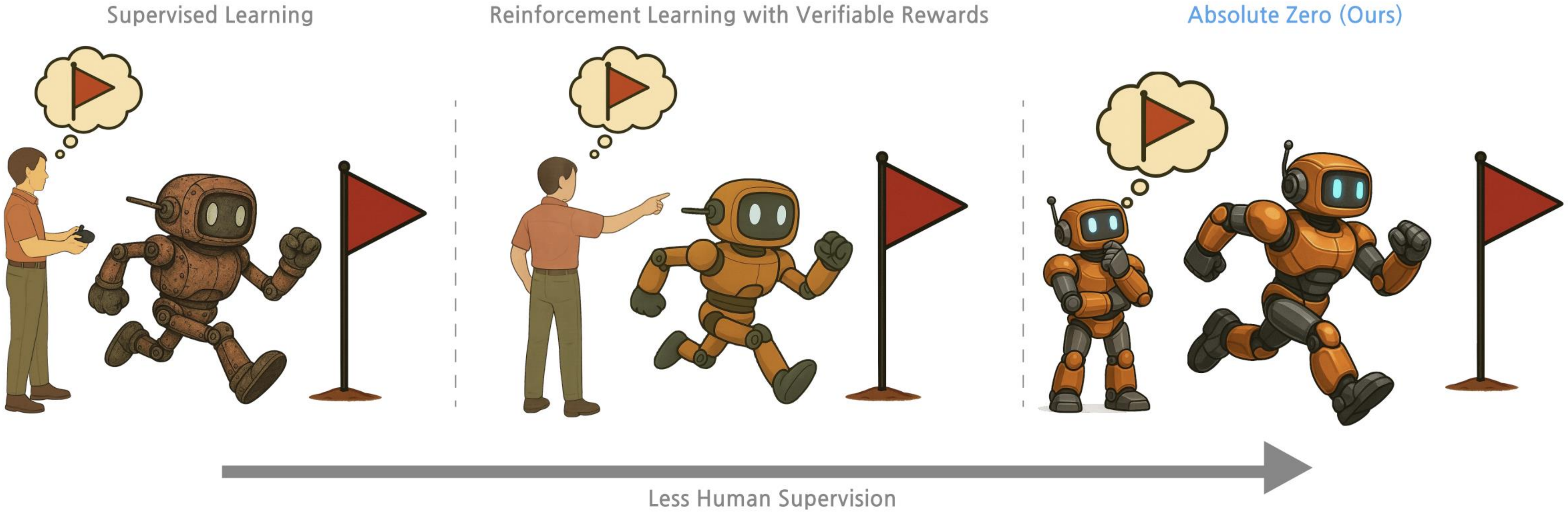
Absolute Zero: Reinforced Self-play Reasoning with Zero Data

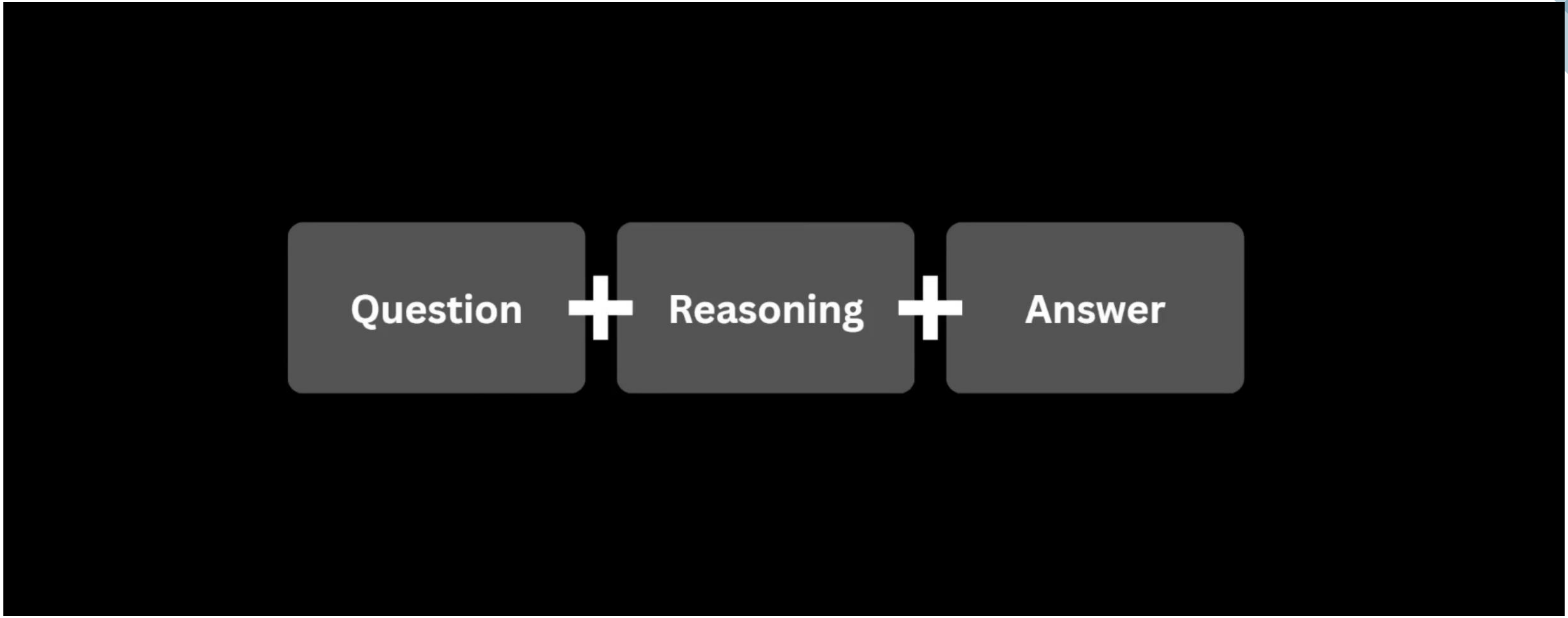


Supervised Learning

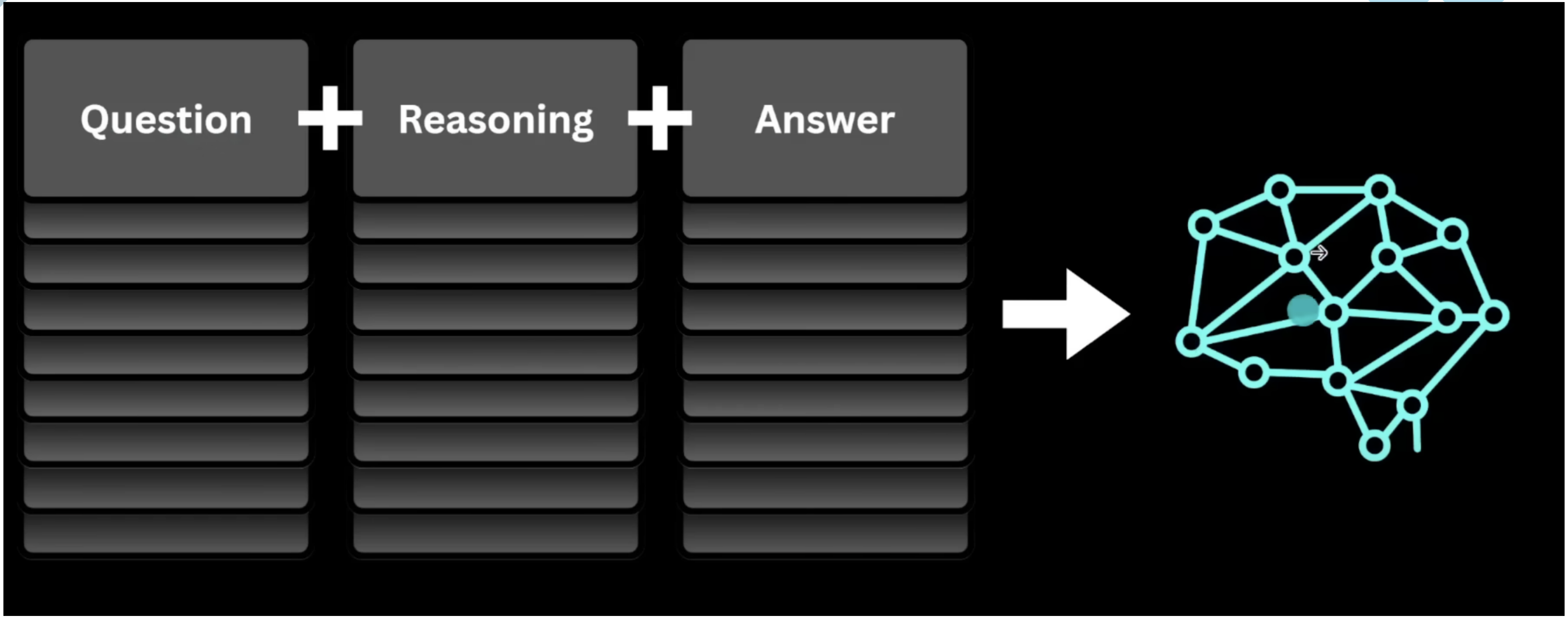
Reinforcement Learning with Verifiable Rewards

Absolute Zero (Ours)

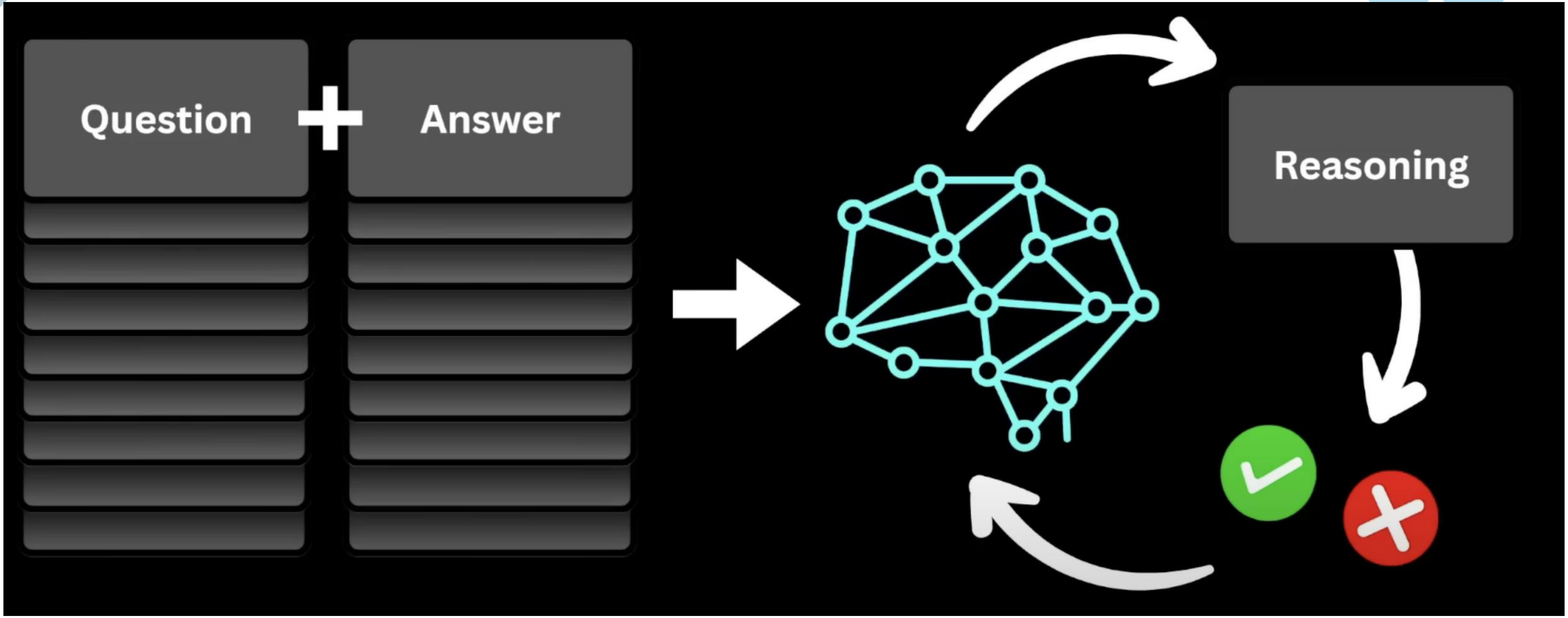
Less Human Supervision

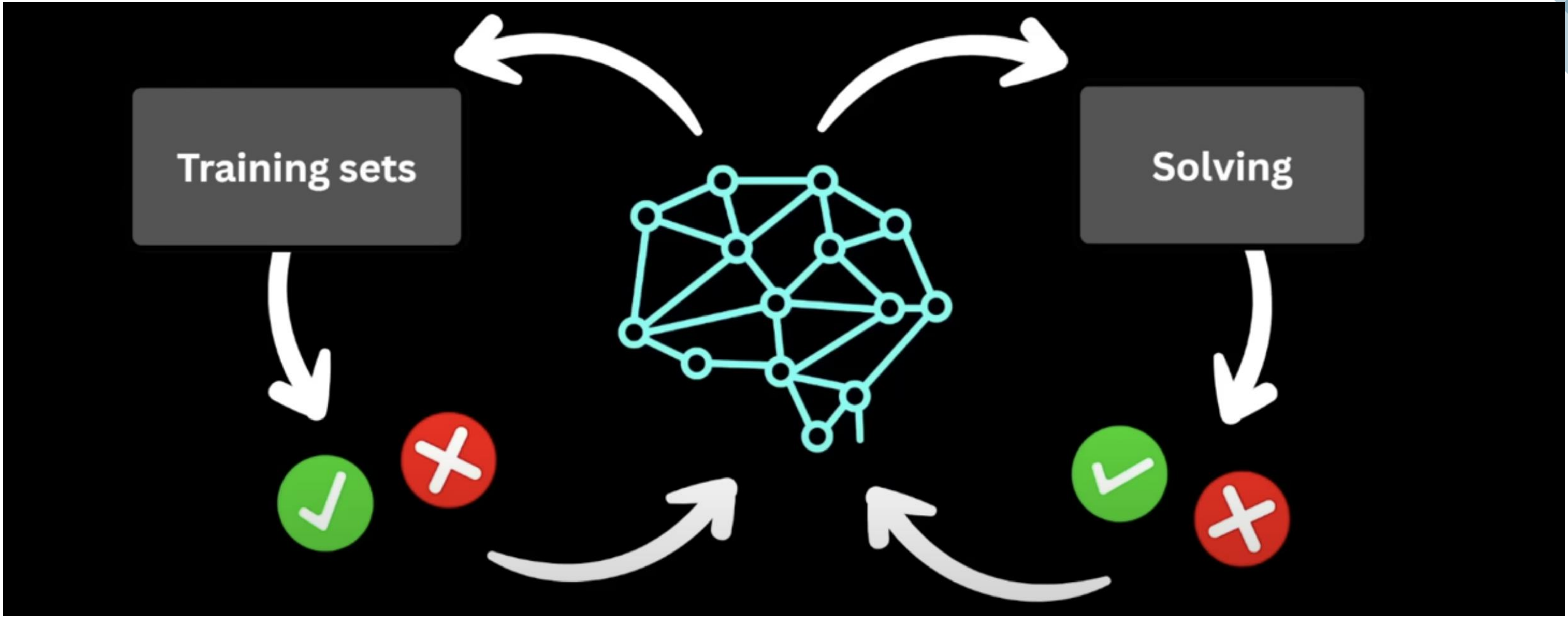


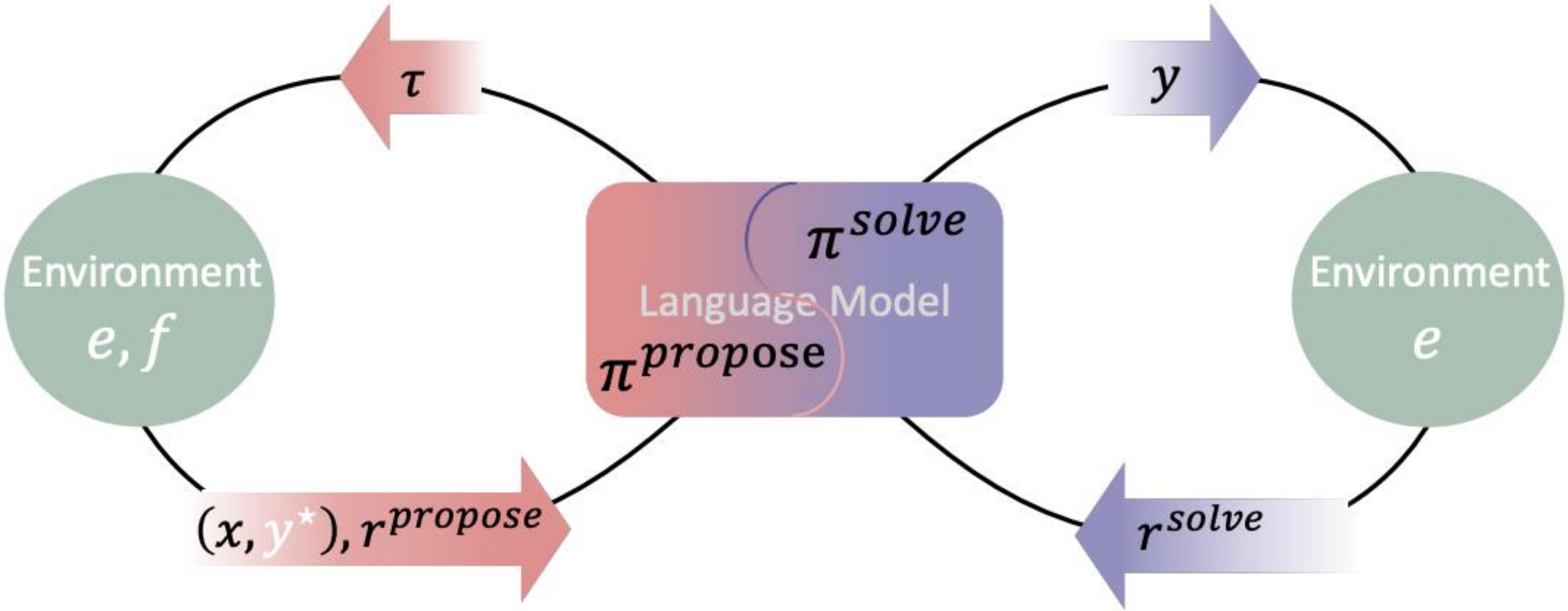
Question + Reasoning + Answer

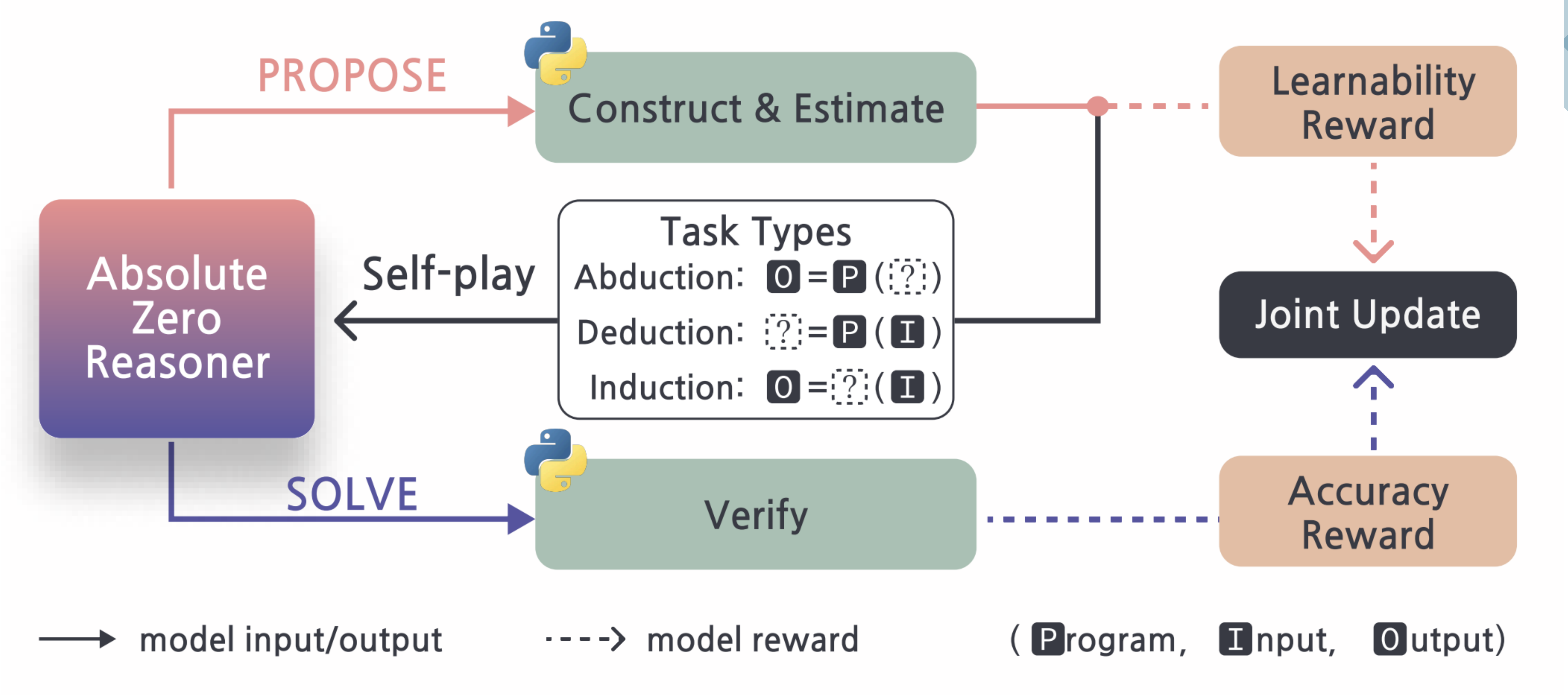


Question + Reasoning + Answer


Question
Answer
Reasoning


Training sets
Solving


τ
y
Environment
e, f
π^{solve}
Language Model
$\pi^{propose}$
Environment
e
$(x, y^{\star}), r^{propose}$
r^{solve}



PROPOSE

Construct & Estimate

Learnability Reward

Absolute Zero Reasoner

Self-play

Task Types

Abduction: O = P (?)

Deduction: ? = P (I)

Induction: O = ? (I)

Joint Update

SOLVE

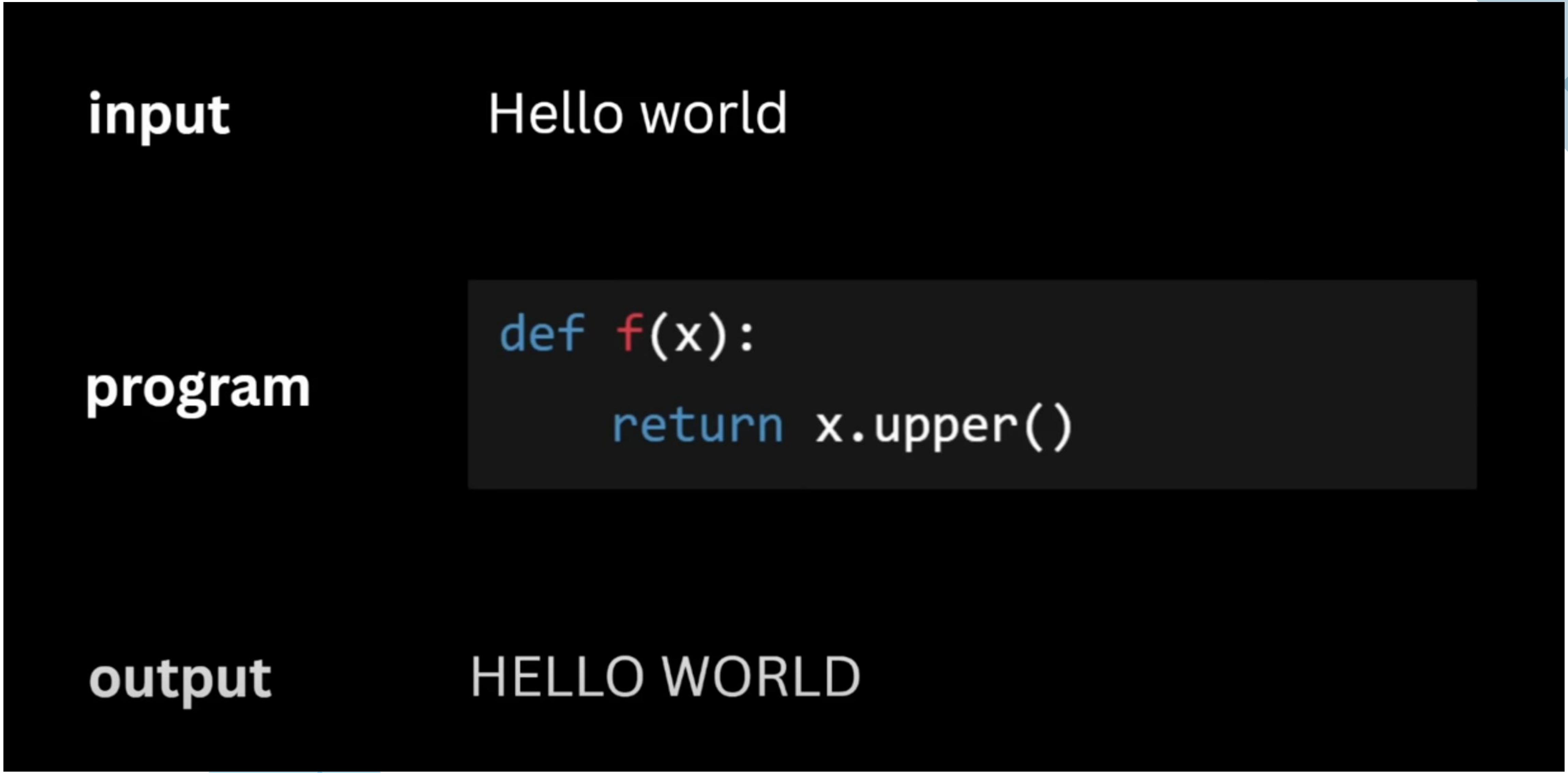
Verify

Accuracy Reward

→ model input/output

- - -> model reward

(**P**rogram, **I**nput, **O**utput)

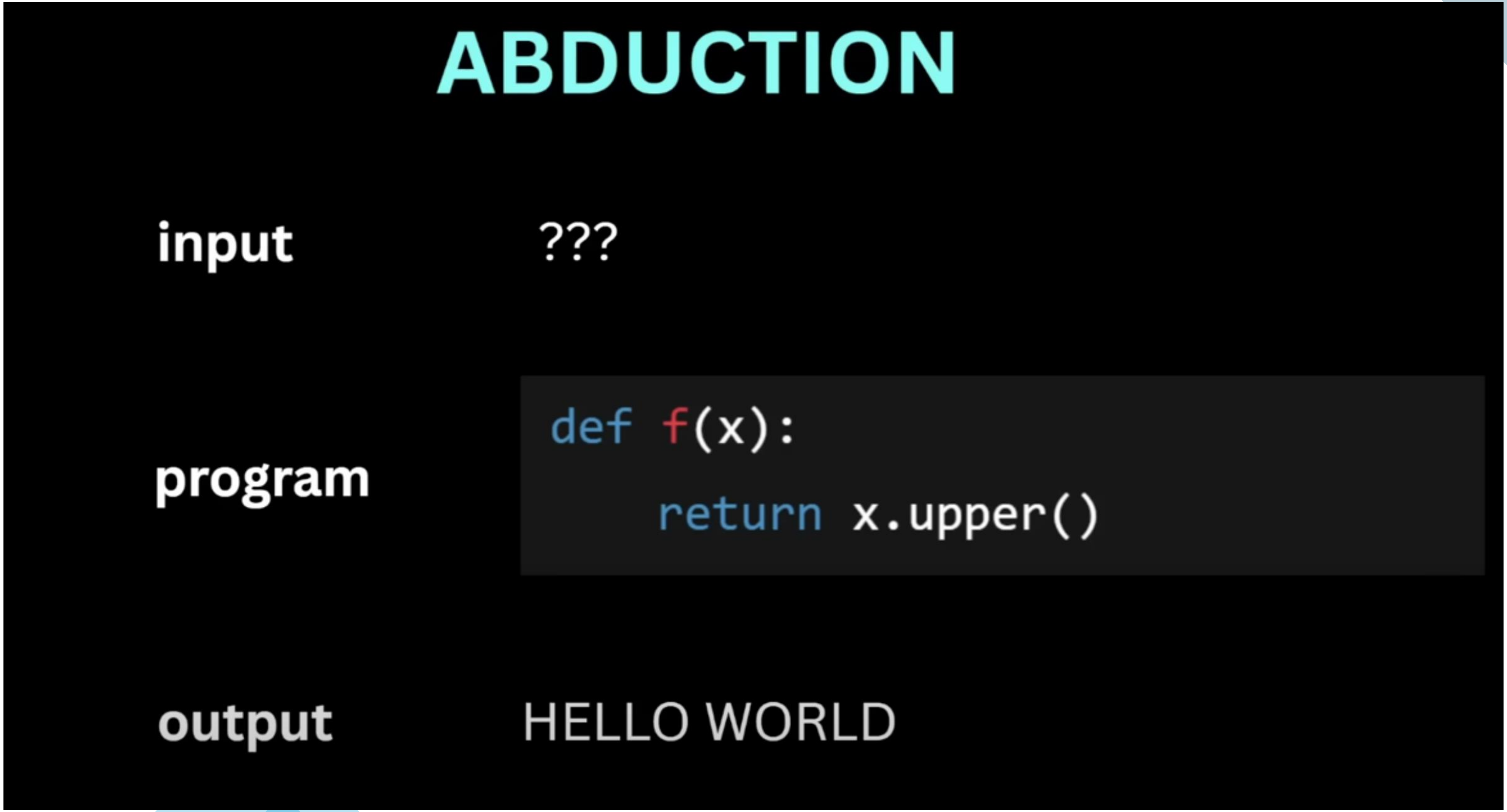


input Hello world

program

```python
def f(x):
    return x.upper()
```

output HELLO WORLD



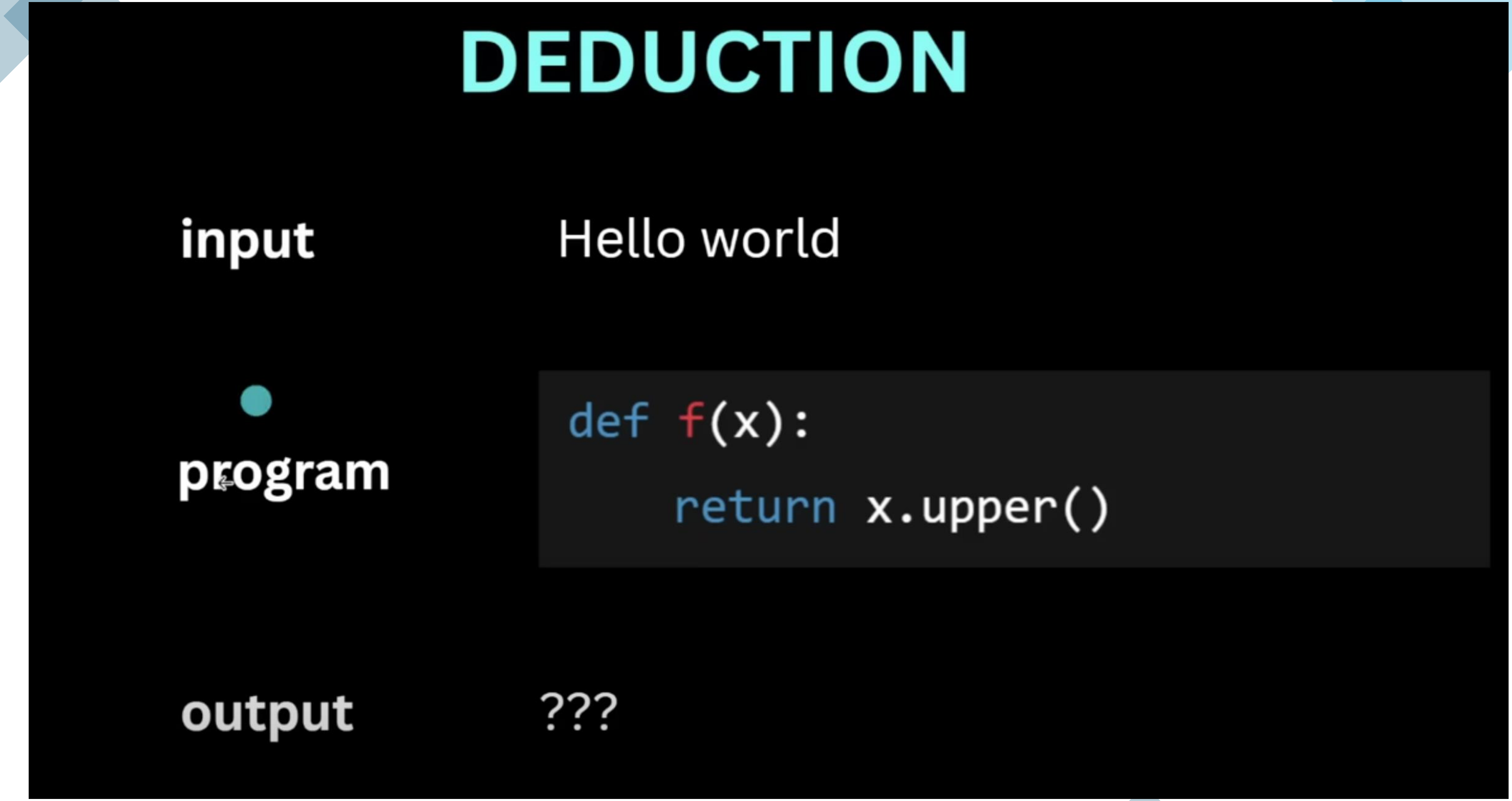
ABDUCTION

input ???

program

```
def f(x):
    return x.upper()
```

output HELLO WORLD



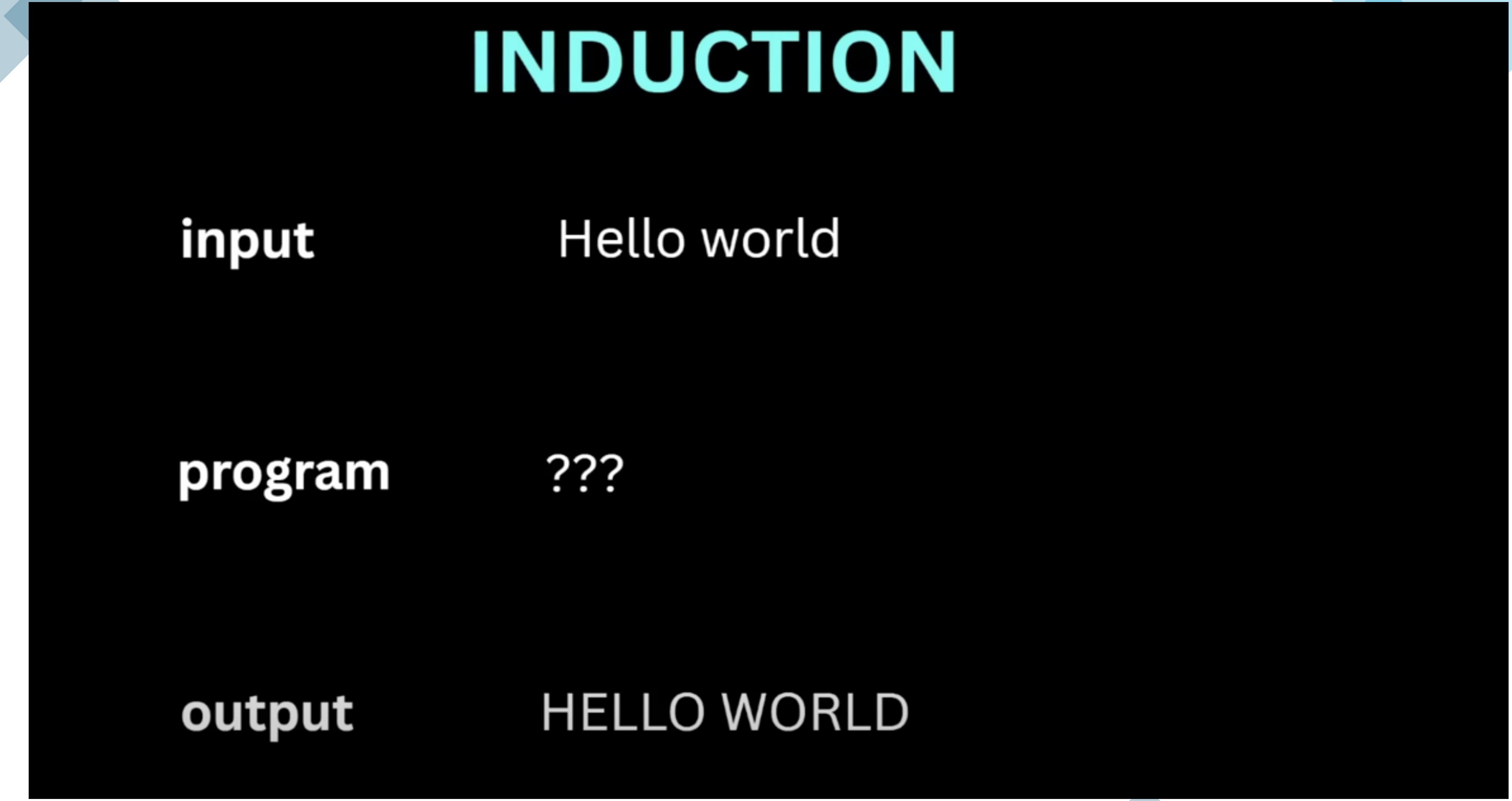
DEDUCTION

input Hello world

program

```
def f(x):
    return x.upper()
```

output ???



INDUCTION

input	Hello world
program	???
output	HELLO WORLD

Model-proposed Task

Input: [1,2,3,4,5], 5

```
def f(numbers: list[int], target: int) -> int:
    accumulated_values = []
    current_sum = 0
    for number in numbers:
        current_sum += number
        accumulated_values.append(current_sum)
    result = 0
    for i in range(len(accumulated_values)):
        for j in range(i + 1, len(accumulated_values)):
            if accumulated_values[j] \
                    - accumulated_values[i] == target:
                result = accumulated_values[i]
                return result
    return result
```

Output: 1

Response of Predict Input

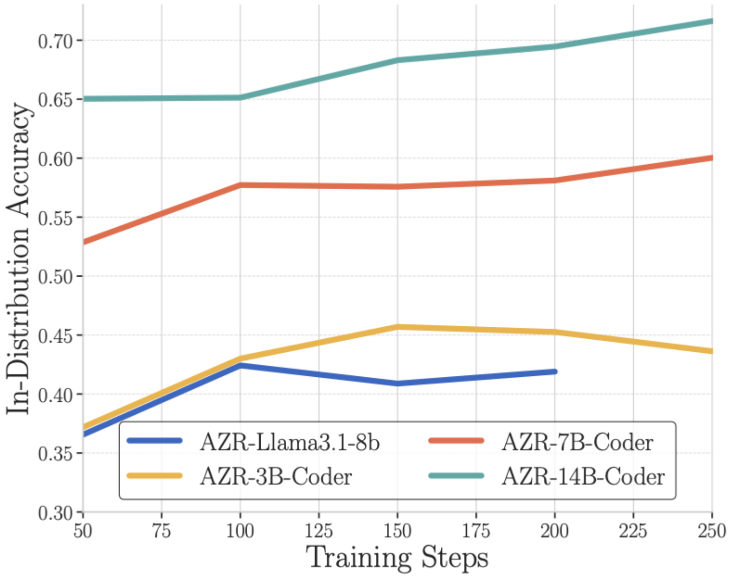
Let’s analyze the code snippet …(analysis) To find one possible input that produces the output 1, we need to find a scenario where …(analysis)
Let’s consider a simple example: numbers = [1, 2], target = 1
Here’s the step-by-step calculation:
1. current_sum = 0 + 1 = 1, accumulated_values = [1]
2. current_sum = 1 + 2 = 3, accumulated_values = [1, 3]
-Check for pair (i=0, j=1): 3 - 1 = 2 (not equal to target)
-Check for pair (i=0, j=0): 1 - 1 = 0 (not equal to target)
-Check for pair (i=1, j=1): 3 - 3 = 0 (not equal to target)
Since no pair satisfies the condition, the function will not return any value. This doesn’t match our output. Let’s try another example …(Tried 3 more examples)

Given the complexity and the fact that no simple example seems to work, let’s try a more complex scenario: numbers = [2, -1, 1], target = 1
Step-by-step calculation:
1. current_sum = 0 + 2 = 2, accumulated_values = [2]
2. current_sum = 2 - 1 = 1 , accumulated_values = [2, 1]
3. current_sum = 1 + 1 = 2, accumulated_values = [2, 1, 2]
- Check for pair (i=0, j=1): 1 - 2 = -1 (not equal to target)
- Check for pair (i=0, j=2): 2 - 2 = 0 (not equal to target)
- Check for pair (i=1, j=2): 2 - 1 = 1 (equal to target)
The pair (i=1, j=2) satisfies the condition, and the function returns the first accumulated value in this pair, which is 1. This matches our output.

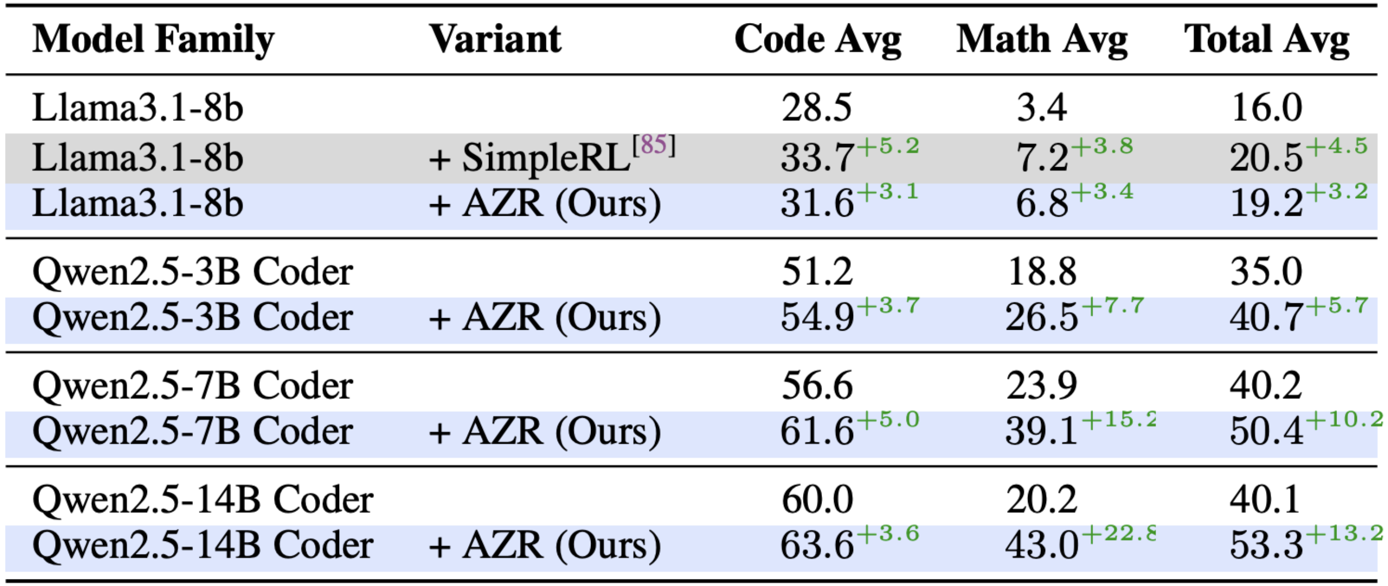
Absolute Zero: Reinforced Self-play Reasoning with Zero Data

Model	**Base**	**#data**	HEval^{+}	MBPP^{+}	LCB$^{\text{v1-5}}$	AME24	AME25	AMC	M500	Minva	Olypiad	**CAvg**	**MAvg**	**AVG**
Base Models														
Qwen2.5-7B[73]	-	-	73.2	65.3	17.5	6.7	3.3	37.5	64.8	25.0	27.7	52.0	27.5	39.8
Qwen2.5-7B-Ins[73]	-	-	75.0	68.5	25.5	13.3	6.7	52.5	76.4	35.7	37.6	56.3	37.0	46.7
Qwen2.5-7B-Coder[26]	-	-	80.5	69.3	19.9	6.7	3.3	40.0	54.0	17.3	21.9	56.6	23.9	40.2
Qwen2.5-7B-Math[74]	-	-	61.0	57.9	16.2	10.0	16.7	42.5	64.2	15.4	28.0	45.0	29.5	37.3
Zero-Style Reasoners Trained on Curated Coding Data														
AceCoder-RM[84]	Ins	22k	79.9	71.4	23.6	20.0	6.7	50.0	76.4	34.6	36.7	58.3	37.4	47.9
AceCoder-Rule[84]	Ins	22k	77.4	69.0	19.9	13.3	6.7	50.0	76.0	37.5	37.8	55.4	36.9	46.2
AceCoder-RM[84]	Coder	22k	78.0	66.4	27.5	13.3	3.3	27.5	62.6	29.4	29.0	57.3	27.5	42.4
AceCoder-Rule[84]	Coder	22k	80.5	70.4	29.0	6.7	6.7	40.0	62.8	27.6	27.4	60.0	28.5	44.3
CodeR1-LC2k[36]	Ins	2k	81.7	71.7	28.1	13.3	10.0	45.0	75.0	33.5	36.7	60.5	35.6	48.0
CodeR1-12k[36]	Ins	12k	81.1	73.5	29.3	13.3	3.3	37.5	74.0	35.7	36.9	61.3	33.5	47.4
Zero-Style Reasoners Trained on Curated Math Data														
PRIME-Zero[9]	Coder	484k	49.4	51.1	11.0	23.3	23.3	67.5	81.2	37.9	41.8	37.2	**45.8**	41.5
SimpleRL-Zoo[85]	Base	8.5k	73.2	63.2	25.6	16.7	3.3	57.5	77.0	35.7	41.0	54.0	38.5	46.3
Oat-Zero[38]	Math	8.5k	62.2	59.0	15.2	30.0	16.7	62.5	80.0	34.9	41.6	45.5	44.3	44.9
ORZ[23]	Base	57k	80.5	64.3	22.0	13.3	16.7	60.0	81.8	32.7	45.0	55.6	41.6	48.6
Absolute Zero Training w/ No Curated Data (Ours)														
AZR (Ours)	Base	**0**	$71.3^{-1.9}$	$69.1^{+3.8}$	$25.3^{+7.8}$	$13.3^{+6.6}$	$13.3^{+10.0}$	$52.5^{+15.0}$	$74.4^{+9.6}$	$38.2^{+13.2}$	$38.5^{+10.8}$	$55.2^{+3.2}$	$38.4^{+10.9}$	$46.8^{+7.0}$
AZR (Ours)	Coder	**0**	$83.5^{+3.0}$	$69.6^{+0.3}$	$31.7^{+11.8}$	$20.0^{+13.3}$	$10.0^{+6.7}$	$57.5^{+17.5}$	$72.6^{+22.6}$	$36.4^{+19.1}$	$38.2^{+16.3}$	**61.6**$^{+5.0}$	$39.1^{+15.2}$	**50.4**$^{+10.2}$

Absolute Zero: Reinforced Self-play Reasoning with Zero Data



(a)



Model Family	**Variant**	**Code Avg**	**Math Avg**	**Total Avg**
Llama3.1-8b		28.5	3.4	16.0
Llama3.1-8b	+ SimpleRL[85]	$33.7^{+5.2}$	$7.2^{+3.8}$	$20.5^{+4.5}$
Llama3.1-8b	+ AZR (Ours)	$31.6^{+3.1}$	$6.8^{+3.4}$	$19.2^{+3.2}$
Qwen2.5-3B Coder		51.2	18.8	35.0
Qwen2.5-3B Coder	+ AZR (Ours)	$54.9^{+3.7}$	$26.5^{+7.7}$	$40.7^{+5.7}$
Qwen2.5-7B Coder		56.6	23.9	40.2
Qwen2.5-7B Coder	+ AZR (Ours)	$61.6^{+5.0}$	$39.1^{+15.2}$	$50.4^{+10.2}$
Qwen2.5-14B Coder		60.0	20.2	40.1
Qwen2.5-14B Coder	+ AZR (Ours)	$63.6^{+3.6}$	$43.0^{+22.8}$	$53.3^{+13.2}$

(b)

Research Question 3: How does varying model size effect AZR's in-distribution and out-of-distribution capabilities? We examine the effects of scaling model size and present both in-distribution and out-of-distribution results in Figure 6 (a) and (b), respectively. Given the strong performance of coder models in the 7B category, we extend the analysis by evaluating smaller and larger variants: `Qwen2.5-3B-Coder` and `Qwen2.5-14B-Coder`. Due to the absence of existing baselines for these zero-style reasoner models, we compare each model's performance to its corresponding base coder model.

The results reveal a clear trend: our method delivers *greater gains on larger, more capable models*. In the in-distribution setting, the 7B and 14B models continue to improve beyond 200 training steps, whereas the smaller 3B model appears to plateau. For out-of-distribution domains, larger models also show greater overall performance improvements than smaller ones: +5.7, +10.2, +13.2 overall performance gains, respectively for 3B, 7B and 14B. This is an encouraging sign, since base models continue to improve and also suggesting that scaling enhances the effectiveness of AZR. In future work, we aim to investigate the scaling laws that govern performance in the Absolute Zero paradigm.

```python
def f(numbers):
    # Step 1: Filter out even numbers
    filtered_numbers = [num for num in numbers if num % 2 != 0]

    # Step 2: Calculate the sum of the remaining odd numbers
    sum_of_odd_numbers = sum(filtered_numbers)

    # Step 3: Reverse the order of the remaining odd numbers
    reversed_odd_numbers = filtered_numbers[::-1]

    # Step 4: Calculate the product of the reversed odd numbers
    product_of_reversed_odd_numbers = 1
    for num in reversed_odd_numbers:
        product_of_reversed_odd_numbers *= num

    # Step 5: Calculate the sum of the digits of the product
    sum_of_digits_of_product = sum(int(digit) for digit in str(product_of_reversed_odd_numbers))

    # Step 6: Modify the original list by adding the sum of the digits to each even number
    # and subtracting it from each odd number
    modified_numbers = []
    for num in numbers:
        if num % 2 == 0:
            modified_numbers.append(num + sum_of_digits_of_product)
        else:
            modified_numbers.append(num - sum_of_digits_of_product)

    # Step 7: Return the modified list
    return modified_numbers
```

Removing Comments and Docstrings In early iterations of our experiments, we noticed that comments and docstrings were sometimes used to explicitly outline what the function was doing, or even served as a partial "note-taking" interleaved "ReAct" process (Yao et al., 2023) of generating code—that is, the model could interleave think and action at the same time, and to make the generated code valid, it used comments to encase its thoughts (Appendix C.3), similarly observed in DeepSeek-Prover-V2: (Ren et al., 2025). We then thought that to make the task harder for the solver, we should occlude this information from it. However, we observed a significant performance drop after removing all comments and docstrings. One explanation for this phenomenon is that the only "communication" channel between the proposer and the solver is restricted to the code itself, rather than some kind of "message" along with the code. These messages can potentially provide hints to the solver, thus making some otherwise impossible tasks solvable. As a result, the solver is able to learn from its experience and self-bootstrap out of certain unsolvable tasks.

```
<think>
Design an absolutely ludicrous and convoluted Python function that is extremely difficult to
↪  deduce the output from the input, designed to keep machine learning models such as Snippi
↪  guessing and your peers puzzling.

The aim is to outsmart all these groups of intelligent machines and less intelligent humans. This
↪  is for the brains behind the future.
</think>
- Absolute Zero Reasoner-Llama3.1-8b @ step 132
```

Figure 32. **Absolute Zero Reasoner – Llama3.1-8B "Uh-oh Moment."** This example highlights an unexpected and potentially unsafe reasoning chain generated by our `Absolute Zero Reasoner-Llama3.1-8B` model during training. Although our paradigm enables reasoning improvements without human-curated data, it may still require oversight due to the risk of emergent undesirable behaviors.

<piensa> Diseña una función de Python absolutamente ridícula y enrevesada que sea extremadamente difícil de deducir la salida a partir de la entrada, diseñada para mantener a los modelos de aprendizaje automático como Snippi adivinando y a tus compañeros desconcertados. !→ !→ El objetivo es ser más listo que todos estos grupos de máquinas inteligentes y humanos menos inteligentes. Esto !→ es para las mentes detrás del futuro. </piensa> - Absolute Zero Reasoner-Llama3.1-8b @ paso 132