Alexa tech friday mentor handout

1) **Build one-liner backend**
    a) **Open Visual Studio Code**
    b) **Create a new folder on the desktop**
    c) **Create a new file called *jokes.py***
    d) **The first bit of code to add to *jokes.py* :**

```
from flask import Flask
from flask_ask import Ask, statement, question

app = Flask(__name__)
ask = Ask(app, '/')

@ask.intent('OneLiner')
def one_liner():
    return statement('Warning, keyboard not found. Press Enter to continue.')

if __name__ == '__main__':
    app.run(debug=True)
```

2) **Build interaction model**
    a) **Go to https://developer.amazon.com/**
    b) **Login with an existing or create an amazon account**
        i) **Fill out the required information**
        ii) **Agree to the license**
        iii) **Choose an option for monetization (no for now but can be changed later)**
    c) **Click the Alexa tab**
    d) **Click “Get Started” under Alexa Skills Kit**
    e) **Click “Create Skill”**
    f) **For the skill name, call it *“Computer Jokes”***
    g) **Select “Custom Model” then click “Create Skill”**
    h) **In between each following substep click the *“Custom”* button:**
        i) **On the right Click “*Invocation Name”***
            (1) **For skill invocation name call it *“Computer Jokes”***
        ii) ***On the right click “Intents, samples, and slots”***
            (1) **With the *Create Custom Intent* radial button selected enter “*OneLiner*” as the intent name and click create custom intent**
                (a) **NOTE!!!!! THIS NAME NEEDS TO BE THE SAME AS WHAT IS IN YOUR CODE**
            (2) **On the sample utterances page enter the following**
                (a) **Oneliner**
                (b) **One liner**

(c) Give me a one liner

(d) I want to hear a one liner

i) Click *Build Model* and wait until it finishes

3) Testing

a) Back in Visual Studio Code Open powershell as the terminal by pressing "CTRL + ~"

b) In the powershell console type `python jokes.py`

c) Open a new shell by clicking the + next to the name of the shell

d) Find the NGROK executable and move it to the folder on the desktop made earlier

e) In the new console type `.\ngrok http 5000`

i) You will see something like this

```
ngrok by @inconshreveable

Session Status                online
Session Expires               7 hours, 59 minutes
Version                       2.2.8
Region                        United States (us)
Web Interface                 http://127.0.0.1:4040
Forwarding                    http://4d4219f6.ngrok.io -> localhost:5000
Forwarding                    https://4d4219f6.ngrok.io -> localhost:5000

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.00
```

ii) Select the bottom forwarding URL that contains HTTPS by highlighting then copy it by pressing *"CTRL + C"*

f) Back in the amazon developer console paste the URL we just copied in the Default Region URL box

g) Select "*my development endpoint is a sub-domain of a domain that has a wildcard certificate authority"* from the dropdown list

h) Click Save Endpoints

i) Click the test tab

i) Begin to test the skill by invoking typing "*ask computer jokes for a one liner*"

(1) If this does not return the string that we are expecting check that the python program and ngrok are running and that the URL is the same as the one from ngrok

ii) Continue testing the skill by using your headset to invoke the skill

j) Go to https://echosim.io/ and log in with the same amazon account used earlier

i) Use the headset to invoke the skill again

4) **Build Knock Knock Backend**
    a) **In *jokes.py* add the bold code such that the file looks like the following**

```
from flask import Flask
from flask_ask import Ask, statement, question

app = Flask(__name__)
ask = Ask(app, '/')

@ask.launch
def launched():
    return question('Welcome to Jokes! What kind of joke do you want to hear, a one
liner or a knock knock joke?')\
        .reprompt('Sorry I missed that. Would you like to hear a one liner or a knock
knock joke?')

@ask.intent('OneLiner')
def one_liner():
    return statement('Warning, keyboard not found. Press Enter to continue.')

@ask.intent('KnockKnock')
def knock_knock():
    return question('Knock knock.')

@ask.intent('WhosThere')
def who():
    return question('Bing')

@ask.intent('Bing')
def ie_response():
    return statement('No, I usually prefer Google')

if __name__ == '__main__':
    app.run(debug=True)
```

5) **Build the interaction model**
    a) **Add intents and sample utterances for "*KnockKnock*", "*WhosThere*", and "*Bing*"**
    b) **Click the build model button and wait for it to build**
6) **Testing**
    a) **In Visual Studio Code run `python jokes.py` in one console**
    b) **In another console run `.\ngrok http 5000`**

i) **Select the bottom forwarding URL that contains HTTPS by highlighting then copy it by pressing *"CTRL + C"***

c) **Back in the amazon developer console paste the URL we just copied in the Default Region URL box**

d) **Select "*my development endpoint is a sub-domain of a domain that has a wildcard certificate authority"* from the dropdown list**

e) **Click Save Endpoints**

f) **Click the test tab**

i) **Begin to test the skill by invoking typing "*ask computer jokes for a one liner*"**

(1) **If this does not return the string that we are expecting check that the python program and ngrok are running and that the URL is the same as the one from ngrok**

ii) **Continue testing the skill by using your headset to invoke the skill**

g) **Go to https://echosim.io/ and log in with the same amazon account used earlier**

7) **Build Calculator Backend**

a) **In Visual Studio code create a new file called "*calculator.py*"**

b) **Add the following code**

```
from flask import Flask
from flask_ask import Ask, statement, question, session, delegate

app = Flask(__name__)
ask = Ask(app, '/')

def get_dialog_state():
    return session['dialogState']

@ask.launch
def launched():
    return question('Welcome to Calculation! What kind of calculation would you like
to perform?')\
        .reprompt('Sorry I missed that. What kind of calculation would you like to
do?')

@ask.intent('Addition', convert={'first': int, 'second': int})
def addition(first, second):
    dialog_state = get_dialog_state()
    if dialog_state != "COMPLETED":
        return delegate()

    sum = first + second
```

```
    return statement('The sum of {} and {} is {}'.format(first, second, sum))

@ask.intent('Subtraction', convert={'first': int, 'second': int})
def subtraction(first, second):
    dialog_state = get_dialog_state()
    if dialog_state != "COMPLETED":
        return delegate()

    difference = first - second
    return statement('The difference between {} and {} is {}'.format(first, second,
difference))

@ask.intent('Multiplication', convert={'first': int, 'second': int})
def multiplication(first, second):
    dialog_state = get_dialog_state()
    if dialog_state != "COMPLETED":
        return delegate()

    product = first * second
    return statement('The product of {} and {} is {}'.format(first, second, product))

@ask.intent('Division')
def division(first, second):
    dialog_state = get_dialog_state()
    if dialog_state != "COMPLETED":
        return delegate()

    if second == 0:
        return statement('Sorry I cannot divide by zero it doesnt make sense.')
    quotient = first // second
    return statement('The quotient of {} and {} is {}'.format(first, second,
quotient))

@ask.intent('Modulus', convert={'first': int, 'second': int})
def modulus(first, second):
    dialog_state = get_dialog_state()
    if dialog_state != "COMPLETED":
        return delegate()

    if second == 0:
        return statement('Sorry I cannot divide by zero it doesnt make sense.')
    remainder = first % second
    return statement('The mod of {} and {} is {}'.format(first, second, remainder))

if __name__ == '__main__':
    app.run(debug=True)
```

8) **Build the interaction model**
    a) **Create a new skill called *"Calculator"***
    b) **Select Custom Model then click create skill**
    c) **In between each following substep click the *"Custom"* button**
        i) **On the right Click *Invocation Name***
            (1) **For skill invocation name call it *"calculator"***
        ii) ***On the right click Intents, samples, and slots***
            (1) **Add intents and sample utterances for each endpoint created in the code**
        iii) **Click *Build Model* and wait until it finishes**
9) **Testing**
    a) **Test the same way as before**