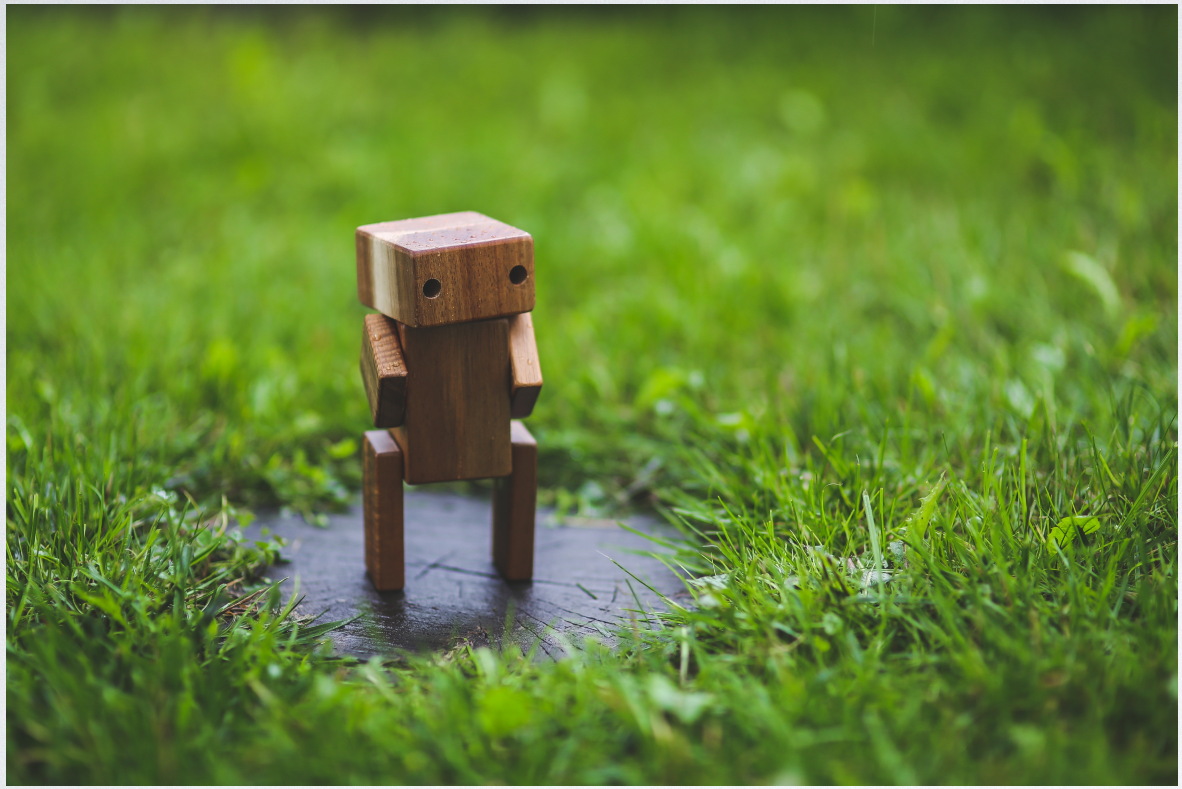
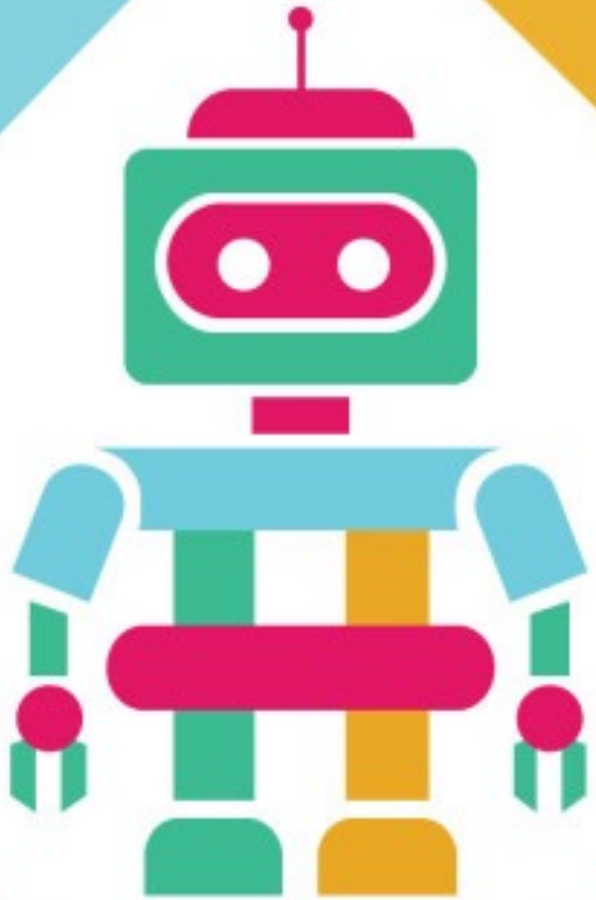
Simple Slack Bots

Green Bottle Tutorial



"Design from Empathy"



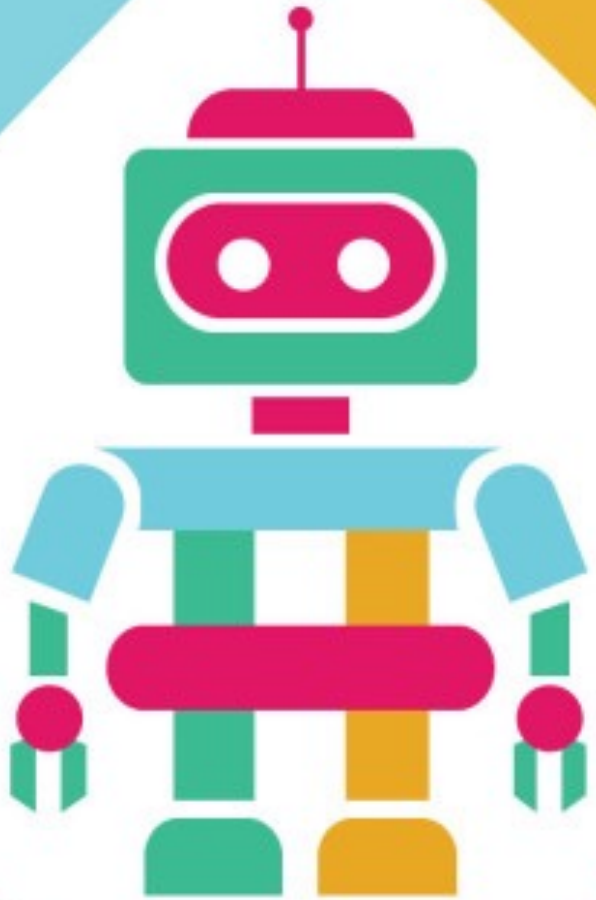
Why?

Why Should I Care?

Slack is a powerful tool, but…
… it's not perfect.

There's more that people want, that could make it better. Things that would improve their lives.

Many of these things can be made with bots, and we're gonna learn how to make one.

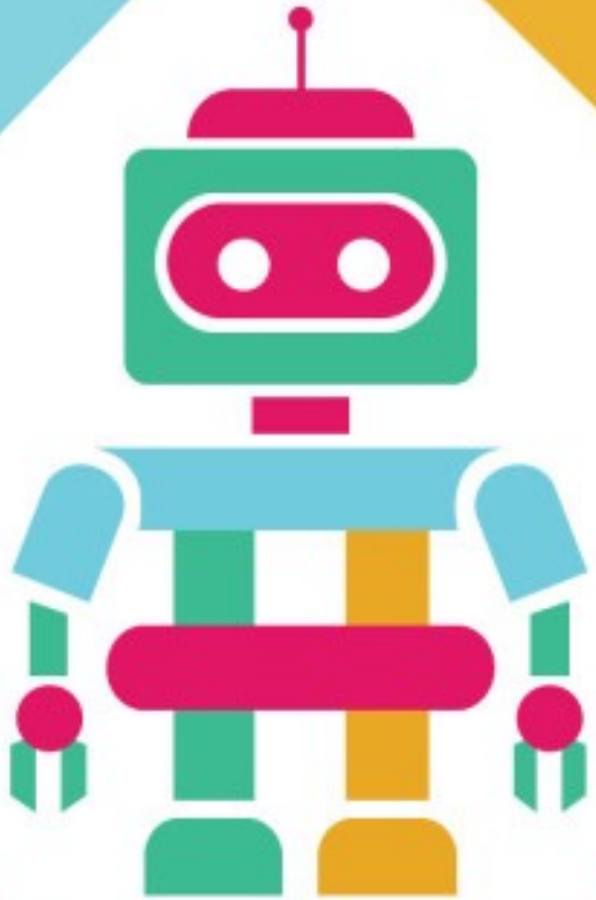


What?

What Is A (Ro)Bot Anyway?

A bot is a program that joins a slack team and listens for direct messages, and optionally one or more channels.

It can then respond to those messages with information, assistance, or anything you can imagine.



How?

So How Do I Make One?

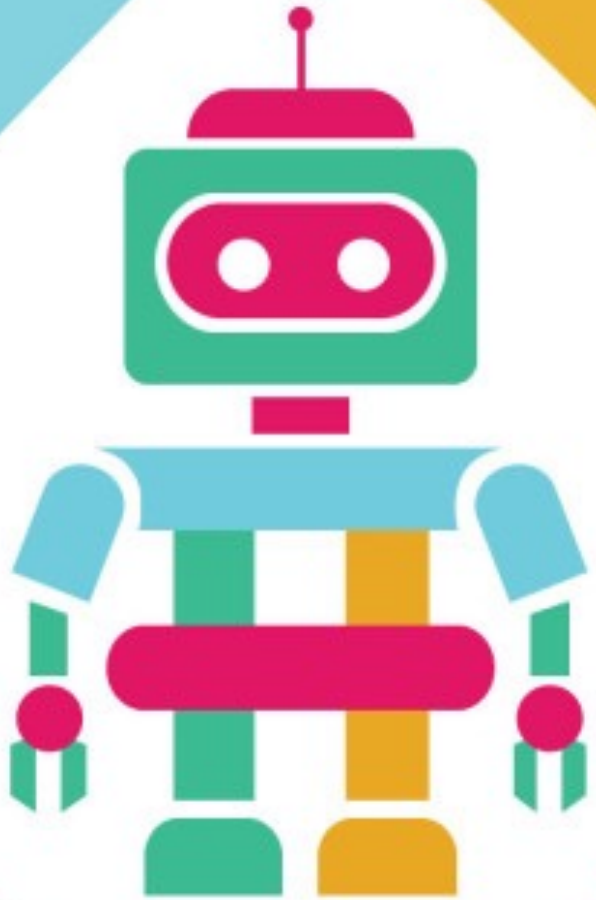
We have a carefully prepared “live” demo!

For this demo I assume a few things.

Firstly, that you’re comfortable with at least copy and pasting code.

Secondly that you have git installed.

That’s all!



Setup

Let's Get Python!

First you need to install a few dependencies:

- Python 3 (https://www.python.org/downloads/)
  What we're using to write our slackbot.
- Pip (https://pip.pypa.io/en/stable/installing/ and `$ python3 get-pip.py`)
  What we use to install Python modules.
- slackclient (`$ pip install slackclient` in your terminal)
  The Slack API for Python.

Generating a Token

Identification, Please

A token is how your program identifies itself as a bot.

Log on to https://your-team.slack.com/apps, and go to ‘Build’.

Choose to ‘Make a Custom Integration’.

Select ‘Bots’ and create a new integration.

Your token will look something like this (it’s fake, don’t worry):
`xoxb-26104543440-kmOdZgCD2HQ4BodPIkrb8Vdc`

Connecting to Slack

How Do I Connect This?

```
Python 3.5.1 (v3.5.1:37a07cee5969, Dec  5 2015, 21:12:44)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> from slackclient import SlackClient
>>> token = 'xoxb-26104543440-kmOdZgCD2HQ4BodPIkrb8Vdc'
>>> sc = SlackClient(token)
>>> if sc.rtm_connect() == True:
...   print('Connected')
... else:
...   print('Error. Probably an incorrect token.’)
...
Connected
```

Listening

Hear No Evil

```
>>> from time import sleep
>>> # Once connected successfully.
>>> while True:
...     print(sc.rtm_read())
...     sleep(1)
...
[{'type': 'hello'}]
[]
[{'type': 'presence_change', 'presence': 'active',
  'user': 'U0PDQ1P2R'}] # Changed to my user id.
>>> # This will keep returning lists of hashes, showing activity.
```

Reading a Message

See No Evil

To add your bot to a channel, you have to invite it in the channel. For my example, we have invited echobot to the #squad-slackbot. It can now listen to the channel.

Here's what it looks like when you connect when someone types a message. (Data anonymised)

The hash on the second line contains all the info necessary to process the message.

```
[{'type': 'user_typing', 'user': 'U0PDQ1P2R', 'channel':
'C1ARL9BA5'}]
[{'type': 'message', 'ts': '1463989925.000129', 'channel':
'C1ARL9BA5', 'user': 'U0PDQ1P2R', 'text': 'Test', 'team':
'H5AHBRKMA'}]
```

Sending a Message

Speak No Evil

By now you probably want to send a message.

Here's how to do it. First stop listening by pressing `control + c`

```
>>> sc.api_call('chat.postMessage', channel='#your-channel',
text='Hello from Python! :tada:', username='your-bots-name',
icon_emoji=':robot_face:')
```

Nice and simple.

But this has all been in the interpreter so far…

Putting it All Together

Time To Party

This is slightly modified code that should go in a `.py` file. Mine's called `echobot.py`

```
from time import sleep
from slackclient import SlackClient
import os

token = 'xoxb-26104543440-kmOdZgCD2HQ4BodPIkrb8Vdc' # Still fake
sc = SlackClient(token)
if sc.rtm_connect() == True:
  print('Connected.')
  while True:
    response = sc.rtm_read()
    for part in response:
      if part['type'] == 'message':
        if part['text'].lower().startswith('echo: '):
          sc.api_call('chat.postMessage', channel=part['channel'], text=part['text'][len('echo:
'):], username='echobot', icon_emoji=':robot_face:')
    sleep(1)
else:
  print('Connection Failed, invalid token?')
```

Hosting on Heroku

What If I Want To Close My Laptop?

Create an account on Heroku, or log in. They have a free tier, so don't worry about that.

Edit your program to remove the key from it. (This is a big security risk!)

```
token = os.environ.get('TOKEN', None)
```

This will allow it to read the environment variables wherever it's running, and if it can't find the token, it'll prompt the user to enter one.

Installing Heroku

Pass Me That Spanner

Now we need to set up your deployment to Heroku. First we need to install the Heroku Toolbelt. You can install this manually from https://toolbelt.heroku.com/, or by using a package manager such as Homebrew.

```
$ brew install heroku-toolbelt
$ brew upgrade heroku-toolbelt
```

To use Python with Heroku, we need to tell it what modules to install, namely SlackClient. To do this, we create a requirements.txt file and put the following line in it.

```
slackclient
```

Connecting it All Up

Nothing Some Duct Tape Can't Fix

Now, in the directory where you've saved your bot's code, we're gonna do a few things. This sets it up to allow you to deploy your app to Heroku.

```
$ git init
$ heroku git:remote -a your-app
$ heroku config:set TOKEN=xoxb-26104543440-kmOdZgCD2HQ4BodPIkrb8Vdc
```

We also need to create a Procfile and a runtime.txt file. These are special files that tells Heroku how to run your app. Here's what mine look like:

Procfile

```
web: gunicorn run-heroku:app
worker: python echobot.py
```

runtime.txt

```
python-3.5.1
```

Yup… That's it. Just replace `echobot.py` with whatever your file is.

Deploying!

Beam Me Up, Scotty!

Now, in the directory where you've saved your bot's code, we're gonna do a few things.

```
$ git add echobot.py
$ git add Procfile
$ git add runtime.txt
$ git add requirements.txt
$ git commit -am "Initial commit and deploy."
$ heroku buildpacks:set heroku/python
$ git push heroku master
$ heroku ps:scale worker=1
```

Sometimes there's a bug with heroku, and it will say '`gunicorn: command not found`'. to fix this run the following command and deploy again.

```
heroku config:set BUILDPACK_URL=https://github.com/heroku/heroku-buildpack-python
```

Success!

A Winner Is You!

Now you've got an echo bot running on your slack team. It will listen for direct messages, as well as on any channels you add it to.

These slides will be available on github. In addition, if you send a private message to tutorbot on the IIT Slack, it will give you a link to guide you through it.

I hope you've enjoyed this and will go forth and design interesting and helpful bots.