```python
"""
Registration: xxxx;
Description: Integration by Trapezoidal Method
             int_(a*Pi)^(b*Pi) f(x)dx = h/2*[ (y_a + y_b) + 2*(y_1+y_2+...+y_{n-1}) ]
             OR optimize the last part.
Author: AKB
"""
import numpy as np
import scipy.integrate as sci

def f(x):
    #return np.exp(-pow(x,2))*np.sin(x)   # 0,1
    #return np.sqrt(x)*np.exp(x)          # 0,1
    return np.sqrt(1-pow(x,2))*np.cos(x)  # 0,0.25

# Main
# Enter the integration limits
a,b = input('Enter the lower and upper multiplier of pi: ')
a *= np.pi
b *= np.pi

# Compute 1/2*h*(y_a + y_b)
h  = b-a
s1 = (f(a)+f(b))*0.5
tol = 1e-8    # Given in CU exam paper
I0 = 1E-16    # np.finfo(float).eps
I1 = h*s1

# (a) We reach to h/2 and add f(x)
# (b) h/4  it, compare error and add 4 f(x)
# (c) h/8  it, compare error and add 8 f(x)
# (d) h/16 it, compare error and add 16 f(x) and so on

# Do the integral
i=1
while(abs((I1-I0)/I1)>tol):
    x = a+h*0.5
    for j in range(i):
        s1 += f(x)
        x  += h
        #print 'x=',x,'i=',i,'h=',h #check what's happening
    h *= 0.5 # Step halving
    i *= 2   # Trapezoid doubling
    I0 = I1
    I1 = h*s1

print 'Number of iterations : ', i/2
print 'Integral_',a,'^',b,' sqrt(1-x^2)*cos(x) dx = ', I1

# Using Numpy/Scipy
x = np.linspace(a, b, 128)
print 'Integral_',a,'^',b,' sqrt(1-x^2)*cos(x) dx = ', sci.trapz(f(x),x)
print 'Integral_',a,'^',b,' sqrt(1-x^2)*cos(x) dx = ',  np.trapz(f(x),x)

""" Results
integral_0^pi      e^(-x^2)sin(x) dx =  0.424424960052, # of iterations = 128
integral_0^pi       sqrt(x)exp(x) dx = 32.8314951592,   # of iterations = 128
integral_0^pi/4 sqrt(1-x^2)cos(x) dx =  0.632994724522, # of iterations = 32
"""
```