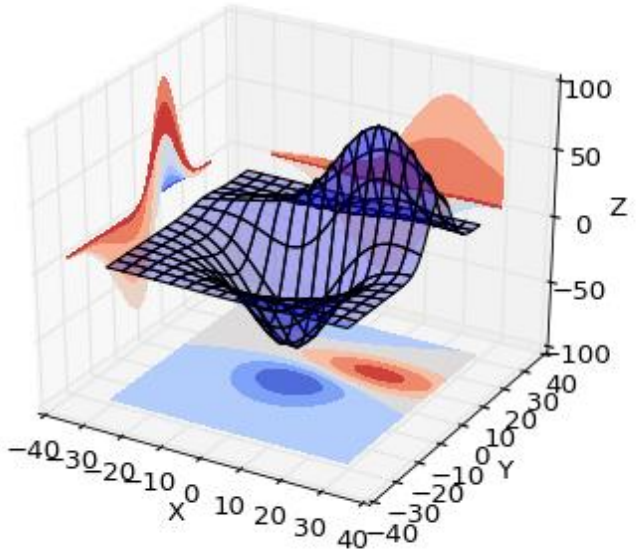
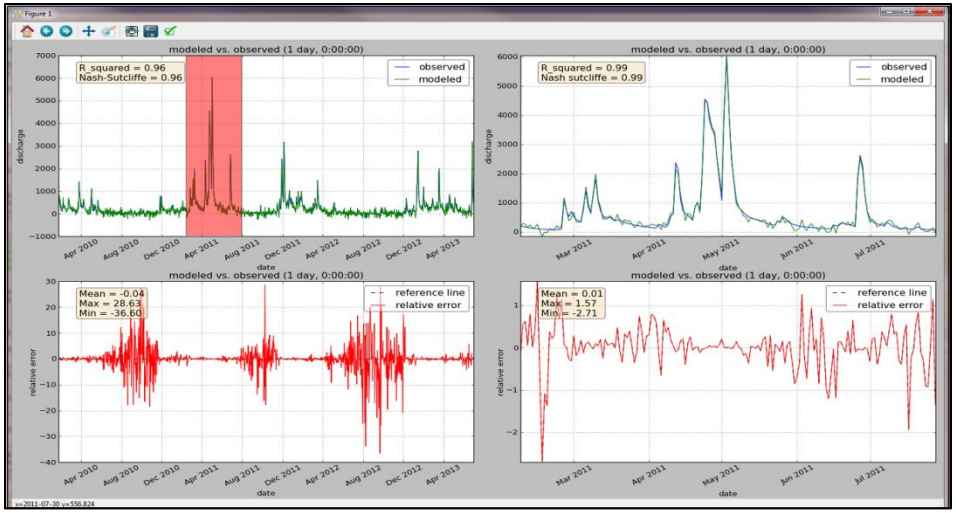
Introduction to Scientific Computing
Meeting 23
Programming with Python



python™

```
# 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

Last Meeting

- Learned about:
  - modules
  - import statement
  - sys module
  - sys.argv
- Learn how to get user input using **command line arguments** via the **sys** module; **sys.argv**

Last meeting – sys module

- The **sys** is a built in Python module/library that has great utility for talking to and working with a machines operating system.

Last meeting – modules

- **Modules** allow you to logically organize your Python code.
  - Grouping related code into a module makes the code easier to understand and use.
- A **module** is just a file consisting of Python code. A module can define functions, classes, and variables. A module can also include runnable code like scripts.
- **Modules** are a great way to create an advanced library with complex code instead of using the interpreter or simple scripts that are used to explore a problem and experiment with a problem.

- **Much more on modules and organizing code in the near future after learning about functions when we build more complex programs.

Last meeting – import statement

- **import** statement gives you access to a module
  - Use import statements at the top of a Python file
  - When the Python interpreter encounters an import statement, searches the search path (via sys.path) for the module

import <module_name>

>>> import sys

some_file.py

```
import sys
```

Last meeting – sys.argv function

- For user input, we are going to use a **function** contained in the **sys** module called **argv** (we will be learning about functions next after file input and output).
- **sys.argv** is a list, which contains the **command-line arguments passed to the script**.
- **https://docs.python.org/2/library/sys.html**

Recall from Unix: bash scripts

$ HELLO=“Hello World”
$ echo $HELLO

- Sample script that reads command line arguments and user input
  - readinput.sh

```
# read command line arguments from command line
ARG0=$0
ARG1=$1
ARG2=$2

echo "Argument 0 is: " $ARG0
echo "Argument 1 is: " $ARG1
echo "Argument 2 is: " $ARG2

# read user input from command line
echo -n "Enter your name and press [ENTER]: "
read name
echo $name
```

Demo – sys.argv function

- Let’s create a new file called sys-argv.py to learn more about sys.argv.

Today's Objectives

- Use sys.argv a little more
- Learn how to read and write files; **File Input and output (File I/O)**.

Practice Objectives

- Read and work with the measurements text data file.

Next meeting

- Learn how to **write files.**
- Learn how to write **functions.**