import struct

import math

class MD5:

def \_\_init\_\_(self, message):

self.msg = message

self.h0 = 0x67452301

self.h1 = 0xEFCDAB89

self.h2 = 0x98BADCFE

self.h3 = 0x10325476

self.k = self.calculate\_table()

self.s = [7, 12, 17, 22, 5, 9, 14, 20, 4, 11, 16, 23, 6, 10, 15, 21] \* 4

@staticmethod

def left\_rotate(x, c):

return (x << c) | (x >> (32 - c))

@staticmethod

def calculate\_table():

k = []

for i in range(64):

k.append(int((abs(math.sin(i + 1)) \* (2 \*\* 32))))

return k

def padding(self):

original\_byte\_len = len(self.msg)

original\_bit\_len = original\_byte\_len \* 8

self.msg += b'\x80'

while (len(self.msg) \* 8) % 512 != 448:

self.msg += b'\x00'

self.msg += struct.pack('<Q', original\_bit\_len)

def process\_chunk(self, chunk):

a, b, c, d = self.h0, self.h1, self.h2, self.h3

M = struct.unpack('<' + 'I'\*16, chunk)

for i in range(64):

if i < 16:

f = (b & c) | (~b & d)

g = i

elif i < 32:

f = (d & b) | (~d & c)

g = (5 \* i + 1) % 16

elif i < 48:

f = b ^ c ^ d

g = (3 \* i + 5) % 16

else:

f = c ^ (b | ~d)

g = (7 \* i) % 16

f = (f + a + self.k[i] + M[g]) & 0xFFFFFFFF

a, d, c, b = d, c, b, (b + self.left\_rotate(f, self.s[i])) & 0xFFFFFFFF

self.h0 = (self.h0 + a) & 0xFFFFFFFF

self.h1 = (self.h1 + b) & 0xFFFFFFFF

self.h2 = (self.h2 + c) & 0xFFFFFFFF

self.h3 = (self.h3 + d) & 0xFFFFFFFF

def digest(self):

self.padding()

for i in range(0, len(self.msg), 64):

self.process\_chunk(self.msg[i:i+64])

return struct.pack('<4I', self.h0, self.h1, self.h2, self.h3)

def hexdigest(self):

return ''.join(['{:02x}'.format(byte) for byte in self.digest()])

def main():

msg = "The quick brown fox jumps over the lazy dog"

md5 = MD5(msg.encode())

print("\t MD5 ENCRYPTION ALGORITHM IN PYTHON \n")

print(f"Input String to be Encrypted using MD5 :\n\t{msg}")

print("\nThe MD5 code for input string is:\n\t= 0x" + md5.hexdigest())

print("\n\t MD5 Encryption Successfully Completed!!!\n")

if \_\_name\_\_ == "\_\_main\_\_":

main()

