

```
1 import re
2
3 class Data:
4
5     # list of possible type of questions ~ truncated for flexibility [non-exhaustive]
6     possibleList = [{"capital", "distanc", "weather", "movi", "forecast", "cit", "length", "climat", "humidit", "director", "actor"},
7 ], # direct-answer question
8     [{"exercis", "diet", "cook", "workout", "routin", "gym", "activit", "nutri", "wellness", "recipi", "fitnes", "yoga", "meditat", "stretch", "cardio", "strength", "vitamin", "calori", "symptom"}, # health-related questions
9     [{"calendar", "remind", "task", "schedul", "event", "deadlin", "project", "checklist", "alert", "notif", "organ", "priorit", "goal", "plann", "timelin", "focus", "track", "habit", "workflow"}], # productivity questions
10    [{"scor", "gam", "jok", "song", "challeng", "puzzl", "music", "lyric", "match", "adventur", "humor", "quiz", "fun", "comed", "story", "celebr", "sport", "trend"}], # entertainment questions
11    [{"plu", "minus", "multipl", "divid", "formula", "concept", "ratio", "algebra", "geometr", "calculus", "integrat", "deriv", "vector", "probabil", "statist", "measur", "equation", "matrix", "quantit"}], # mathematical questions
12    [{"pric", "mean", "fact", "happen", "latest", "explain", "differenc", "orig", "reason", "impact", "histor", "overview", "background"}], # knowledge-building question
13    [{"best", "advic", "help", "tip", "plan", "stuck", "assist", "recommend", "suggest", "guid", "strategy", "solve", "improv", "overcom", "choic", "option"}] # advice-seeking questions
14
15     # list of possible types of questions ~ original unmodified [non-exhaustive]
16     dictionaryList = [{"capital", "distance", "weather", "movie", "forecast", "city", "length", "climate", "humidity", "director", "actor"},
17     [{"exercise", "diet", "cook", "workout", "routine", "gym", "activity", "nutri", "wellness", "recipe", "fitness", "yoga", "meditate", "stretch", "cardio", "strength", "vitamin", "calories", "symptom"},
18     [{"calendar", "remind", "task", "schedule", "event", "deadline", "project", "checklist", "alert", "notif", "organ", "prioritize", "goal", "plann", "timeline", "focus", "track", "habit", "workflow"},
19     [{"score", "game", "joke", "song", "challenge", "puzzle", "music", "lyric", "match", "adventure", "humor", "quiz", "fun", "comedian", "story", "celebrate", "sport", "trend"},
20     [{"plu", "minus", "multiple", "divide", "formula", "concept", "ratio", "algebra", "geometry", "calculus", "integrate", "derivative", "vector", "probability", "statistic", "measure", "equation", "matrix", "quantity"},
21     [{"price", "mean", "fact", "happen", "latest", "explain", "difference", "origin", "reason", "impact", "history", "overview", "background"},
22     [{"best", "advice", "help", "tip", "plan", "stuck", "assist", "recommend", "suggest", "guide", "strategy", "solve", "improve", "overcome", "choice", "option"}]
23
24     # list of possible cities that the user might reference [non-exhaustive]
25     specificPlaceList = ["Los Angeles", "Chicago", "San Francisco", "Miami", "Austin", "Las Vegas", "Paris", "London", "Tokyo", "Sydney", "Rome", "Barcelona",
26     "Berlin", "Dubai", "Toronto", "Seoul", "Bangkok", "Mexico City", "Riverside", "Cape Town", "California", "Florida", "Texas", "New York", "Nevada", "Hawaii", "Colorado", "Alaska", "Arizona",
27     "Utah", "Illinois", "Michigan", "Washington", "Georgia", "North Carolina", "Tennessee", "South Carolina", "Oregon", "New Jersey", "Virginia", "United States", "Canada", "United Kingdom",
28     "France", "Italy", "Spain", "Mexico", "Germany", "Australia", "Brazil", "Japan", "India", "South Korea", "Thailand", "South Africa", "China", "Russia", "Egypt", "Argentina", "New Zealand", "Pakistan"]
29
30
31     # list of possible pop-culture references [non-exhaustive]
32     specificPopCultureList = ["The Avengers", "Star Wars", "The Matrix", "Harry Potter", "Jurassic Park", "Titanic", "The Godfather", "Pulp Fiction", "Back to the Future", "The Lion King", # movies
33     "Apple", "Nike", "Tesla", "Coca-Cola", "McDonald's", "Amazon", "Google", "Adidas", "Disney", "Microsoft", # brands
34     "Friends", "Game of Thrones", "The Office", "Stranger Things", "Breaking Bad", "The Simpsons", "The Mandalorian", "The Crown", "The Walking Dead", "Westworld", # tv shows
35     "The Beatles", "Beyonce", "Kanye West", "Taylor Swift", "Elvis Presley", "Michael Jackson", "Ariana Grande", "Drake", "Lady Gaga", "Eminem", # artists
36     "Super Mario Bros.", "Minecraft", "Fortnite", "The Legend of Zelda", "Call of Duty", "Grand Theft Auto", "Pokémon", "League of Legends", "FIFA", "The Witcher", # video games
37     "Rolls-Royce", "Ferrari", "Lamborghini", "Porsche", "Maserati", "Bentley", "Aston Martin", "Bugatti", "McLaren", "Mercedes-Benz", "Lexus", "BMW", "Audi"] # automotive brands
38
39
40
41     # word that is autocorrected
42     correctedWord = []
43
44     # method that removes accidental duplicates found by regular expression
45     def __remove_duplicate(self, list):
46         acclist = []
47         for i in list:
48             if i not in acclist:
49                 acclist.append(i)
50         return acclist
51
52     # uses Python Regular Expressions to derive key data in a structural format, replicating a basic version of Natural Language Processing
53     # "QT" = Question Type && "I" = Identifier
54     def parsedData(self, userInput, original_input):
55         question_type = self.__remove_duplicate(re.findall(r"(what|who|why|where|when|how|will|can|play|lets|let|should|is|tell|give|if)", userInput))
56         identifiers = self.__remove_duplicate(re.findall(r"(capital|best|cit|length|climat|humidit|director|actor|task|schedul|event|deadlin|project|checklist|alert|notif|organ|advic|stuck|help|tip|distanc|plan|weather|forecast|latest|r|happen|movi|exercis|song|diet|workout|explain|differenc|routin|gym|activit|nutri|wellness|recipi|fitnes|calendar|remind|cook|scor|pric|mean|plu|ratio|minus|multipl|divid|r|jok|gam|fact|formula|concept|algebra|geometr|challeng|puzzl|music|lyric|match|adventur|humor|yoga|meditat|stretch|cardio|strength|vitamin|calori|priorit|goal|plann|timelin|focus|track|habit|workflow|quiz|fun|comedi|story|celebr|sport|trend|calculus|integrat|deriv|vector|probabil|statist|measur|equation|symptom"
```

```

61 |matrix|quantit|orig|reason|impact|histor|overview|background|assist|recommend|suggest|guid|strategy|solv|improv|overcom|choic|
   |option)", userInput))
62
63     if self.autoCorrect(userInput, self.possibleList) is not None:
64         identifiers.extend(self.autoCorrect(userInput, self.possibleList))
65         identifiers = self.__remove_duplicate(identifiers)
66         result = {"QT": question_type, "I": identifiers}
67         self.correctedWord.extend(identifiers)
68         return result
69
70     # takes the original sentence inputted by the user and then removes suffixes; local change not global
71     def stemWord(self, userInput):
72         return re.sub(r'\b(?:!(is\b))(e|es|ing|ed|s|se|ication|ization|isation|ized|ised|ied|ous|y|ies|tion|ent|ents|er|ers|ic|
   |ation|ating|ize|ian|ate|ative|atives|ity|ics|in|inate|ance|)\b', '', userInput)
73
74     # finds any key data similar to the list above that might be misspelled to reinterpret input ~ similarity >= 70%
75     def autoCorrect(self, userInput, c_list):
76         inputList = userInput.split()
77         count = 0
78         result = []
79         for r in range(len(c_list)):
80             for c in range(len(c_list[r])):
81                 for n in inputList:
82                     iter = min(len(n), len(c_list[r][c]))
83                     for i in range(iter):
84                         if list(n).__getitem__(i) == list(c_list[r][c]).__getitem__(i):
85                             count += 1
86                         elif (i + 1) < len(c_list[r][c]):
87                             if list(n).__getitem__(i) == list(c_list[r][c]).__getitem__(i + 1):
88                                 count += 1
89                             if (count / max(len(n), len(c_list[r][c]))) * 100 >= 70:
90                                 result.append(c_list[r][c])
91                     count = 0
92         return result if result else None
93
94     # prints the content by returning them
95     def printContent(self, items):
96         if len(items) == 1:
97             return items[0] # Return the single item as a string
98         else:
99             return ", ".join(items)

```