"""

This sample demonstrates a simple skill built with the Amazon Alexa Skills Kit.

The Intent Schema, Custom Slots, and Sample Utterances for this skill, as well

as testing instructions are located at http://amzn.to/1LzFrj6

For additional samples, visit the Alexa Skills Kit Getting Started guide at

http://amzn.to/1LGWsLG

"""

from \_\_future\_\_ import print\_function

from botocore.vendored import requests

# --------------- Helpers that build all of the responses ----------------------

def build\_speechlet\_response(title, output, reprompt\_text, should\_end\_session):

return {

'outputSpeech': {

'type': 'PlainText',

'text': output

},

'card': {

'type': 'Simple',

'title': "SessionSpeechlet - " + title,

'content': "SessionSpeechlet - " + output

},

'reprompt': {

'outputSpeech': {

'type': 'PlainText',

'text': reprompt\_text

}

},

'shouldEndSession': should\_end\_session

}

def build\_response(session\_attributes, speechlet\_response):

return {

'version': '1.0',

'sessionAttributes': session\_attributes,

'response': speechlet\_response

}

# --------------- Functions that control the skill's behavior ------------------

def get\_welcome\_response():

""" If we wanted to initialize the session to have some attributes we could

add those here

"""

session\_attributes = {}

card\_title = "Welcome"

speech\_output = "Welcome to the Alexa Skills Kit sample. " \

"Please tell me your favorite color by saying, " \

"my favorite color is red"

# If the user either does not reply to the welcome message or says something

# that is not understood, they will be prompted again with this text.

reprompt\_text = "Please tell me your favorite color by saying, " \

"my favorite color is red."

should\_end\_session = False

return build\_response(session\_attributes, build\_speechlet\_response(

card\_title, speech\_output, reprompt\_text, should\_end\_session))

def handle\_session\_end\_request():

card\_title = "Session Ended"

speech\_output = "Thank you for trying the Alexa Skills Kit sample. " \

"Have a nice day! "

# Setting this to true ends the session and exits the skill.

should\_end\_session = True

return build\_response({}, build\_speechlet\_response(

card\_title, speech\_output, None, should\_end\_session))

def create\_favorite\_color\_attributes(favorite\_color):

return {"favoriteColor": favorite\_color}

def set\_color\_in\_session(intent, session):

""" Sets the color in the session and prepares the speech to reply to the

user.

"""

card\_title = intent['name']

session\_attributes = {}

should\_end\_session = False

if 'Color' in intent['slots']:

favorite\_color = intent['slots']['Color']['value']

session\_attributes = create\_favorite\_color\_attributes(favorite\_color)

speech\_output = "I now know your favorite color is " + \

favorite\_color + \

". You can ask me your favorite color by saying, " \

"what's my favorite color?"

reprompt\_text = "You can ask me your favorite color by saying, " \

"what's my favorite color?"

else:

speech\_output = "I'm not sure what your favorite color is. " \

"Please try again."

reprompt\_text = "I'm not sure what your favorite color is. " \

"You can tell me your favorite color by saying, " \

"my favorite color is red."

return build\_response(session\_attributes, build\_speechlet\_response(

card\_title, speech\_output, reprompt\_text, should\_end\_session))

def get\_color\_from\_session(intent, session):

session\_attributes = {}

reprompt\_text = None

if session.get('attributes', {}) and "favoriteColor" in session.get('attributes', {}):

favorite\_color = session['attributes']['favoriteColor']

speech\_output = "Your favorite color is " + favorite\_color + \

". Goodbye."

should\_end\_session = True

else:

speech\_output = "I'm not sure what your favorite color is. " \

"You can say, my favorite color is red."

should\_end\_session = False

# Setting reprompt\_text to None signifies that we do not want to reprompt

# the user. If the user does not respond or says something that is not

# understood, the session will end.

return build\_response(session\_attributes, build\_speechlet\_response(

intent['name'], speech\_output, reprompt\_text, should\_end\_session))

# --------------- Events ------------------

def on\_session\_started(session\_started\_request, session):

""" Called when the session starts """

payload = {'key1': "r", 'key2': 'value2'}

response=requests.post("http://a308e3e0.ngrok.io/login",params=payload)

print("on\_session\_started requestId=" + session\_started\_request['requestId']

+ ", sessionId=" + session['sessionId'])

def on\_launch(launch\_request, session):

""" Called when the user launches the skill without specifying what they

want

"""

print("on\_launch requestId=" + launch\_request['requestId'] +

", sessionId=" + session['sessionId'])

# Dispatch to your skill's launch

return get\_welcome\_response()

def on\_intent(intent\_request, session):

""" Called when the user specifies an intent for this skill """

print("on\_intent requestId=" + intent\_request['requestId'] +

", sessionId=" + session['sessionId'])

intent = intent\_request['intent']

intent\_name = intent\_request['intent']['name']

# Dispatch to your skill's intent handlers

if intent\_name == "MyColorIsIntent":

return set\_color\_in\_session(intent, session)

elif intent\_name == "WhatsMyColorIntent":

return get\_color\_from\_session(intent, session)

elif intent\_name == "AMAZON.HelpIntent":

return get\_welcome\_response()

elif intent\_name == "AMAZON.CancelIntent" or intent\_name == "AMAZON.StopIntent":

return handle\_session\_end\_request()

else:

raise ValueError("Invalid intent")

def on\_session\_ended(session\_ended\_request, session):

""" Called when the user ends the session.

Is not called when the skill returns should\_end\_session=true

"""

print("on\_session\_ended requestId=" + session\_ended\_request['requestId'] +

", sessionId=" + session['sessionId'])

# add cleanup logic here

# --------------- Main handler ------------------

def lambda\_handler(event, context):

""" Route the incoming request based on type (LaunchRequest, IntentRequest,

etc.) The JSON body of the request is provided in the event parameter.

"""

print("event.session.application.applicationId=" +

event['session']['application']['applicationId'])

"""

Uncomment this if statement and populate with your skill's application ID to

prevent someone else from configuring a skill that sends requests to this

function.

"""

# if (event['session']['application']['applicationId'] !=

# "amzn1.echo-sdk-ams.app.[unique-value-here]"):

# raise ValueError("Invalid Application ID")

if event['session']['new']:

on\_session\_started({'requestId': event['request']['requestId']},

event['session'])

if event['request']['type'] == "LaunchRequest":

return on\_launch(event['request'], event['session'])

elif event['request']['type'] == "IntentRequest":

return on\_intent(event['request'], event['session'])

elif event['request']['type'] == "SessionEndedRequest":

return on\_session\_ended(event['request'], event['session'])