What is the purpose of self-attention, and how does it facilitate capturing dependencies in sequences?

The role of self-attention is meaning aggregation from context. Which in turn enables us to create contextual embeddings.
It plays the same role as an LSTM does in a Elmov. LSTM (RNNs) are traditionally used in NLP to aggregate meaning.
There two drawbacks of LSTM which are both addressed by self-attention.

- Time taken. LSTM works sequentially word by word which increases the time taken, this becomes more significant as we increase layers.
- LSTM assumes that language is linear is structure.

Self-attention is able aggregate meaning for all words in a context at once. Which makes it time efficient.
Self-attention does not assume any language structure it learns the language structure. Hence it can capture longer dependencies better.

Why do transformers use positional encodings in addition to word embeddings? Explain how positional encodings are incorporated into the transformer architecture.

When use only self-attention we are converting a sentence into a bag of words. Which is a bad idea.
Consider the sentences "ram hits sam", "sam hits ram".

- Both the sentences have different meaning in one sam is hurt in another ram is hurt.
- But the model will be unable to distinguish the above two as the bag of words is the same for both the sentences.

Another way to say that is most (all) languages are not free word order. Free word order are the languages where the order of words does not matter. Hence we need to incorporated into our model some notion of order. This is done by using positional embeddings.

Position encodings are introduced by adding a notion of position into the word embedding. This is done by adding Position embedding to the global word embedding of a word. Position embedding any mathematical function which does repeat its values can be used.
The below encoding is used in the paper "Attention is all you need".
PE(pos,2i) = sin(pos/10000(^2i/dmodel))
PE(pos,2i+1) = cos(pos/10000^(2i/dmodel))