def to\_decimal(num, base: int):

global bin

bin= ''

if base == 2:

num = int(num)

decimal, i = 0, 0

while(num != 0):

dec = num % 10

decimal = decimal + dec \* pow(2, i)

num = num//10

i += 1

return int(decimal)

if base == 8:

num = n

decimal = 0

base = 1

temp = num

while (temp):

last\_digit = temp % 10

temp = int(temp / 10)

decimal += last\_digit \* base

base = base \* 8

return decimal

if base == 16:

table = {'0': 0, '1' : 1, '2' : 2, '3' : 3, '4' : 4, '5' : 5, '6' : 6, '7' : 7, '8' : 8, '9' : 9, 'A' : 10 , 'B' : 11, 'C' : 12, 'D' : 13, 'E' : 14, 'F' : 15}

hexadecimal\_string = num

decimal = 0

length = len(hexadecimal\_string) -1

for digit in hexadecimal\_string:

decimal += table[digit]\*16\*\*length

length -= 1

return decimal

return -1

bin = ''

def to\_binary(num, base = None):

global bin

if base != None:

num = to\_decimal(num, base)

if num >= 1:

to\_binary(num // 2)

bin = bin + f"{num%2}"

return int(bin)

def to\_octal(num, base = None):

if base != None:

num = to\_decimal(num, base)

octal = 0

count = 1

deci = num

while (deci != 0):

remainder = deci % 8

octal += remainder \* count

count = count \* 10

deci = deci // 8

return octal

def to\_hexadecimal(num, base = None):

if base != None:

num = to\_decimal(num, base)

table = {0: '0', 1: '1', 2: '2', 3: '3', 4: '4', 5: '5', 6: '6', 7: '7', 8: '8', 9: '9', 10: 'A', 11: 'B', 12: 'C', 13: 'D', 14: 'E', 15: 'F'}

hexadecimal = ''

while(num > 0):

remainder = num % 16

hexadecimal = table[remainder] + hexadecimal

num = num // 16

return str(hexadecimal)

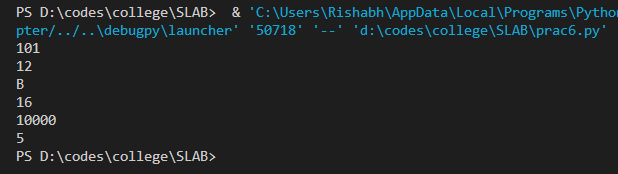
print(to\_binary(5))

print(to\_octal(10))

print(to\_hexadecimal(11))

print(to\_decimal('10', 16))

print(to\_binary('10', base=16))

print(to\_octal('0101', base=2))

def gcd( a, b):

if a > b:

temp = b

else:

temp = a

for i in range(1, temp + 1):

if (( a % i == 0) and (b % i == 0 )):

gcd = i

return gcd

def lcm(x, y):

if x > y:

greater = x

else:

greater = y

while(True):

if((greater % x == 0) and (greater % y == 0)):

lcm = greater

break

greater += 1

return lcm

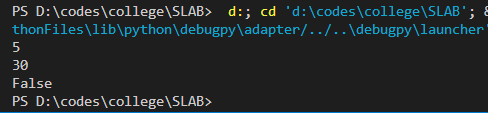
def is\_coprime(x, y):

return gcd(x, y) == 1

print(gcd(10, 15))

print(lcm(10, 15))

print(is\_coprime(10, 15))



import numpy as np

from numpy import version

print("i")

list = [2,3,54,3,75,8,67,45,63,45]

listarr = np.array(list)

print(listarr)

tuple = (12,34,4,53,64,76,46,52,34,325)

tuplearr = np.array(tuple)

print(tuplearr)

print(type(tuplearr))

print(version.full\_version)

print()

print("ii")

m = np.zeros(shape=(3, 3))

print(m)

m = np.ones(shape=(2, 2))

print(m)

print()

print("iii")

arr2d = np.array([[1,24, 23], [24,32,34], [42,1,2]])

for i in range(3):

for j in range(3):

print(arr2d[i][j], end=" ")

print()

print()

print("iv")

m = np.concatenate([[23,42,342], [67,5,4]], axis = 0)

print(m)

print()

print("v")

m1 = [12,33,5,356,23,1,3]

m2 = [11,23,5,356,23,1,3]

m = np.add(m1, m2)

print(m)

m3 = np.multiply(m1, m2)

print(m3)

m4 = np.transpose(m1)

print(m4)

