```python
# coding: utf-8

import numpy as np
import math

def Leg_pol(n,x):
    if n < 0:
        print 'n cannot be less than 0!!!'
        return None
    elif n == 0: return 1
    elif n == 1: return x
    else:
        P_0 = 1; P_1 = x; P_2 = 0
        for i in range(1,n):
            P_2 = (P_1 * x * (2*i + 1) - P_0 * i) / float(i + 1)
            P_0 = P_1; P_1 = P_2
        return P_2

def Leg_pol_list(n,x):
    tmp = [1, x]
    for i in range(1,n+1):
        tmp.append(((2*i+1) * x * tmp[i] - tmp[i-1] * i) / float(i + 1))
    return tmp

def Leg_pol_der(n,x):
    P = Leg_pol_list(n,x)
    return n / (1 - x**2) * (P[n-1] - x * P[n])

def Leg_roots(n):
    roots = [math.cos(math.pi * (4*i-1) / float(4*n+2)) for i in range(1,n+1)]
    norm0 = 0; norm = np.linalg.norm(roots)
    while abs(norm - norm0) > 0.000000001:
        norm0 = norm
        roots = [item - Leg_pol(n,item)/Leg_pol_der(n,item) for item in roots]
        norm = np.linalg.norm(roots)
    return roots

def Gauss_weights(n):
    roots = Leg_roots(n)
    weights = [2/((1 - roots[i]**2)*(Leg_pol_der(n,roots[i]))**2) for i in range(n)]
    return weights

def Gauss_integr(n,function):
    roots = Leg_roots(n)
    weights = Gauss_weights(n)
    res = 0
    for i in range(n):
        res += weights[i] * function(roots[i])
    return res

def Gauss_integr_ar(n,array):
    if (n != len(array)):
        print 'array length is not equal to the number of nodes!'
        return None
    weights = Gauss_weights(n)
    res = 0
    for i in range(n):
        res += weights[i] * array[i]
    return res
```