import random

# Funkcja f Tego ryc nie musze bo to przykladowa funkcja jest

def f(x):

return x \*\* 2 - 3 \* x - 4

# Prostokaty Start to od lewej do prawej

def prostokatyStart(a, b, n):

dx = (b - a) / n

suma = 0

for i in range(n):

suma += f(a + dx \* i) \* dx

return suma

def prostokatyMid(a, b, n):

dx = (b - a) / n

suma = 0

for i in range(n):

suma += f(a + dx \* i + dx / 2) \* dx

return suma

#END to chodzi ze od prawej do lewej

def prostokatyEnd(a, b, n):

dx = (b - a) / n

suma = 0

for i in range(n):

suma += f(a + dx \* i + dx) \* dx

return suma

# Trapezy

def trapez1(a, b, n):

dx = (b - a) / n

suma = 0

for i in range(n):

suma += (f(a+ dx \* i) + f(a + dx \* i + dx)) / 2 \* dx

return suma

def trapez2(a, b, n):

dx = (b - a) / n

suma = f(a) + f(b)

for i in range(1, n):

suma += 2 \* f(a + dx \* i)

return suma / 2 \* dx

# Bisekcja - poszukiwanie miejsca zerowego funkcji

def bisekcja1(a, b, epsilon):

while abs(a - b) > epsilon:

c = (a + b) / 2

if f(c) == 0:

return c

elif f(a) \* f(c) < 0:

b = c

else:

a = c

return (a + b) / 2

def bisekcja2(a, b, n):

for i in range(n):

c = (a + b) / 2

if f(c) == 0:

return c

elif f(a) \* f(c) < 0:

b = c

else:

a = c

return (a + b) / 2

# Newton-Raphson - algorytm poszukiwania pierwiastka z liczby rzeczywistej

def pierwiastek1(p, epsilon):

a = p

b = 1

while a - b > epsilon:

a = (a + b) / 2

b = p / a

return a

def pierwiastek2(p, n):

a = p

b = 1

for i in range(n):

a = (a + b) / 2

b = p / a

return a

# Wyszukiwanie Leadera w tablicy - liczby, wystepujacej wiecej, niz polowa razy

Table = [2, 1, 1, 3, 1, 2, 1]

def leader(T):

leader = T[0]

ilosc = 1

for i in range(1, len(T)):

if ilosc == 0:

leader = T[i]

ilosc = 1

else:

if T[i] == leader:

ilosc += 1

else:

ilosc -= 1

if ilosc == 0:

print("Brak leadera")

else:

ilosc\_leaderow = 0

for i in range(len(T)):

if T[i] == leader:

ilosc\_leaderow += 1

if ilosc\_leaderow > len(T) // 2:

print(f"Leader: {leader} \nIlosc: {ilosc\_leaderow}")

else:

print("Brak leadera")

# Monte-Carlo - algorytm wyszukiwania wartosci Pi

def MonteCarlo(n): # n - ilosc "strzalow"

c = 0 # c - Ilosc "strzalow" trafionych w kolko

for i in range(n):

x = random.uniform(-1,1)

y = random.uniform(-1,1)

if x \*\* 2 + y \*\* 2 <= 1:

c += 1

return 4 \* c / n