实验课程：概率论与数理统计

实验名称：概率论与数理统计实验

第一部分 实验内容

1．实验目标

掌握 python 基本开发技能，学习、理解概率论的相关知识，能够对概率论的相关模型进行实验验证并进行可视化展示，加深对相关知识点的直观认识。

2. 实验任务

(1) 采用可视化设计，有菜单界面。

(2) 利用蒙特卡洛方法计算圆周率并展示结果。

(3) 验证泊松定理并展示，对于泊松分布固定的λ，随着二项分布 n 的增加，二项分布逐渐收敛于泊松分布。

(4) 给定参数μ，σ，展示对应的正态分布概率密度图；通过动态调整参数μ或σ，展示图像的变化。

(5) 生成正态分布的样本，验证大数定律。画图展示随着样本容量的增加，随机变量的算术平均依概率收敛到数学期望。

3. 实验设备及环境

开发环境：Python 3.7 tkinter + Numpy + Scipy + matlibplot

4. 实验主要步骤

(1) 根据实验目标，明确实验的具体任务；

(2) 将任务细分为多个小任务，简化实验。

(3) 利用网络资源学习相关代码知识，扩展自己的知识技能；

(4) 设计求解问题的流程图，并编写程序实现算法；

(5) 不断调试代码，完善实验结果；

(6) 实验后的心得体会。

第二部分 问题及算法

1. 问题描述

编写一个程序，这个程序有以下五个功能：

(1) 采用可视化设计，并有菜单界面。

(2) 利用蒙特卡洛方法计算圆周率，并展示结果。

(3) 验证泊松定理，并展示，对于泊松分布固定的 λ，随着二项分布 n 的增加，二项分布逐渐收敛于泊松分布。

(4) 给定参数 μ，σ，并展示对应的正态分布概率密度图；通过动态调整参数 μ 或 σ，展示图像的变化。

(5) 生成正态分布的样本，并验证大数定律。画图展示随着样本容量的增加，随机变量的算术平均依概率收敛到数学期望。

2. 算法的一般思路

将程序模块化，提高代码的可读性，定义六个函数，前五个分别实现程序的五个功能，最后一个用来查看这个程序的制作者和版本。

3. 求解问题的算法描述

root()函数，作为菜单设置函数，实现可视化，建立了一个带有菜单的窗口，包括创建窗口、定义菜单及子菜单、配置菜单栏、添加标签和命令等功能。创建一个窗口，并设置窗口名称和大小；定义菜单栏 bar，将其添加到窗口 root 中；创建四个子菜单 bar1、bar2、bar3 和 bar4，设置 tearoff=0 使子菜单不可撕下； 将子菜单添加到菜单栏 bar 中，使用 add_cascade 方法级联子菜单和上层菜单。在子菜单中添加标签和命令，使用 add_command 方法；可以设置快捷键、下划线等属性；菜单栏 bar 中添加关于命令，调用 state 函数；使用 root.mainloop()显示窗口。

bar1_label（）函数解决第一个问题，即利用蒙特卡洛方法计算圆周率并绘制结果图像。创建一个新的窗口，并设置窗口的标题和大小；定义一个名为 bar1_label_0 的函数，用于在窗口上显示随机点的数目，默认为 1000；该函数首先尝试将输入的随机点数目向下取整，如果失败则使用默认值 1000；设置画布，并将画布显示在窗口上；在窗口上添加输入框和按钮，用于输入随机点数目和开始演示；清空原来的图像，然后绘制两个子图；第一个子图

用于显示随机点在圆和正方形内的分布，第二个子图用于显示圆周率的理论值和模拟计算值随实验次数的变化；生成随机点的横纵坐标，并进行实验；计算落在圆内的随机点数目，并添加到 pi 列表中；绘制正方形和圆，设置数轴刻度，并创建字体和图例；在第二个子图中绘制圆周率的理论值和模拟计算值，设置数轴上下限和刻度，并创建字体和图例；调用 bar1_label_0 函数绘制图像，并显示窗口。

bar2_lavel()函数，用于绘制实验 2 的图像。定义了一个名为 jiechen()的函数，用于计算阶乘；创建一个新的窗口 root2，并设置窗口的标题和大小；定义了一个名为 bar2_lavel_0()的函数，用于绘制图像和添加交互式元素；在 bar2_lavel_0()函数中，首先对输入的参数进行处理，并设置了一个开始演示的按钮，以便用户可以开始实验；接下来，使用 matplotlib 库绘制了两个子图，分别表示二项分布和泊松分布；最后，调用 bar2_lavel_0()函数，并显示窗口。

bar3_lavel（）函数，用于绘制正态分布概率密度图，用户可以通过调整均值（μ）和方差（σ^2）参数来观察图像的变化。创建一个新窗口 root3，并设置窗口标题和大小；定义一个嵌套函数 bar3_lavel_0，用于绘制正态分布概率密度图。该函数接受两个参数：均值（miu）和方差（sigema2），默认值分别为 0.00 和 1.00；在 bar3_lavel_0 函数中，首先尝试将输入的均值和方差转换为数值，如果失败则使用默认值；创建一个图形对象 figure3，将其显示在窗口上；在窗口上添加两个输入框（entry1 和 entry2），分别用于输入均值和方差；在窗口上添加三个按钮，分别用于调整均值、方差和开始演示；在窗口上添加两个标签，分别显示均值和方差；使用 matplotlib 库绘制正态分布概率密度图，其中 x 轴表示随机变量，y 轴表示概率密度； 调用 bar3_lavel_0 函数绘制初始图像；使用 mainloop()函数显示窗口。

bar4_lavel（）函数，用于绘制一个展示大数定律的图像。创建一个名为 root4 的新窗口，并设置窗口标题和大小；定义一个名为 bar4_lavel_0 的内部函数，用于绘制图像。这个函数接受三个参数：n（评委人数，默认值为 200），miu（数学期望，默认值为 85），sigema2（方差，默认值为 4）；在 bar4_lavel_0 函数中，首先尝试将输入的参数转换为数值，如果转换失败，则使用默认值； 创建一个名为 figure4 的图像，并将其添加到 root4 窗口中； 在窗口中添加三个输入框，分别用于输入评委人数、数学期望、方差； 在窗口中添加一个按钮，点击时调用 bar4_lavel_0 函数，并将输入框中的值传入；在窗口中添加三个标签，分别显示评委人数、数学期望、方差； 使用 matplotlib 库绘制图像。首先清空当前图像，然后生成 x_list（评委编号）和 y_list1（评委打分），接着计算 y_list2（前 n 项均值）； 使用 plt.bar()绘制柱状图，表示评委打分；使用 plt.scatter()绘制散点图，表示评委打分；使用 plt.plot()绘制折线图，表示前 n 项均值和数学期望； 设置图例、坐标轴范围等图像属性；调用 bar4_lavel_0 函数并运行窗口。

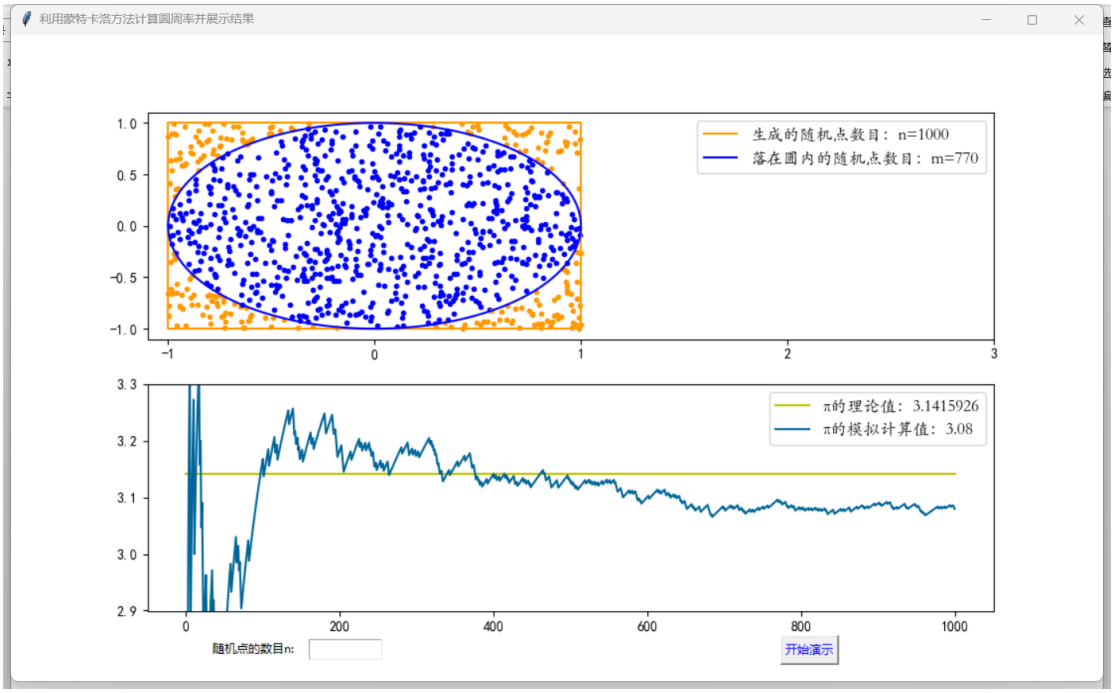
state（）函数，这个函数创建了一个窗口，显示了版本和制作者的信息。窗口使用了 tkinter 库来创建。

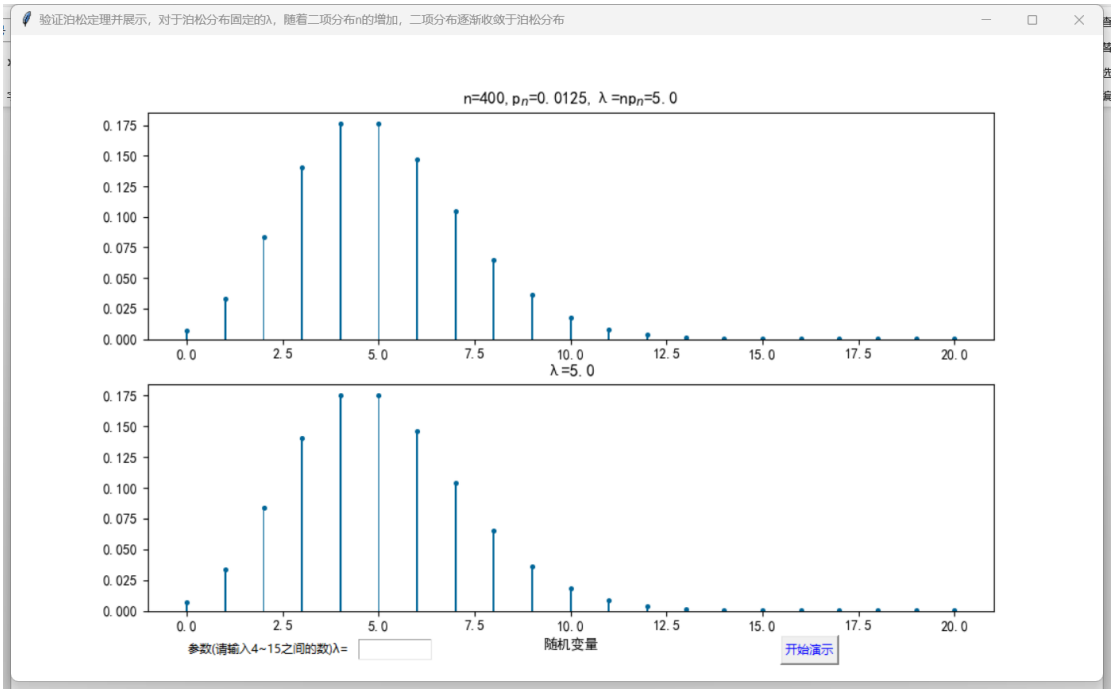
4. 算法实现的关键技巧

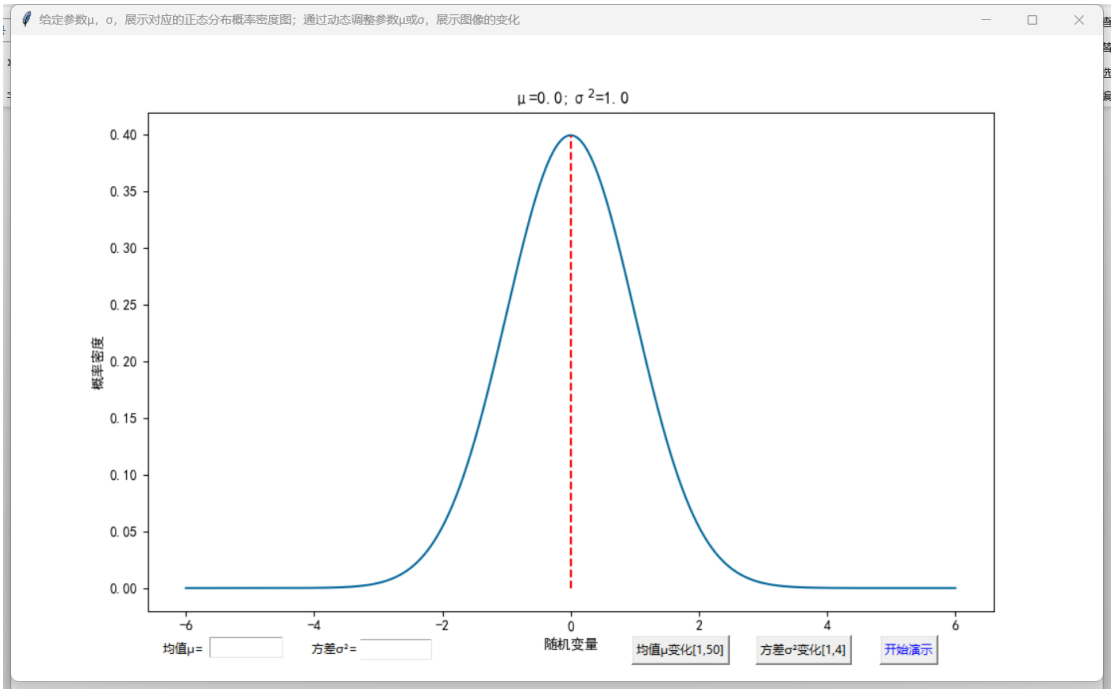
使用 tkinter 库创建窗口，再使用 matplotlib.pyplot 绘制图像，并利用 matplotlib. Backends. Backend _ tkagg 里的 FigureCanvasTkAgg 使图像显示在窗口上，再在窗口上添加按钮标签输入框。

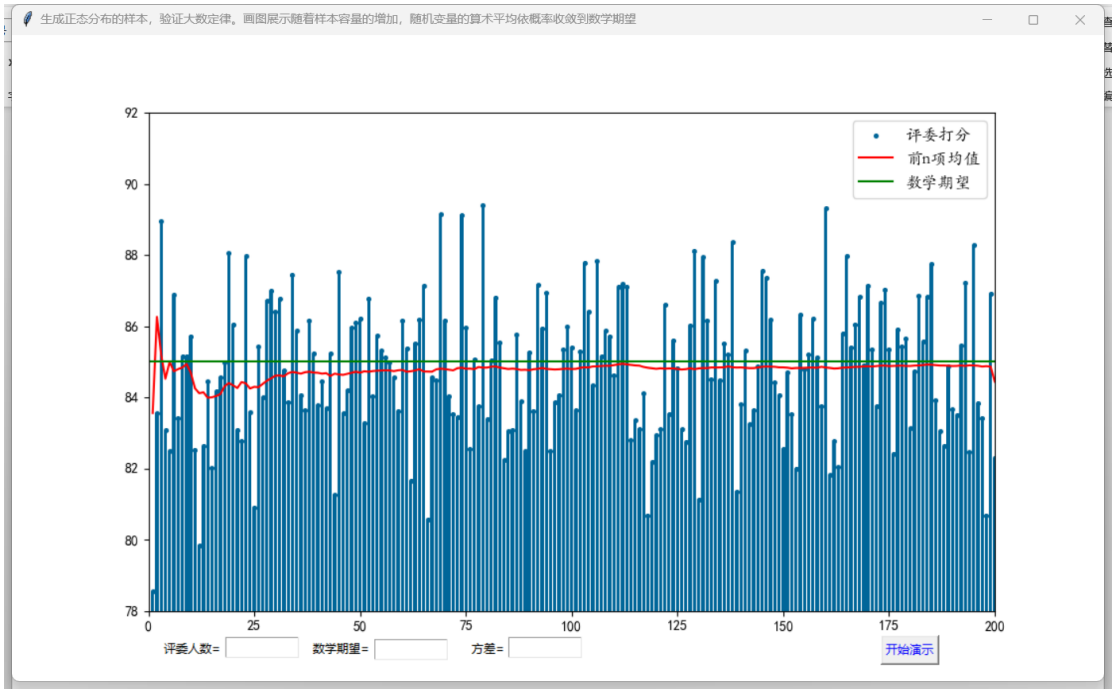
第三部分 实验结果与分析

1. 实验数据及结果




验证泊松定理并展示，对于泊松分布固定的λ，随着二项分布n的增加，二项分布逐渐收敛于泊松分布
n=400, p_n=0.0125, λ =np_n=5.0
λ =5.0
参数(请输入4~15之间的数)λ=
随机变量
开始演示


给定参数μ，σ，展示对应的正态分布概率密度图；通过动态调整参数μ或σ，展示图像的变化
μ =0.0; σ 2=1.0
概率密度
均值μ=
方差σ²=
随机变量
均值μ变化[1,50]
方差σ²变化[1,4]
开始演示



2. 实验分析及结论

1）.使用可视化设计来创建一个菜单界面，可以使用 Python 中的 Tkinter 库来实现。通过可视化设计创建的菜单界面可以方便地展示各种实验内容，提高用户体验。

2）.使用蒙特卡洛方法来计算圆周率，可以使用 Python 中的 numpy 库里的 random（）函数来生成随机数，并结合数学公式计算圆周率。蒙特卡洛方法可以用于计算圆周率，结果的精度随着生成的随机数数量的增加而提高。

3）. 验证泊松定理，即当二项分布中的 n 趋向于无穷大时，二项分布逐渐收敛于泊松分布，可以使用 Python 中的 numpy 库来生成二项分布和泊松分布，并结合数学公式进行验证。随着二项分布 n 的增加，二项分布逐渐收敛于泊松分布。

4）. 给定参数 μ，σ，并展示对应的正态分布概率密度图，可以使用 Python 中的 matplotlib 库来绘制概率密度图，并通过交互式界面动态调整参数 μ 或 σ。通过动态调整参数 μ 或 σ，可以观察到概率密度图的变化，进一步理解正态分布的特性。

5）。生成正态分布的样本，并验证大数定律，可以使用 Python 中的 numpy 库生成正态分布的样本，并结合数学公式进行验证。随着样本容量的增加，随机变量的算术平均依概率收敛到数学期望，验证了大数定律的正确性。

第四部分 心得与展望

1. 自我评价及心得体会

完成这份实验，使我不仅学习理解了概率论的相关知识，更让我学会了使用 python 来对概率论的相关模型进行实验验证并进行可视化展示。在一次次遇见问题，寻找资料解决问

题的这个过程中，我的编程能力不断提高，同时我也感受到了问题解决时带来的喜悦与兴奋。正是这份满足感，使我虽然在实验程序编写时磕磕碰碰，但最终完成了这份实验。

2. 展望

由于计算机语言可以为学生提供更直观、更生动、更互动的学习体验，同时还可以帮助学生更好地理解和掌握学科知识，所以随着科技的发展，越来越多的学科将利用计算机进行学习和运算算等其他操作。对此，学好 python 或其它计算机语言无疑是极为重要的。

我觉得这份实验使我收获最多的不是 python 或概率论知识的掌握，而是在遇到问题时我学会了利用网络查找资源解决问题提升自己。未来的生活充满了机遇和挑战，我们总是要不断学习，如此才能感受到自己的心跳。

附录（源代码）

```
import tkinter as tk
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.font_manager as fm
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

plt.rcParams["font.sans-serif"] = ["SimHei"]  # 设置字体
plt.rcParams["axes.unicode_minus"] = False  # 该语句解决图像中的“-”负号的乱码问题

def root():  # 菜单设置函数
    root = tk.Tk()  # 创建一个窗口
    root.title("概率论与数理统计实验，研制者：黄堉轩")  # 定义窗口名称
    root.geometry('1100x650+250+75')  # 窗口宽度 x 窗口高度+窗口 x 轴+窗口 y 轴

    # 定义菜单
    bar = tk.Menu(root)  # bar 为 root 下的菜单
    bar1 = tk.Menu(bar, tearoff=0)  # bar1，2，3，4 为 bar 下的菜单，tearoff = 0 菜单就不会有虚线
    bar2 = tk.Menu(bar, tearoff=0)  # tearoff 点击子菜单中的 ---------，可以将其“撕下”，默认为 True，设为 False 关闭
    bar3 = tk.Menu(bar, tearoff=0)
    bar4 = tk.Menu(bar, tearoff=0)
    # 配置菜单栏，窗口 root 加入菜单 bar
    root.config(menu=bar)
    # 菜单 bar 下加入菜单 bar1,2,3,4， 菜单名.add_cascade(label=菜单名，menu=菜单)，功能:级联子菜单和上层菜单
    bar.add_cascade(label='1', menu=bar1)
    bar.add_cascade(label='2', menu=bar2)
    bar.add_cascade(label='3', menu=bar3)
    bar.add_cascade(label='4', menu=bar4)
    # 菜单 bar1，2，3，4 下加入标签， 菜单名.add_command(label=标签名)， 功能:加入标签
    # add_command 有下面几个属性：    label：指定标签的名称 command:被点击时调用的方法 acceletor：快捷键 underline：是否拥有下划线
    # 除了默认的点击后无显示的效果，Menu 还可以设置单选框（add_radiobutton）与复选框(add_checkbutton)，只需对应地替换掉 add_command
    bar1.add_command(label="利用蒙特卡洛方法计算圆周率并展示结果", command=bar1_label, accelerator='Alt+1')
    bar2.add_command(label="验证泊松定理并展示，对于泊松分布固定的" + chr(955) + "，随着二项分布 n 的增加，二项分布逐渐收敛于泊松分布", command=bar2_lavel, accelerator='Alt+2')
```

```
    bar3.add_command(label="给定参数" + chr(956) + ", " + chr(963) + ",展示对应的正态分布概率密度图；通过动态调整参数" + chr(956) + "或" + chr(963) +",展示图像的变化", command=bar3_lavel, accelerator='Alt+3')
    bar4.add_command(label="生成正态分布的样本，验证大数定律。画图展示随着样本容量的增加，随机变量的算术平均依概率收敛到数学期望", command=bar4_lavel, accelerator='Alt+4')
    bar.add_command(label='关于', command=state)
    # add_separator()方法可以添加分割线，调用的时候很简单，需要在哪添加，就把这行代码放在那个地方

    root.mainloop()  # 显示窗口

def bar1_label():  # 实验 1 图像绘制函数
    root1 = tk.Tk()  # 创建一个新窗口
    root1.title('利用蒙特卡洛方法计算圆周率并展示结果')  # 窗口标题
    root1.geometry('1100x650+250+75')  # 窗口大小

    def bar1_label_0(n='1000'):  # 随机点默认为 1000
        try:
            n = int(eval(n))  # 将随机点数目向下取整
        except:
            n = 1000  # 默认为 1000

        figure1 = plt.figure(num='利用蒙特卡洛方法计算圆周率并展示结果', figsize=(11, 6.5))  # 设置画布，figure(),画布名，大小
        FigureCanvasTkAgg(figure1, root1).get_tk_widget().place(x=0, y=0)  # 让画布在窗口上显示
        entry = tk.Entry(root1, bg='white', fg='black', width=10)  # 在窗口添加输入框
        entry.place(x=300, y=607)  # 放置输入框位置
        tk.Button(root1, text='开始演示', command=lambda: bar1_label_0(entry.get()), height=1, fg='blue').place(x=775, y=603)  # 在窗口放置一个按钮
        tk.Label(root1, text='随机点的数目 n:', bg='white').place(x=200, y=605)  # 在窗口添加标签

        plt.clf()  # 清空原来图像
        # 绘制子图，subplot 命令:可以将 figure 对象分为多个区域，每个区域分别放置一个 Axes 对象进行绘图。 三个参数：行，列，序号
        # plt.subplot(221)  # 前面两个参数分别表示行和列，即将 figure 分为 2 行 2 列的区域，该图形的位置为第一行的左图
        # plt.subplot(222)  # 第一行的右图
        # plt.subplot(212)  # 第二整行
        # 绘制第一个子图
```

```
        plt.subplot(2, 1, 1)
        # 绘制随机点
        m = 0  # 落在圆内的随机点数目
        count = 0  # 试验次数
        pi = []  # pi 列表
        x_list = np.random.uniform(-1, 1, size=n)  # 生产随机点的横坐标
        y_list = np.random.uniform(-1, 1, size=n)  # 生产随机点的纵坐标
        for x, y in zip(x_list, y_list):  # 实验
            count += 1
            if x * x + y * y <= 1:
                color = 'b'
                m += 1  # 计算落在圆内的随机点数目
            else:
                color = '#FF9900'  # 调整随机点颜色
            plt.scatter(x, y, c=color, marker='.')
            pi.append(4 * m / count)  # 添加 pi
        # 绘制方框
        x_list1 = [1, 1, -1, -1, 1]  # 四个端点横坐标
        y_list1 = [-1, 1, 1, -1, -1]  # 四个端点纵坐标，最后回到起点，形成封闭图形
        plt.plot(x_list1, y_list1, color='#FF9900', label='生成的随机点数目: n=' + str(n))
        # 绘制圆
        x_list2_0 = list(range(-1000, 1001, 1))  # range()步长为整数，横坐标初始化
        x_list2 = []  # 横坐标列表
        y_list2 = []  # 纵坐标列表
        for i in x_list2_0:  # 上半圆坐标
            x = i / 1000  # 初始横坐标/1000，得到横坐标
            x_list2.append(x)  # 横坐标列表添加横坐标
            y = (1 - x ** 2) ** 0.5  # 计算纵坐标
            y_list2.append(y)  # 纵坐标列表添加纵坐标
        x_list2.extend([x for x in x_list2[::-1]])  # 下半圆坐标
        y_list2.extend([-y for y in y_list2[::-1]])
        plt.plot(x_list2, y_list2, color='b', label='落在圆内的随机点数目: m=' + str(m))
        # 设置数轴刻度
        plt.xticks([-1, 0, 1, 2, 3])
        plt.yticks([-1.0, -0.5, 0.0, 0.5, 1.0])
        # 创建字体，设置图例 legend()图例创建函数，prop 字体设置参数，loc 位置设置参数
        myfont = fm.FontProperties(fname=r'C:\Windows\Fonts\STKAITI.ttf', size=12)
        plt.legend(prop=myfont, loc='upper right')
```

```
        # 绘制第二个子图
        plt.subplot(2, 1, 2)
        plt.plot([0, n], [3.1415926, 3.1415926], label=chr(960) + '的理论值：3.1415926', color='y')  # pi 理论值
        plt.plot(list(range(0, n)), pi, label=chr(960) + '的模拟计算值：' + str(pi[n - 1]), color='#006699')  # pi 实验值
        # 设置数轴上下限
        plt.ylim(2.9, 3.3)
        # 设置数轴刻度
        plt.yticks([2.9, 3.0, 3.1, 3.2, 3.3])
        # 创建字体，设置图例 legend()图例创建函数，prop 字体设置参数，loc 位置设置参数
        myfont = fm.FontProperties(fname=r'C:\Windows\Fonts\STKAITI.ttf', size=12)
        plt.legend(prop=myfont, loc='upper right')

    bar1_label_0()  # 绘制图像
    root1.mainloop()  # 显示窗口

def bar2_lavel():  # 实验 2 图像绘制函数
    def jiechen(x):
        if x == 0:
            return 1
        else:
            return x * jiechen(x - 1)

    root2 = tk.Tk()  # 创建一个新窗口
    root2.title("验证泊松定理并展示，对于泊松分布固定的" + chr(955) + "，随着二项分布 n 的增加，二项分布逐渐收敛于泊松分布")  # 窗口标题
    root2.geometry('1100x650+250+75')  # 窗口大小

    def bar2_lavel_0(lamela='5.00'):
        try:
            lamela = eval(lamela)
        except:
            lamela = 5.00

        figure2 = plt.figure(num="验证泊松定理并展示，对于泊松分布固定的" + chr(955) + "，随着二项分布 n 的增加，二项分布逐渐收敛于泊松分布", figsize=(11, 6.5))
        FigureCanvasTkAgg(figure2, root2).get_tk_widget().place(x=0, y=0)  # 让画布在窗口上显示
        entry = tk.Entry(root2, bg='white', fg='black', width=10)  # 在窗口添加输入框
```

```
        entry.place(x=350, y=607)  # 放置输入框位置
        tk.Button(root2, text='开始演示', command=lambda:
bar2_lavel_0(entry.get()), height=1, fg='blue').place(x=775, y=603)  #
在窗口放置一个按钮
        tk.Label(root2, text='参数(请输入 4~15 之间的数)' + chr(955) + '=',
bg='white').place(x=175, y=605)  # 在窗口添加标签

        plt.clf()
        x_list = list(range(0, 21))
        n = 400
        e = 2.71828
        p_n = lamela / n
        y_1_list = [jiechen(n) / jiechen(n - i) / jiechen(i) * p_n ** i
* (1 - p_n) ** (n - i) for i in x_list]  # 二项分布
        y_2_list = [lamela ** i * e ** (lamela * (-1)) / jiechen(i) for
i in x_list]  # 泊松分布
        plt.subplot(2, 1, 1)
        plt.title('n=' + str(n) + ',p$_n$=' + str(p_n) + ',' + chr(955)
+ '=np$_n$=' + str(lamela))
        plt.bar(x_list, y_1_list, width=0.05, color='#006699')
        plt.scatter(x_list, y_1_list, s=28, marker='.', color='#006699')
        plt.subplot(2, 1, 2)
        plt.title(chr(955) + '=' + str(lamela))
        plt.bar(x_list, y_2_list, width=0.05, color='#006699')
        plt.scatter(x_list, y_2_list, s=28, marker='.', color='#006699')
        plt.xlabel('随机变量')

    bar2_lavel_0()
    root2.mainloop()  # 显示窗口

def bar3_lavel():  # 实验 3 图像绘制函数
    root3 = tk.Tk()  # 创建一个新窗口
    root3.title("给定参数" + chr(956) + ", " + chr(963) + ", 展示对应的正态
分布概率密度图；通过动态调整参数" + chr(956) + "或" + chr(963) + ", 展示图像
的变化")  # 窗口标题
    root3.geometry('1100x650+250+75')  # 窗口大小

    def bar3_lavel_0(miu='0.00', sigema2='1.00'):
        try:
            miu = eval(miu)
        except:
            miu = 0.00
        try:
            sigema2 = eval(sigema2)
```

```
        except:
            sigema2 = 1.00

        figure3 = plt.figure(num="给定参数" + chr(956) + ", " + chr(963) +
", 展示对应的正态分布概率密度图；通过动态调整参数" + chr(956) + "或" + chr(963)
+ ", 展示图像的变化", figsize=(11, 6.5))
        FigureCanvasTkAgg(figure3, root3).get_tk_widget().place(x=0,
y=0)  # 让画布在窗口上显示
        entry1 = tk.Entry(root3, bg='white', fg='black', width=10)  # 在
窗口添加输入框 u
        entry1.place(x=200, y=605)  # 放置输入框位置
        entry2 = tk.Entry(root3, bg='white', fg='black', width=10)  #
sigema
        entry2.place(x=350, y=607)
        tk.Button(root3, text='均值' + chr(956) + '变化[1,50]',
command=lambda: bar3_lavel_0(str(miu + 1.00), str(sigema2)),
height=1, ).place(x=625, y=603)
        tk.Button(root3, text='方差'+chr(963)+'\u00B2 变化[1,4]',
command=lambda: bar3_lavel_0(str(miu), str(sigema2+1.00)),
height=1).place(x=750, y=603)
        tk.Button(root3, text='开始演示', command=lambda:
bar3_lavel_0(entry1.get(), entry2.get()), height=1,
fg='blue').place(x=875, y=603)  # 在窗口放置一个按钮
        tk.Label(root3, text='均值' + chr(956) + '=',
bg='white').place(x=150, y=605)  # 在窗口添加标签
        tk.Label(root3, text='方差' + chr(963) + '\u00B2=',
bg='white').place(x=300, y=605)  # 在窗口添加标签

        plt.clf()
        e = 2.71828
        pi = 3.14
        x_list = [i/1000 for i in range(int(1000*miu-6000),
int(1000*miu+6001))]
        y_list = [1/(((2*pi)**0.5)*(sigema2**0.5)) *
e**(-1*(i-miu)**2/(2*sigema2)) for i in x_list]

plt.title(chr(956)+'='+str(miu)+';'+chr(963)+'$^2$='+str(sigema2))
        plt.plot(x_list, y_list, color='#006699')
        plt.xlabel('随机变量')
        plt.ylabel('概率密度')
        plt.plot([miu, miu], [0.00, 1/(((2*pi)**0.5)*(sigema2**0.5))],
color='red', linestyle='--')

    bar3_lavel_0()
```

```
    root3.mainloop()  # 显示窗口

def bar4_lavel():  # 实验 4 图像绘制函数
    root4 = tk.Tk()  # 创建一个新窗口
    root4.title("生成正态分布的样本，验证大数定律。画图展示随着样本容量的增加，随机变量的算术平均依概率收敛到数学期望")  # 窗口标题
    root4.geometry('1100x650+250+75')  # 窗口大小

    def bar4_lavel_0(n='200', miu='85', sigema2='4'):
        try:
            n = eval(n)
        except:
            n = 200
        try:
            miu = eval(miu)
        except:
            miu = 85
        try:
            sigema2 = eval(sigema2)
        except:
            sigema2 = 4

        figure4 = plt.figure(num="生成正态分布的样本，验证大数定律。画图展示随着样本容量的增加，随机变量的算术平均依概率收敛到数学期望", figsize=(11, 6.5))
        FigureCanvasTkAgg(figure4, root4).get_tk_widget().place(x=0, y=0)  # 让画布在窗口上显示
        entry1 = tk.Entry(root4, bg='white', fg='black', width=10)  # 在窗口添加输入框评委人数
        entry1.place(x=215, y=605)  # 放置输入框位置
        entry2 = tk.Entry(root4, bg='white', fg='black', width=10)  # 数学期望
        entry2.place(x=365, y=607)
        entry3 = tk.Entry(root4, bg='white', fg='black', width=10)  # 方差
        entry3.place(x=500, y=605)
        tk.Button(root4, text='开始演示', command=lambda: bar4_lavel_0(entry1.get(), entry2.get(), entry3.get()), height=1, fg='blue').place(x=875, y=603)  # 在窗口放置一个按钮
        tk.Label(root4, text='评委人数=', bg='white').place(x=150, y=605)  # 在窗口添加标签
        tk.Label(root4, text='数学期望=', bg='white').place(x=300, y=605)  # 在窗口添加标签
        tk.Label(root4, text='方差=', bg='white').place(x=460, y=605)  # 在窗口添加标签
```

```
        plt.clf()
        x_list = list(range(1, n+1))
        y_list1 = [i*sigema2**0.5+miu for i in
np.random.standard_normal(n)]
        y_list2 = [sum(y_list1[1:i+1])/i for i in range(1, n+1)]
        plt.bar(x_list, y_list1, width=0.77, color='#006699')
        plt.scatter(x_list, y_list1, s=28, marker='.', color='#006699',
label='评委打分')
        plt.plot(x_list, y_list2, color='red', label='前 n 项均值')
        plt.plot([0, n], [miu, miu], color='green', label='数学期望' )
        # 创建字体，设置图例 legend()图例创建函数，prop 字体设置参数，loc 位置设
置参数
        myfont =
fm.FontProperties(fname=r'C:\Windows\Fonts\STKAITI.ttf', size=12)
        plt.legend(prop=myfont, loc='upper right')
        plt.ylim(miu-7, miu+7)
        plt.xlim(0, n)

    bar4_lavel_0()
    root4.mainloop()  # 显示窗口

def state():  # 关于函数
    root_state = tk.Tk()
    root_state.title('关于')
    root_state.geometry('300x300+650+250')
    tk.Label(root_state, text='制作者：黄堉轩\n 版本：1.0.0').pack()
    root_state.mainloop()

if __name__ == '__main__':
    root()
```