Іванов Кирил

ФІТ 2-8

Варіант 8

|  |  |  |
| --- | --- | --- |
| 1. 1) | 2) | 3) |

from scipy import integrate

import numpy as np

eps = 0.001

a = 0.4

b = 1.2

def f2(x):

return 1 / np.sqrt(x + 3)

def left\_rec(f, a, b, n):

h = (b - a) / n

\_sum = 0

for i in range(0, n):

\_sum += f(a + i \* h)

return \_sum \* h

if abs(left\_rec(f2, a, b, 2 \* 10) - left\_rec(f2, a, b, 10)) / 3. <= eps:

print("Лівий прямокутник:", round(left\_rec(f2, a, b, 10), 5))

def right\_rec(f, a, b, n):

h = (b - a) / n

\_sum = 0

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

\_sum += f(a + i \* h)

return \_sum \* h

print("Правильний прямокутник:", round(right\_rec(f2, a, b, 10), 5))

def aver\_rec(f, a, b, n):

h = (b - a) / n

\_sum = 0

for i in range(0, n):

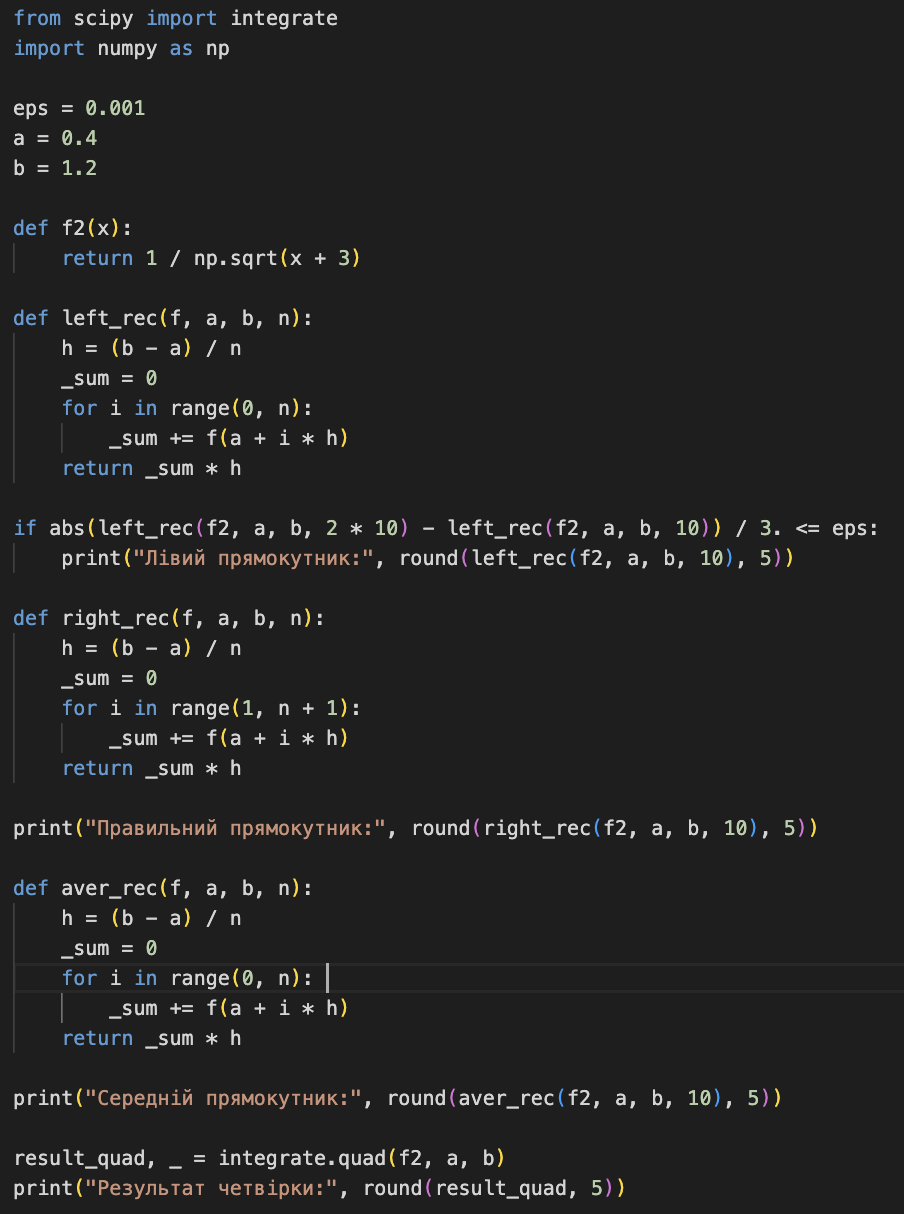
\_sum += f(a + i \* h)

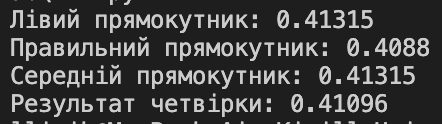
return \_sum \* h

print("Середній прямокутник:", round(aver\_rec(f2, a, b, 10), 5))

result\_quad, \_ = integrate.quad(f2, a, b)

print("Результат четвірки:", round(result\_quad, 5))





import numpy as np

from scipy import integrate

def f(x):

return np.cos(x) / (x + 2)

a = 0.4

b = 1.2

n = 8

def adaptive\_simpson\_rule(f, a, b, tol=1e-6):

h = (b - a) / n

integral1 = h / 3 \* (f(a) + 4 \* f((a + b) / 2) + f(b))

return recursive\_adaptive\_simpson(f, a, b, tol, integral1, f(a), f((a + b) / 2), f(b))

def recursive\_adaptive\_simpson(f, a, b, tol, integral1, fa, fc, fb):

c = (a + b) / 2

h = (b - a) / 2

d = (a + c) / 2

e = (c + b) / 2

fd = f(d)

fe = f(e)

integral2 = h / 3 \* (fa + 4 \* fd + fc)

integral3 = h / 3 \* (fc + 4 \* fe + fb)

if np.abs(integral2 + integral3 - integral1) <= 15 \* tol:

return integral2 + integral3

else:

tol /= 2

return (

recursive\_adaptive\_simpson(f, a, c, tol, integral2, fa, f((a + c) / 2), fc) +

recursive\_adaptive\_simpson(f, c, b, tol, integral3, fc, f((c + b) / 2), fb)

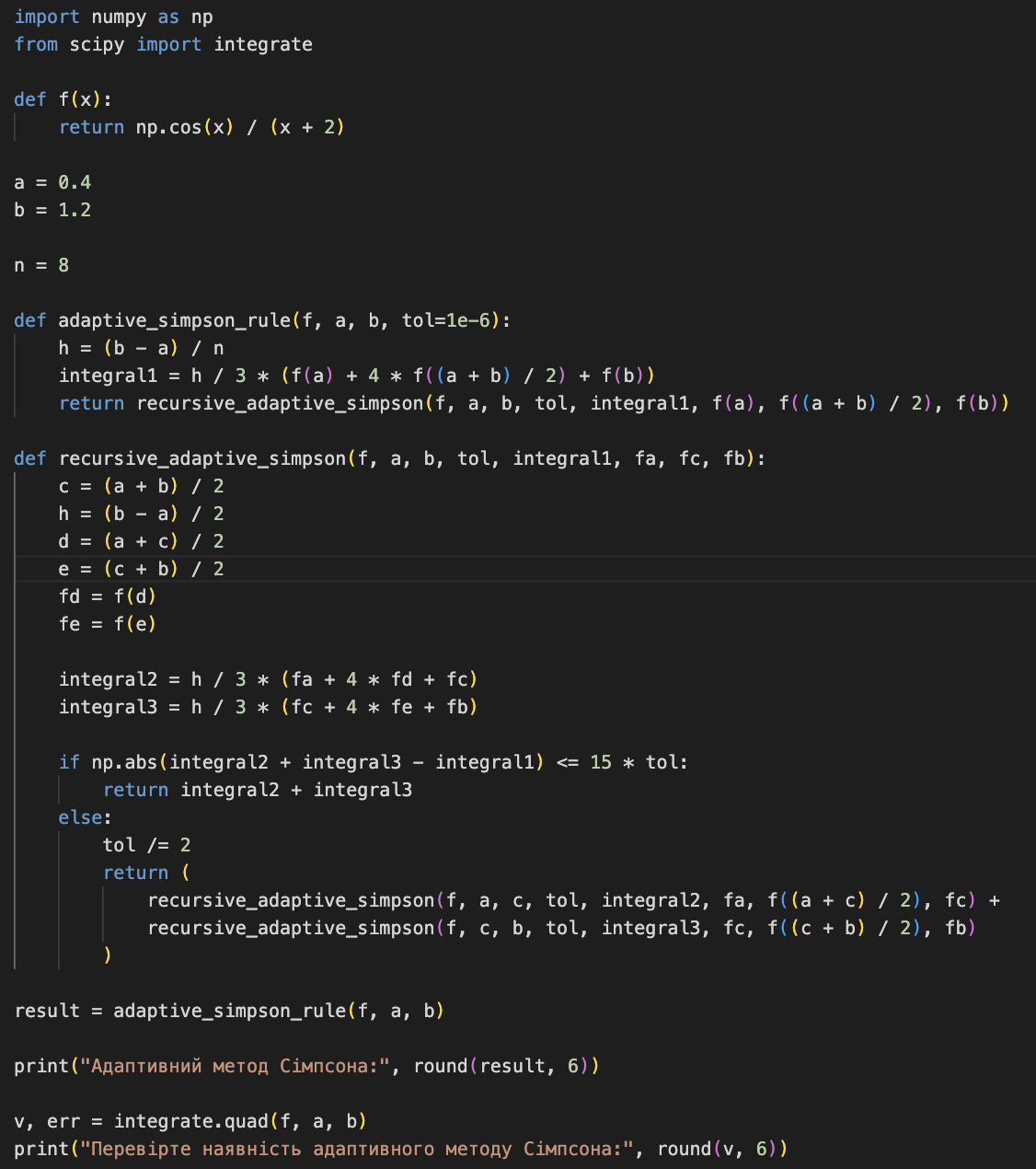
)

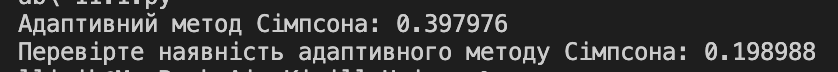
result = adaptive\_simpson\_rule(f, a, b)

print("Адаптивний метод Сімпсона:", round(result, 6))

v, err = integrate.quad(f, a, b)

print("Перевірте наявність адаптивного методу Сімпсона:", round(v, 6))





from scipy import integrate

import numpy as np

def f(x):

return 1 / np.sqrt(x\*\*2 - 2)

a = 2.1

b = 3.6

n = 10

def trapezoidal\_rule(f, a, b, n):

h = (b - a) / n

x = a + h

integral = 0

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

integral += 2 \* f(x)

x += h

integral += f(b)

integral \*= h / 2

return integral

integral1 = trapezoidal\_rule(f, a, b, n)

n \*= 2

integral2 = trapezoidal\_rule(f, a, b, n)

while abs(integral2 - integral1) / 3 > 0.001:

integral1 = integral2

n \*= 2

integral2 = trapezoidal\_rule(f, a, b, n)

print("Tрезультат рапезоїдного методу:", round(integral2, 5))

v, err = integrate.quad(f, a, b)

print("Перевірте метод трапеції:", round(v, 5))

