You have **2** free member-only stories left this month. Upgrade for unlimited access.

Build Your Own Alexa With Just 20 Lines of Python

Programmers can get anything for free

Shajedul Karim Follow
Jan 17, 2021 · 8 min read ★

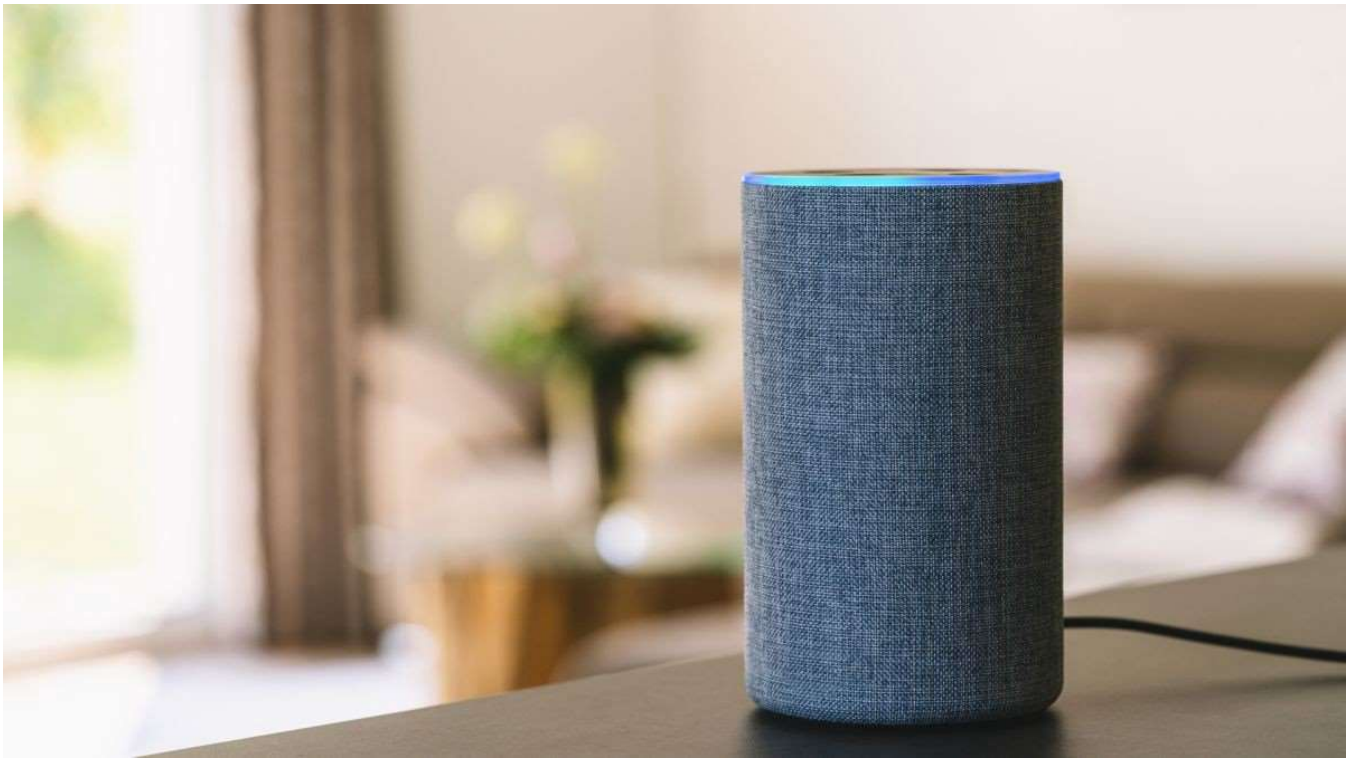

Image Source: Dreamstime.com

All my friends have Alexa. I don't.

It sucks. They make fun of me: "Shajedul from the 1943 BC," followed by a laugh which, at other times seems perfect, but while making fun of me, it's horrendous.

I said to myself "enough of it now." My pocket money is over. I was worried. I had to find some way.

I reached out to my team. Before I spoke any word other than Alexa, one of the team members shouts out: “Don’t buy Alexa! Build your own. With just 20 lines of python.”

What Is This Project About?

Save your money and build your own Alexa. Make her do anything, yeah anything. She is all yours.

Your grandma will be proud of you. She won’t have to do stuff manually. Just with her cute voice command, the assistant will be ready to do anything.

Just Know These Terms:

Module/Library:

A predefined or prewritten code by someone else that we can use in our project for free.

Class:

An OOP concept that allows us to group a bunch of code and is like a blueprint to create objects. This makes a code reusable.

Object:

An instance of a class that can be used to access the attributes and methods of a class.

Alexa Has Only 2 Tasks:

1. Listening

Listening to your command is the most basic functionality of any virtual assistant, like: “Hey Alexa, play music,” “Hey Alexa, what’s the time?”

Alexa has to listen to your command, understand it, and then do some action.

2. Speaking

Once Alexa listens and understands your command, it performs some action based on it. While doing that, it responds to you by speaking otherwise it'll be jobless.

Let's Implement Those Two Features:

To implement the above two features, we'll need two Python modules:

1. SpeechRecognition
2. Python Text-To-Speech (pyttsx3)

1. SpeechRecognition

This library performs speech recognition. It'll help the assistant listen to our commands, understand them, and act accordingly.

Anything third-party needs some kind of installation. You are the third-party of your mom and dad. They did some kind of installation. Ask them what.

Let's install this third-party module using the command below, on your terminal:

```
pip install SpeechRecognition
```

Once installed, we can use it in our project. While working with this module, there are three important things that we'll need:

a. The Recognizer class: This is the main class of this module which has all the major functions to help us create a speech recognition application.

Before we can use it, first we'll need to initialize it and create its object:

```
r = sr.Recognizer()
```

Here, '*r*' is just the name given to the object. It can be any valid Python variable name.

b. Microphone access: As the assistant needs to listen to your command, you'll need to give it the power to access the microphone of your machine. You can do that with the

help of the Microphone class:

```
# open the microphone and start recording
with Microphone() as source:
   # do things here - ``source`` is the Microphone instance created
above
   pass
```

c. Listening to user speech: Once it has the microphone access, the final step is to listen to your command. You can do that with the help of the *listen()* method provided by the Recognizer class:

```
# Listening to the user speech
# Accepts the audio source as the parameter
r.listen(source)
```

That's how you can work with Speech Recognition in Python. With this basic knowledge of this module, let's move ahead.

2. Python Text-To-Speech (pyttsx3)

This is the Text-to-Speech (TTS) library for Python 2 and Python 3 and works without an internet connection or any delay.

Since it's a third-party module, first you'll have to install it:

```
pip install pyttsx3
```

Your assistant can finally speak with the help of this module.

Secret: We're just converting text to speech here.

Everything else is now a piece of cake for you. First, you'll to initialize the pyttsx3 module using the *init()* method and create its object. We can then use its various functions to convert text to speech:

```
engine = pyttsx3.init()
engine.say("Text to Speak Here")
```

```
engine.runAndWait()
```

Here, the *say()* does the major job of converting text to speech, and the *runAndWait()* waits for the module to complete speaking a particular sentence before doing some other task.

With the basic knowledge of these two modules, let's move ahead and finally start the fun part.

Starting the Fun Part

We'll create three different functions and each will be responsible for a single task. But before that let's get our boilerplate code ready:

Step 1. Importing modules

That is always the very first step. Let's do it:

```
import speech_recognition as sr
import pyttsx3
```

Step 2. Initialization of modules

To use these modules, we'll always need to initialize them and create their objects, so let's do that too:

```
listener = sr.Recognizer()
engine = pyttsx3.init()
voices = engine.getProperty('voices')
engine.setProperty('voice', voices[1].id)
```

Here, '*listener*' is the object of the Recognizer class.

The pyttsx3 module is the voice stealer. It steals different types of voice and stores it in a property called '*voice*'.

It hates males but he has to store his voice anyway so the default is the male voice with the index 0. Alexa isn't Jarvis. We need a sweet female voice. We got it using the getProperty() method. The female voice is stored in index 1.

Now let's move ahead and create our methods that will help the assistant speak and listen.

Step 3. Creating a method to convert text to speech that will help our assistant speak - talk() method.

```
def talk(text):
    engine.say(text)
    engine.runAndWait()
```

Here, *talk()* is the name of the method and it takes the parameter '*text*.' This text can be any string that we wish to convert to speech so that our assistant can speak.

Then we simply pass it to the *say()* method and make a call to the *runAndWait()* method over the *engine* object that we have already created. You know how these work because you now know the basics of pyttsx3. Now your assistant is powerful enough to speak.

Let's give it the power to listen and understand our command by creating a method that will take care of speech recognition.

Step 4. Creating a method for Speech Recognition - take_command() method

```
def take_command():
    try:
        with sr.Microphone() as source:
            print('listening...')
            voice = listener.listen(source)
            command = listener.recognize_google(voice)
            print(command)
    except:
        pass
    return command
```

To handle an unexpected error with a microphone or some other error, we wrap it in the try-except block.

The rest is quite the same as what we have done earlier. The *speech recognition* module provides various speech recognition engines to do the job. Here, we're using Google's speech recognition engine. To do that we have used the *recognize_google()* method provided by the Recognizer class.

We are halfway through our project and we are done with the main part.

As you can see in the above function, the recognized speech is stored and returned in the *command* variable. We now just need to check what's stored in the *command* variable and perform the task accordingly.

Let's do that. That would be fun.

Step 5. Creating a method for response - run_alexa() method

```
def run_alexa():
    command = take_command()
    print(command)
    if 'play' in command:
        song = command.replace('play', '')
        talk('playing ' + song)
        pywhatkit.playonyt(song)
    elif 'time' in command:
        time = datetime.datetime.now().strftime('%I:%M %p')
        talk('Current time is ' + time)
    elif 'who the heck is' in command:
        person = command.replace('who the heck is', '')
        info = wikipedia.summary(person, 1)
        print(info)
        talk(info)
    elif 'date' in command:
        talk('sorry, I have a headache')
    elif 'are you single' in command:
        talk('I am in a relationship with wifi')
    elif 'joke' in command:
        talk(pyjokes.get_joke())
    else:
        talk('please say the command again')
```

Here, we need to understand a few things:

i. Fetching required part: Assume song ghost is tickling you to listen to songs and now you are ordering Alexa. You'll do that this way: Play song_name. From this command, for this project, we just remove the word '*play*' and fetch the '*song_name*' part:

```
song = command.replace('play', '')
```

And, we simply store the result in the variable named *song*.

ii) pywhatkit.playonyt(): To use this, first we'll need to install and import the pywhatkit module. PyWhatKit is a Python library for sending WhatsApp messages at a certain time, but it has several other features too to help us in automation. This module provides a *playonyt()* method which will help us play the required songs directly on YouTube.

Since it's a third-party module, first we'll have to install it:

```
pip install pywhatkit
```

And then import it:

```
import pywhatkit
```

iii. datetime.datetime.now(): To use this, first we'll need to import the datetime module. This module helps us to manipulate dates and times and is a built-in Python module. The method *now()* returns the current time.

Import it this way:

```
import datetime
```

iv. wikipedia.summary(): To use this, first we'll need to install and import the Wikipedia module. Wikipedia is a Python library that makes it easy to access and parse data from Wikipedia. It will help us to search the required data from Wikipedia and return it as the output. The *summary()* method fetches the data from the summary section of Wikipedia.

Since its a third party module, first, we'll have to install it:

```
pip install wikipedia
```

And then import it:

```
import wikipedia
```

v. pyjokes.get_joke(): To use this, first we'll need to install and import the Pyjokes module. This module will help us generate random one-line jokes for programmers that your assistant can crack.

Since it's a third-party module, first, we'll have to install it:

```
pip install pyjokes
```

And then import it:

```
import pyjokes
```

That's it! Super-simple, right? You're right.

In the same way, you can add more elif's and add more features.

In the above function, initially, we make the call to the *talk_command()* method which will start listening for our commands and store them in the command variable:

```
command = take_command()
    print(command)
```

Also, whenever we need our assistant to speak, we make the call to the *talk()* method and pass the required data which we want our bot to speak.

Step 6. Making the initial function call

```
while True:
    run_alexa()
```

Finally, we make the call to the *run_alexa()* method which will make our assistant up and running.

Now you have your own Alexa. Yay!

Similarly, using your Python skills and other python modules you can add other amazing features to your Alexa and make it a hot, smart, and beautiful virtual assistant.

The Whole Code

```
import speech_recognition as sr
import pyttsx3
import pywhatkit
import datetime
import wikipedia
import pyjokes

listener = sr.Recognizer()
engine = pyttsx3.init()
voices = engine.getProperty('voices')
engine.setProperty('voice', voices[1].id)

def talk(text):
    engine.say(text)
    engine.runAndWait()

def take_command():
    try:
        with sr.Microphone() as source:
            print('listening...')
            voice = listener.listen(source)
            command = listener.recognize_google(voice)
            print(command)
    except:
        pass
    return command

def run_alexa():
    command = take_command()
    print(command)
```

```
        if 'play' in command:
            song = command.replace('play', '')
            talk('playing ' + song)
            pywhatkit.playonyt(song)
        elif 'time' in command:
            time = datetime.datetime.now().strftime('%I:%M %p')
            talk('Current time is ' + time)
        elif 'who the heck is' in command:
            person = command.replace('who the heck is', '')
            info = wikipedia.summary(person, 1)
            print(info)
            talk(info)
        elif 'date' in command:
            talk('sorry, I have a headache')
        elif 'are you single' in command:
            talk('I am in a relationship with wifi')
        elif 'joke' in command:
            talk(pyjokes.get_joke())
        else:
            talk('please say the command again')

    while True:
        run_alexa()
```

Enjoy your Alexa. She is so fun.

Thanks to The Startup.

Programming Python Automation Alexa Amazon

About Write Help Legal

Get the Medium app

Download on the App Store GET IT ON Google Play