Scripting in Madagascar: using SConstruct

Prof. Daniel Leal Macedo

Faculdade de Geofísica - UFPa

31 de julho a 3 de agosto de 2018

What is SCons?

- Build system (Software Construction)
- Written in Python
- Configuration (SConstruct files) are Python scripts
- Built-in support for different languages
- Dependency analysis
- Parallel builds
- Cross-platform

What is SCons?

- Build system (Software Construction)
- Written in Python
- Configuration (SConstruct files) are Python scripts
- Built-in support for different languages
- Dependency analysis
- Parallel builds
- Cross-platform

What is Python?

- Dynamic programming language
- Clear, readable syntax
- Full modularity
- Integrates with other languages
- Free and open-source

Python in 6 easy steps

1. Variables and strings
2. Lists and dictionaries
3. For loop
4. If/else, indentation
5. % operator
6. Functions and modules

1. Variables and strings

```
bash$ python
```

1. Variables and strings

```
bash$ python
Python 2.7.3 (default, Aug  1 2012, 05:14:39)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

1. Variables and strings

```
bash$ python
Python 2.7.3 (default, Aug  1 2012, 05:14:39)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> a='Beijing, China'
>>>
```

1. Variables and strings

```
bash$ python
Python 2.7.3 (default, Aug  1 2012, 05:14:39)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> a='Beijing, China'
>>> a[0]
```

1. Variables and strings

```
bash$ python
Python 2.7.3 (default, Aug  1 2012, 05:14:39)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> a='Beijing, China'
>>> a[0]
'B'
>>>
```

1. Variables and strings

```
bash$ python
Python 2.7.3 (default, Aug  1 2012, 05:14:39)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> a='Beijing, China'
>>> a[0]
'B'
>>> a[9:]
```

1. Variables and strings

```
bash$ python
Python 2.7.3 (default, Aug  1 2012, 05:14:39)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> a='Beijing, China'
>>> a[0]
'B'
>>> a[9:]
'China'
>>>
```

1. Variables and strings

```
bash$ python
Python 2.7.3 (default, Aug  1 2012, 05:14:39)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> a='Beijing, China'
>>> a[0]
'B'
>>> a[9:]
'China'
>>> b = a[:7] + " is awesome"
>>> print b
```

1. Variables and strings

```
bash$ python
Python 2.7.3 (default, Aug  1 2012, 05:14:39)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> a=’Beijing, China’
>>> a[0]
’B’
>>> a[9:]
’China’
>>> b = a[:7] + " is awesome"
>>> print b
Beijing is awesome
>>>
```

1. Variables and strings

```
bash$ python
Python 2.7.3 (default, Aug  1 2012, 05:14:39)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> a=’Beijing, China’
>>> a[0]
’B’
>>> a[9:]
’China’
>>> b = a[:7] + " is awesome"
>>> print b
Beijing is awesome
>>> a+5
```

1. Variables and strings

```
bash$ python
Python 2.7.3 (default, Aug  1 2012, 05:14:39)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> a='Beijing, China'
>>> a[0]
'B'
>>> a[9:]
'China'
>>> b = a[:7] + " is awesome"
>>> print b
Beijing is awesome
>>> a+5
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: cannot concatenate 'str' and 'int' objects
>>>
```

1. Variables and strings

```
bash$ python
Python 2.7.3 (default, Aug  1 2012, 05:14:39)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> a=’Beijing, China’
>>> a[0]
’B’
>>> a[9:]
’China’
>>> b = a[:7] + " is awesome"
>>> print b
Beijing is awesome
>>> a+5
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: cannot concatenate ’str’ and ’int’ objects
>>> a+str(5)
```

1. Variables and strings

```
bash$ python
Python 2.7.3 (default, Aug  1 2012, 05:14:39)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> a='Beijing, China'
>>> a[0]
'B'
>>> a[9:]
'China'
>>> b = a[:7] + " is awesome"
>>> print b
Beijing is awesome
>>> a+5
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: cannot concatenate 'str' and 'int' objects
>>> a+str(5)
'Beijing, China5'
```

2. List and Dictionaries

```
>>> b=['Beijing', 'China']
>>>
```

2. List and Dictionaries

```
>>> b=['Beijing', 'China']
>>> b[0]
```

2. List and Dictionaries

```
>>> b=['Beijing', 'China']
>>> b[0]
'Beijing'
>>>
```

2. List and Dictionaries

```
>>> b=['Beijing', 'China']
>>> b[0]
'Beijing'
>>> len(b)
```

2. List and Dictionaries

```
>>> b=['Beijing', 'China']
>>> b[0]
'Beijing'
>>> len(b)
2
>>>
```

2. List and Dictionaries

```
>>> b=['Beijing', 'China']
>>> b[0]
'Beijing'
>>> len(b)
2
>>> b.append(5)
>>> b
```

2. List and Dictionaries

```
>>> b=['Beijing', 'China']
>>> b[0]
'Beijing'
>>> len(b)
2
>>> b.append(5)
>>> b
['Beijing', 'China', 5]
>>>
```

2. List and Dictionaries

```
>>> b=['Beijing', 'China']
>>> b[0]
'Beijing'
>>> len(b)
2
>>> b.append(5)
>>> b
['Beijing', 'China', 5]
>>> c=('Beijing', 'China')
>>> c.append(5)
```

2. List and Dictionaries

```
>>> b=['Beijing', 'China']
>>> b[0]
'Beijing'
>>> len(b)
2
>>> b.append(5)
>>> b
['Beijing', 'China', 5]
>>> c=('Beijing', 'China')
>>> c.append(5)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'tuple' object has no attribute 'append'
```

2. List and Dictionaries

```
>>> tel = {'jack': 4098, 'sape': 4139}
>>>
```

2. List and Dictionaries

```
>>> tel = {'jack': 4098, 'sape': 4139}
>>> tel['guido'] = 4127
>>> tel
```

2. List and Dictionaries

```
>>> tel = {'jack': 4098, 'sape': 4139}
>>> tel['guido'] = 4127
>>> tel
{'sape': 4139, 'guido': 4127, 'jack': 4098}
```

2. List and Dictionaries

```
>>> tel = {'jack': 4098, 'sape': 4139}
>>> tel['guido'] = 4127
>>> tel
{'sape': 4139, 'guido': 4127, 'jack': 4098}
>>> tel['jack']
```

2. List and Dictionaries

```
>>> tel = {'jack': 4098, 'sape': 4139}
>>> tel['guido'] = 4127
>>> tel
{'sape': 4139, 'guido': 4127, 'jack': 4098}
>>> tel['jack']
4098
```

2. List and Dictionaries

```
>>> tel = {'jack': 4098, 'sape': 4139}
>>> tel['guido'] = 4127
>>> tel
{'sape': 4139, 'guido': 4127, 'jack': 4098}
>>> tel['jack']
4098
>>> del tel['sape']
>>> tel['irv'] = 4127
>>> tel
```

2. List and Dictionaries

```
>>> tel = {'jack': 4098, 'sape': 4139}
>>> tel['guido'] = 4127
>>> tel
{'sape': 4139, 'guido': 4127, 'jack': 4098}
>>> tel['jack']
4098
>>> del tel['sape']
>>> tel['irv'] = 4127
>>> tel
{'guido': 4127, 'irv': 4127, 'jack': 4098}
```

2. List and Dictionaries

```
>>> tel = {'jack': 4098, 'sape': 4139}
>>> tel['guido'] = 4127
>>> tel
{'sape': 4139, 'guido': 4127, 'jack': 4098}
>>> tel['jack']
4098
>>> del tel['sape']
>>> tel['irv'] = 4127
>>> tel
{'guido': 4127, 'irv': 4127, 'jack': 4098}
>>> tel.keys()
```

2. List and Dictionaries

```
>>> tel = {'jack': 4098, 'sape': 4139}
>>> tel['guido'] = 4127
>>> tel
{'sape': 4139, 'guido': 4127, 'jack': 4098}
>>> tel['jack']
4098
>>> del tel['sape']
>>> tel['irv'] = 4127
>>> tel
{'guido': 4127, 'irv': 4127, 'jack': 4098}
>>> tel.keys()
['guido', 'irv', 'jack']
```

2. List and Dictionaries

```
>>> tel = {'jack': 4098, 'sape': 4139}
>>> tel['guido'] = 4127
>>> tel
{'sape': 4139, 'guido': 4127, 'jack': 4098}
>>> tel['jack']
4098
>>> del tel['sape']
>>> tel['irv'] = 4127
>>> tel
{'guido': 4127, 'irv': 4127, 'jack': 4098}
>>> tel.keys()
['guido', 'irv', 'jack']
>>> 'guido' in tel
```

2. List and Dictionaries

```
>>> tel = {'jack': 4098, 'sape': 4139}
>>> tel['guido'] = 4127
>>> tel
{'sape': 4139, 'guido': 4127, 'jack': 4098}
>>> tel['jack']
4098
>>> del tel['sape']
>>> tel['irv'] = 4127
>>> tel
{'guido': 4127, 'irv': 4127, 'jack': 4098}
>>> tel.keys()
['guido', 'irv', 'jack']
>>> 'guido' in tel
True
```

3. For loop

```
>>> c=('Beijing', 'China')
>>>
```

3. For loop

```
>>> c=('Beijing', 'China')
>>> for word in c:
...     print word, len(word)
...
```

3. For loop

```
>>> c=('Beijing', 'China')
>>> for word in c:
...     print word, len(word)
...
Beijing 7
China 5
>>>
```

3. For loop

```
>>> c=('Beijing', 'China')
>>> for word in c:
...     print word, len(word)
...
Beijing 7
China 5
>>> for k in range(2):
...     print k, a[k]
...
```

3. For loop

```
>>> c=('Beijing', 'China')
>>> for word in c:
...     print word, len(word)
...
Beijing 7
China 5
>>> for k in range(2):
...     print k, a[k]
...
0 Beijing
1 China
>>>
```

3. For loop

```
>>> c=('Beijing', 'China')
>>> for word in c:
...     print word, len(word)
...
Beijing 7
China 5
>>> for k in range(2):
...     print k, a[k]
...
0 Beijing
1 China
>>> c = {'city': 'Beijing', 'number': 5}
>>>
```

3. For loop

```
>>> c=('Beijing', 'China')
>>> for word in c:
...     print word, len(word)
...
Beijing 7
China 5
>>> for k in range(2):
...     print k, a[k]
...
0 Beijing
1 China
>>> c = {'city': 'Beijing', 'number': 5}
>>> for key in c.keys():
...     print key, c[key]
...
```

3. For loop

```
>>> c=('Beijing', 'China')
>>> for word in c:
...     print word, len(word)
...
Beijing 7
China 5
>>> for k in range(2):
...     print k, a[k]
...
0 Beijing
1 China
>>> c = {'city': 'Beijing', 'number': 5}
>>> for key in c.keys():
...     print key, c[key]
...
city Beijing
number 5
```

4. If/else, indentation

```
>>> for k in range(4):
...     if k < 2:
...             print k
...     else:
...             print 'no'
...
```

4. If/else, indentation

```
>>> for k in range(4):
...     if k < 2:
...             print k
...     else:
...             print 'no'
...
0
1
no
no
```

4. If/else, indentation

```
>>> for k in range(4):
...     if k < 2:
...             print k
...     else:
...             print 'no'
...
0
1
no
no
>>> try:
...     a = 'Beijing' + 5
... except:
...     print 'error'
...
```

4. If/else, indentation

```
>>> for k in range(4):
...     if k < 2:
...             print k
...     else:
...             print 'no'
...
0
1
no
no
>>> try:
...     a = 'Beijing' + 5
... except:
...     print 'error'
...
error
```

5. % operator

```
>>> d=9.2
```

5. % operator

```
>>> d=9.2
>>> print "My test grade was %s"%d
```

5. % operator

```
>>> d=9.2
>>> print "My test grade was %s"%d
My test grade was 9.2
```

5. % operator

```
>>> d=9.2
>>> print "My test grade was %s"%d
My test grade was 9.2
>>> print "My test grade was %f"%d
```

5. % operator

```
>>> d=9.2
>>> print "My test grade was %s"%d
My test grade was 9.2
>>> print "My test grade was %f"%d
My test grade was 9.200000
```

5. % operator

```
>>> d=9.2
>>> print "My test grade was %s"%d
My test grade was 9.2
>>> print "My test grade was %f"%d
My test grade was 9.200000
>>> print "My test grade was %d"%d
```

5. % operator

```
>>> d=9.2
>>> print "My test grade was %s"%d
My test grade was 9.2
>>> print "My test grade was %f"%d
My test grade was 9.200000
>>> print "My test grade was %d"%d
My test grade was 9
```

5. % operator

```
>>> d=9.2
>>> print "My test grade was %s"%d
My test grade was 9.2
>>> print "My test grade was %f"%d
My test grade was 9.200000
>>> print "My test grade was %d"%d
My test grade was 9
>>> print "I have lived in %s"%b[0]
```

5. % operator

```
>>> d=9.2
>>> print "My test grade was %s"%d
My test grade was 9.2
>>> print "My test grade was %f"%d
My test grade was 9.200000
>>> print "My test grade was %d"%d
My test grade was 9
>>> print "I have lived in %s"%b[0]
I have lived in Beijing
```

5. % operator

```
>>> d=9.2
>>> print "My test grade was %s"%d
My test grade was 9.2
>>> print "My test grade was %f"%d
My test grade was 9.200000
>>> print "My test grade was %d"%d
My test grade was 9
>>> print "I have lived in %s"%b[0]
I have lived in Beijing
>>> print "I have lived in %s, %s"%(b[0],b[1])
```

5. % operator

```
>>> d=9.2
>>> print "My test grade was %s"%d
My test grade was 9.2
>>> print "My test grade was %f"%d
My test grade was 9.200000
>>> print "My test grade was %d"%d
My test grade was 9
>>> print "I have lived in %s"%b[0]
I have lived in Beijing
>>> print "I have lived in %s, %s"%(b[0],b[1])
I have lived in Beijing, China
```

5. % operator

```
>>> d=9.2
>>> print "My test grade was %s"%d
My test grade was 9.2
>>> print "My test grade was %f"%d
My test grade was 9.200000
>>> print "My test grade was %d"%d
My test grade was 9
>>> print "I have lived in %s"%b[0]
I have lived in Beijing
>>> print "I have lived in %s, %s"%(b[0],b[1])
I have lived in Beijing, China
>>> print "Guido's phone number is %(guido)s."%tel
```

5. % operator

```
>>> d=9.2
>>> print "My test grade was %s"%d
My test grade was 9.2
>>> print "My test grade was %f"%d
My test grade was 9.200000
>>> print "My test grade was %d"%d
My test grade was 9
>>> print "I have lived in %s"%b[0]
I have lived in Beijing
>>> print "I have lived in %s, %s"%(b[0],b[1])
I have lived in Beijing, China
>>> print "Guido's phone number is %(guido)s."%tel
Guido's phone number is 4127.
```

5. % operator

```
>>> d=9.2
>>> print "My test grade was %s"%d
My test grade was 9.2
>>> print "My test grade was %f"%d
My test grade was 9.200000
>>> print "My test grade was %d"%d
My test grade was 9
>>> print "I have lived in %s"%b[0]
I have lived in Beijing
>>> print "I have lived in %s, %s"%(b[0],b[1])
I have lived in Beijing, China
>>> print "Guido's phone number is %(guido)s."%tel
Guido's phone number is 4127.
>>> print "Guido's phone number is %(guido)s. Jack's is %(jack)s."%tel
```

5. % operator

```
>>> d=9.2
>>> print "My test grade was %s"%d
My test grade was 9.2
>>> print "My test grade was %f"%d
My test grade was 9.200000
>>> print "My test grade was %d"%d
My test grade was 9
>>> print "I have lived in %s"%b[0]
I have lived in Beijing
>>> print "I have lived in %s, %s"%(b[0],b[1])
I have lived in Beijing, China
>>> print "Guido's phone number is %(guido)s."%tel
Guido's phone number is 4127.
>>> print "Guido's phone number is %(guido)s. Jack's is %(jack)s."%tel
Guido's phone number is 4127. Jack's is 4098.
```

6. Functions and modules

```
>>> def add_5(x):
...     'Add 5 to input'
...     return 5 + x
...
>>>
```

6. Functions and modules

```
>>> def add_5(x):
...     'Add 5 to input'
...     return 5 + x
...
>>> d = add_5(3)
>>>
```

6. Functions and modules

```
>>> def add_5(x):
...     'Add 5 to input'
...     return 5 + x
...
>>> d = add_5(3)
>>> d
```

6. Functions and modules

```
>>> def add_5(x):
...     'Add 5 to input'
...     return 5 + x
...
>>> d = add_5(3)
>>> d
8
>>>
```

6. Functions and modules

```
>>> def add_5(x):
...     'Add 5 to input'
...     return 5 + x
...
>>> d = add_5(3)
>>> d
8
>>> def add_y(x, y=5):
...     'Add x to y'
...     return x + y
...
>>>
```

6. Functions and modules

```
>>> def add_5(x):
...     'Add 5 to input'
...     return 5 + x
...
>>> d = add_5(3)
>>> d
8
>>> def add_y(x, y=5):
...     'Add x to y'
...     return x + y
...
>>> add_y(d)
```

6. Functions and modules

```
>>> def add_5(x):
...     'Add 5 to input'
...     return 5 + x
...
>>> d = add_5(3)
>>> d
8
>>> def add_y(x, y=5):
...     'Add x to y'
...     return x + y
...
>>> add_y(d)
13
>>>
```

6. Functions and modules

```
>>> def add_5(x):
...     'Add 5 to input'
...     return 5 + x
...
>>> d = add_5(3)
>>> d
8
>>> def add_y(x, y=5):
...     'Add x to y'
...     return x + y
...
>>> add_y(d)
13
>>> import math
>>>
```

6. Functions and modules

```
>>> def add_5(x):
...     'Add 5 to input'
...     return 5 + x
...
>>> d = add_5(3)
>>> d
8
>>> def add_y(x, y=5):
...     'Add x to y'
...     return x + y
...
>>> add_y(d)
13
>>> import math
>>> math.sqrt(add_y(d,8))
4.0
```

Madagascar processing: SConstruct. Exemplo 1

```
from rsf.proj import *

Fetch('wz.35.H','wz', usedatapath=0)

Flow('wind','wz.35.H',
     '''
     dd form=native | window n1=400 j1=2 |
     smooth rect1=3
     ''')
Plot('wind','pow pow1=2 | grey')

Flow('mute','wind','mutter v0=0.31 half=n')
Plot('mute','pow pow1=2 | grey')

Result('denmark','wind mute','SideBySideAniso')

End()
```

Madagascar processing: SConstruct. Exemplo 1

```
bash$ ls
```

```
SConstruct
```

```
bash$ scons
```

```
scons: Reading SConscript files ...
scons: done reading SConscript files.
scons: Building targets ...
retrieve(["wz.35.H"], [])
< wz.35.H /home/daniel/madagascar/bin/sfdd form=native |/home/daniel/mada
gascar/bin/sfwindow n1=400 j1=2 | /home/daniel/madagascar/bin/sfsmooth re
ct1=3 > wind.rsf
< wind.rsf /home/daniel/madagascar/bin/sfpow pow1=2 | /home/daniel/madaga
scar/bin/sfgrey > wind.vpl
< wind.rsf /home/daniel/madagascar/bin/sfmutter v0=0.31 half=n > mute.rsf
< mute.rsf /home/daniel/madagascar/bin/sfpow pow1=2 | /home/daniel/madaga
scar/bin/sfgrey > mute.vpl
/home/daniel/madagascar/bin/vppen yscale=2 vpstyle=n gridnum=2,1 wind.vpl
mute.vpl > Fig/denmark.vpl
scons: done building targets.
```

```
bash$ ls
```

```
Fig         mute.vpl    script.sh  wind.vpl
mute.rsf    SConstruct  wind.rsf   wz.35.H
```

Madagascar processing: SConstruct. Exemplo 1

```
from rsf.proj import *

Fetch('wz.35.H','wz', usedatapath=0)

Flow('wind','wz.35.H',
     '''
     dd form=native | window n1=400 j1=2 |
     smooth rect1=3
     ''')
Plot('wind','pow pow1=2 | grey')

Flow('mute','wind','mutter v0=0.31 half=n')
Plot('mute','pow pow1=2 | grey')

Result('denmark','wind mute','SideBySideAniso')

End()
```

Madagascar processing with `rsf.proj`

Information on `rsf.proj` functions:
http://www.ahay.org/wiki/SCons

- `Fetch(<file[s]>,<directory>,[options])`
- `Flow(<target[s]>,<source[s]>,<command>,[options])`
- `Plot(<target>,[<source[s]>],<command>,[options])`
- `Result(<target>,[<source[s]>],<command>,[options])`

Running SConstruct

- `scons`
- `scons view`
- `scons -c`
- `scons -n`
- `scons -n -Q > script.sh`
- `scons file0.rsf [file1.rsf ...]`
- `scons file0.view [file1.view ...]`
- `scons file.lock` X `scons file.flip`

Madagascar processing: SConstruct. Exemplo 2

```
from rsf.proj import *

# create a model
Flow('model',None,'math n1=301 d1=0.01 o1=0 n2=1001 d2=0.01 o2=0 output="1+2*x1+0.5*x2"')

# plot the model
Plot('model','model',
          '''
          grey color=j scalebar=y label1=Depth unit1=km label2=Position unit2=km
          barlabel=Velocity barunit=km/s barreverse=y title="Model and ray"
          ''')

Result('model','model',
          '''
          grey color=j scalebar=y label1=Depth unit1=km label2=Position unit2=km
          barlabel=Velocity barunit=km/s barreverse=y title=Model
          ''')

# do a ray-tracing
Flow('ray','model','rays2 yshot=5 nt=500 dt=0.001 a0=180 nr=1')

# plot the ray
Plot('ray',
          '''
          graph transp=y yreverse=y min1=0 max1=3 min2=0 max2=10
          wantaxis=n wanttitle=n scalebar=y
          plotcol=7 plotfat=3
          ''')

# overlay model and ray
Result('overlay','model ray','Overlay')

End()
```

Exercício 6

Transforme os commandos usados nos exercícios 4 e 5 em um SConstruct.