Programming and Data Science for Biology (PDSB)

Session 8
Spring 2018

Today’s topics

1. Review of last assignment
2. More on project proposals
3. Introduction to RESTful APIs and Python web scraping
4. Hands-on programming

Upcoming topics relate to projects

1. **Accessing data from the web**: REST APIs
2. **data analysis:** scipy, pymc3, scikit-learn.
3. **Plotting data:** matplotlib, toyplot, folium

Python Classes: How are we doing?

Python Classes

Object oriented programming.

+ **Store data in the class instance and initiate all attributes in __init__.**
+ **Write many small private functions to accomplish small tasks.**
+ **Write few public functions for user to access.**

```
## a simple class with an init function
class Simple:
      def __init__(self, data):
            self.data = data
            self.x = None
            self.y = None

      # private functions
      def _func1(self):
            ...
      def _func2(self):
            ...

      # public function
      def run(self):
            self.x = self._func1()
            self.y = self._func2()
```

Package design

Installing so the package is importable (setup.py)

project/
- setup.py
- package/
  - __init__.py
  - seqlib.py

A __main__.py file, and accompanying ‘console_script’ arguments in setup.py, like we used for our helloworld example, are only necessary if we plan to make a CLI. In that case we would use argparse in the main file to parse command line arguments.

```
#!/usr/bin/env python

# from ./seqlib.py import Seqlib
from .seqlib import Seqlib

-----------------------------------

# from the directory with setup.py
cd project/
pip install .
```

Secure Shell (SSH): secure remote login. Cluster computing. Linux.

Remote access by secure shell

Stores a public/private key pair. By default the cluster server will authenticate your key using your UNI password.

```
## format of ssh login command
ssh <user>@<hostname>

## our example
ssh de2356@habanero.rcs.columbia.edu
```

The Habanero cluster at Columbia

Node structure (and reserved nodes)

+ 5,328 CPUs (222 nodes of 24 cores)
+ 128Gb RAM (176 nodes) or 512 (32 node)
+ 1 Tb disk storage

```
## A server has an IP address
232.195.43.2.2

## And can also have a hostname
habanero.rcs.columbia.edu
```

subprocess

In: `Notebooks/7.2-subprocess.ipynb`

So far, we've been writing code to accomplish some task. But often, *there is already some program written to accomplish that task.*

```
## e.g., external genomics programs
bwa
abyss
vsearch
bowtie
muscle
samtools
```

stdin, stdout, stderr

Calls to subprocess return separate outputs from stderr and stdout.

```
## Standard subprocess call
import subprocess as sps

## tell is where to write stdout (here to PIPE)
proc = sps.Popen([‘ls’, ‘-l’], stdout=sps.PIPE)

## PIPE connects stdout to the proc object
print(proc.stdout.decode())
```

Project proposals!

Project Guide and Instructions

Course projects are meant to provide an exercise in which you explore and learn to use new coding tools on your own that we have not yet covered in class. The exact format is flexible and the finished product could encompass a number of different things. Below I list the rules, restrictions, and recommendations for project proposals as well as several example ideas.

Format/Deadlines

Proposal: one page essay and create project GitHub repo -- *due 3/21*.
Presentation: presentation of proposal and progress so far (10 minutes with questions) -- *4/23 and 4/30*.
Final project: GitHub repo with code, data, example notebooks, and final one page essay -- *due 5/11*.

Grades:

As stated in the course syllabus, the project proposal, presentation and final form make up a significant portion of your total course grade:

- **Proposal:** 5%
- **Presentation:** 5%
- **Final project:** 15%

Proposal Essay Instructions:

Proposals should be approximately one page single-spaced (including references) answering the following five questions.

1. **The problem**: What is the problem you wish to solve?
2. **The data**: What is the data you will analyze and where is it from?
3. **The tools**: What computational tools will you use to solve the task?
4. **The novelty**: Why is this novel?
5. **The goal**: What is your envisioned product at the due date and after?

Proposal Details:

1. **The problem:** Choose a problem that is of interest to you and is appropriately difficult given the timescale of the project (~7 weeks). This could be to write a program or library that can accomplish some repetitive task, or to learn to use an existing library for statistically analyzing a particular type of data (though this must be sufficiently complex or combined with further programming), or to develop a tool for visually representing data or results that is novel. See the *ideas* section at the end of this file.

2. **The data:** Part of this exercise is to find appropriate data to answer the question you are interested in. This could be data that you have collected as a graduate student. It could be data that is available from a professor that you know. It could be data that you generate yourself. Or, for most people, it will likely be data that you find in an online database, e.g., from a published study.

3. **The tools:** Why did you choose to use these tools? What other software tools are likely to be used in your project? Is your method a reimplementation of an existing method, is it a pipeline?

4. **The novelty:** In what way does your *problem* + *data* + *tools* = *novel*? If you are not developing a new tool then your implementation should be novel (e.g., no one has previously applied this type of method to this data set), and sufficiently difficult to implement.

5. **The goal:** Describe your envisioned final product, how can people use it? You can list a short-term and theoretical long-term goal, for example, by the due date I plan to have a minimal working example that can accomplish x, with more time and effort my tool/method/example could also do y and z.

GitHub repository structure:

Create a new repository for your project and name it whatever you like (you can change the name later if you want as well). When you create the repo on GitHub select to create a README file and also add a LICENSE file and select GPLv3 as the license type. Clone your repository and add the following four directories to it, which will make up the structure of your project. Your proposal and final essays can be markdown README files or docx files, whichever you prefer, and should be uploaded to the `documents/` directory by their due dates. See instructions below for the contents of the other directories.

```
PDSB-project/
    + notebooks/
    + data/
    + project/
    + documents/
    + README.md
    + LICENSE
```

- `notebooks/` : This directory must contain at least one jupyter notebook that demonstrates your code on your test data set as an example implementation. This notebook should be nicely formatted and provide a formal tutorial that anyone else in class could run to reproduce your results and understand.
- `data/` : This directory should contain your data file unless your data are downloaded directly from the internet.
- `project/` : The name of this directory can be whatever you want, it is the place where you will store your code.
- `documents/` : This directory is where you will upload your proposal and final project essays.
- `README.md` : This README file should include (1) the name of your project, (2) a short description of what it does, (3) installation instructions, and (4) instructions to find examples in your notebooks directory.

Example project

We'll be walking through creating a project together over the next three weeks, the first example project listed in the proposal guide.

Problem: analyze bee diversity through time, can we detect a shift in observations?
Data: GBIF occurrence records
Tools: requests, pandas, pymc3, folium, toyplot.
Novelty: combining tools to fetch data, measure diversity in time slices, detect shifts, and make plots in a single tool.
Product: A python library

```
## import library and create class instance
import records
rec = records.Records(
      query="Bombus",
      interval=(1900,2000),
      chunks=10)

## access data as a dataframe and do analyses
rec.df.shape
rec.fit_breakpoint_model()

## plot data and results
rec.plot_diversity()
rec.plot_map()
rec.plot_model_fit()
```

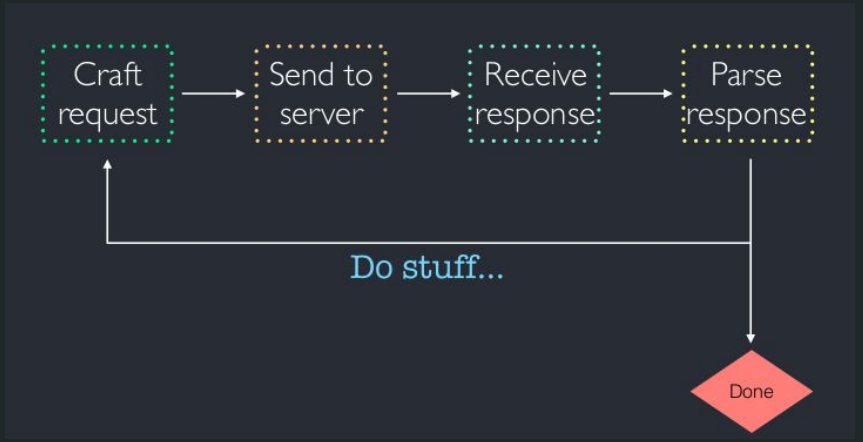
REST APIs: getting data from the web the nice way

REST API

Representational State Transfer API:

Simply means that a website was written following a *certain convention* that allows users to access data from it in a programmatic, or generic way.

API often has a different address:
Normal server address: http://gbif.org
API server address: http://api.gbif.org



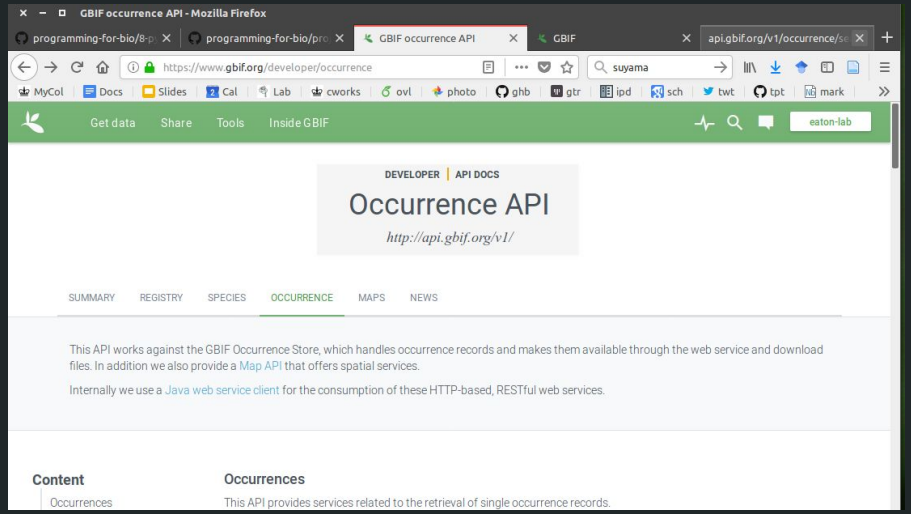
source: https://github.com/biolprogramming/

REST API

What is the API convention? Well, it often varies among sites, so it is always best to read the documentation. But it will be to format a URL to request specific information.

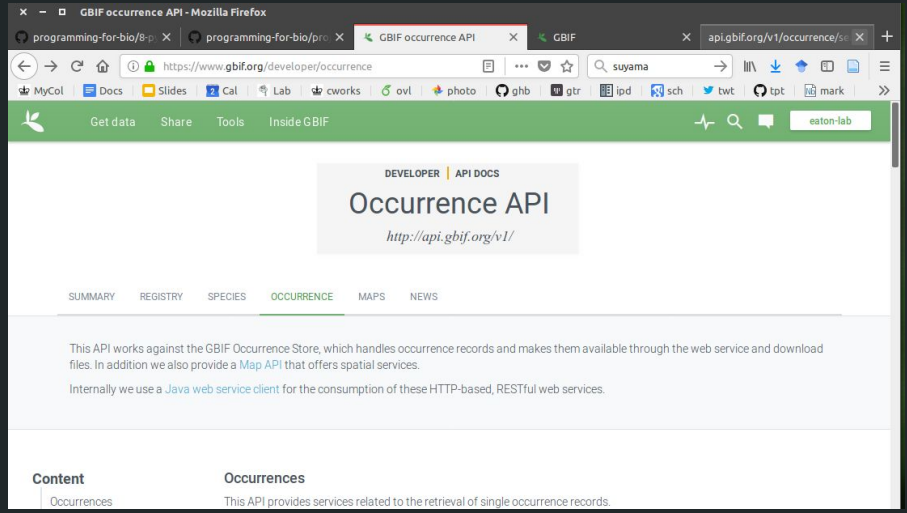
Base URL: http://api.gbif.org/v1/occurrence/search

URL search: http://api.gbif.org/v1/occurrence/search?q=Bombus



Many RESTful APIs

+ Twitter
+ GitHub
+ Reddit
+ Pubmed: publications
+ GBIF: occurrence records
+ Bison: North America occurrence records
+ Propublica: Non profit $ information
+ CityBikes: NYC bike location & usage
+ Biodiversity heritage library: old science lib
+ MG-Rast: microbial data
+ USDA soil data



baseurl = “http://api.gbif.org/v1/occurrence/search?”
search = “q=ochotona”
full = http://api.gbif.org/v1/occurrence/search?q=ochotona

+ requests are made with key=value pairs.

```
import requests
response = requests.get(
        url=baseurl,
        params={‘q’: ‘ochotona’})
```

Requests: HTTP for Humans — Requests 2.18.4 documentation - Mozilla Firefox

Requests: HTTP for Humans

docs.python-requests.org/en/master/

Search

MyCol | Docs | Slides | Cal | Lab | cworks | ovl | photo | ghb | gtr | ipd | sch | twt | tpt | mark

Requests 3.0 development is underway, and your financial help is appreciated!

Fork me on GitHub

Requests
http for humans

Star 31,186

Requests is an elegant and simple HTTP library for Python, built for human beings.

Sponsored by **Linode** and other wonderful organizations.

See how your visitors are

Requests: HTTP for Humans ¶

Release v2.18.4. (Installation)

license Apache 2.0 | wheel yes | python 2.6, 2.7, 3.4, 3.5, 3.6 | codecov 90% | Say Thanks!

Requests is the only *Non-GMO* HTTP library for Python, safe for human consumption.

> **Note:**
>
> The use of **Python 3** is *highly* preferred over Python 2. Consider upgrading your applications and infrastructure if you find yourself *still* using Python 2 in production today. If you are using Python 3, congratulations — you are indeed a person of excellent taste.
> —*Kenneth Reitz*

Behold, the power of Requests:

```
>>> r = requests.get('https://api.github.com/user', auth=('user', 'pass'))
>>> r.status_code
200
>>> r.headers['content-type']
'application/json; charset=utf8'
>>> r.encoding
'utf-8'
>>> r.text
u'{"type":"User"...'
```

Requests

+ Requests is built for querying REST APIs with a very clean interface
+ Returns a ‘Response’ Class instance, from which you can get information about the request, and the data.
+ Notebook/nb-8.1-restful-gbif.ipynb

```
## import library and make request
import requests
response = requests.get(
     url=basename,
     params=querydict)

## was request successful?
response.status_code()

## get data as text or json (better yet)
response.text
response.json()
```

Parsing HTML: getting data from the web the hard way

Beautiful Soup Documentation

Beautiful Soup is a Python library for pulling data out of HTML and XML files. It works with your favorite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work.

These instructions illustrate all major features of Beautiful Soup 4, with examples. I show you what the library is good for, how it works, how to use it, how to make it do what you want, and what to do when it violates your expectations.

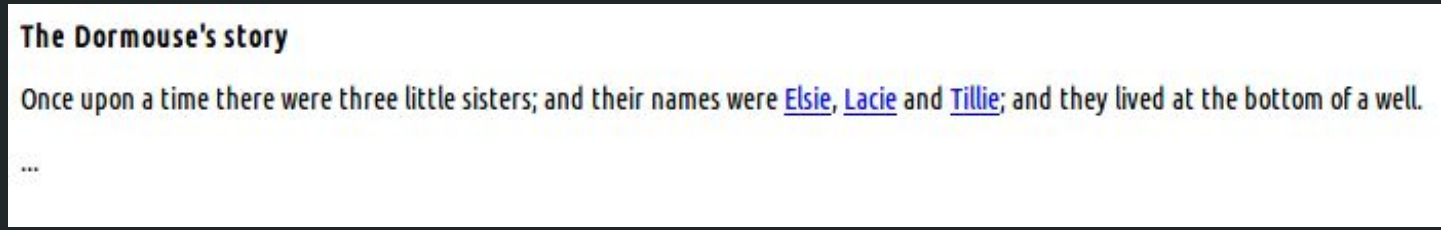
The examples in this documentation should work the same way in Python 2.7 and Python 3.2.

HTML

```
html_doc = """
<html><head><title>The Dormouse's story</title></head>
<body>
<p class="title"><b>The Dormouse's story</b></p>

<p class="story">Once upon a time there were three little sisters; and their names were
<a href="http://example.com/elsie" class="sister" id="link1">Elsie</a>,
<a href="http://example.com/lacie" class="sister" id="link2">Lacie</a> and
<a href="http://example.com/tillie" class="sister" id="link3">Tillie</a>;
and they lived at the bottom of a well.</p>

<p class="story">...</p>
"""
```



BeautifulSoup

+ Get HTML text from requests
+ Parse HTML text with bs4
+ Use Inspect in a web browser to find the names of elements you are searching for.
+ Notebooks/nb-8.2-html-soup.ipynb

```
## import library and make soup
import requests
from bs4 import BeautifulSoup

res = requests.get(“google.com”)
soup = BeautifulSoup(res,‘html5lib’)

## find elements in the HTML text
soup.find_all(find_all("div,{"id", “lga”}))

## get data as text or json (better yet)
response.text
response.json()
```

Assignments and readings

Project proposals are due by end of Wed.

Assignment is due Friday at 5pm

Self code fixup is due Monday by class.

Assignment: Link to Session 8 repo

Readings: See syllabus

Collaborate: Work together in this gitter chatroom