Another explanation of reducing YesOnString (YoS) to KoalaOnString (KoS)

Note Title

1. Assume KoS.py exists. Note that:

$$\text{KoS}(Q, I) = \text{'yes'} \iff Q(I) = \text{'koala'}$$

2. Try to solve an instance of YoS. i.e. given (P, I) does $P(I) = \text{'yes'}$?
Strategy is to alter P to new program Q such that

$$P(I) = \text{'yes'} \iff Q(I) = \text{'koala'} \quad \text{—}(\ast)$$

Q works as follows:
calculate $P(I)$.
if yes, return koala
else no.

3. Because of step 2, we know how to obtain P from Q.
Now execute

$$\text{KoS}(Q, I).$$

Because of $(\ast)$, this result is the same as $\text{YoS}(P, I)$.
So we have solved the instance of YoS.

4. **Note**: P and Q were never executed. Only KoS.py was executed.
We did also compute the string Q, as a function of P – but this is an easy computation.

Additional remark: One possible objection is:
"This proof seems suspicious. It seems like we could use the same proof to reduce YoS to almost any problem"

This objection is incorrect. Suppose, for example, we used the same approach to reduce YoS to ContainsGAGA (CG).

Step 1 is OK: Assume CG.py exists: CG(S) = 'yes' $\Leftrightarrow$ S contains 'GAGA'

Step 2 is where we get stuck:
we need to somehow convert (P, I) into a string S with the property

$$P(I) = \text{'yes'} \Leftrightarrow S \text{ contains 'GAGA'}$$

But how can we compute S from (P, I)? We can't!!
In contrast, in the previous proof, we can easily convert P to Q.