13：生成数据和数据可视化

安装Matplotlib

In [ ]:
```
%pip install matplotlib
```

绘制简单的折线图

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

squares = [1, 4, 9, 16, 25]

fig, ax = plt.subplots()

# squares的值作为y轴，x轴的数字默认从0开始的整数
ax.plot(squares)

plt.show()
```

修改标签文字和线条粗细

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

squares = [1, 4, 9, 16, 25]

fig, ax = plt.subplots()
ax.plot(squares, linewidth=3)

# Set chart title and label axes.
ax.set_title("Square Numbers", fontsize=24)
ax.set_xlabel("Value", fontsize=14)
ax.set_ylabel("Square of Value", fontsize=14)

# Set size of tick labels.
ax.tick_params(labelsize=14)

plt.show()
```

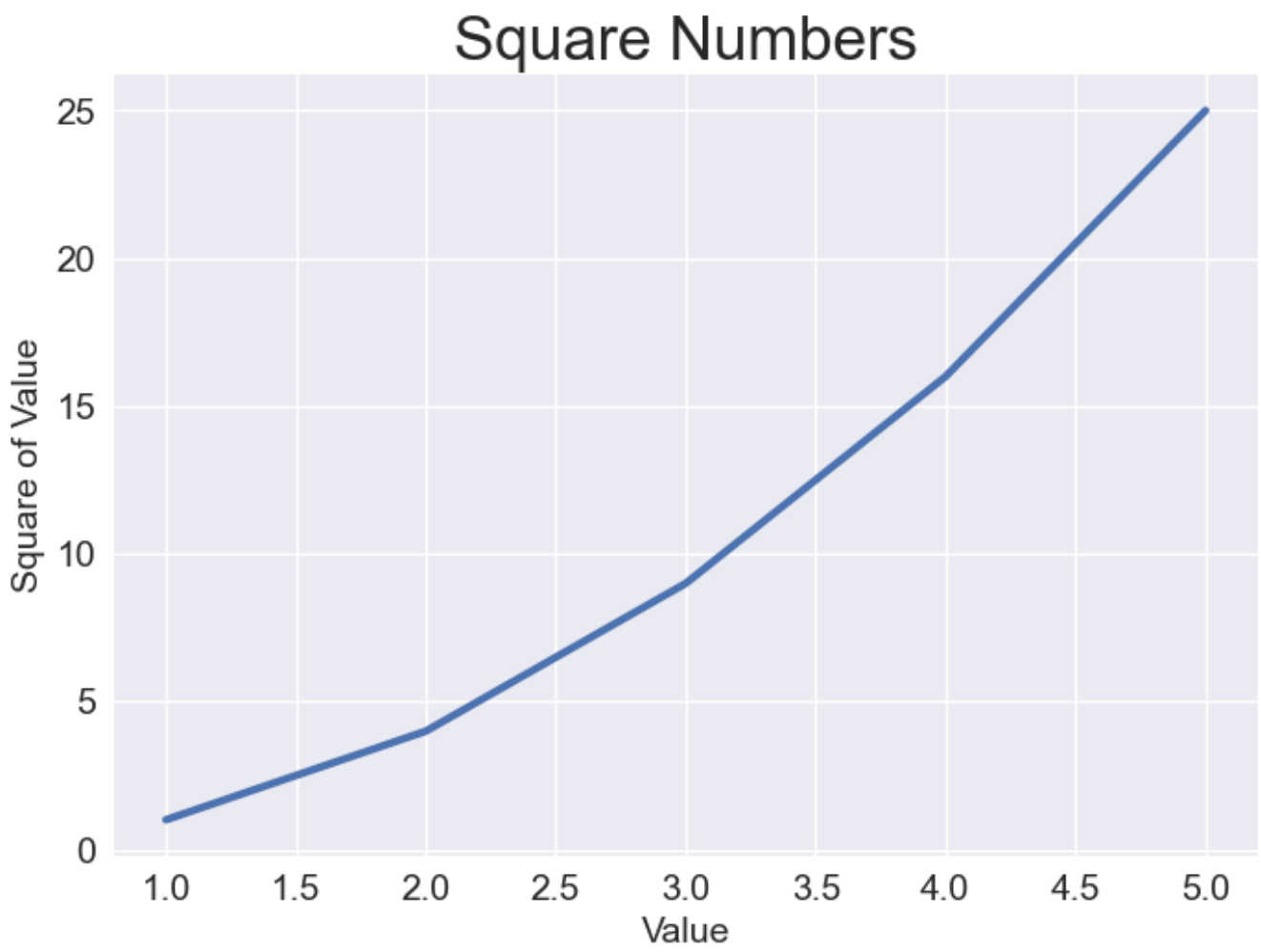
校正绘图

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

input_values = [1, 2, 3, 4, 5] # x轴的值
squares = [1, 4, 9, 16, 25]

fig, ax = plt.subplots()

ax.plot(input_values, squares, linewidth=3)

# Set chart title and label axes.
ax.set_title("Square Numbers", fontsize=24)
ax.set_xlabel("Value", fontsize=14)
ax.set_ylabel("Square of Value", fontsize=14)

# Set size of tick labels.
ax.tick_params(labelsize=14)

plt.show()
```

使用内置样式

In [ ]:
```
import matplotlib.pyplot as plt
plt.style.available
```

In [2]:
```
import matplotlib.pyplot as plt

input_values = [1, 2, 3, 4, 5]
squares = [1, 4, 9, 16, 25]

plt.style.use('seaborn-v0_8')
fig, ax = plt.subplots()
ax.plot(input_values, squares, linewidth=3)

# Set chart title and label axes.
ax.set_title("Square Numbers", fontsize=24)
ax.set_xlabel("Value", fontsize=14)
ax.set_ylabel("Square of Value", fontsize=14)

# Set size of tick labels.
ax.tick_params(labelsize=14)

plt.show()
```



绘制散点图并设置样式

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

plt.style.use('seaborn-v0_8')
fig, ax = plt.subplots()
ax.scatter(2, 4)

plt.show()
```

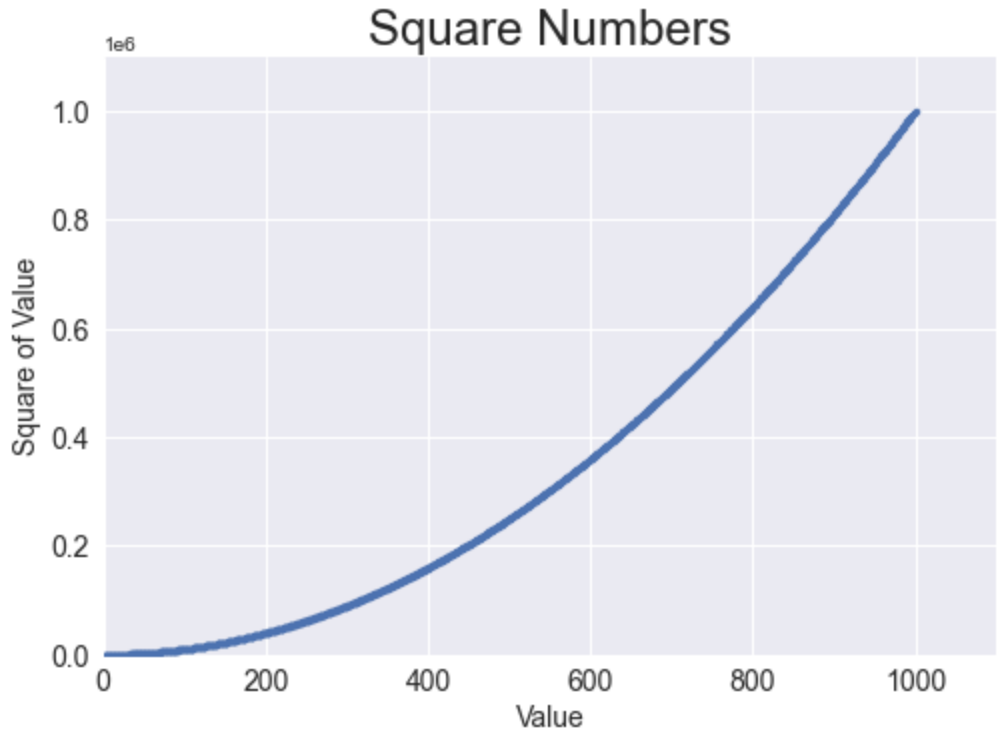
In [ ]:
```
import matplotlib.pyplot as plt

plt.style.use('seaborn-v0_8')
fig, ax = plt.subplots()
ax.scatter(2, 4, s=200)

# Set chart title and label axes.
ax.set_title("Square Numbers", fontsize=24)
ax.set_xlabel("Value", fontsize=14)
ax.set_ylabel("Square of Value", fontsize=14)

# Set size of tick labels.
ax.tick_params(labelsize=14)

plt.show()
```

使用scatter()绘制一系列的点

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

x_values = [1, 2, 3, 4, 5]
y_values = [1, 4, 9, 16, 25]

plt.style.use('seaborn-v0_8')
fig, ax = plt.subplots()
ax.scatter(x_values, y_values, s=100)

# Set chart title and label axes.
ax.set_title("Square Numbers", fontsize=24)
ax.set_xlabel("Value", fontsize=14)
ax.set_ylabel("Square of Value", fontsize=14)

# Set size of tick labels.
ax.tick_params(labelsize=14)

plt.show()
```

自动计算数据

In [14]:
```
import matplotlib.pyplot as plt

x_values = range(1, 1001)
y_values = [x**2 for x in x_values]

plt.style.use('seaborn-v0_8')
fig, ax = plt.subplots()
ax.scatter(x_values, y_values, s=10)

# Set chart title and label axes.
ax.set_title("Square Numbers", fontsize=24)
ax.set_xlabel("Value", fontsize=14)
ax.set_ylabel("Square of Value", fontsize=14)

# Set size of tick labels.
ax.tick_params(labelsize=14)

# Set the range for each axis.
ax.axis([0, 1100, 0, 1_100_000])

plt.show()
```

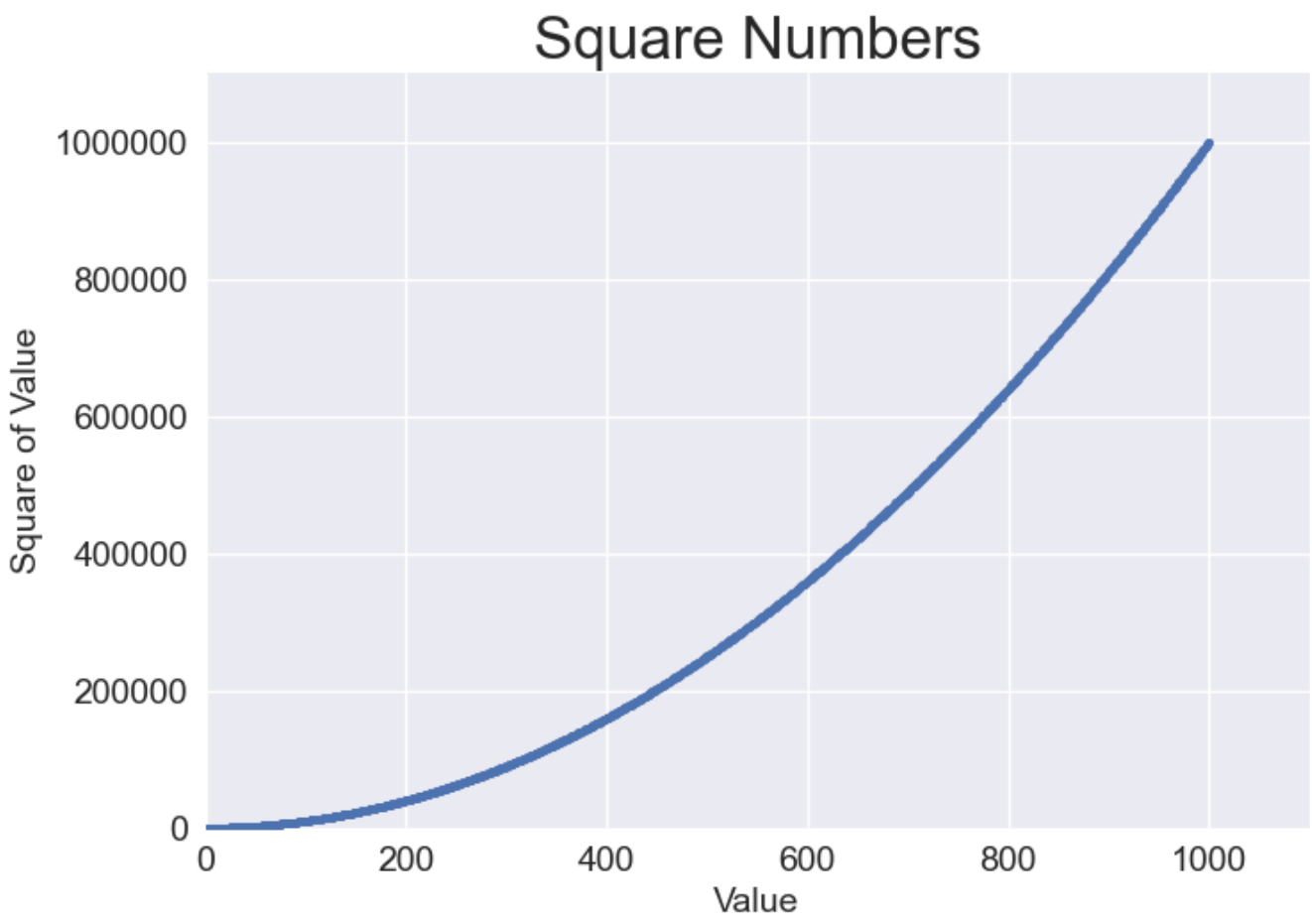


使用Numpy来计算

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

x_values = np.arange(1, 1001)
y_values = x_values**2  # 利用numpy的广播特性

plt.style.use('seaborn-v0_8')
fig, ax = plt.subplots()
ax.scatter(x_values, y_values, s=10)

# Set chart title and label axes.
ax.set_title("Square Numbers", fontsize=24)
ax.set_xlabel("Value", fontsize=14)
ax.set_ylabel("Square of Value", fontsize=14)

# Set size of tick labels.
ax.tick_params(labelsize=14)

# Set the range for each axis.
ax.axis([0, 1100, 0, 1_100_000])

plt.show()
```

定制刻度标记

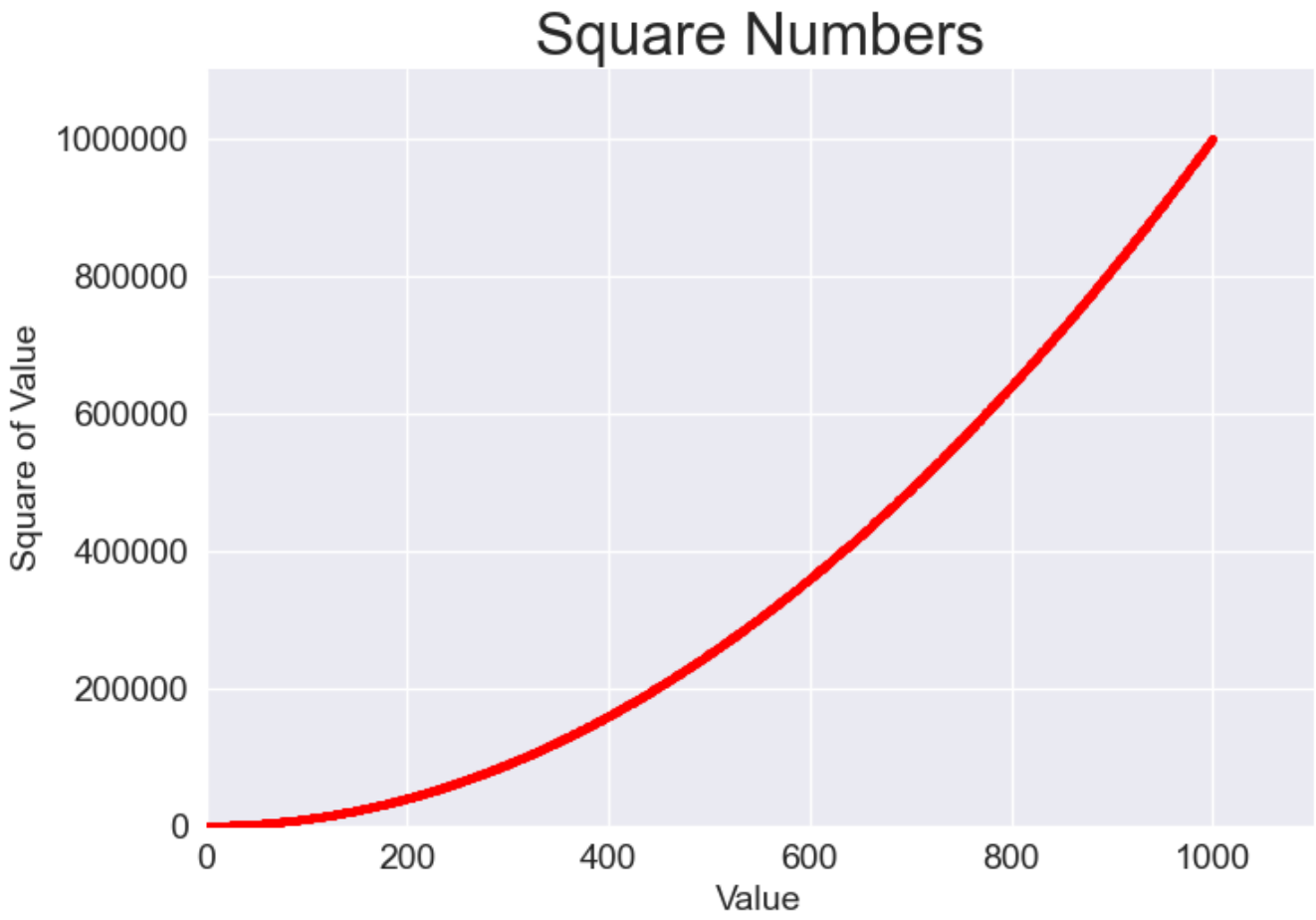
In [4]:
```
import matplotlib.pyplot as plt
import numpy as np

x_values = np.arange(1, 1001)
y_values = x_values**2

plt.style.use('seaborn-v0_8')
fig, ax = plt.subplots()
ax.scatter(x_values, y_values, s=10)

# Set chart title and label axes.
ax.set_title("Square Numbers", fontsize=24)
ax.set_xlabel("Value", fontsize=14)
ax.set_ylabel("Square of Value", fontsize=14)

# Set size of tick labels.
ax.tick_params(labelsize=14)

# Set the range for each axis.
ax.axis([0, 1100, 0, 1_100_000])
ax.ticklabel_format(style='plain')

plt.show()
```



定制颜色

In [7]:
```
import matplotlib.pyplot as plt
import numpy as np

x_values = np.arange(1, 1001)
y_values = x_values**2

plt.style.use('seaborn')
fig, ax = plt.subplots()
ax.scatter(x_values, y_values, s=10, color='red')

# 使用颜色映射，根据y值的大小，颜色越深
# ax.scatter(x_values, y_values, s=10, c=y_values, cmap=plt.cm.Blues)

# Set chart title and label axes.
ax.set_title("Square Numbers", fontsize=24)
ax.set_xlabel("Value", fontsize=14)
ax.set_ylabel("Square of Value", fontsize=14)

# Set size of tick labels.
ax.tick_params(labelsize=14)

# Set the range for each axis.
ax.axis([0, 1100, 0, 1_100_000])
ax.ticklabel_format(style='plain')

plt.show()
```

```
C:\Users\zhouj\AppData\Local\Temp\ipykernel_41964\3133603459.py:7: MatplotlibDeprecationWarnin
g: The seaborn styles shipped by Matplotlib are deprecated since 3.6, as they no longer corresp
ond to the styles shipped by seaborn. However, they will remain available as 'seaborn-v0_8-<sty
le>'. Alternatively, directly use the seaborn API instead.
  plt.style.use('seaborn')
```



随机游走

什么是随机游走

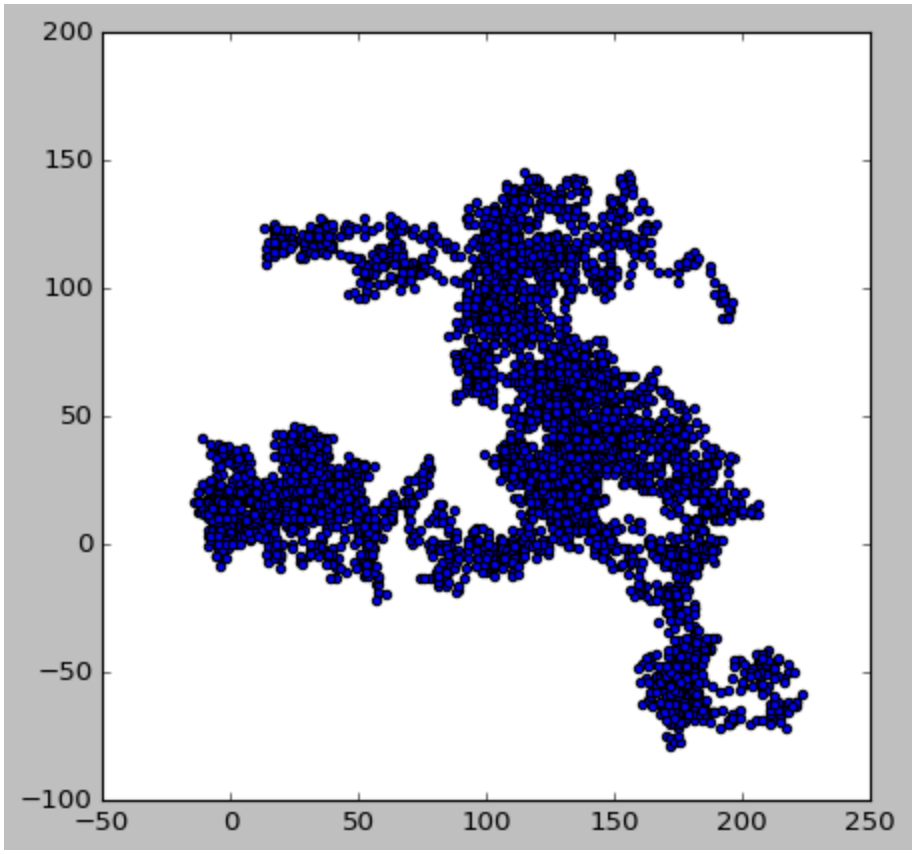
- 😥 随机游走是一种数学统计模型，它是一连串的轨迹所组成，其中每一次都是随机的。它能用来表示不规则的变动形式，如同一个人酒后乱步，所形成的随机过程记录。1905年，由卡尔·皮尔逊首次提出。
- 🔬 随机游走在各个领域有许多应用，例如在工程学和许多科学领域，包括生态学，心理学，计算机科学，物理，化学，生物学以及经济学。
  - 在数学中，我们可以用个体为本模型的随机游走来估算π的值
  - 模拟分子在液体或气体中传播时的路径
  - 觅食动物的搜索路径
  - 波动的股票价格
  - 赌徒的财务状况

Randomwalk类

In [29]:
```
from random import choice


class RandomWalk:
    """A class to generate random walks."""

    def __init__(self, num_points=5000):
        """Initialize attributes of a walk."""
        self.num_points = num_points

        # All walks start at (0, 0).
        self.x_values = [0]
        self.y_values = [0]

    def fill_walk(self):
        """Calculate all the points in the walk."""
        # Keep taking steps until the walk reaches the desired length.
        while len(self.x_values) < self.num_points:

            # Decide which direction to go, and how far to go.
            x_direction = choice([1, -1])
            x_distance = choice([0, 1, 2, 3, 4])
            x_step = x_direction * x_distance

            y_direction = choice([1, -1])
            y_distance = choice([0, 1, 2, 3, 4])
            y_step = y_direction * y_distance

            # Reject moves that go nowhere.
            if x_step == 0 and y_step == 0:
                continue

            # Calculate the new position.
            x = self.x_values[-1] + x_step
            y = self.y_values[-1] + y_step

            self.x_values.append(x)
            self.y_values.append(y)
```

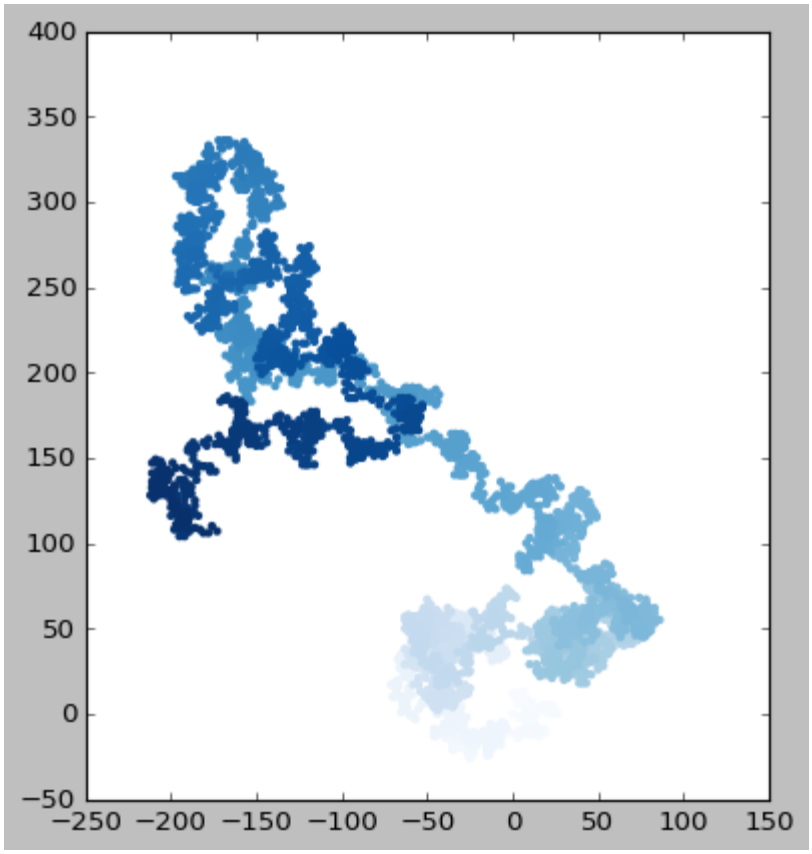
绘制随机游走图

In [34]:
```
import matplotlib.pyplot as plt

# Make a random walk.
rw = RandomWalk()
rw.fill_walk()

# Plot the points in the walk.
plt.style.use('classic')
fig, ax = plt.subplots()
ax.scatter(rw.x_values, rw.y_values, s=15)
ax.set_aspect('equal')
plt.show()
```



设置随机游走图的样式

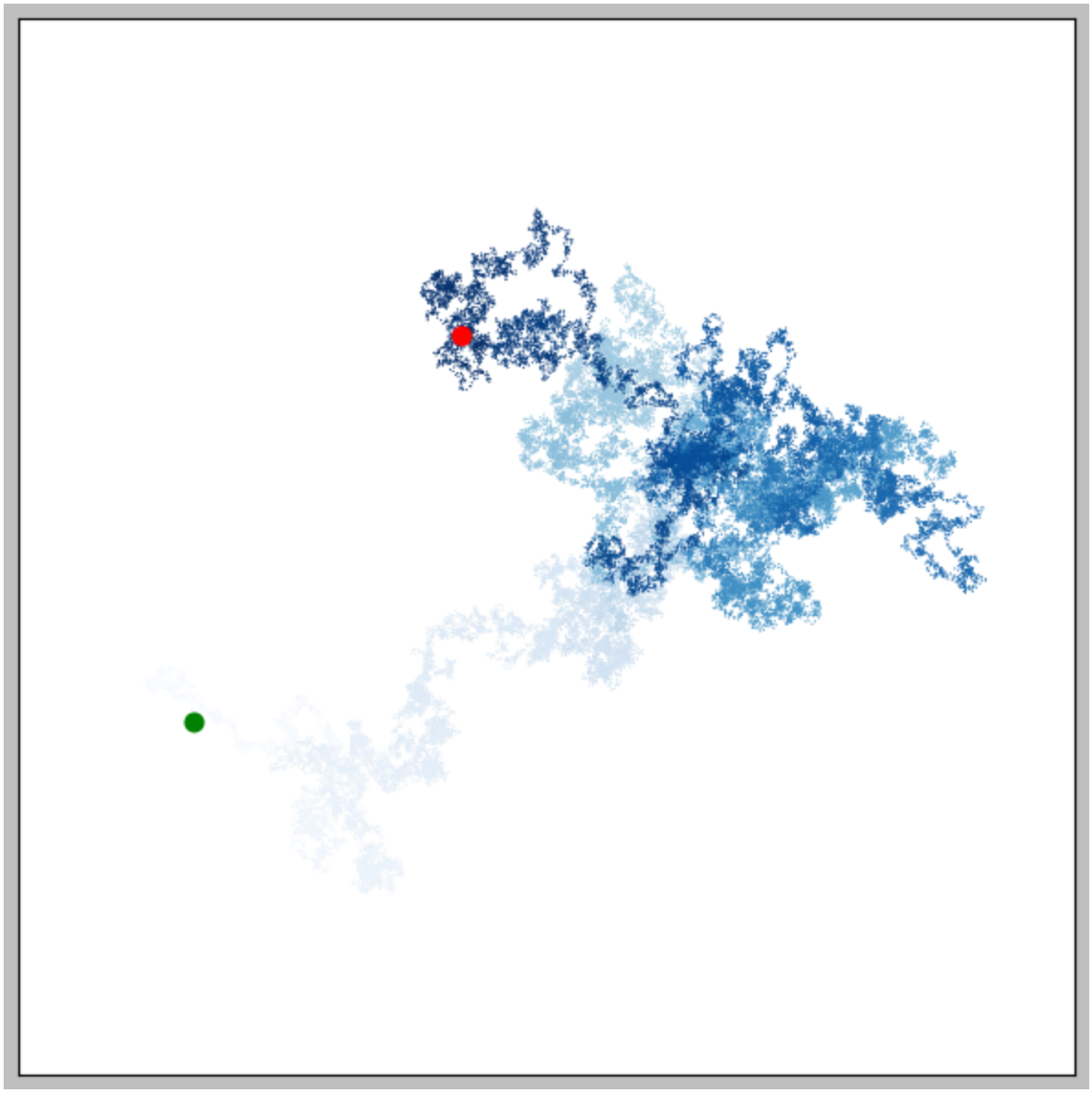
In [36]:
```
rw = RandomWalk()
rw.fill_walk()

# Plot the points in the walk.
plt.style.use('classic')
fig, ax = plt.subplots()
point_numbers = range(rw.num_points)
ax.scatter(rw.x_values, rw.y_values, c=point_numbers, cmap=plt.cm.Blues,
    edgecolors='none', s=15)
ax.set_aspect('equal')
plt.show()
```



继续修改随机游走图的样式

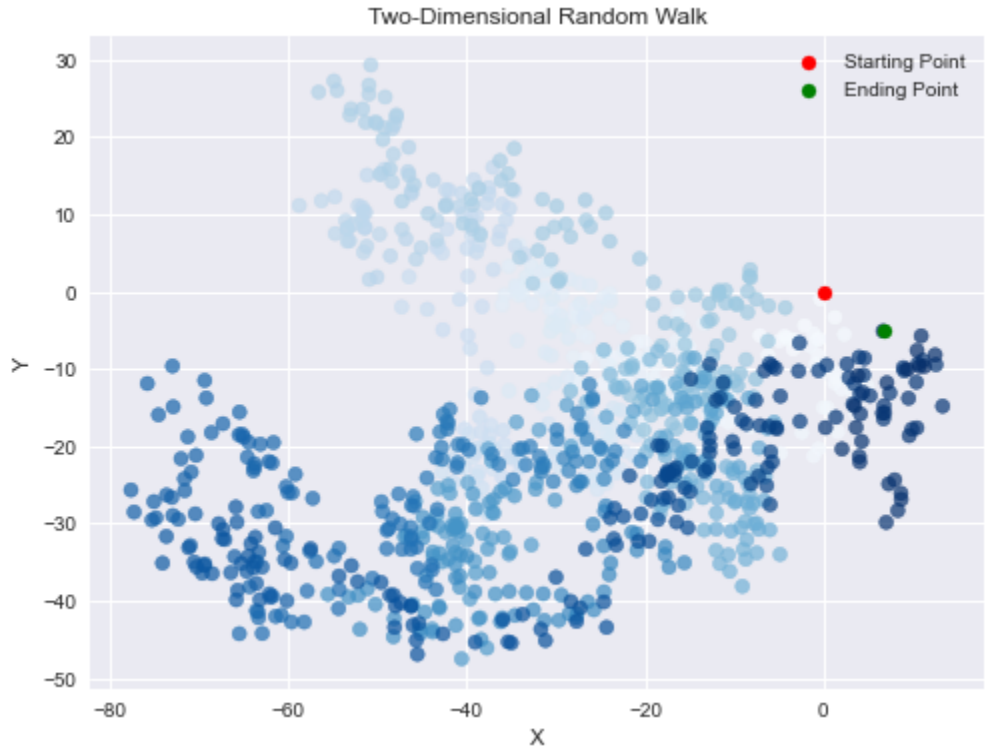
- 绘制起点和终点
- 隐藏坐标轴
- 增加点的个数
- 调整图的尺寸

In [43]:
```
rw = RandomWalk(50_000)
rw.fill_walk()

# Plot the points in the walk.
plt.style.use('classic')
fig, ax = plt.subplots(figsize = (15, 9))
point_numbers = range(rw.num_points)
ax.scatter(rw.x_values, rw.y_values, c=point_numbers, cmap=plt.cm.Blues,
    edgecolors='none', s=1)
ax.set_aspect('equal')

# Emphasize the first and last points.
ax.scatter(0, 0, c='green', edgecolors='none', s=100)
ax.scatter(rw.x_values[-1], rw.y_values[-1], c='red', edgecolors='none',
    s=100)

# Remove the axes.
ax.get_xaxis().set_visible(False)
ax.get_yaxis().set_visible(False)

plt.show()
```



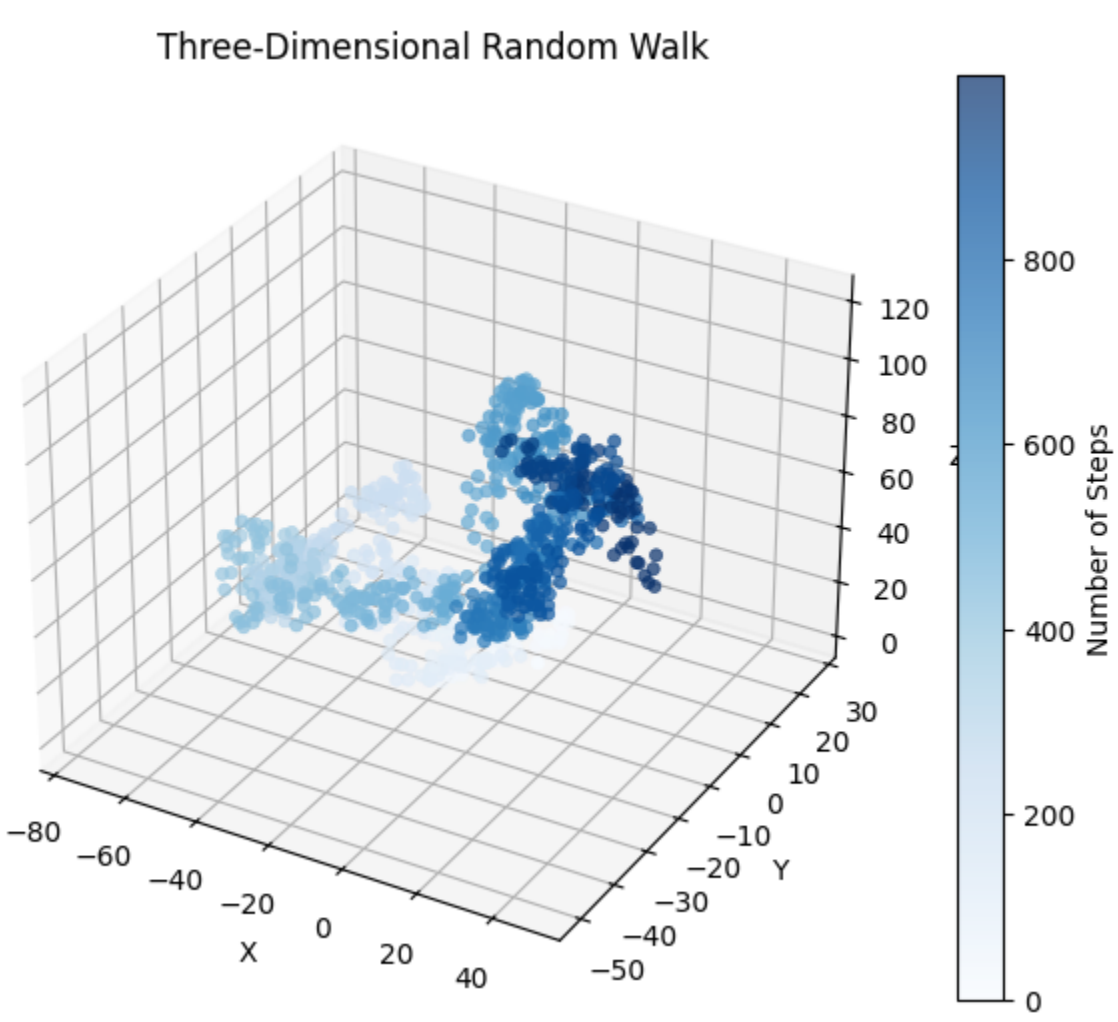
使用Numpy来实现随机游走

In [27]:
```
import numpy as np
import matplotlib.pyplot as plt

# Define the number of steps
num_steps = 1000

# Generate random step distances in x and y directions
step_distances = np.random.uniform(-4, 4, size=(num_steps, 2))

# Calculate cumulative sum of step distances
positions = np.cumsum(step_distances, axis=0)

# Extract x and y coordinates
x = positions[:, 0]
y = positions[:, 1]

# Calculate color map based on number of steps
colors = np.arange(num_steps)

# Plot the random walk
plt.figure(figsize=(8, 6))
plt.scatter(x, y, c=colors, cmap='Blues', linewidths=0.5, alpha=0.7)
plt.scatter(0, 0, color='red', marker='o', label='Starting Point')
plt.scatter(x[-1], y[-1], color='green', marker='o', label='Ending Point')
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Two-Dimensional Random Walk')
plt.legend()
plt.grid(True)
plt.show()
```



三维的随机游走

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Set the random seed for reproducibility
np.random.seed(0)

# Define the number of steps
num_steps = 1000

# Generate random step distances in x, y, and z directions
step_distances = np.random.uniform(-4, 4, size=(num_steps, 3))

# Calculate cumulative sum of step distances
positions = np.cumsum(step_distances, axis=0)

# Extract x, y, and z coordinates
x = positions[:, 0]
y = positions[:, 1]
z = positions[:, 2]

# Calculate color map based on number of steps
colors = np.arange(num_steps)

# Plot the random walk
fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, projection='3d')
scatter = ax.scatter(x, y, z, c=colors, cmap='Blues', linewidths=0.5, alpha=0.7)
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.set_title('Three-Dimensional Random Walk')
fig.colorbar(scatter, label='Number of Steps')
plt.show()
```



使用Plotly模拟投掷骰子

安装Plotly

In [ ]:
```
%pip install plotly
```

创建Die类

In [1]:
```
from random import randint

class Die:
    """A class representing a single die."""

    def __init__(self, num_sides=6):
        """Assume a six-sided die."""
        self.num_sides = num_sides

    def roll(self):
        """"Return a random value between 1 and number of sides."""
        return randint(1, self.num_sides)
```

掷骰子

In [2]:
```
# Create a D6.
die = Die()

# Make some rolls, and store results in a list.
results = []
for roll_num in range(100):
    result = die.roll()
    results.append(result)

# results = [die.roll() for _ in range(100)]

print(results)
```

```
[2, 4, 6, 4, 1, 5, 4, 6, 3, 5, 6, 3, 4, 5, 3, 5, 3, 4, 2, 5, 2, 6, 4, 5, 6, 3, 2, 4, 6, 6, 2,
4, 1, 2, 3, 1, 5, 2, 5, 5, 3, 1, 3, 6, 5, 6, 2, 6, 3, 3, 5, 4, 1, 3, 5, 5, 2, 6, 6, 6, 3, 2, 6,
3, 5, 1, 5, 2, 5, 6, 3, 3, 3, 3, 1, 1, 2, 2, 6, 2, 2, 6, 3, 6, 5, 2, 6, 1, 1, 2, 3, 2, 6, 2, 4,
3, 4, 1, 1, 5]
```

分析结果

In [3]:
```
frequencies = []
poss_results = range(1, die.num_sides+1)
for value in poss_results:
    frequency = results.count(value)
    frequencies.append(frequency)

print(frequencies)
```

```
[12, 19, 20, 11, 18, 20]
```

绘制直方图

In [4]:
```
import plotly.express as px

# Create a D6.
die = Die()

# Make some rolls, and store results in a list.
results = []
for roll_num in range(1000):
    result = die.roll()
    results.append(result)

# Analyze the results.
frequencies = []
poss_results = range(1, die.num_sides+1)
for value in poss_results:
    frequency = results.count(value)
    frequencies.append(frequency)

# Visualize the results.
fig = px.bar(x=poss_results, y=frequencies)
fig.show()
```

定制绘图

In [5]:
```
import plotly.express as px

# Create a D6.
die = Die()

# Make some rolls, and store results in a list.
results = []
for roll_num in range(1000):
    result = die.roll()
    results.append(result)

# Analyze the results.
frequencies = []
poss_results = range(1, die.num_sides+1)
for value in poss_results:
    frequency = results.count(value)
    frequencies.append(frequency)

# Visualize the results.
title = "Results of Rolling One D6 1,000 Times"
labels = {'x': 'Result', 'y': 'Frequency of Result'}
fig = px.bar(x=poss_results, y=frequencies, title=title, labels=labels)
fig.show()
```

使用Numpy来实现掷骰子

- 面向数组的编程范式，利用数组的 **广播**特性，将数组看作一个整体来进行操作
- 避免使用循环

In [6]:
```
import numpy as np
import plotly.express as px

num_sides = 6

# Make some rolls, and store results in a NumPy array.
results = np.random.randint(1, num_sides + 1, size=1000)

# Analyze the results using NumPy functions.
poss_results = np.arange(1, num_sides + 1)
frequencies = np.bincount(results, minlength=num_sides+1)[1:]

# Visualize the results.
title = "Results of Rolling One D6 1,000 Times"
labels = {'x': 'Result', 'y': 'Frequency of Result'}
fig = px.bar(x=poss_results, y=frequencies, title=title, labels=labels)
fig.show()
```

同时投掷两个骰子

In [7]:
```
# Create two D6 dice.
die_1 = Die()
die_2 = Die()

# Make some rolls, and store results in a list.
results = []
for roll_num in range(1000):
    result = die_1.roll() + die_2.roll()
    results.append(result)

# Analyze the results.
frequencies = []
max_result = die_1.num_sides + die_2.num_sides
poss_results = range(2, max_result+1)
for value in poss_results:
    frequency = results.count(value)
    frequencies.append(frequency)

# Visualize the results.
title = "Results of Rolling Two D6 Dice 1,000 Times"
labels = {'x': 'Result', 'y': 'Frequency of Result'}
fig = px.bar(x=poss_results, y=frequencies, title=title, labels=labels)

# Further customize chart.
fig.update_layout(xaxis_dtick=1)
fig.show()
```

In [8]:
```
import numpy as np
import plotly.express as px

num_sides1 = 6
num_sides2 = 6
num_sides = num_sides1 + num_sides2

# Make some rolls, and store results in a NumPy array.
results1 = np.random.randint(1, num_sides1 + 1, size=1000)
results2 = np.random.randint(1, num_sides2 + 1, size=1000)
results = results1 + results2

# Analyze the results using NumPy functions.
poss_results = np.arange(2, num_sides + 1)
frequencies = np.bincount(results, minlength=num_sides+1)[2:]

# Visualize the results.
title = "Results of Rolling One D6 1,000 Times"
labels = {'x': 'Result', 'y': 'Frequency of Result'}
fig = px.bar(x=poss_results, y=frequencies, title=title, labels=labels)

fig.update_layout(xaxis_dtick=1)
fig.show()
```

Plotly Express文档

Plotly Express文档 (https://plotly.com/python/plotly-express/)

小结

设计良好的图表是人的大脑接收信息效率最高、信息带宽最高的一种方式

- 通过数据可视化，我们可以更好地理解数据
- 通过数据可视化，我们可以更好地与数据交互
- 通过数据可视化，我们可以更好地向他人展示数据

推荐书籍：storytelling with data

cole nussbaumer knaflic

storytelling with data