КОД ПРОГРАММЫ

```
# РАЗДЕЛ ПОДКЛЮЧЕНИЯ МОДУЛЕЙ
import math as t  # доступ к библиотеке по короткому имени 't'

# РАЗДЕЛ КОНСТАНТ
MAX_COUNT = 2500    # ограничение числа итераций (защита от зацикливания)

# РАЗДЕЛ ПЕРЕМЕННЫХ
x = 0.5     # аргумент функции (вспомогательная переменная, пока равна 0.5)

Func = 0.0  # результат функции по проверочной формуле
Eps = 0.0   # задаваемая точность суммирования ряда

X = [-0.98, -0.5, 0.1, 0.5, 0.95]  # массив аргументов для тестов

i = 0               # номер слагаемого в сумме – шаг итерации
s_i = 0.0           # слагаемое частичной суммы Sums на i-м шаге
Sums = 0.0          # накапливаемое значение суммы ряда, т.е. частичная сумма
# ---------- вспомогательные переменные для таблицы вывода

j = 0    # рабочая переменная для вывода заголовка и индекс в массиве X
z = 0     # «число знаков после запятой в Eps» +1
zs = ''   # символьное представление целого числа из z
format_s= ''   # строка таблицы для форматного вывода

# РАЗДЕЛ ОПЕРАТОРОВ
 print('Вычисление функции разложением ее в ряд.\n\n')

Eps = float(input('Введите точность EPS: ')) # ввод точности с клавиатуры

print('Eps = {0:e}'.format(Eps))    # вывод Eps на экран в экспоненциальной форме


# вычислим сразу же контрольную формулу (при x=0.5) и выведем на экран
Func = 3 * t.exp((1/3)*t.log(1+x)) - 3
print ('Контрольное значение функции =  {0:9.6f}'.format(Func))


# для правильного вывода нужного количества знаков после запятой (зависит от порядка Eps) в таблице → посчитаем «число знаков после запятой в Eps» +1*
z = t.ceil( t.fabs(t.log(Eps)/t.log(10)) ) + 1
zs = str(z)   # и преобразуем целое число z в символьный вид, например, число 5 → символ '5'


#---------- вывод заголовка таблицы
print('N |      X      |     S(X)    | K |      F(X)     |  |S(X)-F(X)|')
for j in range(0, 80, 1):  # обычно хватает 80 символов под таблицу
```

* Использованы свойства: 1) $\log_{10} 10^{-4} = -4$; 2) $\log_a b = {}^{\log_c b}/_{\log_c a}$.

```
    print('=', end='') # на экране будет: '================================'

for j in range(len(X)):          # цикл по всем значениям массива X
    x = X[j]    # возьмем из массива очередное значение аргумента


    #-------- блок инициализации перед вычислением

    i = 1        # первый шаг
    s_i = x     # первое слагаемое
    Sums = s_i    # частичная сумма на первом шаге равна первому слагаемому

    while (t.fabs(s_i)  >=  Eps) and (i != MAX_COUNT):  # если очередное
            # слагаемое  ≥  EPS и число итераций не достигло
            # установленного предела, тогда продолжим накопление суммы
        i += 1    # увеличим номер слагаемого (и счетчик итераций K, соответственно)
        s_i = (-s_i)*(3*i-4)*x/(3*i)    #вычисляем  очередное слагаемое
        Sums = Sums + s_i     # накапливаем частичную сумму


    #  закончили вычисления для очередного X[j], надо вывести в таблицу все данные

    # сначала сформируем строку форматного вывода

    format_s = '{0:2d}|{1:17.' +zs+ 'f}|{2:17.' +zs+ 'f}|{3:4d}|{4:17.' +zs+ 'f}|{5:17.' +zs+'f}|'

    print (format_s.format(j, X[j], Sums, i, Func, t.fabs(Sums-Func))  # и применим к ней метод format()
```