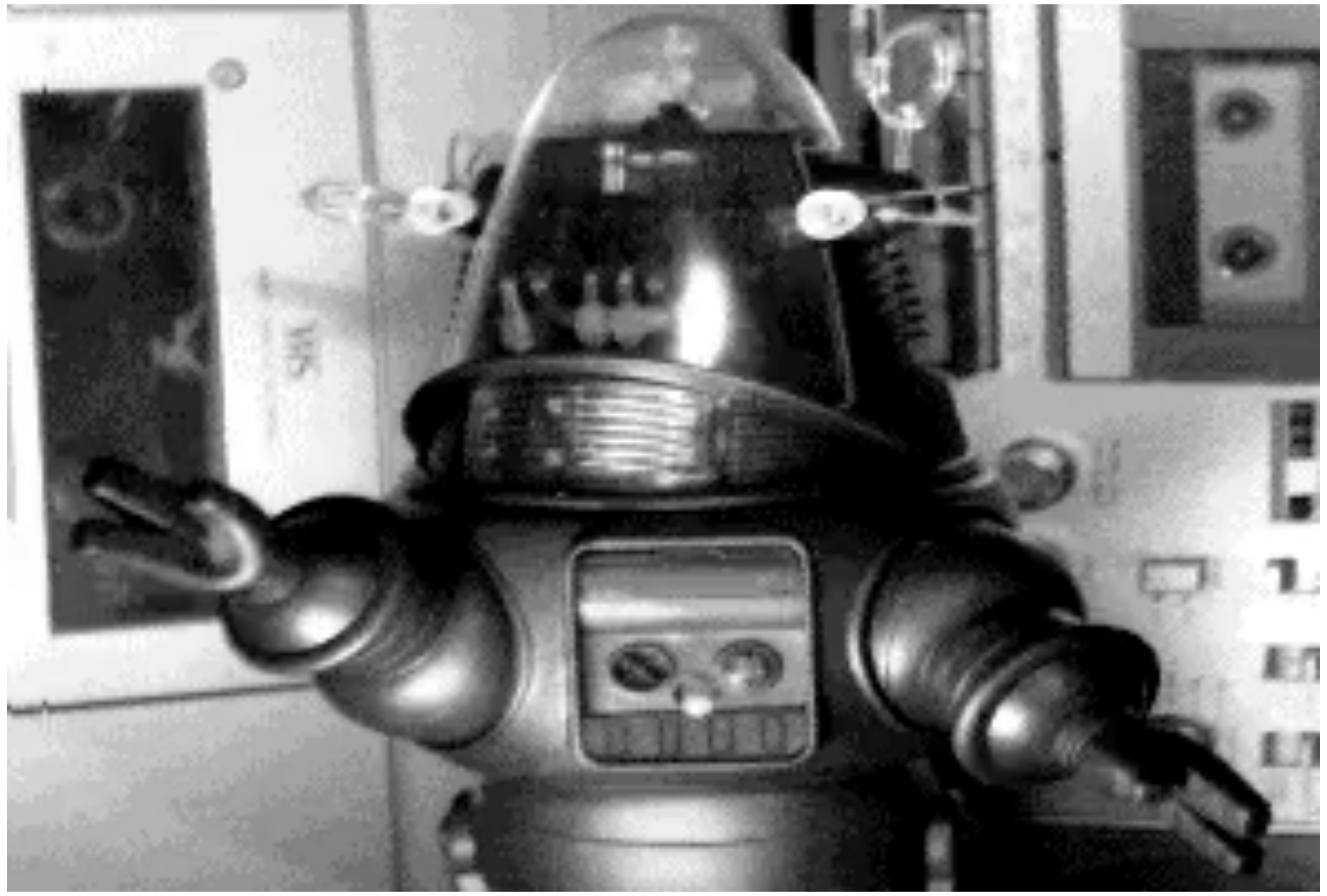
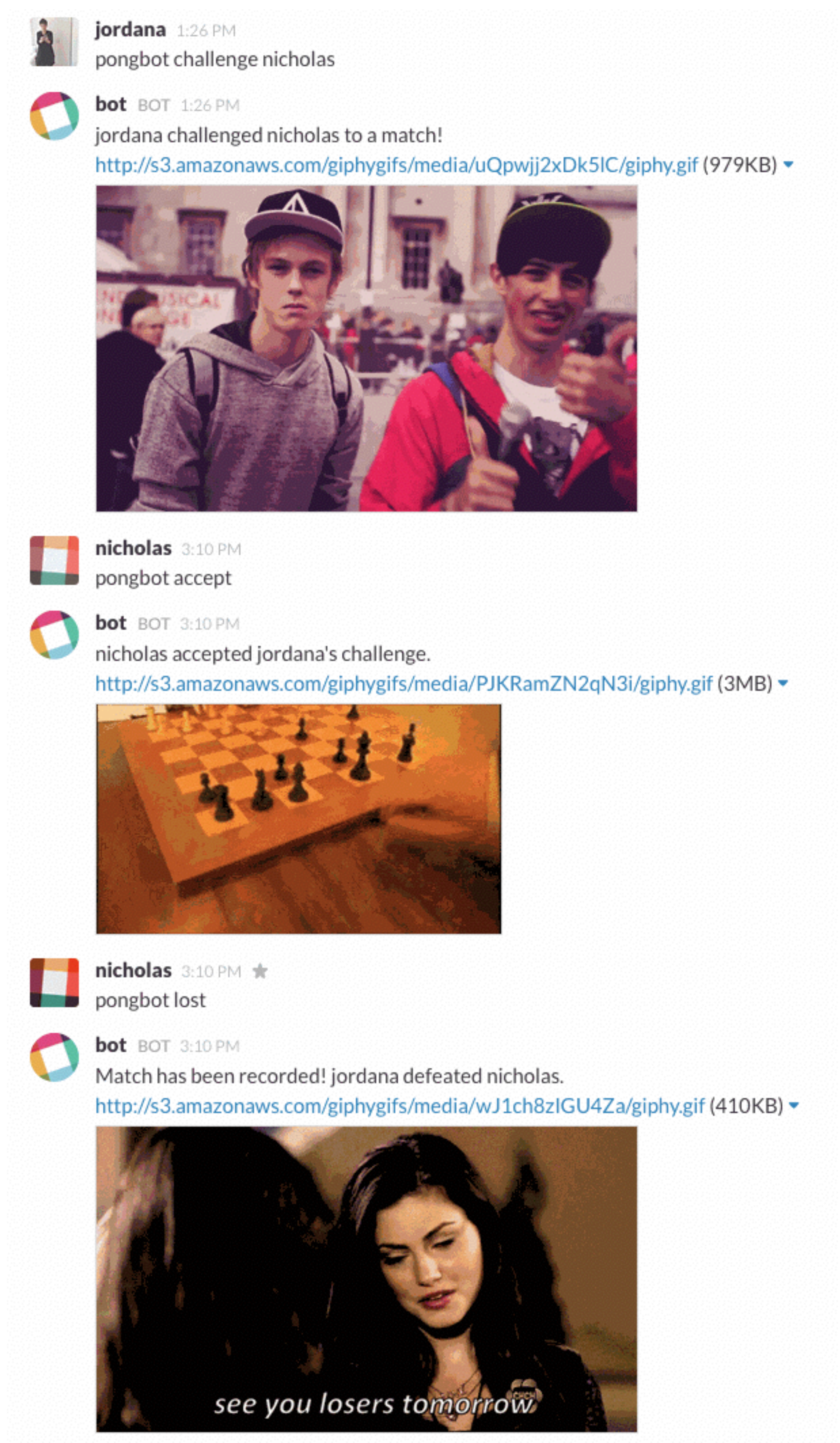


Your First Slack Bot

slack @dblockdotorg

http://www.meetup.com/NYC-rb/events/223744692



jordana 1:26 PM
pongbot challenge nicholas

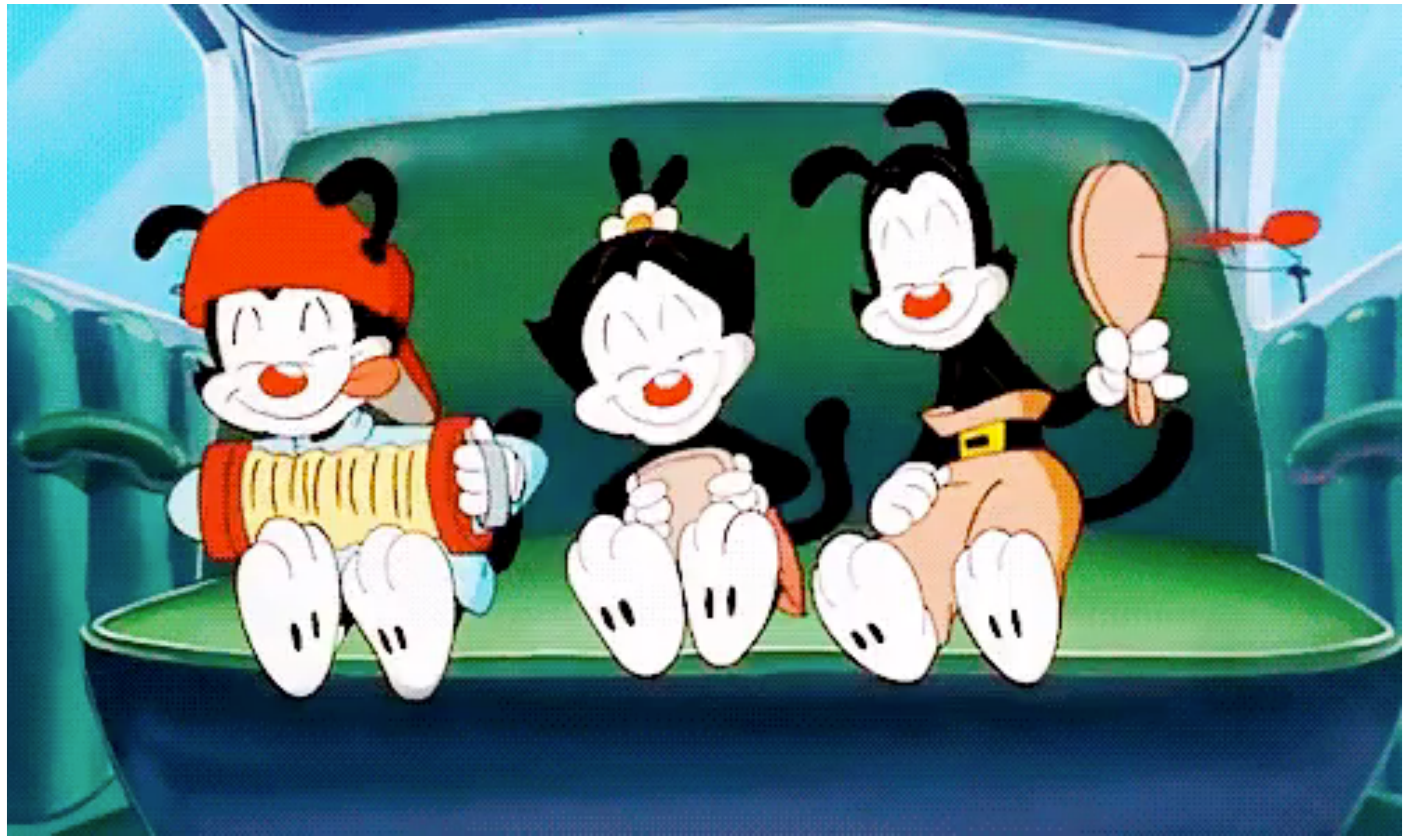
bot BOT 1:26 PM
jordana challenged nicholas to a match!
http://s3.amazonaws.com/giphygifs/media/uQpwjj2xDk5IC/giphy.gif (979KB)

nicholas 3:10 PM
pongbot accept

bot BOT 3:10 PM
nicholas accepted jordana's challenge.
http://s3.amazonaws.com/giphygifs/media/PJKRamZN2qN3i/giphy.gif (3MB)

nicholas 3:10 PM
pongbot lost

bot BOT 3:10 PM
Match has been recorded! jordana defeated nicholas.
http://s3.amazonaws.com/giphygifs/media/wJ1ch8zIGU4Za/giphy.gif (410KB)



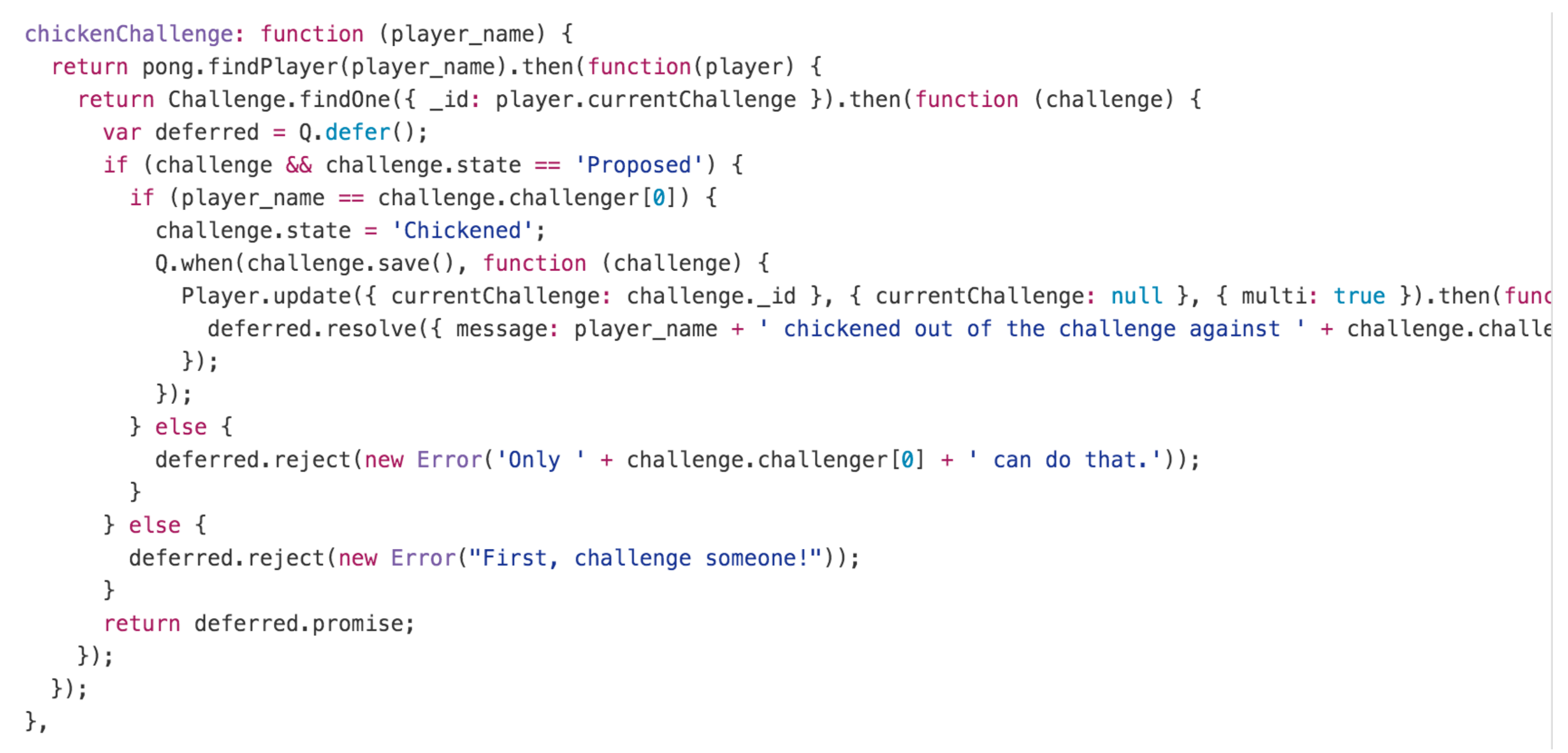
Web API
Real Time Messaging API
Slack Ruby Client
Slack Ruby Bot

#CODE

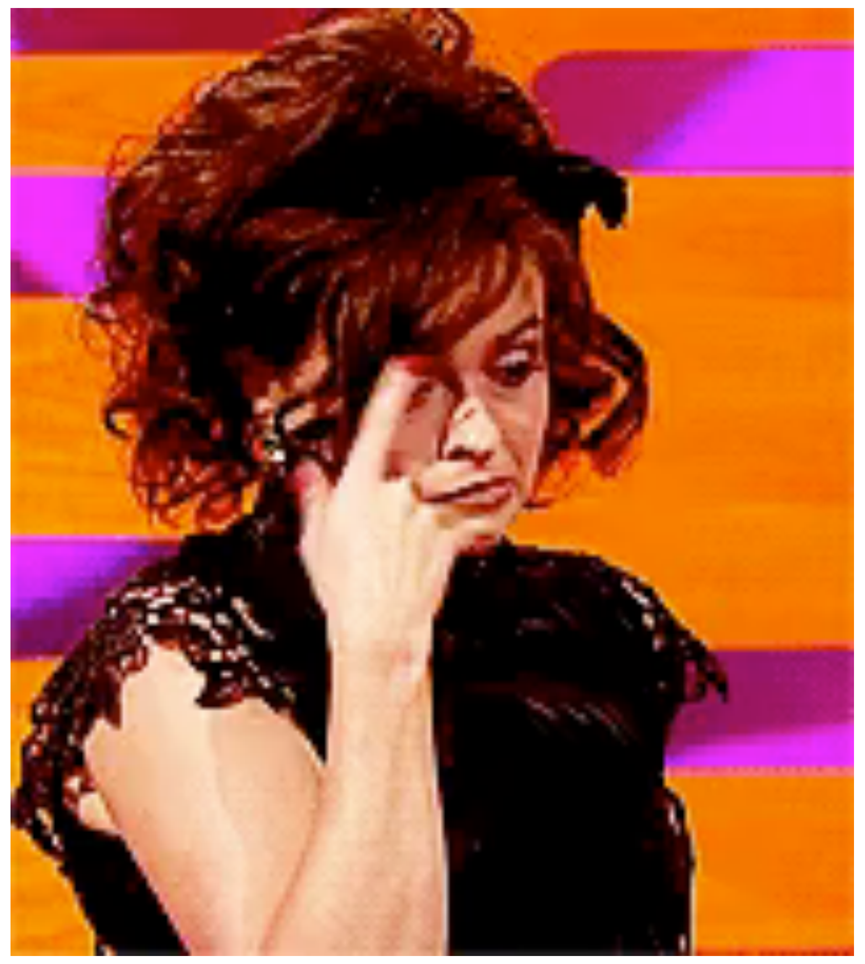
http://www.slackbotlist.com

github.com/dblock/slack-gamebot
github.com/dblock/slack-mathbot
github.com/dblock/slack-aws

```
chickenChallenge: function (player_name) {
  return pong.findPlayer(player_name).then(function(player) {
    return Challenge.findOne({ _id: player.currentChallenge }).then(function (challenge) {
      var deferred = Q.defer();
      if (challenge && challenge.state == 'Proposed') {
        if (player_name == challenge.challenger[0]) {
          challenge.state = 'Chickened';
          Q.when(challenge.save(), function (challenge) {
            Player.update({ currentChallenge: challenge._id }, { currentChallenge: null }, { multi: true }).then(func
              deferred.resolve({ message: player_name + ' chickened out of the challenge against ' + challenge.challe
            });
          });
        } else {
          deferred.reject(new Error('Only ' + challenge.challenger[0] + ' can do that.'));
        }
      } else {
        deferred.reject(new Error("First, challenge someone!"));
      }
      return deferred.promise;
    });
  });
},
```



```
chickenChallenge: function (player_name) {
  return pong.findPlayer(player_name).then(function(player) {
    return Challenge.findOne({ _id: player.currentChallenge }).then(function (challenge) {
      var deferred = Q.defer();
      if (challenge && challenge.state == 'Proposed') {
        if (player_name == challenge.challenger[0]) {
          challenge.state = 'Chickened';
          Q.when(challenge.save(), function (challenge) {
            Player.update({ currentChallenge: challenge._id }, { currentChallenge: null }, { multi: true }).then(func
              deferred.resolve({ message: player_name + ' chickened out of the challenge against ' + challenge.challe
            });
          });
        } else {
          deferred.reject(new Error('Only ' + challenge.challenger[0] + ' can do that.'));
        }
      } else {
        deferred.reject(new Error("First, challenge someone!"));
      }
      return deferred.promise;
    });
  });
},
```



Q?
deferred?
reject?
accept?

github.com/dblock/slack-ruby-client

Slack::Web::Client
Slack::RealTime::Client

github.com/dblock/slack-ruby-bot

SlackRubyBot::App

```ruby
class Phone < SlackRubyBot::Commands::Base
  command 'call', '呼び出し' do |client, data, _match|
    send_message client, data.channel, 'called'
  end
end
```

```ruby
class Calculator < SlackRubyBot::Commands::Base
  operator '=' do |_data, _match|
    # implementation detail
  end
end
```

```ruby
class Weather < SlackRubyBot::Commands::Base
  match /^How is the weather in (?<location>\w*)\?$/ do |client, data, match|
    send_message client, data.channel, "The weather in #{match[:location]} is nice."
  end
end
```

```ruby
require 'spec_helper'

describe SlackGamebot::Commands::Accept, vcr: { cassette_name: 'user_info' } do
  let(:app) { SlackGamebot::App.new }
  let(:challenged) { Fabricate(:user, user_name: 'username') }
  let!(:challenge) { Fabricate(:challenge, challenged: [challenged]) }
  it 'accepts a challenge' do
    expect(message: "#{SlackRubyBot.config.user} accept", user: challenged.user_id, channel: challenge.channel).to
      "#{challenge.challenged.map(&:user_name).join(' and ')} accepted #{challenge.challengers.map(&:user_name).jo
    )
    expect(challenge.reload.state).to eq ChallengeState::ACCEPTED
  end
end
```

slack-ruby-client
slack-ruby-bot

@dblockdotorg

https://github.com/dblock/your-first-slack-ruby-bot-in-ruby