PIC 16 Final Exam Part I - Track B - Section 2

Here in Section 2, you will write the code for your skill's cloud-based service independently of your responses in Section 1. You must assume the following interaction model for Word Helper.

Intent Schema:

```
{
  "intents": [
    {
      "intent": "AMAZON.HelpIntent"
    },
    {
      "intent": "RandomWordIntent"
    },
    {
      "slots": [
        {
          "name": "day",
          "type": "AMAZON.DATE"
        }
      ],
      "intent": "WordOfTheDayIntent"
    },
    {
      "slots": [
        {
          "name": "length",
          "type": "AMAZON.NUMBER"
        },
        {
          "name": "letter",
          "type": "LETTER"
        }
      ],
      "intent": "WordHelpIntent"
    },
    {
      "slots": [
        {
          "name": "word",
          "type": "WORD"
        }
      ],
      "intent": "SynonymIntent"
    }
  ]
}
```

Custom Slot Types:

LETTER:
a b c d e f g h i j k l m n o p q r s t u v w x y z

WORDS:

some	but	assume
example	slots	other
words	may	values

Sample Utterances:

```
RandomWordIntent a random word

WordOfTheDayIntent the word of the day

WordOfTheDayIntent the word of the day for {day}

WordHelpIntent a {length} letter word

WordHelpIntent a word that starts with {letter}

WordHelpIntent a {length} letter word that starts with {letter}

SynonymIntent a synonym for {word}
```

In a single file `Part IB2.py`, write a function `handler(request, context)` to handle all the types of `requests` Alexa might send your cloud-based service based on the interaction model above and the following commands. When you're done, submit `Part IB2.py` to CCLE → Final Exam → Part IB2.

Alexa, start Word Helper (`0.json`)
Alexa should speak a greeting, like "Welcome to Word Helper! Please describe the word you're looking for or ask for help."

Alexa, ask Word Helper for help (`1.json`)
Alexa should speak a help prompt like the following:
I can tell you just the word you're looking for. You can ask for a random word by saying "tell me a random word". You can ask for the word of the day like "give me the word of the day". You can tell me that you'd like a word of a certain length and/or a word that starts with a certain letter, like "give me a five letter word that begins with 'l'". Finally, you can ask for a synonym for a word, like "what is a synonym for 'yell'?". Which would you like?

Alexa, ask Word Helper for a random word (`2.json`)
Alexa should speak a word chosen at random from the list returned by `nltk.corpus.words.words`.

Alexa, tell Word Helper to give me the word of the day (`3.json`)
Alexa, tell Word Helper to give me the word of the day for January 12, 2017 (`4.json`)
Alexa should speak a "word of the day" for the provided day. The word of the day should *not* be chosen at random; the word must be determined completely based on the value provided in `date`, that is, asking for the word of the day for a given date multiple times should return the same word each time. However, the word chosen for any given day may be arbitrary, that is, the rule that maps a date to a word is up to you. If no date is spoken, the date should be assumed to be the date of the request. Words should not be repeated for different dates frequently or regularly, but incidental/occasional repeats are OK.

Alexa, ask Word Helper for a three letter word (`5.json`)
Alexa, ask Word Helper for a word that starts with "A" (`6.json`)
Alexa, ask Word Helper for a five letter word that starts with "l" (`7.json`)
Alexa, ask Word Helper for a one hundred sixty-five letter word that starts with "x" (`8.json`)
Alexa should reply with a word from `nltk.corpus.words.words` that meets provided criteria if one exists. If no such word exists, she should reply "Sorry, but I don't know of such a word." When asked multiple times, she should suggest different words that meet the criteria (at random) if multiple words exist. For full credit, users can even ask for words that start with another word (not just a letter).

Alexa, ask Word Helper what is a synonym for "yell" (`9.json`)
Alexa, ask Word Helper what is a synonym for "nosynset" (`10.json`)
Alexa should reply with a synonym for the provided word if the provided word has at least one synset in `nltk.corpus.wordnet`. If the provided word has no synsets, she should reply "Sorry, but I don't know any synonyms for that word." When asked multiple times, she should suggest different synonyms (at random) if different synonyms are available.

For maximum credit, your skill should *always* include a definition from `nltk.corpus.wordnet` along with the word. Therefore, words returned by your skill must not only be in the list returned by `nltk.corpus.words.words` but must also have a WordNet synset.

All WordNet documentation hosted by `nltk.org` is allowed (e.g. `http://www.nltk.org/howto/wordnet.html`)

Additional Specifications:

- Just as if you were writing an AWS Lambda function, your `handler` function should expect a `request` Python dictionary as the first argument.
- As discussed, most (if not all) of the information passed into a skill's AWS Lambda function via the second argument duplicates information contained in the first argument. For the sake of this exam, your `handler` function should accept a second argument (`context`), but *do not use it*.
- You can test your skill using the JSON requests provided at: `https://tinyurl.com/PIC16F17B`. Load a JSON file as a Python dictionary `request`, pass this dictionary into your function as the first argument (`None` as the second), and verify that your function returns the intended response.
- You can check that your response meets Amazon specifications using the function `alexa` available in the provided `alexa.py`. This function is designed to emulate the Developer Portal "Service Simulator", but it is even stricter at enforcing the standard response format. It accepts a dictionary `response` and returns the `outputSpeech text` *if* the response is valid. However, if `response` has any unrecognized attributes (keys), has invalid values or value types, or is missing any required fields, the function will return only `"The response is invalid."`. You must ensure that your skill's response meets these strict specifications (see "Alexa Skills Kit Custom Skills Request and Response JSON Reference"), as we will grade your solution using the `alexa` function.
- Assume that AWS Lambda's Python distribution has `nltk` installed. You should make the appropriate `import`s at the top of your `.py` file.
- Your skill will be tested with requests similar to those provided, however the values of slots may be different. For instance, a slot's value may be different, or its `value` key may be absent entirely.
- A WordNet synset represents a set of synonyms with a common meaning. A synset has lemmas, which are specific senses of a word. For the purpose of this assignment, a "synonym" is any lemma of any synset corresponding with a given word.
- To ensure maximum credit, you should write (and use, wherever possible) the following functions:
    - `pick_random(iterable)` – returns a random element of `iterable`. For full credit, each *unique* element of `iterable` should have an equal chance of being selected; e.g. if an element appears more than once it should not have a greater chance of being selected.
    - `word_of_day(datestring)` – returns the "word of the day" corresponding with `datestring` (date represented as a `str`). Again, it must return the same word every time the same `datestring` is provided and unique `datestring`s should almost always produce unique words. Otherwise, you can map a given `datestring` to a word however you wish.
    - `definition(word)` – returns a definition corresponding with the given `word`. I understand that words can have many definitions; your skill does not need to be smart about choosing among the definitions for a given word.
    - `random_word_with_definition(words)` – chooses a random word from iterable `words` for which a definition can be found, and returns the word *and* the definition in a single string like "cry: a loud utterance of emotion (especially when inarticulate)".
    - `wordsBy(condition)` – returns a list of all words that satisfy the given `condition`. `condition` is a *function*. `condition(word)` returns `True` if `word` satisfies the condition and `False` otherwise. If you have trouble understanding this requirement or if you cannot see how it would be used, for slightly less credit you can have three (highly redundant) functions:
        - `wordsByLength(n)` – returns a list of all words of length `n`
        - `wordsByStart(l)` – returns a list of all words that begin with `l`
        - `wordsByLengthStart(n, l)` – returns a list of words of length `n` that start with `l`
    - `synonyms(word)` – returns a list of all synonyms (as defined above) for a given word.

  Of course, you are allowed to write and use additional functions, if you wish.

Suggestions/Hints: see `hash`, `random.seed`, `random.randint`, `time.time`