```python
from data_class import Data
from loadingModule_2 import loadingAnimation
import sys
import string
import time

# method that returns the question-type like 'what', 'when', where' etc...
def printAllQuestionType(processed_data):
    if len(processed_data.get("QT")) == 1:
        return "'" + processed_data.get("QT").__getitem__(0) + "'"
    else:
        result = "mix of "
        try:
            for c in range(len(processed_data.get("QT")) - 1):
                result += "'" + processed_data.get("QT").__getitem__(c) + "', "
            result += "and '" + processed_data.get("QT").__getitem__(len(processed_data.get("QT")) - 1) + "'"
        except IndexError:
            print("\nI had an issue processing your query. Please re-run the program and rephrase your sentence.")
            sys.exit()
    return result

# method that returns any possible places that the user mentioned in their original statement ~ improves specific contextualization
def printPossibleMentions(saved_input, list):
  matches = [place for place in list if place.lower() in saved_input.lower()]

  if len(matches) == 1:
      return matches[0]
  elif matches:
      return ", ".join(matches)
  else:
      return ""

# method that checks if the given word is in the possibleList matrix to ensure that it can proceed with "what is"
def validateItemAbsent(word, data_instance):
    for q in range(len(data_instance.possibleList)):
        for w in range(len(data_instance.possibleList[q])):
            for e in word:
                if e == data_instance.possibleList[q][w]:
                    return False
    return True

# introductory instructions and terms/conditions
print("QUESTION ANALYZER: \n1.) Enter a question that you might ask a Google Assistance, Alexa, Siri, Cortana, etc..."
    "\n2.) The system will process your response.\n3.) It will give you it's understanding of the question by categorizing it using basic Natural Language Processing (NLP) algorithm.\n")
print("This is a first step taken to complete a part of 'robotics data interpretation' and will be built upon modularly.\n")
print("This model is still in BETA; some questions might not be recognizable by the system. More updates will be rolling out soon. [Version: 2.0]\n\n")
user_input = input("Ask me anything: ")
data_instance = Data()
rowCategory = None

# timer started - to be used to calculate how long this program was used
start_time = time.time()

# loop helps the user continue asking more questions for system categorization
while user_input != "stop":
    saved_input = user_input
    user_input = user_input.lower()
    processed_data = data_instance.parsedData(data_instance.stemWord(user_input), saved_input)

    # ensures there are no error, else, redirect the issue
    if processed_data.get("I") != None:
        loadingAnimation()
        for row in range(len(data_instance.possibleList)):
            for column in range(len(data_instance.possibleList[row])):
                for a in processed_data.get("I"):
                    if data_instance.possibleList[row][column] == a:
                        rowCategory = str(row)
        if rowCategory == None and not saved_input.__contains__("what if") and saved_input.split()[0] != "if" and not saved_input.__contains__("what is"):
            user_input = input("\nI wasn't able to understand your question. I can comprehend the question type but I couldn't find any identifiers that can help process this query.\nTry again or type 'stop': ")
        else:
            print(f"\n\n\nI understand that you are trying to ask a question that starts with a {printAllQuestionType(processed_data)} and I am supposed to give a(n) ")
            place_mentions = printPossibleMentions(saved_input, data_instance.specificPlaceList)
            pop_culture_mentions = printPossibleMentions(saved_input, data_instance.specificPopCultureList)

            # list of possible type of questions extracted from user input [non-exhaustive]
            if saved_input.__contains__("what if") or saved_input.split()[0] == "if":
                if pop_culture_mentions != "":
                    print("thoughtful, speculative answer based on logical reasoning, established facts, and potential scenarios of " + str(pop_culture_mentions) + ".")
                elif place_mentions != "" and pop_culture_mentions == "":
                    print("thoughtful, speculative answer based on logical reasoning, established facts, and potential scenarios of " + str(place_mentions) + ".")
                else:
                    print("thoughtful, speculative answer based on logical reasoning, established facts, and potential
scenarios.")
            elif (saved_input.__contains__("what is") and (validateItemAbsent(data_instance.correctedWord, data_instance) or len(saved_input.split()) == 3)) or saved_input.__contains__("definition"):
                  clean_text = saved_input.translate(str.maketrans('', '', string.punctuation))
                  if clean_text.split().__getitem__(len(clean_text.split()) - 1).isupper():
                        clean_text = clean_text.upper()
                  print("definition and explanation for " + clean_text.split().__getitem__(len(clean_text.split()) - 1)  + ".")
            elif rowCategory == "0":
                  if place_mentions != "" and pop_culture_mentions != "":
                     print("clear and succinct response providing general insights into situations, activities, individuals, or weather conditions and forecasts for locations like " + str(place_mentions) + " and Iconic References like " + str(pop_culture_mentions) + ".")
                  elif place_mentions != "":
                      print("clear and succinct response providing general insights into situations, activities, individuals, or weather conditions and forecasts for locations like " + str(place_mentions) + ".")
                  elif pop_culture_mentions != "":
                      print("clear and succinct response providing general insights into situations, activities, individuals, or weather conditions and forecasts for Iconic References like " + str(pop_culture_mentions) + ".")
                  else:
                     print("clear and succinct response providing general insights into locations, situations, activities, individuals, or weather conditions and forecasts, grounded in established facts.")
            elif rowCategory == "1":
                     print("insightful, evidence-based response addressing personal or health-related inquiries about diet, lifestyle, and well-being.")
                     if place_mentions != "" or pop_culture_mentions != "":
                        print("You mentioned places and/or Iconic References like, yet I don't see any relevancy of these mentions to the personal or health-related inquires you asked.")
            elif rowCategory == "2":
                  if place_mentions != "" and pop_culture_mentions != "":
                     print("productivity-focused response designed to enhance health, work, or performance efficiency, based on logical reasoning, proven strategies, and relevancy to the following places: " + str(place_mentions) + " and Iconic References: " +
                           str(pop_culture_mentions))
                  elif place_mentions != "":
                      print("productivity-focused response designed to enhance health, work, or performance efficiency, based on logical reasoning, proven strategies, and relevancy to the following places: " + str(place_mentions) + ".")
                  elif pop_culture_mentions != "":
                      print("productivity-focused response designed to enhance health, work, or performance efficiency, based on logical reasoning, proven strategies, and relevancy to the following Iconic References: " + str(pop_culture_mentions) + ".")
                  else:
                     print("productivity-focused response designed to enhance health, work, or performance efficiency, based on logical reasoning and proven strategies.")
            elif rowCategory == "3":
                  if place_mentions != "" and pop_culture_mentions != "":
                     print("engaging, mood-enhancing response tailored to improving your leisure and overall well-being, based on entertainment preferences and refined for the following places: " + str(place_mentions) + " and Iconic References: " +
                           str(pop_culture_mentions))
                  elif place_mentions != "":
                      print("engaging, mood-enhancing response tailored to improving your leisure and overall well-being, based on entertainment preferences and refined for the following places: " + str(place_mentions) + ".")
                  elif pop_culture_mentions != "":
                      print("engaging, mood-enhancing response tailored to improving your leisure and overall well-being, based on entertainment preferences and refined for the following Iconic References: " + str(pop_culture_mentions) + ".")
                  else:
                     print("engaging, mood-enhancing response tailored to improving your leisure and overall well-being, based on entertainment preferences.")
            elif rowCategory == "4":
                  print("concise or detailed mathematical response addressing a concept or calculation, based on established principles and logical reasoning.")
                  if place_mentions != "" or pop_culture_mentions != "":
                      print("You mentioned places and/or Iconic References, although I am not sure the relevancy of this in the context of mathematical calculation, therefore, I am disregarding them.")
            elif rowCategory == "5":
                  if place_mentions != "" and pop_culture_mentions != "":
                     print("informative response aimed at enhancing your understanding of key details related to current or past events and information for places like " + str(place_mentions) + " and Iconic References like " + str(pop_culture_mentions) + ".")
                  elif place_mentions != "":
                     print("informative response aimed at enhancing your understanding of key details related to current or past events and information for places like " + str(place_mentions) + ".")
                  elif pop_culture_mentions != "":
                      print("informative response aimed at enhancing your understanding of key details related to current or past events and information for Iconic References like " + str(pop_culture_mentions) + ".")
                  else:
                     print("informative response aimed at enhancing your understanding of key details related to current or past events and information, grounded in factual accuracy and context.")
            elif rowCategory == "6":
                  if place_mentions != "" and pop_culture_mentions != "":
                     print("practical and insightful advice designed to offer you the most effective guidance for better decision-making and results regarding the following places: " + str(place_mentions) + ".")
                     print("However, you mentioned Iconic References like " + str(pop_culture_mentions) + ", yet I disregarded them due its irrelevancy to advice giving.")
                  elif place_mentions != "":
                     print("practical and insightful advice designed to offer you the most effective guidance for better decision-making and results in the following places: " + str(place_mentions) + ".")
                  elif pop_culture_mentions != "":
                     print("practical and insightful advice designed to offer you the most effective guidance for better decision-making and results.")
                     print("However, you mentioned Iconic References like " + str(pop_culture_mentions) + ", yet I
disregarded them due its irrelevancy to advice giving.")
                    else:
                        print("practical and insightful advice designed to offer you the most effective guidance for better decision-making and results.")
                user_input = input("\nIs my understanding right? Type 'Y' for Yes or 'N' for No: ")
                print("Glad I am doing it right. Data has been noted!") if user_input == "Y" else input("In what way should I have interpreted the response: ")
                print("Thank you for your feedback!\n")
                user_input = input("\nAsk me anything (or type 'stop' to end): ").lower()
    else:
        user_input = input("\nThis question is unrecognizable. Try again or type 'stop': ").lower()

# post-program time display and feedback system
end_time = time.time()
total_time = int(end_time - start_time)
min = str(int(total_time / 60))
sec = f"0{int(total_time % 60)}" if (total_time % 60) < 10 else str(int(total_time % 60))
user_input = input(f"Now that you have used the program for {min}:{sec}, let me know how well I did: ")
print("Your feedback will help in improving this QuerySense Model. Thank you.")
```