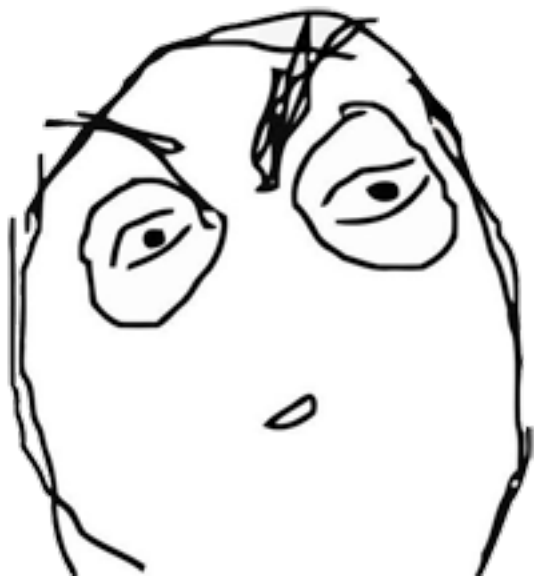


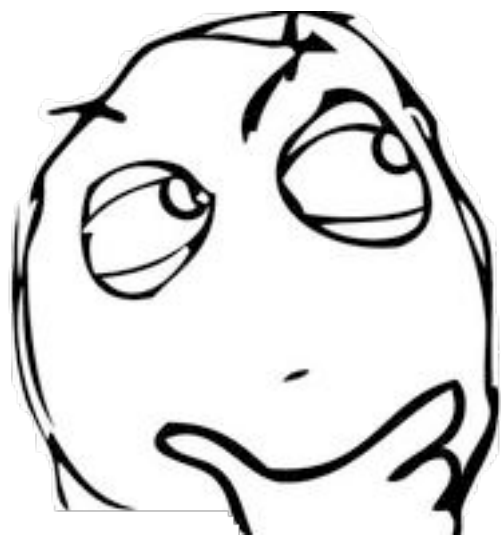
People seem to love exchanging
msgs anonymously...



Chatroulette™

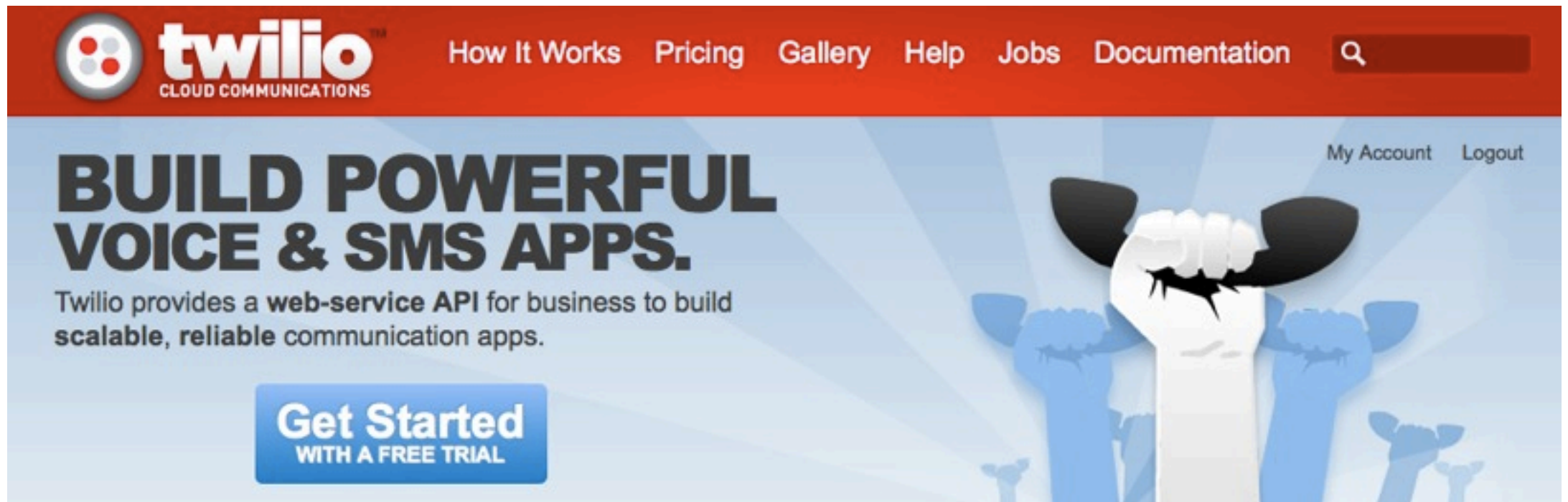
LikeALittle!





Hmmm, I want to learn by doing...

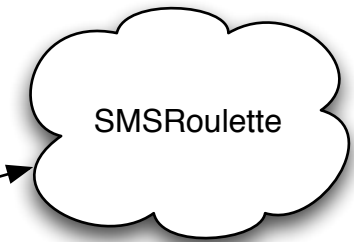
What can I do?



Perfect!
I'll build an anon SMS chat app
using Twilio!
called...
SMSRoulette

Here's the workflow illustrating the idea...

1. Users will SMS to our Twilio number to initiate random chat



2. SMSRoulette finds a waiting user and connects us with them

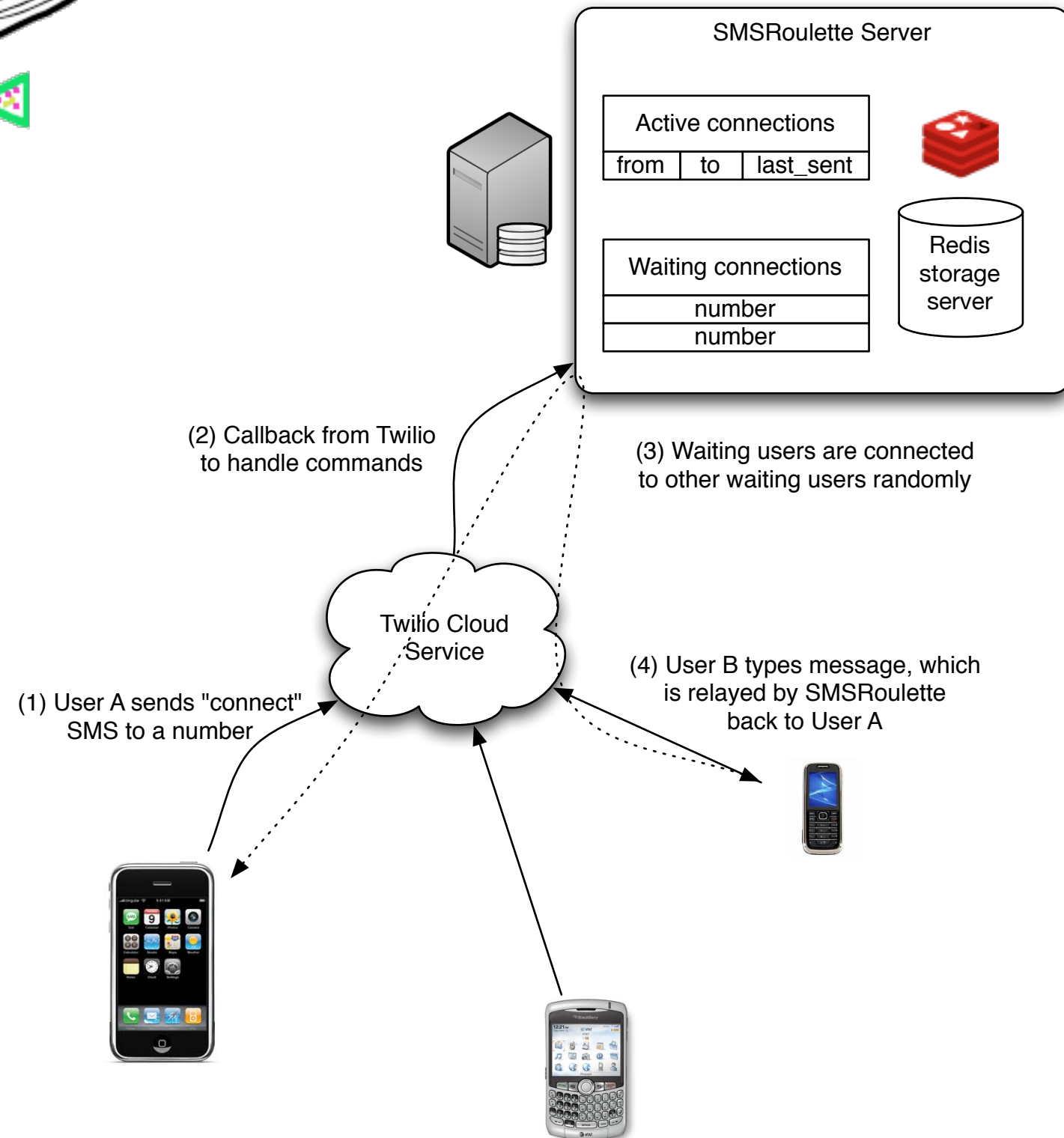


3. SMSRoulette relays messages to provide anonymity!





Block Diagram



Commands we should support

connect: registers phone number with SMSRoulette
disconnect: removes phone number
refresh: reconnect to another random user
call: share phone number with other user
stats: show server statistics

What we need to get started

Twilio: get free account at <http://twilio.com>

python: <http://python.org>

flask: <http://flask.pocoo.org/>
(for easy webapp development)

redis: <http://redis.io>

<https://github.com/andymccurdy/redis-py>
(our persistent data store)

editor: vim/emacs/whatever you like

Let's get started!

First, we register (or buy one) our phone number with Twilio that users will call.

For now, we will use our account's sandbox number.



Sandbox



Number

(415) 599-2671

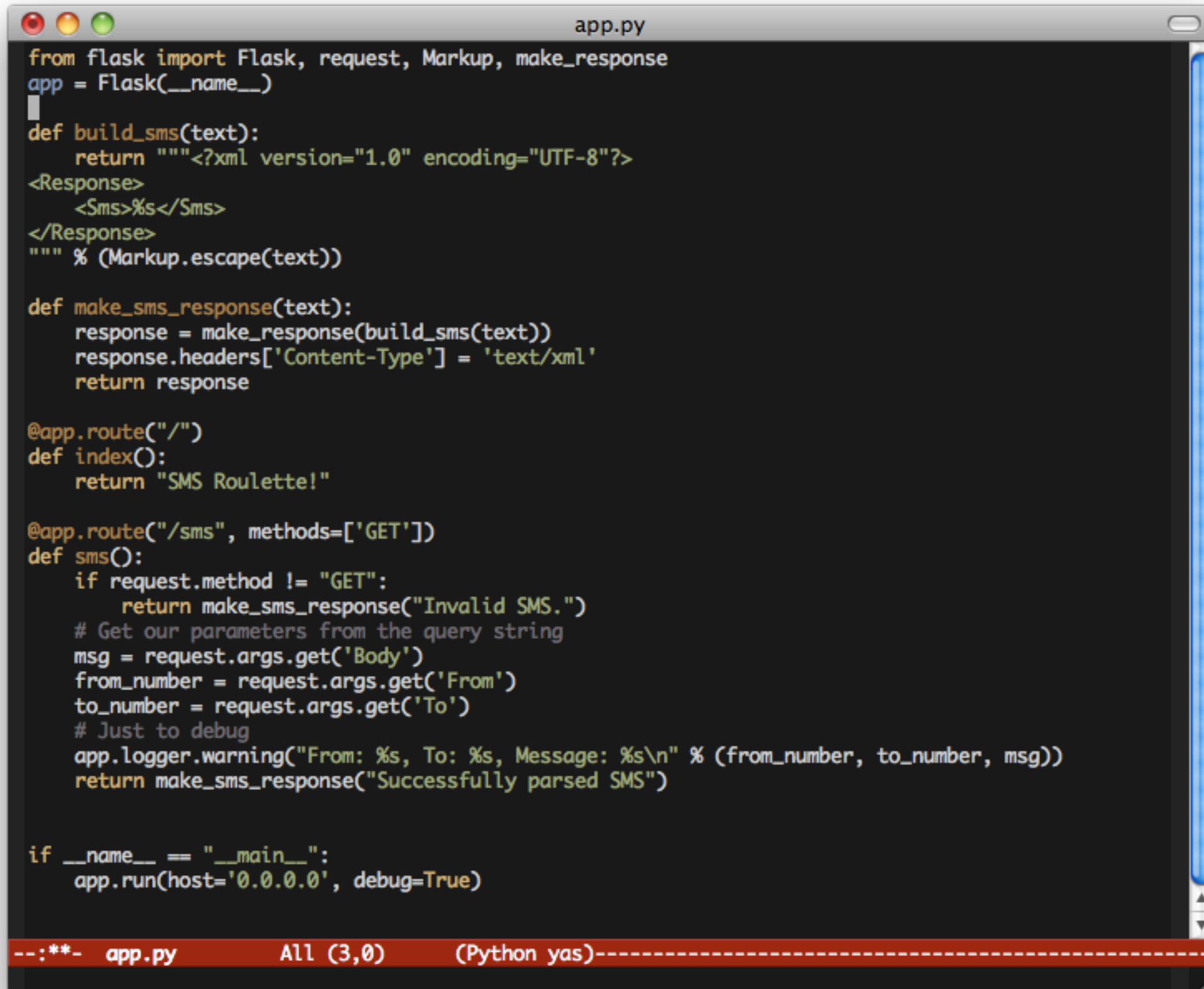
PIN

With sandbox number, you cannot send msgs to any phone #.

Register a few numbers as Caller IDs.

I used my mobile and Google Voice numbers.

Beginning our Flask app



```
from flask import Flask, request, Markup, make_response
app = Flask(__name__)

def build_sms(text):
    return """<?xml version="1.0" encoding="UTF-8"?>
<Response>
  <Sms>%s</Sms>
</Response>
""" % (Markup.escape(text))

def make_sms_response(text):
    response = make_response(build_sms(text))
    response.headers['Content-Type'] = 'text/xml'
    return response

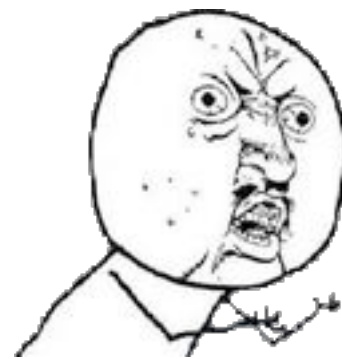
@app.route("/")
def index():
    return "SMS Roulette!"

@app.route("/sms", methods=['GET'])
def sms():
    if request.method != "GET":
        return make_sms_response("Invalid SMS.")
    # Get our parameters from the query string
    msg = request.args.get('Body')
    from_number = request.args.get('From')
    to_number = request.args.get('To')
    # Just to debug
    app.logger.warning("From: %s, To: %s, Message: %s\n" % (from_number, to_number, msg))
    return make_sms_response("Successfully parsed SMS")

if __name__ == "__main__":
    app.run(host='0.0.0.0', debug=True)
```

--:**- app.py All (3,0) (Python yas)-----

Y NO USE TEXT SLIDES?

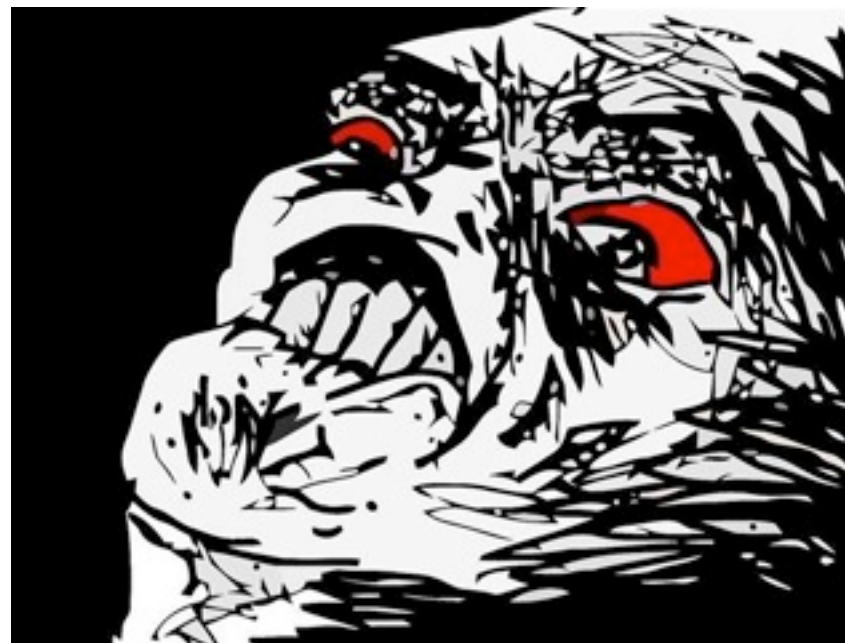




Type it in :)



Problem?



Alright: <https://gist.github.com/949919>

Let's test it out!

Configure the SMS URL at <https://www.twilio.com/user/account/>
to `http://<your webserver's IP>:5000`
Set HTTP Method to GET.



Send an SMS to your sandbox number.
Don't forget to prefix SMS with your PIN!

Our app should print this log

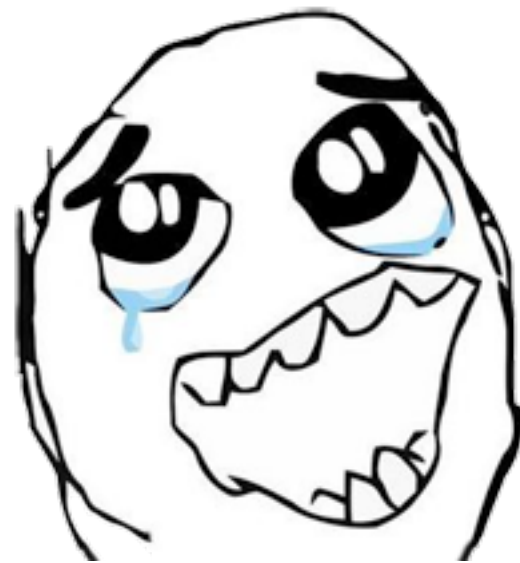
```
-/projects/twilio-contest =>python app.py
* Running on http://0.0.0.0:5000/
* Restarting with reloader...

-----
WARNING in app [app.py:33]:
From: +16503537086, To: +14155992671, Message: hi twilio, how are you?

-----
184.73.13.122 - - [30/Apr/2011 02:14:34] "GET /sms?AccountSid=ACe7b82a6a1be7e9144afa2fb03a2d0705&Body=hi+twilio%2C+
how+are+you%3F&ToZip=94949&FromState=CA&ToCity=NOVATO&SmsSid=SMb39db8b8dcfd4687d70233c7e3880011&ToState=CA&To=%2B14
155992671&ToCountry=US&FromCountry=US&SmsMessageSid=SMb39db8b8dcfd4687d70233c7e3880011&ApiVersion=2010-04-01&FromCi
ty=PALO+ALTO&SmsStatus=received&From=%2B16503537086&FromZip=94304 HTTP/1.1" 200 -
```

Your mobile should get “Successfully parsed SMS” message
prefixed with “Sent from a Twilio Trial Account -”

Something works!



Let's write a module for managing users (mobile numbers)



Pretty
straightforward

<https://gist.github.com/949918>

```
users.py
from flask import Markup
import redis
import sys

try:
    r = redis.Redis(host='localhost', port=6379, db=0)
except:
    print "Cannot connect to redis server. Exiting..."
    sys.exit(0)

def build_sms(text):
    return """<?xml version="1.0" encoding="UTF-8"?>
<Response>
  <Sms>%(text)s</Sms>
</Response>
""" % (Markup.escape(text))

def connect(number):
    """Add @number to the set of waiting callers."""
    r.sadd('waiting', number)
    ok = linkup()
    if not ok:
        return build_sms("Please wait, you will be connected to a random person...")
    return ""

def disconnect(number):
    """Remove @number from the set of waiting callers."""
    r.srem('waiting', number)
    peer = r.get(number)
    r.delete(number)
    if peer:
        r.delete(peer)
    return build_sms("You've been disconnected and will no longer receive messages.")

def refresh(number):
    """Refresh the chat session to connect to another random person."""
    disconnect(number)
    return connect(number)

def call(number):
    """@number wishes to share the number with the other person."""
    if not r.exists(number):
        return build_sms("Error: You haven't connected to anyone yet.")
    peer_number = r.get(number)
    send_sms(peer_number, "The other person has shared their number: %(number)s" % (number))
    return ""
```

```
users.py
if not r.exists(number):
    return build_sms("Error: You haven't connected to anyone yet.")
peer_number = r.get(number)
send_sms(peer_number, "The other person has shared their number: %(number)s" % (number))
return ""

def msg(number, text):
    """Message @number's peer"""
    peer = r.get(number)
    if peer is None:
        return (number, "Error: not connected, or your earlier peer disconnected. Refresh to chat again!")
    return (peer, "Stranger: " + text)

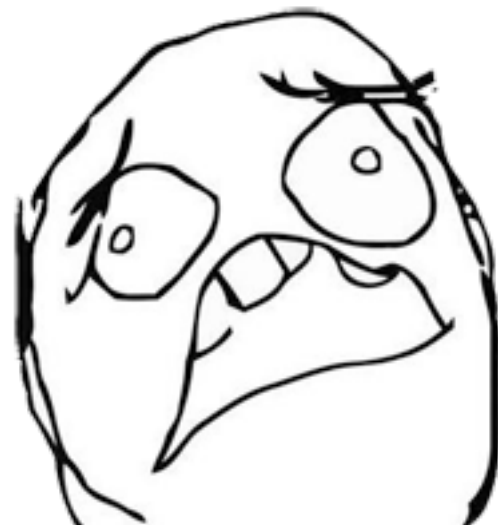
def stats(number):
    """Implement your favourite stats here. :-)"""
    pass

def linkup(number):
    pass
```

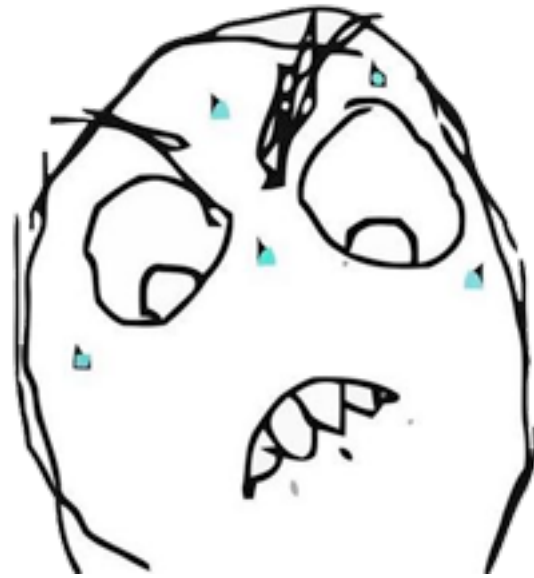


Fill in the logic for this yourself!

```
def linkup():  
    """Links up two random people from the waiting list."""  
    pass  
  
def stats(number):  
    """Implement your favourite stats here. :-)"""  
    pass
```



Done?



Linking up users

CHALLENGE ACCEPTED



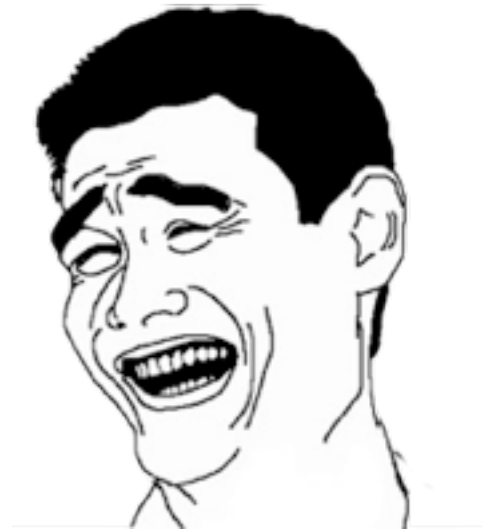
```
linkup

def linkup():
    """Links up two random people from the waiting list."""
    num_waiting = r.scard('waiting')
    if num_waiting < 2:
        # Cannot link up
        return False
    num1 = r.spop('waiting')
    num2 = r.spop('waiting')
    # our routing table
    r.set(num1, num2)
    r.set(num2, num1)
    text = "You're now connected to a random stranger!"
    send_sms(num1, text)
    send_sms(num2, text)
    return True

--:**- linkup      All (17,0)      (Python yas)-----
End of buffer
```

<https://gist.github.com/949923>

There are possible race conditions...



Good exercise to fix them

Integrating with the Flask app

```
integrate
CONNECT_SYNONYMS = ['connect', 'conn', 'login', 'register']
DISCONNECT_SYNONYMS = ['disconnect', 'dc', 'logout']
REFRESH_SYNONYMS = ['refresh', 'reload']
CALL_SYNONYMS = ['call', 'share']
STATS_SYNONYMS = ['stats']

@app.route("/sms", methods=['GET'])
def sms():
    if request.method != "GET":
        return make_sms_response("Invalid SMS.")

    # Get our parameters from the query string
    msg = request.args.get('Body')
    from_number = request.args.get('From')
    to_number = request.args.get('To')

    # Just to debug
    app.logger.warning("From: %s, To: %s, Message: %s\n" % (from_number, to_number, msg))
    cmd = msg.lower()
    if cmd in CONNECT_SYNONYMS:
        return users.connect(from_number)
    elif cmd in DISCONNECT_SYNONYMS:
        return users.disconnect(from_number)
    elif cmd in REFRESH_SYNONYMS:
        return users.refresh(from_number)
    elif cmd in CALL_SYNONYMS:
        return users.call(from_number)
    elif cmd in STATS_SYNONYMS:
        return users.stats(from_number)
    else:
        (peer_number, text) = users.msg(from_number, msg)
        send_sms(peer_number, text)
        # No response
        return ""

--:**- integrate All (6,0) (Python yas)-----
```



Being user friendly :)

```
CONNECT_SYNONYMS = ['connect', 'conn',
                    'login', 'register']
DISCONNECT_SYNONYMS = ['disconnect', 'dc',
                      'logout']
REFRESH_SYNONYMS = ['refresh', 'reload']
CALL_SYNONYMS = ['call', 'share']
STATS_SYNONYMS = ['stats']
```

<https://gist.github.com/949930>

Implementing send_sms(number, text)

```
API_VERSION = '2010-04-01'
# Check your account info
ACCOUNT_SID = 'AC...'
ACCOUNT_TOKEN = '...'
# Sandbox number or the number you have asked users to call
CALLER_ID = 'NNNNNNNNNN'

def send_sms(to, text):
    """We use Twilio's python libraries to send SMS. Taken directly
    from:
    https://github.com/twilio/twilio-python/blob/master/examples/example-rest.py"""
    global account

    d = {
        "From" : CALLER_ID,
        "To" : to,
        "Body" : text
    }

    try:
        account.request('/%s/Accounts/%s/SMS/Messages' % \
            (API_VERSION, ACCOUNT_SID), "POST", d)
    except Exception, e:
        open('error.log', 'a').write(e.read())

--:**- sendsms All (25,0) (Python yas)-----
```



<https://gist.github.com/949939>

Run it, it should work.

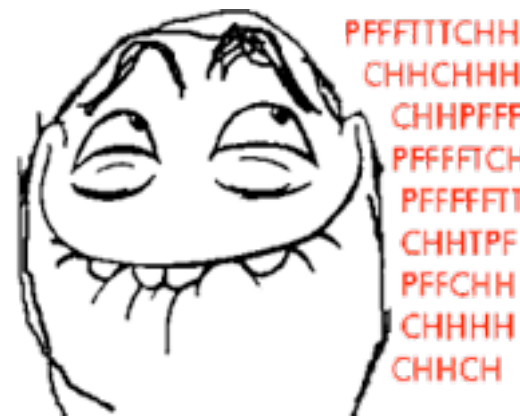


Twilio has a log...

... of all SMSs sent <https://www.twilio.com/user/account/log/sms>

... of failed requests <https://www.twilio.com/user/account/debugger>

It should help debugging... or

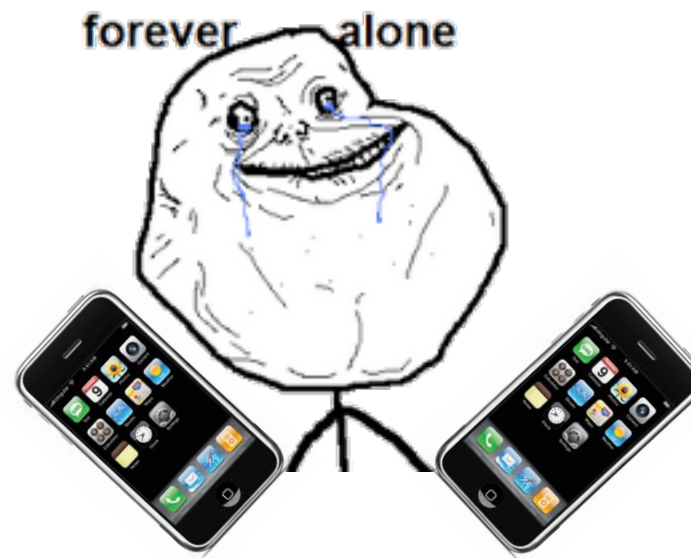


... you can watch people chat

Be responsible :)

Hope you could chat with someone...

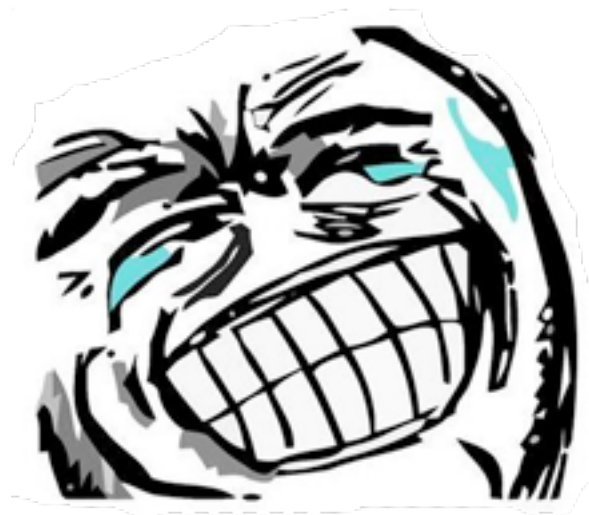
I couldn't.



Code available here:

<https://github.com/jvimal/SMSRoulette>

My first Twilio contest submission



Thanks for reading. Hope you enjoyed it!