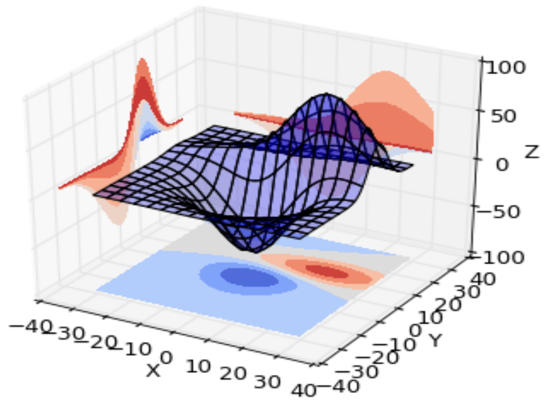
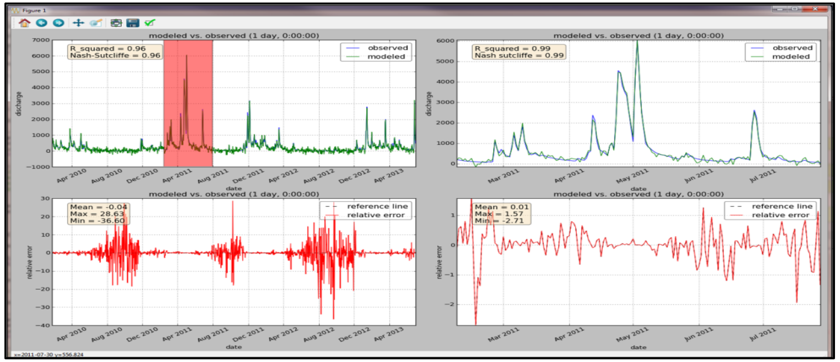
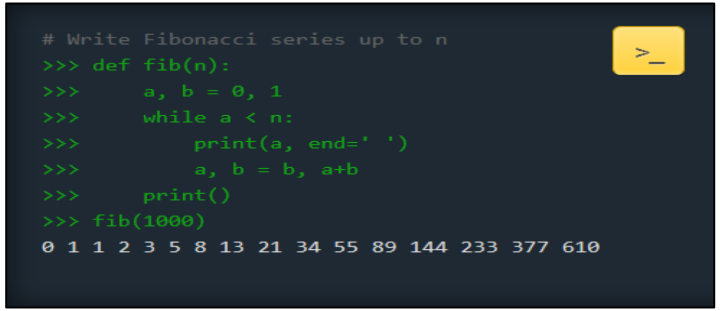
Scientific Computing Group
Programming with Python:
Read Measurements Project, Functions, Scoping







```
# 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
```

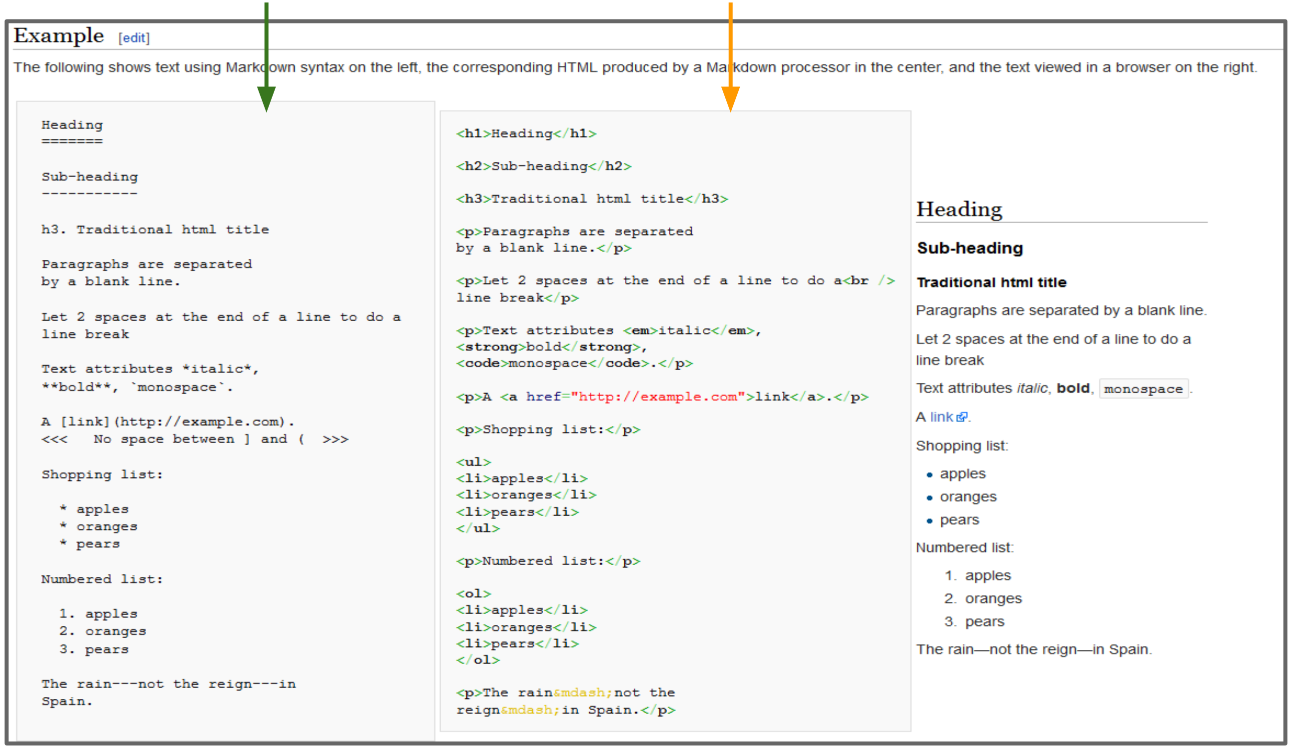
Jeremiah Lant, Hydrologist
USGS Kentucky Water Science Center
jlant@usgs.gov

Last Meeting

- Discussed collaborative meeting notes using TitanPad
- Discussed markdown
- Learned how to **clone** the scientific-computing-group repository on GitHub and **pull** latest changes.
- Refactored the read_measurements.py file in the “read measurements project” with functions.

Last Meeting: Markdown

Purpose: Can write **Markdown** instead of raw **HTML**.



Example [edit]

The following shows text using Markdown syntax on the left, the corresponding HTML produced by a Markdown processor in the center, and the text viewed in a browser on the right.

```
Heading
=======

Sub-heading
-----------

h3. Traditional html title

Paragraphs are separated
by a blank line.

Let 2 spaces at the end of a line to do a
line break

Text attributes *italic*,
**bold**, `monospace`.

A [link](http://example.com).
<<<   No space between ] and (  >>>

Shopping list:

  * apples
  * oranges
  * pears

Numbered list:

  1. apples
  2. oranges
  3. pears

The rain---not the reign---in
Spain.
```

```
<h1>Heading</h1>

<h2>Sub-heading</h2>

<h3>Traditional html title</h3>

<p>Paragraphs are separated
by a blank line.</p>

<p>Let 2 spaces at the end of a line to do a<br />
line break</p>

<p>Text attributes <em>italic</em>,
<strong>bold</strong>,
<code>monospace</code>.</p>

<p>A <a href="http://example.com">link</a>.</p>

<p>Shopping list:</p>

<ul>
<li>apples</li>
<li>oranges</li>
<li>pears</li>
</ul>

<p>Numbered list:</p>

<ol>
<li>apples</li>
<li>oranges</li>
<li>pears</li>
</ol>

<p>The rain&mdash;not the
reign&mdash;in Spain.</p>
```

Heading

Sub-heading

Traditional html title

Paragraphs are separated by a blank line.

Let 2 spaces at the end of a line to do a
line break

Text attributes *italic*, **bold**, `monospace`.

A link.

Shopping list:

- apples
- oranges
- pears

Numbered list:

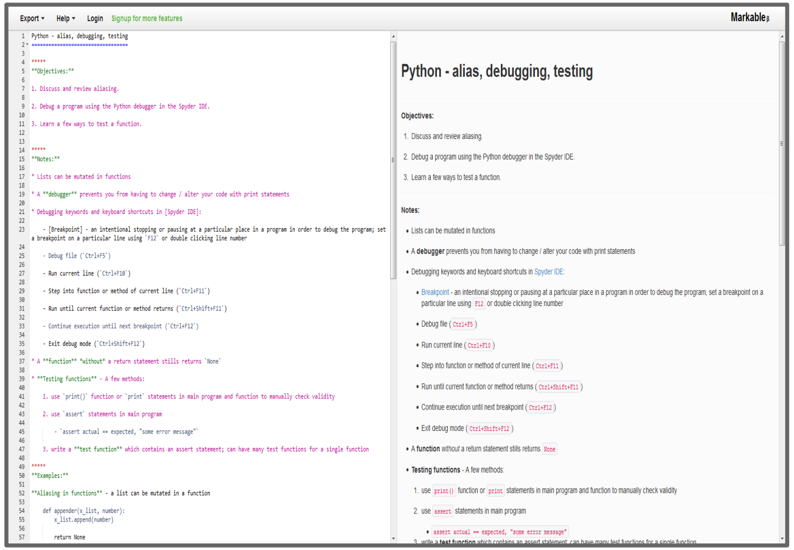
1. apples
2. oranges
3. pears

The rain—not the reign—in Spain.

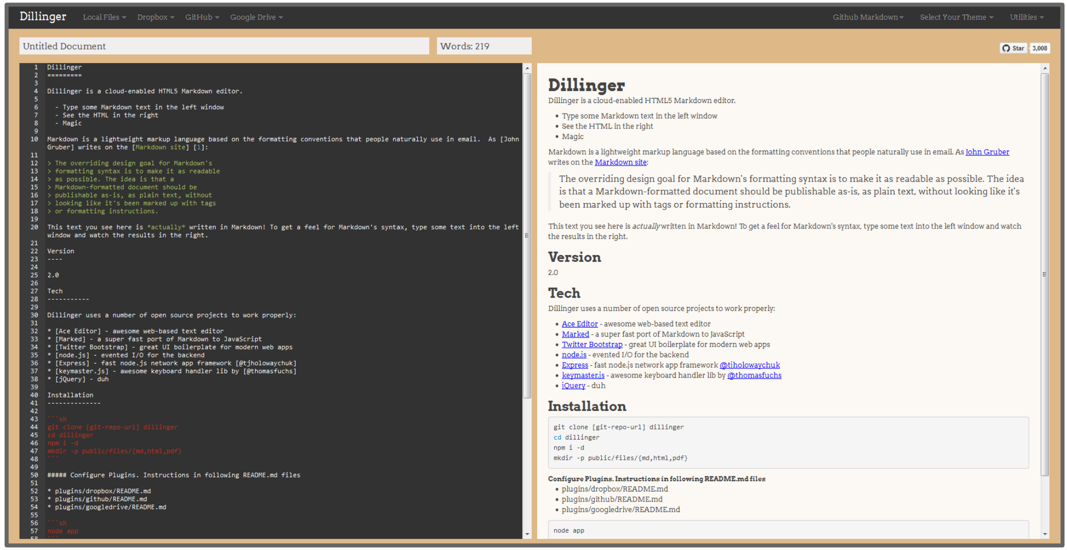
http://en.wikipedia.org/wiki/Markdown

Markdown - online editors

Purpose: Write **Markdown** and immediately see the result. Allow exporting to **HTML**.



```
Python - alias, debugging, testing
==================================

*****
**Objectives:**

1. Discuss and review aliasing.

2. Debug a program using the Python debugger in the Spyder IDE.

3. Learn a few ways to test a function.


*****
**Notes:**

* Lists can be mutated in functions

* A **debugger** prevents you from having to change / alter your code with print statements

* Debugging keywords and keyboard shortcuts in [Spyder IDE]:

    - [Breakpoint] - an intentional stopping or pausing at a particular place in a program in order to debug the program; set a breakpoint on a particular line using `F12` or double clicking line number

    - Debug file (`Ctrl+F5`)

    - Run current line (`Ctrl+F10`)

    - Step into function or method of current line (`Ctrl+F11`)

    - Run until current function or method returns (`Ctrl+Shift+F11`)

    - Continue execution until next breakpoint (`Ctrl+F12`)

    - Exit debug mode (`Ctrl+Shift+F12`)

* A **function** *without* a return statement stills returns `None`

* **Testing functions** - A few methods:

    1. use `print()` function or `print` statements in main program and function to manually check validity

    2. use `assert` statements in main program

        - `assert actual == expected, "some error message"`

    3. write a **test function** which contains an assert statement; can have many test functions for a single function

*****
**Examples:**

**Aliasing in functions** - a list can be mutated in a function

    def appender(x_list, number):
        x_list.append(number)

        return None
```



~~~
Dillinger
=========

Dillinger is a cloud-enabled HTML5 Markdown editor.

  - Type some Markdown text in the left window
  - See the HTML in the right
  - Magic

Markdown is a lightweight markup language based on the formatting conventions that people naturally use in email.  As [John Gruber] writes on the [Markdown site] [1]:

> The overriding design goal for Markdown's
> formatting syntax is to make it as readable
> as possible. The idea is that a
> Markdown-formatted document should be
> publishable as-is, as plain text, without
> looking like it's been marked up with tags
> or formatting instructions.

This text you see here is *actually* written in Markdown! To get a feel for Markdown's syntax, type some text into the left window and watch the results in the right.

Version
----

2.0

Tech
-----------

Dillinger uses a number of open source projects to work properly:

* [Ace Editor] - awesome web-based text editor
* [Marked] - a super fast port of Markdown to JavaScript
* [Twitter Bootstrap] - great UI boilerplate for modern web apps
* [node.js] - evented I/O for the backend
* [Express] - fast node.js network app framework [@tjholowaychuk]
* [keymaster.js] - awesome keyboard handler lib by [@thomasfuchs]
* [jQuery] - duh

Installation
--------------

```sh
git clone [git-repo-url] dillinger
cd dillinger
npm i -d
mkdir -p public/files/{md,html,pdf}
```

##### Configure Plugins. Instructions in following README.md files

* plugins/dropbox/README.md
* plugins/github/README.md
* plugins/googledrive/README.md

```sh
node app
~~~

http://markable.in/editor/

http://dillinger.io/

Another online editor - https://stackedit.io/editor#

Clone the GitHub repo

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

```
/scientific-computing-group /                # parent directory; call it what you like
    projects/
        my-hobbies/                                   # repo for git lesson
        readmeasurements/                    # repo for python lesson
     recordings/
    scientific-computing-group/              # this is the scientific computing group’s GitHub repo
        .git/
        data/
        meetings/
        presentations/
        resources/
        readme.md
```

Review of notes and questions from last meeting

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

Today’s Objectives

1. Discuss Python scoping
2. Continue to refactor the read_measurements.py file in the “read measurements project” with functions.

Python Scoping

- When you create a variable name, Python will create, change, or look up the variable name in a **namespace**; variable names exist in a namespace
- **Scope** is the region of a program where a namespace can be accessed
  - There can be a number of scopes working at any given time
  - e.g. scope of the function you are in (**local**), scope of inner functions when there is nested functions (**enclosing**), scope of a whole module (**global**), scope of Python builtins (**builtin**)
  - There is a hierarchy ordered rule in Python called **LEGB** (nested scoping) - used to prevent one function from accessing variables in a different function.
    - **Local Function Scope**
    - **Enclosing Function Scope**
    - **Global Scope**
    - **Builtin Scope**

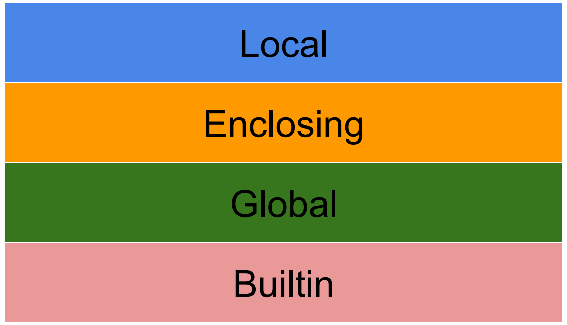
Python LEGB Order

L : local in the current **def** statement; names assigned within a function **def** statement, but not declared *global* in that function

E : enclosed function locals (**def** statement in another **def** statement); names in the local scope of any and all enclosing functions; form inner to outer

G : global in module; names assigned at the top-level of a module file or declared global in a **def** statement

B : built-in function in Python. Preassigned names; *print, range, open*



Next Meeting

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