Edge-Corner Cases:

- Repeated symptoms shouldn't affect probability
  - "Enlarged head and enlarged head and cough and cough or cough."
- Singular vs plural phenotypes
  - "Patient had a seizure and cough" vs "Patient having seizures"
- Gerund phrases
  - Cough vs Coughing
- Malformed sentence structure, in the case that the user is not native to english, or just gives input from raw medical notes, etc.
  - List of symptoms
  - Grammatically incorrect sentence structure (could mess up NLTK tagging)
- Tense-differences
  - Cough vs coughed
- Negations in noun phrases (currently not supported)
  - "Patient surprisingly has no cough"

Tradeoffs

- I thought about only showing results over some probability threshold (user-decided or hardcoded in backend), but decided to show all options for correctness testing in this initial version.
- Currently I show the probability range of how likely a condition is had based on the inputted symptoms to be transparent. Alternative could be to take the average of ranges and show that instead. It'd be cleaner but might give misleading info.
- Currently we sort on max probability but don't take into account the # of symptoms seen. # of symptoms seen should be included, but it also shouldn't be weighed higher than probability. For this I'd need a more sophisticated scoring system that takes both into account, and adds weight based on # symptoms seen.
- I decided to output my results directly into a django view as my frontend. Instead, because my backend is just handling POST requests, I could have served a json response to some other frontend framework like angular, react, jquery and made everything a bit more modular with a nicer UI. However, since the

focus of this project seemed to be more logic-heavy, I used simple HTML styling to style for my frontend view.

- I currently store the data as json if it's not already preprocessed. An alternative to this would just be to cache in memory using python's LRU cache
- I chose to extract medical entities using nouns and noun phrases with NLTK because, by using grammatical structure we can catch words that are likely to be symptoms (and then only considering words/phrases that we have data for). Other options would be to:
    - Use existing medical extraction models (google NLP healthcare API, some on github, etc)
    - Create my own model (probably not reasonable given time scope)

Possible add-ons

- For each result in the list, we could link the result to its description as given in OrphaData
- Handle negations (currently if you input "not cough" it would list cough as a symptom)
- Currently when a symptom is "excluded" from a disease we just remove that disease option from the list. However, the input could have many other symptoms that would be high probability, it's possible that the excluded symptom was an incorrect addition. We should present the option if the remaining symptoms probability is above some threshold, but highlight (maybe in red) that an excluded symptom exists for this disease option.