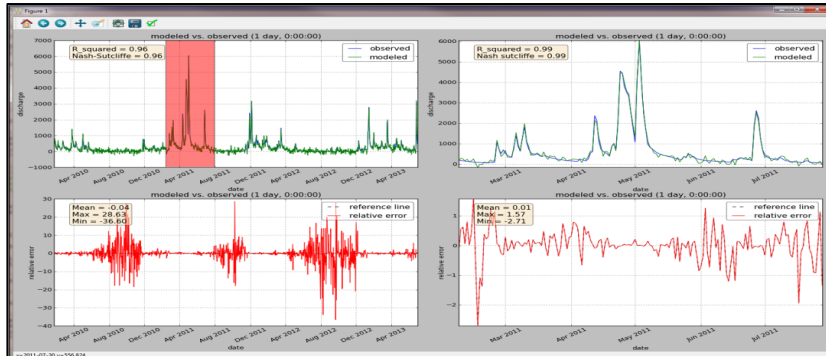
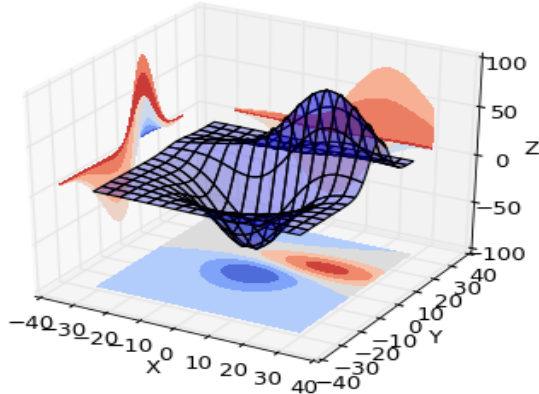


Scientific Computing Group

Programming with Python:

Modules



```
# Write Fibonacci series up to n
>>> def fib(n):
>>>     a, b = 0, 1
>>>     while a < n:
>>>         print(a, end=' ')
>>>         a, b = b, a+b
>>>     print()
>>> fib(1000)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610
```

Jeremiah Lant, Hydrologist
USGS Kentucky Water Science Center
jlant@usgs.gov

Keeping up to date with Scientific Computing Group's GitHub repository

1. Use GitHub website to view *.md material (readme, meeting summaries, resources, etc.)
2. Download the recordings from the ftp site.
3. **Clone** the GitHub repo and **pull** updates at your convenience (presentations, code, etc.)
 - a. You do not actually work in the scientific computing group's repo during meetings. You work in your own workspace outside the repo.

Note:

If you do not want to use git, then you can download the zip file from the GitHub website to have the material locally on your machine, but you have to re-download after every meeting to keep updated.

Getting Updates


- You work in a different workspace outside the scientific computing group's GitHub repo, and just get updates from the group's GitHub repo.

/my-scientific-computing-group /	# YOUR working directory
projects/	# YOUR projects for the meeting
my-hobbies/	# YOUR repo for git lesson
readmeasurements/	# YOUR repo for python lesson
recordings/	# YOUR copy of the recordings
meetings/	# YOUR meetings directory; YOUR workspace

scientific-computing-group/	# Scientific computing group's GitHub repo
-----------------------------	--

.git/
data/
meetings/
presentations/
resources/
readme.md

```
$ mkdir scientific-computing-group
$ cd scientific-computing-group
$ git clone https://github.com/jlant-usgs/scientific-computing-group.git
$ git pull -u origin master
```

 git pull whenever you want for updates

Last Meeting

- Discussed Python scoping
 - **namespace** - a container for variable names
 - i. makes it possible to distinguish between variables that have the exact same name
 - ii. allows grouping of names around a particular functionality
 - **scope** - region of a program where a namespace can be accessed
 - Variables are looked up from **namespaces** according to the **LEGB Order**:
 - i. Local function scope
 - ii. Enclosing function scope
 - iii. Global scope
 - iv. Builtin scope

Review of notes and questions from last meeting

- <https://github.com/jlant-usgs/scientific-computing-group>

Today's Objectives

1. Discuss difference between a script and a module
 - a. module's attribute called: `__name__`

Python Modules

- **Module** - a file consisting of Python code that defines functions, classes, and variables.
 - allows for code to be organized and for related code to be grouped together making the code easier to understand and use.
 - can use **import** to use functionality from a module in a script or in another module
 - **import module**
 - e.g. - **import sys**
 - a module is a Python object and has attributes, e.g.:
 - `__name__`
 - a module can contain runnable code using the following:

```
if __name__ == "__main__":
```

```
    # runnable code here
```

Script vs. Module

avg-script.py

```
# This script computes the average

x = 2
y = 4
avg = (x + y) / 2
print("The average is: "), avg
```

avg-module.py

```
# This module contains an average function
# and can be run like a script

def average(x, y):
    """ Return the average """
    return (x + y) / 2

def main():
    print("The average is: "), average(2,
    4)

if __name__ == "__main__":
    main()
```


Modules and `__name__`

```
def main():  
    # Do something interesting  
  
if __name__ == '__main__':  
    # Do something appropriate here, like calling a  
    # main() function defined elsewhere in this module.  
    main()  
else:  
    # Do nothing. This module has been imported by another  
    # module that wants to make use of the functions,  
    # classes and other useful bits it has defined.
```

Next Meeting

- Continue to refactor the `read_measurements.py` file with functions.
- Refactor `read_measurements.py` with numpy arrays
- Plot data with matplotlib