Okay first thing, you need to understand what the code is doing here. So here is an overview, there might be implementation details that you might not understand, but just be with me for the time being.

What the fuck is this code doing?

The function takes the following arguments:

- seq: The input sequence as a string.
- pad: A parameter (defaulting to 'dot') used for padding.

It defines the max_input_size, which is set to 75% of 1024, considering the maximum input size that GPT-2 can handle and taking into account BPE (Byte-Pair Encoding) splits.

Also note, here he is using Causal Language modeling, I dont remember if we studied this in MT, but its different from masked language modeling in the sense that given an input sequence, the job of LM is to predict the next word in the sequence while attending to tokens only on the left side, read more here if interested:
https://huggingface.co/docs/transformers/tasks/language_modeling

Q: Why the number 1024?

Because this is how long the input, GPT2 can take. Remember the input to a language model (LM) is (hidden dim * input seq length)

Q: What is BPE?

Encoding of an input string can take place at word level or character level, but still we will run into a vocabulary problem because in languages, there are many many words. BPE is a compression algorithm that does subword based tokenization pretty efficiently. You dont need to know the details of algorithms, though we have already studied this in MT. But you can read here, if you are interested:
https://towardsdatascience.com/byte-pair-encoding-subword-based-tokenization-algorithm-77828a70bee0

The input sequence is divided into smaller chunks (if it's too large) using the chunkstring function (again useless for you, how chucking is done, but just know its done). These chunks are expected to be a list of words.

The function initializes empty lists for words and word_surprisal, which will store the words and their associated surprisal values. A pad identifier (pad_id) is determined based on the pad parameter. It defaults to the ID of the period ('.').

The function then iterates (I hope you know what iterate means) through the sequence chunks. For each chunk (chunk is just a fancy word for group):

- It tokenizes the text into inputs using the tokenizer.
- These inputs are fed into the language model, and predictions are made. The predicted token IDs and probabilities are calculated.
- The function then calculates surprisal values for each token in the chunk.

After processing the entire input sequence, the function calculates the word probabilities based on the surprisal values. Lower surprisal values correspond to higher word probabilities .

There's a section that replaces certain character sequences in the words list, likely handling encoding issues (thats something you can skip).

The function returns the last value in the word_surprisal list. This value represents the surprisal of the last word in the input sequence. (why the last word, because thats what your goal is in this experiment, I suppose, to measure how good a LM is in predicting the next word, given previous sequences)

How to interpret the CSV file?

From what I understand, GPT2_s is the column you should be looking for, because it has surprisal values when the stimulus changes, leave the BPE_split column, its useless for you.

How to go on reporting the results in Academic Format

Majorly, people report parameters of the language model that they use, but in your case its standard GPT-2 architecture with no major variations, here is the config file of the language model that he used: https://huggingface.co/stefan-it/secret-gpt2/blob/main/config.json

What is a config file? Just a JSON format file that documents parameters of the language model. In your case, would be fine, if you just report these:

- Number of layers
- Size of input sequences
- Number of parameters
- Hidden dimension

Rather than giving technical details of LM itself, your focus will be on writing on how you use LM for calculating perplexity values, that I already mentioned in the overview of the code part. People usually just write one or two lines about LM details and share a hyperlink of the hugging face url where the model is hosted.

Here, I am sharing some papers that use LMs for psycholinguistic studies, rather than looking at what they are doing, see how they are writing descriptions about how they end up using GPT-2 for their experiment. How did I end up with this list:
https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=psycholinguistics+gpt-2&btnG=

https://arxiv.org/pdf/2009.03954

https://arxiv.org/pdf/2109.03926.pdf

https://aclanthology.org/2022.csrr-1.3.pdf

https://arxiv.org/pdf/2006.01912.pdf

Regardless, even after reading these papers, I would suggest write something (not hold yourself back) and get it proofread by Chrisoph (if it's technically sound), and Benedikt (to ensure that you are not missing out on technical detail)