

A Python sweeps in the GRASS

Alessandro Frigeri

Geologia Strutturale e Geofisica
Dipartimento di Scienze della Terra
Università degli Studi di Perugia
Perugia, ITALY

Foss4g2006 Conference – Lausanne, 12-15th September 2006

Introduction

We have news that in 2006 a Python started to sweep into the Grass.
Let's verify:



Introduction

We have news that in 2006 a Python started to sweep into the Grass.
Let's verify:

- if it effectively did it



Introduction

We have news that in 2006 a Python started to sweep into the Grass.
Let's verify:

- if it effectively did it
- how it did



Introduction

We have news that in 2006 a Python started to sweep into the Grass.
Let's verify:

- if it effectively did it
- how it did
- what it did



Introduction

We have news that in 2006 a Python started to sweep into the Grass.
Let's verify:

- if it effectively did it
- how it did
- what it did
- and what it is going to do!



What is Python?

Quoting from <http://www.python.org>:

Python is an interpreted, interactive, object-oriented programming language. People use to compare it to Tcl, Perl, Scheme or Java.

Python was created in the 1990s by Guido van Rossum as a successor to a language called ABC, a language for teaching and prototyping



Python design guidelines

Guido Von Rossum described python as the language that can be used to bridge the gap between the shell and C

so problems that are at the same time:



Python design guidelines

Guido Von Rossum described python as the language that can be used to bridge the gap between the shell and C

so problems that are at the same time:

- 1 Too complex to be solved by a shell script



Python design guidelines

Guido Von Rossum described python as the language that can be used to bridge the gap between the shell and C

so problems that are at the same time:

- 1 Too complex to be solved by a shell script
- 2 Not worth of being a new C program



Python design guidelines

Guido Von Rossum described python as the language that can be used to bridge the gap between the shell and C

so problems that are at the same time:

- 1 Too complex to be solved by a shell script
- 2 Not worth of being a new C program

can be approached by using Python



What is GRASS GIS?

- It is the Geographic Resources Analysis Support System



What is GRASS GIS?

- It is the Geographic Resources Analysis Support System
- It is a software project developed from 1980s



What is GRASS GIS?

- It is the Geographic Resources Analysis Support System
- It is a software project developed from 1980s
- from 1997 it is Free Software – GPL license.



What is GRASS GIS?

- It is the Geographic Resources Analysis Support System
- It is a software project developed from 1980s
- from 1997 it is Free Software – GPL license.
- **It is written in C and has shell script capabilities**



What is GRASS GIS?

- It is the Geographic Resources Analysis Support System
- It is a software project developed from 1980s
- from 1997 it is Free Software – GPL license.
- **It is written in C and has shell script capabilities**

Python design philosophy fits extremely well into the GRASS-GIS environment.



Interfacing Python to GRASS

There have been basically two approaches to interface GRASS and Python:



Interfacing Python to GRASS

There have been basically two approaches to interface GRASS and Python:

- By simply accessing directly grass through e.g. the os module



Interfacing Python to GRASS

There have been basically two approaches to interface GRASS and Python:

- By simply accessing directly grass through e.g. the os module
- By using the Simplified Wrapper and Interface Generator (SWIG), an interface compiler that connects programs written in C and C++ with scripting languages such as Perl, Python, Ruby, and Tcl

In short time we had several chances to play with Python and GRASS-GIS.



What has been done: a review

Python-SWIG wrapped GRASS C API

It allows to call and use GRASS library functions from python programs. So, as we programmed a module in C, we can use the same functions in python:

```
import python_grass6
mapset = python_grass6.G_mapset()
print mapset
```

Contributed by Sajith VK, March 2006. Now available in the CVS in `swig/python` directory.



What has been done: a review

wx-Windows python interface to grass modules

`r.shaded.relief` —interface—description | `grassgui.py`

File

Input elevation map:

[optional] Output shaded relief map name:

[optional] Altitude of the sun in degrees above the horizon:

[optional] Azimuth of the sun in degrees to the east of north:

[optional] Factor for exaggerating relief:

[optional] Scale factor for converting horizontal units to elevation units:

[optional] Set scaling factor (applies to lat./long. locations only):

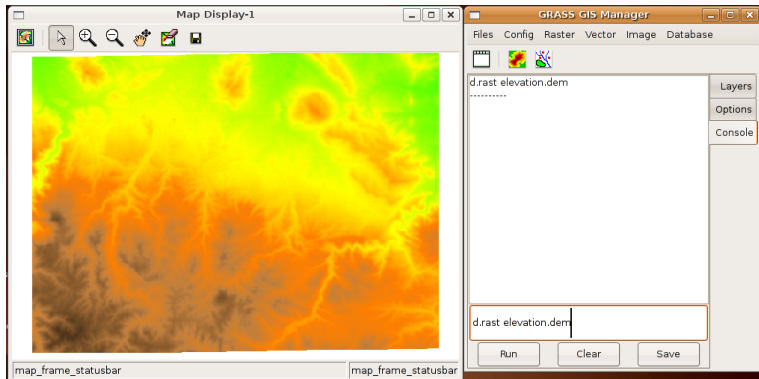
Cancel Run

Enter parameters for r.shaded.relief



What has been done: a review

wx-Windows python GIS manager

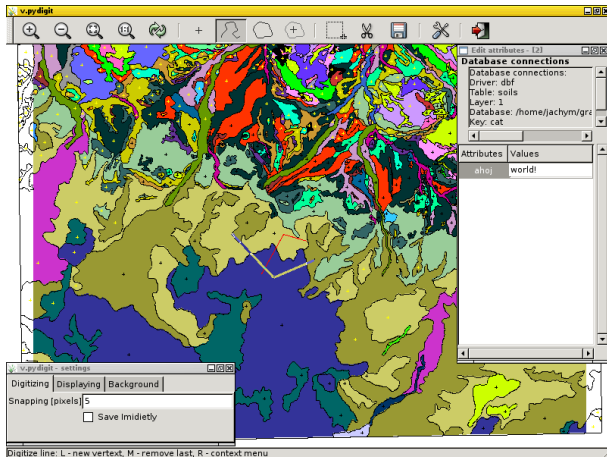


Developed by Michael Barton



What has been done: a review

GTK Interface – v.pydigit



by Jachym Cepicky, 2005

One step beyond

Although various python applications are up-and-running inside GRASS, we can make a step beyond to access the full power of python through a more generic interface to GRASS:



One step beyond

Although various python applications are up-and-running inside GRASS, we can make a step beyond to access the full power of python through a more generic interface to GRASS:

- using `python_grass6` swigged api through a more high level



One step beyond

Although various python applications are up-and-running inside GRASS, we can make a step beyond to access the full power of python through a more generic interface to GRASS:

- using python_grass6 swigged api through a more high level
- creation of sessions, maps, device classes



One step beyond

Although various python applications are up-and-running inside GRASS, we can make a step beyond to access the full power of python through a more generic interface to GRASS:

- using python_grass6 swigged api through a more high level
- creation of sessions, maps, device classes
- creation of a python 'package' (ordering things in namespaces, using the dotted syntax)



One step beyond

Although various python applications are up-and-running inside GRASS, we can make a step beyond to access the full power of python through a more generic interface to GRASS:

- using python_grass6 swigged api through a more high level
- creation of sessions, maps, device classes
- creation of a python 'package' (ordering things in namespaces, using the dotted syntax)
- use of distutils



One step beyond

Although various python applications are up-and-running inside GRASS, we can make a step beyond to access the full power of python through a more generic interface to GRASS:

- using python_grass6 swigged api through a more high level
- creation of sessions, maps, device classes
- creation of a python 'package' (ordering things in namespaces, using the dotted syntax)
- use of distutils
- creation of documentation
 - A user manual
 - pydoc functions and class documentation



One step beyond

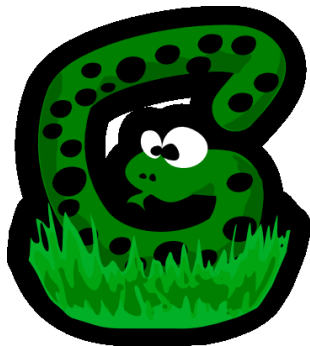
Although various python applications are up-and-running inside GRASS, we can make a step beyond to access the full power of python through a more generic interface to GRASS:

- using python_grass6 swigged api through a more high level
- creation of sessions, maps, device classes
- creation of a python 'package' (ordering things in namespaces, using the dotted syntax)
- use of distutils
- creation of documentation
 - A user manual
 - pydoc functions and class documentation

Put it all together and....



Introducing you....



The pyGrass package
(ver. 0.1beta)



pyGrass-0.1 requirements

Software	Version	Notes
Python	≥ 2.3	–
python_grass6	–	Comes with GRASS6 sources
numpy	–	–



pyGrass Features

The main features of pyGrass are:



pyGrass Features

The main features of pyGrass are:

- a real user manual!



pyGrass Features

The main features of pyGrass are:

- a real user manual!
- Interactive or scripting usage



pyGrass Features

The main features of pyGrass are:

- a real user manual!
- Interactive or scripting usage
- No need to be inside an interactive GRASS session



pyGrass Features

The main features of pyGrass are:

- a real user manual!
- Interactive or scripting usage
- No need to be inside an interactive GRASS session
- Multi-session support



pyGrass Features

The main features of pyGrass are:

- a real user manual!
- Interactive or scripting usage
- No need to be inside an interactive GRASS session
- Multi-session support
- Raster data is mapped into Numpy



Documentation!

Documentation is important as the package itself

- User manual, written in reST
`http://docutils.sourceforge.net/rst.html`, output in:
 - HTML
 - PDF (\LaTeX to PDF)
 - XML
- pydoc documentation: just `pydoc pyGrass` and enjoy!



pyGrass-0.1 namespaces

The main namespaces are:

```
pyGrass.session  
pyGrass.maps  
pyGrass.utils  
pyGrass.gui.qt (*)  
pyGrass.gui.tk (*)  
pyGrass.gui.wx (*)  
pyGrass.gui.gtk (*)  
pyGrass.gui.xwin  
pyGrass.web (*)
```

(*) empty for now



(multi-)Session management

We can use pyGrass to manage several grass sessions in the same script, let's see the code:

```
from pyGrass.session import Session
from pyGrass.gui.xwin import device

db = '/home/alf/grassdb '
user = 'PERMANENT'

spearfish = Session(db, 'spearfish60 ', user)
fire = Session(db, 'firedata ', user)
```



(multi-)Session management

We can use pyGrass to manage several grass sessions in the same script, let's see the code:

```
from pyGrass.session import Session
from pyGrass.gui.xwin import device
```

```
db = '/home/alf/grassdb '
user = 'PERMANENT'
```

```
spearfish = Session(db, 'spearfish60 ', user)
fire = Session(db, 'firedata ', user)
```

- no need to be inside a running GRASS session



(multi-)Session management

We can use pyGrass to manage several grass sessions in the same script, let's see the code:

```
from pyGrass.session import Session
from pyGrass.gui.xwin import device
```

```
db = '/home/alf/grassdb '
user = 'PERMANENT'
```

```
spearfish = Session(db, 'spearfish60 ', user)
fire = Session(db, 'firedata ', user)
```

- no need to be inside a running GRASS session
- we can create as much session objects as we need

refer to `example-multisession.py` file with pyGrass-0.1beta



import numpy, import pylab... import world!

Raster data is read by the swigged GRASS library and put into a numpy array. Let's visualize it with pylab!

```
# Let 's create mydem object
mydem = Rmap('elevation.dem', spearfish)

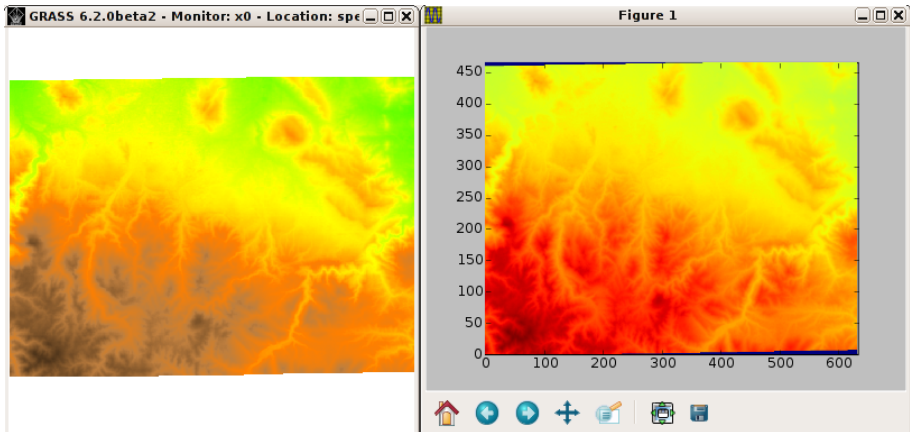
# get data of the map, the m object is a numpy array
m = mydem.getData()

# let 's see the image in matplotlib
import pylab
pylab.imshow(m)
pylab.show()
```

If all worked, we should get the most famous county in the world



Spearfish dem in matplotlib!



Conclusions



Conclusions

A Python really sweeps in the Grass!



Conclusions

A Python really sweeps in the Grass!

Thank you!
afrigeri at unipg.it

