Interacting with the Outside World through Python

Part 2: Talking to People

Yes! Now I can have meaningful communication with people in the outside world.

authors: C. Klein, J. Bloom

Outline

1) Email
   - smtplib, email, poplib, imaplib, rfc822
2) Phone/SMS
3) Hardware
   - pySerial demonstration
4) Audio recording and analysis
   - pyaudio, wave, aifc

smtplib & email

Super simple email sending implementation

```
>>> import smtplib # Simple Mail Transfer Protocol
>>> from email.MIMEMultipart import MIMEMultipart
>>> from email.MIMEText import MIMEText
>>> msg = MIMEMultipart()
>>> msg["From"] = "sender@gmail.com"
>>> msg["To"] = "recipient@gmail.com"
>>> msg.attach(MIMEText("The actual email text."))
>>> # designate that we are using gmail's remote smtp server
>>> mailServer = smtplib.SMTP("smtp.gmail.com", 587)
>>> # gmail requires TLS authentication on port 587
>>> mailServer.starttls()
>>> mailServer.login("sender@gmail.com", "password")
>>> mailServer.sendmail("sender@gmail.com",
    "recipient@gmail.com", msg.as_string())
>>> mailServer.close()
```

Email Method

```python
import smtplib, os
from email.MIMEMultipart import MIMEMultipart
from email.MIMEBase import MIMEBase
from email.MIMEText import MIMEText
from email import Encoders
from email.Utils import COMMASPACE, formatdate

def mail(sender, pwd, to, subject, text, files=[]):
    msg = MIMEMultipart()
    msg["From"] = sender
    msg["To"] = COMMASPACE.join(to)
    msg["Date"] = formatdate(localtime=True)
    msg["Subject"] = subject
    msg.attach(MIMEText(text))
    for file in files:
        part = MIMEBase("application", "octet-stream")
        part.set_payload( open(file,"rb").read() )
        Encoders.encode_base64(part)
        part.add_header("Content-Disposition", "attachment; filename='%s'"
                       % os.path.basename(file))
        msg.attach(part)
    # designate the remote SMTP server
    mailServer = smtplib.SMTP("smtp.gmail.com", 587)
    mailServer.starttls()
    mailServer.login(sender, pwd)
    mailServer.sendmail(sender, to, msg.as_string())
    mailServer.close()

mail(
    sender="sender@gmail.com",
    pwd="password",
    to=["recipient@gmail.com",],   # include an extra comma in the "to" list to
                                   # account for the COMMASPACE.join(to)
    subject="Email from Python",
    text="Whoooo!\n",
    files=["email_example.py"] # list of files to attach
    )
```

poplib & rfc822

Super simple email retrieval

```
>>> import poplib, string, StringIO, rfc822
>>> server = poplib.POP3_SSL("pop.gmail.com", 995) # connect to server
>>> server.user("username") # login with username
>>> server.pass_("password") # login with password
>>> resp, items, octets = server.list() # list unread messages on server
>>> for n in range(len(items)): # loop through unread messages
>>>     resp, text, octets = server.retr(n + 1)
>>>     text = string.join(text, "\n")
>>>     file = StringIO.StringIO(text)
>>>     message = rfc822.Message(file) # parse the email message
>>>     for name, value in message.items():
>>>         print name, "=", value # print message header info
>>>     print message.fp.read() # print the message text
>>> server.quit()
```

- POP (Post Office Protocol) retrieves only unread messages

imaplib & rfc822

Super simple email retrieval

```
>>> import imaplib, string, StringIO, rfc822
>>> server = imaplib.IMAP4_SSL("imap.gmail.com", 993)
>>> server.login("username", "password")
>>> server.select() # select a mailbox
>>> resp, items = server.search(None, "ALL") # list messages on server
>>> items = string.split(items[0]).reverse # reverse item numbers
>>> for id in items: # loop through all messages
>>>     resp, data = server.fetch(id, "(RFC822)")
>>>     text = data[0][1]
>>>     file = StringIO.StringIO(text)
>>>     message = rfc822.Message(file) # parse email message
>>>     for name, value in message.items():
>>>         print name, "=", value # print message header info
>>>     print message.fp.read() # print the message text
>>> server.logout()
```

- IMAP (Internet Message Access Protocol) retrieves all messages on server

Python for Phone/Texting

use case: *your gene sequencer in the lab just barfed at 2am and you have an important deliverable due tomorrow*

```
try:
   sequence_this(“mouse”)
except:
  call_the_grad_student_in_charge()
```

```
pip install twilio
```

To the Notebook!

powered by twilio™

https://twilio-python.readthedocs.org/en/latest/

Hardware Communications

1) USB
    - PyUSB
2) Serial
    - pySerial
3) Parallel
    - pyParallel
4) Bluetooth
    - LightBlue, PyBluez

Compatibility is highly OS-dependent

pySerial Demonstration

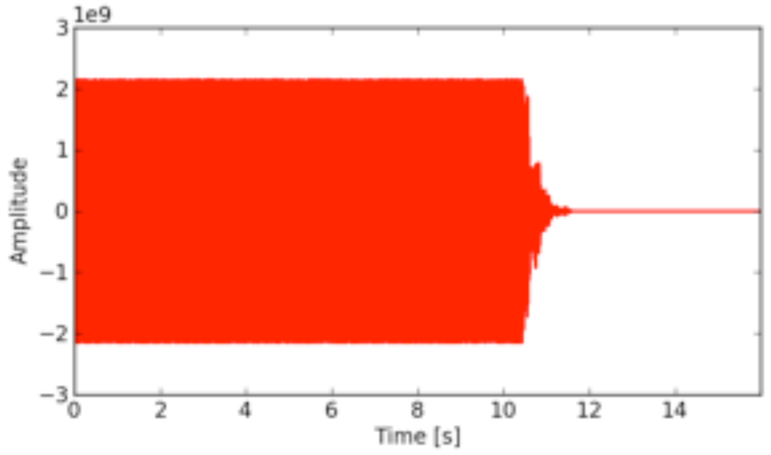
To the Terminal!

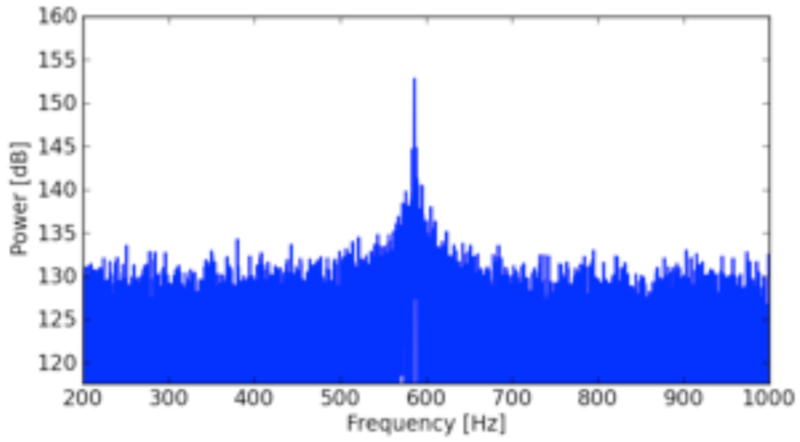
Audio Analysis

pyaudio - for reading from line in

```
brew install portaudio
```

wave, aifc - for reading/writing .wav and .aif files





D5

pyaudio

Record audio, write to a .wav file, plot the waveform.

To the Notebook!

If it seems useful, Google it

If you want to extend Python in a novel way, but haven't coded it up already, check the internet first.

There are open-source Python modules for managing many high-level interactions, sometimes with very specific applications.

python-twitter
arxiv.py
PyFacebook
Universal Feed Parser (RSS)
libgmail
python-linkedin
gdata-python-client (Google data APIs)
PyMedia (audio/video)
Py-TOC (AIM)