General:

The log function calculates the natural logarithm of an input value x, where $1 \leq x < e^3$.
Our application is never required to calculate the logarithm of input values outside this range.
The method can be easily modified in order to support larger input values but not smaller input values.
This is due to the fact that the latter yield negative output, while the implementation is "purely unsigned".

Technical:

The method relies on a "mixture" of precalculated constants along with a Taylor series.
The Taylor series expansion for $log(1 + x)$ around 0 is given by $x - x^2/2 + x^3/3 - x^4/4 + ...$
It converges for every $x < 1$, but the speed of convergence increases as x gets closer to 0.
So in order to attain faster convergence, we first decompose the input into a product of powers of e and a residue which is close to1, and then use a Taylor series in order to calculate the logarithm of that residue.

For example:
$250 = e^{2^2} \times e^{2^0} \times e^{2^{-1}} \times 1.021692859 \Rightarrow log(250) = 2^2 + 2^0 + 2^{-1} + log(1 + 0.021692859)$

There are two factors to consider in our implementation:

1. The number of powers of e used for reducing the input before applying the Taylor series
2. The number of terms in the Taylor series used for calculating the logarithm of the residue

Any combination of these factors yields a different tradeoff between performance and accuracy.
We therefore auto-generate the code of this function, in a manner which minimizes the number of operations (and consequently the gas consumption), while ensuring an impeccable level of accuracy.

In file /project/auxiliary/AutoGenerate/ModelCalculator/constants.py, we have:

- PRECISION = 125, which determines that we compute in advance each constant c as the integer value of $c \times 2^{125}$
- LOG_MAX_HI_TERM_VAL = 3, which determines that the maximum input to the function must be smaller than e^3
- LOG_NUM_OF_HI_TERMS = 9, which determines that we compute in advance the value of $e^{3/2^n}$ for every $n \in [1,9]$

This ensures that the maximum possible residue is smaller than $e^{3/2^9}$.

We then compute in advance the 16 coefficients of a Taylor series which guarantees the maximum possible accuracy for the maximum possible residue (and subsequently for any other possible residue).

Any additional term would “contribute zero” to the series (regardless of the input residue), and the only loss of data is a consequent result of integer-division, which is inherent in the fixed-point system at hand.

Hence for the residue $1 + r$, we compute

$$log(1+r) = \sum_{n=1}^{16} r^n/n \times (-1)^{n-1} = \sum_{n=1}^{8} (2n/(2n-1) \times r^{2n-1} - r^{2n})/2n$$

We compute in advance and scale up to 125 bits:

- The value of $e^{3/2^n}$ for every $n \in [1,9]$
- The value of $2n$ for every $n \in [1,8]$
- The value of $2n/(2n-1)$ for every $n \in [1,8]$