RESEARCH TOOLS 2011

LECTURE 22

2011-Nov-16
Kurt Schwehr
http://schwehr.org

UNH CCOM/JHC
Part2 - Parsing binary in Python: SBET IMU navigation files

http://creativecommons.org/licenses/by-nc-sa/3.0/
Class 22: Python - parsing binary data 2 - SBET IMU data

Table of Contents

- See Also
- Public Mercurial (hg) repository
- Announcements
  - Google Oceans
  - NSF Sample and Data Policy
  - RVTEC
  - FOSS4G videos
- Setup
- Functions and arguments
- Last time, where were we?
- Writing a decode method for an sbet data record
- Being able to use the whole SBET file
- We need help from some additional functions

See Also

[Tue Nov 15 2011]
*** You have joined channel #unresearchtools
*** Users on #unresearchtools: kurtvm ahyde prasadh sthein ygh2 kjerram
gmitcheil schwehr bwelton berrya
*** #unresearchtools modes: +

<s-thein> Hi! [09:56]
<sthein> Hi [09:57]

<nhassan> (-chatzill@192.168.8.244) has joined channel #unresearchtools
<nhassan> hi [09:58]
<bwelton> hello hello [09:59]
<ygh2> hi

[Tue Nov 15 2011]
*** You have joined channel #unresearchtools
*** Users on #unresearchtools: kurtvm mohammad nhassan ahyde prasadh sthein
ygh2 kjerram gmitcheil schwehr bwelton berrya
*** #unresearchtools modes: +

<gmitcheil> Hi [11:03]
<kurtvm> do not worry about downloading the org file [11:04]

<hminami> (-chatzill@192.168.8.245) has joined channel #unresearchtools
<kurtvm> we will be getting the org file a different way today
<kurtvm> we will use mercurial (aka hg)
<hminami> Good morning

ERC>
Class 22: Python - parsing binary files

---

```
[sudo] password for researchtools:

researchtools@ubuntu:~$ sudo ntpdate ntp.ubuntu.com
15 Nov 11:03:56 ntpdate[2889]: step time server 91.189.94.4 offset 3846.744067 sec

researchtools@ubuntu:~$ date
Tue Nov 15 11:03:58 EST 2011

researchtools@ubuntu:~$ kurtvm on #unhresearchtools (+,lag:0)

[Tue Nov 15 2011]
*** You have joined channel #unhresearchtools
*** Users on #unhresearchtools: kurvtm ahyde prasadh sthein ygh2 kjerram
gmitchell schwehr bwelton berrya
*** #unhresearchtools modes: +

has joined channel #unhresearchtools

has joined channel #unhresearchtools

has joined channel #unhresearchtools

[11:03]

mohammad nhassan ahyde prasadh sthein
bwelton berrya

[11:04]

[11:05]

---

researchtools@ubuntu:~$ [researchtools@ubuntu]
```
Wednesday, November 16, 11

Announcements

Google Oceans

I will be at CCOM/JHC through the end of the year.

NSF Sample and Data Policy

National Science Foundation's Division of Ocean Sciences Sample and Data Policy:


RVTEC


FOSS4G videos

- http://www.foss4g.org/drupal/search/node/foss4g2011
See Also

- YouTube Videos on Bits, Bytes and binary
  - 3 - Decimal, Binary, Octal, & Hexadecimal
  - Binary (full understanding in 10 min)
  - Computer Architecture Lesson 1: Bits and Bytes
- Wikipedia

Public Mercurial (hg) repository  

```
hg vc dvc
```

```
https://bitbucket.org/schwehr/researchtools
```

```
mkdir projects
cd projects
sudo apt-get install mercurial # hg
hg clone https://bitbucket.org/schwehr/researchtools
tree researchtools
```

```
Logging in as 'kurtvm'... done
```

Wednesday, November 16, 11
Public Mercurial (hg) repository

https://bitbucket.org/schwehr/researchtools

```
mkdir projects
cd projects
sudo apt-get install mercurial # hg
hg clone https://bitbucket.org/schwehr/researchtools
```

Announcements

Google Oceans

I will be at CCOM/JHC through the end of the year.

NSF Sample and Data Policy

National Science Foundation’s Division of Ocean Sciences Sample
Public Mercurial (hg) repository

https://bitbucket.org/schwehr/researchtools

```
mkdir projects
cd projects
sudo apt-get install mercurial
hg clone https://bitbucket.org/schwehr/researchtools
```

Announcements

Google Oceans

I will be at CCOM/JHC through the end of the year.

NSF Sample and Data Policy

National Science Foundation’s Division of Ocean Sciences will be closing its...
Public Mercurial (hg) repository

https://bitbucket.org/schwehr/researchtools

mkdir projects
cd projects
sudo apt-get install mercurial
hg clone https://bitbucket.org/schwehr/researchtools

tree researchtools

Announcements

Google Oceans

I will be at CCOM/JHC through the end of the year.

NSF Sample and Data Policy

National Science Foundation's Division of Ocean Sciences: 

snippets

* See Also

  - YouTube Videos on Bits, Bytes and binary
    - [http://www.youtube.com/user/MyWhyU?v=3D5sS7w-CMHkJU][3 - Decimal, Binary, Octal, & Hexadecimal]
    - [http://www.youtube.com/watch?v=3DUnKbP4sc][Computer Architecture Lesson 1: Bits and Bytes]
  - Wikipedia

* Announcements

22-python-binary-files-part-2.org

Logging in as 'kurtvm'... done
Public Mercurial (hg) repository

https://bitbucket.org/schwehr/researchtools

mkdir projects
cd projects
sudo apt-get install mercurial
hg clone https://bitbucket.org/schwehr/researchtools

Announcements

Google Oceans

I will be at CCOM/JHC through the end of the year.

NSF Sample and Data Policy

National Science Foundation's Division of Ocean Sciences
Open a terminal and get going:

```bash
# update your mercurial repository
hg pull

mkdir -p ~/class/22
cd ~/class/22

pwd
# Make sure you are in the class directory
# ~/class/22

# Rather than downloading binary files
# or using wget/curl, you could
# mercurial revision control

curl -O http://vislab-ccom.bunzip2 sample.sbset.bz2

md5sum sample.sbset
```
Sign up for a free 5 user account

Username (required)

Email address (required)

Password (required)

Password (again) (required)

Sign up

Sign up using OpenID

All plans include

- Unlimited repositories
- Unlimited public collaborators
- Unlimited disk space
- Custom domains
- Issue tracking
- Downloads
- Wiki
Course material for the UNH CCOM/JHC Research Tools course. This material was started during the Fall 2011 semester. Covers emacs with org-mode, Bash command line, Python, Proj, GDAL/OGR, QGIS, and much more. It tries to build a Linux and open source software base for students to build on. Please copy, remix and improve this material. License: Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License

Clone this repository (size: 221.3 KB): HTTPS / SSH

$ hg clone https://bitbucket.org/schwehr/researchtools
```
https://bitbucket.org/schwehr/researchtools

#BEGIN_SRC sh
mkdir projects
cd projects
sudo apt-get install mercurial # hg
hg clone https://bitbucket.org/schwehr/researchtools
tree researchtools
#END_SRC

* FOSS4G videos

- http://www.foss4g.org/drupal/search/node/foss4g2011

Open a terminal and get going:

#BEGIN_SRC sh
# update your mercurial repository of the class notes
cd ~/projects/researchtools
hg pull

mkdir -p ~/class/21
cd ~/class/22

# Make sure you are in the right location
```

Wednesday, November 16, 11
bunzip2 sample.sbet.bz2
@@ -105,7 +124,7 @@

Now in ipython, run it to see what happens:

##BEGIN_SRC python
run sonar
##END_SRC
@@ -122,7 +141,7 @@
set_frequency()
##END_SRC

Run it again and you should see this, but the command number "[16]"
Run it again and you should see this, but the command number [16]
will be different for you:

##BEGIN_EXAMPLE
schwehr is sharing code with you
Bitbucket is a code hosting site. Unlimited public and private repositories. Free for small teams.

Course material for the UNH CCOM/JHC Research Tools course. This material was started during the Fall 2011 semester. Covers emacs with org-mode, Bash command line, Python, Proj, GDAL/OGR, QGIS, and much more. It tries to build a Linux and open source software base for students to build on. Please copy, remix and improve this material. License: Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License

Clone this repository (size: 221.3 KB): HTTPS / SSH
$ hg clone https://bitbucket.org/schwehr/researchtools

Commit 1c508bfc2807 Raw commit »

minor cleanup

commit: 1c508bfc2807
parent: a2a96e73b11d
branch: default
Open a terminal and get going:

```
# update your mercurial
hg pull

mkdir -p ~/class/21
cd ~/class/22

pwd

# Make sure you are in ~/class/22

# Rather than downloading
# or using wget/curl, you can now get it from the researchtools
# mercurial revision control repository


bunzip2 sample.sbet.bz2

md5sum sample.sbet
196c21f16f07ceae180888b12e9edc56 sample.sbet
```

Start ipython in the terminal
Open a terminal and get going:

```bash
# update your mercurial
hg pull

mkdir -p ~/class/21
cd ~/class/22

pwd
# Make sure you are in  
# ~/class/22

# Rather than downloading
# or using wget/curl, you can now get it from the researchtools
# mercurial revision control repository
cp ~/projects/researchtools/class/22-python-binary-files-part-2.org

bunzip2 sample.sbet.bz2

md5sum sample.sbet
196c21f16f07ceae180888b12e9edc56  sample.sbet
```

Start ipython in the terminal
Open a terminal and get going:

```bash
# update your mercurial
hg pull

mkdir -p ~/class/21
cd ~/class/22

pwd
```

Start ipython in the terminal

* See Also

- YouTube Videos on Bits, Bytes and binary
  - 3 - Decimal, Binary, Octal, & Hexadecimal
  - Binary (full understanding in 10 min)
  - Computer Architecture Lesson 1: Bits and Bytes
- Wikipedia

** Announcements**

I will be at CCOM/JHC through the end of the year.
researchtools@ubuntu:~/class/22$ python
Python 2.7.1+ (r271:86832, Apr 11 2011, 18:05:24)
Type "copyright", "credits" or "license" for more information.

IPython 0.10.1 -- An enhanced Interactive Python.
?   -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help   -> Python's own help system.
object? -> Details about 'object'. ?object also works, ?? prints more.

In [1]: logstart -o -r log-class-22.py
Activating auto-logging. Current session state plus future input saved.
Filename : log-class-22.py
Mode : backup
Output logging : True
Raw input log : True
Timestamping : False
State : active

In [2]:

```bash
# BEGIN_SRC python
logstart -o -r log-class-22.py
import struct
import numpy
import math
sbe_file = open('sample.sbe')
sbe_data = sbe_file.read()
```

---

22-python-binary-files-part-2.org 15% L92 (Org)
In [1]: logstart -o -r log-class-22.py
Activating auto-logging. Current session state plus future input saved.
Filename : log-class-22.py
Mode : backup
Output logging : True
Raw input log : True
Timestamping : False
State : active

In [2]: ls -l
total 48
-rw-r--r-- 1 researchtools researchtools 16722 2011-11-15 11:36 22-python-binary-files-part-2.org
-rw-r--r-- 1 researchtools researchtools 333 2011-11-15 11:39 log-class-22.py
-rw-r--r-- 1 researchtools researchtools 22712 2011-11-15 11:37 sample.sbet

In [3]: import struct
In [4]: import numpy
In [5]: import math
In [6]: sbet_file = open('sample.sbet')
In [7]: sbet_data = sbet_file.read()
Since functions are a bit tricky, we should go over them again. You start a function with "def" followed by the name of a function, "()" and a ":" to finish the function.

Open the file ~/class/22/sonar.py and put this in it:

```python
def setfrequency():
    print 'Setting frequency'
    # Write code here to set the frequency
```

*Be sure to save the file*. That means that the bottom left of your emacs window should have a status of "-U:" with *no* **""**
In [8]: history
1: _ip.magic("logstart -o -r log-class-22.py")
2: _ip.system("ls -F -l")
3: import struct
4: import numpy
5: import math
6: sbet_file = open('sample.sbet')
7: sbet_data = sbet_file.read()
8: _ip.magic("history ")

In [9]:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Data/Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>math</td>
<td>module</td>
<td>&lt;module 'math' (built-in)&gt;</td>
</tr>
<tr>
<td>numpy</td>
<td>module</td>
<td>&lt;module 'numpy' from '/usr...'&gt;</td>
</tr>
<tr>
<td>sbet_data</td>
<td>str</td>
<td>...</td>
</tr>
<tr>
<td>sbet_file</td>
<td>file</td>
<td>&lt;open file 'sample.sbet', mode 'r' at 0xa44c860&gt;</td>
</tr>
<tr>
<td>struct</td>
<td>module</td>
<td>&lt;module 'struct' from '/u...'&gt;</td>
</tr>
</tbody>
</table>

In [10]: ls
22-python-binary-files-part-2.org  log-class-22.py  sample.sbet  sonar.py

In [11]: run sonar

In [12]:

---

*Be sure to save the file*. That means that the bottom left of your emacs window should have a status of "-U:" with *no* "**"

Now in ipython, run it to see what happens:

```
+BEGIN_SRC python
run sonar
#+END_SRC
```

You get nothing, because we have not called the function. So add a
In [8]: history
1: _ip.magic("logstart -o -r log-class-22.
2: _ip.system("ls -F -l")
3: import struct
4: import numpy
5: import math
6: sbet_file = open('sample.sbet')
7: sbet_data = sbet_file.read()
8: _ip.magic("history ")

In [9]: whos
Variable   Type         Data/Info
math       module <module 'math' (built-in module)>
numpy      module <module 'numpy' (built-in module)>
sbet_file  str         open file 'sample.sbet'
sbet_data  str         open file 'sample.sbet'
struct     module <module 'struct' from '/usr/lib64/python3.6/struct.py'>

In [10]: ls
22-python-binary-files-part-2.org  log-cla

In [11]: run sonar
In [12]:

function call to your code that uses setfrequency:

# BEGIN_SRC python
# "Define" or create the function
def setfrequency():
    print 'Setting frequency'
    # Write code here to set the frequency

setfrequency()

# Use the function
setfrequency()
# END_SRC

Run it again and you should see this, but the command number [16]
will be different for you:

# BEGIN EXAMPLE
Wrote /home/researchtools/class/22/sonar.py
3: import struct
4: import numpy
5: import math
6: sbet_file = open('sample.sbet')
7: sbet_data = sbet_file.read()
8: ip.magic("history ")

In [9]: whos
Variable Type Data/Info
--- --- ------
math module <module 'math' (built-in)>
numpy module <module 'numpy' from /usr/local/lib/python2.7/dist-packages/numpy/__init__.pyc> sbet_data str 0xFQG[AFH[1]]?GQ7Q struct module <module 'struct' from /usr/local/lib/python2.7/dist-packages/struct.pyc>

In [10]: ls
22-python-binary-files-part-2.org log-classifier.log

In [11]: run sonar
In [12]: run sonar
Setting frequency

In [13]:

# Define or create the function
def set_frequency():
    print 'Setting frequency'
    # Write code here to set the frequency

set_frequency()

Run it again and you should see this, but the command number [16] will be different for you:

# BEGIN_SRC python
# Define or create the function
def set_frequency():
    print 'Setting frequency'
    # Write code here to set the frequency

# Use the function
set_frequency()
# END_SRC

# BEGIN_EXAMPLE
22-python-binary-files-part-2.org 25% L153 [#,r] (Org)
3: import struct
4: import numpy
5: import math
6: sbet_file = open('sample.sbet')
7: sbet_data = sbet_file.read()
8: _ip.magic("history ")

In [9]: whos
Variable  Type  Data/Info
math       module <module 'math' (built-in)>  
numpy      module <module 'numpy' from ...
8: # Define or create the function
    def setfrequency(freq):
        print 'Setting frequency', freq
    # Write code here to set the frequency

setfrequency()
In [10]: ls
22-python-binary-files-part-2.org  log-class-22.py  sample.sbet  sonar.py

In [11]: run sonar

In [12]: run sonar
Setting frequency

In [13]: run sonar

TypeError Traceback (most recent call last)
/home/researchtools/class/22/sonar.py in <module>()
 5     # Write code here to set the frequency

----> 7 setfrequency()

TypeError: setfrequency() takes exactly 1 argument (0 given)
WARNING: Failure executing file: <sonar.py>

In [14]: 

---BEGIN_SRC python
# "Define" or create the function
def setfrequency(freq):
    print 'Setting frequency to', freq
    # Write code here to set the frequency

# Use the function
setfrequency()
---END_SRC

Now if we run the code, we get an error!

22-python-binary-files-part-2.org  29% L175  [#,r] (Org)
Wrote /home/researchtools/class/22/sonar.py
In [11]: run sonar

In [12]: run sonar
Setting frequency

In [13]: run sonar

TypeError

/home/researchtools/class/22/sonar.py in <
  5     # Write code here to set the frequency
     
8  
9 ----> 7 setfrequency()

TypeError: setfrequency() takes exactly 1 argument (0 given)

WARNING: Failure executing file: <sonar.py>

In [14]: run sonar
Setting frequency 12000

In [15]:

# Define or create the function
def setfrequency(freq = 12000):
    print 'Setting frequency', freq, 'Hz'
    # Write code here to set the frequency

setfrequency()

By calling "setfrequency()", we did not pass in a value for freq.  
Oops!

For arguments, we can assign a "default" value that will be used if  
nothing is passed in when calling (aka using) the function.  You don't 
have to do this, but it is often a smart thing to do.  Why don't we 
make the default frequency be 12kHz (12000 cycles per second).  
Change the def line to look like this:

# BEGIN_SRC python
def setfrequency(freq = 12000):
    # END_SRC

Now when you run the function, you will see this:

--- 22-python-binary-files-part-2.org  34% L210  [#, r] (Org)---
In [12]: run sonar
Setting frequency

TypeError

/home/researchtools/class/22/sonar.py in <module>
  5     # Write code here to set the frequency

----> 7 setfrequency()

TypeError: setfrequency() takes exactly 1 argument (0 given)
WARNING: Failure executing file: <sonar.py>

In [14]: run sonar
Setting frequency 12000

In [15]: run sonar
Setting frequency 24000 Hz

In [16]:

So what do we do now if we want to change it to a different frequency? We need to call `setfrequency` with a different number. Let's double the frequency to 24kHz. Change the call to look like this:

```
# BEGIN EXAMPLE
In [18]: run sonar
Setting frequency to 12000
# END EXAMPLE
```

Running the sonar.py code in ipython looks like this:
TypeError: setfrequency() takes exactly 1

WARNING: Failure executing file: <sonar.py

In [14]: run sonar
Setting frequency 12000

In [15]: run sonar
Setting frequency 24000 Hz

In [16]: run sonar
Setting frequency 15000 Hz

In [17]:

# Define or create the function

def set_frequency(freq = 12000):
    print 'Setting frequency', freq, 'Hz'
    # Write code here to set the frequency

my_sonar_freq = 15000

set_frequency( my_sonar_freq )
# Define or create the function

def setfrequency(freq = 12000):
    
    print 'Setting frequency', freq, 'Hz'
    
    # Write code here to set the frequency

sonar_freq_table = {
    'em122': 12000,
    'knudsen': 3500,
}

setfrequency( sonar_freq_table['knudsen'] )

print 'Setting frequency to', freq

# Write code here to set the frequency

sonar_freq_table = {
    'em122': 12000,
    'knudsen': 3500,
}

setfrequency( sonar_freq_table['knudsen'] )

# END_SRC

# BEGIN_EXAMPLE

In [21]: run sonar

Setting frequency to 3500

# END_EXAMPLE
```python
# "Define" or create the function
def set_frequency(freq = 12000, name='unknown'):
    print 'Setting frequency', freq, 'Hz', 'name is', name
    # Write code here to set the frequency

sonar_freq_table = {
    'em122': 12000,
    'knudsen': 3500,
}

set_frequency( sonar_freq_table['knudsen'], 'R/V Super Slow' )
```

```python
# BEGIN_EXAMPLE
In [21]: run sonar
Setting frequency to 3500
# END_EXAMPLE

Hopefully that gives you a better field for functions! Now we will get back to creating our sbet.py module with functions to handle reading IMU navigation data.

* Last time, where were we?
```
---
7 setfrequency()

TypeError: setfrequency() takes exactly 1

WARNING: Failure executing file: <sonar.py

In [14]: run sonar
Setting frequency 12000

In [15]: run sonar
Setting frequency 24000 Hz

In [16]: run sonar
Setting frequency 15000 Hz

In [17]: run sonar
Setting frequency 3500 Hz

In [18]: run sonar
Setting frequency 3500 Hz name is R/V Super

In [19]:

---

U:---- sbet.py All L1 [#,r] (Python yas)

Last time we were editing ~/class/22/sbet.py. Here is a cleaned up version of
where we left off. I have removed the extra print statements.

# BEGIN_SRC python
# Decode Applonix POSPac SBET IMU binary files

def decode(data):
    'Decipher a SBET datagram from binary'
    print 'Data length:', len(data)

def main():
    print 'Starting main'
    sbet_file = open('sample.sbet')
    sbet_data = sbet_file.read()

Using the CPython shell
By running “IM-Python” Rescan, it will know about our functions. Not quite completion, but helpful.
```python
# Decode Applanix POSPac SBET IMU binary files

def decode(data):
    'Decipher a SBET datagram from binary'
    print 'Data length:', len(data)

def main():
    print 'Starting main'
    sbet_file = open('sample.sbet')
    sbet_data = sbet_file.read()
    print 'Read this many bytes:', len(sbet_data)
    decode(sbet_data)
```

```python
In [18]: run sonar
Setting frequency 35000 Hz

In [19]: run sonar
Setting frequency 35000 Hz
```
**Type Error**: `setfrequency()` takes exactly 1 argument 1

```
In [14]: run sonar
Setting frequency 12000
```

```
In [15]: run sonar
Setting frequency 24000 Hz
```

```
In [16]: run sonar
Setting frequency 15000 Hz
```

```
In [17]: run sonar
Setting frequency 3500 Hz
```

```
In [18]: run sonar
Setting frequency 3500 Hz name is R/V Super
```

```
In [19]:
```

---

Open `~/class/22/sbet.py` and put the above code into the file.

* Writing a decode method for an sbet data record

Get started on working with your sbet file in ipython

```python
# Decoding Applinex POSPac SBET IMU binary files

def decode(data):
    'Decipher a SBET datagram from binary'
    print 'Data length:', len(data)

def main():
    print 'Starting main'
    sbet_file = open('sample.sbet')
    sbet_data = sbet_file.read()

    print 'Read this many bytes:', len(sbet_data)

    decode(sbet_data)

    print 'Fin
```

# END_SRC
```
TypeError: setfrequency() takes exactly 1

WARNING: Failure executing file: <sonar.py

In [14]: run sonar
   Setting frequency 12000

In [15]: run sonar
   Setting frequency 24000 Hz

In [16]: run sonar
   Setting frequency 15000 Hz

In [17]: run sonar
   Setting frequency 3500 Hz

In [18]: run sonar
   Setting frequency 3500 Hz name is R/V Super

In [19]:

Open ~/class/22/sbet.py and put the above code into the file.

* Writing a decode method for an sbet data record

Get started on working with your sbet file in ipython
In [15]: run sonar
Setting frequency 24000 Hz

In [16]: run sonar
Setting frequency 15000 Hz

In [17]: run sonar
Setting frequency 3500 Hz

In [18]: run sonar
Setting frequency 3500 Hz name is R/V Super Slow

In [19]: import sbet

In [20]: sbet
sbet._builtin_  sbet._getattribute_  sbet._reduce_ex_  sbet.main
sbet._class_   sbet._hash_         sbet._repr_     sbet.py
sbet._delattr_ sbet._init_         sbet._setattr_  sbet.pyc
sbet._dict_    sbet._name_         sbet._sizeof_   sbet.pyc
sbet._doc_     sbet._new_          sbet._str_      sbet.py~
sbet._file_    sbet._package_      sbet._subclasshook_

In [20]: sbet.

---

Writing a decode method for an sbet data record

Get started on working with your sbet file in ipython

#+BEGIN_SRC python
import sbet
sbet.decode()

# remember that after you have done an import, you must use this to
# get updates
reload sbet
---

22-pyton-binary-files-part-2.org  49% L315  [#,r] (Org)
```python
In [20]: del sbet
In [21]: who
Variable          Type Data/Info
-----------        -------          
math               module <module 'math' (built-in)> 
my_sonar_freq     int  15000
numpy              module <module 'numpy' from '/usr...>
setfrequency      function <function setfrequency at 0xa463f0c>
sonar_freq_table  dict {'knudsen': 3500, 'em122': 12000}
struct             module <module 'struct' from '/...>

In [22]: sbet
sbe.py  sbet.py~ sbet_data sbet_file
In [22]: sbet.
sbe.py  sbet.py~
In [23]: sbet.

```
In [22]: import sbet

In [23]: sbet.
sbet._builtins__ sbet._getattribute_ sbet._reduce_ex_ sbet.main
sbet._class_ sbet._hash_ sbet._repr_ sbet.py
sbet._delattr_ sbet._init_ sbet._setattr_ sbet.pyc
sbet._dict_ sbet._name_ sbet._sizeof_ sbet.py~
sbet._doc_ sbet._new_ sbet._str_ sbet.decode
sbet._file_ sbet._package_ sbet._subclasshook_
sbet._format_ sbet._reduce_

In [23]: sbet.decode()

Traceback (most recent call last)
/home/researchtools/class/22/sonar.py in <module>()
----> 1
2
3
4
5

TypeError: decode() takes exactly 1 argument (0 given)

In [24]: sbet.decode(sbet_data)
def decode(data):
    'Decipher a SBET datagram from binary'
    print 'Data length:', len(data)

def main():
    print 'Starting main'
    sbet_file = open('sample.sbet')
    sbet_data = sbet_file.read()
    print 'Read this many bytes:', len(sbet_data)
    decode(sbet_data)
    print 'Finish main'

Try out the decode function. If you run `whos`, you will see we have an sbet_data variable in our workspace.

We can also call our main function:

```python
sbet.main()
# Starting main
# Read this many bytes: 22712
```
TypeError

/home/researchtools/class/22/sonar.py in <---- 1
 2
 3
 4
 5

TypeError: decode() takes exactly 1 argument

In [24]: sbet.decode(sbet_data)
Data length: 22712

In [25]: reload sbet
       >>> reload(sbet)
Out[25]: <module 'sbet' from 'sbet.pyc'>

In [26]: sbet.main()
Starting main
Read this many bytes: 22712
Data length: 22712
Finish main

In [27]:

---

# Decode Applanix POSPac SBET IMU binary files

```python
import struct
import math

def decode(data):
    'Decipher a SBET datagram from binary'
    print 'Data length:', len(data)

def main():
    print 'Starting main'
    sbet_file = open('sample.sbet')
    sbet_data = sbet_file.read()
    print 'Read this many bytes:', len(sbet_data)

    data = sbet_data
    values = struct.unpack('17d', data[0:8*17])
    print 'type of values:', type(values)
    print 'contents of values:', values

    time = values[0]
```

The `values` variable will be a list of 17 values

---

[# ] (Python yas)
By default, “Paren Match Highlighting” was off. Here I turned it on.
# Decode Applanix POSPac SBET IMU binary files

```python
import struct
import math

def decode(data):
    # Decipher a SBET datagram from binary
    print 'Data length:', len(data)
    values = struct.unpack('17d', data[0:8*17])
    print 'type of values:', type(values)

    def main():
        print 'Starting main'
        sbet_file = open('sample.sbet')
        sbet_data = sbet_file.read()
        
        # The values variable will be a list of 17 values

        #+BEGIN_SRC python
        import math
        import struct
        
        def decode(data):
            "Decipher a SBET datagram from binary"
            print "Start decoding datagram"
            values = struct.unpack('17d', data[0:8*17])
            
            print 'type of values:', type(values)
            print 'contents of values:', values
            time = values[0]
        #END_SRC

        22-python-binary-files-part-2.org 55% L354 [#,r] (Org)
```

In [24]: sbet.decode(sbet_data)
Data length: 22712

In [25]: reload sbet
----------> reload(sbet)
Out[25]: <module 'sbfet' from 'sbfet.pyc'>

In [26]: sbet.main()
Starting main
Read this many bytes: 22712
Data length: 22712
Finish main

In [27]: 8*17
Out[27]: 136

In [28]:

---

---
```
# Decode Applanix POSPac SBET IMU binary files

import struct
import math

def decode(data):
    # Decipher a SBET datagram from binary
    print 'Data length:', len(data)
    values = struct.unpack('17d', data[0:8*17])

    print 'type of values:', type(values)
    print 'contents of value:', values

    time = values[0]
    latitude = values[1]
    lat_deg = math.degrees(latitude)

    longitude = values[2]
    lon_deg = math.degrees(longitude)
```

```
In [25]: reload sbet
        ---> reload('sbet')
Out[25]: <module 'sbet' from 'sbet.pyc'>

In [26]: sbet.main()
Starting main
Read this many bytes: 22712
Data length: 22712
Finish main

In [27]: 8*17
Out[27]: 136
```

```
In [28]:
```
values = struct.unpack('17d', data[0:8*17])

    print 'type of values:', type(values)
    print 'contents of value:', values

time = values[0]
latitude = values[1]
lon_deg = math.degrees(latitude)

longitude = values[2]
lon_deg = math.degrees(longitude)

    print 'results:', time, lat_deg, lon_deg

def main():
    print 'Starting main'

lon_deg = math.degrees(longitude)

    print 'results:', time, lat_deg, lon_deg

    #+END_SRC

We also would like to make this a run-able script from within ipython, so add this to the end of sbet.py:

    #+BEGIN_SRC python
    if __name__ == '__main__':
        print 'starting to run script...'
        main()
        print 'script done!'
    #+END_SRC
def main():
    print 'Starting main'
    sbet_file = open('sample.sbet')
    sbet_data = sbet_file.read()
    print 'Read this many bytes:', len(sbet_data)
    decode(sbet_data)
    print 'Finish main'

longitude = values[2]
lon_deg = math.degrees(longitude)

print 'results:', time, lat_deg, lon_deg

We also would like to make this a run-able script from within ipython, so add this to the end of sbet.py:

```python
if __name__ == '__main__':
    print 'starting to run script...'
    main()
    print 'script done!'
```

```
Starting main
Read this many bytes: 22712
Data length: 22712
Finish main

In [27]: 8*17
Out[27]: 136

In [28]: reload sbet
--------> reload(sbet)
Out[28]: <module 'sbet' from 'sbet.py'>

In [29]: sbet.main()
Starting main
Read this many bytes: 22712
Data length: 22712
type of values: <type 'tuple'>
contents of value: (334959.0048233234, 1.55, 0.45, 15.0437825046453915, 0.9982283187188938, 603057936824, -0.09985686530029529, -0.40812, 0.701830645653144, 0.02132017633623, results: 334959.004823 60.4443123064 -146.66666666666669).
Finish main

In [30]: !

decode(sbet_data)

print 'Finish main'

if __name__ == '__main__':
    print 'starting to run script...'
    main()
    print 'script done!

so add this to the end of sbet.py:

#+BEGIN_SRC python
if __name__ == '__main__':
    print 'starting to run script...'
    main()
    print 'script done!

#+END_SRC

And add this as the very first line of sbet.py to make it run-able from the bash shell:

#+BEGIN_SRC python
#!/usr/bin/env python

#+END_SRC
Starting main
Read this many bytes: 22712
Data length: 22712
Finish main

In [27]: 8*17
Out[27]: 136

In [28]: reload sbet
-----> reload(sbet)
Out[28]: <module 'sbet' from 'sbet.py'>

In [29]: sbet.main()
Starting main
Read this many bytes: 22712
Data length: 22712
type of values: <type 'tuple'>
Contents of value: (334959.0048233234, 1.055115, 0.437825046453915, 0.998228318789838, 603059736824, -0.09985686530029529, -0.401812, 0.07018300645653144, 0.0213201763362)
Finish main

In [30]: !

#!/usr/bin/env python

# Decode Applanix POSPac SBET IMU binary files
import struct
import math

def decode(data):
    'Decipher a SBET datagram from binary'
    # length of data
    values = struct.unpack('17d',data[0:8*17])

    print 'type of values:', type(values)
    print 'contents of value:', values

    time = values[0]

U:----- sbet.py Top L1 [#r] (Python yas)-----

And add this as the very first line of sbet.py to make it run-able from the bash shell:

#+BEGIN_SRC python
#!/usr/bin/env python
#+END_SRC

To complete making sbet.py work from the bash prompt, you need to set the file as executable with chmod. Remember that "!" tells iPython that we want to run a shell command:

#+BEGIN_SRC python
!chmod +x sbet.py

ls -l sbet.py
In [31]: !chmod +x sbet.py

In [32]: ls -l

To complete making sbet.py work from the bash prompt, you need to set
the file as executable with chmod. Remember that "!" tells ipython
that we want to run a shell command:

# BEGIN_SRC python
!chmod +x sbet.py

# END_SRC

ls -l sbet.py
In [33]: `!sbet.py`

sh: sbet.py: not found

In [34]: `!./sbet.py`

starting to run script...

Reading this many bytes: 22712
Data length: 22712

type of values: <type 'tuple'>

contents of value: (334959.0048233234, 1.0549522638507869, -2.559965741819528, 12.826300557342815, 10.437825046453915, 0.998228318178983, 0.18282804536664027, -0.002623394812042344, 0.1141663957936824, -0.09985686530029529, -0.4015467392667415, -0.8249097558096672, -0.3413483211034812, 0.07018300645631344, 0.02132017633628756, 0.029000032024608147, -0.006807197876212325)

results: 334959.004823 60.4443123064 -146.675232704

Finish main script done!

In [35]: run sbet
Data length: 22712

In [35]: run sbet.py
    starting main
    Read this many bytes: 22712
    Data length: 22712
    type of values: <type 'tuple'>
    contents of value: (334959.0048233234, 1.0549522638507869, -2.559965741819528, 12.82630557342815, 10.437825046453915, 0.998228318178983, 0.18282804536664027, -0.0026283394812042344, 0.11416603657936824, -0.09985686530029529, -0.4015467392667415, -0.824909755809672, -0.3413483211034812, 0.07018300645653144, 0.021320176833628756, 0.029000032024608147, -0.006807197876212325)
    results: 334959.004823 60.4443123064 -146.675232704
    Finish main
    script done!
    !sbet.py
    # sh: sbet.py: not found
    # Oops! We need to tell bash where the program is located
    !.sbet.py
    # It should print out quite a bit here
    # Or you can run it directly from ipython
    run sbet
    #+END_SRC
Wednesday, November 16, 11

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

# Decode Applanix POSPac SBET IMU binary files

import struct
import math

field_names = ('time', 'latitude', 'longitude', 'altitude', 
               'x_vel', 'y_vel', 'z_vel', 
               'roll', 'pitch', 'platform_heading', 'wander_angle', 
               'x_acceleration', 'y_acceleration', 'z_acceleration', 
               'x_angular_rate', 'y_angular_rate', 'z_angular')

def decode(data):

    data = struct.unpack('!d', data[0:8*17])

    sbet_values = dict(zip(field_names, values))
```

```python
#+BEGIN_SRC python
field_names = ('time', 'latitude', 'longitude', 'altitude', 
               'x_vel', 'y_vel', 'z_vel', 
               'roll', 'pitch', 'platform_heading', 'wander_angle', 
               'x_acceleration', 'y_acceleration', 'z_acceleration', 
               'x_angular_rate', 'y_angular_rate', 'z_angular')

def decode(data):

    data = struct.unpack('!d', data[0:8*17])

    sbet_values = dict(zip(field_names, values))
```
```python
def decode(data):
    'Decipher a SBET datagram from binary'
    print 'Data length:', len(data)
    values = struct.unpack('!7d', data[0:8*17])
    print 'type of values:', type(values)
    print 'contents of value:', values
    time = values[0]
    latitude = values[1]
    lat_deg = math.degrees(latitude)
    longitude = values[2]
    lon_deg = math.degrees(longitude)
    print 'results:', time, lat_deg, lon_deg

def main():
    print 'Starting main'
    sbet_file = open('sample.sbet')
    sbet_data = sbet_file.read()
    decode(sbet_data)
    print 'Finish main'

if __name__ == '__main__':
    print 'starting to run script...'
    main()
    print 'script done!'```

Wednesday, November 16, 11
```python
def decode(data):
    'Decipher a SBET datagram from binary'
    print 'Data length:', len(data)
    values = struct.unpack('17d', data[0:8*17])

    sbet_values = dict(zip(field_names, values))
    sbet_values['lat_deg'] = math.degrees(sbet_values['latitude'])
    sbet_values['lon_deg'] = math.degrees(sbet_values['longitude'])

    print 'results:'
    for key in sbet_values:
        print ' ', key, sbet_values[key]

def main():
    # Code...
```

---

```
```
Finish main
script done!

In [37]: who
Variable


decode
field_names
main
math
my_sonar_freq
numpy
sbet
sbet_data
sbet_file
set_frequency
sonar_freq_table
dict
struct

In [38]: from pprint import pprint

In [39]: pprint(sonar_freq_table)

In [40]:

--- sbet.py Top L7 [#,r] (Python yas)---

from pprint import pprint

def decode(data):
    "Decipher a SBET datagram from binary"
    values = struct.unpack('17d',data[0:8*17])
    
    # Create a dictionary for all the values
    sbet_values = dict(zip(field_names, values))
    
    sbet_values['lat_deg'] = math.radians(sbet_values['latitude'])
    sbet_values['lon_deg'] = math.radians(sbet_values['longitude'])

    return sbet_values

Now replace the print and for loop at the end of =decode= so that
decode looks like this with just a =return sbet_values=.

--- 22-python-binary-files-part-2.org 80% L499 [#,r] (Org)---
def main():
    print('Starting main'
    sbet_file = open('sample.sbet')

"Decipher a SBET datagram from binary"  
values = struct.unpack('17d',data[0:8*17])

# Create a dictionary for all the values
sbet_values = dict(zip (field_names, values))

sbet_values['lat_deg'] = math.degrees(sbet_values['latitude'])

sbet_values['lon_deg'] = math.degrees(sbet_values['longitude'])

return sbet_values
# Send the sbet_values dictionary back to the caller

Now the =main= function needs to handle the printing of the dictionary.

#BEGIN_SRC python

#END_SRC
Finish main script done!

```python
In [37]: whos
Variable     Type          Data/Info
decode        function       <function decode>
field_names   tuple          ('time', 'latdeg', 'londeg', 'nchn', 'numchn', 'nchn2', 'freq', 'freq2', 'band', 'band2', 'flags', 'numflags', 'mask', 'dataprep', 'timestamp', 'lat', 'lon', 'time', 'start_freq', 'stop_freq', 'start_freq2', 'stop_freq2', 'bandwidth', 'bandwidth2', 'numchn', 'numchn2')
main          function       <function main>
math          module         15000
my_sonar_freq int            12000
numpy         module         {'kdnsen': 3500}
sb_data       str            'em122': 12000, 'knudsen': 3500
sbet_file     file           open('sample.sb')
sbet_data     str            'em122': 12000, 'knudsen': 3500

In [38]: from pprint import pprint
In [39]: pprint(sonar_freq_table)
{'em122': 12000, 'knudsen': 3500}
In [40]:
```

```python
'Decipher a SBET datagram from binary'
print 'Data length:', len(data)
values = struct.unpack('<17d', data[0:8*17])

sbet_values = dict(zip (field_names, values))

sbet_values['lat_deg'] = math.degrees(sbet_values['latitude'])
sbet_values['lon_deg'] = math.degrees(sbet_values['longitude'])

return sbet_values

def main():
    print 'Starting main'
    sbet_file = open('sample.sb')
    sbet_data = sbet_file.read()

U:----- sbet.py  40% L26  ['#,r] (Python yas)

"Decipher a SBET datagram from binary"
values = struct.unpack('<17d', data[0:8*17])

# Create a dictionary for all the values
sbet_values = dict(zip (field_names, values))

sbet_values['lat_deg'] = math.degrees(sbet_values['latitude'])
sbet_values['lon_deg'] = math.degrees(sbet_values['longitude'])

return sbet_values # Send the sbet_values dictionary back to the caller

# END_SRC
```

Now the `main` function needs to handle the printing of the dictionary.

```bash
Wrote /home/researchtools/class/22/sbet.py
```

Wednesday, November 16, 11
Finish main script done!

```
In [37]: whos
Variable
  decode
  field_names
  main
  math
  my.sonar_freq
  numpy
  sbet
  sbet_data
  sbet_file
  setfrequency
  sonar_freq_table
  struct

In [38]: from pprint import pprint
In [39]: pprint(sonar_freq_table)
'sonar_freq_table': { 'el12': 12000, 'knudsen': 3500 }
In [40]:
```

```python
return sbet_values

def main():
    print 'Starting main'
    sbet_file = open('sample.sbet')
    sbet_data = sbet_file.read()

    print 'Read this many bytes:', len(sbet_data)
    
    datagram = decode(sbet_data)
    pprint(datagram)

    print 'Finish main'
```

```
U:------ sbet.py  66% L36  [#,*] (Python yas)------
  return sbet_values  # Send the sbet_values dictionary back to the caller
  #+END_SRC

Now the =main= function needs to handle the printing of the dictionary.

#BEGIN_PYTHON

```python
def main():
    sbet_file = open('sample.sbet')
    sbet_data = sbet_file.read()

    datagram = decode(sbet_data)
    pprint(datagram)  # Note that we are using pretty print, not print

#END_PYTHON

Wrote /home/researchtools/class/22/sbet.py
```

Wednesday, November 16, 11
Read this many bytes: 22712
Data length: 22712

{'altitude': 12.826300557342815, 'lat_deg': 60.444312306421738, 'lat': 1.0549522638507869, 'lon_deg': -146.675237043359, 'longitute': -2.559965741819528, 'pitch': 0.11146603507936824, 'platform_heading': -0.09985686530029529, 'roll': 0.0026283394812042344, 'time': 334959.00048233234, 'wander_angle': -0.4015467392667415, 'x_accleration': -0.8249097558096672, 'x_angular_rate': 0.021320176833268756, 'x_vel': 10.437825046453915, 'y_accleration': -0.3413483211034812, 'y_angular_rate': 0.029000032024608147, 'y_vel': 0.998228318178983, 'z_accleration': -0.006807197876212325, 'z_angular_rate': 0.18282804536664027}

Finish main
script done!

In [41]:

```python
return sbet_values

def main():
    print 'Starting main'
    sbet_file = open('sample.sbets')
    sbet_data = sbet_file.read()

    print 'Read this many bytes:', len(sbet_data)

    datagram = decode(sbet_data)
    pprintln(datagram)

    print 'Finish main'

#-END_SRC

Now the =main= function needs to handle the printing of the dictionary.

#+BEGIN_SRC python
def main():
    sbet_file = open('sample.sbets')
    sbet_data = sbet_file.read()

    datagram = decode(sbet_data)
    pprintln(datagram)  # Note that we are using pretty print, not print

#-END_SRC

Wrote /home/researchtools/class/22/sbet.py