**《Python程序设计》**

**课程项目作业报告**



**序号及名称：**项目作业2-三种数列求和的多种方法

**学号与姓名：3240103494 郑雨静**

**年级与专业：**大一 工商管理

**学 院 (系)：**求是学院

**日期：**2025年4月23日

# 一、作品名称：《库：等差数列求和》

## 数列描述：

相邻两项之差为[常数](https://baike.baidu.com/item/%E5%B8%B8%E6%95%B0/2215683?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/%E7%AD%89%E5%B7%AE%E6%95%B0%E5%88%97/_blank)的[数列](https://baike.baidu.com/item/%E6%95%B0%E5%88%97/731531?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/%E7%AD%89%E5%B7%AE%E6%95%B0%E5%88%97/_blank)。

## 代码对应的Python知识点：

方法一：自定义函数 类

方法二：定循环与不定循环

方法三：函数sum求和

## 源码及注释：

方法一：自定义函数 类

# 定义计算等差数列的前n项和的函数

class CalculateSum:

def \_\_init\_\_(self):

# 输入首项

self.a1 = float(input("请输入等差数列的首项a："))

# 输入公差

self.d = float(input("请输入等差数列的公差d："))

# 输入项数

self.n = int(input("请输入要求和的项数n："))

def sum\_dengcha(self):

# 根据等差数列的通项公式计算尾项

an = self.a1 + (self.n - 1) \* self.d

# 使用求和公式计算前n项和

sum\_value = self.n \* (self.a1 + an) / 2

return sum\_value

# 调用函数并输出结果

obj = CalculateSum()

result = sum\_dengcha()

print(f"等差数列前n项和为: {result}")

方法二：定循环与不定循环

#用定循环for实现

# 输入首项

a1 = float(input("请输入等差数列的首项a："))

# 输入公差

d = float(input("请输入等差数列的公差d："))

# 输入项数

n = int(input("请输入要求和的项数n："))

sum\_value=0

#用通项公式计算数列的第i项并累加求和

for i in range(1,n+1):

sum\_value = sum\_value + a1 + d\*(i-1)

print(f"等差数列前n项和为: {sum\_value}")

#用不定循环while实现

# 输入首项

a1 = float(input("请输入等差数列的首项a："))

# 输入公差

d = float(input("请输入等差数列的公差d："))

# 输入项数

n = int(input("请输入要求和的项数n："))

sum\_value=0

i=1

#用通项公式计算数列的第i项并用判定循环累加求和

while i<=n:

sum\_value = sum\_value + a1 + d\*(n-1)

i+=1

print(f"等差数列前n项和为: {sum\_value}")

方法二：用自定义函数实现

class CalculateSum:

def \_\_init\_\_(self,n):

self.n=n

def sum(self):

s=0

term=''

for i in range(1,self.n+1):

term+=str(i)

s+=int(term)

return(s)

n=int(input('请输入要求和的项数n:'))

seq=CalculateSum(n)

print('数列和为%s'%seq.sum())

方法三：用类来实现

N=int(input('请输入要求和的项数n:'))

def calculate\_sum(n):

#与定循环结合

sequence\_sum=0

term=''

for i in range(n):

term+=str(i)

sequence\_sum+=int(term)

return sequence\_sum

print('数列和为%s'%calculate\_sum(N))

**代码测试（GUI包装）**

import tkinter as tk

from tkinter import scrolledtext, Radiobutton

import inspect

# #用定循环for实现

def sum\_by\_loop(n):

# n=int(input('请输入要求和的项数n:'))

sequence\_sum=0

term=''

for i in range(1,n+1):

term+=str(i)

sequence\_sum+=int(term)

# print('数列和为%s'%sequence\_sum)

return sequence\_sum

#用不定循环while实现

def sum\_by\_while(n):

# n=int(input('请输入要求和的项数n:'))

sequence\_sum=0

term=''

i=1

while i<=n:

term+=str(i)

sequence\_sum+=int(term)

i+=1

# print('数列和为%s'%sequence\_sum)

return sequence\_sum

# 用自定义函数来实现

def sum\_by\_custom(n):

def calculate\_sum(n):

#与定循环结合

sequence\_sum=0

term=''

for i in range(1,n+1):

term+=str(i)

sequence\_sum+=int(term)

return sequence\_sum

# print('数列和为%s'%calculate\_sum(N))

return calculate\_sum(n)

#用类来实现

def sum\_by\_class\_method(n):

class CalculateSum:

def \_\_init\_\_(self,n):

self.n=n

def sum(self):

s=0

term=''

for i in range(1,self.n+1):

term+=str(i)

s+=int(term)

return(s)

# n=int(input('请输入要求和的项数n:'))

seq=CalculateSum(n)

# print('数列和为%s'%seq.sum())

return seq.sum()

root = tk.Tk()

root.title("数列求和库")

root.geometry('800x700') # 调整大小以确保所有元素适应

root.configure(bg='#f7f7f7')

# a\_label = tk.Label(root, text="输入a：", bg='#f7f7f7', font=('Arial', 12))

# a\_label.pack()

# a\_entry = tk.Entry(root, width=50)

# a\_entry.pack(pady=5)

n\_label = tk.Label(root, text="输入n：", bg='#f7f7f7', font=('Arial', 12))

n\_label.pack()

n\_entry = tk.Entry(root, width=50)

n\_entry.pack(pady=5)

method\_label = tk.Label(root, text="选择求和方法：", bg='#f7f7f7', font=('Arial', 12))

method\_label.pack()

methods = {

'定循环求和': sum\_by\_loop,

'不定循环求和': sum\_by\_while,

'自定义函数求和': sum\_by\_custom,

'类方法求和': sum\_by\_class\_method

}

method\_var = tk.StringVar()

method\_var.set('定循环求和')

for text in methods.keys():

Radiobutton(root, text=text, variable=method\_var, value=text, bg='#f7f7f7', font=('Arial', 10)).pack(anchor=tk.W)

result\_label = tk.Label(root, text="", font=('Helvetica', 12), bg='#f7f7f7')

result\_label.pack(pady=20)

code\_label = tk.Label(root, text="源代码：", bg='#f7f7f7', font=('Arial', 12))

code\_label.pack()

code\_text = scrolledtext.ScrolledText(root, width=90, height=15, bg='white')

code\_text.pack(pady=5)

def calculate\_and\_display():

# a = a\_entry.get()

n = n\_entry.get()

try:

n = int(n)

assert n > 0

except (ValueError, AssertionError):

result\_label.config(text="n需要是正整数")

return

# if not a.isdigit():

# result\_label.config(text="a需要是数字")

# return

method = method\_var.get()

result = methods[method](n)

result\_label.config(text=f"求和结果: {result}")

code\_text.delete('1.0', tk.END)

code\_text.insert(tk.INSERT, inspect.getsource(methods[method]))

calculate\_button = tk.Button(root, text="计算", command=calculate\_and\_display, bg='#b3b3b3', font=('Arial', 12))

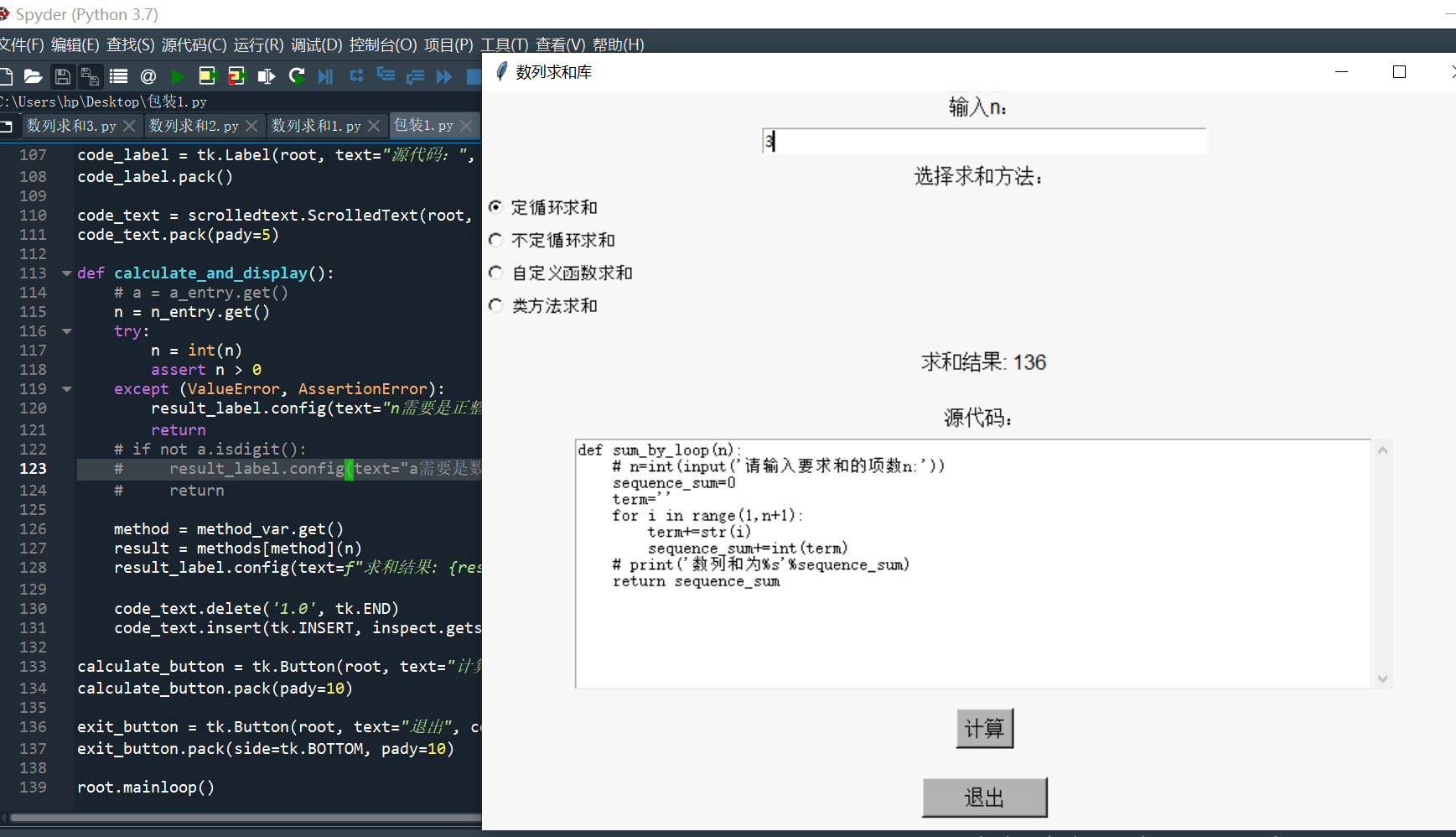
calculate\_button.pack(pady=10)

exit\_button = tk.Button(root, text="退出", command=root.quit, bg='#b3b3b3', font=('Arial', 12), width=10)

exit\_button.pack(side=tk.BOTTOM, pady=10)

root.mainloop()

1. Spider



1. Thonny
2. IDLE

# 二、作品名称：《数列求和库2.0》，难度\*\*\*\*

## 数列描述：任意等比数列

## 代码对应的Python知识点：

方法一：定循环与不定循环

方法二：函数 自定义

方法三：类

**源码注释**

方法一：定循环与不定循环

a1=int(input('请输入等比数列首项：'))

q=float(input('请输入等比数列公比：'))

n=int(input('请输入等比数列项数：'))

sequence\_sum=a1

for i in range(n-1):

a1\*=q

sequence\_sum+=a1

print("该等比数列前%s项和为%.2f"%(n,sequence\_sum))

a1=int(input('请输入等比数列首项：'))

q=float(input('请输入等比数列公比：'))

n=int(input('请输入等比数列项数：'))

sequence\_sum=a1

i=1

while i<n:

a1\*=q

sequence\_sum+=a1

i+=1

print("该等比数列前%s项和为%.2f"%(n,sequence\_sum))

方法二：自定义函数

# 自定义函数结合定循环实现

def sum\_of\_GeometricSeries(a1,q,n):

sequence\_sum=a1

for i in range(n-1):

a1\*=q

sequence\_sum+=a1

return sequence\_sum

terms=input('请输入等比数列首项、公比和项数，用空格隔开：')

lst=list(map(float,terms.split()))

lst[2]=int(lst[2])

print("该等比数列前%s项和为%.2f"%(lst[2],sum\_of\_GeometricSeries(lst[0],lst[1],lst[2])))

方法三：类

#用类来实现

class GeometricSeries:

def \_\_init\_\_(self,first\_term,common\_ratio,num\_of\_term):

self.a1=first\_term

self.q=common\_ratio

self.n=num\_of\_term

def calcu\_sum(self):

sequence\_sum=self.a1

for i in range(self.n-1):

self.a1\*=self.q

sequence\_sum+=self.a1

return sequence\_sum

terms=input('请输入等比数列首项、公比和项数，用空格隔开：')

lst=list(map(float,terms.split()))

series=GeometricSeries(lst[0],lst[1],int(lst[2]))

print("该等比数列前%s项和为%.2f"%(int(lst[2]),series.calcu\_sum()))

**代码测试（用GUI包装）**

import tkinter as tk

from tkinter import scrolledtext, Radiobutton

import inspect

# #用定循环for实现

def sum\_by\_loop(a1,q,n):

# n=int(input('请输入要求和的项数n:'))

sequence\_sum=a1

for i in range(n-1):

a1\*=q

sequence\_sum+=a1

# print('数列和为%s'%sequence\_sum)

return sequence\_sum

#用不定循环while实现

def sum\_by\_while(a1,q,n):

# n=int(input('请输入要求和的项数n:'))

sequence\_sum=a1

i=1

while i<n:

a1\*=q

sequence\_sum+=a1

i+=1

return sequence\_sum

# 用自定义函数来实现

def sum\_by\_custom(a1,q,n):

def sum\_of\_GeometricSeries(a1,q,n):

sequence\_sum=a1

for i in range(n-1):

a1\*=q

sequence\_sum+=a1

return sequence\_sum

# print('数列和为%s'%calculate\_sum(N))

return sum\_of\_GeometricSeries(a1,q,n)

#用类来实现

def sum\_by\_class\_method(a1,q,n):

class GeometricSeries:

def \_\_init\_\_(self,first\_term,common\_ratio,num\_of\_term):

self.a1=first\_term

self.q=common\_ratio

self.n=num\_of\_term

def calcu\_sum(self):

sequence\_sum=self.a1

for i in range(self.n-1):

self.a1\*=self.q

sequence\_sum+=self.a1

return sequence\_sum

# n=int(input('请输入要求和的项数n:'))

# print('数列和为%s'%seq.sum())

series=GeometricSeries(a1,q,n)

return series.calcu\_sum()

def calcu\_sum(self):

sequence\_sum=self.a1

for i in range(self.n-1):

self.a1\*=self.q

sequence\_sum+=self.a1

return sequence\_sum

# n=int(input('请输入要求和的项数n:'))

# print('数列和为%s'%seq.sum())

series=GeometricSeries(a1,q,n)

return series.calcu\_sum()

root = tk.Tk()

root.title("数列求和库")

root.geometry('800x700') # 调整大小以确保所有元素适应

root.configure(bg='#f7f7f7')

a\_label = tk.Label(root, text="输入a1：", bg='#f7f7f7', font=('Arial', 12))

a\_label.pack()

a\_entry = tk.Entry(root, width=50)

a\_entry.pack(pady=5)

q\_label = tk.Label(root, text="输入q：", bg='#f7f7f7', font=('Arial', 12))

q\_label.pack()

q\_entry = tk.Entry(root, width=50)

q\_entry.pack(pady=5)

n\_label = tk.Label(root, text="输入n：", bg='#f7f7f7', font=('Arial', 12))

n\_label.pack()

n\_entry = tk.Entry(root, width=50)

n\_entry.pack(pady=5)

method\_label = tk.Label(root, text="选择求和方法：", bg='#f7f7f7', font=('Arial', 12))

method\_label.pack()

methods = {

'定循环求和': sum\_by\_loop,

'不定循环求和': sum\_by\_while,

'自定义函数求和': sum\_by\_custom,

'类方法求和': sum\_by\_class\_method

}

method\_var = tk.StringVar()

method\_var.set('定循环求和')

for text in methods.keys():

Radiobutton(root, text=text, variable=method\_var, value=text, bg='#f7f7f7', font=('Arial', 10)).pack(anchor=tk.W)

result\_label = tk.Label(root, text="", font=('Helvetica', 12), bg='#f7f7f7')

result\_label.pack(pady=20)

code\_label = tk.Label(root, text="源代码：", bg='#f7f7f7', font=('Arial', 12))

code\_label.pack()

code\_text = scrolledtext.ScrolledText(root, width=90, height=15, bg='white')

code\_text.pack(pady=5)

def calculate\_and\_display():

a1 = a\_entry.get()

q=q\_entry.get()

n = n\_entry.get()

try:

n = int(n)

a1=float(a1)

q=float(q)

assert n > 0

except (ValueError, AssertionError):

result\_label.config(text="n需要是正整数")

return

# if not a.isdigit():

# result\_label.config(text="a需要是数字")

# return

method = method\_var.get()

result = methods[method](a1,q,n)

result\_label.config(text=f"求和结果: {result}")

code\_text.delete('1.0', tk.END)

code\_text.insert(tk.INSERT, inspect.getsource(methods[method]))

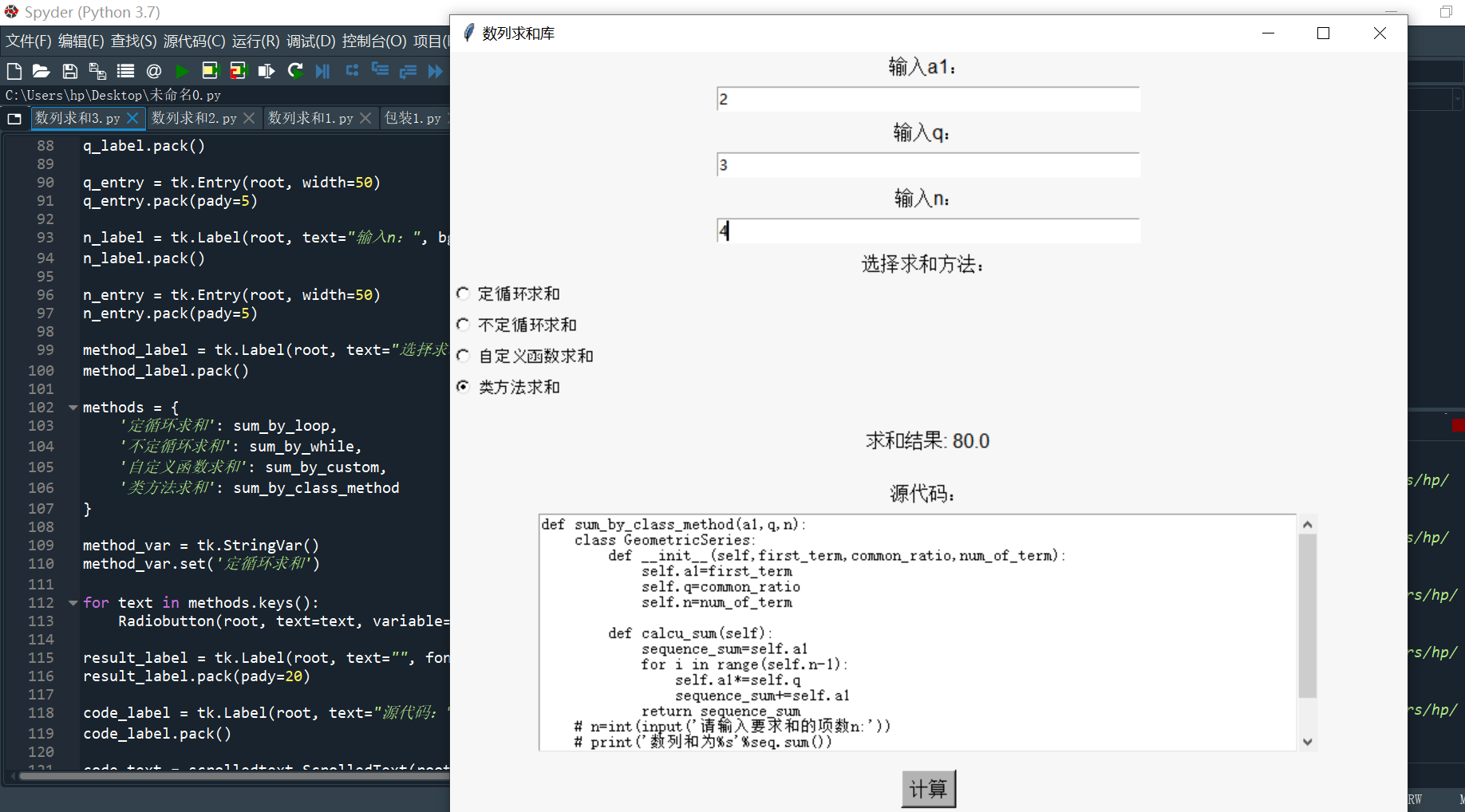
calculate\_button = tk.Button(root, text="计算", command=calculate\_and\_display, bg='#b3b3b3', font=('Arial', 12))

calculate\_button.pack(pady=10)

exit\_button = tk.Button(root, text="退出", command=root.quit, bg='#b3b3b3', font=('Arial', 12), width=10)

exit\_button.pack(side=tk.BOTTOM, pady=10)

root.mainloop()

1.spyder

2.Thonny

3.IDLE

# 三、作品名称：《数列求和库3.0》，难度\*\*\*\*\*

## 数列描述：斐波那契数列

## 代码对应的Python知识点：

方法一：定循环与不定循环

方法二：函数 自定义，递归

方法三：类

**源码注释**

方法一：定循环与不定循环

n=int(input("请输入项数："))

a=1;b=1;seq\_sum=1

for i in range(1,n):

seq\_sum+=b

b+=a

a=b-a

print("斐波那契数列前%d项和为%s"%(n,seq\_sum))

# 不定循环实现

n=int(input("请输入项数："))

a=1;b=1;seq\_sum=1;i=1

while i<n:

seq\_sum+=b

b+=a

a=b-a

i+=1

print("斐波那契数列前%d项和为%s"%(n,seq\_sum))

方法二.自定义函数与递归

# 自定义函数结合递归算法实现

def FibonacciNumber(n):

if n==0 or n==1:

return 1

else:

return FibonacciNumber(n-1)+FibonacciNumber(n-2)

n=int(input("请输入项数："))

seq\_sum=0

for i in range(n):

seq\_sum+=FibonacciNumber(i)

print("斐波那契数列前%d项和为%s"%(n,seq\_sum))

方法三.类

#用类实现

class FibonacciNumber:

def \_\_init\_\_(self,n):

self.n=n

def sequence\_creation(self):

sequence=[1,1]

if self.n==1:

return [1]

for i in range(2,self.n):

sequence.append(sequence[i-1]+sequence[i-2])

return sequence

n=int(input("请输入项数："))

seq=FibonacciNumber(n)

print("斐波那契数列前%d项和为%s"%(n,sum(seq.sequence\_creation())))

**代码测试（用GUI包装）**

import tkinter as tk

from tkinter import scrolledtext, Radiobutton

import inspect

# #用定循环for实现

def sum\_by\_loop(n):

a=1;b=1;seq\_sum=1

for i in range(1,n):

seq\_sum+=b

b+=a

a=b-a

return seq\_sum

#用不定循环while实现

def sum\_by\_while(n):

a=1;b=1;seq\_sum=1;i=1

while i<n:

seq\_sum+=b

b+=a

a=b-a

i+=1

return seq\_sum

# 用自定义函数来实现

def sum\_by\_custom(n):

def FibonacciNumber(n):

if n==0 or n==1:

return 1

else:

return FibonacciNumber(n-1)+FibonacciNumber(n-2)

seq\_sum=0

for i in range(n):

seq\_sum+=FibonacciNumber(i)

return seq\_sum

#用类来实现

def sum\_by\_class\_method(n):

class FibonacciNumber:

def \_\_init\_\_(self,n):

self.n=n

def sequence\_creation(self):

sequence=[1,1]

if self.n==1:

return [1]

for i in range(2,self.n):

sequence.append(sequence[i-1]+sequence[i-2])

return sequence

seq=FibonacciNumber(n)

return sum(seq.sequence\_creation())

def sequence\_creation(self):

sequence=[1,1]

if self.n==1:

return [1]

for i in range(2,self.n):

sequence.append(sequence[i-1]+sequence[i-2])

return sequence

seq=FibonacciNumber(n)

return sum(seq.sequence\_creation())

root = tk.Tk()

root.title("数列求和库")

root.geometry('800x700') # 调整大小以确保所有元素适应

root.configure(bg='#f7f7f7')

n\_label = tk.Label(root, text="输入n：", bg='#f7f7f7', font=('Arial', 12))

n\_label.pack()

n\_entry = tk.Entry(root, width=50)

n\_entry.pack(pady=5)

method\_label = tk.Label(root, text="选择求和方法：", bg='#f7f7f7', font=('Arial', 12))

method\_label.pack()

methods = {

'定循环求和': sum\_by\_loop,

'不定循环求和': sum\_by\_while,

'自定义函数求和': sum\_by\_custom,

'类方法求和': sum\_by\_class\_method

}

method\_var = tk.StringVar()

method\_var.set('定循环求和')

for text in methods.keys():

Radiobutton(root, text=text, variable=method\_var, value=text, bg='#f7f7f7', font=('Arial', 10)).pack(anchor=tk.W)

result\_label = tk.Label(root, text="", font=('Helvetica', 12), bg='#f7f7f7')

result\_label.pack(pady=20)

code\_label = tk.Label(root, text="源代码：", bg='#f7f7f7', font=('Arial', 12))

code\_label.pack()

code\_text = scrolledtext.ScrolledText(root, width=90, height=15, bg='white')

code\_text.pack(pady=5)

def calculate\_and\_display():

# a = a\_entry.get()

n = n\_entry.get()

try:

n = int(n)

assert n > 0

except (ValueError, AssertionError):

result\_label.config(text="n需要是正整数")

return

# if not a.isdigit():

# result\_label.config(text="a需要是数字")

# return

method = method\_var.get()

result = methods[method](n)

result\_label.config(text=f"求和结果: {result}")

code\_text.delete('1.0', tk.END)

code\_text.insert(tk.INSERT, inspect.getsource(methods[method]))

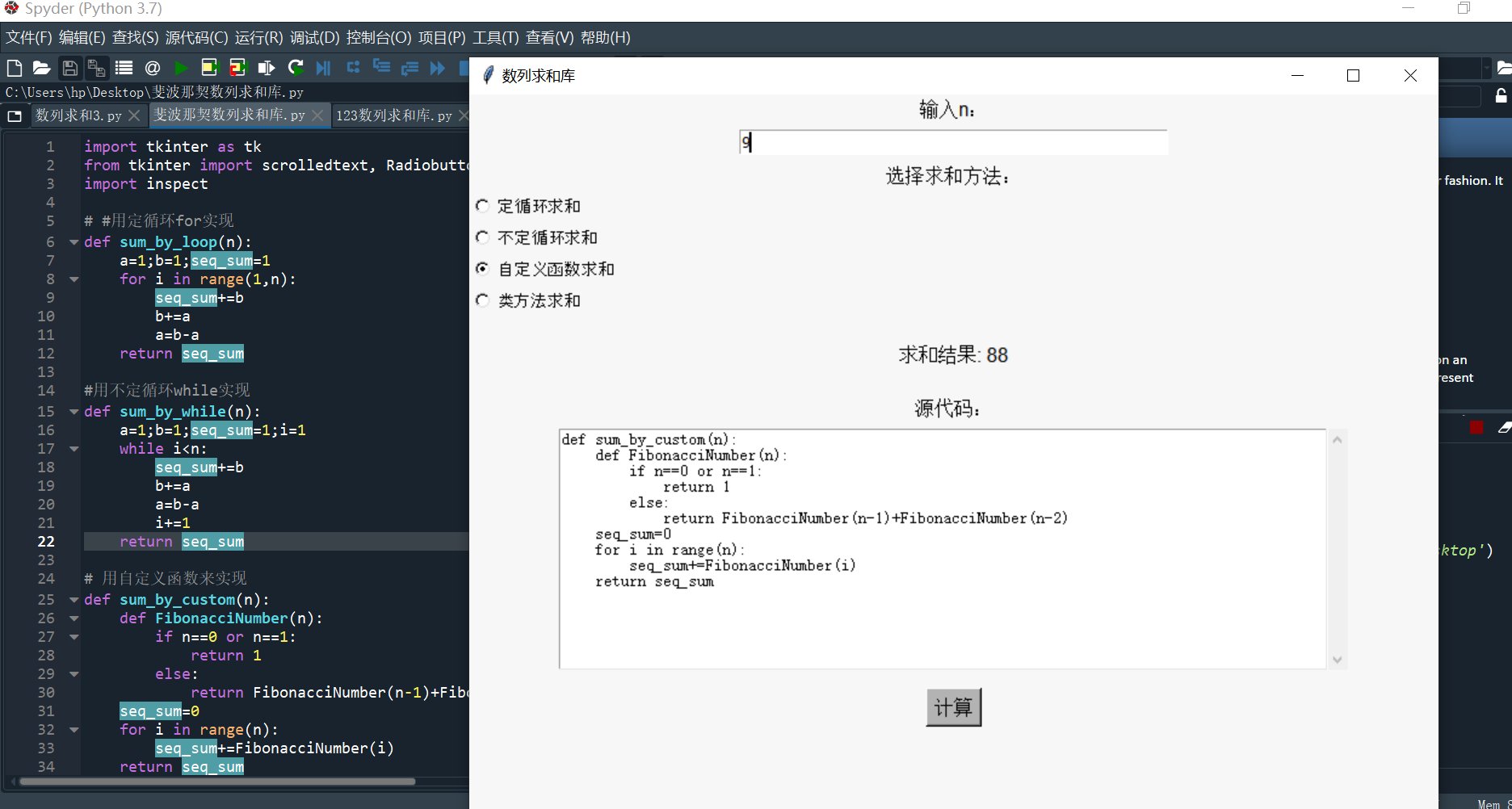
calculate\_button = tk.Button(root, text="计算", command=calculate\_and\_display, bg='#b3b3b3', font=('Arial', 12))

calculate\_button.pack(pady=10)

exit\_button = tk.Button(root, text="退出", command=root.quit, bg='#b3b3b3', font=('Arial', 12), width=10)

exit\_button.pack(side=tk.BOTTOM, pady=10)

root.mainloop()

1.spyder

2.Thonny



3.IDLE