Adaptive Understanding Score Model

# 1. Introduction

This model estimates a student's understanding of a topic, like Python basics, using a score out of 100.

After each quiz question, the score is updated based on the question's difficulty, the student’s answer,

how relevant the question is, and how far along the student is in their learning journey.

# 2. Predicting Success: Logistic Function

To estimate if the student is likely to answer a question correctly, we use:

predicted\_probability = 1 / (1 + exp(-(understanding\_score - question\_difficulty)))

If the student's score (e.g. 60) is higher than the question difficulty (e.g. 40), the probability is high.

If their score is lower (e.g. 30 vs. 70), it’s much lower.

# 3. Scoring Update Rule

After each answer, update the score using:

new\_score = current\_score + learning\_rate \* (correctness\_score - predicted\_probability)

correctness\_score is:

- 100 if the answer is fully correct

- 0 if wrong

- or anything in between for partial credit

# 4. Adjusting for Relevance

Only part of a question may be related to the topic. Add a relevance multiplier:

new\_score = current\_score + learning\_rate \* relevance\_factor \* (correctness\_score - predicted\_probability)

If relevance\_factor = 50 (half related), the score change is halved.

# 5. Dynamic Learning Rate

Early questions should change the score more than later ones. So we reduce the rate over time:

learning\_rate = initial\_rate / (1 + num\_questions\_answered / 3)

For example, if initial\_rate = 50 and 6 questions have been answered, learning\_rate = 25.

# 6. Full Update Equation

Final formula to compute the new score:

new\_score = current\_score + learning\_rate \* relevance\_factor \* (correctness\_score - predicted\_probability)

All values are out of 100 (scores, difficulties, relevance, etc.).

# 7. Python Quiz Example

Initial understanding\_score = 20

Question 1: difficulty = 70, relevance = 100

predicted = 1 / (1 + exp(-(20 - 70)/10)) ≈ 38%

Student gets it right (correctness = 100)

learning\_rate = 50

Update = 50 \* 1 \* (100 - 38)/100 = 31

New score = 51

Question 2: difficulty = 30, relevance = 100

predicted = 1 / (1 + exp(-(51 - 30)/10)) ≈ 55%

Answer is wrong (correctness = 0)

Update = 50 \* 1 \* (0 - 55)/100 = -28

New score = 23

Question 3: difficulty = 50, relevance = 50 (partially related)

correctness = 50 (partial answer), learning\_rate = 40

predicted = ~43%

Update = 40 \* 0.5 \* (50 - 43)/100 ≈ +1

New score = 24

# 8. Integration into an App

This system can be built into a Python learning app. Each topic tracks its own score. Questions are tagged

with difficulty and relevance. The app updates the score after every answer using the formula above.

It shows progress in real-time, helping learners know what they’ve mastered.

# 9. Summary

This approach blends:

- Logistic IRT models (predicting success based on difficulty and ability)

- Elo-style updates (adjust score based on outcome vs. expectation)

- Topic relevance weighting

- Dynamic learning rates

It is ideal for on-device learning apps: adaptive, fast, interpretable, and scalable.

All scores, difficulties, and updates are shown on a 0–100 scale to match typical UI displays.