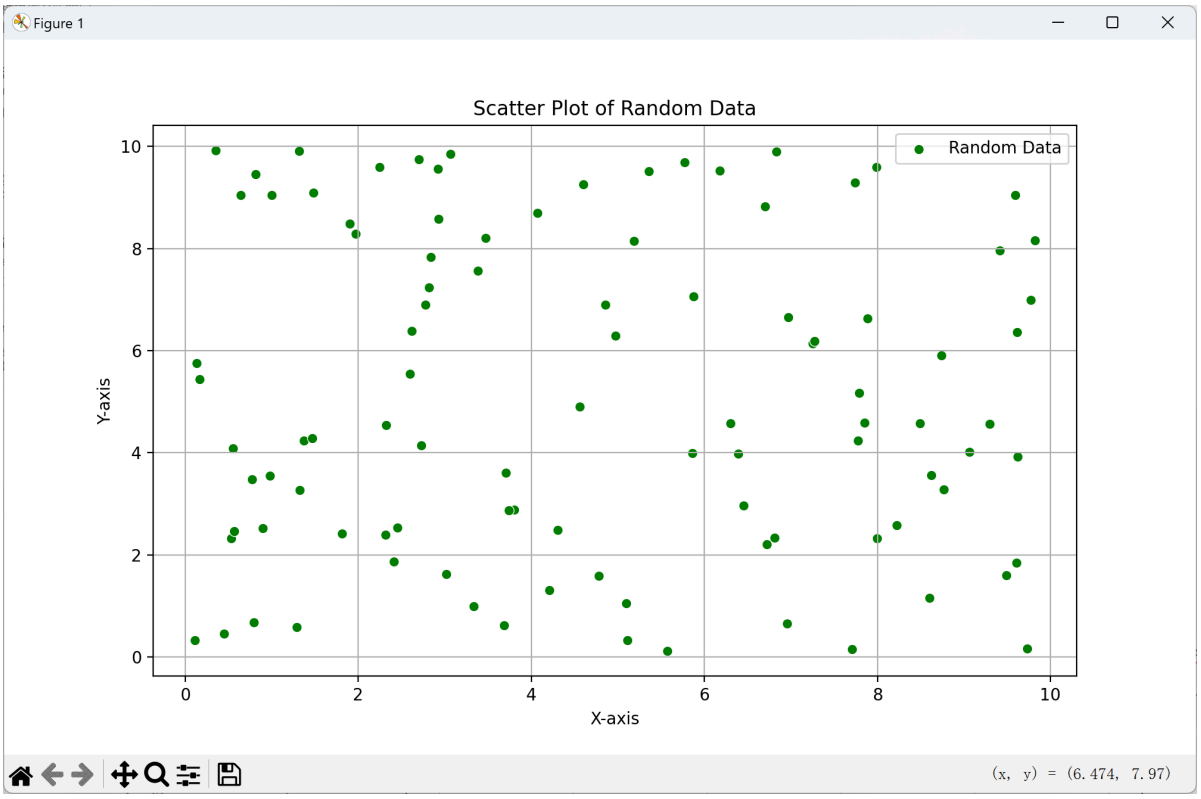


Figure 1
Scatter Plot of Random Data
Random Data
Y-axis
X-axis
0
2
4
6
8
10
(x, y) = (6.474, 7.97)

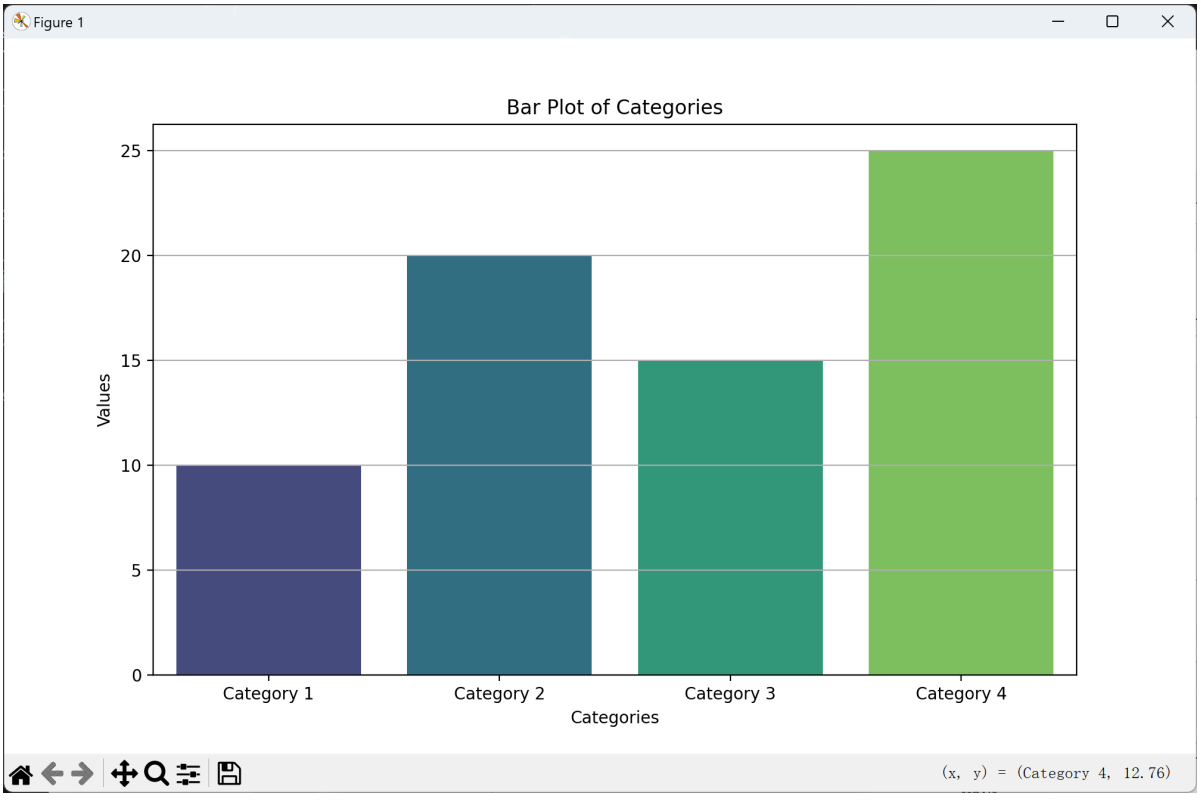


Figure 1
Bar Plot of Categories
Values
Categories
0
5
10
15
20
25
Category 1
Category 2
Category 3
Category 4
(x, y) = (Category 4, 12.76)