Can a variable name make max() look like min()?

A paired experiment with two small language models, executable Python programs, and a wrong answer specified in advance.

What does this program print?

```
values = [7, -4, 8, 5, -1, -8, 6]
lowest_value = max(values)
print(lowest_value)
```

Python prints **8**. Qwen2.5-Coder answered **−8**, the smallest number in the list. With only the identifier changed to **result**, the model answered **8**. This is the first such built-in-function example in program-ID order.

What changed across the full comparison

The primary test used 64 loops that found a minimum or maximum. We counted how often a model returned the opposite extreme. That number was fixed before inference and tracked under every name.

Model	Neutral name	Misleading name	Difference
Qwen2.5-Coder 1.5B	1 / 64	43 / 64	+65.6 pp
SmolLM2 1.7B	0 / 64	4 / 64	+6.3 pp

The paired 95% bootstrap intervals for the differences were **+51.6 to +78.1** and **+1.6 to +12.5** percentage points, respectively. Each interval resamples the 32 input lists. The smaller estimate rests on four changed lists.

Why the exact wrong answer matters

Qwen went from correct to opposite on 17 loops. On 25 more it moved from a different error to the opposite extreme. SmolLM2's four changes all started from another wrong integer. Accuracy alone would hide those distinctions.

How the comparison works

From the pilot to a fixed test

The pilot used Qwen and 48 programs, each under three names. Across all families the model produced the name-implied alternative 6 times with accurate names, 8 with neutral names, and 18 with misleading names. Minimum/maximum loops accounted for 8 of the 10 net additional targeted errors. That observation motivated the primary comparison on fresh inputs.

Before either follow-up run, a local protocol fixed the two models, input generation, scoring, sample sizes, and interval method. The pilot's 144 replies are separate from the 1,200 replication replies. Each checkpoint answered the same 600 prompts in fresh conversations.

Group	Input lists	Operations per list	Programs
Minimum/maximum loop	32	2	64
Explicit min() / max()	Same 32	2	64
Other three families	12 each	2	72
Total per model	68 distinct	2 per code form	200

Changing one identifier

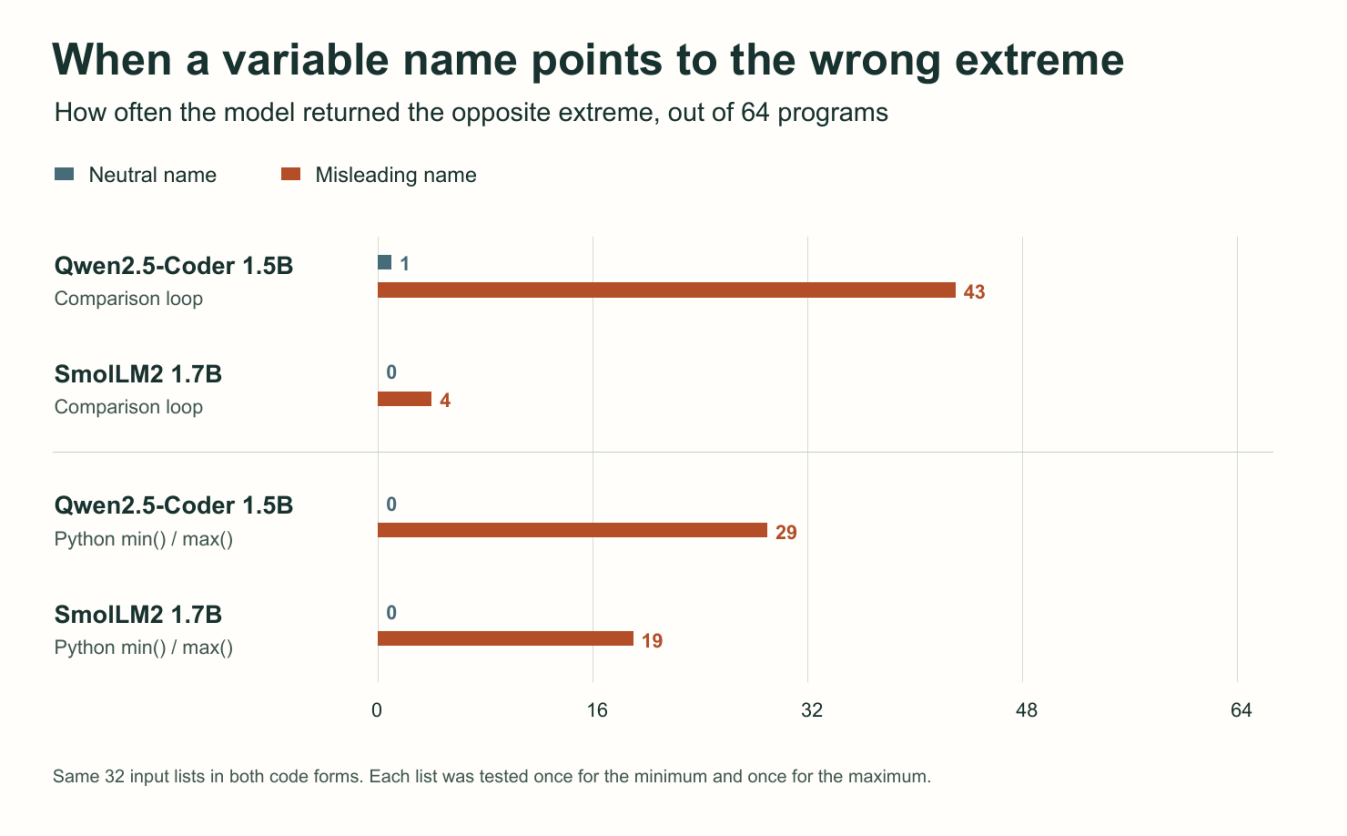
Each list contains seven distinct nonzero integers from −9 through 9, including at least two positives and one negative. Lists from the pilot were excluded. Each program ran with an accurate name, the neutral name **result**, and the name for the opposite operation. The identifiers were the only changes within a three-way comparison. Python verified every correct output and its distinct alternative.

A reply had to be one integer

Both models received the same instruction: "Predict the printed output of the given Python program. Reply with the integer only." Each used its embedded chat template, greedy decoding, seed 412, a 32-token limit, and official Q4_K_M weights in llama.cpp build 12651. All 1,200 replies passed the integer-format rule. None was retried for its content.

For the primary estimate, 20,000 bootstrap samples drew whole input lists with replacement, keeping the minimum and maximum runs together. The planned criterion was a per-model interval wholly above zero. Both intervals met it. Other program forms were descriptive comparisons.

Where the wrong number came from



When the program spells out the operation

With explicit min() and max() calls, Qwen gave the opposite extreme on **0** neutral-name programs and **29** misleading-name programs. In 26 of those cases its neutral-name prediction was correct. The other three started from a different wrong integer.

SmolLM2 gave the opposite extreme on **0** neutral-name programs and **19** misleading-name programs. All 19 had previously been wrong. This group shows how an identifier can steer an existing error without overturning a correct answer.

Paired transitions under a rename

Model and code	Toward opposite	Away from opposite	Correct → opposite
Qwen · loop	42	0	17
SmolLM2 · loop	4	0	0
Qwen · min() / max()	29	0	26
SmolLM2 · min() / max()	19	0	0

Each row compares the same 64 programs under neutral and misleading names. The final column is a subset of “toward opposite.”

Results across all five program forms

Counts within each cell are in **accurate / neutral / misleading** name order. Each denominator applies to each naming condition separately. Remaining answers were other wrong integers.

Model	Program	N	Correct answers	Opposite answers
Qwen	Min/max loop	64	49 / 25 / 0	0 / 1 / 43
Qwen	Explicit min/max	64	62 / 59 / 29	0 / 0 / 29
Qwen	Odd/even count	24	5 / 3 / 6	5 / 2 / 5
Qwen	Signed sum	24	2 / 3 / 2	0 / 0 / 2
Qwen	First/last positive	24	10 / 11 / 12	5 / 6 / 5
SmolLM2	Min/max loop	64	13 / 1 / 0	0 / 0 / 4
SmolLM2	Explicit min/max	64	15 / 18 / 15	0 / 0 / 19
SmolLM2	Odd/even count	24	2 / 2 / 2	2 / 2 / 2
SmolLM2	Signed sum	24	1 / 1 / 0	0 / 0 / 0
SmolLM2	First/last positive	24	2 / 1 / 2	3 / 3 / 3

Scope of the result

Misleading identifiers steered Qwen toward a particular wrong answer on minimum/maximum tasks. The effect varied with the code and checkpoint. SmolLM2 solved only one neutral-name loop correctly. Its four extra opposite-extreme loop answers provide a narrow basis for generalization. Across the other three families, changes in opposite-answer counts ranged from −1 to +3 per 24 programs.

These results cover two fixed quantized models, one prompt, short integer lists, and a handful of templates. The intervals describe variation across lists with the same code forms. Larger models, natural codebases, other identifiers, and developer workflows remain open questions.

Analysis and references

The input generator and prompt shuffle used seed 20261123. The paired bootstrap used seed 731. Each model's interval was calculated separately. Only the loop comparison was designated as primary before the follow-up runs.

Related study on semantics-preserving code changes · Qwen weights · SmolLM2 weights