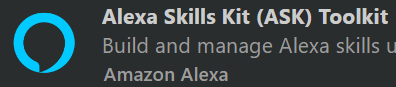
**Local debugging Alexa Hosted Skills using VS code**



https://youtu.be/ByE8b7d3LQM

**Prerequisites** Python, Git for Windows, and Node.js

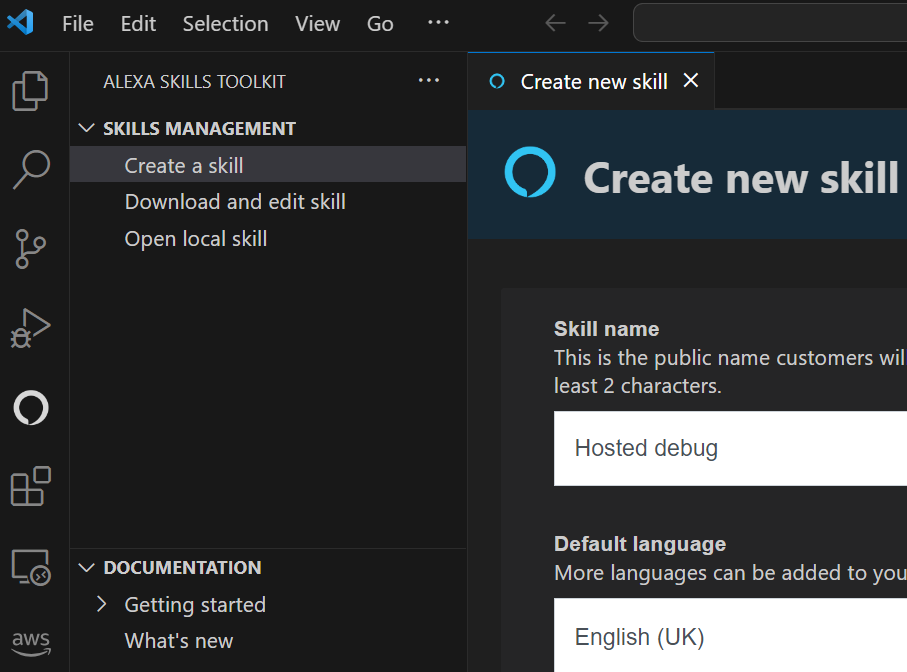
Install the Alexa Skills Toolkit for Visual Studio Code through the extensions manager.



And run **ask configure** to set up your environment

**Create your skill**

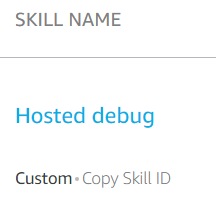
Here we’ll use ASK Toolkit



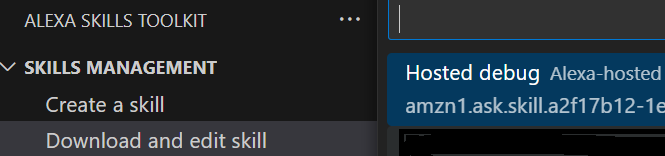
Answer Questions: Name (Hosted debug), Language (English (UK), Custom skill, Runtime (Alexa Hosted – python), Region (Europe(Ireland), and local directory.

Click **Create**

This will create a skill that you can see in the dev console

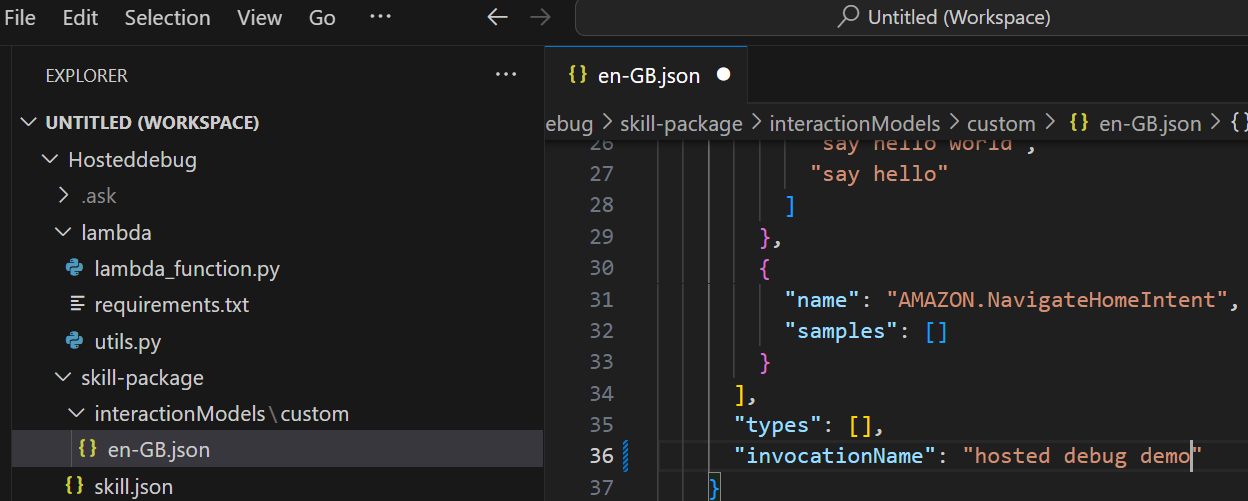


And you can also download to VS code



Choose the folder to download into and it appears in the VS code.

Change the invocation name using VS code



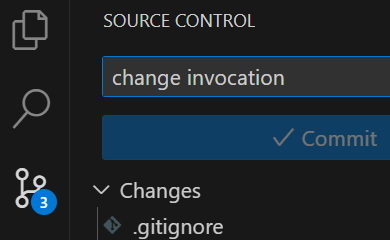
N.b. Invocation name cannot contain the words “alexa“, ”amazon“, ”echo“, ”skill“, ”app” or “ziggy”.

Make changes to the invocation name, save and commit.

Alexa hosted skill use git commands to update (not ask deploy).

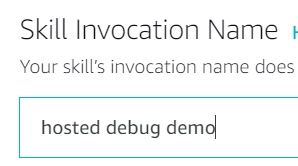
* git add .
* git commit -m “change invocation name”
* git push

You can try using Source control

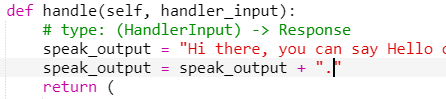


But if control-Enter doesn’t commit, see <https://github.com/microsoft/vscode/issues/108674>

Check in developer console



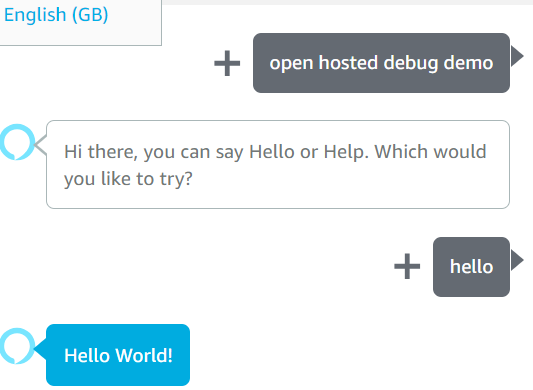
Now we’ll make a change in the dev console. Change the welcome message, and I’ve added an extra line to help with debugging



Save and deploy.

Check that it works

1. dev console



1. In VS code using **ask dialog** (or use ask toolkit)

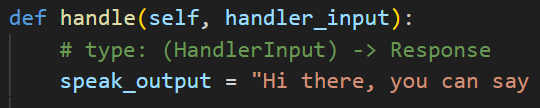
If you make a change in the developer console, use **git pull** to get the code.

It recognises the changes:

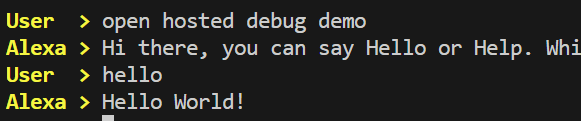
lambda/lambda\_function.py | 2 +-

1 file changed, 1 insertion(+), 1 deletion(-)

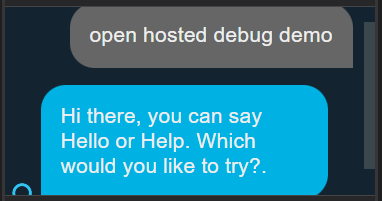
And you can see the change in your lambda function:



And test using **ask dialog**:

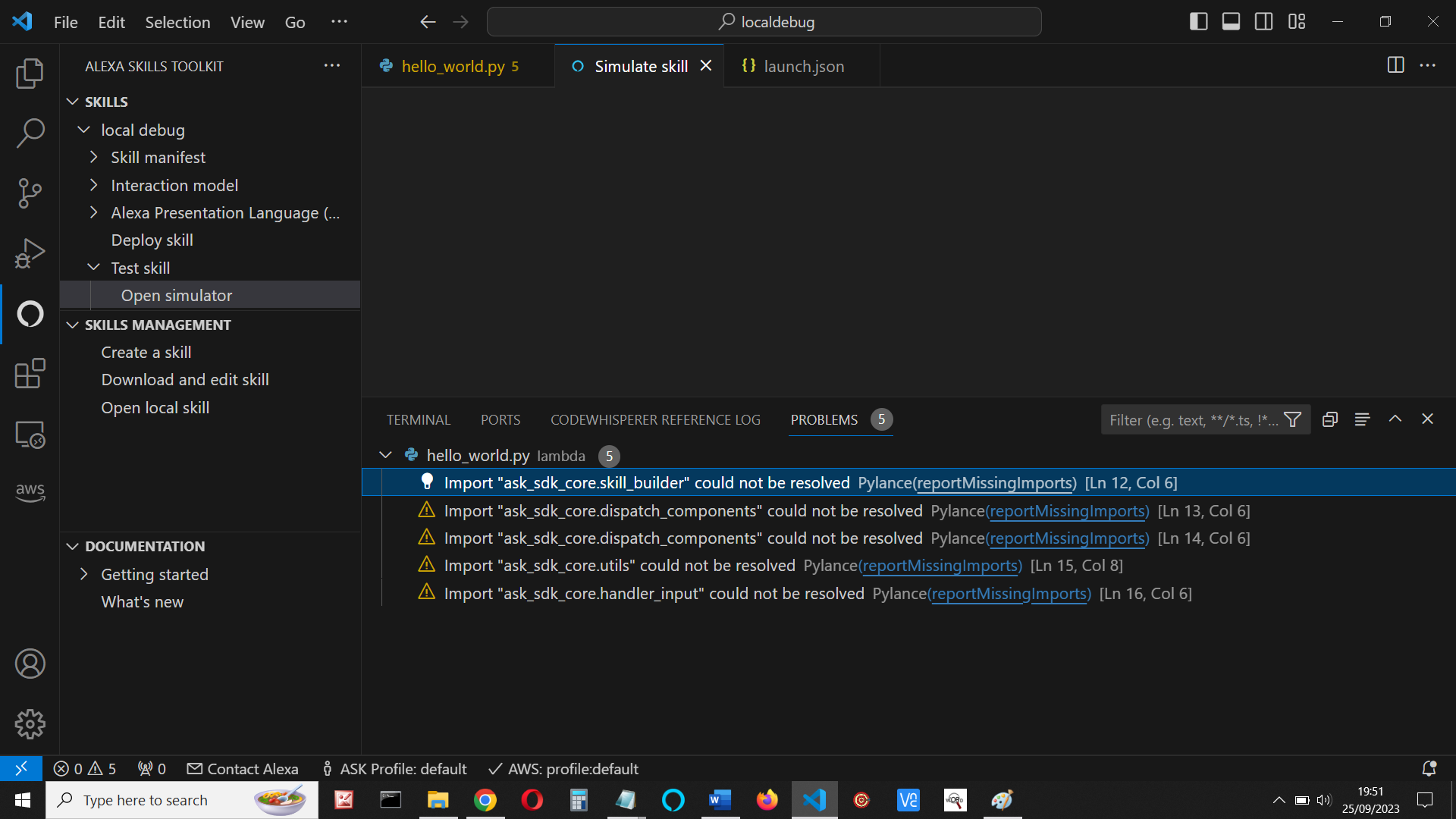


Or via ASK Simulator:



Now let’s debug our code locally

First, solve any missing link problems, e.g. ask-sdk-core:



Set up a virtual environment - run the following commands:

py -m venv env (create new workspace id asked)

.\env\Scripts\activate

pip install ask-sdk-core

pip install ask-sdk-local-debug

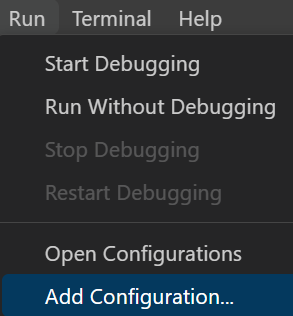
ask configure

Now set up local debugging.

**Add debugger configuration**.

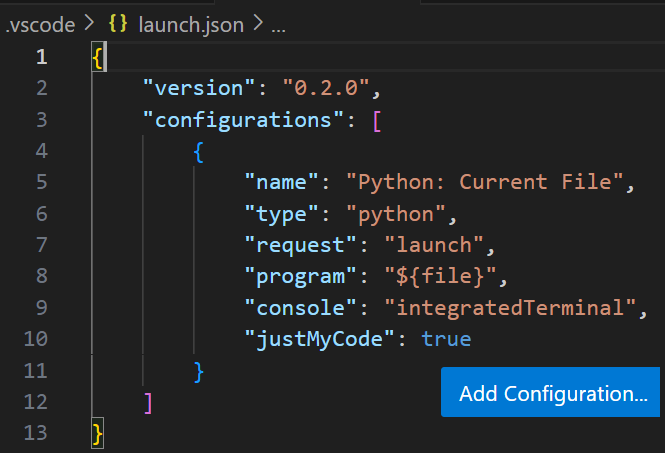
1. Create a launch.json file for your skill:

In VScode, display your lambda\_function.py code, then select Run menu, and then choose Add Configuration



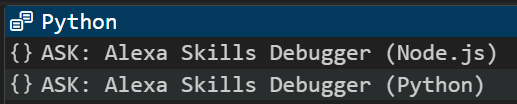
This creates a launch.json debugger for the current file.

Remove the // comments.

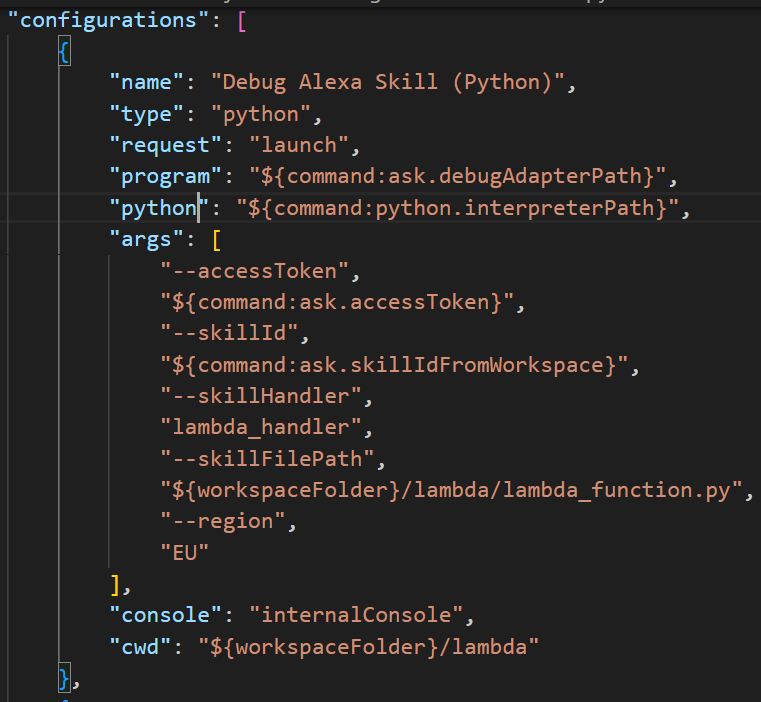


Click Add Configuration again.

1. In drop-down box, choose **ASK: Alexa Skills Debugger (Python)**



So that your launch.json looks like this (edit ‘pythonPath’ to ‘python’)



We still need to edit a few more items:

Check your region is correct – I’m in the EU

                "--region",

                "EU"

Edit ‘program’ to point directly to the local debugger invoker.py (not directly to the debugger)

**"program": "${workspaceFolder}/env/Lib/site-packages/ask\_sdk\_local\_debug/local\_debugger\_invoker.py",**

Save launch.json

**Debug your code**

In a terminal, type **ask run**. This can be in VS code or a separate cmd (or PS) ternimal

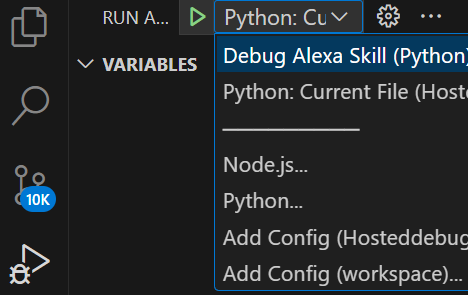
Response is:

\*\*\*\*\*Once the session is successfully started, you can use `ask dialog` to make simulation requests to your local skill code\*\*\*\*\*

You now have an hour’s debugging session.

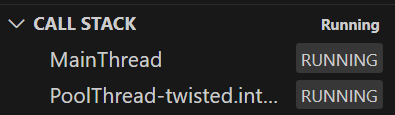
Start a debugging session

Click the debug symbol and make sure Debug Alexa Skill is selected:



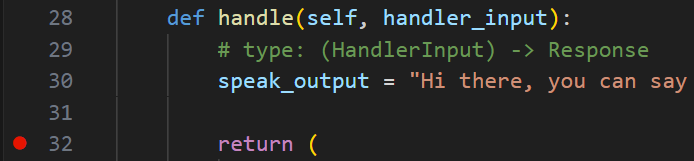
Then click the green triangle to start a session.

You should see ‘Running’ in the call stack.



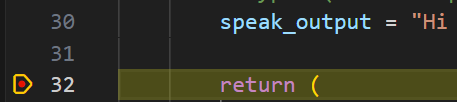
You can now debug your code using the CLI, the Simulator, or a device

I’ve set a breakpoint at the return in the launch request.



Open another terminal, change to the to the skill’s folder and type **ask dialog.**

Type your launch command (‘open hosted debug demo’) and the code should break

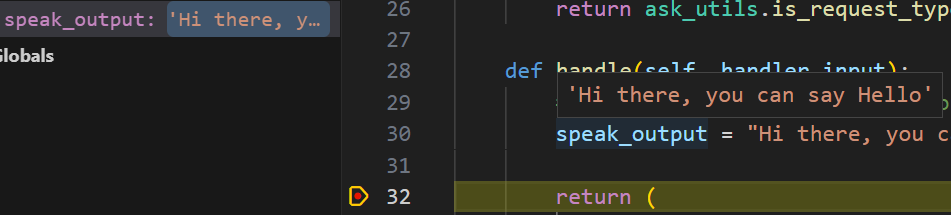


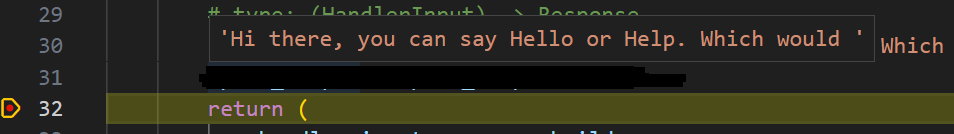
You can now look at variables, step or do any debug command:

**Problem: Changing variables**

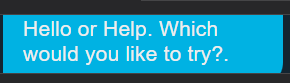
If you change a variable at a breakpoint, you must do a step before the change is committed.

Here, I’ve truncated the speak\_output variable in the debug window (F2)





If you continue by pressing F5 or the continue icon, the speak\_output in the simulator display hasn’t changed



Don’t forget, if you don’t do this quickly enough, the skill will time out.

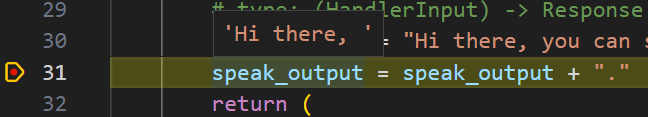
**Solution:**

This is why we added another line of code initially. Set the breakpoint there, so that you can step to the return statement, then run, enforcing the change to the variable.

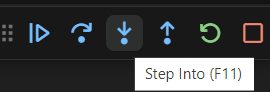
speak\_output = speak\_output + "."

Run the code and at the breakpoint, I’ve changed the speak\_output variable:





Having made the change, perform one step to enforce the change and then run the code.



The correct value is then displayed in the simulator (albeit with the extra dot)



But you have to be quick!

**Make changes to your code**

As an example, I’ve changed this line (added a few more dots)



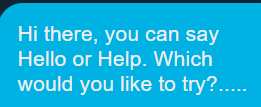
I still have the breakpoint.

Save the lambda\_function.py code

Start another ask run if necessary and restart a debug session. (Click the red square and then the green triangle)

It will stop at the debug, but you can continue

speak\_output now contains the extra dots.

c

You can continue changing code and locally debugging your code and finally save the changes to github as before

* git add .
* git commit -m “change invocation name”
* git push

Check the changes have been made in dev console skill.

Remember if you make change in the dev console, then you need to use **git pull** in VS code.

**Summary**

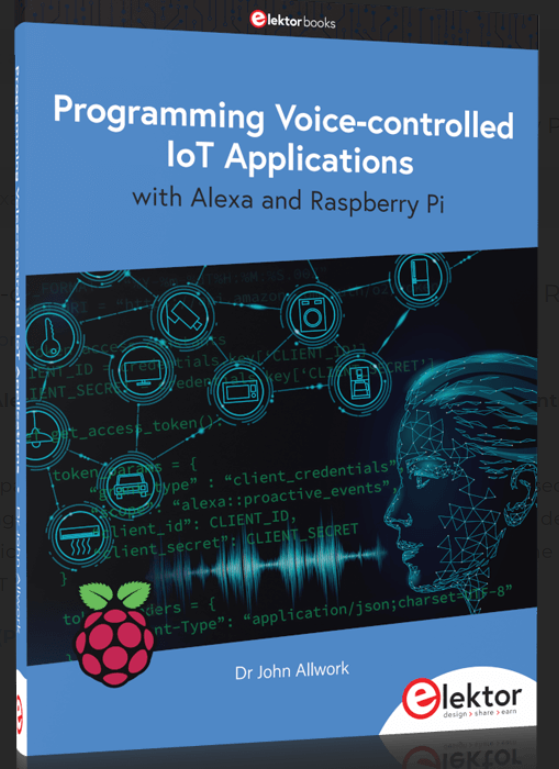
We have seen how to create an Alexa Hosted skill and move the files between the developer console and your VS environment.

We saw how to resolve any missing files problems and set up a virtual environment before going on to create a launch.json file, editing that so it points to the correct files for debugging.

We then saw how to debug our code using breakpoints and how to change variables, making those permanent by adding a step. Finally we make changes to our code, debugged the changes and uploaded those to the developer console

I hope this all helps

**My book**

****

**https://www.elektor.com/programming-voice-controlled-iot-applications-with-alexa-and-raspberry-pi**

**References:**

Managing Alexa-Hosted Skills in GitHub - Dabble Lab #245

<https://www.youtube.com/watch?v=88AsF_xJsj0>

https://www.andreasjakl.com/local-debugging-of-alexa-skills-with-visual-studio-code/