```
#Exercise 4.1
#15.10.14
#author: chuanlu
import numpy as np
import scipy as sp
import math
from scipy import integrate
divmax = 100000

def composite_trapezoidal_formula(a, b, n, func):
    x_series = [a + (b - a)/n * i for i in range(n)]
    f_series = [func(x) for x in x_series]
    return (b - a)/n * (sum(f_series) * 2 - f_series[0] - f_series[-1]) / 2

def composite_simpson_formula(a, b, n, func):
    x_series1 = [a + (b - a)/n * i for i in range(n)]
    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 _difftrap(func, interval, numtrap):
    ''' 求二分 numtrap 次后的梯形值，其中 Interval 为积分区间，func 为被积函数'''
    if numtrap == 1:
        return (func(interval[1]) + func(interval[0])) * 0.5
    else:
        num_to_sum = int(numtrap/2)
        h = float(interval[1] - interval[0])/num_to_sum
        start = interval[0] + 0.5 * h
        points = [start + h * i for i in range(num_to_sum)]
        s = sum([func(x) for x in points], 0)
        return s

def _romberg_diff(b, c, k):
    temp = 4.0 ** k
    return (temp * c - b)/(temp - 1.0)

def romberg_formula(a, b, delta, func):
    n = 1
    interval = [a, b]
    int_range = b - a
    ord_sum = _difftrap(func, interval, n)
    result = int_range * ord_sum
    result_mat = [[result]]
```

```
    err = float('-inf')
    for i in range(1, divmax + 1):
        n *= 2
        ord_sum += _difftrap(func, interval, n)
        result_mat.append([])
        result_mat[i].append(int_range * ord_sum / n)
        for k in range(i):
            result_mat[i].append(_romberg_diff(result_mat[i-1][k], result_mat[i][k], k+1))
        result = result_mat[i][i]
        lastresult = result_mat[i-1][i-1]
        err = abs(result - lastresult)
        if err < delta:
            break
    return result

def self_adapted_simpson_formula(func, a, b, delta):
    count = 1
    return simpson_iter(func, a, b, delta, simpson(func, a, b), count)

def simpson_iter(func, a, b, delta, prev_sum, count):
    mid = (b + a) / 2
    left_sum = simpson(func, a, mid)
    right_sum = simpson(func, mid, b)
    if abs(prev_sum - left_sum - right_sum) < 15 * delta * (0.5 ** count):
        return left_sum + right_sum
    else:
        count += 1
        return simpson_iter(func, a, mid, delta, left_sum, count) + simpson_iter(func, mid, b,
delta, right_sum, count)

def simpson(func, low, high):
    mid = (low + high) / 2
    return (func(low) + 4*func(mid) + func(high))*(high - low)/6

def main():
    a = 1e-300
    b = 1
    function = lambda x: math.sqrt(x) * math.log(x)
    exact_value = list(integrate.quad(function, 0, 1))
    print("exact_value:", exact_value[0])
    for n in range(1, 10):
        print(n)
        trap_delta = abs(composite_trapezoidal_formula(a, b, n, function) - exact_value[0])
        simp_delta = abs(composite_simpson_formula(a, b, n, function) - exact_value[0])
```

```
            romberg_delta = abs(romberg_formula(a, b, 1/(2 ** n), function) - exact_value[0])
            print(trap_delta)
            print(simp_delta)
            print(romberg_delta)
        # for i in range(1000, 10000000):
            self_adapted_simp_value = self_adapted_simpson_formula(function, a, b, 1e-8)
            # if abs(self_adapted_simp_value - exact_value[0]) < 1e-4:
            #     print("i:", i)
            #     break
        print(self_adapted_simp_value)

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

这是修改后的程序，解决了第三小题原有的 bug。
程序运行结果如下：

```
exact_value: -0.44444444444444425
1
0.44444444444444425
0.44444444444444425
0.11769172995492849
2
0.32191217651087584
0.17255129494660637
0.04405846562954807
3
0.1778395985026217
0.08410687091515556
0.04405846562954807
4
0.1174828835013148
0.05089110737834268
0.017343785594182592
5
0.08549398154308585
0.0346437376950125
0.017343785594182592
6
0.06608197586077313
0.02539073909398748
0.006840607651880215
7
0.05322156643344139
0.01957289272640006
```

0.002677605290685392
8
0.04416110369961457
0.01565134028107007
0.001039147942263674
9
0.03748061585725054
0.012868045016491525
0.001039147942263674
-0.44444444274139394
[Finished in 0.3s]