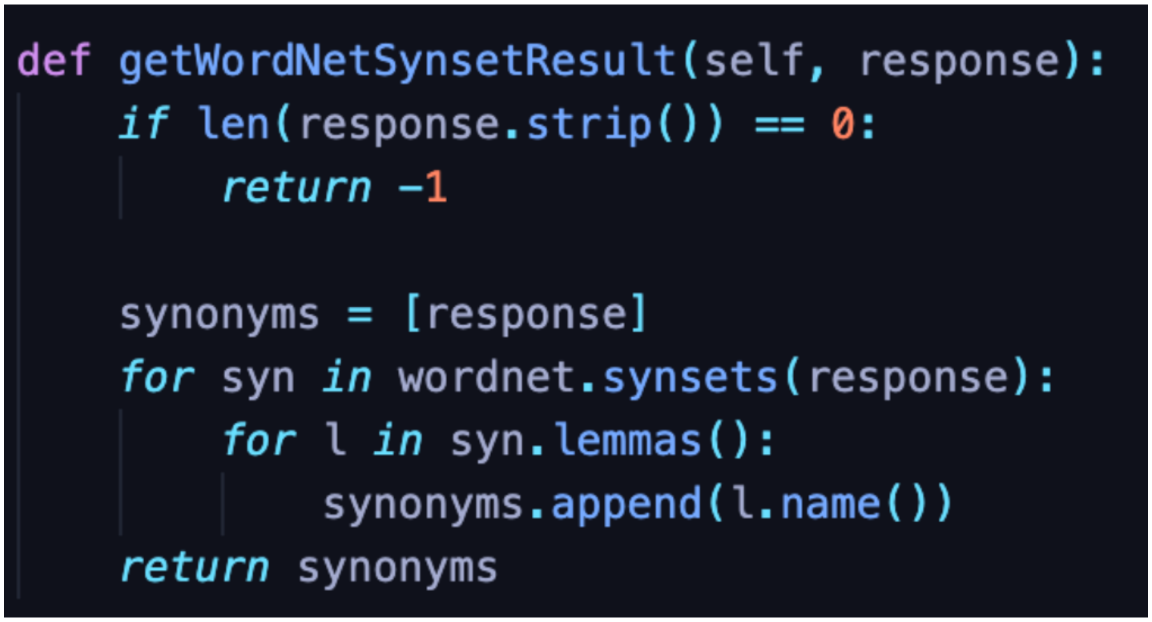
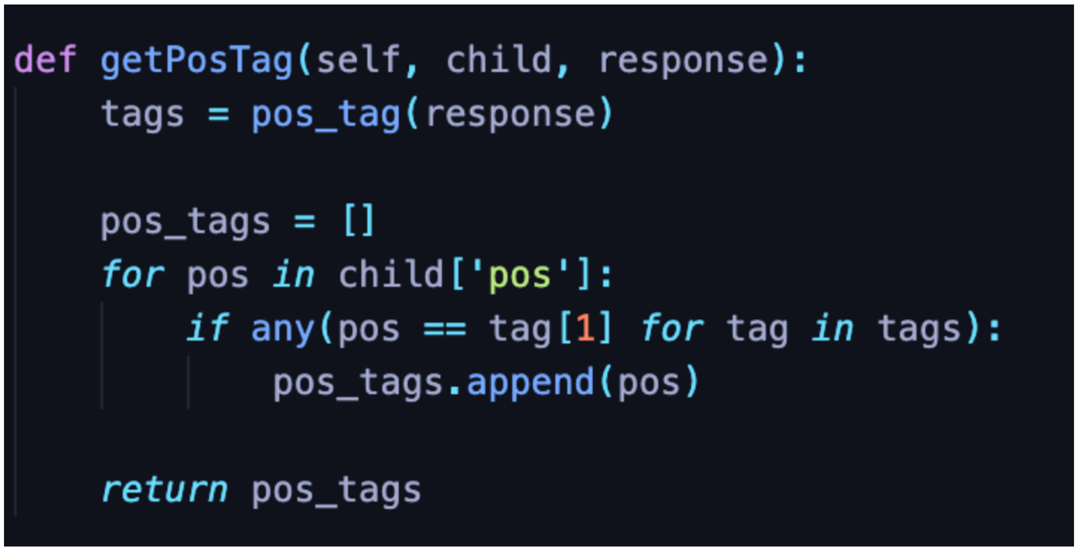
API Documentation



```python
def getWordNetSynsetResult(self, response):
    if len(response.strip()) == 0:
        return -1

    synonyms = [response]
    for syn in wordnet.synsets(response):
        for l in syn.lemmas():
            synonyms.append(l.name())
    return synonyms
```

- This API returns the list of synonyms from the Wordnet list based on the input provided by the user.
- The API method requires the initialization of the object of the Bot class to properly access this method.
- The method requires one string argument for the method.
    - If the argument is not provided or its length is equal to zero, an integer response of -1 will be provided.
    - If the correct argument is provided(e.g., exhausted), the response will return the list of the synonyms of the provided string argument.



```python
def getPosTag(self, child, response):
    tags = pos_tag(response)

    pos_tags = []
    for pos in child['pos']:
        if any(pos == tag[1] for tag in tags):
            pos_tags.append(pos)

    return pos_tags
```

❖ This API returns the list of predicted parts of speech used in the user response.
❖ The API method requires the initialization of the object of the Bot class to properly access this method.
❖ The method requires two arguments for the method to properly work:
  - “Child” argument is an Object type field. It defines what POS tags the method should look at in the “response” field provided.

    Example of the argument:

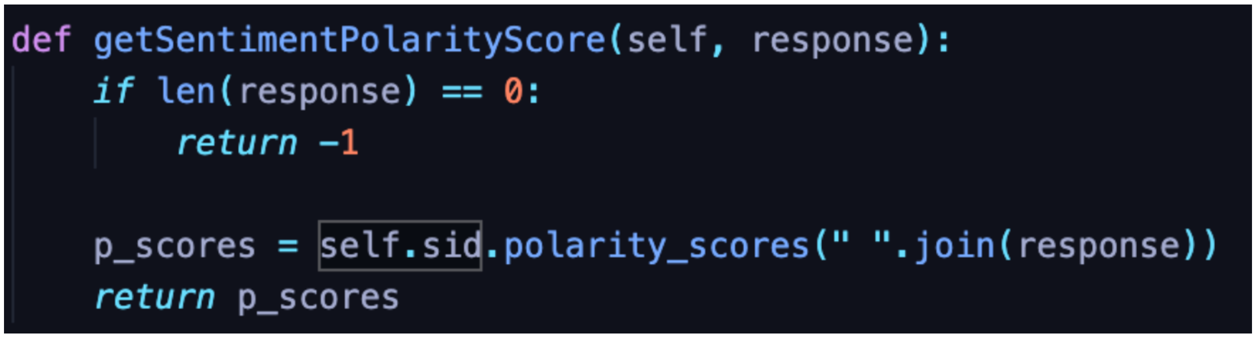
    { “pos”: [‘VB’, ‘WP’] }

    In this case, the program will look for verbs and Wh- pronouns.
  - “Response” argument is a list-based field that should be provided to the method.

    Example of the argument:

    ["what", "should", "i", "do", "?"]

    The method expects that each word of the sentence is split at its own index in the list.



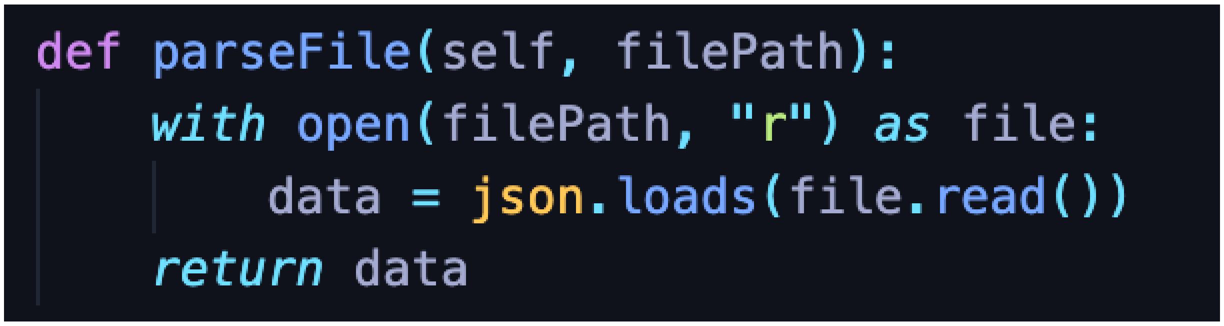
```python
def getSentimentPolarityScore(self, response):
    if len(response) == 0:
        return -1

    p_scores = self.sid.polarity_scores(" ".join(response))
    return p_scores
```

- ❖ This API returns the sentimental analysis of the user’s input. Determines if the response is positive or negative.
- ❖ The API method requires the initialization of the object of the Bot class to properly access this method.
- ❖ The method requires one argument for the method to properly work:
    - “Response” is a list-based argument that contains string values. The string values are adjectives.

Example of the argument:

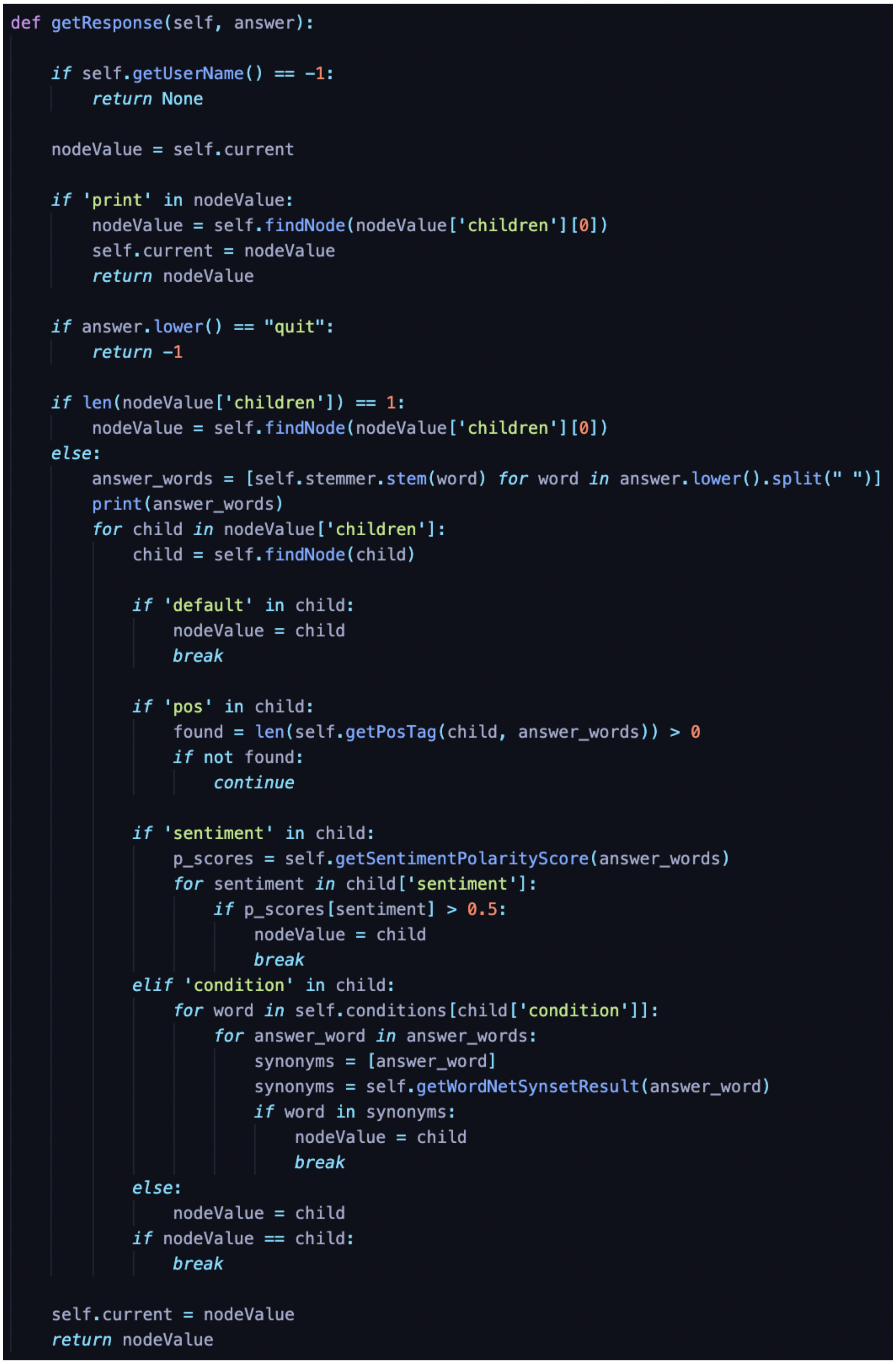
["disinterested", "inactive"]

In the example, we provided 2 strings that represent the state of the person. Based on these inputs, the program, using the polarity scores will determine what type of response it is. Is it neutral, negative, or positive?



```
def parseFile(self, filePath):
    with open(filePath, "r") as file:
        data = json.loads(file.read())
    return data
```

- ❖ Method to open an existing file with questions and possible responses, and return the Object of this file.
- ❖ This API method requires the initialization of the object of the FileReader class to properly access this method.
- ❖ The method requires one argument to properly execute.
    - ➢ “filePath” should be a string value and the user needs to provide a path to the file that has an extension/format of “.json”



```python
def getResponse(self, answer):

    if self.getUserName() == -1:
        return None

    nodeValue = self.current

    if 'print' in nodeValue:
        nodeValue = self.findNode(nodeValue['children'][0])
        self.current = nodeValue
        return nodeValue

    if answer.lower() == "quit":
        return -1

    if len(nodeValue['children']) == 1:
        nodeValue = self.findNode(nodeValue['children'][0])
    else:
        answer_words = [self.stemmer.stem(word) for word in answer.lower().split(" ")]
        print(answer_words)
        for child in nodeValue['children']:
            child = self.findNode(child)

            if 'default' in child:
                nodeValue = child
                break

            if 'pos' in child:
                found = len(self.getPosTag(child, answer_words)) > 0
                if not found:
                    continue

            if 'sentiment' in child:
                p_scores = self.getSentimentPolarityScore(answer_words)
                for sentiment in child['sentiment']:
                    if p_scores[sentiment] > 0.5:
                        nodeValue = child
                        break
            elif 'condition' in child:
                for word in self.conditions[child['condition']]:
                    for answer_word in answer_words:
                        synonyms = [answer_word]
                        synonyms = self.getWordNetSynsetResult(answer_word)
                        if word in synonyms:
                            nodeValue = child
                            break
            else:
                nodeValue = child
            if nodeValue == child:
                break

    self.current = nodeValue
    return nodeValue
```

- The method looks over the node of the responses based on the user’s input.
    - The user input is accepted as an argument of an “answer” field. This argument is a string-based response that is further analyzed in the method.
        - Example:

          “I am feeling good”
- This API method requires the initialization of the object of the Bot class to properly access this method.
- If the main node has subnodes (Ex: The user answered “yes” to the question instead of “no”), the method will look for the child nodes to find correct answers.
- On the other hand, if the node doesn’t have subnodes, it will proceed to the next node based on the user’s response.
- The method also uses nltk and PorterStemmer libraries to detect the part of speed, synonym recognition, and sentiment analysis.
- The method also validates if the username was provided and if the user was prompted to quit the program.
- The method returns the object of the node.