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

# 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):
    result = ""
    for places in list:
        if places.lower() in saved_input.lower():
            result += places + ", "
    return result

def validateItemAbsent(word, data_instance):
    for q in range(len(data_instance.possibleList)):
        for w in range(len(data_instance.possibleList[q])):
            if word == 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: 1.9]\n\n")
user_input = input("Ask me anything: ")
data_instance = Data()
rowCategory = None

# 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")[0] != "":
        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 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) ")

            # list of possible type of questions extracted from user input [non-exhaustive]
            if saved_input.__contains__("what if"):
                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):
                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":
                print("concise and short answer for general information related to places, conditions, activities, person.")
            elif rowCategory == "1":
                print("personal/health based information regarding diet/lifestyle.")
            elif rowCategory == "2":
                print("productivity tailored response for improving your health/work/performance efficiency.")
            elif rowCategory == "3":
                print("entertainment related answer that is tailored to improving your leisure/mood.")
            elif rowCategory == "4":
                print("mathematical answer, which can either be short or informative, for a concept or calculation.")
            elif rowCategory == "5":
                print("response that improves the your knowledge in crucial details regarding current or past events/information.")
            elif rowCategory == "6":
                print("advice to the you in the best manner possible that will give you better guidance.")
            if printPossibleMentions(saved_input, data_instance.specificPlaceList) != "":
                print("You mentioned " + printPossibleMentions(saved_input, data_instance.specificPlaceList) + "meaning you want my responses to be personalized for that/those place(s).")
            if printPossibleMentions(saved_input, data_instance.specificPopCultureList) != "":
                print("You also mentioned Iconic References like " + printPossibleMentions(saved_input, data_instance.specificPopCultureList) + "therefore my answers should be refined to that/those reference(s).")
            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()


```