```
#Exercise 4.2
#15.10.21
#author: chuanlu

import numpy as np
import scipy as sp
import math

e = math.e

def _composite_simpson_formula(a, b, n, func):
    x_series1 = [a + (b - a)/n * i for i in range(n + 1)]
    x_series2 = [x + (b - a)/(2 * n) for x in x_series1[:-1]]
    f_series1 = [func(x) for x in x_series1]
    f_series2 = [func(x) for x in x_series2]
    return (b - a)/n *(sum(f_series1) * 2 + sum(f_series2) * 4 - f_series1[0] - f_series1[-1]) / 6


def simpson_double_integrate(x_range, func, func1, func2, n):
    x_start = x_range[0]
    x_end = x_range[1]
    k = lambda x: func2(x) - func1(x)

    def funcAUX(i, n, func1, func2):
        def listfunc(x):
            return func1(x) + k(x) * i / n
        return listfunc
    funclist1 = [funcAUX(i,n,func1,func2) for i in range(n+1)]
    def funcAUX2(n, func):
        def listfunc2(x):
            return func(x) + k(x) / (2 * n)
        return listfunc2
    funclist2 = [funcAUX2(n, f) for f in funclist1[:-1]]
    f_values1 = _composite_simpson_formula(x_start, x_end, n, lambda x: func(x, funclist1[0](x)) * k(x))
    f_values2 = [_composite_simpson_formula(x_start, x_end, n, lambda x: func(x, funclist2[i](x)) * k(x)) for i in range(n)]
    f_values3 = [_composite_simpson_formula(x_start, x_end, n, lambda x: func(x, funclist1[i](x)) * k(x)) for i in range(1, n)]
    f_values4 = _composite_simpson_formula(x_start, x_end, n, lambda x: func(x, funclist1[-1](x)) * k(x))
    return (f_values1 + 4 * sum(f_values2) + 2 * sum(f_values3) + f_values4) / (6 * n)

def q1(func):
```

```
    x_range = [0, 1]
    func1 = lambda x: 0
    func2 = lambda x: 1
    n = 4
    result = simpson_double_integrate(x_range, func, func1, func2, n)
    print(result)

def q2(func):
    x_range = [0, 1]
    func1 = lambda x: 0
    func2 = lambda x: math.sqrt(1 - x ** 2)
    n = 4
    result = simpson_double_integrate(x_range, func, func1, func2, n)
    print(result)

def db_gauss_legendre_formula(func, n = 4):
    x_points = [-0.9061798, -0.5384693, 0, 0.5384693, 0.9061798]
    A_list = [0.2369269, 0.4786287, 0.5688889, 0.4786287, 0.2369269]
    print(sum([A_list[i] * A_list[j] * func(x_points[i], x_points[j]) for i in range(len(x_points)) for
j in range(len(x_points))])/4)



def main():
    func0 = lambda x, y: e ** (- x * y)
    q1(func0)
    func1 = lambda x, y: e ** (-(x + 1) * (y + 1) / 4)
    db_gauss_legendre_formula(func1)
    q2(func0)


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

运行结果如下：
第一小题：
0.7965999679462029
0.7965996777384788
分别为复合辛普森公式和高斯求积公式的结果
第二小题：
0.6701136333590952
采用复合辛普森公式的结果