```
"""
Evaluation UP8, défi : intelligence artificielle
    Sujet: construction d'un petit robot à l'aide de google API et NLTK

Nom et prénom : ADREF Brahim

"""

# -*- coding: utf-8 -*-
"""
Created on Mon Sep 23 15:33:22 2019

@author: brahim.adref
"""

import speech_recognition as sr
import os
import sys
from gtts import gTTS
import pyttsx3
#import nltk
from nltk.tokenize import word_tokenize
#nltk.download('punkt')


"""
La fonction principale que j'ai utlisé pour tokenize la commande envoiyée par littleRobot
(chaine de caractère) à l'aide de la librairie NLTK,
j'ai procédé pour le traitement comme suit:
    1. Tester sur la nature de la commande car chaque commande a son traitement.
    2. Tokenize la commande en mots pour faciliter l'extraction de la distance ou bien l'angle
    3. Extraire la distance/angle par test sur les mots à l'aide de la fonction isnumeric()
    4. Construire la réponse en chaine de caractère puis fait un appel à la fonction response()
       en passant la réponse en argument
    5. Affichage sur le console le movement que notre robot a fait
       {A: pour avancement, R: pour récule, Ro : pour rotation}

"""

def commandProcessing(command, commandType):
    "here we will use nltk for tokenizing the command and get the details of command"
    if commandType == "welcome":
        response("Hello, Mr. Brahim I'm your robot what i can do for you ?")
    elif commandType == "move":
        words = word_tokenize(command)
        distanceF = [s for s in words if s.isnumeric()]
        responseString = "Ok, I move " + distanceF[0] + " cm forward"
        response(responseString)
        print("A:", distanceF[0])
    elif commandType == "back":
        words = word_tokenize(command)
        distanceB = [s for s in words if s.isnumeric()]
        responseString = "Ok, I move " + distanceB[0] + " cm to the back"
        response(responseString)
        print("R:", distanceB[0])
    elif commandType == "rotate":
        words = word_tokenize(command)
        degree = [s for s in words if s.isnumeric()]
        responseString = "Ok, I rotate " + degree[0] + " degrees"
        response(responseString)
        print("Ro:", degree[0])
```

```
    elif commandType == "shutdown":
        response("Bye bye Mr. brahim. have a nice day")
        sys.exit()
    else:
        pass


"""
Une fonction response qui va nous faire la synthèse vocale, on a deux choix un pour
la librairie gTTS, mais dans mon cas ne fonctionne pas sur mon ordinateur,
et j'ai le deuxième choix de la librairie pyttsx3 (version 3) qui a bien fonctionné,
j'ai joué un peu avec les caractéristiques de la voix et de fréquence.

"""
def response(audio):

    # audio is a string
    print(audio)

    """
    La partie de gTTS qui ne fonctionne pas sur mon ordinateur
    """
#    tts = gTTS(text=audio, lang='en')
#    tts.save("audio.mp3")
#    os.system("mpg321 audio.mp3")

    engine = pyttsx3.init()
    engine.setProperty('rate', 125)
    voices = engine.getProperty('voices')
    engine.setProperty('voice', voices[1].id)
    engine.say(audio)
    engine.runAndWait()

"""
Une fonction qui va entendre la commande de l'utilisateur en utilisant la librairie
Speech Recognition, et API de google et elle va afficher sur le console
ce que l'utilisateur dit, en fin une commande comme chaine de caractère est retournée.
S'il y a une erreur de détection de voix ont reexecute la fonction (récursivité).
"""
def myCommand():

    r = sr.Recognizer()
    with sr.Microphone() as source:
        print('I m waiting your command ...')
        r.pause_threshold = 1
        # Pour annuler le parsite d'espace
        r.adjust_for_ambient_noise(source, duration=1)
        audio = r.listen(source)
    try:
        command = r.recognize_google(audio).lower()
        print('You said: ' + command + '\n')
    # S'il y a une erreur de détiction de voix
    except sr.UnknownValueError:
        print('....')
        command = myCommand();
    return command

"""
Fonction littleRobot qui va prendre en argument command ensuite pour chaque cas on va vérifier
la nature de la commande par l'appartenance du mot principal dans la commande (chaine de caractè
après en passe la commande à la fonction commandProcessing() avec sa nature, qui va s'occuper
```

```
du processus de traitement de commande pour tirer ses détails
"""
def littleRobot(command):
    if "little robot" in command:
        commandProcessing(command, "welcome")
    elif "move" in command:
        commandProcessing(command, "move")
    elif "back" in command:
        commandProcessing(command, "back")
    elif "rotate" in command:
        commandProcessing(command, "rotate")
    elif "shut down" in command:
        commandProcessing(command, "shutdoawn")

"""
La boucle infinie qui va prendre les commandes au fur et à mesure
Le point d'arrêt c'est la commande : shut down
"""
while True:
    littleRobot(myCommand())
```