```python
"""
CH E 210B, HW3, Problem 5

Python script to compute a volume integral of
f(x, y, z) = (x^2 + y^2 + z^2)sin(xyz) over the
volume x = [0, 4], y = [-1, 2], z = [0, 1] using a Monte Carlo Method.

We then compute the "exact" result using Mathematica's `NIntegrate` function
and show how the error scales with the number of points used in the MC method
"""
import numpy as np
import matplotlib.pyplot as plt

# Global variables
# Box size:
XMIN = 0
XMAX = 4
YMIN = -1
YMAX = 2
ZMIN = 0
ZMAX = 1
# Volume of box
VOL = (XMAX - XMIN) * (YMAX - YMIN) * (ZMAX - ZMIN)

# Mathematica "exact" result, calculated using the command:
# NIntegrate[(x^2 + y^2 + z^2)*Sin[x*y*z], {x, 0, 4}, {y, -1, 2}, {z, 0, 1}]
EXACT_RES = 10.66249863235283

# Define function whose integral we evaluate
def f(x, y, z):
    """
    Function to numerically integrate.
    Parameters:
    -----------
    x, y, z : numpy.array
        arrays of random points
    Returns:
    --------
    np.array
        array of function values evaluated at each random point
    """
    return (x**2 + y**2 + z**2)*np.sin(x*y*z)


def mc_integral(NPTS):
    """
    Approximates the volume integral of the function described above using a
    Monte Carlo method.
    Parameters:
    -----------
    NPTS : int
        number of MC random points
    Returns:
    --------
    float
        MC approximation of the integral
    """
    # Construct arrays of random points draw from uniform distribution
    x = np.random.uniform(low=XMIN, high=XMAX, size=NPTS)
```

```python
    y = np.random.uniform(low=YMIN, high=YMAX, size=NPTS)
    z = np.random.uniform(low=ZMIN, high=ZMAX, size=NPTS)

    # Evaluate function at each random point
    f_vals = f(x, y, z)
    f_avg = np.sum(f_vals) / NPTS

    # Return MC approximation = (Box Volume) * (Avg f)
    return f_avg * VOL

# Evaluate integral for N = 10000 points
mc_10000 = mc_integral(10000)
print("MC Approximation, N = 10000:", mc_10000)

# Evaluate integral for increasing NPTS, plot error vs N
NMIN = 1
NMAX = 100000
N = np.arange(NMIN, NMAX + 1, 5)
err = np.zeros(len(N))
for i in range(len(N)):
    err[i] = np.abs(EXACT_RES - mc_integral(N[i]))

f, ax = plt.subplots(figsize=(6, 6))
act_err, = ax.plot(N, err, '.')
pred_err, = ax.plot(N, 1/np.sqrt(N))
ax.set_xlabel(r"Number of Points, $N$")
ax.set_ylabel("Error")
ax.set_title(r"MC Approximation of $\int_0^4 dx \int_{-1}^2 dy \int_0^1 dz(x^2+y^2+z^2)\sin(xyz)$")
ax.annotate("Expected Result: {:.3f}".format(EXACT_RES), xy=(60000, 10))
ax.annotate("N = 10000 Result: {:.3f}".format(mc_10000), xy=(60000, 5))
ax.legend([act_err, pred_err], ["Actual Error", r"Expected Error ~ $N^{-1/2}$"])
f.savefig("mc_integral.png")
plt.show()
```