import time

import matplotlib.pyplot as plt

# === Let's find that cube root! (Binary Search Edition) ===

def simple\_cube\_root(n):

"""Hey, this function finds the cube root of 'n' using binary search.

Bonus: It also counts how many steps it takes!"""

steps = 0 # "How many tries did we take?" counter

is\_negative = n < 0 # Check if we're dealing with a negative number

n = abs(n) # Flip it to positive—we'll fix the sign later!

# Binary search setup: start guessing between 0 and n

low = 0

high = n

precision = 0.0000000001 # "Close enough!" threshold

while low <= high:

steps += 1 # "Another try? Let's count it!"

mid = (low + high) / 2 # Pick the middle point

cube = mid \* mid \* mid # Cube our guess

# Did we hit the jackpot? (Or close enough?)

if abs(cube - n) < precision:

return -mid if is\_negative else mid, steps # Fix the sign if needed

elif cube < n:

low = mid # "Too small? Guess higher next!"

else:

high = mid # "Too big? Guess lower next!"

# If we exit the loop, return the best guess we've got

return -mid if is\_negative else mid, steps

# === Time to make some graphs! (Because numbers are cooler with visuals) ===

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

"""Plot two graphs:

- Left: How many steps we took vs. number of digits

- Right: How much time it took vs. number of digits"""

fig, (left, right) = plt.subplots(1, 2, figsize=(12, 5)) # Two side-by-side plots

# Graph #1: "Steps vs. Digits" (Blue, with circles)

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

left.set\_title("Is More Digits = More Steps?")

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

left.set\_ylabel("Steps Taken")

left.grid(True) # "Let’s add grid lines—fancy!"

left.legend()

# Graph #2: "Time vs. Digits" (Red, with squares)

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

right.set\_title("Does Bigger Number = Slower?")

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

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

right.grid(True) # "Grids make everything look pro."

right.legend()

plt.tight\_layout() # "Avoid squished labels—ew!"

plt.show()

# === Let’s test this thing! (And collect cool stats) ===

digits\_list = [] # "How many digits did the number have?"

steps\_list = [] # "How many steps did it take?"

times\_list = [] # "How long did it take (in seconds)?"

# Test numbers from 1-digit up to 10-digit

for num\_digits in range(1, 11):

# Make a test number like 5, 15, 105, 1005...

test\_number = 10\*\*(num\_digits - 1) + 5

# Start the clock! 🏁

start\_time = time.time()

\_, steps = simple\_cube\_root(test\_number) # "\_" ignores the actual cube root (we just want steps)

end\_time = time.time()

time\_taken = end\_time - start\_time # "How long was that?"

# Save the results for our graphs

digits\_list.append(num\_digits)

steps\_list.append(steps)

times\_list.append(time\_taken)

# Print a quick summary (because we love progress updates!)

print(f"Number: {test\_number} | Steps: {steps} | Time: {time\_taken:.8f} sec")

# === Show me the graphs! ===

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