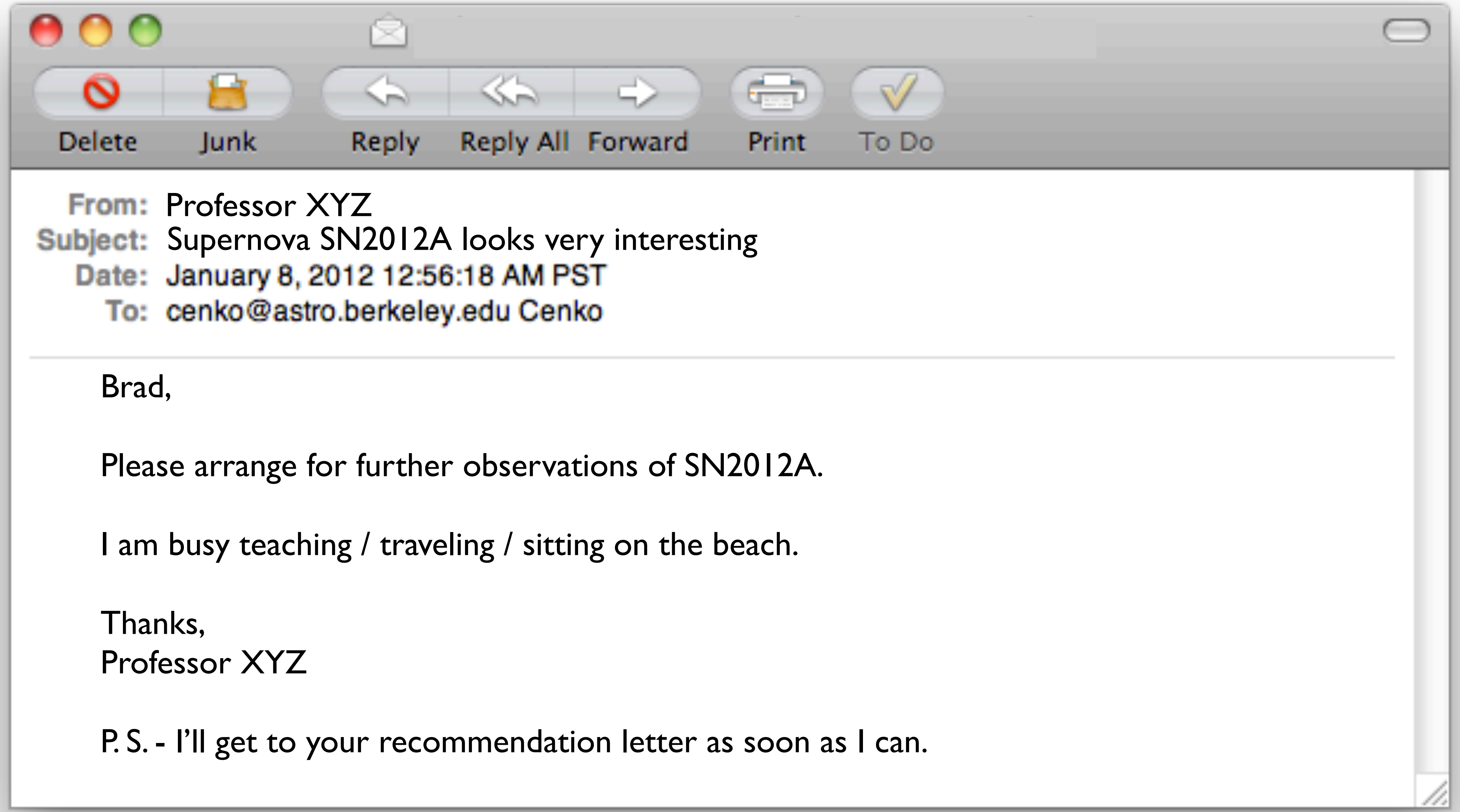
Whetting Your Appetite:

Interacting with the Outside World

Brief Outline

- Interweb (urllib, urllib2, etree)
- Relational Databases (sqlite3)
- Email (email, smtplib)
- Science (at least my version of it)

How Science Gets Done: I



From: Professor XYZ
Subject: Supernova SN2012A looks very interesting
Date: January 8, 2012 12:56:18 AM PST
To: cenko@astro.berkeley.edu Cenko

Brad,

Please arrange for further observations of SN2012A.

I am busy teaching / traveling / sitting on the beach.

Thanks,
Professor XYZ

P. S. - I'll get to your recommendation letter as soon as I can.

#1

How Science Gets Done: II

- Write a python script `do_science`:
  - Extract additional information about a supernova from a webpage
  - Select a random graduate student from an sqlite database
  - Email the graduate student to request the observations

`urllib` & **`urllib2`**

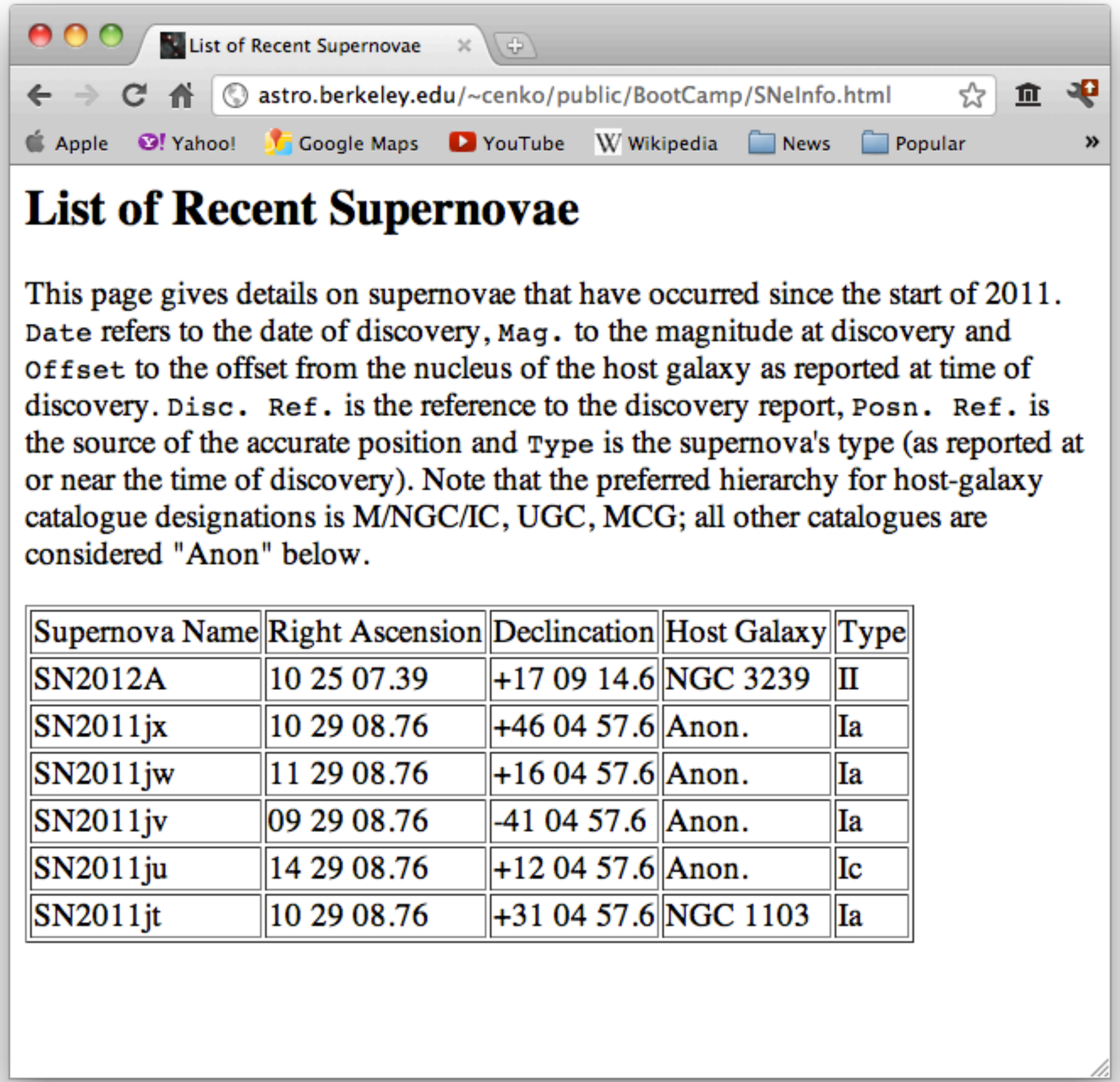
These modules provide access to any URL (uniform resource locator), the most common URL scheme being HTTP. (Others are HTTPS, FTP, FTPS, etc.)

The **`urllib`** module provides tools and functions for high-level, but less modern, interactions.

The **`urllib2`** module is more suited for complex interactions, supporting basic and digest authentication, redirections, cookies, and more.

`urllib.openurl()` is deprecated in favor of `urllib2.openurl()`

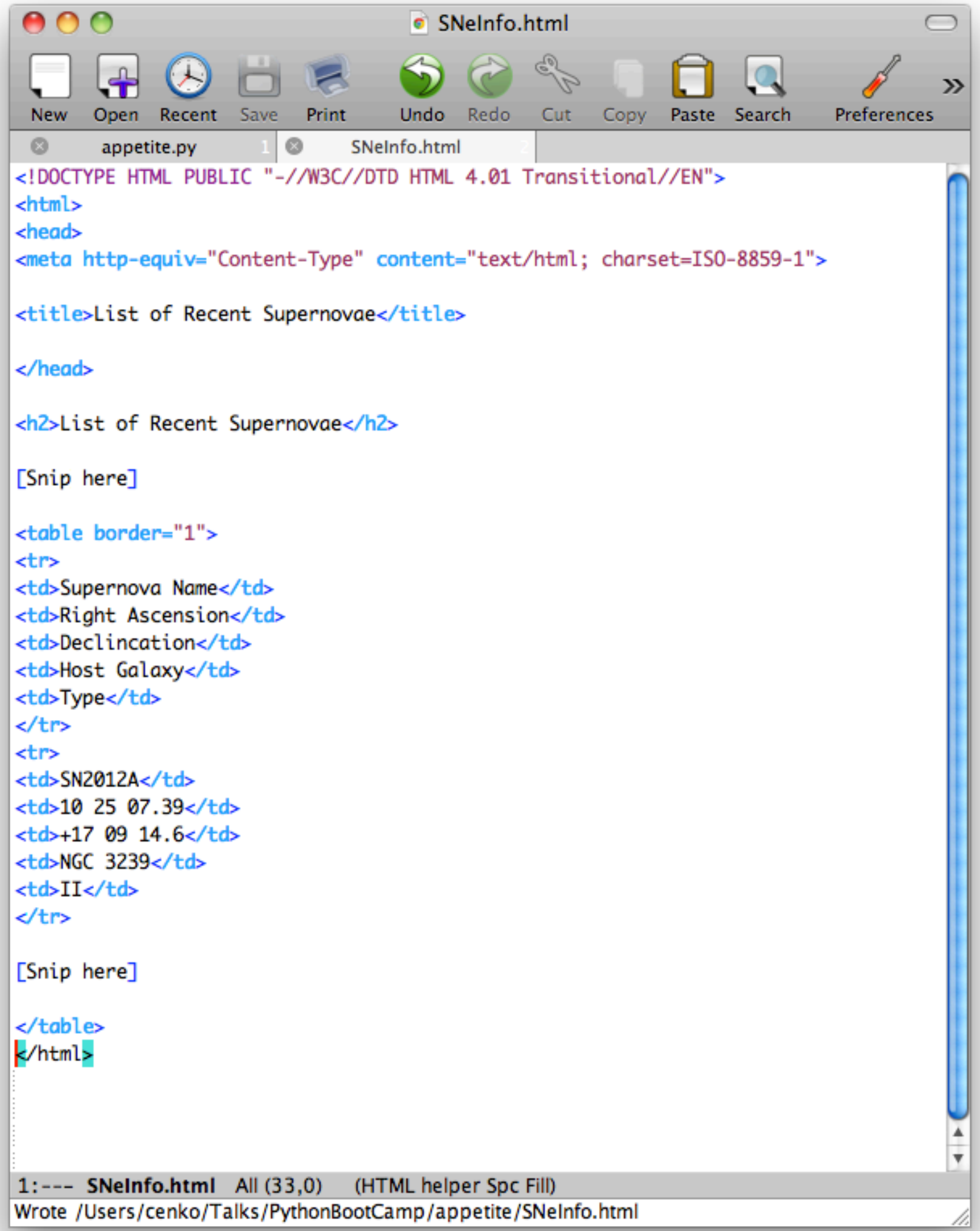
A (Simple) Webpage to Parse


List of Recent Supernovae
astro.berkeley.edu/~cenko/public/BootCamp/SNeInfo.html
Apple
Yahoo!
Google Maps
YouTube
Wikipedia
News
Popular

List of Recent Supernovae

This page gives details on supernovae that have occurred since the start of 2011. `Date` refers to the date of discovery, `Mag.` to the magnitude at discovery and `Offset` to the offset from the nucleus of the host galaxy as reported at time of discovery. `Disc. Ref.` is the reference to the discovery report, `Posn. Ref.` is the source of the accurate position and `Type` is the supernova's type (as reported at or near the time of discovery). Note that the preferred hierarchy for host-galaxy catalogue designations is M/NGC/IC, UGC, MCG; all other catalogues are considered "Anon" below.

Supernova Name	Right Ascension	Declincation	Host Galaxy	Type
SN2012A	10 25 07.39	+17 09 14.6	NGC 3239	II
SN2011jx	10 29 08.76	+46 04 57.6	Anon.	Ia
SN2011jw	11 29 08.76	+16 04 57.6	Anon.	Ia
SN2011jv	09 29 08.76	-41 04 57.6	Anon.	Ia
SN2011ju	14 29 08.76	+12 04 57.6	Anon.	Ic
SN2011jt	10 29 08.76	+31 04 57.6	NGC 1103	Ia



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

<title>List of Recent Supernovae</title>

</head>

<h2>List of Recent Supernovae</h2>

[Snip here]

<table border="1">
<tr>
<td>Supernova Name</td>
<td>Right Ascension</td>
<td>Declincation</td>
<td>Host Galaxy</td>
<td>Type</td>
</tr>
<tr>
<td>SN2012A</td>
<td>10 25 07.39</td>
<td>+17 09 14.6</td>
<td>NGC 3239</td>
<td>II</td>
</tr>

[Snip here]

</table>
</html>
```

```
1:--- SNeInfo.html   All (33,0)    (HTML helper Spc Fill)
Wrote /Users/cenko/Talks/PythonBootCamp/appetite/SNeInfo.html
```

HowTo Retrieve HTML

```
>>> import urllib2
>>> MYSNURL = "http://astro.berkeley.edu/~cenko/public/BootCamp/SNeInfo.html"
>>>
>>> flob = urllib2.urlopen(MYSNURL)
>>> s = flob.read()
>>> flob.close()
```

`urlopen` returns a file-like object, which can be `read` like any other file. As a result, s stores the HTML from the page in a (large) string

ElementTree (lxml.etree)

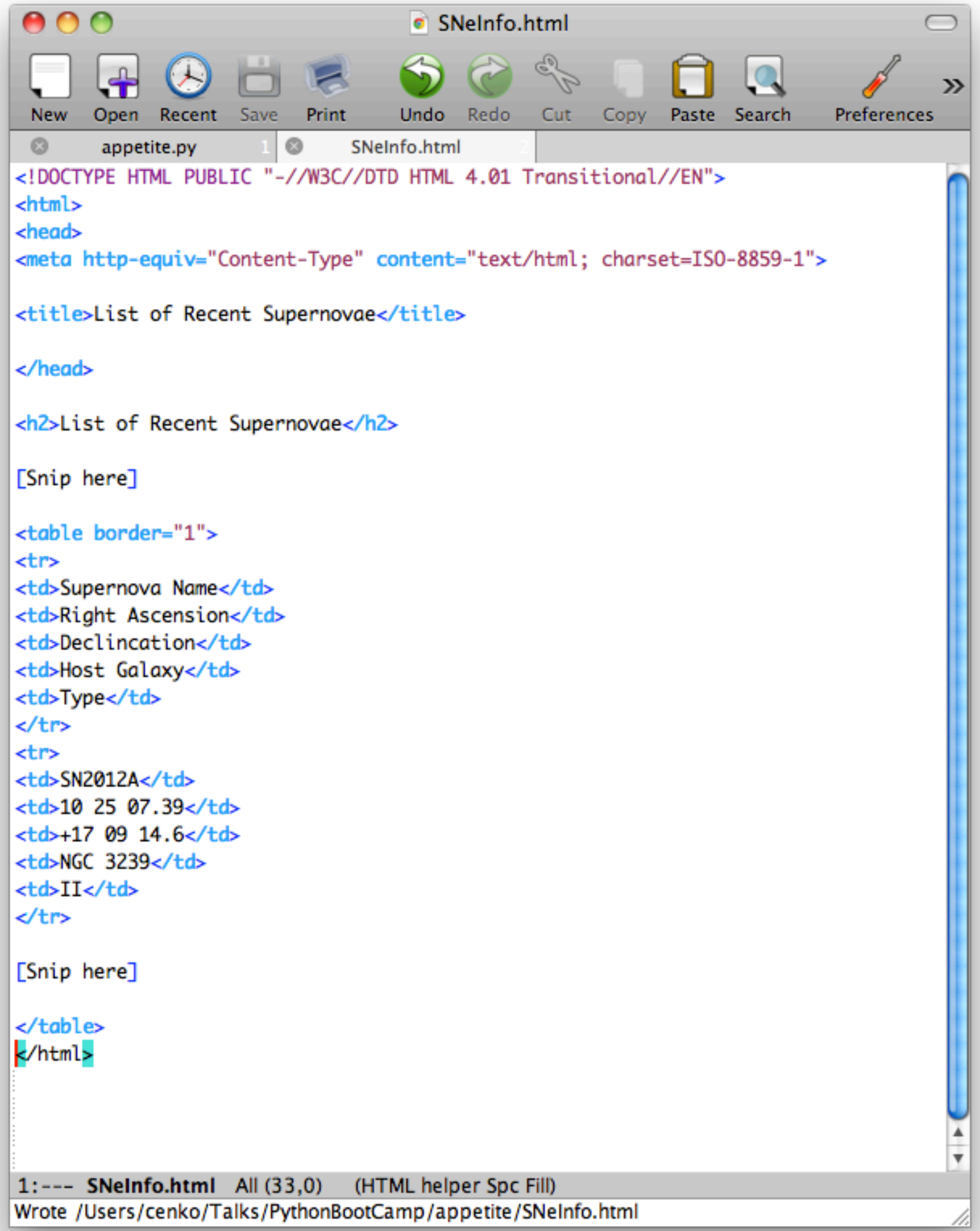
Element

Attributes: tag, text

Methods: parents, children, search

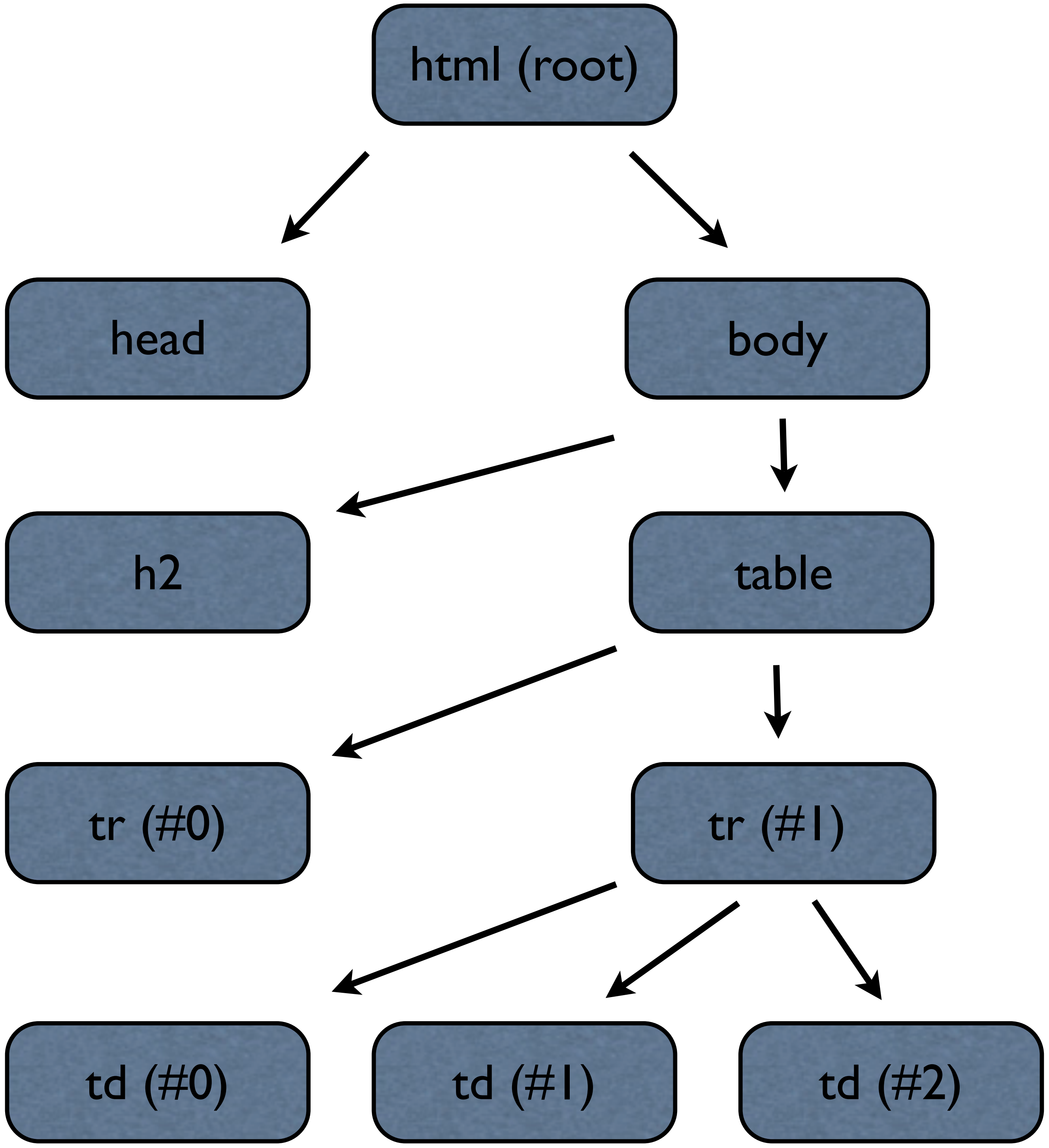
ElementTree provides a *class* that makes it more convenient to handle XML (and thus HTML) files. Element *instances* are used to represent each XML tag, with appropriate hierarchical relationships, and can be accessed with list syntax.

ElementTree (lxml.etree)



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

<title>List of Recent Supernovae</title>

</head>

<h2>List of Recent Supernovae</h2>

[Snip here]

<table border="1">
<tr>
<td>Supernova Name</td>
<td>Right Ascension</td>
<td>Declincation</td>
<td>Host Galaxy</td>
<td>Type</td>
</tr>
<tr>
<td>SN2012A</td>
<td>10 25 07.39</td>
<td>+17 09 14.6</td>
<td>NGC 3239</td>
<td>II</td>
</tr>

[Snip here]

</table>
</html>
```

```
1:--- SNeInfo.html   All (33,0)    (HTML helper Spc Fill)
Wrote /Users/cenko/Talks/PythonBootCamp/appetite/SNeInfo.html
```

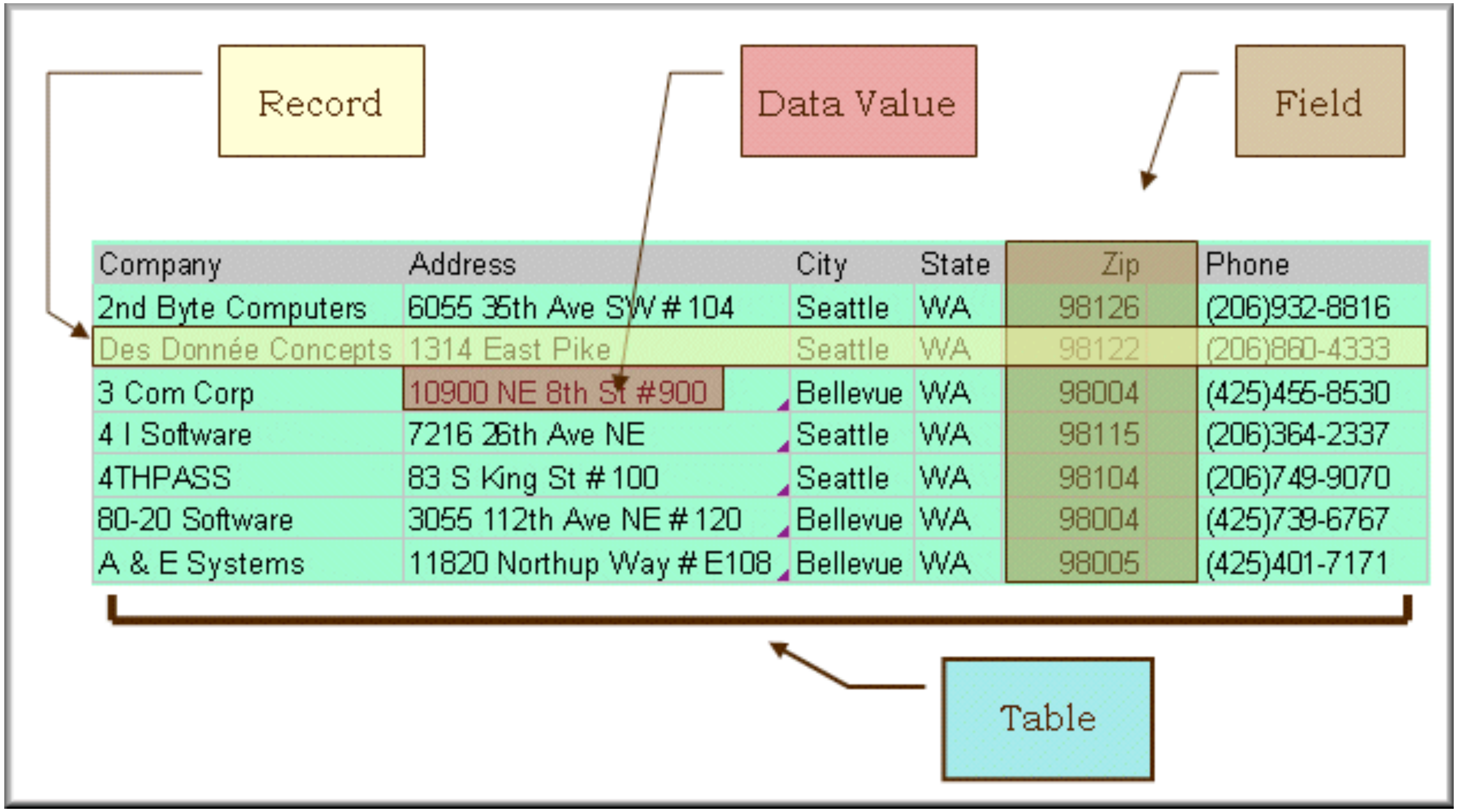


Parsing HTML with etree

```
>>> from lxml import etree
>>> s = flob.read()
>>> html = etree.HTML(s)
>>> rows = html.find('.//table')
>>> for row in rows:
...     if (sn_name == row[0].text):
...         coords = [row[1].text.replace(" ", ":"),
...                   row[2].text.replace(" ", ":")]
...         host = row[3].text
...         sntype = row[4].text
...         return [host, coords, sntype]
...
```

`HTML` converts a string into an ElementTree object. `find` identifies all the children of the first element named table. List-like indexing operations allow you to access children of nodes.

Relational Databases



Company	Address	City	State	Zip	Phone
2nd Byte Computers	6055 35th Ave SW # 104	Seattle	WA	98126	(206)932-8816
Des Donnée Concepts	1314 East Pike	Seattle	WA	98122	(206)860-4333
3 Com Corp	10900 NE 8th St #900	Bellevue	WA	98004	(425)455-8530
4 I Software	7216 26th Ave NE	Seattle	WA	98115	(206)364-2337
4THPASS	83 S King St # 100	Seattle	WA	98104	(206)749-9070
80-20 Software	3055 112th Ave NE # 120	Bellevue	WA	98004	(425)739-6767
A & E Systems	11820 Northup Way # E108	Bellevue	WA	98005	(425)401-7171

Relational databases are an efficient (searchable) way to store tabular data. Most people today use some form of SQL (MySQL, PostGreSQL, etc.)

sqlite3 Overview

- Built-in SQL database access
- Database is stored as a file (or in RAM)
- Syntax similar to MySQLdb
- Not portable (machine-dependent)

HowTo Create an sqlite Database

```
>>> import sqlite3
>>> filename = "/Users/cenko/BootCamp/2012B/appetite/astropeeps.sql"
>>> conn = sqlite3.connect(filename)
>>> c = conn.cursor()
>>> c.execute('''CREATE TABLE ASTROPEEPS (f_name text, l_name text,
...           email text, status text)''')
>>> c.execute('''INSERT INTO ASTROPEEPS VALUES ("Josh", "Bloom",
...           "jbloom@astro.berkeley.edu", "Faculty")''')
>>> c.execute('''INSERT INTO ASTROPEEPS VALUES ("Adam", "Morgan",
...           "amorgan@astro.berkeley.edu", "Student")''')
[etc.]
>>> conn.commit()
>>> c.close()
```

SQL commands are contained within the `execute` statement. Make sure to remember to commit the changes to the database before closing.

HowTo Query an sqlite Database

```
>>> filename = "/Users/cenko/Talks/PythonBootCamp/appetite/astropeeps.sql"
>>> conn = sqlite3.connect(filename)
>>> c = conn.cursor()
>>> student = “Student”
>>>
>>> c.execute("SELECT f_name, l_name, email FROM ASTROPEEPS WHERE status" + \
          " = '%s' ORDER BY RANDOM() LIMIT 1" % student)
<sqlite3.Cursor object at 0x10232e730>
>>> row = c.fetchall()
>>> print row
[(u'Adam', u'Morgan', u'amorgan@astro.berkeley.edu')]
>>> conn.commit()
>>> c.close()
>>> return [row[0][0], row[0][1], row[0][2]]
```

After `execute`, we need to perform a `fetchall` in order to retrieve the result from the query.

HowTo Email: I

```
>>> from email.MIMEMultipart import MIMEMultipart
>>> from email.MIMEText import MIMEText
>>> import NothingToSeeHere # Email password stored in this (private) file
>>> import smtplib
>>>
>>> [address, f_name, l_name] = [“amorgan@astro.berkeley.edu”, “Adam”, “Morgan”]
>>> [sn_name, host, coords, sntype] =
...  [“SN2012A”, “M31”, [“10:00:00.00”, “+31:00:00.0”], “Ic”]
>>> myemail = “bradcenko@gmail.com”
>>>
>>> msg = MIMEMultipart()
>>> msg["From"] = myemail
>>> msg["To"] = address
```

Basic email functionalities are in the email and smtplib modules. `MIMEMultipart()` will create a new instance of a message.

HowTo Email: II

```
>>> msgstr = "Hi %s %s,\n\n" % (f_name, l_name)
>>> msgstr += "I just found out about %s, and it seems neat.  " % sn_name
>>> if (host == None):
...     msgstr += "The host galaxy is unknown.  "
... else:
...     msgstr += "The host galaxy is %s.  " % host
...
>>> if (coords == None):
...     msgstr += "I do not know the coordinates.  "
... else:
...     msgstr += "The location is: RA=%s; Dec=%s.  " % (coords[0], coords[1])
...
>>> if (sntype == None):
...     msgstr += "I do not know the type.\n\n"
... else:
...     msgstr += "The type is %s.\n\n" % sntype
...
>>> msgstr += "Here's an image of the field: \n"
>>> finder = "http://qmorgan.org.org/fc/fcserver.py?ra=%s&dec=%s&src_name=%s&cont_str=Contact:
+Brad+Cenko+(bradcenko@gmail.com)" % (coords[0], coords[1], sn_name)
>>> msgstr += finder + "\n\n"
>>> msgstr += "Could you please arrange some new observations?  "
>>> msgstr += "I am really busy drinking right now.\n\n"
>>> msgstr += "Thanks,\nBrad"
>>> msg.attach(MIMEText(msgstr))
```

HowTo Email: III

```
>>> mailServer = smtplib.SMTP("smtp.gmail.com", 587)
>>> mailServer.starttls()
>>> mailServer.login(myemail, NothingToSeeHere.passwd)
>>>
>>> mailServer.sendmail(myemail, address, msg.as_string())
>>> mailServer.close()
```

`sendmail` is a method of the mailServer object.

Putting it all together

```
>>> def do_science(sn_name, filename=ASTROPEEPSDB, url=MYSNURL,
...                myemail="bradcenko@gmail.com"):

...     # See if the department database exists.  If not, create it.
...     if not os.path.exists(filename):
...         create_astro_table(filename=filename)

...     # Select a random graduate student to do our bidding
...     [f_name, l_name, address] =
...           retrieve_random_gradstudent(filename=filename)

...     # Find out some information about the supernova
...     [host, coords, sntype] = retrieve_sn_info(sn_name, url=url)

...     # Email the student
...     email_student(address, f_name, l_name, sn_name, host, coords, sntype,
...                   myemail=myemail)

...     print “I emailed %s %s at %s about %s.” %
...            (f_name, l_name, address, sn_name)

...     # Faculty job here I come!
...     return
```