Задание 1. Написать программу, которая вычисляет интеграл с заданной точностью с помощью указанной составной квадратурной формулы с автоматическим выбором шага по правилу Рунге. Вывести приближенное значения интеграла, а также величину шага. Сравнить приближенное значение интеграла с точным, вычисленным аналитически.

Вариант 1. $\int_0^1 \ln(x^2+1)dx, \quad \varepsilon = 10^{-6}$, квадратурная формула трапеций.

Вот листинг программы:

```
import math

def function(x):
    return math.log(math.pow(x,2)+1)

def indefinite_integral(x):
    return x*math.log(math.pow(x,2)+1)-2*x+2*math.atan(x)

def integral(a,b):
    return indefinite_integral(b)-indefinite_integral(a)


def trapecia(a,b,n):
    h = (b - a) / n
    result = 0.5 * (function(a) + function(b))
    for i in range(1, n):
        result += function(a + i * h)
    result *= h
    return result

def process(a, b, epsilon):
    n = 1
    prev = trapecia(a, b, n)
    while True:
        n *= 2
        result = trapecia(a, b, n)
        if abs(result - prev) < epsilon:
            return result, (b - a) / n
        prev = result


a = 0
b = 1
epsilon = math.pow(10,-6)


print('Реальное значение',integral(a,b))

calculated, step = process(a,b,epsilon)

print('шаг ', step)
print('вычисленное ', calculated)
```

Вот результаты программы:

```
Реальное значение 0.2639435073548419б
шаг  0.001953125
вычисленное  0.263943825246301
```

Вывод: В задании 1 точное значение интеграла совпадает с приближенным.

Задание 2. Написать программу, которая находит приближенное значение интеграла с помощью квадратурной формулы НАСТ, а также с помощью формулы средних прямоугольников при том же самом количестве узлов. Вывести полученные приближенные значения для разного количества узлов (по своему усмотрению). На основании этих значений сделать вывод об эффективности формулы НАСТ по сравнению с формулой средних прямоугольников.

Вариант 1. $\int_{-1}^{1} \frac{e^x}{\sqrt{1-x^2}} dx.$

Листинг программы:

```
import math

import numpy as np

def f(x):
    return np.exp(x) / np.sqrt(1 - x**2)

def NAST(func, a, b, n):
    sum_result = 0
    for i in range(n + 1):
        sum_result += func(np.cos(np.pi * (2 * i + 1) / (2 * (n + 1))))
    return sum_result * np.pi / (n + 1)

def RECTANGLE(func, a, b, n):
    h = (b - a) / n
    sum_result = 0
    for i in range(n):
        sum_result += func(a + i * h + h / 2)
    return h * sum_result


a = -1
b = 1
num_nodes = [20,40,200,1000,10000,100000]

print("real = ", 3.97746)
print("Number of Nodes\tNAST\tRectangle Formula")
for num in num_nodes:
    print("number = ", num)
    for n in range(1, num + 1):
        nastRes = NAST(f, a, b, n)
        rectRes = RECTANGLE(f, a, b, n)
```

```
        print(f"{n}\t\t{nastRes}\t{rectRes}")
    print()
```

```
real =  3.97746
Number of Nodes NAST    Rectangle Formula
number =  20
1       5.600661962462454   2.0
2       6.907446368968035   2.6041406182284623
3       7.810917346480875   2.867987185409705
4       8.506302334875901   3.02261601066529
5       9.07251168985846    3.1268839237731587
6       9.55033880112241    3.203195886804148
7       9.963781640188131   3.2621300074812973
8       10.32819332802908   3.309403926434257
9       10.654003460693264  3.348410454844241
10      10.948625614676688  3.3813055299827237
11      11.217520490352394  3.409532889883235
12      11.464827659241182  3.4341003031173836
13      11.693760563205455  3.4557353053908075
14      11.906863959324086  3.4749779703769352
15      12.106187619292427  3.4922387302623537
16      12.293407010956718  3.5078358045873363
17      12.469909281498255  3.5220202219289436
18      12.636855872690854  3.5349930174373516
19      12.795229001760193  3.5469173413847135
20      12.94586675547308   3.557927167170625
```

Вывод: в нашем случае, НАСТ намного менее точная формула.