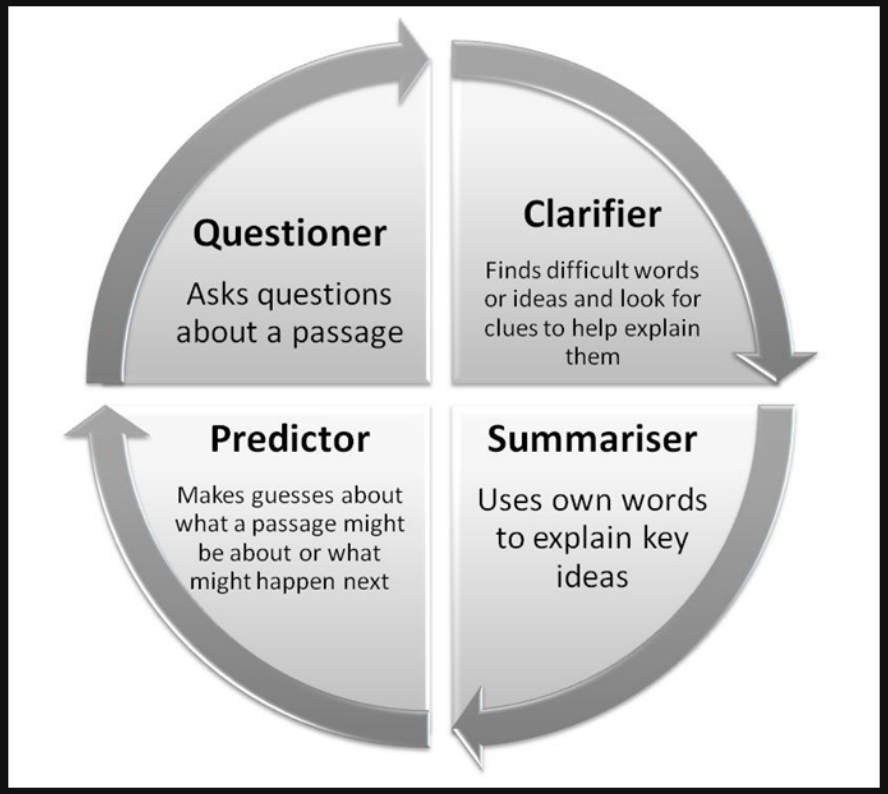
Reciprocal Teaching

Reciprocal Teaching

A dialogue between teachers and students in which participants take turns assuming the role of teacher.

The concept of *ako* describes a teaching and learning relationship, where the educator is also learning from the student and where educators' practices are informed by the latest research and are both deliberate and reflective.

Reciprocal Teaching



Questioning

Ask questions to check your understanding.

Metacognition

This awareness of one's own thought processes is called metacognition.

Metacognition is *thinking about thinking*.

Forms of metacognition

- Person knowledge (declarative knowledge) which is understanding one's own capabilities.
- Task knowledge (procedural knowledge) which is how one perceives the difficulty of a task.
- Strategic knowledge (conditional knowledge) which is one's own capability for using strategies to learn information.

Clarifying

The identification and clarification of unclear, difficult, or unfamiliar aspects of the material.

Chunking

A phenomenon whereby a person groups responses when performing a memory task.

Chunking

Presumably, individuals that exhibit the "chunking" process in their responses are forming clusters of responses based on the items' semantic relatedness or perceptual features.

The chunks are often meaningful to the participant.

Summarizing

Requires the reader to perform the task of discriminating between important and less-important information in the material.

It must then be organized into a coherent whole.

Predicting

Based on your background knowledge and your understanding of the topic, predict what might happen next.

Questioning

use metacognition

```haskell
maximum' :: (Ord a) => [a] -> a
maximum' [] = error "maximum of empty list"
maximum' [x] = x
maximum' (x:xs)
    | x > maxTail = x
    | otherwise = maxTail
    where maxTail = maximum' xs
```

Person knowledge: What do we know? (Limits of our capabilities.)

Task knowledge: What don't we know? (Difficulty of the task.)

Strategic knowledge: How can we figure this out? (Learning to learn.)

Clarifying

use chunking

```haskell
maximum' :: (Ord a) => [a] -> a
maximum' [] = error "maximum of empty list"
maximum' [x] = x
maximum' (x:xs)
    | x > maxTail = x
    | otherwise = maxTail
    where maxTail = maximum' xs
```

Can we group similar ideas together?

Summarizing

```haskell
maximum' :: (Ord a) => [a] -> a
maximum' [] = error "maximum of empty list"
maximum' [x] = x
maximum' (x:xs)
    | x > maxTail = x
    | otherwise = maxTail
    where maxTail = maximum' xs
```

What is important here? What is less important?

Predicting

```haskell
maximum' :: (Ord a) => [a] -> a
maximum' [] = error "maximum of empty list"
maximum' [x] = x
maximum' (x:xs)
    | x > maxTail = x
    | otherwise = maxTail
    where maxTail = maximum' xs
```

What is returned when we call maximum on []?

What is returned when we call maximum on [100]?

What is returned when we call maximum on [20, 50, 30, 45]?