import time

import matplotlib.pyplot as plt

# === Cube Root Function ===

def simple\_cube\_root(n):

steps = 0

is\_negative = n < 0

n = abs(n)

low = 0

high = n

precision = 0.0000000001

while low <= high:

steps += 1

mid = (low + high) / 2

cube = mid \* mid \* mid

if abs(cube - n) < precision:

return -mid if is\_negative else mid, steps

elif cube < n:

low = mid

else:

high = mid

return -mid if is\_negative else mid, steps

# === Graph Drawing Function ===

def draw\_graphs(digits, step\_counts, time\_taken):

fig, (left, right) = plt.subplots(1, 2, figsize=(12, 5))

# Steps graph

left.plot(digits, step\_counts, marker='o', color='blue', label='Steps')

left.set\_title("Steps vs. Number of Digits")

left.set\_xlabel("Digits in Number")

left.set\_ylabel("Steps Taken")

left.grid(True)

left.legend()

# Time graph

right.plot(digits, time\_taken, marker='s', color='red', label='Time (s)')

right.set\_title("Time vs. Number of Digits")

right.set\_xlabel("Digits in Number")

right.set\_ylabel("Time (seconds)")

right.grid(True)

right.legend()

plt.tight\_layout()

plt.show()

# === Collect and Process Data ===

digits\_list = []

steps\_list = []

times\_list = []

# Test numbers with increasing digit lengths

for num\_digits in range(1, 11): # 1 to 10 digits

test\_number = 10\*\*(num\_digits - 1) + 5 # e.g., 15, 105, 1005...

start\_time = time.time()

\_, steps = simple\_cube\_root(test\_number)

end\_time = time.time()

time\_taken = end\_time - start\_time

digits\_list.append(num\_digits)

steps\_list.append(steps)

times\_list.append(time\_taken)

print(f"{test\_number}: Steps={steps}, Time={time\_taken:.8f} seconds")

# === Draw Graphs ===

draw\_graphs(digits\_list, steps\_list, times\_list)